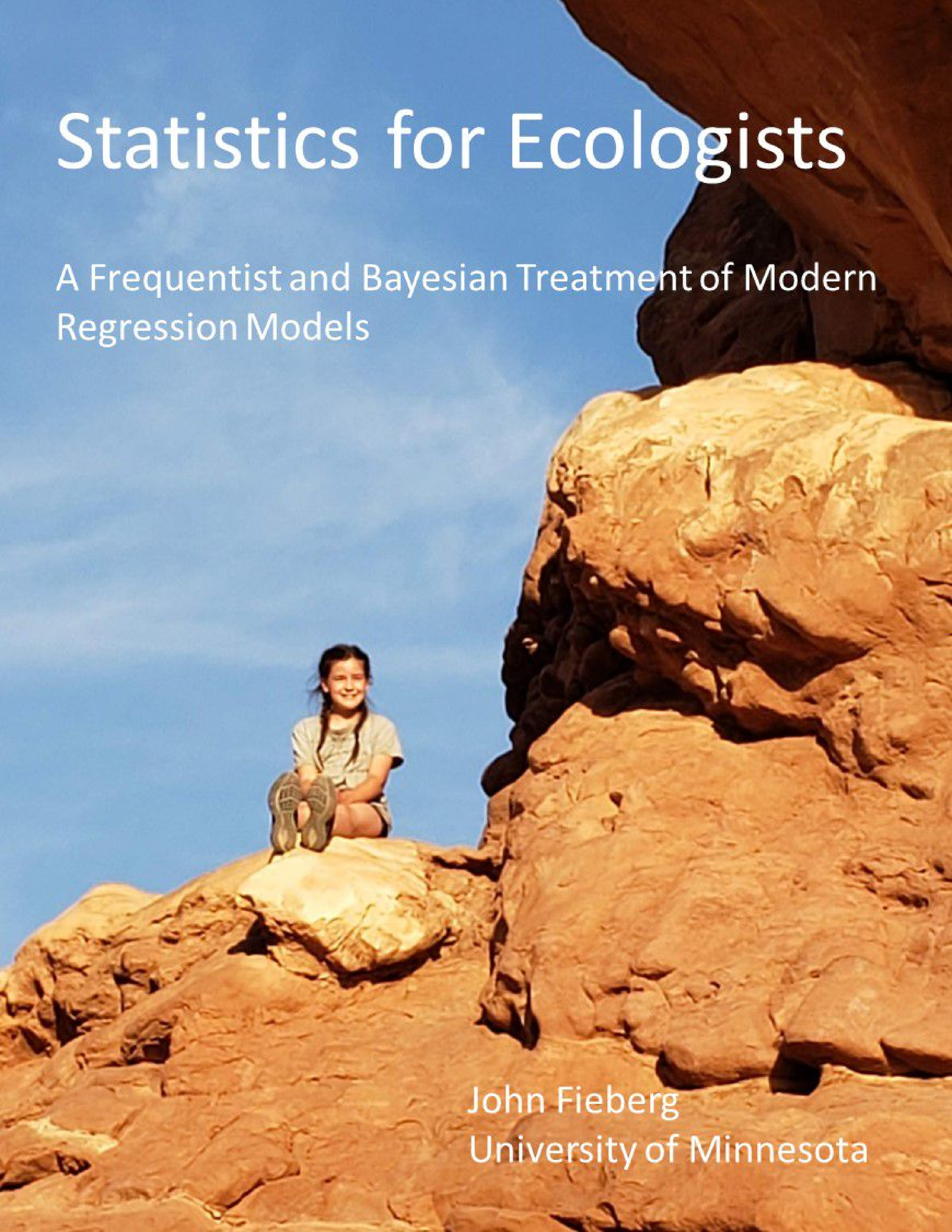


Statistics for Ecologists

A Frequentist and Bayesian Treatment of Modern Regression Models

John Fieberg
University of Minnesota

Suggested Citation:

Fieberg, John. (2024). Statistics for Ecologists: A Frequentist and Bayesian Treatment of Modern Regression Models. University of Minnesota Libraries Publishing. Retrieved from the University of Minnesota Digital Conservancy. https://doi.org/10.24926/9781959870029

Alernative version of the book

An html version of the book can be accessed at https://statistics4ecologists-v2.netlify.app/

Cover photograph:

Caroline Fieberg at Arches National Park, summer 2021.

ISBN: 978-1-959870-02-9

Edition 1

Contents

About the Author

I think it is always interesting to learn about a person's background as it provides a window into the experiences that shape them into who they are and how they think. So, just to let you know a little bit about myself. . .

I grew up in St. Louis, Missouri. My dad was a math teacher and my mom stayed at home with my brother and me until we were in middle or high school at which point she also taught high school math[1]. I had friends in college that always assumed I would become a math teacher too but that was far down on my list of career choices (after baseball player, gym teacher, physical therapist, etc.). Funny how life works out.

I received a BA in Mathematics from Westminster College in Fulton Missouri, where I played baseball for 2 years (as a pitcher, outfielder, and more frequently, a pinch runner). Westminster is a small school (around 700 students when I attended), and is most well known as the location for Winston Churchill's "Iron Curtain Speech". When I was a student, Mikhail Gorbachev also spoke on campus in front of a sculpture constructed from the Berlin Wall by Churchill's granddaughter. Small claim to fame - I was one of 7 students that led the processional for that event and even unknowingly photo-bombed Gorbachev and the Governor before selfies or photo-bombs were even a thing (that's me in the middle of Figure 1, with more hair than I have now).

After undergrad, I wanted to live somewhere warmer. I ended up going to graduate school in North Carolina. I received my MS in Biostatistics from the University of North Carolina-Chapel Hill and a PhD in Biomathematics from North Carolina State University. My PhD research was focused on comparing and evaluating methods used to perform population viability analyses. During this time, I fell in love with college basketball, especially after experiencing 4 NCAA final fours while living in North Carolina (go UNC!). My son is named after a player on the 2005 National Championship team and my youngest daughter is named Caroline Anne[2].

Prior to coming to the University of Minnesota, I worked for 2 1/2 years as a Biometrician for the Northwest Indian Fisheries Commission in Olympia, WA and 10 years as a Wildlife Biometrician for the Minnesota Department of Natural Resources. I had the opportunity to work with a lot of great people on a variety of research projects. These environments provided a fantastic opportunity to grow my quantitative and communication skills. I also learned much about salmon, the historic injustices faced by native people, and the important role that hunting and fishing (and hunters and fishers) play in conserving our natural resources.

My wife, Ann, is a biostatistician on East Bank of the University of Minnesota. If you take a class from me, you will probably hear stories about my family throughout the semester.

[1]This was actually their second life. My dad graduated from high school and became a Christian Brother. My mom left home at age 16 after World War II to go to the convent - it was going to be her vehicle for changing the world since she couldn't join the army. She remained a a nun for 20 years until she met my dad, they fell in love, got married, and the rest is history.

[2]My oldest daughter is named Zoe, which was my mom's name when she was a nun.

FIGURE 1 Left Panel: Church of St Mary Aldermanbury and Breakthrough sculpture. Photo by Rangermike at English Wikipedia. CC BY-SA 3.0. Right Panel: Photo of Gorbachev, Governor Ashcroft, and me during Churchill's visit to Westminster College in 1992 (copyright 1992 Gold/McCowen, courtesy of America's National Churchill Museum).

FIGURE 2 My family during a trip to the Grand Canyon and Zion National Park during the summer of 2021 just prior to starting a sabbatical to work on this book.

Preface

Ecological data pose many challenges to statistical inference. Most data come from observational studies rather than designed experiments; observational units are frequently sampled repeatedly over time, resulting in multiple, non-independent measurements; response data are often binary (e.g., presence-absence data) or non-negative integers (e.g.., counts), and therefore, the data do not fit the standard assumptions of linear regression (Normality, independence, and constant variance). This book will familiarize readers with modern statistical methods that address these complexities using both frequentist and Bayesian frameworks.

In the first part of the book, we focus on models appropriate for Normally distributed response variables. We begin by reviewing key concepts in frequentist statistics (sampling distributions, confidence intervals, p-values, etc.) in the context of linear regression (Section 1), but also introduce the bootstrap as a useful tool for inference when common assumptions may not be met (Section 2). We then move to multiple regression (Section 3), emphasizing the role of design matrices when formulating models that include categorical predictor variables, interaction terms, and non-linear predictor-response relationships (Section 4).

In part 2 of the book, we consider methods for choosing which variables to include in a model. We explore the impact of collinearity on parameter uncertainty (Section 6) and the important role that causal diagrams should play when choosing which variables to include in a model (Section 7). Lastly, we discuss different modeling objectives (describing, predicting, and inferring), and we explore the relative utility of various modeling strategies for meeting these objectives (Section 8).

In part 3 of the book, we explore other statistical distributions besides the Normal distribution (Section 9). We also review important concepts in mathematical statistics, which allow us to develop a better understanding of maximum likelihood (Section 10) and Bayesian inferential frameworks (Sections 11, 12, 13). Then, in part 4 of the book, we consider extensions for modeling non-Normal data using generalized linear models (Sections 14, 15, 16) and models that account for zero-inflated data (Section 17). Lastly, in part 5 of the book, we discuss methods for modeling correlated data, including mixed models (Section 18 and 19) and generalized estimating equations (Section 20).

Formatting and conventions

I frequently use red to highlight important concepts when first introduced (e.g., sampling distribution).

Before compiling the code in each chapter, I run a short script that sets the default theme for all ggplots:

```
library(ggthemes)
theme_set(theme_bw())
```

Pre-requisites

Ideally, you will have an understanding of key statistical concepts, including hypothesis tests, the Normal distribution, and linear regression. In addition, you should have a working knowledge of the R programming language (e.g., be able to read in data, work with common object types such as lists, matrices, data frames, install and load packages, access help functions, and construct simple plots).

Although I envision this book as appropriate for graduate students taking a second course in statistics, I recognize that not all introductory statistics courses are created equal; the experience and background knowledge of students entering my graduate-level statistics class is highly variable. Thus, although it would be nice if all students came in with a thorough understanding of key statistical concepts, an ability to code in R, and a solid foundation of linear models, it is rare that any student comes in with a solid foundation in all three areas. I generally try to include enough background and supporting material to overcome these deficiencies. Yet, some readers may find it beneficial to also seek out additional resources on one or more of these topics.

Learning objectives

The overarching goal of this book is to help train students to effectively analyze the data they collect as part of their research. Much of the book focuses on commonly used statistical models (e.g., linear and generalized linear models). Of course, it is impossible to cover all statistical methods that one might someday need. Further, a superficial understanding of topics only gets on one so far. Therefore, I try to emphasize key concepts and techniques that provide a solid foundation for further learning rather than a statistical "cookbook" with a set of recipes for different data situations. By working with several different classes of models, and both frequentist and Bayesian implementations, my hope is that you will develop strong coding skills and an enhanced ability to reason using mathematics and statistics. The repeated exposure to mathematical concepts and formulas should also make it easier for you to read and understand a wider range of literature. Hopefully, you will find quantitative papers to be much less intimidating!

By the end of the book, you should be able to:

- Construct models that address specific biological objectives.
- Understand the role of random variables and common statistical distributions in formulating modern regression models.
- Demonstrate *model literacy*, i.e., you should be able to describe a variety of statistical models and their assumptions using equations and text and match parameters in these equations to estimates in computer output.
- Identify key model assumptions, utilize diagnostic tools to assess the validity of these assumptions, and conduct sensitivity analyses to evaluate model robustness to assumption violations.
- Gain an appreciation for challenges associated with selecting among competing models and performing multi-model inference.
- Critique statistical methods used in the applied literature, identify strengths and weaknesses of different modeling approaches, and select appropriate analyses in your work.

To achieve the above learning objectives, you will be expected to develop new statistical modeling and computing skills (see *Skills Objectives*).

Skills objectives

By the end of this course, you should be able to:

- Construct predictor variables that allow fitting of models with categorical predictors and that allow for non-linear relationships between explanatory and response variables.
- Fit and evaluate a variety of regression models in both frequentist and Bayesian frameworks using open-source software (R and JAGS).
- Use simulation-based methods to test your understanding of key statistical concepts and models and to evaluate the plausibility of different model assumptions.
- Estimate quantities of interest along with their associated measures of uncertainty (e.g., confidence and prediction intervals) for a variety of commonly used regression models.
- Model non-Normal data using generalized linear models.
- Model correlated data using mixed models and generalized estimating equations; estimate robust standard errors by performing a cluster-level bootstrap (resampling independent observational units).

Real versus simulated data

In this book, we will use a combination of real and simulated data sets. Most of the data sets encountered in this book are contained in various R packages, including the `Data4Ecologists` package, which I created to go along with this book. This package can be installed in R using:

```r
devtools::install_github("jfieberg/Data4Ecologists")
```

Students are sometimes skeptical of the value of working with simulated data, but there are many reasons why working with simulated data is useful (Kéry 2010):

- With simulated data, we know the truth, and thus, we can compare estimates to truth. This is the best way to see if we can recover the parameter values used to generate the simulated data.
- Simulations can help us determine if we have coding errors, particularly when we are writing new code or developing a new analytic method. If we cannot recover the parameters values used to simulate data, that may indicate we have a bug in our code.
- Simulated data can facilitate understanding of sampling distributions, one of the most important concepts in statistics.
- We can study the properties of an estimator (its mean, its variance, and whether it is robust to assumption violations).

At the same time, we should be wary of methods only shown to work with simulated data, especially when all assumptions are met. With real data, common assumptions are almost never perfectly met. What then? We need to be able to identify the most critical assumptions, evaluate the impact of violations to them, and come up with strategies that perform well even when assumptions are not perfectly met. Again, simulations can play a significant role here.

Standing on the shoulders of…

I have borrowed many ideas, data sets, and in some cases code from a variety of sources when putting together this book. Of particular importance were:

- The Lock family's book, Unlocking the Power of Data (R. H. Lock et al. 2020), which I have used to teach my undergraduate course in introductory statistics for many years.
- Jack Weiss's courses on statistics for ecologists and environmental scientists at the University of North Carolina-Chapel Hill (unfortunately, he passed away a few years ago, and his web sites are no longer easy to track down).
- Marc Kery's Introduction to WinBugs for Ecologists (Kéry 2010)
- Zuur et al's Mixed Effects Models and Extensions in Ecology with R (Zuur et al. 2009)
- Ben Bolker's Ecological Models and Data in R (B. M. Bolker 2008)

I typically include both Kery's and Zuur et al's books as recommended texts when I teach FW8051, Statistics for Ecologists at the University of Minnesota. Ben Bolker's book is also a great resource for ecologists interested in constructing semi-mechanistic models and does a nice job of introducing frequentist and Bayesian inferential frameworks.

I began writing this book after having taught for several years, using my lecture slides to seed the content. I have attempted to trace ideas back to original authors and credit their work whenever possible. Unless otherwise indicated, third-party texts, images, and other materials quoted in these materials are included on the basis of fair use as described in the Code of Best Practices for Fair Use in Open Education. If you see any material, copyrighted or otherwise, that is not properly acknowledged, please let me know so that I can correct any mistakes that I have made.

How to use this book and associated resources for teaching

I am currently developing other resources concurrent with drafting this book. These resources include:

- A separate companion document with suggested exercises. A current draft can be found here. If you are teaching from the book, I am also happy to share solutions to the exercises.
- An R package containing data sets used in the book and in classroom and homework exercises.

As you read through the book, you will occasionally see *Think-Pair-Share* questions posed, which are meant to force the reader to pause, and reflect on a concept that has recently been introduced. In the classroom, these questions can lead to useful discussions among peers and serve as a test of their understanding.

Feedback

I hope this resource will be useful for others, especially biologists trying to increase their statistical abilities. If you use this book in your course, or if you find any errors or have suggestions or feedback for improving the content, please let me know using this google form. In addition, if you have data sets that would be useful for creating additional exercises, please reach out me. Ultimately, I would like the second edition of the Exercise Book to be titled, *Exercises: Statistics for Ecologists by Ecologists*.

Lastly, I have adopted some code from Ben Marwick's modified "A Minimal Bookdown Example" that allows readers to provide feedback directly on each page of the book using hypothes.is. Readers can highlight text or provide comments once they sign up for an account. This idea was inspired by Matthew Salganik's Open Review Toolkit. The code for the Open Review Toolkit has an MIT license and Ben Marwick's code is is licensed under a Creative Commons Attribution 4.0 International License.

Acknowledgements

I thank the many students I have had in class that have made teaching so worthwhile, and my wife and family for their patience with me as I worked on the book. I thank Alex Bajcz, Garrett Street, and Olivia Douell for reading through the first several chapters and providing thoughtful comments and suggestions. I also thank the many people who have provided comments and suggestions using hypothes.is, especially Robert Buck (StatAnswers Consulting LLC) and Smith Freeman, who read through and commented on the full book, Bert van der Veen (Norwegian University of Science and Technology) who provided extremely helpful suggestions on the multicollinearity chapter, Jarrett Byrnes (UMASS Boston) who has used the book for one of his courses and had several students comment using hypothes.is, and Tiago Marques (University of St. Andrews) who offered extra credit to his students for sending useful comments (there were many!). I appreciate helpful comments and suggestions from Jordan Heiman, Dani Freund, Maija Weaver, Paul Freetown, and the list of hypothes.is user names provided below – if you find yourself in this list and are willing to be acknowledged here, please contact me!

Hypothes.is user names: angeliquedenise, bbrandon, brow6589, chris_laRosee, cpolik, dochvam, forester, irrigavi, michael_d, qnn, Teleopsis, and tschafer.

Part I

Models for Normally Distributed Responses

1 Linear regression review

Learning Objectives

We have 3 objectives in this first section:

1. To review linear regression and its assumptions.
2. To review important statistical concepts (i.e., sampling distributions, p-values, confidence intervals, and prediction intervals) in the context of linear regression.
3. To sharpen your R skills.

1.1 R Packages

As with most chapters, we will load a few key packages upfront:

```
library(ggplot2) # for plots
theme_set(theme_bw()) # black and white background
library(dplyr) # for data wrangling
library(knitr) # for including graphics and changing output options
library(kableExtra) # for creating tables
# See https://haozhu233.github.io/kableExtra/bookdown/use-bootstrap-tables-in-gitbooks-epub.html
options(kableExtra.html.bsTable = T)
```

In addition, we will use data and functions from the following packages:

- `abd` for the `LionNoses` data set
- `Lock5Data` for the cricket chirp data set
- `car` for a residual diagnostic plot
- `ggResidpanel` for residual plots using the `resid_panel` function
- `patchwork` for creating a multi-panel plot
- `performance` for evaluating assumptions of the linear regression model

1.2 Data example: Sustainable trophy hunting of male African lions

We will review the basics of linear regression using a data set contained in the `abd` library in R (Middleton and Pruim 2015). These data come from a paper by Whitman et al. (2004), in which the authors address the impact of trophy hunting on lion population dynamics. The authors note that removing male lions can lead to increases in infanticide, but the authors' simulation models suggest that removal of only older males (e.g., greater than 6 years of age) could minimize these impacts.[1]

How could a researcher (or hunter/guide) tell the age of a lion from afar, though? It turns out that it is possible to get a rough idea of how old a male lion is from the amount of black pigmentation on its nose (Figure 1.1; for more information on how to age lions, see this link).

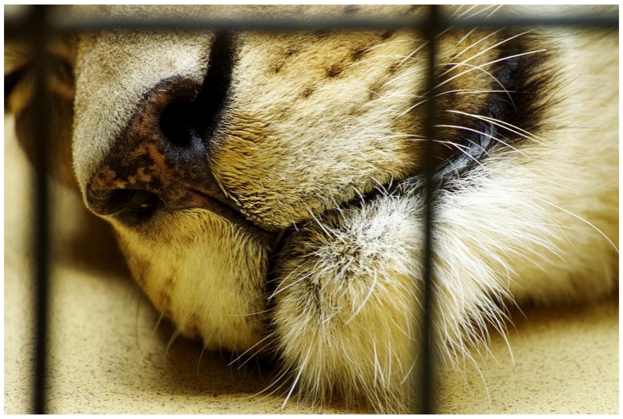


FIGURE 1.1 The amount of black pigmentation on lions noses increases as the lions age.

The `LionNoses` data set in the `abd` library contains 32 observations of ages and pigmentation levels of African lions. To access the data, we need to make sure the `abd` package is installed on our computer (using the `install.packages` function). We then need to tell R that we want to "check out" the package (and data) using the `library` function.

```
install.packages("abd")  # only if not already installed
library(abd) # Each time you want to access the data in a new R session
```

We can then print the first several rows of data, showing the age of the observed lions and their respective levels of nose pigmentation.[2].

```
data(LionNoses)
head(LionNoses)
```

```
##   age proportion.black
## 1 1.1             0.21
## 2 1.5             0.14
## 3 1.9             0.11
## 4 2.2             0.13
## 5 2.6             0.12
## 6 3.2             0.13
```

[1]Whitlock and Schluter (2015) use this data set to introduce linear regression and to highlight differences between confidence intervals and prediction intervals (see Section 1.9). Whereas Whitlock and Schluter (2015) emphasize analytical formulas (e.g., for calculating the least squares regression line, sums of squares, standard errors, and confidence intervals), our focus will be on the implementation of linear regression in R. In addition, we are assuming that most readers will have already been introduced to the mechanics of linear regression in a previous statistics class. Our primary reason for covering linear regression here is that we want to use it as an opportunity to review key statistical concepts in frequentist statistics (e.g., sampling distributions).

[2]We can also get additional information about the data set from its help file by typing `?LionNoses`

Let's plot the data, showing the proportion of the lions' noses that are black on the x-axis and the ages of each lion on the y-axis. We will also plot the best-fit line determined using linear regression (Figure 1.2).

```
ggplot(LionNoses, aes(proportion.black, age)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x) +
  xlab("Proportion Black") +
  ylab("Age")
```

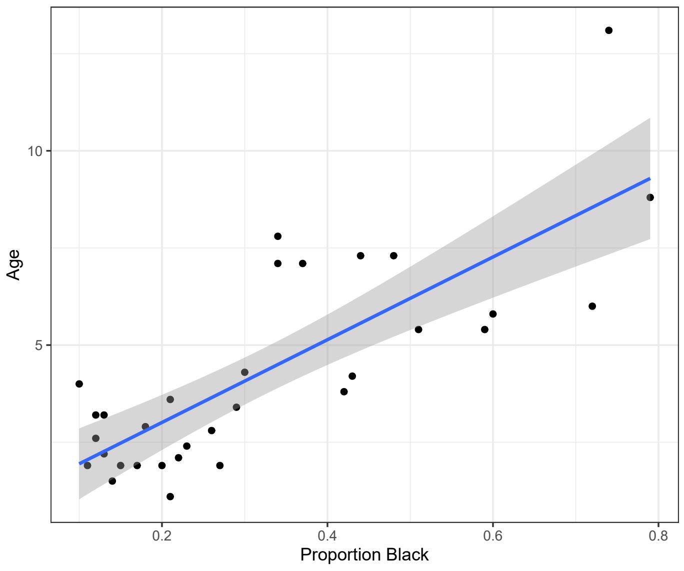


FIGURE 1.2 Lion age versus proportion of their nose that is black, along with a best-fit regression line.

Let's also look at the fitted regression line:

```
lm.nose <- lm(age ~ proportion.black, data = LionNoses)
lm.nose
```

```
##
## Call:
## lm(formula = age ~ proportion.black, data = LionNoses)
##
## Coefficients:
##      (Intercept)  proportion.black
##            0.879            10.647
```

Using the coefficients, we have the following linear model:

$$Y_i = 0.879 + (10.65)X_i + \epsilon_i$$

where Y_i is the age of the lion, X_i is the proportion of the lion's nose that is black, and the ϵ_i's represents random variability about the regression line. We can predict the age of a lion, such as one with 40% of its nose black, by plugging in this value for X_i and setting $\epsilon_i = 0$:

$$\hat{Y}_i = 0.879 + (10.65)(0.4) = 5.14 \text{ (years old)}$$

Note, here and throughout the book, we will use the ˆ symbol when referring to estimates of parameters or functions of parameters.

1.3 Interpreting slope and intercept

Let's write the equation for the linear regression line in terms of the actual variables we used:

$$Age_i = 0.879 + 10.65 Proportion.black_i + \epsilon_i$$

We can also write the model in terms of the expected or average age among lions that have the same level of pigmentation in their nose, $E[Age|Proportion.black]$, by dropping the ϵ_i term (since the $E[\epsilon_i] = 0$)[3]:

$$E[Age|Proportion.black] = 0.879 + 10.65 Proportion.black \tag{1.1}$$

The slope of the regression line, $\beta_1 = 10.65$ tells us that the average Age changes by 10.65 years for each increase of 1 unit of $Proportion.black$, i.e., $10.65 = \frac{\Delta E[Age]}{\Delta Proportion.Black}$. However, this number is difficult to interpret in this case because we cannot change $Proportion.black$ by 1 unit (since all proportions have to be ≤ 1).

We could instead represent nose pigmentation using the percentage of the nose that is black. We add this variable to our data set using the `mutate` function in the `dplyr` package (Wickham et al. 2021) and then refit our model using our new variable in place of our old one.

```
LionNoses <- LionNoses %>%
  mutate(percentage.black = 100*proportion.black)
lm.nose2 <- lm(age ~ percentage.black, data = LionNoses)
lm.nose2
```

```
##
## Call:
## lm(formula = age ~ percentage.black, data = LionNoses)
##
## Coefficients:
##      (Intercept)  percentage.black
##           0.8790            0.1065
```

We see that our slope changes to 0.1065, suggesting that the average Age changes by 0.1065 years for each increase of 1 unit of $percent.black$. This example highlights an important point: *the magnitude of β is not very useful for deciding on the relative importance of different predictors since it is tied to the scale of measurement.* If we change the units of a predictor variable from m to km, our β will change by three orders of magnitude ($\beta_{km} = 1000\beta_m$), yet the importance of the predictor remains the same. One strategy for addressing this issue is to scale predictors using their sample standard deviation, which can facilitate comparisons among

[3]Expected value will be defined formally in Section 9

coefficients in models that contain multiple predictors (Schielzeth 2010); we will demonstrate this approach in Section 3.

To interpret the intercept, it helps to eliminate everything on the right hand side of the regression equation (equation (1.1)) except for the intercept. For example, if we set $Proportion.black_i = 0$, we see that the intercept estimates the *mean* (or expected) *Age* of lions that have no black pigmentation on their nose:

$$E[Age|Proportion.black = 0] = 0.879 \tag{1.2}$$

More generally,

- The **intercept** = the average value of Y when all predictors are set equal to 0 ($E[Y|X = 0]$).
- The **slope** = predicted change in Y per unit increase in X

In many cases, interpreting the intercept will require extrapolating outside of the range of the observed data, and therefore the intercept may be misleading or biologically uninterpretable. To quickly illustrate, consider the simple example of predicting the outside temperature from the number of cricket chirps heard per minute (this is a famous data set often used in introductory statistics classes). The data are included in the `Lock5Data` package (R. Lock 2017).

```
library(Lock5Data)
data(CricketChirps)
```

```
ggplot(CricketChirps, aes(Chirps, Temperature)) +
  geom_point() + geom_smooth(method = "lm", formula = y ~ x) +
  xlab("Chirps per minute") + ylab("Temperature (F)") +
  theme_bw()
```

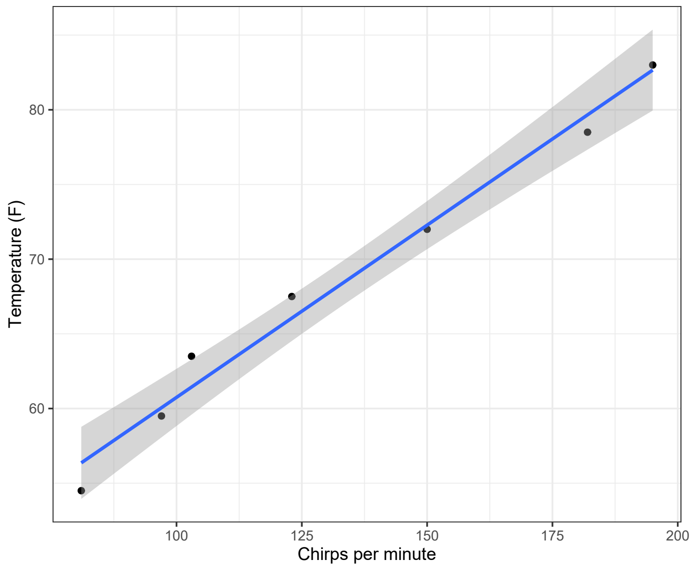


FIGURE 1.3 Linear regression model relating ambient temperature to cricket chirp rate.

```
lm(Temperature ~ Chirps, data = CricketChirps)
```

```
##
## Call:
## lm(formula = Temperature ~ Chirps, data = CricketChirps)
##
## Coefficients:
## (Intercept)       Chirps
##     37.6786       0.2307
```

The intercept predicts that the average temperature will be 37° degrees when crickets stop chirping. As the scatterplot clearly shows (Figure 1.3), this calculation requires extrapolating well outside of the range of the observed data. In fact, crickets stop chirping well before the temperature hits 37° F, but we have no data at lower chirp rates or temperatures to inform the model that this is the case.

Often, it will be helpful to center predictors, which means subtracting the sample mean from all observations. This will create a new variable that has a mean of 0. An advantage of this approach is that our intercept now tells us about the average or expected value of Y when our original variable, X, is set to its mean value instead of to 0. For example, we can fit the same model to the cricket data, below, but after having subtracted the mean number of Chirps in the data set[4].

```
CricketChirps %>% dplyr::summarize(mean(Chirps))
```

```
##   mean(Chirps)
## 1          133
```

```
lm(Temperature ~ I(Chirps - mean(Chirps)), data = CricketChirps)
```

```
##
## Call:
## lm(formula = Temperature ~ I(Chirps - mean(Chirps)), data = CricketChirps)
##
## Coefficients:
##              (Intercept)  I(Chirps - mean(Chirps))
##                  68.3571                    0.2307
```

In our revised model, the slope stays the same, but the intercept provides an estimate of the temperature when the number of Chirps per minute = 133, the mean value in the data set.

At this point, we have reviewed the basic interpretation of the regression model, but we have yet to say anything about uncertainty in our estimates or statistical inference (e.g., confidence intervals or hypothesis tests for the intercept or slope parameters). Statistical inference in frequentist statistics relies heavily on the concept of a sampling distribution (Section 1.6). To derive an appropriate sampling distribution, however, we will need to make a number of assumptions, which we outline in the next section.

[4]Note, we use the `I()` function to create our new variable within the call to `lm`, but we could have used `mutate` to create this variable and add it to our data set instead.

1.4 Linear regression assumptions

Mathematically, we can write the linear model as:

$$\begin{aligned} Y_i &= \beta_0 + \beta_1 X_i + \epsilon_i \\ \epsilon_i &\sim N(0, \sigma^2), \end{aligned} \tag{1.3}$$

where we have assumed the errors, ϵ_i, representing deviations of each Y_i from the regression line, follow a Normal distribution with mean 0 and constant variance, σ^2. Alternatively, we could write down the model as:

$$Y_i \sim N(\mu_i, \sigma^2), \text{ with } \mu_i = \beta_0 + \beta_1 X_i \tag{1.4}$$

In this alternative format, we have made it clear that we are assuming the distribution of Y_i is Normal with mean that depends on X_i. The advantage of writing the model using eq. (1.4) is that this format is more similar to what we will use to describe generalized linear models that rely on statistical distributions other than the Normal distribution (e.g., Section 14).

The equations capture three of the basic assumptions of linear regression:

- Homogeneity of variance (constancy in the scatter of observations above and below the line); $Var(\epsilon_i) = Var(Y_i|X_i) = \sigma^2$.

- Linearity: $E[Y_i \mid X_i] = \mu_i = \beta_0 + X_i\beta_1$; this assumption tells us that the mean value of Y depends on X and is given by the line $\mu_i = \beta_0 + X_i\beta_1$
- Normality: ϵ_i, or equivalently, $Y_i|X_i$ follows a Normal (Gaussian) distribution

In addition, we have a further assumption:

- Independence of the errors: Correlation$(\epsilon_i, \epsilon_j) = 0$ for all pairs of observations i and j. This means that knowing how far observation i will be from the true regression line tells us nothing about how far observation j will be from the regression line.

Of the assumptions of linear regression, the assumptions of linearity and constant variance are often more important than the assumption of Normality (see e.g., Knief and Forstmeier 2021 and references therein), especially for large sample sizes. As we will see when discussing maximum likelihood estimators (Section 10)[5], one can often justify using a Normal distribution for inference when sample sizes are large due to the Central Limit Theorem.

Think-Pair-Share: How are these assumptions reflected in the Figure 1.4? How can we evaluate the assumptions of a linear regression using our data?

1.5 Evaluating assumptions

We can use residual plots to graphically evaluate the assumptions of linear regression. A plot of residuals, $r_i = (Y_i - \hat{Y}_i)$ versus predicted or fitted values $(\hat{Y}_i)$ can be used to evaluate:

[5]An estimator is a *method* of generating an estimate of an unknown parameter from observed data.

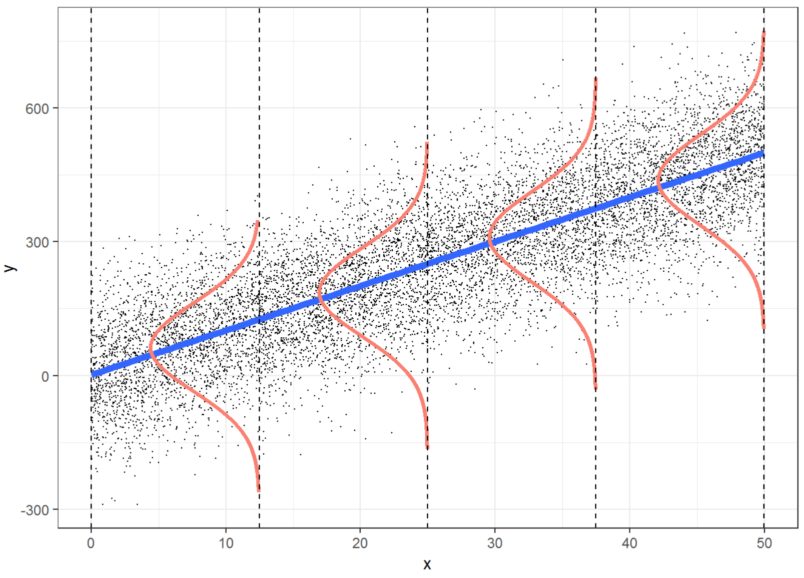


FIGURE 1.4 Figure illustrating the assumptions associated with simple linear regression. This figure was produced using modified code from an answer on stackoverflow by Roback and Legler (2021) (https://bookdown.org/roback/bookdown-bysh/).

a) the linearity assumption (there should be no patterns as we move from left to right along the x-axis).
b) the constant variance assumption (the degree of scatter in the residuals above and below the horizontal line at 0 should be the same for all fitted values).

Figure 1.5 depicts linear regression lines and plots of residuals versus fitted values for situations in which the assumptions are violated. In panel A), the linearity assumption is violated. This results in a non-linear trend in the residuals across the range of fitted values (panel C); Y is under-predicted for small and large values of X and overpredicted for moderate values of X. In panel B), the data exhibit non-constant variance with larger residuals for larger values of X. This assumption violation results in a fan-shaped pattern in the residual versus fitted value plot (panel D). This pattern is common to many count data sets and can be accommodated using a statistical distribution where the variance increases with the mean (e.g., Poisson or Negative Binomial; Section 15) or by using a model that explicitly allows for non-constant variance (Section 5).

1.5.1 Exploring assumptions using R

Linear regression objects in R have a default `plot` function associated with them, which we can use to evaluate the assumptions of linear regression. Let's explore it using our regression model fit to the lion nose data set (Figure 1.6).

```r
par(mfrow=c(2, 2)) # create a 2 x 2 panel plot
plot(lm.nose)
```

The top left panel of Figure 1.6 is our residual versus fitted value plot. The red line, included by default, is a smooth through the data and can be used to evaluate the linearity assumption. For small data sets, like this one, this line can be deceiving. To eliminate it, we could add `add.smooth=FALSE` to the plot function.

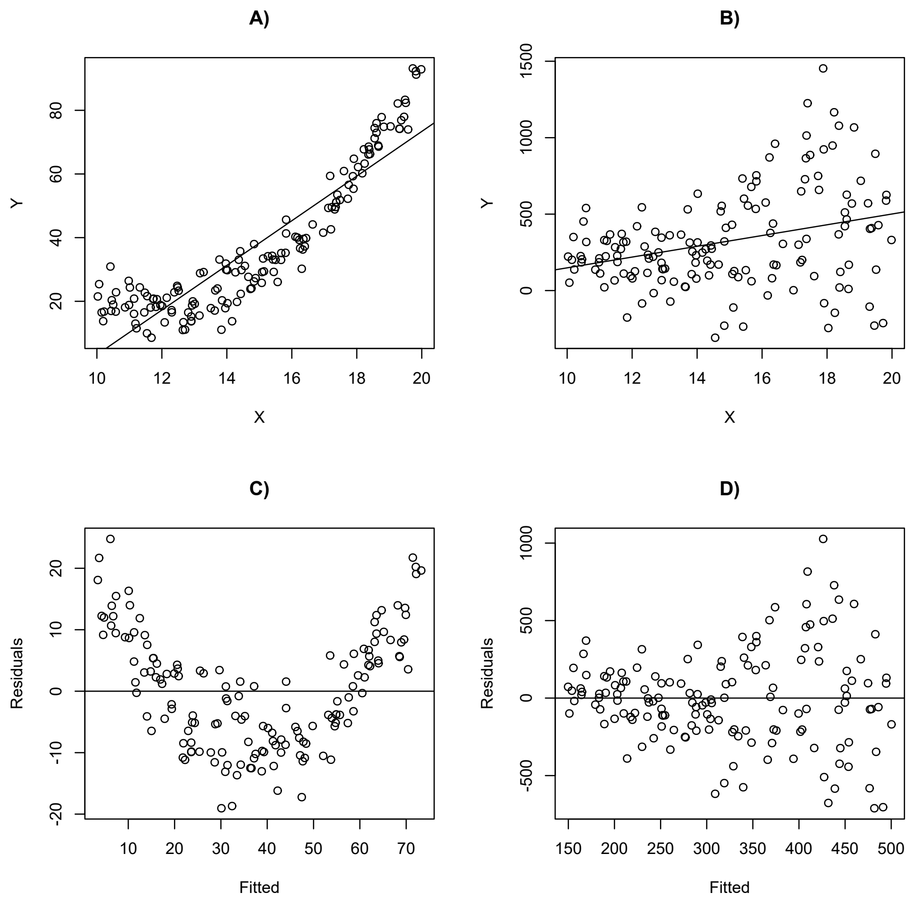


FIGURE 1.5 Linear regression lines with data overlayed and residual versus fitted values plots for situations in which the assumptions are violated. Panels (A, C) depict a situation where the linearity assumption is violated. Panels (B, D) depict a scenario where the constant variance assumption is violated.

The other 3 plots all use standardized residuals, which are also typically referred to as studentized residuals. Standardized residuals are formed by taking the residuals, $Y_i - \hat{Y}_i$ and dividing them by an estimate of their standard deviation, $\sqrt{\widehat{var}(r_i)}$. Standardized residuals offer a few advantages. First, if all assumptions hold, standardized residuals will follow a $t-$distribution with $n-p$ degrees of freedom, where n is the sample size and p is the number of regression parameters in the model (two in the case of simple linear regression); thus, they provide an easy way to check for outliers (e.g., when n is large, we should expect roughly 99% of standardized residuals to occur between -3 and 3). Second, they will have constant variance, whereas non-standardized residuals will not. This latter advantage may seem surprising, especially since an assumption of the linear regression model is that the errors, ϵ_i, have constant variance. We must remember, however, that the residuals, $r_i = Y_i - \hat{Y}_i$, are *estimates* of the errors, $\epsilon_i = Y_i - E[Y_i|X_i]$. These estimates will be more variable as we move away from the average value of X.

The top right panel of Figure 1.6 is a Q-Q plot, which shows quantiles of the standardized residuals versus quantiles of a standard Normal distribution. If the Normality assumption holds, we would expect to see the residuals largely fall along the dotted line indicating that the quantiles of the residuals (e.g., their 5th percentile, median, and 90th percentile, etc.) line up with the quantiles of the standard Normal distribution. In the lower left panel of of Figure 1.6, we see a plot of the square-root of absolute values of the standardized

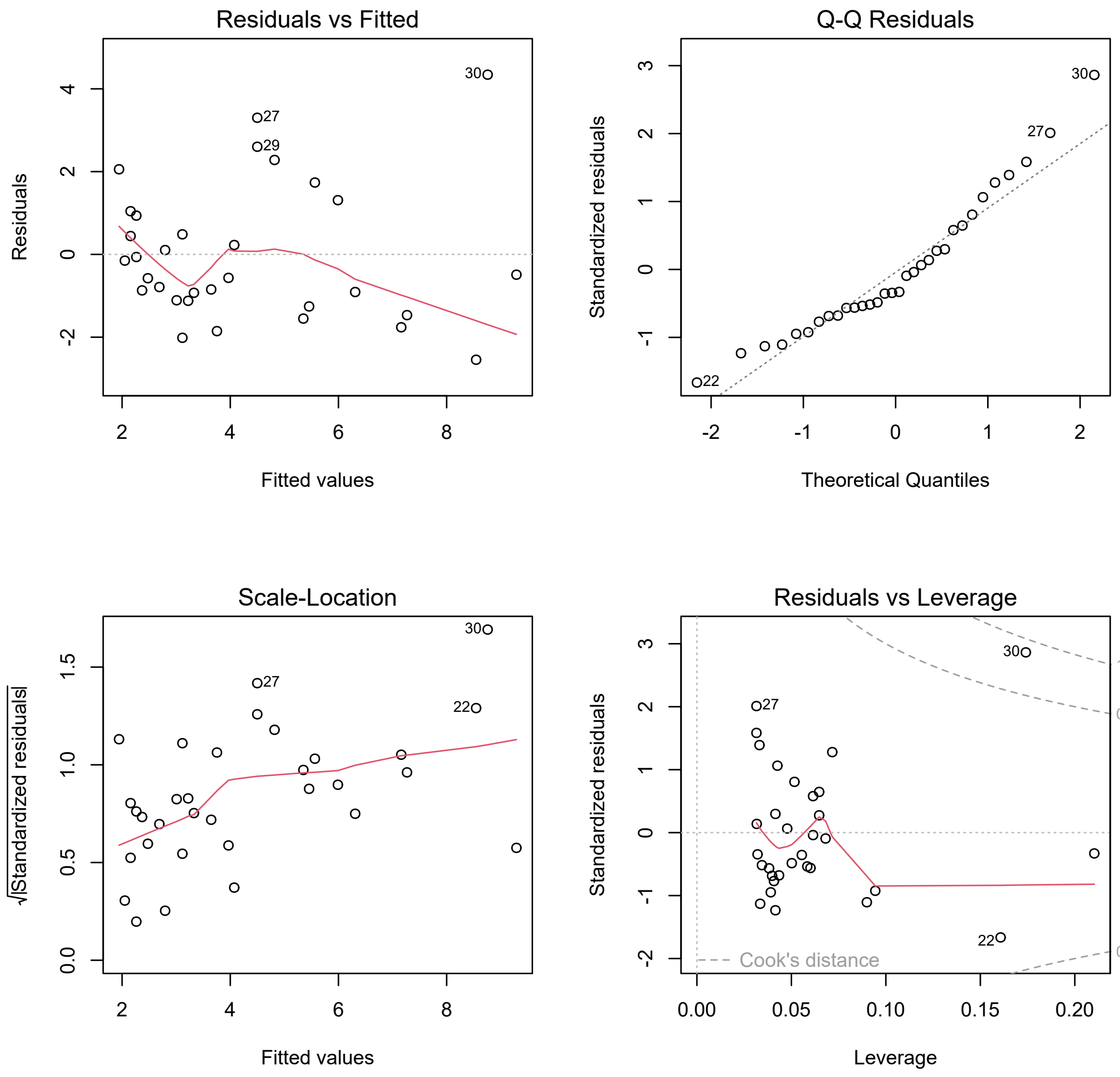


FIGURE 1.6 Residual diagnostic plots constructed using the `plot` function with a fitted linear regression model in R.

residuals versus fitted values. Here, we should see constant scatter along the x-axis if the constant variance assumption holds. Taking the absolute value of the residuals effectively reflects the negative residuals above the horizontal line $y = 0$, making it easier to see departures from constant variance in small data sets. Lastly, the lower right panel plots residuals versus leverage, a measure of how far away each observation is in "predictor space" from the other observations. Observations with high leverage are often, but not always, highly influential points, meaning that deletion of these points leads to large changes in the regression parameters. In each of the panels, interesting and potentially problematic or influential points are indicated by their row number (which might suggest here that we should take a closer look at observations in rows 22, 27, 29, and 30).

Interpreting these plots usually involves some subjectivity. In the case of the lion nose data, the diagnostic plots look OK to me, though there is some evidence that the variance may increase as $\hat{Y}$ increases (lower left panel of Figure 1.6); there are also a few potential outliers that show up in each of the panels that might be worth inspecting (e.g., to see if they could represent data entry errors or if there may be some other explanation for why they differ from most other observations).

It can be instructive to create diagnostic plots using simulated data for which all assumptions are met. It is also possible to add confidence bands to residual plots using what is referred to as a parametric bootstrap (i.e., by repeatedly simulating data from the fitted model). For example, the `qqPlot` function in the car package (Fox and Weisberg 2019) will add confidence bands to facilitate diagnosing departures from Normality (Figure 1.7).

```
car::qqPlot(lm.nose, id=FALSE)
```

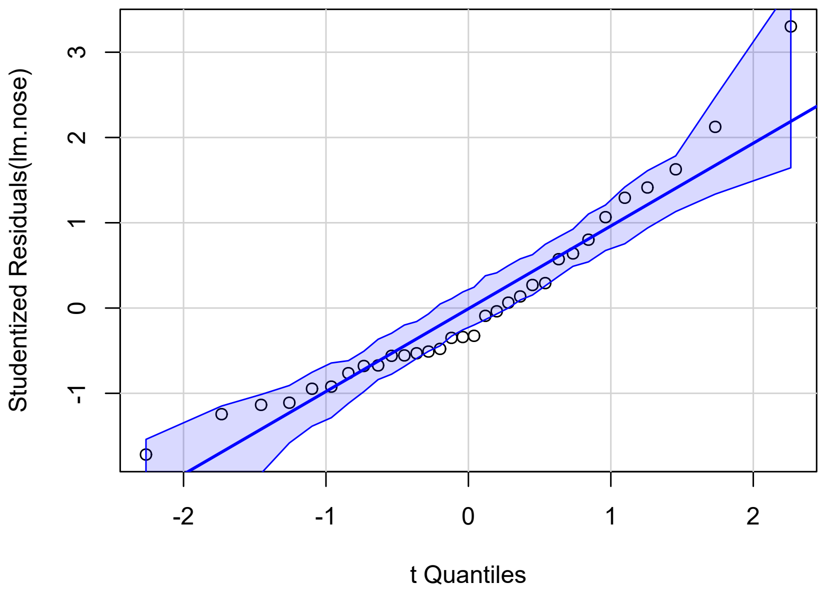


FIGURE 1.7 Output from running the `qqplot` function from the car package (Fox and Weisberg 2019). If the Normality assumption is reasonable, we should expect most points to fall within the confidence bands, which are computed using a parametric bootstrap.

For a prettier set of residual plots, we could explore options in other packages. For example, the `resid_panel` function in the `ggResidpanel` package (Goode and Rey 2019) provides the following 4 plots

- the typical residual versus fitted value plot (top left)
- a qqplot for evaluating Normality (top right)
- an index plot showing residuals versus observation number in the data frame (this could be useful for exploring the independence assumption if observations are ordered temporally or spatially)
- a histogram of residuals with a best-fit Normal distribution overlayed.

```
library(ggResidpanel)
resid_panel(lm.nose)
```

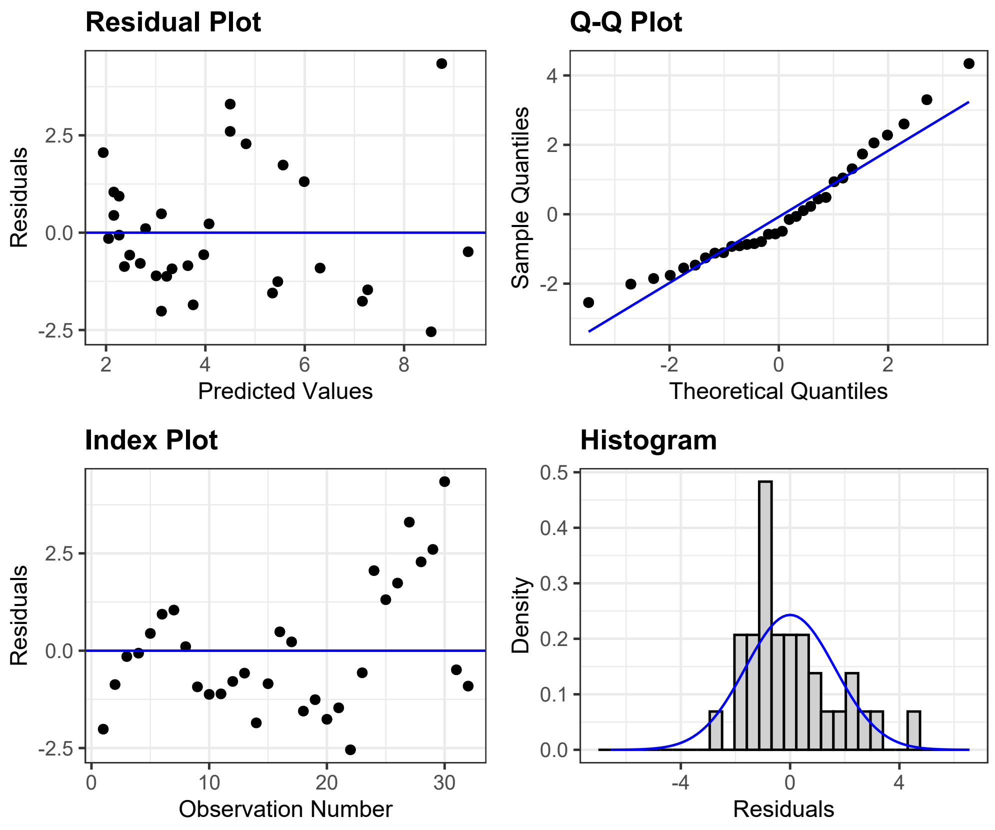


FIGURE 1.8 Residual plots constructed using the `ggResidpanel` package.

The `plot_model` function in the `sjPlot` package (Lüdecke 2021) also provides the following diagnostic plots (with hints about what to look for in each plot):

- a qqplot and a density plot of the residuals to evaluate Normality
- a residual versus fitted value plot to evaluate linearity and constant variance

```
library(patchwork) # to create the multi-panel plot
dplots <- sjPlot::plot_model(lm.nose, type="diag")
((dplots[[1]] + dplots[[2]])/ dplots[[3]])
```

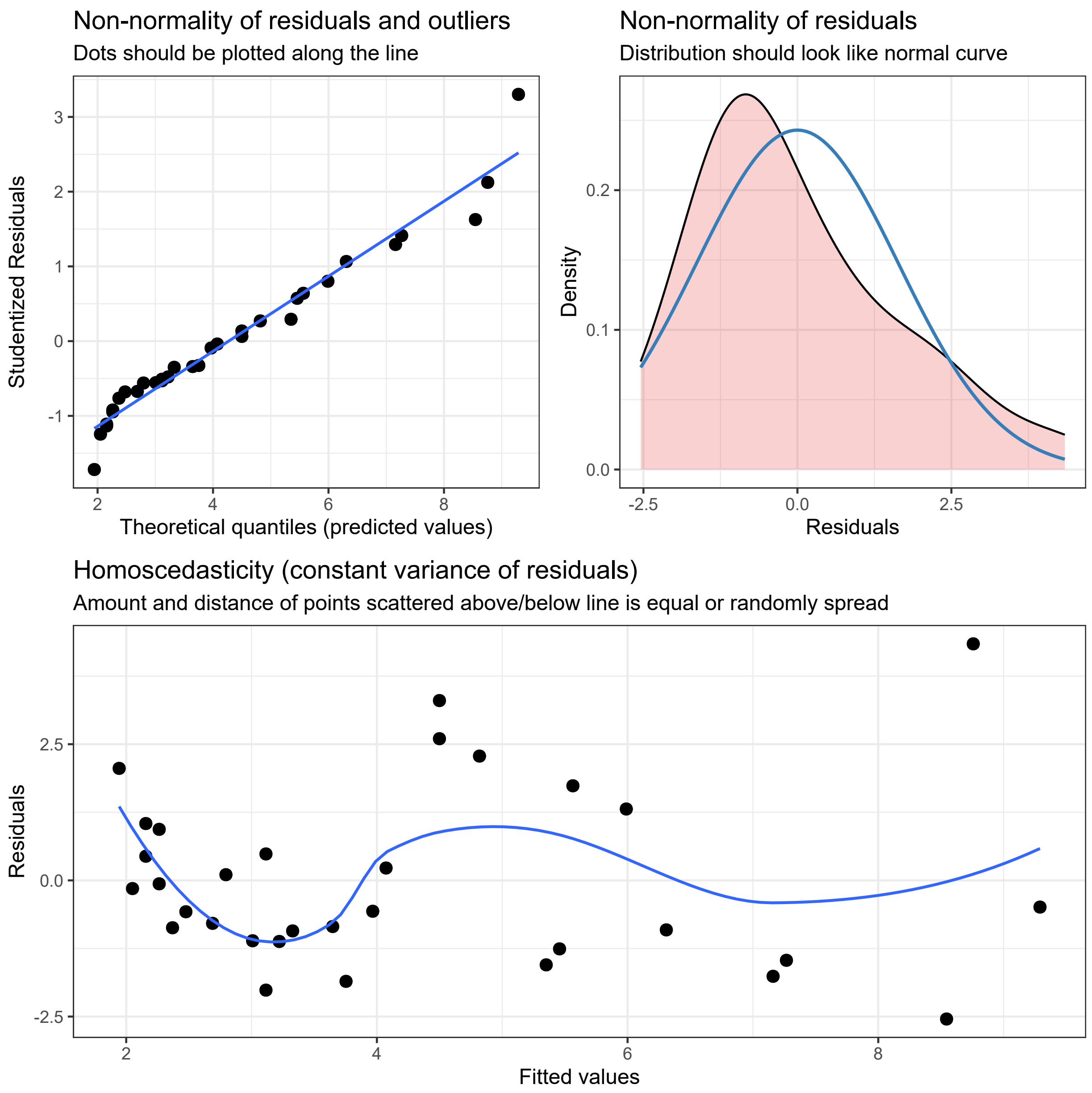


FIGURE 1.9 Diagnostic plots using the `plot_model` function in the `sjPlot` package.

Lastly, we will often use the `check_model` function in the performance' package (Lüdecke et al. 2021), which offers a variety of different diagnostic plots (Figure 1.10).

```r
library(performance)
check_model(lm.nose, check = c("linearity", "homogeneity", "qq", "normality"))
```

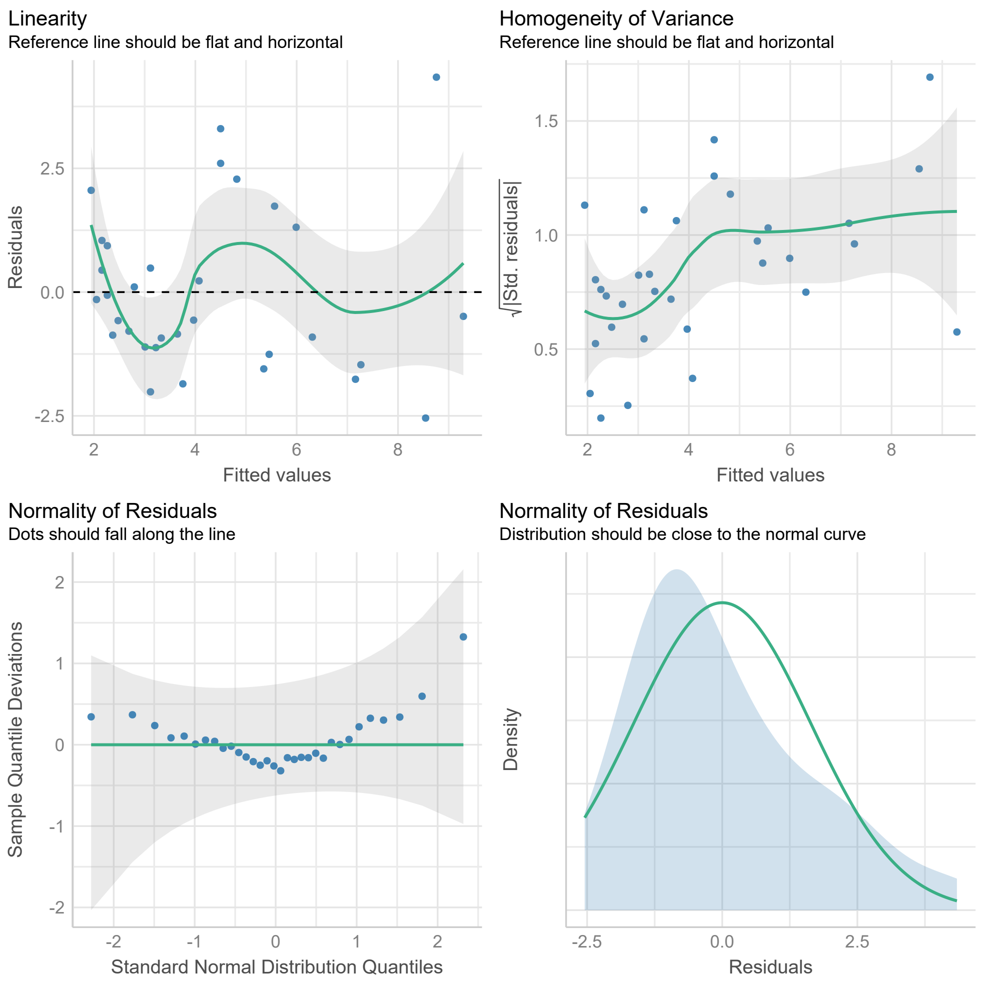


FIGURE 1.10 Diagnostic plots created using the `check_model` function of the `performace` package.

One thing to note here is that the `performance` package displays a detrended qqplot (lower left of Figure 1.10)), in which we expect the observations to fall near the horizontal line at y = 0 if all assumptions are met.

1.6 Statistical inference: Sampling distributions

R has a `summary` function that allows us to explore fitted regression models in more detail. Let's have a look at the linear regression model we fit to the lion nose data.

```
summary(lm.nose)
```

```

Call:
lm(formula = age ~ proportion.black, data = LionNoses)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.5449 -1.1117 -0.5285  0.9635  4.3421 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)        0.8790     0.5688   1.545    0.133    
proportion.black  10.6471     1.5095   7.053 7.68e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.669 on 30 degrees of freedom
Multiple R-squared:  0.6238,	Adjusted R-squared:  0.6113 
F-statistic: 49.75 on 1 and 30 DF,  p-value: 7.677e-08
```

Our goal should be to understand most of this output at a relatively high level. We have discussed the coefficient estimates in the summary table, but what about the `Std. Error`, `t value`, and `Pr(>|t|)` columns? The `t value` and `Pr(>|t|)` columns are associated with hypothesis tests for the slope and intercept parameters. Specifically, the first row provides a test statistic and p-value for the following null hypothesis test:

H_0: $\beta_0 = 0$ versus H_A: $\beta_0 \neq 0$,

where β_0 is the intercept and H_0 and H_A represent the null and alternative hypotheses. The second row provides a test statistic and p-value for testing whether the slope parameter is 0. You may have some notion of what a standard error represents (e.g., you might have used $\pm 2SE$ to calculate approximate confidence intervals in the past). To really understand the inner workings of these different summaries, however, requires a firm grasp of sampling distributions. In frequentist statistics, we characterize uncertainty by considering the variability of our statistics (e.g., means, regression parameters, etc) across repeated random "trials." To understand what a sampling distribution is, we must imagine (or simulate!) the process of repeatedly collecting new data sets of the same size as our original data set and using the same methods of data collection, and then analyzing these data in the same way as we did in our original analysis. We (almost) never take repeated samples in practice, which makes sampling distributions difficult to conceptualize. However, they are easy to explore using computer simulations.

Formally, a sampling distribution is the distribution of sample statistics computed for different samples of the same size, from the same population. A sampling distribution shows us how a sample statistic, such as a mean, proportion, or regression coefficient, varies from sample to sample. For an entertaining, and useful review, see:

To understand the sampling distribution of our regression parameter estimators, we need to conceptualize repeating the process of:

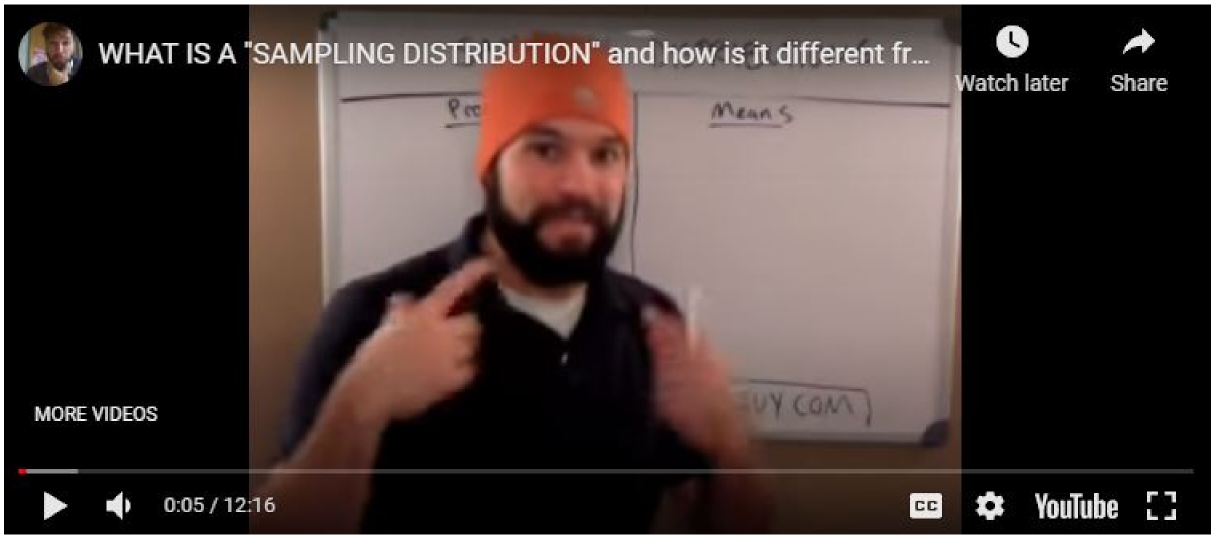

WHAT IS A "SAMPLING DISTRIBUTION" and how is it different fr...
Watch later
Share
MORE VIDEOS
0:05 / 12:16
YouTube

FIGURE 1.11 Sampling distribution video, by Mr. Nystrom, an Advanced Placement Statistics Instructor who is very good at his craft! See: https://www.youtube.com/embed/uPX0NBrJfRI.

1. Collecting a new data set of the same size and generated in the same way as our original data set (e.g., by sampling from the same underlying population or by assuming the data generating process specified by our regression model can be replicated many times).
2. Fitting the same regression model to these new data from step [1].
3. Collecting the estimates of the slope (β_1) and intercept (β_0).

The sampling distribution of $(\hat{\beta}_0, \hat{\beta}_1)$ is the distribution of $(\hat{\beta}_0, \hat{\beta}_1)$ values across the different samples. The standard deviation of a sampling distribution is called standard errors (or SE for short).

1.6.1 Exploring sampling distributions through simulation

Let's assume that the true relationship between the age of lions and the proportion of their nose that is black can be described as follows:

$$Age_i = 0.88 + 10.65 Proportion.black_i + \epsilon_i$$
$$\epsilon_i \sim N(0, 1.67^2)$$

Let's generate a single simulated data set of size $n = 32$.

```r
# Sample size of simulated observations
n <- 32

# Use the observed proportion.black in the data when simulating new observations
p.black <- LionNoses$proportion.black

# True regression parameters
sigma <- 1.67 # residual variation about the line
betas <- c(0.88, 10.65) # Regression coefficients
```

```
# Create random errors (epsilons) and random responses
epsilon <- rnorm(n, 0, sigma) # Errors
y <- betas[1] + p.black*betas[2] + epsilon # Response
```

Now, having simulated a new data set, we can see how well we can estimate the true parameters $(\beta_0, \beta_1, \sigma)$ = (0.88, 10.65, 1.67).

```
lm.sim1 <- lm(y ~ p.black)
summary(lm.sim1)
```

```

Call:
lm(formula = y ~ p.black)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.2401 -0.8812 -0.3871  0.9053  3.2192 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   1.7028     0.5627   3.026  0.00505 ** 
p.black       8.9392     1.4934   5.986 1.45e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.651 on 30 degrees of freedom
Multiple R-squared:  0.5443,	Adjusted R-squared:  0.5291 
F-statistic: 35.83 on 1 and 30 DF,  p-value: 1.45e-06
```

As we might expect, the estimates of our parameters are close to the true values, but not identical to them (note the estimate of σ is given by the `Residual standard error`, equal to 1.651). We can repeat this process, and each time we simulate a new data set, we will get a different set of parameter estimates. This is a good time to introduce the concept of a `for` loop, which will allow us to repeat this simulation process many times and store the results from each simulation.

1.6.2 Writing a for loop

Usually, when it is convenient to do so, we will want to avoid writing code in R that requires the use of loops since loops in R can be slow to run. Nonetheless, there are times when loops are difficult to avoid and other times where they are incredibly convenient. Furthermore, a loop will help us to conceptualize the steps required to form a sampling distribution. In addition, `for` loops will feature prominently when we begin to write JAGS code for implementing models in a Bayesian framework.

When writing a `for` loop, I usually:

1. Create a variable that will specify the number of times to run the operations to be repeated (`nsims`, below).
2. Set up an object, such as a matrix or array, to store results computed during each iteration of the loop (e.g., `sampmeans`, below).

A for loop requires a `for` statement containing a counter (`i`, below) that can take on a set of values (`1:nsims`). Everything that needs to be executed repeatedly is then included between braces `{ }`. As a relatively simple example, we could generate 1000 different sample means from data sets of size 100 generated from a $N(5, 3)$ distribution using:

```
# Number of simulated means
nsims <- 1000
# Set up a matrix to contain the nsims x 1 sample means
sampmeans<-matrix(NA, nrow = nsims, ncol = 1)
for(i in 1:nsims){ # i will be our counter
  #Save the ith sample mean from each iteration
  sampmeans[i] <- mean(rnorm(100, mean = 5, sd = 3))
}
head(sampmeans)
```

```
##          [,1]
## [1,] 4.868652
## [2,] 5.304014
## [3,] 5.663501
## [4,] 5.463369
## [5,] 5.037228
## [6,] 5.098850
```

Each time through the loop, R will generate 100 observations from a Normal distribution using the `rnorm` function and then take the mean of these values using the `mean` function. The result from the i^{th} time through the loop is stored in the i^{th} row of our `sampmeans` matrix.

1.6.3 Exercise: Sampling distribution

Let's use a `for` loop to:

1. Generate 5000 data sets using the code we wrote for simulating observations from the linear regression model with true parameters $(\beta_0, \beta_1, \sigma) = (0.88, 10.65, 1.67)$
2. Fit a linear regression model to each of these data sets
3. Store $\hat{\beta}_1$ in a 5000 x 1 matrix labeled `beta.hat`

Each of these 3 steps (simulating data, fitting the model, and storing the results) should happen between the `{}` of your `for` loop. When you are done, calculate the standard deviation of the 5000 $\hat{\beta}_1$'s. This is an estimate of the standard error of $\hat{\beta}_1$ – i.e., remember that the *standard deviation* of a statistic across repeated samples (i.e., the standard deviation of the sampling distribution) is referred to as the *standard error* of our statistic. This standard error is what is reported in the `Std. Error` column of the regression output (but, it is calculated analytically by the `lm` function rather than through simulation). It tells us how much we should expect our estimates to vary across repeated trials, where a trial represents collecting a new data set of the same size as the original data set and then fitting a linear regression model to that data set. This variability provides us with a useful measure to quantify uncertainty associated with our estimates, $\hat{\beta}_0$ and $\hat{\beta}_1$.

1.7 Sampling distribution of the t-statistic

You may recall from an introductory statistics class that the sampling distribution of standardized regression parameters follows a t-distribution with $n - p$ degrees of freedom (provided all of our assumptions from Section 1.4 hold), where n is our sample size and p is the number of regression parameters in the model (2 in our case, an intercept and slope). Specifically,

$$\frac{\hat{\beta}_1 - \beta_1}{\widehat{SE}(\hat{\beta}_1)} \sim t_{n-2}$$

We can verify this result via simulation.

Think of many repetitions of:

1. Collecting a new data set (of the same size as our original data set)
2. Fitting the regression model

3. Calculating: $t = \frac{\hat{\beta}_1 - \beta_1}{\widehat{SE}(\hat{\beta}_1)}$ (we can do this with simulated data since in this case we know the true β)

A histogram of the different t values should be well described by a Student's t-distribution with $n-2$ degrees of freedom.

1.7.1 Class exercise: sampling distribution of the t-statistic

Let's build on the previous exercise, adding one more step to our loop. For each fitted regression model, let's store $\hat{\beta}_1$ and calculate: $t = \frac{\hat{\beta}_1 - \beta_1}{\widehat{SE}(\hat{\beta}_1)}$.

Helpful hints:

- β_1 = true value used to simulate the data = 10.65
- $\hat{\beta}_1$ is extracted using: `coef(lm.temp)[2]`, where `lm.temp` is the object holding the results of the regression for the i^{th} simulated data set.
- $\widehat{SE}(\hat{\beta}_1)$ is extracted using `sqrt(vcov(lm.temp)[2,2])`

In Figure 1.12, I've included a histogram of my 5000 estimates, with a t-distribution with 2 degrees of freedom overlaid (a link to the code used to create the histogram can be found at the end of this chapter). As expected, the t-distribution provides a perfect fit to the sampling distribution.[6] OK, next, we will consider how we can use this result to create a confidence interval for β_1 or to calculate a p-value to test whether we have sufficient evidence to conclude that $\beta_1 \neq 0$.

1.8 Confidence intervals

Here is how R. H. Lock et al. (2020) define confidence intervals:

[6] One might even say it fits to a t. . . yes, I love dad jokes.

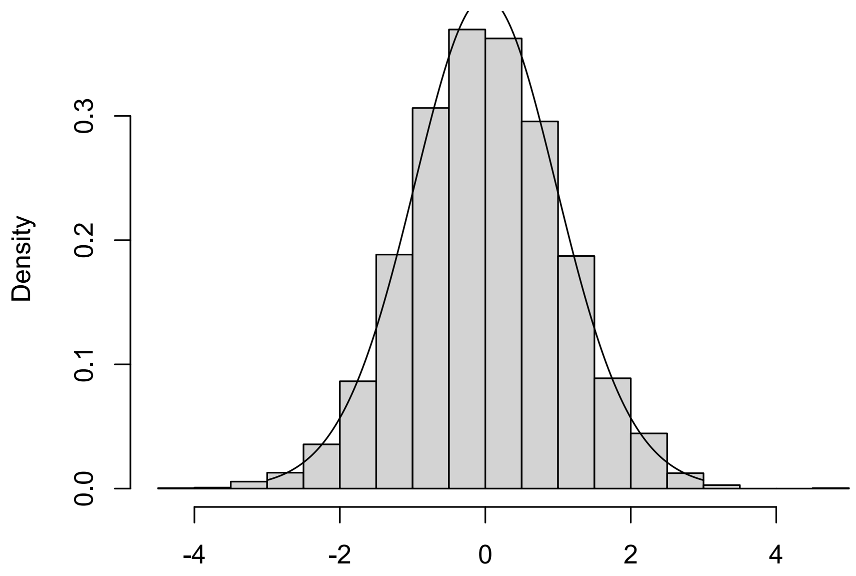


FIGURE 1.12 Sampling Distribution of the t-statistic, $t = \frac{\hat{\beta}_1 - \beta_1}{\widehat{SE}(\hat{\beta}_1)}$. A link to the code used to create the histogram can be found at the end of this chapter.

A confidence interval for a parameter is an interval computed using sample data by a method that will contain the parameter for a specified proportion of all samples. The success rate (proportion of all samples whose intervals contain the parameter) is known as the confidence level.

The key to understanding confidence intervals is to recognize that:

- the parameter we are trying to estimate is a *fixed* unknown (i.e., it is not varying across samples)
- the endpoints of our confidence interval are *random* and will change every time we collect a new data set (the endpoints themselves actually have a sampling distribution!)

Consider repeating the process of randomly sampling from the same population (or, generating new data sets based on the same underlying data-generating model). If we estimate the same parameter and calculate a 95% confidence interval for that parameter with each of these data sets, we should expect that 95% of our intervals should contain the true population parameter. The actual proportion of intervals that contain the true parameter is referred to as the coverage rate.

To construct a confidence interval for β, we will rely on what we just showed to be true, i.e., that:

$$\frac{\hat{\beta}_1 - \beta_1}{\widehat{SE}(\hat{\beta}_1)} \sim t_{n-2}$$

For the lion data set, we had an $n = 32$, so let's look at the t-distribution with 30 degrees of freedom (Figure 1.13).

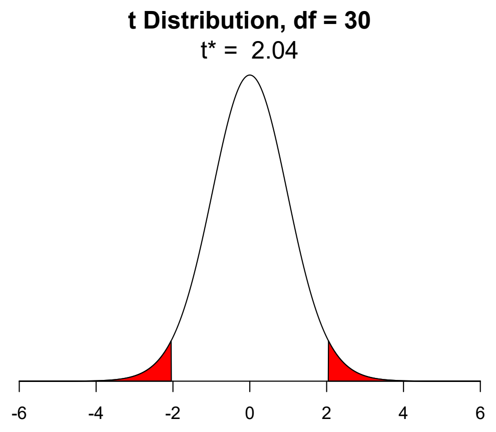


FIGURE 1.13 t-distribution with n-2 degrees of freedom. Shaded red areas demark the outermost 2.5% of the distribution on each side.

In Figure 1.13, I have shaded in red the outer 2.5% of the t-distribution (on both sides), leaving unshaded, the middle 95% of values of the distribution. We can determine the quantiles, sometimes referred to as the critical values, that demarcate these shaded areas using the `qt` function in R (here `q` stands for quantile and `t` for the $t-$distribution). These quantiles are given by:

```
qt(c(0.025, 0.975), df = 30)
```

```
## [1] -2.042272  2.042272
```

As an aside, we could use the `qnorm` function to determine the 2.5^{th} and 97.5^{th} quantiles of a Normal distribution in much the same way:

```
qnorm(c(0.025, 0.975), mean = 0, sd = 1)
```

```
## [1] -1.959964  1.959964
```

The quantiles of the t-distribution are slightly larger in absolute value than those of a Normal distribution since the t-distribution has "wider tails" (it is more spread out than a Normal distribution).

So, we know that 95% of the time when we collect and fit a linear regression model, $\frac{\hat{\beta}-\beta}{\widehat{SE}(\hat{\beta})}$ will fall between -2.04 and 2.04 as long as the assumptions from Section 1.5 are met. This allows us to write a probability statement (eq. (1.5)) from which we can derive the formula for a 95% confidence interval for β_1:

$$P(-2.04 < \frac{\hat{\beta}-\beta}{\widehat{SE}(\hat{\beta})} < 2.04) = 0.95 \tag{1.5}$$

From here, we just use algebra to derive the formula for a 95% confidence interval, $\hat{\beta} \pm 2.04\widehat{SE}(\hat{\beta})$:

$$\begin{aligned} P(-2.04 < \frac{\hat{\beta}-\beta}{\widehat{SE}(\hat{\beta})} < 2.04) = 0.95 \\ \Rightarrow P(-2.04\widehat{SE}(\hat{\beta}) < \hat{\beta}-\beta < 2.04\widehat{SE}(\hat{\beta})) = 0.95 \\ \Rightarrow P(-\hat{\beta}-2.04\widehat{SE}(\hat{\beta}) < -\beta < -\hat{\beta}+2.04\widehat{SE}(\hat{\beta})) = 0.95 \\ \Rightarrow P(\hat{\beta}+2.04\widehat{SE}(\hat{\beta}) > \beta > \hat{\beta}-2.04\widehat{SE}(\hat{\beta})) = 0.95 \end{aligned}$$

Thus, if our linear regression assumptions are valid, and if we consider repeating the process of (collecting data, fitting a model, forming a 95% confidence interval using $\hat{\beta} \pm 2.04\widehat{SE}(\hat{\beta})$), we should find that the true β is contained in 95% of our intervals. Hopefully, this helps clarify the definition given by R. H. Lock et al. (2020):

A confidence interval for a parameter is an interval computed from sample data by a method that will capture the parameter for a specified proportion of all samples.

In R, we can also use the `confint` function to calculate the confidence interval directly from the linear model object.

```
confint(lm.nose)
```

```
                      2.5 %    97.5 %
(Intercept)      -0.2826733  2.040686
proportion.black  7.5643082 13.729931
```

Interpretation: We say that we are 95% sure that the true slope (relating the proportion of nose that is black to a lion's age) falls between 7.56 and 13.73. We are 95% sure, because this method should work 95% of the time!

False interpretation: It is tempting to say there is a 95% probability that β is in the interval we have constructed. Technically, this is an incorrect statement, at least as far as frequentist statistics go. For any *particular interval*, β is either in the interval or it is not (the probability is either 0 or 1). We just don't know which situation applies for our particular interval. We will see later, however, that this type of interpretation can be used with Bayesian credibility intervals (Section 11).

1.8.1 Exercise: explore confidence intervals through simulation

Let's simulate another 5000 data sets in R, this time adding code to:

1. Determine 95% confidence limits for each simulated data set.
2. Determine whether or not each CI contains the true β_1.
3. Estimate the coverage rate as the proportion of the 5000 confidence intervals that contain the true β_1.

To hammer home that the limits of a confidence interval have a *sampling distribution*, I plotted estimates of β_1 and their associated confidence intervals for 100 simulated data sets (Figure 1.14). The solid vertical line shows us the true β_1 (a fixed, non-varying parameter). With each simulated data set, we get a different estimate and a different confidence interval. The green intervals contain the true parameter and the red ones do not. On average, 95% of the intervals should be green and contain the true parameter.

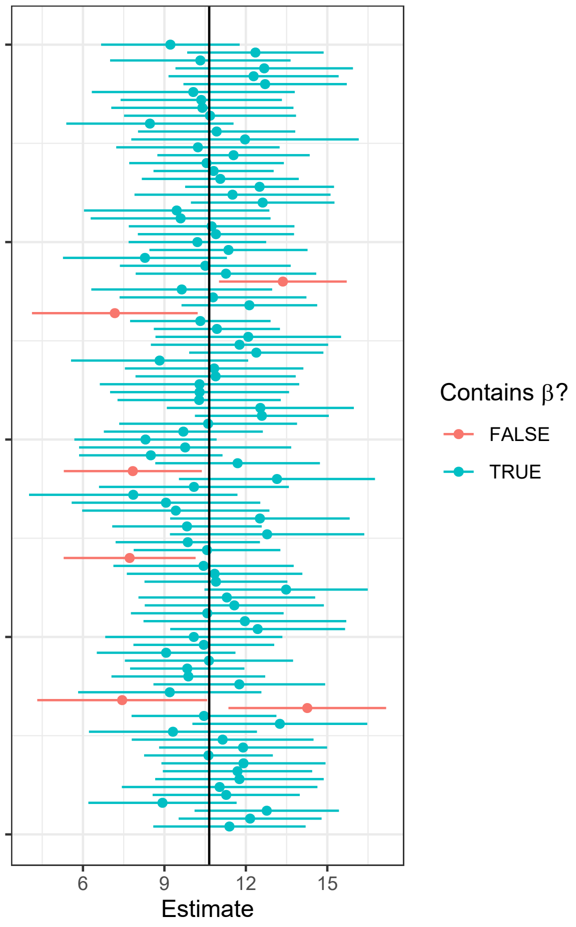


FIGURE 1.14 First 100 confidence intervals associated with the simulated data from the lion nose regression model. Points give estimates of the slope parameter and lines indicate 95% confidence intervals. Green intervals capture the true parameter value used to simulate the data and red intervals do not capture this parameter value.

1.9 Confidence intervals versus prediction intervals

The past few sections covered methods for quantifying uncertainty in our regression parameter estimators (using SE's and confidence intervals). How do we quantify uncertainty associated with the predictions we derive from our model? Well, that depends on whether we are interested in the *average age* of lions with a specific proportion of their nose that is black ($\hat{\mu}_i$) or we are interested in the age of a *specific lion* (singular), $\hat{Y}_i$.

In the former case, we are interested in a confidence Interval for the mean (synonymous with the location of the regression line) for a given value of X, whereas in the latter case, we are interested in a prediction interval associated with a particular case.

- A (1-α)% confidence interval will capture the true *mean* Y at a given value of X (1-α)% of the time.
- A (1-α) prediction interval will capture the Y value for a *particular case (or individual)* (1-α)% of the time.

We can calculate confidence and prediction intervals using the `predict` function.

```r
newdata <- data.frame(proportion.black = 0.4)
predict(lm.nose, newdata, interval = "confidence", level = 0.95)
```

```
       fit      lwr      upr
1 5.137854 4.489386 5.786322
```

```r
predict(lm.nose, newdata, interval = "prediction", level = 0.95)
```

```
       fit      lwr     upr
1 5.137854 1.668638 8.60707
```

This tells us that we are 95% sure that the *mean age* of lions with noses that are 40% black is between 4.49 and 5.79. And, we can be 95% sure that the age of a randomly chosen *individual lion* that has 40% of its nose black will fall between 1.67 and 8.60. Note the prediction interval is considerably wider. Why? The prediction interval must consider not only the uncertainty in the location of the regression line but also the variability of the points around the line. Remember, this variability is represented by a Normal distribution (see Figure 1.4) and is quantified using the residual standard error:

$$\hat{\sigma} = s_\epsilon = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n-2}}$$

We plot confidence and prediction intervals across the range of observed X values, below:

```r
newdata <- data.frame(proportion.black = seq(0.08, 0.81, length = 100))
predict.mean <- cbind(newdata, predict(lm.nose, newdata, interval = "confidence"))
predict.ind <- cbind(newdata, predict(lm.nose, newdata, interval = "prediction"))
ggplot(LionNoses, aes(proportion.black, age)) +geom_point() +
  geom_line(data=predict.mean, aes(proportion.black, lwr), color="red", linetype = "dashed") +
  geom_line(data=predict.mean, aes(proportion.black, upr), color="red", linetype = "dashed") +
  geom_line(data=predict.ind, aes(proportion.black, lwr), color="black", linetype = "dashed") +
  geom_line(data=predict.ind, aes(proportion.black, upr), color="black", linetype = "dashed") +
  geom_line(data = predict.mean, aes(proportion.black, fit)) +
  xlab("Proportion black") + ylab("Age")
```

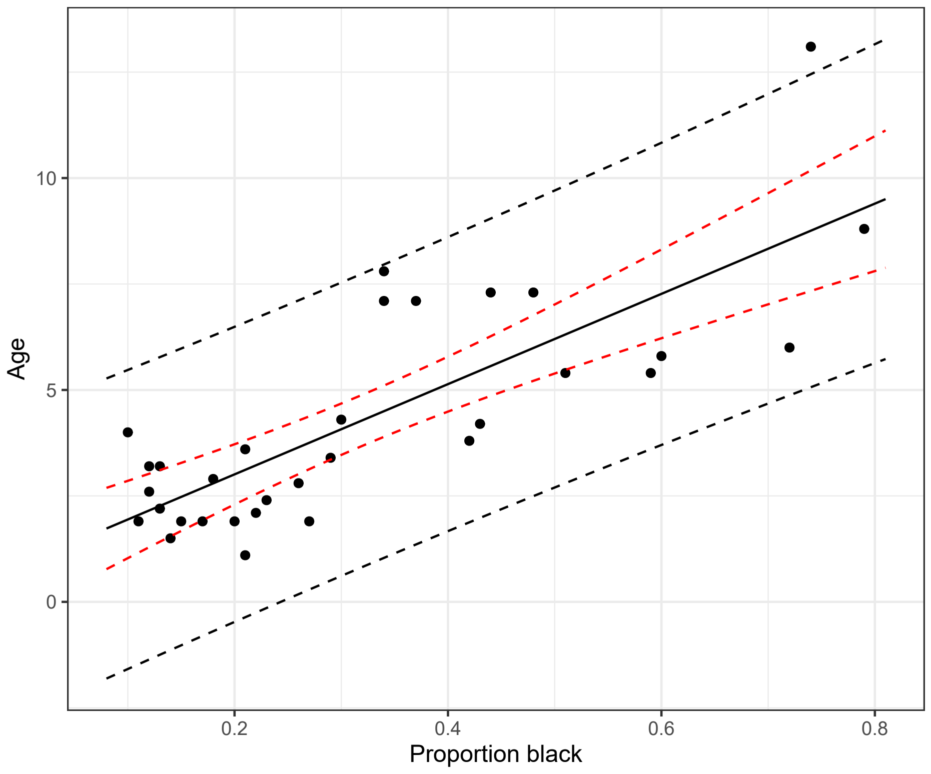


FIGURE 1.15 Confidence and prediction intervals for the regression line relating the age of lions to the proportion of their nose that is black.

1.10 Hypothesis tests and p-values

Let's next consider the `t value` and `Pr(>|t|)` columns in the summary output of our fitted regression model. Before we do this, however, it is worth considering how null hypothesis tests work in general. In essence, null hypothesis testing works by: 1) making an assumption about how the world works (more specifically, assumptions about the processes that gave rise to our data[7]; these assumptions are captured by our null hypothesis); 2) deducing what sorts of data we should expect to see if the null hypothesis is true; 3) determining if our observed data are plausible in light of these expectations; and 4) (potentially) making a formal conclusion about the null hypothesis based on the level of plausibility of the data given the null hypothesis.

Null hypotheses are typically (though not always) framed in a way that suggests that there is nothing

[7]We may refer to a data generating process (DGP)

interesting going on in the data – e.g., there is no difference between treatment groups, the slope of a regression line is 0, the errors about the regression line are Normally distributed, etc. We then look for evidence in the data that might suggest otherwise (i.e., that the null hypothesis is false). If there is sufficient evidence to suggest the null hypothesis is false, then we reject it and conclude that the alternative must be true (i.e., there is a difference between treatment groups, the slope of the regression line is not 0, the errors are not Normally distributed). If we reject the null hypothesis, then the next step should be to determine if the patterns are strong enough to be of interest to us. In other words, are the differences between treatment groups big enough that we should care about them? Is the slope of the regression line sufficiently different from 0 to indicate a meaningful relationship between explanatory and response variables? Is the distribution of the errors sufficiently different from Normal that we might worry about the impact of this assumption on our inferences. If the hypothesis test focuses on a population parameter (e.g., a difference in population means or a regression slope), then we should follow up by calculating a confidence interval to determine values of the parameter that are consistent with the data (i.e., we cannot rule out any of the values that lie within the confidence interval).

Importantly, if we *fail to reject the null hypothesis* that does not mean that the null hypothesis is true! We just lack sufficient evidence to conclude it is false, and lack of evidence may be due to having too small of a sample size. If we fail to reject the null hypothesis, we should again consider a confidence interval to determine if we can safely rule out meaningful effect sizes (e.g., by examining whether a confidence interval excludes them).[8]

How do we measure evidence against the null hypothesis or the degree to which the data deviate from what we would expect if the null hypothesis were true? First, we need to choose a test statistic to summarize our data (e.g., a difference in sample means, a regression slope, or a standardized version of these statistics where we divide by their standard error). We are free to choose any statistic that we want, but it should be sensitive to deviations from the null hypothesis – i.e., the distribution of the statistic should be different depending on whether the null hypothesis is true versus situations that deviate substantially from the null hypothesis. We then calculate the probability of getting a statistic as extreme or more extreme than our observed statistic *when the null hypothesis is true*. This probability, our p-value, measures the degree of evidence against the null hypothesis.

1.10.1 Formal decision process

When used as part of a formal decision process, p-values get compared to a significance level, α; often this significance level is arbitrarily set to $\alpha = 0.05$. If the p-value is less than α, we reject the null hypothesis; otherwise, we *fail to reject* the null hypothesis. This decision process is often compared to decisions in a court of law (Figure 1.16), with 4 potential outcomes:

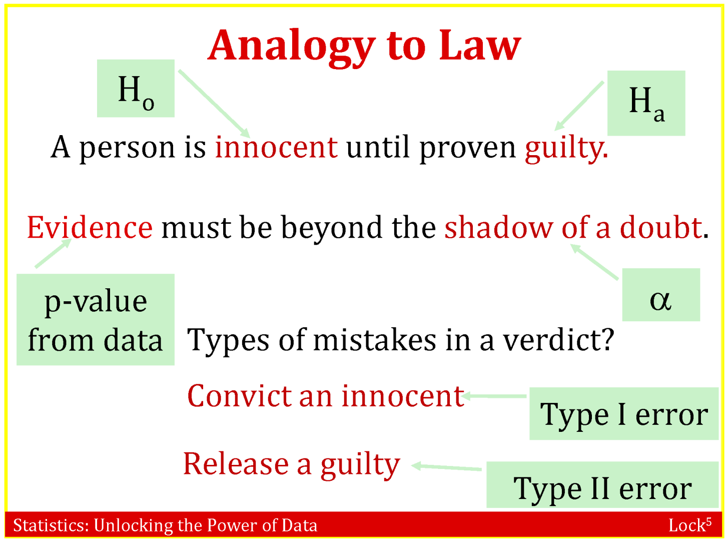
If the Null hypothesis is true, we may:

- Correctly fail to reject it.
- Reject it, leading to what is referred to as a type I error with probability α.

If the null hypothesis is false, we may:

- Correctly reject it.
- Fail to reject it, leading to a what is referred to as a type II error with probability β.

[8]For hypothesis tests that do not involve a population parameter (e.g., tests of Normality or goodness-of-fit tests), we may need to use other methods (e.g., simulations) to evaluate the extent to which deviations from the null Hypothesis impact our inferences.



Analogy to Law

A person is innocent until proven guilty.

Evidence must be beyond the shadow of a doubt.

Types of mistakes in a verdict?

Convict an innocent

Release a guilty

H_o
H_a
p-value from data
α
Type I error
Type II error
Statistics: Unlocking the Power of Data
Lock^5

FIGURE 1.16 Slide accompanying the textbook, R. H. Lock et al. (2020), that makes the analogy between Null Hypothesis testing and decisions in a court of law.

If you are familiar with the story, *A boy who cried wolf*, it may help with remembering the difference between type I and type II errors. In the story, a boy tasked with keeping watch over a town's flock of sheep amuses himself by yelling "Wolf!" and seeing the townspeople run despite knowing that there are no wolves present. Eventually, a wolf shows up and the townspeople ignore the boy's call when a wolf is actually present. If we consider the null hypothesis, H_0, that no wolf is present, the townspeople can be thought of as making several type I errors (falsely rejecting the null hypothesis) before committing a type II error (failing to reject the null hypothesis).

1.10.2 Issues with null hypothesis testing

Null hypothesis testing has been criticized for many good reasons (see e.g., D. H. Johnson 1999). Some of these include:

1. The arbitrary use of $\alpha = 0.05$, which is equivalent to the probability of making a type I error. The smaller the type I error (α), the larger the type II error (β). Given the tradeoff between α and β, it is important to consider both types of errors when attempting to draw conclusions from data. Ideally, a power analysis would be conducted prior to collecting any data. The power of a hypothesis test is the probability of rejecting the null hypothesis when the null hypothesis is false (i.e., power $= 1 - \beta$). The power of a test will depend on how far the truth deviates from the null hypothesis, the sample size, and the characteristics of the data and the test statistic. A power analysis uses assumptions and possibly prior data to determine how the power of a hypothesis test depends on these characteristics (e.g., so that the user can determine an appropriate sample size).
2. Related to [1], small data sets typically have low power (i.e., we are unlikely to reject the null hypothesis even when it is false). Too often researchers *accept* the Null hypothesis as being true when getting a p-value $> \alpha$ rather than reporting that they *failed to reject* it.
3. Similarly, large data sets will often lead us to reject the null hypothesis even when effect sizes are

small. Thus, we may get excited about a statistically significant result that is in essence not very interesting or meaningful.

4. Most null hypotheses are known to be false, so it makes little sense to test them. We don't need a hypothesis test to tell us that survival rates differ for adults and juveniles or across different years. We should instead ask how big the differences are and focus on estimating effect sizes and their uncertainty (e.g. using confidence intervals).

Given the widespread issues associated with how hypothesis tests are used in practice, it is extremely important to be able to understand what a p-value is and how to interpret the results of null hypothesis tests (De Valpine 2014). Many of the above issues can be solved by proper study design (e.g., having a large enough sample size to ensure power is higher for detecting meaningful differences) and by reporting effect sizes and confidence intervals along with p-values.

1.10.3 Exemplification: Testing whether the slope is different from 0

Here, we outline the basic steps of a null hypothesis test using the test for the slope of the least squares regression line.

Steps:

1. State the null and alternative hypotheses. The null hypothesis is $H_0 : \beta_1 = 0$. The alternative hypothesis is $H_A : \beta_1 \neq 0$. Or, if we have *a priori* expectations that $\beta_1 > 0$ or $\beta_1 < 0$, we might specify a "one-sided" alternative hypothesis, $H_A : \beta_1 > 0$ or $H_A : \beta_1 < 0$. For a discussion of when it might make sense to use a one-sided alternative hypothesis, see Ruxton and Neuhäuser (2010). If you use a one-sided test, you should be prepared to explain why you are not interested in detecting deviations from the null hypothesis in the other direction.
2. Calculate some statistic from the sample data that can be used to evaluate the strength of evidence against the null hypothesis. For testing $H_0 : \beta_1 = 0$, we will use $t = \frac{\hat{\beta}}{\widehat{SE}(\hat{\beta})}$ as our test statistic. This test statistic is reported in the `t stat` column of our regression output, $t = \frac{\hat{\beta}}{\widehat{SE}(\hat{\beta})} = \frac{8.9376}{1.4923} = 7.053$.
3. Determine the sampling distribution of the statistic in step 2, under the additional constraint that *the null hypothesis is true*. From Section 1.7, we know that:

$$t = \frac{\hat{\beta}_1 - \beta_1}{\widehat{SE}(\hat{\beta}_1)} \sim t_{n-2}$$

If the null hypothesis is true, $H_0 : \beta_1 = 0$, so:

$$t = \frac{\hat{\beta}_1 - 0}{\widehat{SE}(\hat{\beta}_1)} \sim t_{n-2}$$

4. Determine how likely we are to see a sample statistic that is as extreme or more extreme as the statistic we calculated in step 2, *if the null hypothesis is true.* The p-value is the area under the curve of the sampling distribution associated with these more extreme values. To calculate the p-value, we need to overlay our test statistic, $t = 7.053$ on the t_{30} distribution and determine the probability of getting a t-statistic as extreme or more extreme as 7.053 when the null hypothesis is true.

In this case, our t-statistic falls in the tail of the t-distribution – the probability of getting a t-statistic ≥ 7.03 or ≤ -7.03 is very close to 0 (so, the null hypothesis is unlikely to be true!). In Section 9, we will learn about

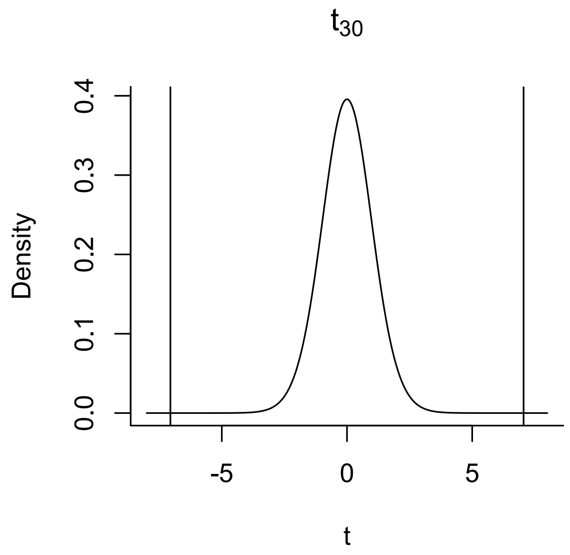


FIGURE 1.17 The p-value associated with the null hypothesis that $\beta_1 = 0$ versus $H_A : \beta_1 \neq 0$ is given by the area under the curve to the right of 7.053 and to the left of -7.053.

functions in R that will allow us to work with different probability distributions. We could calculate the exact p-value in this case as `2*pt(7.053, df=30, lower.tail=FALSE)` = 7.68e-08. Thus, we can reject the null hypothesis that $\beta_1 = 0$.

Still feel unsure about what a p-value measures? For another entertaining video from our AP stats teacher, this time discussing p-values, see Figure 1.18.

1.11 R^2: Amount of variance explained

Lastly, we might be interested in quantifying the amount of variability in our observed response data that is explained by the model. We can do this using the model's coefficient of determination or R^2. In the case of simple linear regression (models with a single variable), $R^2 = r^2$, where $r = cor(x, y)$. To understand how R^2 is calculated, we need to explore sums of squares. Specifically, we decompose the total sums of squares (SST), representing the overall variability in Y, in terms of sums of squares/variability explained by the model (SSR) and residual sums of squares (variability not explained by the model):

- SST (Total sum of squares) = $\sum_i^n (Y_i - \bar{Y})^2$
- SSE (Sum of Squares Error) = $\sum_i^n (Y_i - \hat{Y})^2$
- SSR = SST - SSE = $\sum_i^n (\hat{Y}_i - \bar{Y})^2$

$R^2 = \frac{SST-SSE}{SST} = \frac{SSR}{SST}$ = proportion of the variation explained by the linear model.

To explore these concepts further, you might explore the following shiny app:

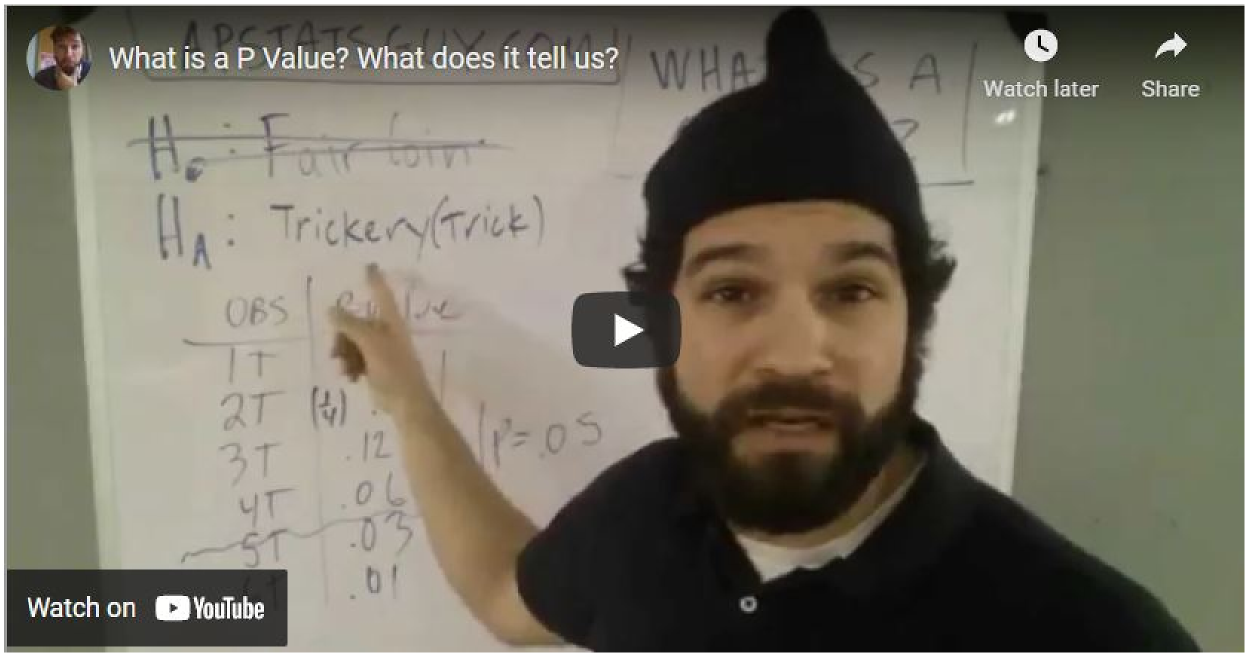


FIGURE 1.18 Explanation of p-values by Mr. Nystrom, an Advanced Placement Statistics Instructor.

https://paternogbc.shinyapps.io/SS_regression/

If you look closely at the summary of the linear model object, below, you will see that there are actually 2 different R^2 values reported, a `Multiple R-squared:  0.6238` and `Adjusted R-squared:  0.6113`. The R^2 above refers to the former. We will discuss differences between these two measures when we discuss multiple regression (see Section 3.4).

```
summary(lm.nose)
```

```
##
## Call:
## lm(formula = age ~ proportion.black, data = LionNoses)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5449 -1.1117 -0.5285  0.9635  4.3421
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.8790     0.5688   1.545    0.133
## proportion.black  10.6471     1.5095   7.053 7.68e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.669 on 30 degrees of freedom
## Multiple R-squared:  0.6238, Adjusted R-squared:  0.6113
## F-statistic: 49.75 on 1 and 30 DF,  p-value: 7.677e-08
```

1.12 Coded answers to exercises

1. Sampling Distribution of β_1(Section 1.6.3).
2. Sampling Distribution of t-statistic (Section 1.7).
3. Confidence Intervals (Section 1.8.1).

2

Bootstrapping

Learning objectives

1. To understand how a bootstrap can be used to quantify uncertainty, particularly when assumptions of linear regression may not be met.[1]
2. To further your coding skills by implementing a cluster-level bootstrap appropriate for certain types of data sets containing repeated observations.

Zuur et al. (2009) lead off their immensely popular book, Mixed Effects Models and Extensions in Ecology with R, with the statement (p. 19), "In Ecology, the data are seldom modeled adequately by linear regression models. If they are, you are lucky." Statistical methods, such as generalized linear models and random effects models, allow one to relax the assumptions of Normality, constant variance, and independence that constrain the applicability of linear regression. Most of this book will be devoted to these alternative methods. For now, however, we will explore how we can use a different tool, bootstrapping, to quantify uncertainty when the assumptions of linear regression do not hold. The bootstrap is also a useful approach for understanding the concept of a sampling distribution, because it requires us to repeatedly generate and analyze data sets to quantify how parameter estimates vary across multiple sampling events (see Fieberg, Vitense, and Johnson 2020 for further discussion and examples).

2.1 R Packages

In this section, we will use data and functions from the following packages:

- `ggplot` for plotting
- `abd` for the `LionNoses` data set
- `Data4Ecologists` for the `RiKZdat`data set
- `ggResidpanel` for residual plots

2.2 Motivating data example: RIKZ [Dutch governmental institute] data

We will consider a data set from Zuur et al. (2009) containing abundances of ~ 75 invertebrate species measured on various beaches along the Dutch Coast. The data were originally published in GM Janssen and Mulder (2004) and Gerard Janssen and Mulder (2005). The pictures, below, were sent to me by Professor Gerard Janssen who did some of the field sampling.

[1]There are several ways to use bootstrapping to quantify uncertainty associated with regression models. Here we will consider a specific method that uses case resampling.

FIGURE 2.1 Photos provided by Dr. Gerard Janssen who led the field sampling associated with the RIKZ data set.

The data, which can be accessed via the `Data4Ecologists` library, were collected at 5 stations at each of 9 beaches. The beaches were located at 3 different levels of exposure to waves (high, medium, or low). For now, we will focus on the following variables:

- `Richness` = species richness (i.e., the number of species counted).
- `NAP` = height of the sample site (relative to sea level). Lower values indicate sites closer to or further below sea level, which implies more time spent submerged.
- `Beach` = a unique identifier for the specific beach where the observation occurred

We will load the data, turn `Beach` into a factor variable, and plot `Richness` versus `NAP`, which suggests a line does a fairly good job of capturing the general pattern (Figure 2.2).

```
library(Data4Ecologists) # for data
library(ggplot2) # for plotting
theme_set(theme_bw()) # black and white background
data(RIKZdat)
RIKZdat$Beach <- as.factor(RIKZdat$Beach)
```

```
ggplot(RIKZdat, aes(NAP, Richness)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x,  se = FALSE) + xlab("NAP") +
  ylab("Richness")
```

We then use the `lm` function to determine the best-fit line.

```
lmfit.R <- lm(Richness ~ NAP, data = RIKZdat)
summary(lmfit.R)
```

```
Call:
lm(formula = Richness ~ NAP, data = RIKZdat)
```

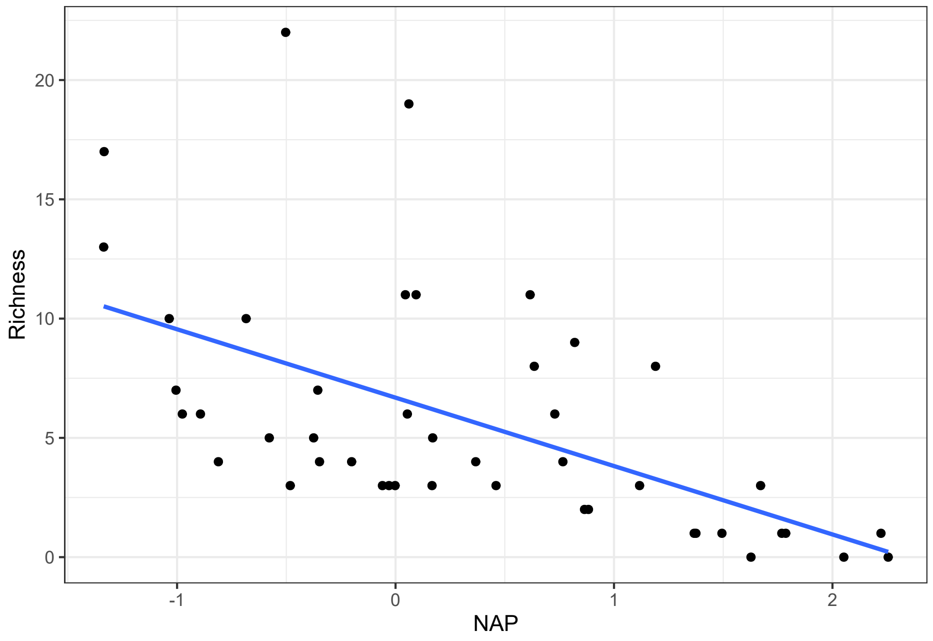


FIGURE 2.2 Species richness versus NAP for data collected from 9 beaches in the Netherlands.

```
Residuals:
    Min      1Q  Median      3Q     Max
-5.0675 -2.7607 -0.8029  1.3534 13.8723

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.6857     0.6578  10.164 5.25e-13 ***
NAP          -2.8669     0.6307  -4.545 4.42e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.16 on 43 degrees of freedom
Multiple R-squared:  0.3245,    Adjusted R-squared:  0.3088
F-statistic: 20.66 on 1 and 43 DF,  p-value: 4.418e-05
```

Before we trust the standard errors and p-values in the summary output, we should make sure our assumptions hold.

Recall, simple linear regression models can be described using the following equations:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

$$\epsilon_i \sim N(0, \sigma^2)$$

Think-pair-share: Which of the following assumptions do you think are met based on the output from the model?

Assumptions:

- **Homogeneity of variance** (constant scatter above and below the line); i.e., $Var(\epsilon_i) = \sigma^2$ (a constant)
- **Independence**: Correlation$(\epsilon_i, \epsilon_j) = 0$ for all $i \neq j$.
- **Linearity**: $E[Y_i \mid X] = \beta_0 + X_i\beta_1$
- **Normality**: ϵ_i come from a Normal (Gaussian) distribution

The answer is probably "none of the above!" Lets look at residual plots using the `resid_panel` function in the `ggResidPanel` package (Goode and Rey 2019):

```
ggResidpanel::resid_panel(lmfit.R)
```

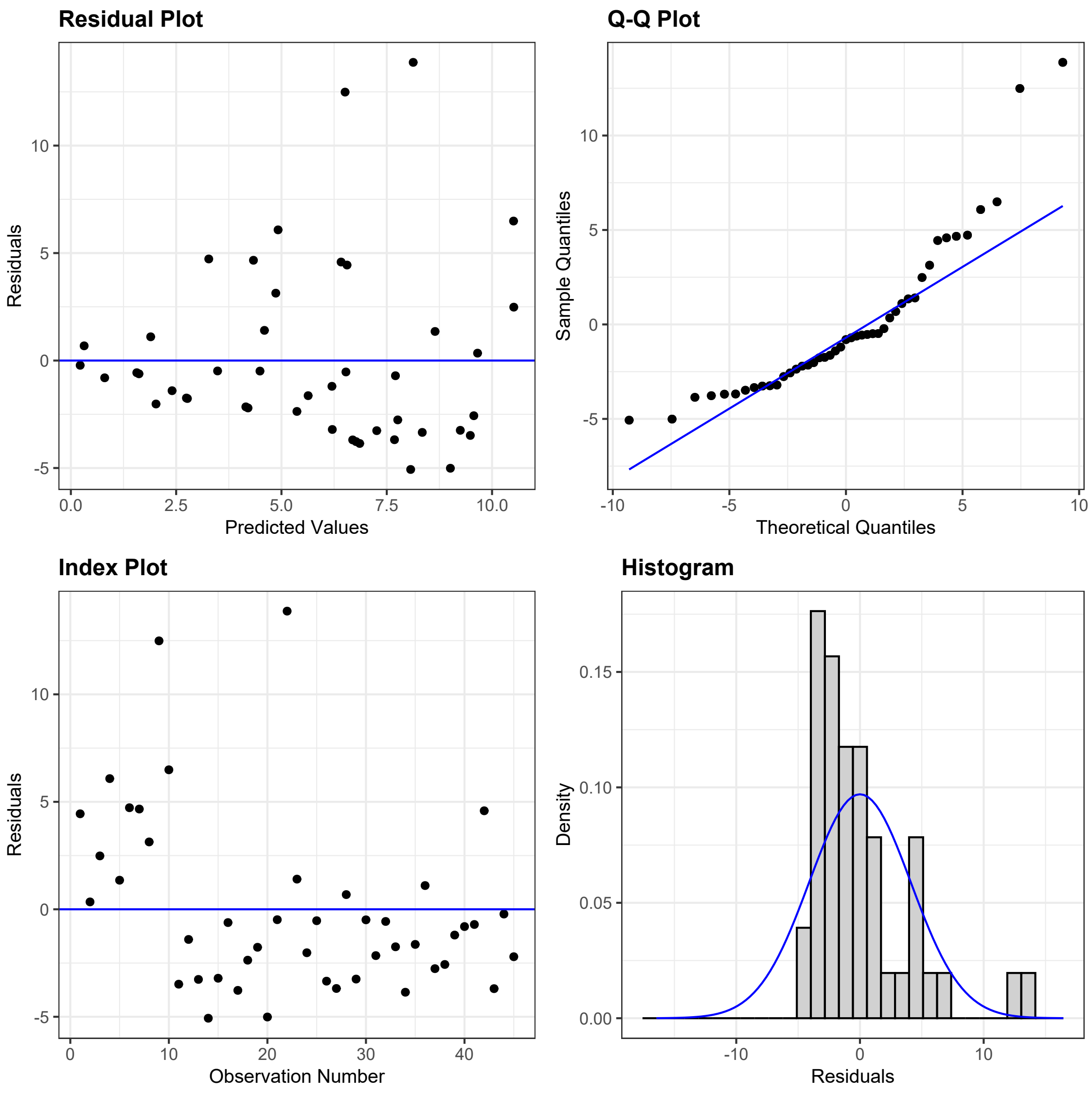


FIGURE 2.3 Residual plots for the linear regression model relating species `Richness` to `NAP` in the `RIKZdat` data set formed using the `resid_panel` function in the `ggResidpanel` package (Goode and Rey 2019).

Homogeneity of variance: the plot of residuals versus fitted values shows a clear increase in variance as we move from left to right (i.e., from smaller to larger predicted values).

Gaussian distribution: the histogram of the residuals suggests they are right-skewed rather than Normally distributed, and the Q-Q plot shows notable deviations from the Q-Q line.

Linearity: the average value of `Richness` at any given value of `NAP` looks like it is fairly well described by the line. But, what happens when NAP gets large? If we extrapolate outside of the range of the data, then our line will predict negative values of `Richness` which is clearly not possible.

Independence: observations are probably not independent since we collected five observations from each beach. Any two observations from the same beach are likely to be more alike than two observations from two different beaches (even after accounting for the effect of NAP). The index plot shows that the first ten observations (from beaches 1 and 2) all had positive residuals, whereas most of the residuals for the other three beaches are negative.

2.3 Consequences of assumption violations

The regression coefficients in a linear regression model are estimated by minimizing the sums of squared deviations between observations and the fitted line:

$\hat{\beta}$ minimizes: $\sum_{i=1}^{n}(Y_i - [\beta_0 + X_i\beta_1])^2$.

Least-squares can be motivated from just the *linearity* and *independence* assumptions. The *linearity* assumption seems reasonable over the range of the data, so as long as we do not extrapolate outside the range of our data, we may be OK with this assumption. But, what are the consequences of other assumption violations?

Homogeneity of variance: Least squares assigns equal weight to all observations. If our observations are independent, and our model for the mean is correct, then equal weighting of observations should not introduce bias into our estimates of β. However, other weighting schemes can result in more precise estimates (i.e., estimates that vary less from sample to sample). Approaches like *generalized least squares* (Section 5) and *generalized linear models* (Section 14) assign more weight to observations in regions of predictor space where the responses are less variable.

Normality: This assumption is needed to derive the sampling distribution of $\frac{\hat{\beta}}{SE(\hat{\beta})}$. Without this assumption, the sampling distribution of our regression parameter estimators does not follow a t-distribution, and thus, the p-values for the hypothesis tests associated with the intercept and slope may not be accurate. However, the Central Limit Theorem tells us that the sampling distribution of least-squares estimators will converge to that of a Normal distribution as our sample size moves towards infinity. Differences between the t-distribution and Normal distribution also become negligible as the the sample size increases. Therefore, violations of the Normality assumption may not be too consequential if samples sizes are large.

Independence: The main consequence of having non-independent data is that our SE's for $\hat{\beta}$ will typically be too small, confidence intervals too narrow, and p-values too small. We will often refer to groups that share unmeasured characteristics (e.g., observations from the same beach) as clusters. When data are naturally grouped or clustered in some way, we will have less information than we would have if we were able to collect an *equally-sized* data set in which all observations were independent.

Again, the rest of this book will be about developing model-based solutions to allow for non-Normal, clustered data.

If we conclude a linear model is still appropriate for describing the relationship between the average species `Richness` and `NAP`, then we can still use least squares to estimate β_0 and β_1; however, we need a different way to calculate our standard errors, calculate confidence intervals, and conduct hypothesis tests – we need a method that works well even when our data are not independent or Normally distributed. One option may be to use a bootstrap to quantify uncertainty.

2.4 Introduction to the Bootstrap

Let's return to the idea of a sampling distribution from Section 1. We can visualize the process of generating a sampling distribution from multiple samples of data, all of the same size and taken from the same population, and then calculating the same statistic from each data set (in our case, $\hat{\beta}$) - see Figure 2.4 from Fieberg, Vitense, and Johnson (2020).

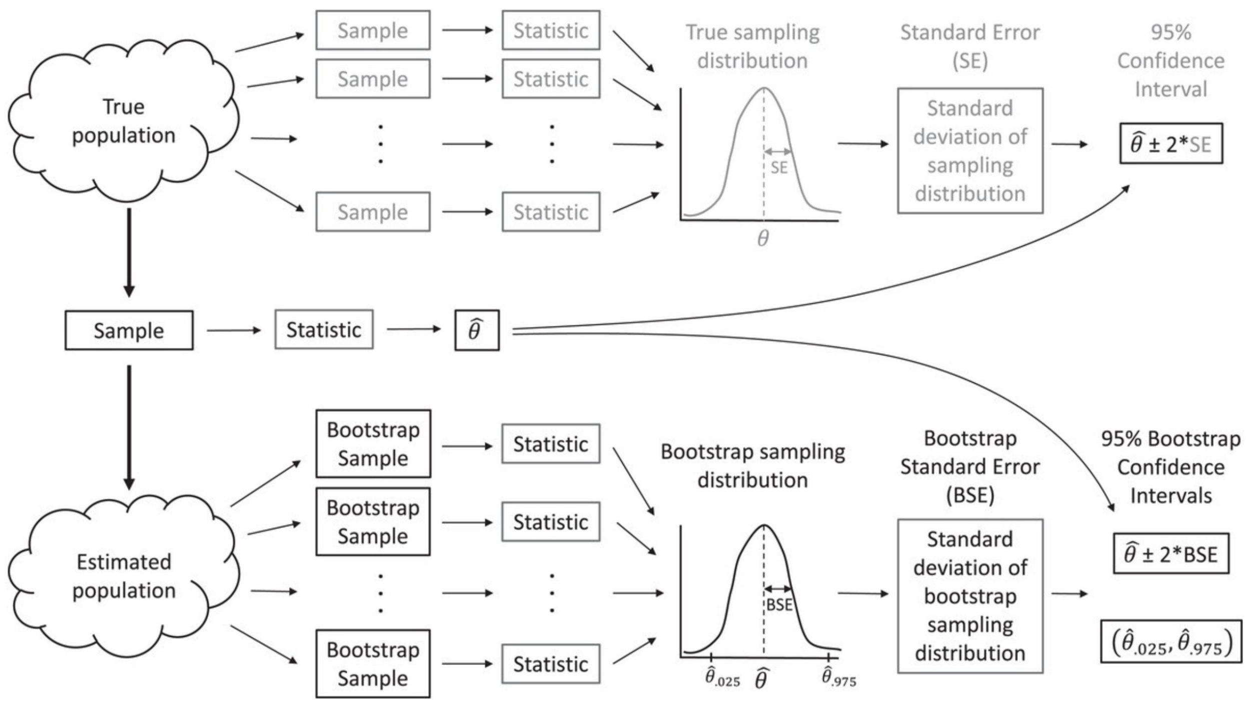

True population
Sample
Statistic
True sampling distribution
SE
θ
Standard Error (SE)
Standard deviation of sampling distribution
95% Confidence Interval
$\hat{\theta} \pm 2$*SE
Sample
Statistic
$\hat{\theta}$
Estimated population
Bootstrap Sample
Statistic
Bootstrap sampling distribution
BSE
$\hat{\theta}_{.025}$ $\hat{\theta}$ $\hat{\theta}_{.975}$
Bootstrap Standard Error (BSE)
Standard deviation of bootstrap sampling distribution
95% Bootstrap Confidence Intervals
$\hat{\theta} \pm 2$*BSE
$(\hat{\theta}_{.025}, \hat{\theta}_{.975})$

FIGURE 2.4 The bootstrap allows us to estimate characteristics of the sampling distribution (e.g., its standard deviation) by repeatedly sampling from an estimated population. Figure from Fieberg, J. R., Vitense, K., and Johnson, D. H. (2020). Resampling-based methods for biologists. PeerJ, 8, e9089. DOI: 10.7717/peerj.9089/fig-2.

If all of our assumptions hold, we know:

$$\frac{\hat{\beta} - \beta}{SE(\hat{\beta})} \sim t_{n-2}$$

However, this will not be the case if our assumptions do not hold. Furthermore, we cannot derive the sampling distribution like we did in Section 1 if we are uncomfortable with these assumptions about the true data-generating process (DGP). What do we mean by the data-generating process? The assumed data-generating process describes our assumptions about what the population looks like and also the mechanisms by which we observe a sample of observations from the population. In the linear model, the assumed DGP has the following characteristics:

- the values of Y_i at any given value of X_i are Normally distributed with mean $\mu_i = \beta_0 + \beta_1 X_i$ and variance σ^2.

- the sample values of Y_i and X_i that we observe are independent and representative of the larger population of values

If we are uncomfortable making these assumptions, what can we do? We might try changing our assumptions about the DGP. Maybe we drop the Normality assumption but keep the second assumption - i.e., that the data we have are independent and representative of the larger population of (X_i, Y_i) pairs. Or, perhaps we assume the *clusters* of observations from the different beaches are independent and representative of the clusters of observations that make up the population. This leads us to the concept of a non-parametric bootstrap.

Bootstrapping provides a way to estimate the SE of a statistic (in this case, $\hat{\beta}$) by resampling our original data with replacement. Essentially, we are using the distribution of values in our sample data as an estimate of the distribution of values in the population. Then, by 1) repeatedly resampling our data with replacement and 2) calculating the same statistic for each of these generated data sets, we can approximate the sampling distribution of our statistic under a DGP where the primary assumption is that our sample observations are independent and representative of the larger population.

2.4.1 Bootstrap procedure

If our sample data are representative of the population, as we generally assume when we have a large number of observations selected by simple random sampling, then we can use the distribution of values in our sample to approximate the distribution of values in the population. For example, we can make many copies of our sample data and use the resulting data set as an estimate of the whole population. With this estimated population in place, we could repeatedly sample from it, calculate the statistic for each of these sampled data sets, and then compute the standard deviation of these sample statistics to form our estimated standard error. In practice, we do not actually need to make multiple copies of our sample data to estimate the population; instead, we form new data sets by sampling our sample data with replacement, which effectively does the same thing. This means that each observation in the original data set can occur zero, one, two, or more times in the generated data set, whereas it occurred exactly once in the original data set.

This process allows us to determine how much our estimates vary across repeated samples. We quantify this variability using the standard deviation of our estimates across repeated samples and refer to this standard deviation as our bootstrap standard error (SE).

- A bootstrap sample is a random sample taken with replacement from the original sample, of the same size as the original sample – a "simulated" new sample, in a sense.
- A bootstrap statistic is the statistic of interest, such as the mean or slope of the least-squares regression line, computed from a bootstrap sample
- A bootstrap distribution is the distribution of bootstrap statistics derived from many bootstrap samples.

Whereas the sampling distribution will typically be centered on the population parameter, the bootstrap distribution will typically be centered on the sample statistic.[2] That is OK, what we really want from a bootstrap distribution is a measure of variability from sample to sample. The variability of the bootstrap statistics should be similar to the variability of sample statistics in a sampling distribution (if we could actually repeatedly sample from the population) – the spread of the bootstrap and sampling distributions should be very similar even if their centers are not. Thus, we can estimate the standard error using the standard deviation of the bootstrap distribution.

[2]When this is not the case, the bootstrap can be used to estimate estimator bias.

2.4.2 Bootstrap example: Simple linear regression

To demonstrate the bootstrap, let's revisit the `LionNoses` data set from Section 1. We will calculate a 95% confidence interval for the intercept and slope parameters using a bootstrap, which we will then compare to standard t-based confidence intervals for these parameters. To generate the bootstrap confidence interval, we will:

1. Generate a bootstrap data set by resampling our original observations with replacement.
2. Calculate our bootstrap statistics, $\hat{\beta}_0$ and $\hat{\beta}_1$, by fitting a linear model to our bootstrap data set.

We will repeat steps 1 and 2 a large number of times to form our bootstrap distribution, which we then plot (Figure 2.5).

```
set.seed(08182007)
library(abd)
data("LionNoses")
nboot <- 10000 # number of bootstrap samples
nobs <- nrow(LionNoses)
bootcoefs <- matrix(NA, nboot, 2)
for(i in 1:nboot){
  # Create bootstrap data set by sampling original observations w/ replacement
  bootdat <- LionNoses[sample(1:nobs, nobs, replace=TRUE),]
  # Calculate bootstrap statistic
  lmboot <- lm(age ~ proportion.black, data = bootdat)
  bootcoefs[i,] <- coef(lmboot)
}
par(mfrow = c(1, 2))
hist(bootcoefs[,1], main = expression(paste("Bootstrap distribution of ", hat(beta)[0])), xlab = "")
hist(bootcoefs[,2], main = expression(paste("Bootstrap distribution of ", hat(beta)[1])), xlab = "")
```

The bootstrap distributions are relatively bell-shaped and symmetric (Figure 2.5), so we could use a Normal approximation to generate confidence intervals for the intercept and slope parameters. To do so, we would calculate the bootstrap standard error as the standard deviation of the bootstrap statistics. We would then take our original sample statistics $\pm 1.96SE$. Alternatively, we could use percentiles of the bootstrap distribution to calculate a confidence interval. We demonstrate these two approaches, below, and compare them to the standard t-based interval (Figure 2.6).

```
# Fit the model to the original data to get our estimates
lmnoses <- lm(age ~ proportion.black, data = LionNoses)

# Calculate bootstrap standard errors
se<-apply(bootcoefs, 2, sd)

# Confidence intervals
# t-based
confdat.t <- confint(lmnoses)
# bootstrap normal
confdat.boot.norm <- rbind(c(coef(lmnoses)[1] - 1.96*se[1], coef(lmnoses)[1] + 1.96*se[1]),
                           c(coef(lmnoses)[2] - 1.96*se[2], coef(lmnoses)[2] + 1.96*se[2]))
# bootstrap percentile
confdat.boot.pct <- rbind(quantile(bootcoefs[,1], probs = c(0.025, 0.975)),
```

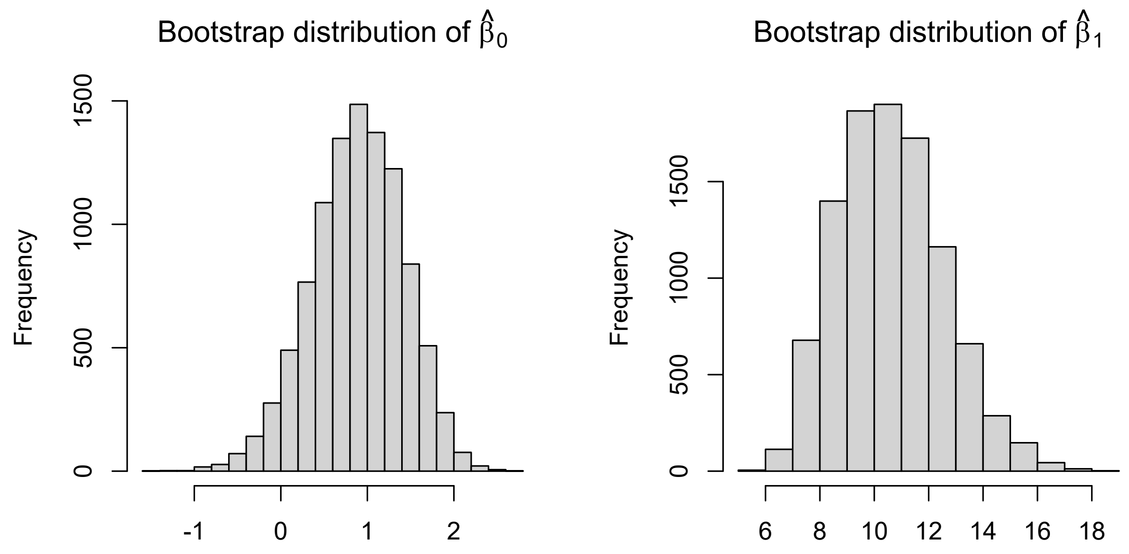


FIGURE 2.5 Bootstrap distribution of regression coefficients relating the age of lions to the proportion of their nose that is black.

```
                          quantile(bootcoefs[,2], probs = c(0.025, 0.975)))

# combine and plot
confdats <- rbind(confdat.t, confdat.boot.norm, confdat.boot.pct)
confdata <- data.frame(LCL = confdats[,1],
                       UCL = confdats[,2],
                       method = rep(c("t-based", "bootstrap-Normal", "bootstrap-percentile"),
                                    each=2),
                       parameter = rep(c("Intercept", "Slope"), 3))
confdata$estimate <- rep(coef(lmnoses),3)
ggplot(confdata, aes(y = estimate, x = " ", col = method)) +
  geom_point() +
  geom_pointrange(aes(ymin = LCL, ymax = UCL),  position = position_dodge(width = 0.9)) +
  facet_wrap(~parameter, scales="free") +xlab("")
```

For this particular example, we see that the bootstrap confidence intervals are slightly narrower than the t-based interval for the intercept and slightly wider for the slope. Thus, we pay a small price, in terms of precision, if we are unwilling to make certain assumptions about the DGP (e.g., the residuals are Normally distributed with constant variance) and go with the bootstrap confidence interval for the slope instead. For bell-shaped bootstrap distributions, Normal-based and percentile-based bootstrap confidence intervals often work well and will usually give similar results. For small samples or skewed bootstrap distributions, however, better methods exist and should be considered (see e.g., Davison and Hinkley 1997; Hesterberg 2015; Puth, Neuhäuser, and Ruxton 2015).

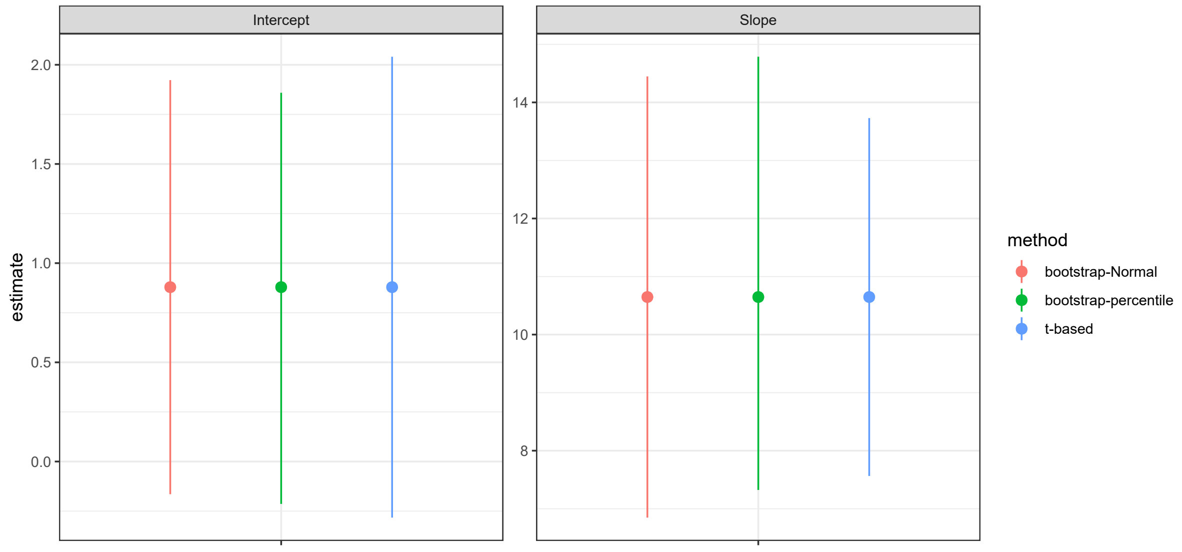


FIGURE 2.6 Comparison of t-based and bootstrap intervals for the intercept and slope of the regression relating the age of lions to the proportion of their nose that is black.

2.5 Replicating the sampling design

When we use a bootstrap, we are explicitly assuming that it captures the main features of the DGP that resulted in our original sample data. Thus, bootstrap sampling should mimic the original sampling design. What do we do if we have clustered data (i.e., multiple observations taken from the same sampling unit, such as a beach)? Assuming clusters are independent from one another (e.g., knowing the ϵ for an observation at beach 1 tells us nothing about the likely ϵ for an observation at beach 2), we can resample clusters with replacement (keeping all observations associated with the cluster). The number of clusters in the bootstrap sample should be the same as in the original sample (Figure 2.7).

Important caveats: The cluster-level bootstrap that is described above works best when we have balanced data (all clusters have the same number of observations; in this case we have five observations at each beach). This approach may be sub-optimal when applied to unbalanced data and potentially problematic depending on the mechanisms causing variability in sample sizes among clusters (e.g., if the size of the cluster is in some way related to the response of interest; Williamson 2014). In addition, this approach makes the most sense when we are interested in the effect of predictor variables that do not vary within a cluster (Fieberg, Vitense, and Johnson 2020). One of the exercises in the companion exercise book asks you to implement this approach by resampling beaches with replacement. Example code for generating a single bootstrap sample in this case is given below.

```
# let's use the bootstrap to deal with non-constant variance and clustering
# Determine the number of unique beaches
uid<-unique(RIKZdat$Beach)
nBeach<-length(uid)

# Single bootstrap, draw nBeaches with replacement from the
```

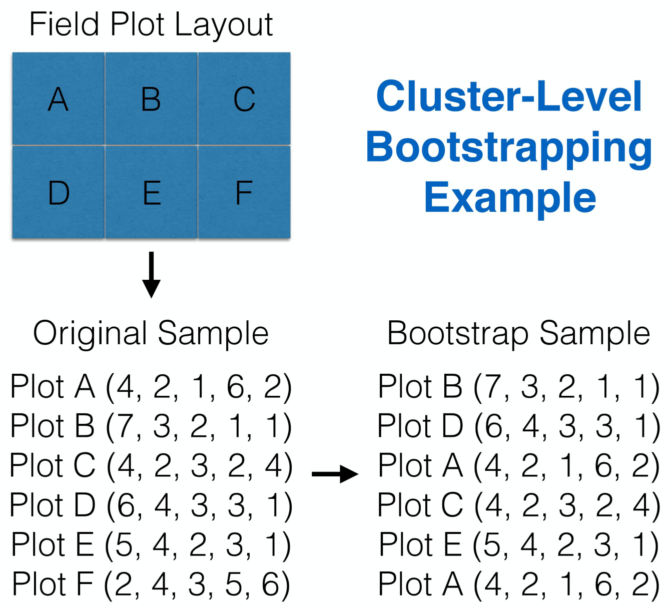


FIGURE 2.7 Cluster-level bootstrapping resamples whole clusters of observations with replacement. Figure constructed by Althea Archer, a former postdoc in the Fieberg lab.

```r
# original pool of beaches
(bootids<-data.frame(Beach=sample(uid, nBeach, replace=T)))
```

```
  Beach
1     3
2     7
3     7
4     1
5     3
6     5
7     4
8     2
9     2
```

```r
bootdat<-merge(bootids, RIKZdat)
# Verify this worked
table(bootdat$Beach)
```

```

 1  2  3  4  5  6  7  8  9
 5 10 10  5  5  0 10  0  0
```

2.6 Utility of the bootstrap

There are 2 primary reasons why you may want to consider a bootstrap. The first is that you may want to relax common assumptions about the underlying GDP, particularly when assumptions of common statistical methods (e.g., Normally, constant variance) do not seem appropriate. The second reason is that you may be interested in estimating uncertainty associated with a parameter or a function of one or more parameters for which an analytical expression for the standard error is not available. To highlight this latter point, consider that analytical expressions for the standard error are available for many estimators of the parameters you encounter in an introductory statistics class:

- the sample mean as an estimator of the population mean: $\text{SE}(\overline{x}) = \frac{\sigma}{\sqrt{n}}$
- the sample proportion as an estimator of the population proportion: $\text{SE}(\hat{p}) = \sqrt{\frac{p(1-p)}{n}}$
- the slope of the least-squares regression line: $\text{SE}(\hat{\beta}_1) = \frac{\sqrt{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2/(n-2)}}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}}$

But, what if you want to estimate the uncertainty with $1/\beta_1$ or the ratio of two means or medians? Or, you want to fit a mechanistic model of growth to weight-at-age data (see Section 10.7), and need to calculate uncertainty with the predicted weight at any given age, which is given by a non-linear function of your parameters $(L_\infty, K, t_0) = L\infty\left(1 - e^{-K(Age-t_0)}\right)$? You will not find an analytical expression for the standard error in either of these cases, but you can always use the bootstrap to estimate a confidence interval. Two other interesting applications of the bootstrap include estimating model-selection uncertainty (Section 8.8.2) and uncertainty associated with clades on a phylogenetic tree (Felsenstein 1985 ; Efron, Halloran, and Holmes 1996).

Later, we will see other options for calculating uncertainty intervals for parameters and functions of parameters estimated using maximum likelihood (Section 10.6, Section 10.8.1, Section 10.10) or using Bayesian methods of inference (Section 11). It is extremely useful to become comfortable with at least one of these approaches (bootstrapping, delta method, or Bayesian credible intervals).

3

Multiple regression

Learning Objectives

1. Understand how to specify regression models using **matrix notation**.
2. Become familiar with creating **dummy variables** to code for categorical predictors.
3. **Interpret** the results of regression analyses that include both categorical and quantitative variables.
4. Understand approaches for visualizing the results of multiple regression models.

3.1 R Packages

We begin by loading a few packages upfront:

```
library(kableExtra) # for creating tables
options(kableExtra.html.bsTable = T)
library(tidyverse) # for data wrangling and plotting via ggplot2
library(rgl) # for 3D interactive graphics
library(ggeffects) # for effect plots summarizing fitted regression models
library(modelsummary) # for creating tables summarizing regression output
library(ggthemes) # for colorblind palette
theme_set(theme_bw()) # black and white background
```

In addition, we will use data and functions from the following packages:

- `Data4Ecologists` for the `RiKZdat`data set and the `partialr` data set.
- `ggformula` for creating a side-by-side histogram with Normal distribution overlaid
- `emmeans` for pairwise comparisons of means for different levels of a categorical variable
- `car` for partial residual and added-variable plots and F-tests involving regression coefficients
- `patchwork` for creating multi-panel plots

3.2 Introduction to multiple regression

So far, we have considered regression models containing a single predictor, often referred to as simple linear regression models. In this section, we will consider models that contain more than one predictor. We can again write our model of the data generating process (DGP) in two ways:

$$Y_i = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \ldots + \beta_p X_{i,p} + \epsilon_i \quad (i = 1, \ldots, n)$$
$$\epsilon_i \sim N(0, \sigma^2)$$

Or:

$$Y_i \sim N(\mu_i, \sigma^2)$$
$$\mu_i = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \ldots + \beta_p X_{i,p} \quad (i = 1, \ldots, n)$$

The above expressions define linear regression models in terms of individual observations. It will also be advantageous, at times, to be able to define the model for all observations simultaneously using matrices. Doing so will provide insights into how models are represented in statistical software, including models that allow for non-linear relationships between predictor and response variables (Section 4). In addition, matrix notation will provide us with a precise language for describing uncertainty associated with the predictions of linear models. Use of matrix notation will be important for:

- understanding methods for calculating uncertainty in $\hat{\mu}_i = \hat{\beta}_0 + \hat{\beta}_1 X_{i,1} + \hat{\beta}_2 X_{i,2} + \ldots \hat{\beta}_p X_{i,p}$ (we glossed over this in Section 1.9 where we showed code for calculating confidence and prediction intervals in regression but did not provide equations or derive expressions for these interval estimators).
- specifying models for observations that are not independent (e.g., Section 5 and Section 18).

3.3 Matrix notation for regression

Let's start by writing our linear regression model as:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i \quad (i = 1, \ldots, n)$$
$$\epsilon_i \sim N(0, \sigma^2)$$

This implies:

$$\begin{aligned} Y_1 &= \beta_0 + \beta_1 X_1 + \epsilon_1 \\ Y_2 &= \beta_0 + \beta_1 X_2 + \epsilon_2 \\ &\vdots \\ Y_n &= \beta_0 + \beta_1 X_n + \epsilon_n \end{aligned}$$

Alternatively, We can write this set of equations very compactly using matrices:

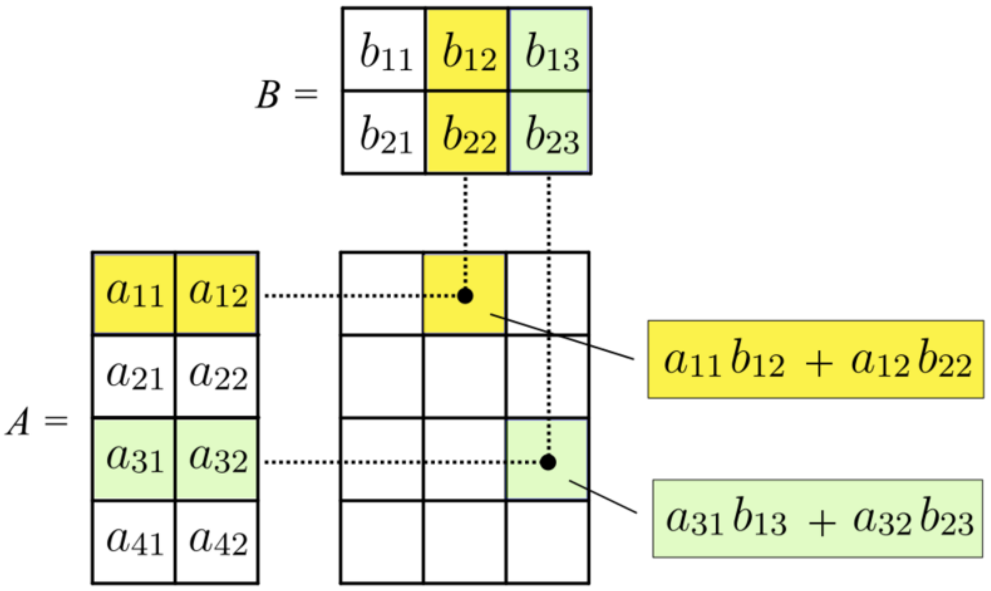
$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 & X_1 \\ 1 & X_2 \\ \vdots & \vdots \\ 1 & X_n \end{bmatrix} \times \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

or

$$Y_{[n\times 1]} = X_{[n\times 2]}\beta_{[2\times 1]} + \epsilon_{[n\times 1]}$$

where the subscript gives the dimension (rows $\times$ colums) in each matrix.

We can multiply two matrices, A and B, together if the number of columns of the first matrix is equal to the number of rows in the second matrix. Let the dimensions of A be $n \times m$ and the dimension of B be $m \times p$. Matrix multiplication results in a new matrix with the number of rows equal to the number of rows in A and number of columns equal to the number of columns in B, i.e., the matrix will be of dimension $n \times p$. The (i, j) entry of this matrix is formed by taking the dot product of row i and column j, where the dot product is the sum of element-wise products (Figure 3.1).



$B =$
b_{11} b_{12} b_{13}
b_{21} b_{22} b_{23}
$A =$
a_{11} a_{12}
a_{21} a_{22}
a_{31} a_{32}
a_{41} a_{42}
$a_{11}b_{12} + a_{12}b_{22}$
$a_{31}b_{13} + a_{32}b_{23}$

FIGURE 3.1 Matrix multiplication by Svjo – CC BY-SA 4.0.. If we multiply matrices $A_{4\times 2}$ and $B_{2\times 3}$. we end up with a new matrix that is of dimension 4×3 with the (i, j) entry of this matrix formed by taking the dot product of row i and column j.

Using matrix notation, we can generalize our model to include any number of predictors (shown below for $p - 1$ predictors):

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 & X_{11} & X_{12} & \cdots & X_{1,p-1} \\ 1 & X_{21} & X_{22} & \cdots & X_{2,p-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & X_{n1} & X_{n2} & \cdots & X_{n,p-1} \end{bmatrix} \times \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{p-1} \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

$$Y_{[n\times 1]} = X_{[n\times p]}\beta_{[p\times 1]} + \epsilon_{[n\times 1]}$$

The matrix, X, is referred to as the design matrix and encodes all of the information present in our predictors. In this chapter, we will learn how categorical variables and interactions are represented in the design matrix. In Chapter 4, we will learn how various methods for modeling non-linearities in the relationship between our predictors and our dependent variable can be represented in a design matrix.

We can, as an alternative, use the following equation with matrix notation to describe our linear regression model:

$$Y \sim N(X\beta, \Sigma) \tag{3.1}$$

Here, Σ is an $n \times n$ variance-covariance matrix. Its diagonal elements capture the variances of the observations and the off-diagonal elements capture covariances. Under the assumptions of linear regression, our observations are independent (implying the covariances are 0) and have constant variance, σ^2. Thus:

$$\Sigma_{n\times n} = \begin{bmatrix} \sigma^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma^2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma^2 \end{bmatrix} = \sigma^2 I_{[n\times n]}, \text{ where } I_{[n\times n]} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

.

3.4 Parameter estimation, sums-of-squares, and R^2

We can use `lm` to find estimates of intercept (β_0) and slope parameters ($\beta_1, \beta_2, \ldots$) that minimize the sum of squared differences between our observations and the model predictions:

$\sum_i^n (Y_i - \hat{Y}_i)^2 = \sum_i^n (Y_i - [\hat{\beta}_0 + \hat{\beta}_1 X_{1,i} + \hat{\beta}_2 X_{2,i} + \ldots])^2$

Further, we can again decompose the total variation, quantified by the total sums of squares (SST), into variation explained by the model (SSR) and residual sums of squares (SSE):

- $SST_{df=n-1} = \sum_i^n (Y_i - \bar{Y})^2$
- $SSE_{df=n-p} = \sum_i^n (Y_i - \hat{Y})^2$
- $SSR_{df=p-1} = SST - SSE = \sum_i^n (\hat{Y}_i - \bar{Y})^2$

The Coefficient of Determination (R^2) is calculated the same way as with simple linear regression ($R^2 = SSR/SST$). However, adding additional predictors *always* increases R^2. Thus, we will eventually also want to consider an **adjusted** R^2, quantified as:

$$R^2_{\text{adj}} = \frac{\frac{\text{SSR}}{p-1}}{\frac{SST}{n-1}} = \left(\frac{n-1}{p-1}\right)\frac{\text{SSR}}{\text{SST}}$$

The adjusted-R^2 penalizes for additional predictors and so will not always increase as you add predictors. Thus, it should provide a more honest measure of variance explained, particularly when the model contains many predictors. We will consider this measure in more detail when we compare the fits of multiple competing models (Section 8).

We have the same assumptions (linearity, constant variance, Normality) as we do with simple linear regression and can use the same diagnostic plots to evaluate whether these assumptions are reasonably met. However, it's also important to diagnose the degree to which explanatory variables are correlated with each other (a topic we will address in more detail when we cover multicollinearity in Section 6).

TABLE 3.1 Estimates of regression parameters (SE) for regression parameters in one and two-variable models fit to the RIKZ data.

	(1)	(2)
(Intercept)	6.686 (0.658)	5.459 (0.830)
NAP	−2.867 (0.631)	−2.512 (0.623)
humus		21.942 (9.710)
R2	0.325	0.398
R2 Adj.	0.309	0.369

3.5 Parameter interpretation: Multiple regression with RIKZ data

Recall the RIKZ data from Section 2. We will continue to explore this data set, assuming (naively) that assumptions of linear regression hold. Earlier, we fit a model relating species `Richness` to the height of the sample site relative to sea level, `NAP`. From our regression results, we can write our estimate of the best-fit line as:

$$Richness_i = 6.886 - 2.867NAP_i + \epsilon_i$$

What if we also hypothesized that **humus** (amount of organic material) also influences `Richness` (*in addition to* `NAP`)? The multiple linear regression model formula would look like:

$$Richness_i = \beta_0 + \beta_1 NAP_i + \beta_2 Humas_i + \epsilon_i$$

Let's fit this model in R and compare it to the model containing only `NAP` (Table 3.1.

```
library(Data4Ecologists)
data(RIKZdat)
lmfit1 <- lm(Richness ~ NAP, data = RIKZdat)
lmfit2 <- lm(Richness ~ NAP + humus, data = RIKZdat)
```

```
# Create table comparing the models
modelsummary(list(lmfit1, lmfit2),
  gof_omit = "^(?!R2)",
  estimate = "{estimate} ({std.error})",
  statistic = NULL,
  title = "Estimates of regression parameters (SE) for regression parameters
           in one and two-variable models fit to the RIKZ data.")
```

We see that the slope for `NAP` changed slightly (from -2.9 to -2.5) and the adjusted R^2 went from 0.31 to 0.37. Instead of a best-fit line through data in two dimensions, we now have a best fit plane through data in three dimensions (Figure 3.2).

Our interpretation of regression parameters is similar to that in simple linear regression, except now we have to consider a change in one variable while *holding other variables in the model constant*:

- β_1 describes the change in `Richness` for every 1 unit increase in `NAP` *while holding* ***`Humus`*** *constant.*

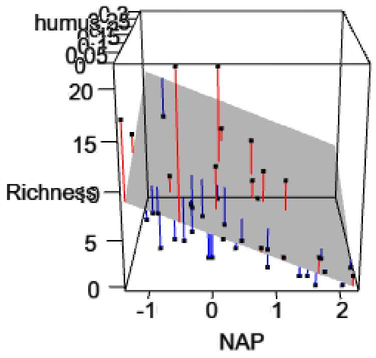


FIGURE 3.2 Visualizing the regression model with 2 predictors. Instead of a best-fit line, we have a best-fit plane. Code was adapted from https://stackoverflow.com/questions/47344850/scatterplot3d-regression-plane-with-residuals.

- β_2 describes the change in `Richness` for every 1 unit increase in `Humus` *while holding `NAP` constant*.
- β_0: the level of `Richness` if `Humus` and `NAP` are both simultaneously equal 0.

Although it is easy to fit multiple regression models with more than two predictors, we will no longer be able to visualize the fitted model in higher dimensions. Before we consider more complex models, however, we will first explore how to incorporate categorical predictors into our models.

3.6 Categorical predictors

To understand how categorical predictors are coded in regression models, we will begin by making a connection between the standard t-test and a linear regression model with a categorical predictor taking on one of two values.

3.6.1 T-test as a regression

Here, we will consider mandible lengths (in mm) of 10 male and 10 female golden jackal (*Canis aureus*) specimens from the British Museum (Manly 1991).

```
males<-c(120, 107, 110, 116, 114, 111, 113, 117, 114, 112)
females<-c(110, 111, 107, 108, 110, 105, 107, 106, 111, 111)
```

We might ask: Do males and females have, on average, different mandible lengths? Let's consider a formal hypothesis test and confidence interval for the difference in population means:

$$H_0 : \mu_m = \mu_f \text{ versus } H_a : \mu_m \neq \mu_f \tag{3.2}$$

FIGURE 3.3 Golden jackals (*Canis aureus*) in the Danube Delta, Romania. Image by Andrei Prodan from Pixabay.

where μ_m and μ_f represent population means for male and female jackals, respectively.

If we assume that mandible lengths are Normally distributed in the population[1], and that male and female jaw lengths are equally variable[2], then we can use the following code to conduct a t-test for a difference in means:

```
t.test(males, females, var.equal = T)
```

```

	Two Sample t-test

data:  males and females
t = 3.4843, df = 18, p-value = 0.002647
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.905773 7.694227
sample estimates:
mean of x mean of y 
    113.4     108.6 
```

We can also conduct this same test using a regression model with `sex` as the only predictor. First, we will have to create a data.frame with mandible lengths (quantitative) and `sex` (categorical).

```
jawdat <- data.frame(jaws = c(males, females),
                     sex = c(rep("M",10), rep("F", 10)))
head(jawdat)
```

```
  jaws sex
1  120   M
2  107   M
```

[1]Note: the normality assumption is required for small data sets, but the Central Limit Theorem (CLT) guarantees that sampling distribution for a difference in sample means will be approximately Normally distributed for large samples; a common rule is that we need roughly 30 observations in both groups for the CLT to apply

[2]Note: there are other variations on the t-test that could be applied if the variances of the two groups are not assumed to be equal

```
3  110   M
4  116   M
5  114   M
6  111   M
```

We can then fit a linear regression model to these data and inspect the output:

```
lm.jaw<-lm(jaws ~ sex, data = jawdat)
summary(lm.jaw)
```

```

Call:
lm(formula = jaws ~ sex, data = jawdat)

Residuals:
   Min     1Q Median     3Q    Max 
  -6.4   -1.8    0.1    2.4    6.6 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 108.6000     0.9741 111.486  < 2e-16 ***
sexM          4.8000     1.3776   3.484  0.00265 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.08 on 18 degrees of freedom
Multiple R-squared:  0.4028,    Adjusted R-squared:  0.3696 
F-statistic: 12.14 on 1 and 18 DF,  p-value: 0.002647
```

We see that the `t value` and `Pr(>|t|)` values for `sexM` are identical to the t-statistic and p-value from our two-sample t-test. Also, the estimated intercept is identical to the sample mean for females. And, if we look at a confidence interval for the regression parameters, we see that the interval for the `sexM` coefficient is the same as the confidence interval for the difference in means that is output by the `t.test` function (and, in fact, the coefficient for `sexM` is equal to the difference in sample means).

```
confint(lm.jaw)
```

```
                 2.5 %     97.5 %
(Intercept) 106.553472 110.646528
sexM          1.905773   7.694227
```

To understand these results, we need to know how R accounts for `sex` in the model we just fit.

3.6.2 Dummy variables: Reference (or effects) coding

We can use the `model.matrix` function to see the design matrix that R uses to fit the regression model. Here, we print the 2nd, 3rd, 16th, and 17th rows of this matrix (so that we see observations from both sexes):

```
model.matrix(lm.jaw)[c(2, 3, 16, 17), ]
```

```
##    (Intercept) sexM
## 2            1    1
## 3            1    1
## 16           1    0
## 17           1    0
```

Let's also look at the data from these cases:

```
jawdat[c(2, 3, 16, 17),]
```

```
##    jaws sex
## 2   107   M
## 3   110   M
## 16  105   F
## 17  107   F
```

We see that R created a new variable, `sexM`, to indicate which cases are males (`sexM` = 1) and which are females (`sexM` = 0). Knowing this allows us to write our model in matrix notation (shown here for these 4 observations):

$$\begin{bmatrix} 107 \\ 105 \\ 107 \\ 107 \\ 106 \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \end{bmatrix} \times \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \vdots \end{bmatrix}$$

Think-Pair-Share: How can we estimate μ_m, the mean jaw length for males, from the fitted regression model?

To answer this question, let's write our model as:

$$Y_i = \beta_0 + \beta_1 I(\text{sex=male})_i + \epsilon_i, \text{ with}$$

$$I(\text{sex=male})_i = \begin{cases} 1 & \text{if individual i is a male} \\ 0 & \text{if individual i is a female} \end{cases}$$

Here, I have used the notation I(*condition*) to indicate that we want to create a variable that is equal to 1 when *condition* is `TRUE` and 0 otherwise. In general, we will refer to this type of variable as an indicator variable or, more commonly, a dummy variable.

Using this model description, we can estimate the mean jaw length of males by plugging in a 1 for $I(\text{sex=male})_i$:

$$E[Y_i|\text{sex} = \text{male}] = \beta_0 + \beta_1(1)$$

I.e., we can estimate the mean jaw length of males by summing the two regression coefficients:

$$E[Y_i|\text{sex} = \text{male}] = \hat{Y}_i = 108.6 + 4.8 = 113.4,$$

which is the mean for males that is reported by the `t.test` function.

In summary, the default method used to account for categorical variables in R is to use reference coding, sometimes referred to as effects or treatments coding, such that:

- the intercept represents the mean for a *reference category* (when all other predictors in the model are set to 0); in the above example, females serve as the reference category.
- dummy or indicator variables represent differences in means between other categories and the reference category.

3.6.3 Dummy variables: Cell means coding

It turns out that there are other ways to code the same information contained in the variable `sex`, and these different parameterizations lead to the exact same model but expressed with a different set of coefficients (for an overview, see Schad et al. 2020). In this section, we will consider what is often called cell-means or means coding:

$$Y_i = X_{i,m}\beta_m + X_{i,f}\beta_f + \epsilon_i$$

$$X_{i,m} = \begin{cases} 1 & \text{if the } i^{th} \text{ observation is from a male} \\ 0 & \text{otherwise} \end{cases}$$

$$X_{i,f} = \begin{cases} 1 & \text{if the } i^{th} \text{ observation is from a female} \\ 0 & \text{otherwise} \end{cases}$$

Although male and female labels do not represent everyone's experience or identity, and more inclusive categories should be considered in human-subjects research, each jackal will be either male or female. Thus, either $X_{i,f}$ or $X_{i,m}$ will be 1 and the other will be 0, and β_m will represent the mean Y for males and β_f the mean Y for females.

In R, we can fit this model using means coding using:

```
lm.jaws.means <- lm(jaws ~ sex - 1, data = jawdat)
summary(lm.jaws.means)
```

```

Call:
lm(formula = jaws ~ sex - 1, data = jawdat)

Residuals:
   Min     1Q Median     3Q    Max 
  -6.4   -1.8    0.1    2.4    6.6 

Coefficients:
     Estimate Std. Error t value Pr(>|t|)    
sexF 108.6000     0.9741   111.5   <2e-16 ***
sexM 113.4000     0.9741   116.4   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.08 on 18 degrees of freedom
```

```
Multiple R-squared:  0.9993,	Adjusted R-squared:  0.9992
F-statistic: 1.299e+04 on 2 and 18 DF,  p-value: < 2.2e-16
```

The `-1` here tells R to remove the column of 1s in our design matrix (which is otherwise used to represent the intercept). With means coding, our model is parameterized in terms of the two group means rather than using the mean for females and the difference in means between males and females. Note: we cannot have a model with all of these parameters (μ_f, μ_m, and $\mu_m - \mu_f$) since one of these is completely determined by the other two. The `-1` in the formula (`jaws ~ sex - 1`) tells R not to include an overall intercept, which permits estimation of the second group mean in its place.

3.6.4 Comparing assumptions: Linear model and t-test

What are the assumptions of our model for the jaw lengths? Well, they are the same ones that we have for fitting linear regression models with continuous predictors:

- constant variance of the errors (i.e., the two groups are assumed to have equal variance, σ^2)
- the residuals, and by extension, the data within each group, are Normally distributed

These are the same assumptions of the two-sample t-test. Let's see if they are reasonable by creating side-by-side histograms with a Normal density overlaid (Figure 3.4). This plot is relatively easy to create using the `ggformula` package (Daniel Kaplan and Pruim 2021).

```r
library(ggformula)
gf_dhistogram(~ jaws | sex, data = jawdat) %>% gf_fitdistr(dist = "dnorm", col = "red")
```

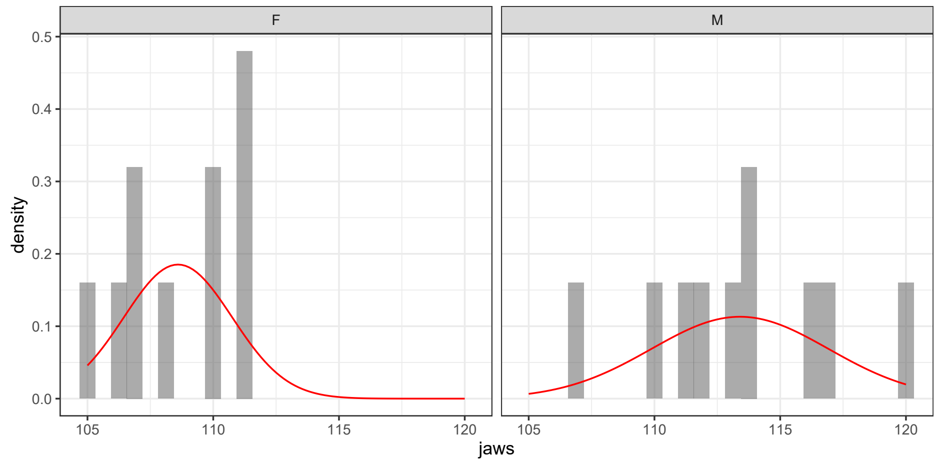


FIGURE 3.4 Evaluating the Normality and constant variance assumptions associated with golden jackal jaw-length data from the British Museum (Manly 1991).

We have a small data set, so it is difficult to say anything definitively, but it appears that the variance may be larger for males. Later, we will see how we can relax these assumptions using the `gls` function in the `nlme` package (see Section 5) and using JAGS (Section 11)

3.7 Categorical variables with > 2 levels or categories

To account for differences among k groups of a categorical variable, we can again use either reference/effects coding or means coding.

3.7.1 Effects coding

With effects coding, we include an overall intercept and $k - 1$ dummy variables. Each dummy variable is used to identify group membership for one of the $k - 1$ groups other than the reference; membership in the k^{th} group is indicated by having 0's for all of the other $k - 1$ dummy variables. The intercept will again represent a *reference group*, and the coefficients for the dummy variables will represent differences between each of the other $k - 1$ categories and the reference group. R will create these dummy variables for us, but understanding how the effects of categorical variables are encoded in regression models will facilitate parameter interpretation and allow us to fit customized models when desired (e.g., see Section 3.8.3). In addition, we will need to create our own dummy variables when fitting models in a Bayesian framework using JAGS (Section 12.5).

Let's return to the RIKZ data and our model of species `Richness`. What if we suspected that some species were only present in some weeks such that `Richness` varied by week *in addition to* `NAP`? Let's see what happens if we add `week` to the model:

```
lm.week <- lm(Richness ~ NAP + week, data = RIKZdat)
summary(lm.week)
```

```

Call:
lm(formula = Richness ~ NAP + week, data = RIKZdat)

Residuals:
    Min      1Q  Median      3Q     Max
-5.2493 -2.4558 -0.7746  1.4261 15.7933

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   9.0635     1.6291   5.563 1.68e-06 ***
NAP          -2.6644     0.6327  -4.211 0.000131 ***
week         -1.0492     0.6599  -1.590 0.119312
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.088 on 42 degrees of freedom
Multiple R-squared:  0.3629,    Adjusted R-squared:  0.3326
F-statistic: 11.96 on 2 and 42 DF,  p-value: 7.734e-05
```

Since `week` is coded as an integer (equal to 1, 2, 3 or 4), we see that R assumes it is a continuous predictor. Thus, R estimates a single coefficient representing the overall linear trend (slope) in `Richness` over time. Specifically, the model suggests we will lose roughly 1 species each week ($\hat{\beta} = -1.05$). Although this model can account for a linear increase or decrease in species `Richness` during the duration of the sampling effort, it will likely be better to model `week` as a categorical variable to allow for greater flexibility in how species richness changes over time. Doing so will require 3 dummy variables because there are $k = 4$ levels in our `week` categorical variable.

In R, we can use `as.factor` to convert `week` to a categorical variable and then refit the model:

```
RIKZdat <- RIKZdat %>% mutate(week.cat = as.factor(week))
lm.ancova <- lm(Richness ~ NAP + week.cat, data = RIKZdat)
summary(lm.ancova)
```

```
##
## Call:
## lm(formula = Richness ~ NAP + week.cat, data = RIKZdat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0788 -1.4014 -0.3633  0.6500 12.0845
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.3677     0.9459  12.017 7.48e-15 ***
## NAP          -2.2708     0.4678  -4.854 1.88e-05 ***
## week.cat2    -7.6251     1.2491  -6.105 3.37e-07 ***
## week.cat3    -6.1780     1.2453  -4.961 1.34e-05 ***
## week.cat4    -2.5943     1.6694  -1.554    0.128
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.987 on 40 degrees of freedom
## Multiple R-squared:  0.6759, Adjusted R-squared:  0.6435
## F-statistic: 20.86 on 4 and 40 DF,  p-value: 2.369e-09
```

Aside: historically, statistical inference from a model with a single categorical variable and a continuous variable (and no interaction between the two) was referred to as an analysis of covariance or ANCOVA. We could also have considered a model with only `week.cat` (and not `NAP`), which would have led to an analysis of variance or ANOVA. This distinction (between ANOVA, ANCOVA, and other regression models) is more historical than practical, however, as each of these approaches shares the same underlying statistical machinery.[3]

The model with `NAP` and `week.cat` can be written as:

$$Richness_i = \beta_0 + \beta_1 NAP_i + \beta_2 I(week = 2)_i + \beta_3 I(week = 3)_i + \beta_4 I(week = 4)_i + \epsilon_i,$$

where $I(week = 2)_i$, $I(week = 3)_i$, and $I(week = 4)_i$ are indicator variables for Week 2, 3, and 4, respectively. For example,

$$I(week = 2)_i = \begin{cases} 1 & \text{if the } i^{th} \text{ observation was from week 2} \\ 0 & \text{otherwise} \end{cases}$$

Let's again inspect the design matrix that R creates when fitting the model. Here, we will look at the 1^{st} observation from each week by selecting the 10th, 20th, 30th and 25th observations:

[3]In fact, many more connections can be made between linear regression models and common statistical methods; see e.g., https://lindeloev.github.io/tests-as-linear/.

```r
RIKZdat[c(10, 20, 30, 25),c("Richness", "week", "NAP")]
```

```
##    Richness week    NAP
## 10       17    1 -1.334
## 20        4    2 -0.811
## 30        4    3  0.766
## 25        6    4  0.054
```

```r
model.matrix(lm.ancova)[c(10, 20, 30, 25),]
```

```
##    (Intercept)    NAP week.cat2 week.cat3 week.cat4
## 10           1 -1.334         0         0         0
## 20           1 -0.811         1         0         0
## 30           1  0.766         0         1         0
## 25           1  0.054         0         0         1
```

We see that R created 3 dummy variables representing weeks 2, 3, and 4 and that we can identify observations from week 1 as having 0's for all 3 dummy variables. In matrix form, we can write our model for these 4 observations as:

$$\begin{bmatrix}17\\4\\4\\6\end{bmatrix} = \begin{bmatrix}1 & -1.134 & 0 & 0 & 0\\1 & -0.811 & 1 & 0 & 0\\1 & 0.766 & 0 & 1 & 0\\1 & 0.054 & 0 & 0 & 1\end{bmatrix} \times \begin{bmatrix}\beta_0\\\beta_1\\\beta_2\\\beta_3\\\beta_4\end{bmatrix} + \begin{bmatrix}\epsilon_1\\\epsilon_2\\\epsilon_3\\\epsilon_4\end{bmatrix}$$

Because the effect of `NAP` and `week.cat` are additive (we did not include an interaction between these two variables), the effect of `NAP` on species `Richness` is assumed to be the same for all weeks (i.e., there is a common slope for all 4 weeks). The intercept, however, is different for each week. We can see this by writing down a separate equation for the data collected from each week formed by plugging in appropriate values for each of our indicator variables and then collecting like terms:

- Week 1: $Richness_i = \beta_0 + \beta_1 NAP_i + \epsilon_i$
- Week 2: $Richness_i = [\beta_0 + \beta_2(1)] + \beta_1 NAP_i + \epsilon_i$
- Week 3: $Richness_i = [\beta_0 + \beta_3(1)] + \beta_1 NAP_i + \epsilon_i$
- Week 4: $Richness_i = [\beta_0 + \beta_4(1)] + \beta_1 NAP_i + \epsilon_i$

By comparing weeks 2 and 1, we can see that β_2 represents the difference in expected `Richness` between week 2 and week 1 (if we hold `NAP` constant)[4]. Similarly, β_3 and β_4 represent differences in expected `Richness` between week 3 and week 1 and week 4 and week 1, respectively (if, again, we hold `NAP` constant).

Lastly, it helps to visualize the model and the implied relationship betweeen `Richness` and `NAP` each week (Figure 3.5). This plot makes it clear that the effect of `NAP` is assumed to be constant for all of the weeks and that the expected `Richess` when `NAP` $= 0$ varies by week (i.e., we have a model with constant slope but varying intercepts).

[4]We refer to expected `Richness` here to signify that we need to average over ϵ_i

```
# add the fitted values to our RIZK data
RIKZdat <- RIKZdat %>% mutate(p.ancova = predict(lm.ancova))

# plot using ggplot
ggplot(data = RIKZdat,
      aes(x = NAP, y = Richness, color = week.cat)) +
      geom_point() + geom_line(aes(y = p.ancova)) +
      scale_colour_colorblind()
```

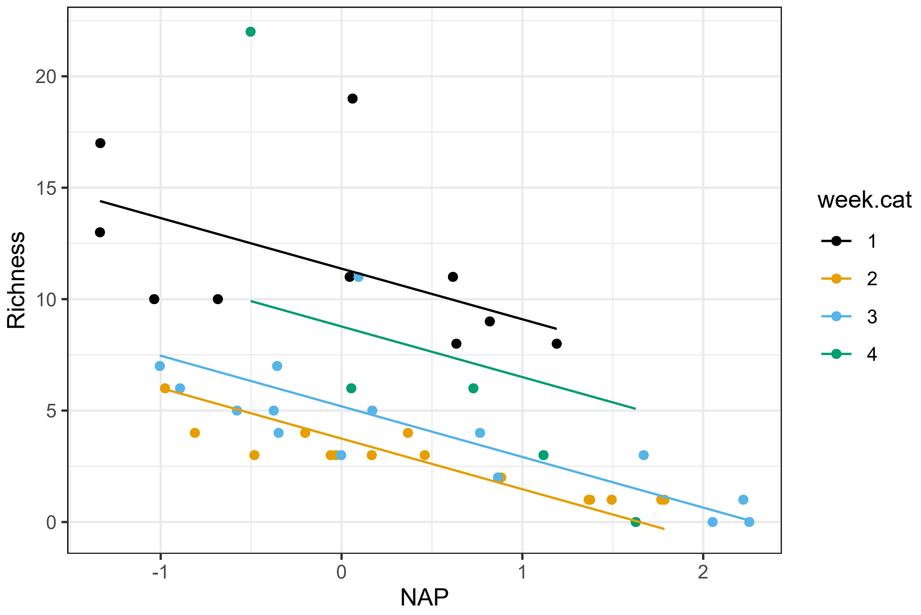


FIGURE 3.5 Expected `Richness` as a function of `NAP` for each week in a model without interactions.

3.7.2 Means coding

We can fit the same model using means coding by removing the intercept:

```
lm.ancova.2 <- lm(Richness ~ NAP + week.cat - 1, data = RIKZdat)
summary(lm.ancova.2)
```

```
##
## Call:
## lm(formula = Richness ~ NAP + week.cat - 1, data = RIKZdat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0788 -1.4014 -0.3633  0.6500 12.0845
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## NAP        -2.2708     0.4678  -4.854 1.88e-05 ***
## week.cat1  11.3677     0.9459  12.017 7.48e-15 ***
## week.cat2   3.7426     0.8026   4.663 3.44e-05 ***
## week.cat3   5.1897     0.7979   6.505 9.24e-08 ***
## week.cat4   8.7734     1.3657   6.424 1.20e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.987 on 40 degrees of freedom
## Multiple R-squared:  0.8604, Adjusted R-squared:  0.843
## F-statistic: 49.32 on 5 and 40 DF,  p-value: 4.676e-16
```

We see that the coefficient for `week.cat1` is identical to the intercept in the effects model, since week 1 is the reference level in that model. In addition, the other coefficients for the `week.cat` variables represent the intercepts in the other weeks. If we look at the design matrix for our same 4 observations, we see R creates a separate dummy variable for each week and that the intercept column has been removed.

```
model.matrix(lm.ancova.2)[c(10,20,30,25),]
```

```
##       NAP week.cat1 week.cat2 week.cat3 week.cat4
## 10 -1.334         1         0         0         0
## 20 -0.811         0         1         0         0
## 30  0.766         0         0         1         0
## 25  0.054         0         0         0         1
```

Thus, the means model is parameterized with a separate intercept for each week, and the design matrix for these observations can be written as:

$$\begin{bmatrix}17\\4\\4\\6\end{bmatrix} = \begin{bmatrix}-1.134 & 1 & 0 & 0 & 0\\-0.811 & 0 & 1 & 0 & 0\\0.766 & 0 & 0 & 1 & 0\\0.054 & 0 & 0 & 0 & 1\end{bmatrix} \times \begin{bmatrix}\beta_1\\\beta_2\\\beta_3\\\beta_4\\\beta_5\end{bmatrix} + \begin{bmatrix}\epsilon_1\\\epsilon_2\\\epsilon_3\\\epsilon_4\end{bmatrix}$$

3.8 Models with interactions

Let's inspect the residuals from our model fit with effects coding for `week.cat` plus `NAP` (the residuals will be identical using either model formulation, though). In this case, we will see how we can create our own customized residual plot using the `fortify` function along with `ggplot` (the `fortify` function in the `broom` package augments the data set used to fit the model with various outputs from the fitted model, including fitted values and residuals). This will allow us to use color to indicate the week of each observation (Figure 3.6).

```
head(fortify(lm.ancova))
```

```
  Richness    NAP week.cat      .hat   .sigma      .cooksd   .fitted     .resid
1       11  0.045        1 0.1005319 3.025215 0.0001963069 11.265517 -0.2655173
```

```
2       10 -1.036       1 0.1213725 2.958047 0.0487598818 13.720207 -3.7202072
3       13 -1.336       1 0.1373130 3.015884 0.0081202327 14.401435 -1.4014348
4       11  0.616       1 0.1126489 3.020466 0.0034083423  9.968914  1.0310858
5       10 -0.684       1 0.1082954 2.984729 0.0260386947 12.920900 -2.9209002
6        8  1.190       1 0.1409419 3.023361 0.0018954268  8.665499 -0.6654988
    .stdresid
1 -0.09371168
2 -1.32849079
3 -0.50505662
4  0.36638770
5 -1.03538063
6 -0.24034207
```

```r
ggplot(fortify(lm.ancova)) +
  geom_point(aes(x = .fitted, y = .resid, col = week.cat)) +
  geom_hline(yintercept = 0)+theme_bw() +  scale_colour_colorblind()
```

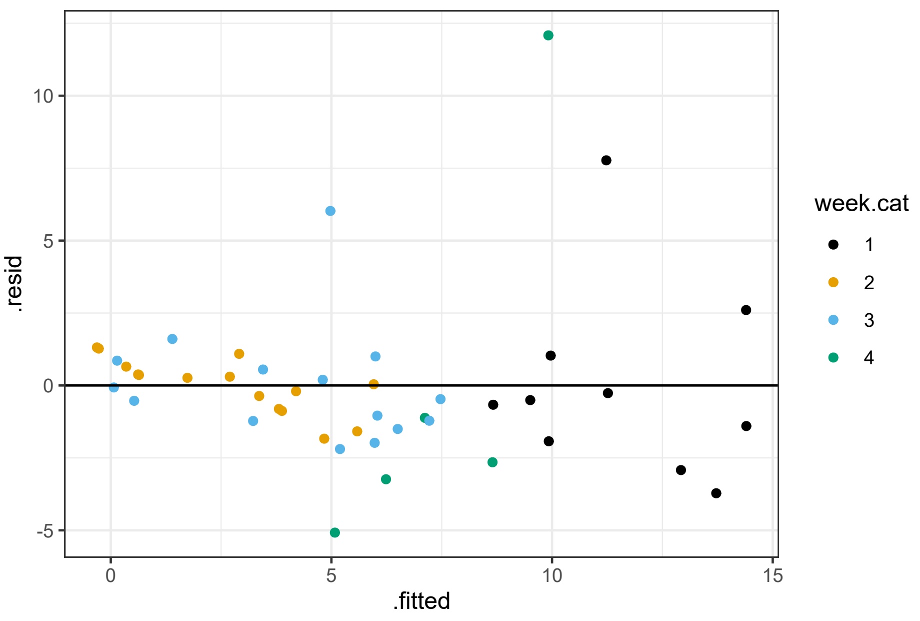


FIGURE 3.6 Residual versus fitted value plot for the model relating species richness to week and `NAP` (the ANCOVA model).

We see that the residuals appear to increase in variance with higher fitted values (this is common when modeling count data [5]). There are also a few residuals with really high absolute values.

Think-pair-share: How can we address these issues?

It is not uncommon to find that model assumptions are not met perfectly when analyzing real data. When this happens, it is tempting to search for model-based solutions, which can send you down a spiraling path towards models with more and more complexity, landing you well outside your comfort zone. We will discuss these challenges more down the road when we cover modeling strategies (see Chapter 8). For now, let's

[5]Note that our species richness measure is just the count of species on the sampled beach

assume you have a reason to believe the effect of `NAP` varies from week to week. If this were the case, we might consider adding an interaction between `NAP` and `week.cat`.[6]

3.8.1 Effects coding

We can include an interaction between `NAP` and `week.cat` in R using either `Richness ~ NAP * week.cat` or `Richness ~ NAP + week.cat + NAP:week.cat`:

```
lmfit.inter <- lm(Richness ~ NAP * week.cat, data = RIKZdat)
summary(lmfit.inter)
```

```

Call:
lm(formula = Richness ~ NAP * week.cat, data = RIKZdat)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.3022 -0.9442 -0.2946  0.3383  7.7103 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)   11.40561    0.77730  14.673  < 2e-16 ***
NAP           -1.90016    0.87000  -2.184 0.035369 *  
week.cat2     -8.04029    1.05519  -7.620 4.30e-09 ***
week.cat3     -6.37154    1.03168  -6.176 3.63e-07 ***
week.cat4      1.37721    1.60036   0.861 0.395020    
NAP:week.cat2  0.42558    1.12008   0.380 0.706152    
NAP:week.cat3 -0.01344    1.04246  -0.013 0.989782    
NAP:week.cat4 -7.00002    1.68721  -4.149 0.000188 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.442 on 37 degrees of freedom
Multiple R-squared:  0.7997,	Adjusted R-squared:  0.7618 
F-statistic: 21.11 on 7 and 37 DF,  p-value: 3.935e-11
```

Let's look at the design matrix for a few different observations:

```
RIKZdat[c(10,20,30,25),c("Richness", "week", "NAP")]
```

```
##    Richness week    NAP
## 10       17    1 -1.334
## 20        4    2 -0.811
## 30        4    3  0.766
## 25        6    4  0.054
```

[6]It is common with experimental data to test for significant interactions prior to testing main effects of individual predictors. For observational data, however, it is prudent to be more cautious. A sensible approach is often to include interactions only when they can be justified *a priori* based on biological grounds. Here, for illustrative purposes only, we will explore a model that includes an interaction between `NAP` and `week.cat`, but we suspect it would be difficult to motivate the need for this interaction, and the researchers did not design their study to test for it.

```
model.matrix(lmfit.inter)[c(10,20,30,25),]
```

```
##    (Intercept)    NAP week.cat2 week.cat3 week.cat4 NAP:week.cat2 NAP:week.cat3
## 10           1 -1.334         0         0         0         0.000         0.000
## 20           1 -0.811         1         0         0        -0.811         0.000
## 30           1  0.766         0         1         0         0.000         0.766
## 25           1  0.054         0         0         1         0.000         0.000
##    NAP:week.cat4
## 10         0.000
## 20         0.000
## 30         0.000
## 25         0.054
```

Here, we see that we added 3 new predictors to our design matrix last seen in Section 3.7. These columns are formed by multiplying our original 3 dummy variables (indicating weeks 2, 3, and 4) by `NAP`. Thus, our model can be written as:

$$\begin{aligned}Richness_i = \beta_0 + \beta_1 NAP_i + \beta_2 I(week = 2)_i + \beta_3 I(week = 3)_i + \beta_4 I(week = 4)_i + \\ \beta_5 NAP_i I(week = 2)_i + \beta_6 NAP_i I(week = 3)_i + \beta_7 NAP_i I(week = 4)_i + \epsilon_i\end{aligned}$$

Or, in matrix notation (for our 4 observations above):

$$\begin{bmatrix}11\\10\\13\\11\end{bmatrix} = \begin{bmatrix}1 & -1.334 & 0 & 0 & 0 & 0 & 0 & 0\\ 1 & -0.811 & 1 & 0 & 0 & -0.811 & 0 & 0\\ 1 & 0.766 & 0 & 1 & 0 & 0 & 0.766 & 0\\ 1 & 0.054 & 0 & 0 & 1 & 0 & 0 & 0.054\end{bmatrix} \times \begin{bmatrix}\beta_0\\\beta_1\\\beta_2\\\beta_3\\\beta_4\\\beta_5\\\beta_6\\\beta_7\end{bmatrix} + \begin{bmatrix}\epsilon_1\\\epsilon_2\\\epsilon_3\\\epsilon_4\end{bmatrix}$$

In this model, we have a separate slope and intercept for each week, which becomes more evident when we write out equations for each week (by plugging in appropriate values for our indicator variables and then collecting like terms):

- Week 1: $Richness_i = \beta_0 + \beta_1 NAP_i + \epsilon_i$
- Week 2: $Richness_i = [\beta_0 + \beta_2(1)] + [\beta_1 + \beta_5(1)]NAP_i + \epsilon_i$
- Week 3: $Richness_i = [\beta_0 + \beta_3(1)] + [\beta_1 + \beta_6(1)]NAP_i + \epsilon_i$
- Week 4: $Richness_i = [\beta_0 + \beta_4(1)] + [\beta_1 + \beta_7(1)]NAP_i + \epsilon_i$

Thus, we see that β_0 and β_1 represent the intercept and slope for our reference category (week 1). The parameters β_2, β_3, and β_4 represent differences in intercepts for weeks 2, 3, and 4 relative to week 1. And, the parameters β_5, β_6, and β_7 represent differences in slopes (associated with `NAP`) for weeks 2, 3, and 4 relative to the slope during week 1.

Visualizing this model (Figure 3.7), we see that the slope for `NAP` during week 4 differs most notably from those of the other weeks.

```
ggplot(fortify(lmfit.inter), aes(NAP, Richness, col = week.cat))+
  geom_line(aes(NAP, .fitted, col = week.cat)) + geom_point() +
  scale_colour_colorblind()
```

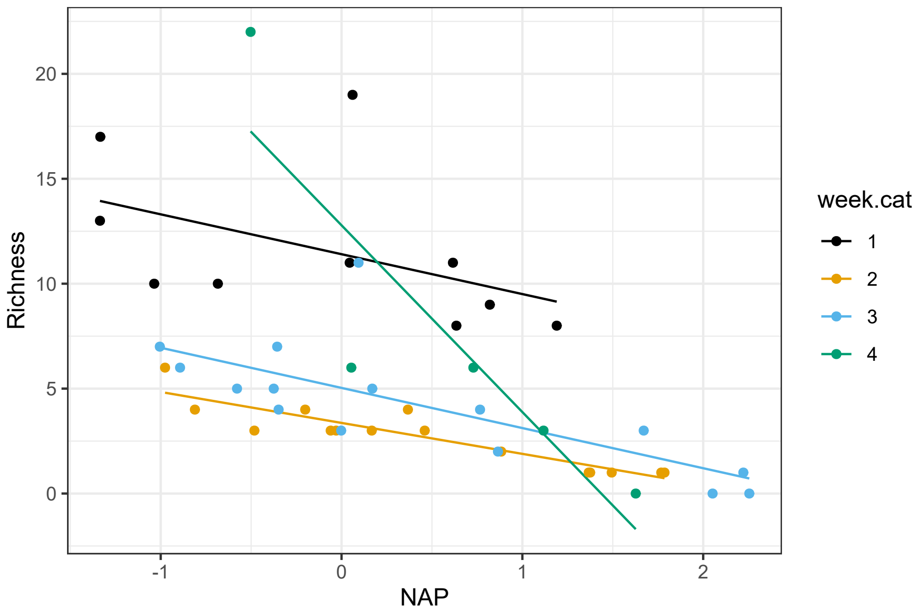


FIGURE 3.7 Regression model relating species richness to week, `NAP` and including an interaction between `NAP` and `week`.

3.8.2 Means model

To fit the means parameterization of the model, we need to drop the columns of the design matrix associated with the intercept and slope for week 1 using the following syntax:

```
lmfit.inter2 <- lm(Richness ~ NAP * week.cat - 1 - NAP, data = RIKZdat)
summary(lmfit.inter2)
```

```
##
## Call:
## lm(formula = Richness ~ NAP * week.cat - 1 - NAP, data = RIKZdat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3022 -0.9442 -0.2946  0.3383  7.7103
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## week.cat1     11.4056     0.7773  14.673  < 2e-16 ***
## week.cat2      3.3653     0.7136   4.716 3.38e-05 ***
## week.cat3      5.0341     0.6784   7.421 7.85e-09 ***
## week.cat4     12.7828     1.3989   9.138 5.05e-11 ***
## NAP:week.cat1  -1.9002     0.8700  -2.184  0.03537 *
## NAP:week.cat2  -1.4746     0.7055  -2.090  0.04353 *
## NAP:week.cat3  -1.9136     0.5743  -3.332  0.00197 **
## NAP:week.cat4  -8.9002     1.4456  -6.157 3.85e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.442 on 37 degrees of freedom
## Multiple R-squared:  0.9138, Adjusted R-squared:  0.8951
## F-statistic:    49 on 8 and 37 DF,  p-value: < 2.2e-16
```

We can inspect the design matrix and write the model for our 4 observations in matrix notation:

$$\begin{bmatrix} 11 \\ 10 \\ 13 \\ 11 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & -1.334 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -0.811 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0.766 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0.054 \end{bmatrix} \times \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \\ \beta_7 \\ \beta_8 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \end{bmatrix}$$

```r
model.matrix(lmfit.inter2)[c(10,20,30,25),]
```

```
##    week.cat1 week.cat2 week.cat3 week.cat4 NAP:week.cat1 NAP:week.cat2
## 10         1         0         0         0        -1.334         0.000
## 20         0         1         0         0         0.000        -0.811
## 30         0         0         1         0         0.000         0.000
## 25         0         0         0         1         0.000         0.000
##    NAP:week.cat3 NAP:week.cat4
## 10         0.000         0.000
## 20         0.000         0.000
## 30         0.766         0.000
## 25         0.000         0.054
```

In this formulation of the model, we directly estimate separate intercepts and slopes for each week (rather than parameters that describe deviations from a reference group):

$$Richness_i = \beta_1 I(week = 1)_i + \beta_2 I(week = 2)_i + \beta_3 I(week = 3)_i + \beta_4 I(week = 4)_i +$$
$$\beta_5 NAP_i I(week = 1) + \beta_6 NAP_i I(week = 2)_i + \beta_7 NAP_i I(week = 3)_i + \beta_8 NAP_i I(week = 4)_i + \epsilon_i$$

This gives us the following equations for the observations from each week:

- Week 1: $Richness_i = \beta_1 + \beta_5 NAP_i + \epsilon_i$
- Week 2: $Richness_i = \beta_2 + \beta_6 NAP_i + \epsilon_i$
- Week 3: $Richness_i = \beta_3 + \beta_7 NAP_i + \epsilon_i$
- Week 4: $Richness_i = \beta_4 + \beta_8 NAP_i + \epsilon_i$

3.8.3 Creating flexible models with dummy variables

After looking at Figure (3.7), and also noting that the interaction terms for weeks 2 and 3 are not significantly different from 0 in the effects model (`lmfit.inter`; Section 3.8.1), we might decide that we want to fit a model that allows each week to have its own intercept, but that the effect of `NAP` is the same in weeks 1-3 and differs only in week 4. If we understand how categorical variables are encoded in regression models, we can fit this model quite easily. We need to include `week.cat` to allow each week to have its own intercept. We also create a single dummy variable (equal to 1 if `week` is equal to 4 and 0 otherwise) and include the interaction between this dummy variable and `NAP`:

```
lm.datadriven <- lm(Richness ~ NAP + week.cat + NAP:I(week==4), data = RIKZdat)
summary(lm.datadriven)
```

```
##
## Call:
## lm(formula = Richness ~ NAP + week.cat + NAP:I(week == 4), data = RIKZdat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3022 -0.9762 -0.0838  0.6269  7.6894
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           11.4187     0.7558  15.108  < 2e-16 ***
## NAP                   -1.7722     0.3875  -4.573 4.77e-05 ***
## week.cat2             -7.9124     0.9996  -7.915 1.23e-09 ***
## week.cat3             -6.4463     0.9965  -6.469 1.16e-07 ***
## week.cat4              1.3641     1.5623   0.873    0.388
## NAP:I(week == 4)TRUE  -7.1280     1.4652  -4.865 1.92e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.387 on 39 degrees of freedom
## Multiple R-squared:  0.7983, Adjusted R-squared:  0.7725
## F-statistic: 30.88 on 5 and 39 DF,  p-value: 1.425e-12
```

We can write down this model as:

$$Richness_i = \beta_0 + \beta_1 NAP_i + \beta_2 I(week = 2)_i + \beta_3 I(week = 3)_i + \beta_4 I(week = 4)_i + \\ \beta_5 NAP_i I(week = 4)_i + \epsilon_i$$

Which implies:

- Week 1: $Richness_i = \beta_0 + \beta_1 NAP_i + \epsilon_i$
- Week 2: $Richness_i = [\beta_0 + \beta_2] + \beta_1 NAP_i + \epsilon_i$
- Week 3: $Richness_i = [\beta_0 + \beta_3] + \beta_1 NAP_i + \epsilon_i$
- Week 4: $Richness_i = [\beta_0 + \beta_4] + [\beta_1 + \beta_5] NAP_i + \epsilon_i$

This model appears to fit the data well (Figure 3.8).

```
ggplot(fortify(lm.datadriven), aes(NAP, Richness, col = week.cat))+
  geom_line(aes(NAP, .fitted, col = week.cat)) + geom_point() +
  scale_colour_colorblind()
```

Although these results may look convincing, we arrived at this result in a very data-driven way. As we will

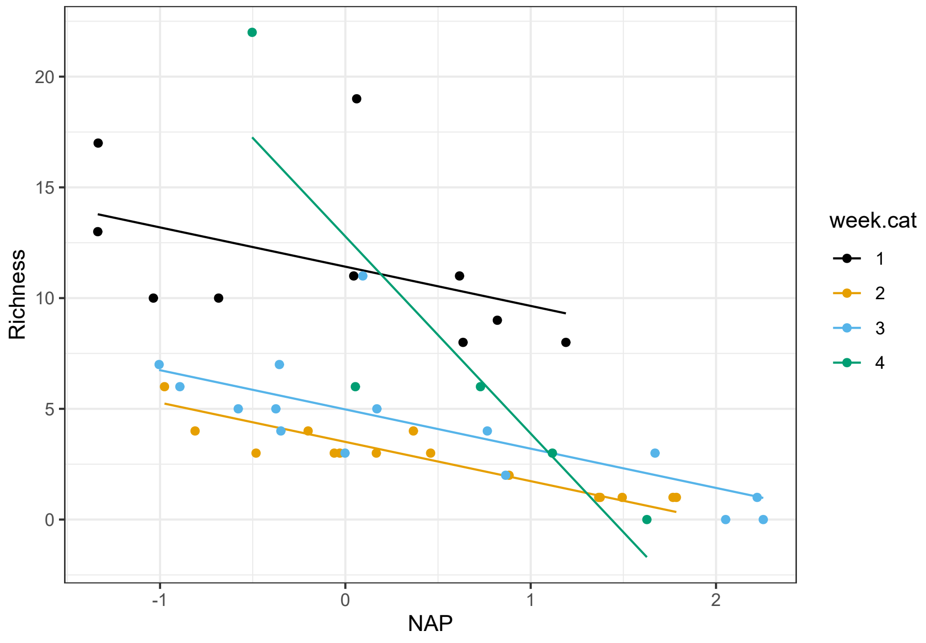


FIGURE 3.8 Species richness versus `NAP` for a model allowing the effect of `NAP` to differ in week 4 versus all other weeks.

discuss in Chapter 8, it is easy to develop models that fit your data well but that perform poorly when applied to new data. In general, you should be skeptical of relationships discovered based on intensive data exploration that were not expected *a priori*. Also, remember that this is an unbalanced design, with only 5 observations during week 4. Hence, this interaction model should be interpreted with great caution. It is quite possible that we are just fitting a model that explains noise in the data. Note, for example, the largest observed species richness value is associated with week 4 (Figure 3.6). This point could be the result of measurement error or some other factor that we have not accounted for in our model and the sole reason for the "need" for the interaction. In addition, if we plot residuals versus fitted values, we still see there are several large outliers (Figure 3.9). So issues remain with our model that may warrant further consideration.

```
ggplot(fortify(lm.datadriven), aes(.fitted, .resid, col = week.cat))+
  geom_point() + geom_hline(yintercept = 0) +
  scale_colour_colorblind()
```

3.8.4 Improving parameter interpretation through centering

When we fit a model with an interaction between a continuous and categorical variable, we are explicitly assuming that the difference between any 2 groups depends on the value of the continuous variable – for example, the difference in species richness between weeks 1 and 2 depends on the value of `NAP` (Figure 3.7). Furthermore, the coefficients associated with the categorical variable represent differences in intercepts, or in other words, mean responses when all other variables are set to 0. As we saw in Section 1.3, intercepts may be difficult to interpret or misleading when they require extrapolating outside of the range of the observed data.

Centering the continuous variable (i.e., subtracting the mean from each observation) can make it easier to interpret the parameters associated with the categorical variable in models with interactions (Schielzeth

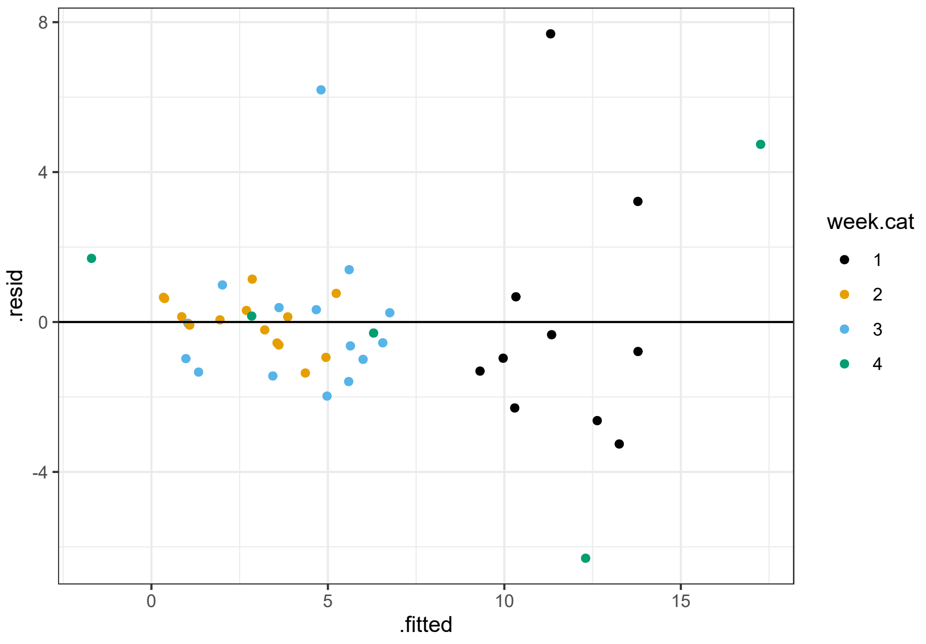


FIGURE 3.9 Residual versus fitted value plot for the model allowing the effect of `NAP` to differ in week 4 versus all other weeks.

2010). For example, if we refit our model using effects coding after centering `NAP` by its mean, then the coefficients for the different weeks will represent contrasts between each group and the reference group when `NAP` is set to its mean (rather than 0).

3.9 Pairwise comparisons

Consider the model below, fit to the `RIKZ` data set. The model includes two quantitative variables (`NAP` and `exposure`) and a categorical variable with more than 2 categories (`week`).

```
lm.RIKZ <- lm(Richness ~ NAP + exposure + week.cat, data = RIKZdat)
summary(lm.RIKZ)
```

```
Call:
lm(formula = Richness ~ NAP + exposure + week.cat, data = RIKZdat)

Residuals:
   Min     1Q Median     3Q    Max 
-4.912 -1.621 -0.313  1.004 11.903 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)   
(Intercept)  23.9262     7.3960   3.235  0.00248 **
```

```
NAP          -2.4344     0.4668  -5.215 6.33e-06 ***
exposure     -1.3972     0.8164  -1.711  0.09495 .
week.cat2    -4.7364     2.0827  -2.274  0.02854 *
week.cat3    -4.2269     1.6671  -2.535  0.01535 *
week.cat4    -1.0814     1.8548  -0.583  0.56323
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.918 on 39 degrees of freedom
Multiple R-squared:  0.6986,    Adjusted R-squared:  0.6599
F-statistic: 18.08 on 5 and 39 DF,  p-value: 2.991e-09
```

For `NAP` and `exposure`, we can use the t-tests in the output above to test whether there is sufficient evidence to conclude the coefficients are not equal to 0. We can also easily determine if we have evidence to suggest there are differences between week 2 and week 1, week 3 and week 1, and week 4 and week 1 (after controlling for `NAP` and `exposure`). What if we want to test for differences between say weeks 2 and 4, or between all pairs of weeks?

This question brings up the thorny issue of multiple comparisons[7]. There are "4 choose 2" = 6 possible pairwise comparisons we could consider when comparing the 4 weeks, and in general $\binom{k}{2} = \frac{k!}{2!(k-2)!}$ possible comparisons if there are k groups[8]. If the probability of making a type I error (i.e., rejecting the null hypothesis when it is true) is α, and each test is independent from all the other tests, we would have a $1-(1-\alpha)^{n_{tests}}$ probability of of rejecting at least one null hypothesis even if all of them were true (roughly an 26% chance if $\alpha = 0.05$ and we conduct 6 tests). This error rate is often referred to as the family-wise error rate. The problem increases with the number of categories associated with the categorical variable and applies also to confidence intervals (i.e., the chance of at least one confidence interval failing to capture the true mean pairwise difference increases with the number of confidence intervals considered).

The usual way of dealing with multiple comparisons is to apply a correction factor that adjusts the p-values associated with individual hypothesis tests (or, alternatively, adjusts the critical values with which the p-values are compared when deciding if there is evidence to reject the null hypothesis). Similarly, one can make adjustments to confidence intervals to make them wider in hopes of controlling the family-wise error rate. There are multiple packages in R for conducting pairwise comparisons with adjustments. We briefly demonstrate one option using the `emmeans` package (Lenth 2021). We begin by estimating the mean `Richness` for each week when `NAP` and `exposure` are set to their mean values using the `emmeans` function:

```r
library(emmeans)
weekcontrasts<-emmeans(lm.RIKZ, "week.cat")
weekcontrasts
```

```
##  week.cat emmean    SE df lower.CL upper.CL
##  1          8.80 1.406 39     5.95    11.64
##  2          4.06 0.995 39     2.05     6.07
##  3          4.57 0.761 39     3.03     6.11
##  4          7.72 1.320 39     5.05    10.38
##
## Confidence level used: 0.95
```

We can then use the `pairs` function to calculate all 6 pairwise differences between weekly means and test

[7] This issue is also relevant to situations where many tests are conducted to determine which of several variables in a model have non-zero regression coefficients (Chapter 8).

[8] The exclamation point is used to represent *factorials*, with $n! = n \times n-1 \times n-2 \cdots \times 1$

whether these differences are statistically significant (i.e., whether we have evidence that the true difference is likely non-zero). We can also request confidence intervals for the pairwise differences in means by supplying the argument `infer = c(TRUE, TRUE)`.

```
pairs(weekcontrasts, infer = c(TRUE, TRUE))
```

```
##  contrast              estimate   SE df lower.CL upper.CL t.ratio p.value
##  week.cat1 - week.cat2    4.736 2.08 39   -0.852   10.325   2.274  0.1217
##  week.cat1 - week.cat3    4.227 1.67 39   -0.247    8.700   2.535  0.0699
##  week.cat1 - week.cat4    1.081 1.85 39   -3.896    6.058   0.583  0.9366
##  week.cat2 - week.cat3   -0.509 1.20 39   -3.725    2.706  -0.425  0.9738
##  week.cat2 - week.cat4   -3.655 1.71 39   -8.241    0.931  -2.139  0.1589
##  week.cat3 - week.cat4   -3.146 1.53 39   -7.252    0.961  -2.055  0.1858
##
## Confidence level used: 0.95
## Conf-level adjustment: tukey method for comparing a family of 4 estimates
## P value adjustment: tukey method for comparing a family of 4 estimates
```

Each row represents a different pairwise comparison identified by the label in the first column. By default, `emmeans` uses Tukey's Honest Significant Difference (HSD) to adjust the p-values and confidence intervals associated with each comparison (Abdi and Williams 2010), thus controlling the family-wise error rate (i.e., the probability that we incorrectly reject at least 1 null hypothesis when all of the null hypotheses are true). Using a family-wise error rate of $\alpha = 0.05$, we would conclude that we do not have enough evidence to reject the null hypothesis for any of the pairwise comparisons, as they all have p-values > 0.05. We can see our conclusions are more conservative than if we had not done any adjustments:

```
pairs(weekcontrasts, infer = c(TRUE, TRUE), adjust= "none")
```

```
##  contrast              estimate   SE df lower.CL upper.CL t.ratio p.value
##  week.cat1 - week.cat2    4.736 2.08 39    0.524    8.949   2.274  0.0285
##  week.cat1 - week.cat3    4.227 1.67 39    0.855    7.599   2.535  0.0153
##  week.cat1 - week.cat4    1.081 1.85 39   -2.670    4.833   0.583  0.5632
##  week.cat2 - week.cat3   -0.509 1.20 39   -2.933    1.914  -0.425  0.6730
##  week.cat2 - week.cat4   -3.655 1.71 39   -7.112   -0.198  -2.139  0.0388
##  week.cat3 - week.cat4   -3.146 1.53 39   -6.241   -0.050  -2.055  0.0466
##
## Confidence level used: 0.95
```

Without any adjustment, we would have concluded that weeks 1 and 2, weeks 1 and 3, weeks 2 and 4, and weeks 3 and 4 all differ from one another. Clearly, then, there is a tradeoff involved when adjusting for multiple comparisons. We can reduce the family-wise type I error rate at the expense of increasing the type II error rate (failing to reject a null hypotheses when it is indeed false). Thus, correcting for multiple comparisons is not without its critics (e.g., Perneger 1998; Moran 2003; Nakagawa 2004). Rather than attempt to control the family-wise error rate (i.e., probability of incorrectly rejecting one or more null hypotheses), many statisticians now advocate for controlling the false discovery rate, defined as the proportion of significant results that reflect type I errors (see e.g., Benjamini and Hochberg 1995; García 2004; Verhoeven, Simonsen, and McIntyre 2005; Pike 2011). In other words, rather than attempt to avoid rejecting *any* null hypotheses that are true, the goal is to ensure that most of the hypotheses that are rejected are indeed false. Controlling the false discovery rate results in more powerful tests, meaning we are more likely to reject hypotheses when they are false, and less conservative adjustments than controlling for the family-wise error rate.

Another option that is sometimes used to control the family-wise error rate is what is called Fisher's least significant difference (LSD) procedure in which a global, multiple degree-of-freedom test is conducted first. If this test is significant, then one proceeds with further pairwise comparisons. If the global test is not significant, then no pairwise comparisons are conducted. This approach is capable of controlling the family-wise error rate when there are only 3 groups under consideration (Meier 2006). We discuss the multiple degree-of-freedom test in Section 3.10. Lastly, it is often beneficial to limit the number of tests conducted to just those comparisons that are of primary interest.

3.10 Multiple degree-of-freedom hypothesis tests

The Fisher's LSD procedure would require us to first test the global null hypothesis that the coefficients for `week.cat2`, `week.cat3` and `week.cat4` are all 0 (i.e., all weeks have the same species `Richness` after adjusting for `NAP` and `exposure`) versus an alternative hypothesis that at least 1 of the coefficients is non-zero. Tests of joint hypotheses (i.e., hypotheses involving multiple parameters set to different values, usually 0), can be conducted using either a χ^2 or F distribution. Tests using the χ^2 distribution are usually based on large sample approximations that lead to a Normal distribution (the square of a Normal random variable is distributed as χ^2_1). The F distribution, like the $t-$distribution, is more appropriate when all of the assumptions of the linear regression model are met. The two tests will be equivalent as sample sizes approach infinity. The χ^2 and F distributions only assign probabilities to positive values and therefore, p-values are calculated as areas to the right of our test statistic, calculated from the observed data.

F-statistics have an associated numerator and denominator degrees of freedom and can be most easily understood in terms of comparing two models – a full model (M_F) and a reduced model (M_R) in which some subset of parameters have been set equal to 0. Let p_F and p_R be the number of parameters in the full and reduced models, respectively. The numerator degrees of freedom, k_1, is equal to the difference in the number of parameters ($k_1 = p_F - p_R = 3$ when testing the null hypothesis that the coefficients for `week.cat2`, `week.cat3` and `week.cat4` are all 0). The denominator degrees of freedom, k_2, is determined from the full model that includes these additional parameters and is equal to $n - p_F$ where n is the sample size. The F-statistic can be calculated as:

$$F = \frac{(SSE_{M_R} - SSE_{M_F})/(p_F - p_R)}{SSE_{M_F}/(n-p_F)} = \frac{(SSR_{M_F} - SSR_{M_R})/(p_F - p_R)}{SSE_{M_F}/(n-p_F)}, \tag{3.3}$$

where SSE_{M_R} and SSR_{M_R} are the residual and regression sums-of-squares for the restricted/constrained model (in this case, with parameters for `week.cat2`, `week.cat3`, and `week.cat4` set to 0), and SSE_{M_F} and SSR_{M_F} are the residual and regression sums-of-squares for the full model (where these parameters are also estimated). Thus, the numerator captures the additional variability that is explained by including the additional parameters and should be close to 0 if the null hypothesis is true.

When using software to calculate sums of squares, you may be surprised to learn that sums-of-squares may be calculated in different ways, depending on whether we consider constructing a model sequentially, adding one variable at a time (so called type I sums of squares), or we consider removing a variable from the model containing all other terms (type III sums of squares); there is also a type II sums of squares that considers removing the term from a model containing all other terms except those that involve the focal predictor (e.g., interaction terms are not included);[9]. For our current model, type II and type III sums of squares will be identical.

The bottom line is that whether we build the model in a "forward" or "backward" direction influences

[9]For more information, see https://www.r-bloggers.com/2011/03/anova-%E2%80%93-type-iiiiii-ss-explained/)

what other variables are included or adjusted for when calculating regression sums of squares. When testing hypotheses, I recommend a backwards model-selection approach, which is implemented using the `Anova` function in the `car` package (its default is to use type II sums of squares). By contrast, the `anova` function in base R will implement a sequential approach, in which case the results of the test will depend on the ordering of the variables when you specify the model.

Let's start with the `Anova` function:

```
library(car)
Anova(lm.RIKZ)
```

```
Anova Table (Type II tests)

Response: Richness
          Sum Sq Df F value    Pr(>F)
NAP       231.59  1 27.1999 6.335e-06 ***
exposure   24.94  1  2.9289   0.09495 .
week.cat   73.19  3  2.8654   0.04888 *
Residuals 332.07 39
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `Anova` function returns tests appropriate for *backwards selection* (see Section 8.4.1) - meaning that these tests determine if we have enough evidence to suggest that the variable of interest is associated with the response variable, after adjusting for the other variables in the model. The F-tests for `NAP` and `exposure` are equivalent to the t-tests in the summary of the `lm` (in fact, the F-statistics are equal to the square of the $t-$statistics we saw previously). The advantage of using `Anova` is that it also returns a *multiple degree-of-freedom test* for `week.cat`. The associated p-value (0.0488) suggests we have enough evidence in the data to conclude that at least one of the weeks differs from the others (in terms of species richness after adjusting for `exposure` and `NAP`).

If we had used the `anova` function, we would end up with a different set of tests resulting from *sequentially* adding variables one at a time. In this case, the order in which the predictor variables appear matters. We will compare two different calls to `lm` below to demonstrate this:

```
anova(lm(Richness ~ week.cat + exposure + NAP, data = RIKZdat))
```

```
Analysis of Variance Table

Response: Richness
          Df Sum Sq Mean Sq F value    Pr(>F)
week.cat   3 534.31 178.104 20.9177 3.060e-08 ***
exposure   1   3.67   3.675  0.4316    0.5151
NAP        1 231.59 231.593 27.1999 6.335e-06 ***
Residuals 39 332.07   8.514
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-values for these tests are associated with the following alternative hypotheses:

- at least one coefficient associated with `week.cat` is non-zero in a model that only includes `week.cat` (since it was specified first)

- the coefficient for `exposure` is non-zero in a model with `week.cat` and `exposure`
- the coefficient for `NAP` is non-zero in a model that contains `week.cat`, `exposure`, and `NAP`.

If we reverse the order variables are entered into the model, we get a different set of p-values, with the test for `week.cat` now matching the test from the `Anova` function.

```
  anova(lm(Richness ~ NAP + exposure + week.cat, data = RIKZdat))
```

```
## Analysis of Variance Table
##
## Response: Richness
##           Df Sum Sq Mean Sq F value    Pr(>F)
## NAP        1 357.53  357.53 41.9907 1.117e-07 ***
## exposure   1 338.86  338.86 39.7977 1.931e-07 ***
## week.cat   3  73.19   24.40  2.8654   0.04888 *
## Residuals 39 332.07    8.51
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In addition to using an F-test for categorical variables with more than 2-levels, we can also use an F-test to evaluate whether any of our predictors explain a significant proportion of variability in the response as we will see in the next section.

3.11 Regression F-statistic

When using the `summary` function with a fitted regression model, you may notice an F-statistic and p-value at the bottom of the output:

```
summary(lm.RIKZ)
```

```
##
## Call:
## lm(formula = Richness ~ NAP + exposure + week.cat, data = RIKZdat)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.912 -1.621 -0.313  1.004 11.903
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  23.9262     7.3960   3.235  0.00248 **
## NAP          -2.4344     0.4668  -5.215 6.33e-06 ***
## exposure     -1.3972     0.8164  -1.711  0.09495 .
## week.cat2    -4.7364     2.0827  -2.274  0.02854 *
## week.cat3    -4.2269     1.6671  -2.535  0.01535 *
## week.cat4    -1.0814     1.8548  -0.583  0.56323
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.918 on 39 degrees of freedom
## Multiple R-squared:  0.6986, Adjusted R-squared:  0.6599
## F-statistic: 18.08 on 5 and 39 DF,  p-value: 2.991e-09
```

This F-statistic is testing whether all regression coefficients (other than the intercept) are simultaneously 0 versus an alternative hypothesis that at least one of the coefficients is non-zero. Thus, the numerator degrees of freedom is equal to $p-1$ (eq. (3.3)), where p is the number of parameters in the model ($p-1=5$ in the above case). The denominator degrees of freedom is again equal to $n-p$. Similar to the idea behind Fisher's least significant difference (LSD) procedure, we might consider only testing hypotheses involving individual coefficients when this global test is rejected.

3.12 Contrasts: Estimation of linear combinations of parameters

Often, we are interested in estimating some linear combination (i.e., a weighted sum) of our regression parameters. Consider again the model with only `NAP` and `week.cat` fit using effects coding:

$$Richness_i = \beta_0 + \beta_1 NAP_i + \beta_2 I(week=2)_i + \beta_3 I(week=3)_i + \beta_4 I(week=4)_i + \epsilon_i,$$

We saw how we can use the functions in the `emmeans` package to estimate the difference in `Richness` between weeks 2 and 3 (as well as between other weeks), while controlling for `NAP`. We can estimate this contrast between mean `Richness` in weeks 2 and 3 as: $\hat{\beta}_2 - \hat{\beta}_3$. To correctly estimate its uncertainty requires considering how much $\hat{\beta}_2$ and $\hat{\beta}_3$ vary as well as how much they co-vary across data sets if we could replicate the sampling design many times. I.e., we must consider the variance/covariance matrix of our regression parameter estimators, $\hat{\Sigma}_{\hat{\beta}}$:

$$\hat{\Sigma}_{\hat{\beta}} = \begin{bmatrix} Var(\hat{\beta}_0) & Cov(\hat{\beta}_0,\hat{\beta}_1) & Cov(\hat{\beta}_0,\hat{\beta}_2) & Cov(\hat{\beta}_0,\hat{\beta}_3) & Cov(\hat{\beta}_0,\hat{\beta}_4) \\ Cov(\hat{\beta}_0,\hat{\beta}_1) & Var(\hat{\beta}_1) & Cov(\hat{\beta}_1,\hat{\beta}_2) & Cov(\hat{\beta}_1,\hat{\beta}_3) & Cov(\hat{\beta}_1,\hat{\beta}_4) \\ Cov(\hat{\beta}_0,\hat{\beta}_2) & Cov(\hat{\beta}_1,\hat{\beta}_2) & Var(\hat{\beta}_2) & Cov(\hat{\beta}_2,\hat{\beta}_3) & Cov(\hat{\beta}_2,\hat{\beta}_4) \\ Cov(\hat{\beta}_0,\hat{\beta}_3) & Cov(\hat{\beta}_1,\hat{\beta}_3) & Cov(\hat{\beta}_2,\hat{\beta}_3) & Var(\hat{\beta}_3) & Cov(\hat{\beta}_3,\hat{\beta}_4) \\ Cov(\hat{\beta}_0,\hat{\beta}_4) & Cov(\hat{\beta}_1,\hat{\beta}_4) & Cov(\hat{\beta}_2,\hat{\beta}_4) & Cov(\hat{\beta}_3,\hat{\beta}_4) & Var(\hat{\beta}_4) \end{bmatrix}$$

.

We can obtain $\hat{\Sigma}_{\hat{\beta}}$ using the `vcov` function applied to our linear model object:

```r
(Sigma_b<- vcov(lm.ancova))
```

```
##             (Intercept)         NAP  week.cat2  week.cat3  week.cat4
## (Intercept)  0.89479838  0.02238355 -0.9054037 -0.9045546 -0.9083360
## NAP          0.02238355  0.21880298 -0.1260524 -0.1177525 -0.1547156
## week.cat2   -0.90540371 -0.12605240  1.5601330  0.9603457  0.9816402
## week.cat3   -0.90455462 -0.11775247  0.9603457  1.5508847  0.9757713
## week.cat4   -0.90833595 -0.15471559  0.9816402  0.9757713  2.7869250
```

The diagonal elements of $\hat{\Sigma}_{\hat{\beta}}$ contain the estimated variances of the regression parameter estimators. If we take their square root, we obtain the SE's reported using the summary function:

```
summary(lm.ancova)
```

```
##
## Call:
## lm(formula = Richness ~ NAP + week.cat, data = RIKZdat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0788 -1.4014 -0.3633  0.6500 12.0845
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.3677     0.9459  12.017 7.48e-15 ***
## NAP          -2.2708     0.4678  -4.854 1.88e-05 ***
## week.cat2    -7.6251     1.2491  -6.105 3.37e-07 ***
## week.cat3    -6.1780     1.2453  -4.961 1.34e-05 ***
## week.cat4    -2.5943     1.6694  -1.554    0.128
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.987 on 40 degrees of freedom
## Multiple R-squared:  0.6759, Adjusted R-squared:  0.6435
## F-statistic: 20.86 on 4 and 40 DF,  p-value: 2.369e-09
```

```
sqrt(diag(Sigma_b))
```

```
## (Intercept)         NAP   week.cat2   week.cat3   week.cat4
##   0.9459378   0.4677638   1.2490529   1.2453452   1.6694086
```

The off diagonal elements of $\hat{\Sigma}_{\hat{\beta}}$ tell us how our parameter estimates are expected co-vary (if we were to repeatedly generate new data sets and fit the same model). Note that $Cov(x, y) = Cov(y, x)$, and therefore, $\hat{\Sigma}_{\hat{\beta}}$ is a symmetric matrix.

For constants a and b, the $Var(ax + by) = a^2Var(x) + b^2Var(y) + 2abCov(x, y)$. Thus, $Var(\hat{\beta}_2 - \hat{\beta}_3) = Var(\hat{\beta}_2) + Var(\hat{\beta}_3) - 2Cov(\hat{\beta}_2, \hat{\beta}_3)$. We can calculate this variance using matrix multiplication. Define the transpose of a column vector, c as c'. We will use $c' = c(0, 0, 1, -1, 0)$ to estimate our contrast of interest (i.e., the difference in expected species richness for weeks 2 and 3, $\hat{\beta}_2 - \hat{\beta}_3$) via matrix multiplication:

$$c'\hat{\beta} = \begin{bmatrix} 0 & 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\beta}_3 \\ \hat{\beta}_4 \end{bmatrix} = \hat{\beta}_2 - \hat{\beta}_3$$

.

The variance of this contrast is given by:

$$Var(\hat{\beta}_2 - \hat{\beta}_3) = c'\Sigma_b c$$

To verify, let's calculate the standard error of this contrast (equivalent to the square-root of the variance) using matrix multiplication in R.

```
cmat <- c(0, 0, 1, -1, 0)
cmat%*%coef(lm.ancova) # estimate of week 2 - week 3
```

```
##           [,1]
## [1,] -1.447196
```

```
(SEcontrast <- sqrt(t(cmat)%*%Sigma_b%*%(cmat))) # se = sqrt(variance)
```

```
##          [,1]
## [1,] 1.091021
```

```
pairs(emmeans(lm.ancova, "week.cat"), adjust = "none")
```

```
##  contrast              estimate   SE df t.ratio p.value
##  week.cat1 - week.cat2     7.63 1.25 40   6.105  <.0001
##  week.cat1 - week.cat3     6.18 1.25 40   4.961  <.0001
##  week.cat1 - week.cat4     2.59 1.67 40   1.554  0.1280
##  week.cat2 - week.cat3    -1.45 1.09 40  -1.326  0.1922
##  week.cat2 - week.cat4    -5.03 1.54 40  -3.258  0.0023
##  week.cat3 - week.cat4    -3.58 1.54 40  -2.320  0.0255
```

We see that we get an equivalent SE to the one returned by the `pairs` function for the difference between weeks 2 and 3. At this point, you might be wondering why you need to know how to calculate contrasts and their uncertainty using matrix algebra if `emmeans` will do all the hard work for you. Good question! There are times when you may be interested in something other than a simple pairwise difference. For example, we could test whether the last two weeks had higher species richness, on average, than the first two using $c' = (0, 1/2, 1/2, -1/2, -1/2)$, giving $\left(\frac{\hat{\beta}_1+\hat{\beta}_2}{2}\right) - \left(\frac{\hat{\beta}_3+\hat{\beta}_4}{2}\right)$. For an application, see Iannarilli et al. (2021). For a more thorough discussion of the importance of contrasts, see Schad et al. (2020).

3.13 Aside: Revisiting F-tests and comparing them to Wald χ^2 tests

In Section 3.10, we considered the F-statistic written in terms of sums of squares. We can also formulate F-tests using matrix algebra. Similar to the previous section, we will use C (here as a matrix instead of a vector) to identify one or more linear combinations of our regression parameters. Let's consider again the model from Section 3.10:

$$Richness_i = \beta_0 + \beta_1 NAP_i + \beta_2 exposure_i + \beta_3 I(week = 2)_i + \beta_4 I(week = 3)_i + \quad (3.4)$$

$$\beta_5 I(week = 4)_i + \epsilon_i, \quad (3.5)$$

Define our contrast matrix, C as:

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.6)$$

Using matrix multiplication, $C\beta$, identifies the parameters involved in our multiple degree of freedom hypothesis test:

$$C\beta = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \end{bmatrix} = \begin{bmatrix} \beta_3 \\ \beta_4 \\ \beta_5 \end{bmatrix} \tag{3.7}$$

The F-statistic can be written as:

$$F = \frac{1}{k_1}(C\hat{\beta})'(C\hat{\Sigma}_{\hat{\beta}}C')^{-1}(C\hat{\beta}) \tag{3.8}$$

where C is our contrast matrix, k_1 is the number of rows in this contrast matrix and the degrees of freedom associated with the hypothesis test, and $\hat{\Sigma}_{\hat{\beta}}$ is once again our estimated variance-covariance matrix associated with $\hat{\beta}$. We can use matrix algebra to verify the F-statistic from the test calculated using the `Anova` function:

```
cmat <- matrix(c(0, 0, 0, 1, 0, 0,
                 0, 0, 0, 0, 1, 0,
                 0, 0, 0, 0, 0, 1), byrow=TRUE, ncol=6)
t(cmat%*%coef(lm.RIKZ))%*%solve(cmat%*%vcov(lm.RIKZ)%*%t(cmat))%*%(cmat%*%coef(lm.RIKZ))/3
```

```
##          [,1]
## [1,] 2.865437
```

```
Anova(lm.RIKZ)
```

```
## Anova Table (Type II tests)
##
## Response: Richness
##           Sum Sq Df F value    Pr(>F)
## NAP       231.59  1 27.1999 6.335e-06 ***
## exposure   24.94  1  2.9289   0.09495 .
## week.cat   73.19  3  2.8654   0.04888 *
## Residuals 332.07 39
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Alternatively, we could consider a χ^2 test, with the statistic calculated as follows:

$$\chi^2 = (C\hat{\beta})'(C\hat{\Sigma}_{\hat{\beta}}C')^{-1}(C\hat{\beta}) \tag{3.9}$$

with degrees of freedom equal to the number of rows in C.

```
chisq<-t(cmat%*%coef(lm.RIKZ))%*%solve(cmat%*%vcov(lm.RIKZ)%*%t(cmat))%*%(cmat%*%coef(lm.RIKZ))
pchisq(chisq, df=3, lower.tail=FALSE)
```

```
##            [,1]
## [1,] 0.03516874
```

We get a slightly smaller p-value in this case relative to the F-test, similar to what you would expect if you used a Normal distribution rather than a t-distribution to conduct a hypothesis test with small sample sizes.

The two tests are asymptotically equivalent (i.e., for large sample sizes). The χ^2 test can be motivated by noting that asymptotically:

$$C\hat{\beta} \sim N(C\beta, C\hat{\Sigma}_{\hat{\beta}}C')$$

Thus, in the 1-dimensional case, the χ^2 statistic is equivalent to the square of a z-statistic:

$$\left(\frac{C\hat{\beta} - 0}{SE(C\hat{\beta})}\right)^2$$

Lastly, we can also test hypotheses in which the regression parameters are set to specific values other than 0 by replacing C with $C - \tilde{\beta}$ in equations (3.8) and (3.9), where $\tilde{\beta}$ represents the values of β under the null hypothesis.

3.14 Visualizing multiple regression models

Consider a regression model with two explanatory variables, X_1 and X_2.

$$Y_i = \beta_0 + X_{i,1}\beta_1 + X_{i,2}\beta_2 + \epsilon_i$$

We have already noted that β_1 reflects the "effect" of X_1 on Y after accounting for X_2. Specifically, it describes the expected change in Y for every 1 unit increase in X_1 while holding X_2 constant. If we want to visualize this effect, a simple strategy that is often used is to:

1. Create a data set with X_1 taking on a range of values and with X_2 set to its mean value (for quantitative predictors) or its modal value (for categorical predictors).
2. Generate predictions, $\hat{Y}$, for each value in this data set and plot $\hat{Y}$ versus X_1.

This strategy is easy to implement using the `predict` function in R and generalizes to models with more than two predictors. In addition, there are various packages that will construct this type of effect plot for you. In particular, we will look at the `effects` package (Fox 2003; Fox and Weisberg 2018, 2019) for creating these types plots in Chapter 16.

When fitting a linear regression model with only 1 predictor, it is common to create a scatterplot of Y versus X along with the fitted regression line to visualize the effect of X on Y. This type of plot allows us to quickly visualize the amount of variability in Y explained by X (and also the amount of unexplained variability remaining). It would be nice to have a similar tool available for multiple regression models where model predictions are shown together with data. In the next sections, we will explore two options:

- Added variable plots, also known as *partial regression plots*
- Component + residual plots, also known as *partial residual plots*

These types of plots are not well known among ecologists and are arguably underutilized (Moya-Laraño and Corcobado 2008). There are several functions in R that can be used to create added variable and component + residual plots. Added variable plots can be constructed using the `avPlots` function in the `car` package

(Fox and Weisberg 2019). Component + residual plots can be created using the `termplot` function in base R or the `crPlots` in the `car` package (Fox and Weisberg 2019). We will explore the `avPlots` and `crPlots` functions in the sections that follow and `termplot` in Section 4.7.

3.14.1 Added variable plots

Added variable plots allow us to visualize the effect of X_k after accounting for all other predictors. These plots can be constructed using the following steps:

1. Regress Y against X_{-k} (i.e., all predictors *except* X_k), and obtain the residuals.
2. Regress X_k against all other predictors (X_{-k}) and obtain the residuals.
3. Plot the residuals from [1] (i.e., the part of Y not explained by other predictors) against the residuals from [2] (the part of the focal predictor not explained by the other predictors). If we add a least-squares-regression line relating these two sets of residuals, the slope will be equivalent to the slope in our full model containing all predictors.

Although there are functions in R to construct added variable plots (Section 3.14), we will demonstrate these steps using a simulated data set in the `Data4Ecologists` package. Specifically, the `partialr` data set was simulated so that `y` has a positive association with `x1`, a negative association with `x2` (which is also negatively correlated with `x1`), a quadratic relationship with `x3`, and a spurious relationship with `x4` (due to its correlation with `x1`). A pairwise scatterplot of the data set is show in Figure 3.10.

```
data(partialr) # from Data4Ecologists package
```

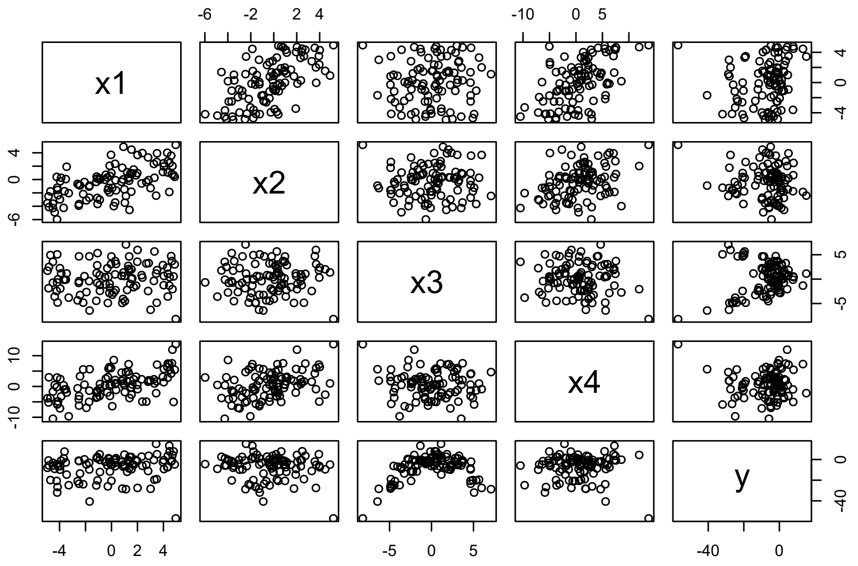


FIGURE 3.10 Pairwise scatterplot of the predictors in the `partialr` data set in the `Data4ecologists` package (Fieberg 2021). These data were simulated so that `y` has a positive association with `x1`, a negative association with `x2` (which is also negatively correlated with `x1`), a quadratic relationship with `x3`, and a spurious relationship with `x4` (due to its correlation with `x1`).

TABLE 3.2 Regression coefficients in simple linear regression models and a multivariate regression model fit to the ‘partialr‘ data set.

	Model 1	Model 2	Model 3	Model 4	Model 5
(Intercept)	−7.359	−7.479	−7.384	−7.300	−7.865
x1	0.692				1.562
x2		−0.495			−1.505
x3			0.925		0.842
x4				−0.025	−0.210

First, note that whenever predictor variables are correlated, as they will be in any observational data set, regression coefficients will change when we add or drop predictors from a model as these correlated variables “compete” to predict variance in the response variables (see e.g., Table 3.2). The magnitude and direction of these changes will depend on the sign and strength of the correlations among the different predictor variables (see Sections 6 and 7 and Fieberg and Johnson 2015). Thus, choosing an appropriate model can be challenging and should ideally be informed by one’s research question and an assumed Directed Acyclical Graph (DAG) capturing assumptions about how the world works (i.e., causal relationships between the predictor variables and the response variable; see Section 7).

```r
lmx1.y <- lm(y ~ x1, data=partialr)
lmx2.y <- lm(y ~ x2, data=partialr)
lmx3.y <- lm(y ~ x3, data=partialr)
lmx4.y <- lm(y ~ x4, data=partialr)
lmxall.y <- lm(y ~ x1 + x2 + x3 + x4, data=partialr)
```

For now, let’s assume we have chosen to focus on the model containing all four predictor variables and we want to display the effect of `x1` after accounting for the other predictors in the model using an added-variable plot. Let’s walk through the steps of this process:

1. Regress Y against X_{-k} (i.e., all predictors *except* `x1`).

```r
lm.nox1.y <- lm(y ~ x2 + x3 + x4, data=partialr)
```

2. Regress X_k against all other predictors (X_{-k}).

```r
lm.x1.allotherx <- lm(x1 ~ x2 + x3 + x4, data=partialr)
```

3. Plot the residuals from [1] against the residuals from [2] along with a regression line relating these two sets of residuals. We also add a regression line through the origin with the slope coefficient for `x1` from the original regression.

```r
plot(resid(lm.x1.allotherx), resid(lm.nox1.y),
     xlab="E(X1 | X2, X3, X4)", ylab = "E(Y | X2, X3, X4)")
abline(c(0,coef(lmxall.y)[2]), col = "red", lty = 2, lwd = 3)
(lmpartial<-lm(resid(lm.nox1.y) ~ resid(lm.x1.allotherx) - 1))
```

```
##
## Call:
## lm(formula = resid(lm.nox1.y) ~ resid(lm.x1.allotherx) - 1)
##
## Coefficients:
## resid(lm.x1.allotherx)
##                  1.562
```

```
abline(lmpartial)
```

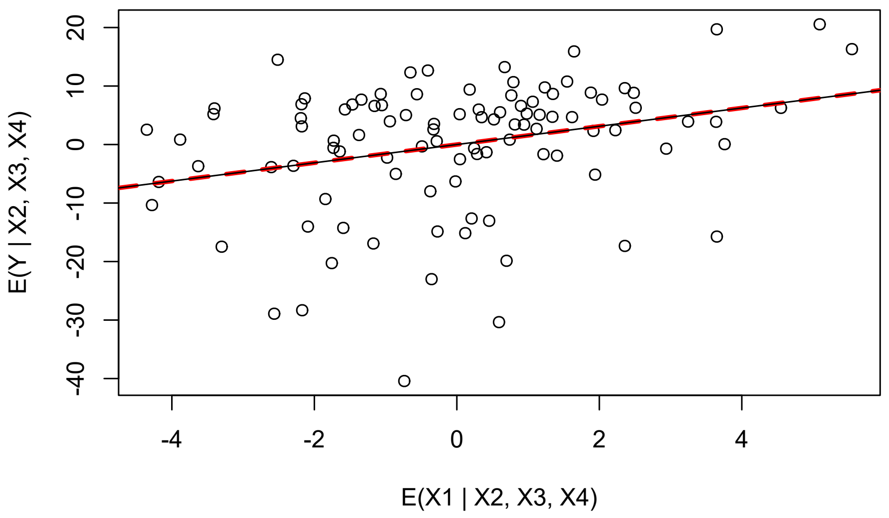


FIGURE 3.11 Added variable plot for variable `x1` in the `partialr` data set.

We see that the slope from the original regression is equivalent to the slope of the regression line relating the residuals from [1] to the residuals from [2]. Rather than construct similar plots for the other variables, we will use the `avPlots` function in the `car` package to produce the full suite of added variable plots (Figure 3.12).

We see that the slope of the line is near 0 in the plot for `x4` (lower right panel of Figure 3.12), suggesting that `x4` provides little to no additional information useful for predicting `y` that is not already contained in the other predictor variables. Furthermore, in the plot for `x3` (lower-left panel of Figure 3.12), we see that it has a clear non-linear relationship with `y` even after accounting for the effects of `x1`, `x2`, and `x4`. Thus, we may want to add a quadratic term or use splines to relax the linearity assumption for this variable (see Section 4).

In summary, added-variable plots depict the slope and the scatter of points around the partial regression line in an analogous way to bi-variate plots in simple linear regression. These plots can be helpful for:

- visualizing the effect of predictor variables (given everything else already in the model)
- diagnosing whether some variables have a non-linear relationship with the response variable
- identifying potential influential points and outliers (`avPlots` highlights these with the row number in the data set)

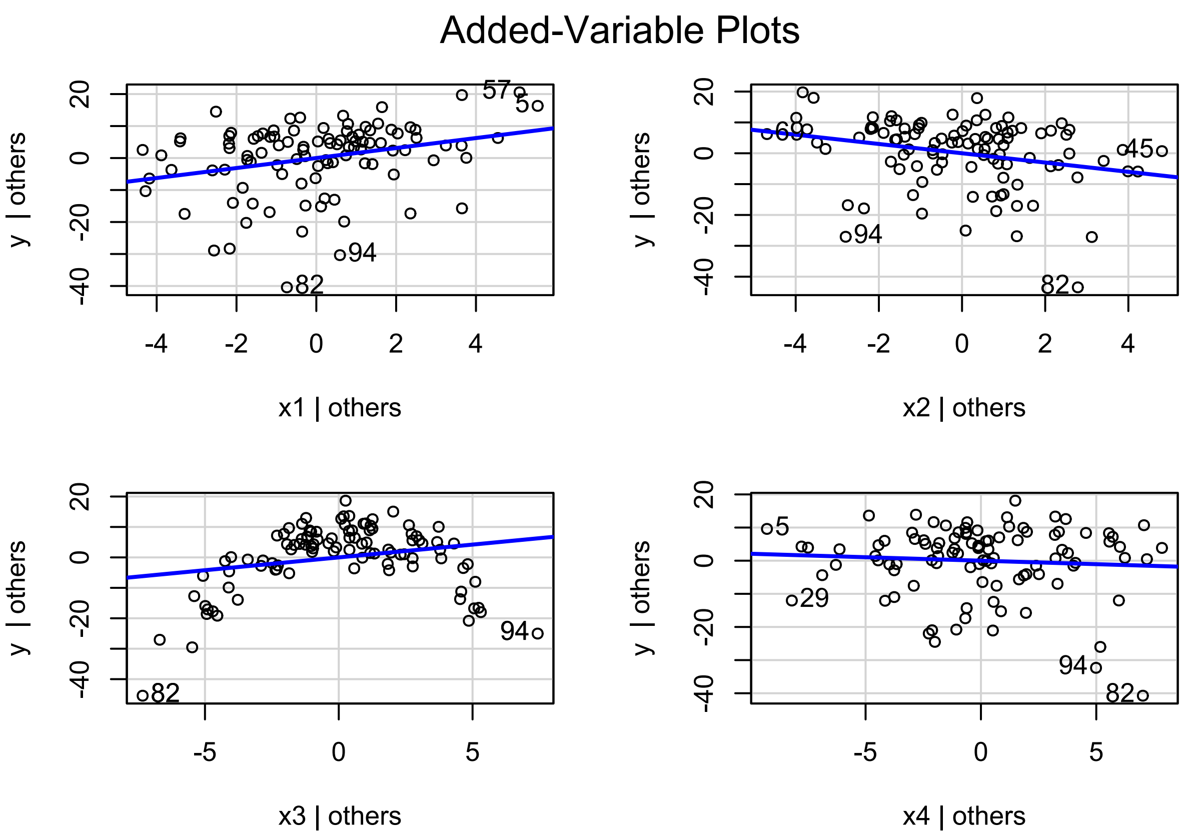


FIGURE 3.12 Added variable plots using the `partialr` data set in the `Data4Ecologists` package (Fieberg 2021) calculated using the `avplots` function in the `car` package (Fox and Weisberg 2019).

One downside to added-variable plots, however, is that the scales on the x- and y-axes do not match the scales of the original variables in the regression model.

3.14.2 Component + residual plots or partial residual plots

Component + residual plots, which are sometimes referred to as partial residual plots, offer a slightly different visualization by plotting:

$$X_i\hat{\beta}_i + \hat{\epsilon}_i \text{ versus } X_i, \text{where}$$

.

X_i is the i^{th} predictor variable and is the variable of interest. As shown, below, $X_i\hat{\beta}_i + \hat{\epsilon}$ represents the part of Y explained by X_i that is not already explained by all the other predictors:

$$Y - \sum_{j\neq i} X_j\hat{\beta}_j = \hat{Y} + \hat{\epsilon} - \sum_{j\neq i} X_j\hat{\beta}_j = X_i\hat{\beta}_i + \hat{\epsilon}$$

There are a number of options in R for creating component + residual plots (see Section 3.14), and the approach can be easily generalized to more complicated models that allow for non-linear relationships (e.g., using quadratic terms) by replacing $X_i\hat{\beta}_i$ with multiple terms corresponding to the columns in the design matrix associated with the i^{th} explanatory variable; however, component + residual plots are not appropriate if you include interactions in the model. Moya-Laraño and Corcobado (2008) suggest that component + residual plots are sometimes better than added variable plots at diagnosing non-linearities, but they are not as good as added-variable plots at depicting the amount of variability explained by each predictor (given everything else in the model).

Component + residual plots using the `crPlots` function in the `car` package are depicted in Figure 3.13.

```
crPlots(lmxall.y)
```

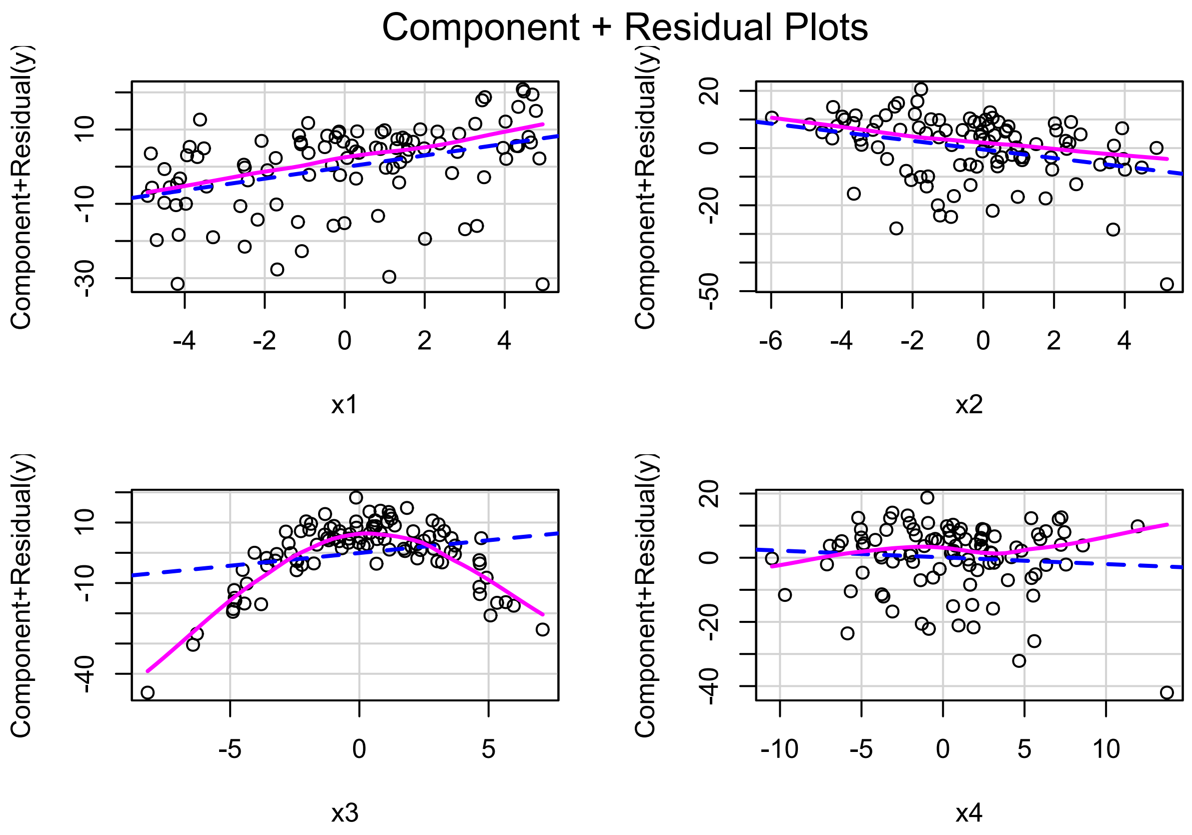


FIGURE 3.13 Component + residual plots constructed using the `partialr` data set in the `Data4Ecologists` package (Fieberg 2021) calculated using the `crPlots` function in the `car` package (Fox and Weisberg 2019).

3.14.3 Effect plots

Another way to visualize fitted regression models is to form effect plots using what Lüdecke (2018) refers to as either *adjusted* or *marginal means*. Plots of adjusted means are formed using predictions where a focal variable is varied over its range of observed values, while all non-focal variables are set to constant values (e.g., at their means or modal values). Marginal means are formed in much the same way, except that predictions are averaged across different levels of each categorical variable. These two types of means are equivalent if there are no categorical predictors in the model.

Marginal means can be calculated using the `effects` function in the `effects` package and then plotted. Alternatively, we can use the `ggeffect` function in the `ggeffects` package (Lüdecke 2018) to format the output and create plots using `ggplot2` (Wickham 2016). Adjusted means can be created using the `ggpredict` function in the `ggeffects` package or the `visreg` function in the `visreg` package (Breheny and Burchett 2013). The `visreg` package also provides an option for producing *contrast* plots, which compare adjusted means to predictions obtained by setting all predictors (including the focal predictor) to specific reference values.

Below, we briefly illustrate the `ggeffect` and `ggpredict` functions using the `RIKZdat` data set and our linear model containing `week` and `NAP` (but not their interaction). If we use `ggpredict` or `ggeffect` with the argument `terms = c("NAP", "week.cat")`, we get predictions for a range of NAP values associated with each week. The output of these functions is a list with an associated `print` function that provides nicely formatted output.

```
pad1 <- ggeffect(lm.ancova, terms = c("NAP", "week.cat"))
pad1
```

```
## # Predicted values of Richness
##
## # week.cat = 1
##
##   NAP | Predicted |         95% CI
## ----------------------------------
## -1.40 |     14.55 | [12.28, 16.82]
## -0.60 |     12.73 | [10.76, 14.70]
##  0.00 |     11.37 | [ 9.46, 13.28]
##  0.80 |      9.55 | [ 7.46, 11.64]
##  2.20 |      6.37 | [ 3.48,  9.27]
##
## # week.cat = 2
##
##   NAP | Predicted |        95% CI
## ---------------------------------
## -1.40 |      6.92 | [ 4.56, 9.28]
## -0.60 |      5.11 | [ 3.24, 6.97]
##  0.00 |      3.74 | [ 2.12, 5.36]
##  0.80 |      1.93 | [ 0.34, 3.52]
##  2.20 |     -1.25 | [-3.51, 1.00]
##
## # week.cat = 3
##
##   NAP | Predicted |         95% CI
## ----------------------------------
## -1.40 |      8.37 | [ 6.04, 10.70]
## -0.60 |      6.55 | [ 4.71,  8.39]
##  0.00 |      5.19 | [ 3.58,  6.80]
##  0.80 |      3.37 | [ 1.78,  4.97]
##  2.20 |      0.19 | [-2.09,  2.48]
##
```

```
## # week.cat = 4
##
##   NAP | Predicted |        95% CI
## ---------------------------------
## -1.40 |     11.95 | [8.65, 15.25]
## -0.60 |     10.14 | [7.21, 13.07]
##  0.00 |      8.77 | [6.01, 11.53]
##  0.80 |      6.96 | [4.25,  9.66]
##  2.20 |      3.78 | [0.68,  6.87]

##
## Not all rows are shown in the output. Use ‘print(..., n = Inf)‘ to show
##   all rows.
```

We can then use a built in `plot` function to visualize these predictions with partial residuals overlaid by adding `residuals = TRUE` (Figure 3.14).

```
plot(pad1, residuals = TRUE, facet = TRUE)
```

```
## Data points may overlap. Use the ‘jitter‘ argument to add some amount of
##   random variation to the location of data points and avoid overplotting.
```

Alternatively, if we want to create a plot just for `NAP`, we can use either `ggpredict` (for adjusted means) or `ggeffect` (for marginal means).

```
padj <- ggpredict(lm.ancova, terms = c("NAP"))
padj
```

```
## # Predicted values of Richness
##
##   NAP | Predicted |         95% CI
## ----------------------------------
## -1.40 |     14.55 | [12.28, 16.82]
## -1.00 |     13.64 | [11.55, 15.73]
## -0.40 |     12.28 | [10.35, 14.21]
##  0.00 |     11.37 | [ 9.46, 13.28]
##  0.40 |     10.46 | [ 8.49, 12.43]
##  0.80 |      9.55 | [ 7.46, 11.64]
##  1.20 |      8.64 | [ 6.37, 10.91]
##  2.20 |      6.37 | [ 3.48,  9.27]
##
## Adjusted for:
## * week.cat = 1

##
## Not all rows are shown in the output. Use ‘print(..., n = Inf)‘ to show
##   all rows.
```

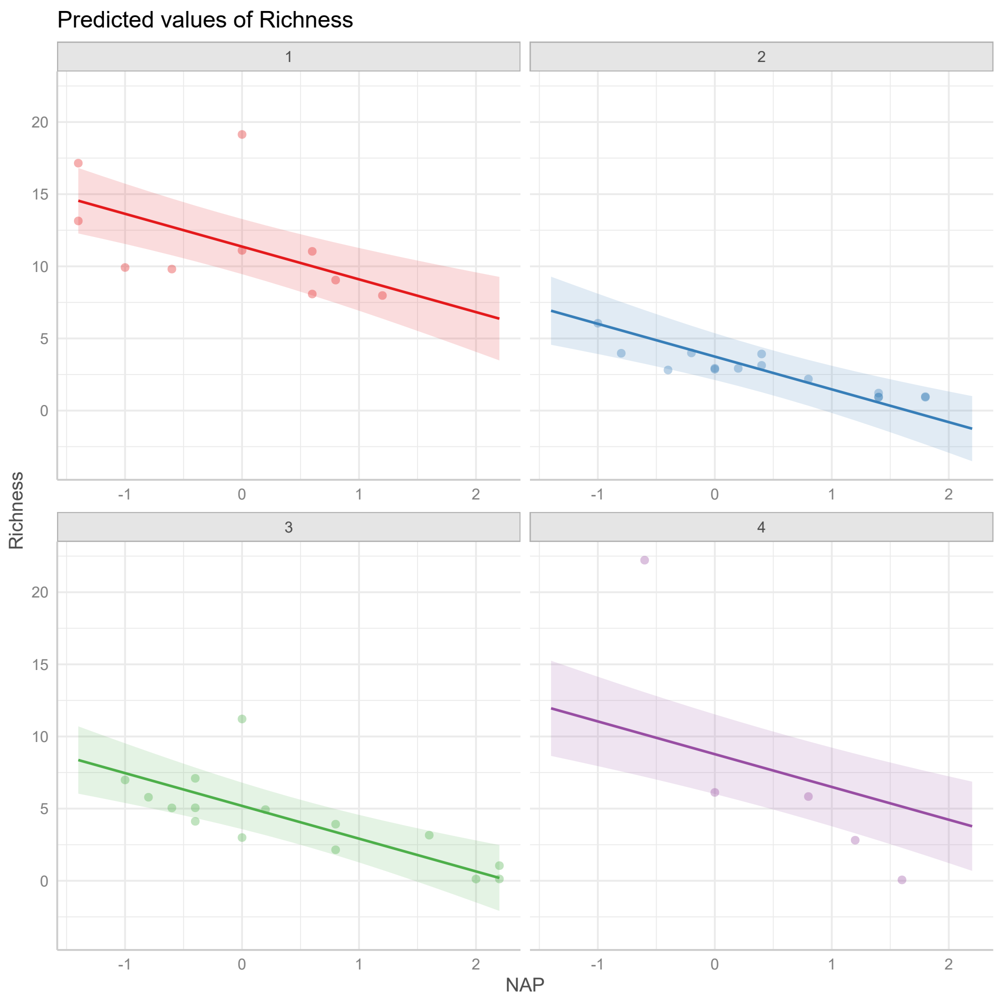


FIGURE 3.14 Effect plot created using ‘ggpredict‘ showing adjusted means for different combinations of ‘week.cat‘ and ‘NAP‘ along with partial residuals.

```
pm <- ggeffect(lm.ancova, terms = c("NAP"))
pm
```

```
## # Predicted values of Richness
##
##   NAP | Predicted |         95% CI
## ----------------------------------
## -1.40 |      9.66 | [ 7.78, 11.54]
## -1.00 |      8.75 | [ 7.19, 10.31]
## -0.40 |      7.39 | [ 6.24,  8.53]
##  0.00 |      6.48 | [ 5.52,  7.44]
##  0.40 |      5.57 | [ 4.67,  6.47]
##  0.80 |      4.66 | [ 3.67,  5.66]
##  1.20 |      3.75 | [ 2.55,  4.96]
##  2.20 |      1.48 | [-0.49,  3.45]
```

```
##
## Not all rows are shown in the output. Use ‘print(..., n = Inf)‘ to show
##   all rows.
```

`ggpredict` forms predictions where `week.cat` is set to 1 (its reference value), whereas `ggeffect` generates predictions for each week, then averages these predictions, weighted by the proportion of observations in each week (for more on these calculations, see Section 16.6.4). As a result, the absolute values of the predictions will differ even though the effect of `NAP` will look similar when we visualize the output (i.e., the slope of the depicted line is the same in both panels of Figure 3.15)

```
library(patchwork)
p1 <- plot(padj, residuals = TRUE, facet = TRUE)
p2 <- plot(pm, residuals = TRUE, facet = TRUE)
p1 + p2
```

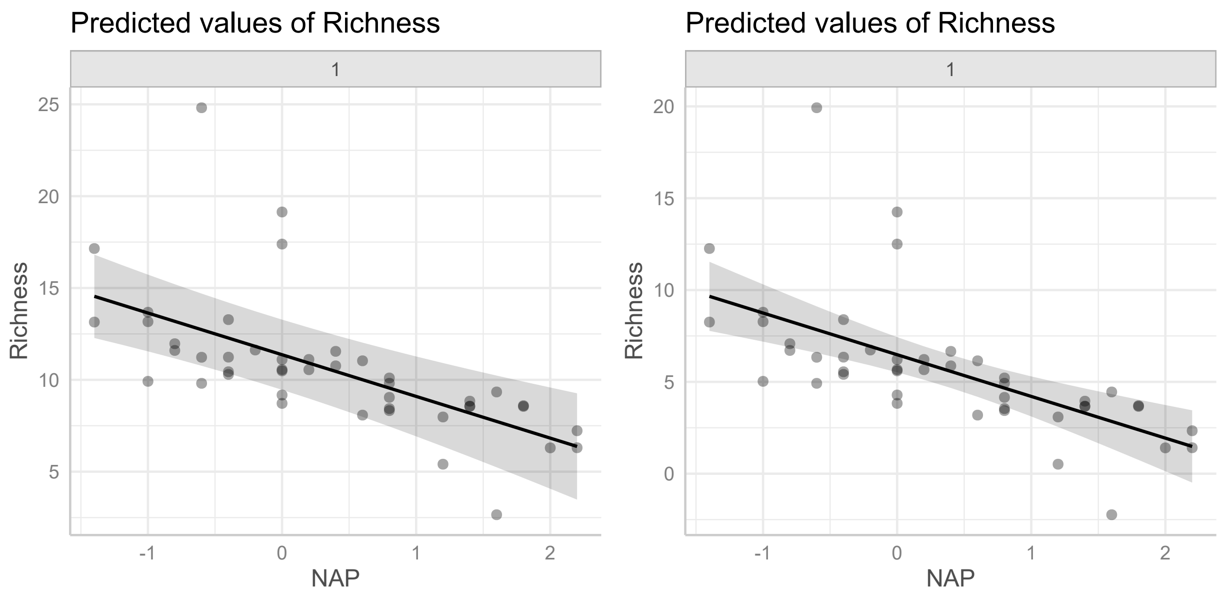


FIGURE 3.15 Effect plots created using `ggpredict` (left) and `ggeffect` (right) showing adjusted and marginal means, with partial residuals.

4

Modeling Non-linear relationships

Learning objectives:

1. Be able to implement common approaches for modeling non-linear relationships between Y_i and X_i:

    - Polynomials using the `poly` function in R
    - Linear regression splines using code you write yourself
    - Cubic regression splines using the `ns` function (`splines` package)
    - Smoothing splines (generalized additive models or GAMS).

2. Understand how model predictions are constructed when using polynomials or splines.

4.1 R Packages

We being by loading several R packages:

```
library(Data4Ecologists) # for the clutch and gala data sets
library(splines) # spline basis functions for fitting non-linear relationships
library(ggplot2) # for plotting
theme_set(theme_bw()) # black and white background
library(gridExtra) # for producing multi-panel plots
library(ggthemes) # for additional ggplot palettes
library(dplyr) # for data wrangling
```

In addition, we will use functions from:

- `ggeffects` for creating effect plots illustrating non-linear relationships
- `mgcv` for fitting generalized additive models
- `car` for F-tests involving coefficients that code for non-linear effects

4.2 Modeling non-linear relationships

A line will often be useful for approximating the relationship between two variables over a given, potentially narrow range of data. However, most relationships are non-linear. As one example, consider the relationship between clutch size and Julian date from data collected on mallard ducks (*Anas platyrhynchos*) in Minnesota (Figure 4.1; Zicus, Fieberg, and Rave 2003). A line will nicely describe the relationship between clutch size and nest initiation date in early spring, but the relationship appears to change around the end of May.

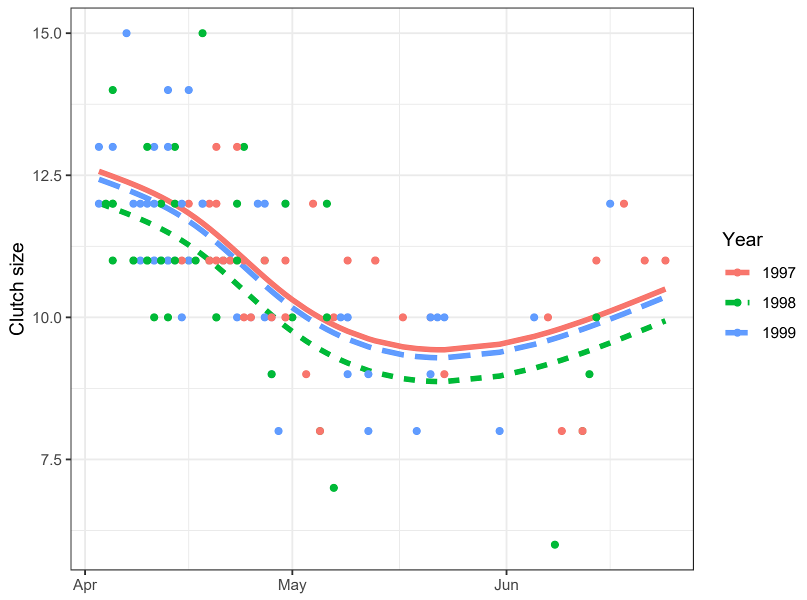


FIGURE 4.1 Clutch size of mallards nesting in nest boxes in Minnesota versus nest initiation date. Data from Zicus, Fieberg, and Rave (2003).

In the above example, the relationship between clutch size and nest initiation date was modeled using a polynomial, but other options are possible. To help understand different approaches to fitting non-linear relationships, we will begin by considering simple examples using data on plant species richness for 29 islands in the Galapagos Islands archipelago (M. P. Johnson and Raven 1973). This section borrows heavily from Jack Weiss's lecture notes from his Statistical Ecology course (which unfortunately are no longer accessible), but also considers additional curve fitting approaches using polynomials and regression splines.

Ecologists have long been interested in how the number of unique species (S) scales with area (A), or so called species-area curves (for an overview, see Conor and McCoy 2013). This interest comes from both a desire to understand the underlying mechanisms that create these relationships as well as from potential applications in research design (e.g., use in developing appropriate sampling designs for describing ecological communities) and conservation and wildlife management (e.g., design of protected areas and estimation of impacts due to habitat loss). Several different statistical models have been proposed for describing species-area curves. We will consider a few of these models here, including:

- $S_i = \beta_0 + \beta_1 A_i + \epsilon_i$ (linear relationship between species and area)
- $S_i = \beta_0 + \beta_1 log(A_i) + \epsilon_i$ (linear relationship between species and log area)
- $S_i = aA_i^b\gamma_i$, which can also be expressed on a log-log scale as $log(S_i) = \beta_0 + \beta_1 log(A_i) + \epsilon_i$ (with $\beta_0 = \log(a)$, $\beta_1 = b$, and $\epsilon_i = \log(\gamma_i)$)

Let's begin by plotting the data on plant species richness relative to area in a few different ways:

```
data(gala)  # in Data4Ecologists package
gala$logarea<- log(gala$Area)
```

```
p1 <- ggplot(gala, aes(x = Area, y = Species)) + geom_point(size = 3)  +
  xlab("Area")
p2 <- ggplot(gala, aes(x = logarea, y = Species)) + geom_point(size = 3)  +
  xlab("log(Area)")
p3<- ggplot(gala, aes(x = logarea, y = log(Species))) + geom_point(size = 3)  +
  xlab("log(Area)")  + ylab("log(Species)")
gridExtra::grid.arrange(p1, p2, p3, ncol=3)
```

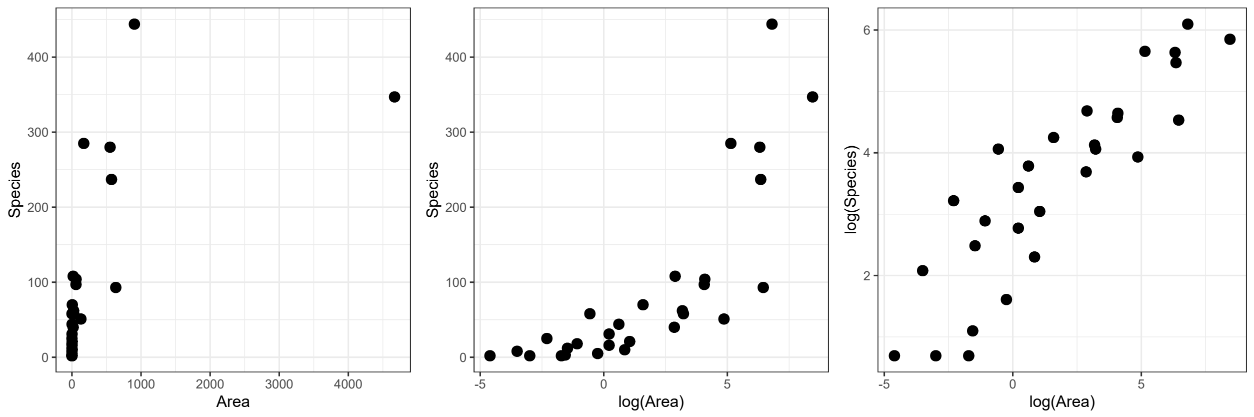


FIGURE 4.2 Plant species richness versus area for 29 islands in the Galapagos Islands archipelago. Data are from (M. P. Johnson and Raven 1973).

We clearly see that the first two models are not appropriate as the relationships between S_i and A_i and between S_i and $\log(A_i)$ are not linear. On the other hand, a linear model appears appropriate for describing the relationship between $\log(S_i)$ and $\log(A_i)$. Below, we fit this model and explore whether the assumptions of the model hold (Figure 4.3):

```
lmloglog <- lm(log(Species) ~ log(Area), data=gala)
summary(lmloglog)
```

```
##
## Call:
## lm(formula = log(Species) ~ log(Area), data = gala)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.44745 -0.40009  0.06723  0.51319  1.45385
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.83384    0.15709  18.040  < 2e-16 ***
## log(Area)    0.40427    0.04121   9.811 2.13e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7578 on 27 degrees of freedom
## Multiple R-squared:  0.7809, Adjusted R-squared:  0.7728
## F-statistic: 96.25 on 1 and 27 DF,  p-value: 2.135e-10
```

```r
performance::check_model(lmloglog,check = c("linearity", "homogeneity", "qq", "normality"))
```

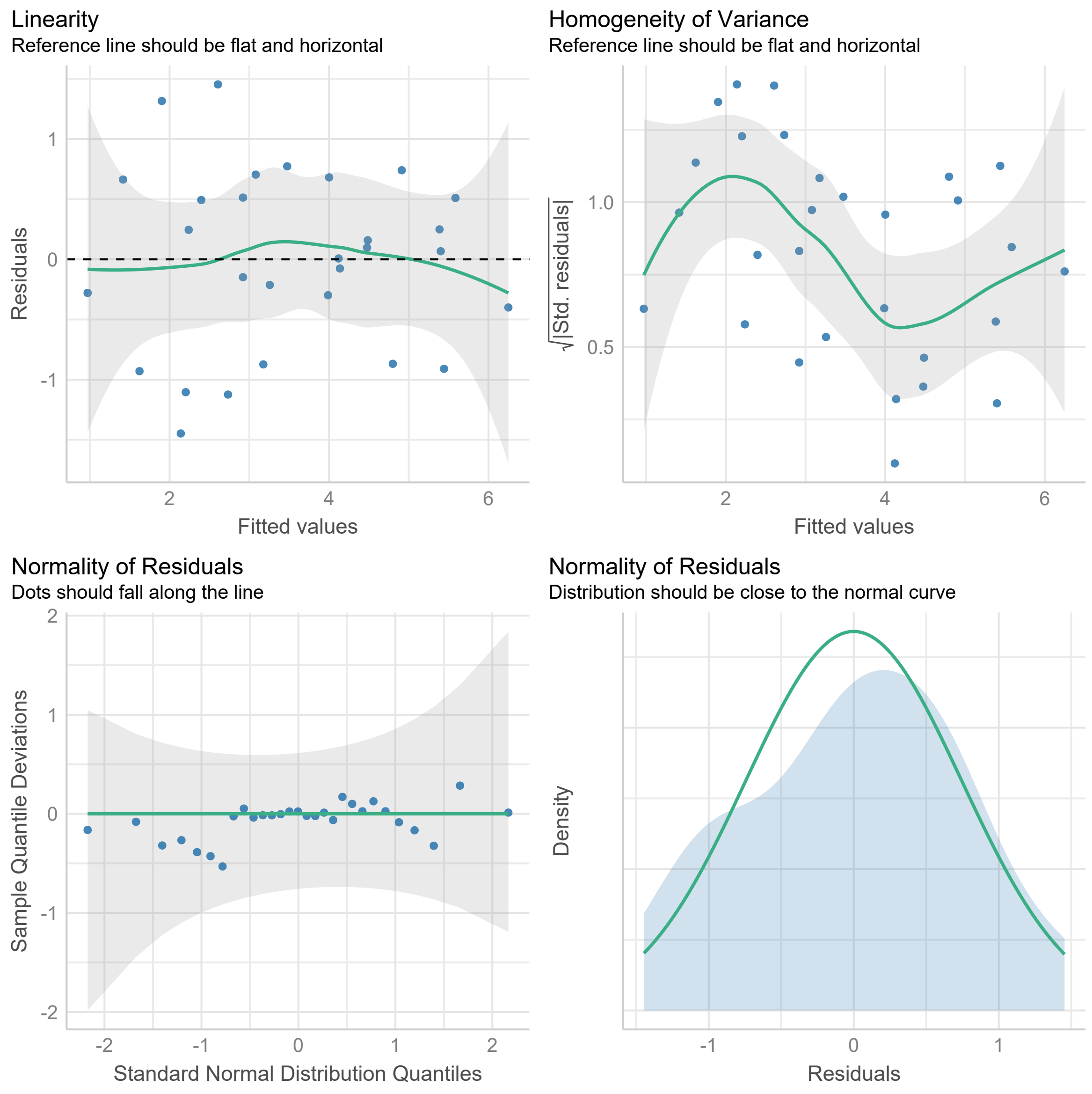


FIGURE 4.3 Residual plots for a linear model relating log plant species richness to log species area for 29 islands in the Galapagos Islands archipelago. Data are from (M. P. Johnson and Raven 1973).

Everything looks OK, except for perhaps the constant variance assumption (i.e., the green line in the upper right panel of Figure 4.3 is not straight). Thus, we might conclude that this is a reasonable model for the data-generating process.

Importantly, this model is a *linear model*, but for the transformed response, $\log(S_i)$ and for the transformed predictor, $\log(A_i)$. This brings up an important point, namely that we can use a linear modeling framework to describe non-linear relationships. By linear model framework, we mean that we have a model where the response variable can be written as a "linear combination" of parameters and predictor variables (i.e., a weighted sum of the predictor variables).

$$Y_i = \beta_0 + X_{i,1}\beta_1 + X_{i,2}\beta_2 + \ldots + \epsilon_i$$

We can accommodate non-linear relationships within a linear models framework by using:

- Transformations of X or Y or both (e.g., $\log(X)$, $\sqrt{Y}$, $\exp(X)$).
- Polynomials (e.g.,including X plus X^2 as predictors)
- Splines or piecewise polynomials that are unique to different segments of the data

I.e., the following models are still technically "linear models" and can be easily fit using `lm`:

$$Y_i = \beta_0 + X_i\beta_1 + X_i^2\beta_2 + \ldots + \epsilon_i$$

$$\sqrt{Y_i} = \beta_0 + log(X_i)\beta_1 + \ldots + \epsilon_i$$

Thus, we can continue to use all the same diagnostic and inferential tools we learned about in Chapter 1 (e.g., residual plots, t-tests, R^2, etc). For the Galapagos data, the log transformation of X and Y results in a reasonable model for the data. The primary downside is that our model describes the relationship between the *log* species richness and *log* area. Although it is tempting to use the inverse transformation (i.e., exp()) to allow inference back on the original scale, doing so will not preserve the mean – i.e., $\exp(E[\log(S)]) \neq E[S]$. Furthermore, the variability about the mean will no longer be described by a normal distribution. Essentially, we have a model that assumes the errors, $\gamma_i = \exp(\epsilon_i)$ are multiplicative and log-normally distributed (Section 9.11.2). It is also important to note that metrics used to compare models (e.g., the AIC; Section 8.4) are not straightforward when considering models for both the original and transformed data (i.e., S_i and $\log(S_i)$). We could use results from Section 9.11.2 to derive the mean back on the original scale or simulate data from our model of the data generating process to facilitate further inference. An alternative approach would be to use a generalized linear model appropriate for count data, which we will eventually see in Section 15. In the next sections, we will consider alternative approaches for curve fitting using polynomials and regression splines.

4.3 Polynomials

If we look at the relationship between `Species` and log(`Area`) (middle panel of Figure 4.2), it appears that a quadratic function might describe the mean species-area curve well. We can explore this model using ggplot by adding `geom_smooth(method="lm", formula=y~poly(x,2), se=TRUE)` (Figure 4.4).

```
ggplot(gala, aes(x=logarea, y=Species)) + geom_point(size=3)+
   geom_smooth(method="lm", formula=y~poly(x,2), se=TRUE)
```

How can we fit this model using `lm`? One intuitive option would be to create a new "predictor" variable in our data set – the square of `logarea` and add this predictor to the linear model:

```
gala$logarea.squared <- gala$logarea^2
lm.poly <- lm(Species ~ logarea + logarea.squared, data = gala)
summary(lm.poly)
```

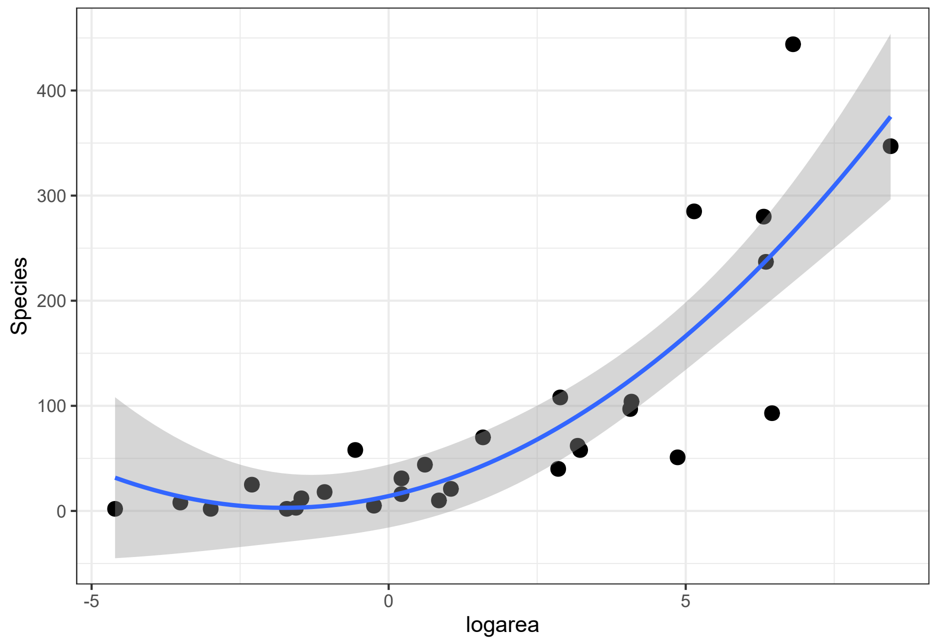


FIGURE 4.4 Quadratic polynomial fit to species-area relationship.

```
Call:
lm(formula = Species ~ logarea + logarea.squared, data = gala)

Residuals:
     Min       1Q   Median       3Q      Max
-151.009  -27.361   -1.033   20.825  178.805

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      14.1530    14.5607   0.972 0.340010
logarea          12.6226     4.8614   2.596 0.015293 *
logarea.squared   3.5641     0.9445   3.773 0.000842 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 59.88 on 26 degrees of freedom
Multiple R-squared:  0.7528,    Adjusted R-squared:  0.7338
F-statistic:  39.6 on 2 and 26 DF,  p-value: 1.285e-08
```

Think-Pair-Share: Write down equations for this model using matrix notation. Include the specific elements of the design matrix for the first few observations in the data set below:

```
head(gala, 3)
```

```
##      Island Species Endemics  Area Elevation Nearest.dist Santacruz.dist
## 1    Baltra      58       23 25.09       100          0.6            0.6
## 2 Bartolome      31       21  1.24       109          0.6           26.3
## 3  Caldwell       3        3  0.21       114          2.8           58.7
##   Adjacent.isl.area    logarea logarea.squared
## 1              1.84  3.2224694     10.38430878
## 2            572.33  0.2151114      0.04627291
## 3              0.78 -1.5606477      2.43562139
```

A better, option would be to use R's built in `poly` function within the call to `lm`.

```
lm.poly1.raw <- lm(Species ~ poly(logarea, 2, raw = TRUE), data = gala)
summary(lm.poly1.raw)
```

```
Call:
lm(formula = Species ~ poly(logarea, 2, raw = TRUE), data = gala)

Residuals:
     Min       1Q   Median       3Q      Max 
-151.009  -27.361   -1.033   20.825  178.805 

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)    
(Intercept)                    14.1530    14.5607   0.972 0.340010    
poly(logarea, 2, raw = TRUE)1  12.6226     4.8614   2.596 0.015293 *  
poly(logarea, 2, raw = TRUE)2   3.5641     0.9445   3.773 0.000842 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 59.88 on 26 degrees of freedom
Multiple R-squared:  0.7528,    Adjusted R-squared:  0.7338 
F-statistic:  39.6 on 2 and 26 DF,  p-value: 1.285e-08
```

We get the exact same fit, but as we will see later, the advantage of this approach is that various functions used to summarize the model will know that `logarea` and its squared term "go together." As such, we can more easily test for the combined linear and non-linear effect of `logarea` (e.g., using `Anova` in the `car` package), and we can construct a meaningful component + residual plot using `termplot` (see e.g., Section 4.11 for more on constructing models using polynomial terms and an explanation of what the argument `raw = TRUE` is doing).

4.4 Basis functions/vectors

Let's dig deeper to see what is going on under the hood in our polynomial model:

$$Y_i = \beta_0 + \beta_2 X_i + \beta_3 X_i^2 + \epsilon_i$$
$$\epsilon_i \sim N(0, \sigma^2)$$

We see that our predicted response, $E[Y_i|X_i] = \hat{Y}_i$ is formed by taking a weighted sum of X_i and X_i^2, with the weights given by the regression coefficients for X_i and X_i^2:

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_2 X_i + \hat{\beta}_3 X_i^2 \tag{4.1}$$

More generally, we can write any *linear model* as a sum of parameters β_j times predictors, where some of these predictors are constructed from the original predictors (i.e. they are a function of the original predictors):

$$Y_i = \sum_{j=1}^{P} \beta_j b_j(X_i) + \epsilon_i$$

The $b_j(X_i)$ are called basis functions or basis vectors and are calculated using our original variable, $X =$ `logarea` (Hefley et al. 2017). Our polynomial model can be written in terms of 3 basis vectors $b_j(X_i) = 1, X_i,$ and X_i^2 (noting that $1 = X^0$). We plot these 3 basis vectors against `logarea` (our X) in Figure 4.5.

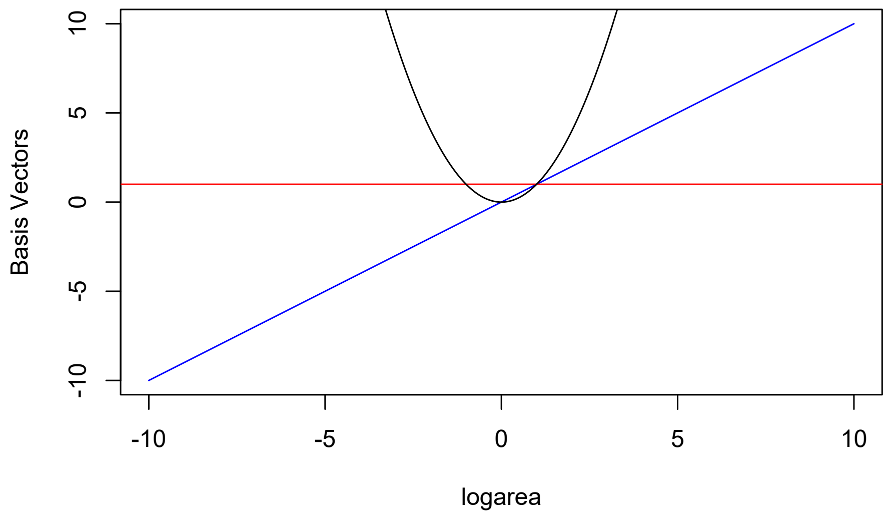


FIGURE 4.5 Set of basis vectors in the quadratic polynomial model relating plant species richness to log(Area) and log(Area)^2 for 29 islands in the Galapagos Islands archipelago. Data are from (M. P. Johnson and Raven 1973).

Let's look at our design matrix using `model.matrix` for the first 6 observations and also the first 6 values of `logarea`.

```
head(model.matrix(Species~ poly(logarea,2, raw=TRUE), data=gala))
```

```
  (Intercept) poly(logarea, 2, raw = TRUE)1 poly(logarea, 2, raw = TRUE)2
1           1                     3.2224694                   10.38430878
2           1                     0.2151114                    0.04627291
3           1                    -1.5606477                    2.43562139
4           1                    -2.3025851                    5.30189811
5           1                    -2.9957323                    8.97441185
6           1                    -1.0788097                    1.16383029
```

```r
gala$logarea[1:6]
```

```
[1]  3.2224694  0.2151114 -1.5606477 -2.3025851 -2.9957323 -1.0788097
```

We see that `poly` created two variables in our design matrix, X and X^2 corresponding to `logarea` and `logarea*logarea`.

Our predicted species richness at any given value of `logarea`, $E[Y_i|X_i] = \hat{Y}_i$ (eq. (4.1)), is formed by weighting the 3 columns of our design matrix by their regression coefficients:

$$E[Y_i|X_i] = \beta_0 1 + \beta_2 X_i + \beta_3 X_i^2 = 14.15 + 12.62\log(Area)_i + 3.56\log(Area)_i^2$$

In other words, $E[Y_i|X_i]$ is given by a linear combination of the horizontal line (1, from the intercept), a line through the origin (X), and the quadratic centered on the origin (X^2) as depicted in Figure 4.5. Note that a negative coefficient for X^2 would result in a downward-facing parabola. Thus, the quadratic polynomial model can fit any quadratic function, not just those that are concave up.

More generally, a model with polynomial of degree D in X can be written as a linear combination of basis vectors representing powers of X up to D:

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \ldots + \beta_D X_i^D + \epsilon_i$$

Specific polynomials:

- **Linear**: $Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$
- **Quadratic**: $Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \epsilon_i$
- **Cubic**: $Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \beta_3 X_i^3 + \epsilon_i$

The design matrix for a regression model with n observations and a polynomial of degree D for X will be given by:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & \ldots & x_1^D \\ 1 & x_2 & x_2^2 & x_2^3 & \ldots & x_2^D \\ 1 & x_3 & x_3^2 & x_3^3 & \ldots & x_3^D \\ & & \vdots & & & \\ 1 & x_n & x_n^2 & x_n^3 & \ldots & x_n^D \end{bmatrix}$$

Higher order polynomials can allow for more complex patterns, but they have “global constraints” on their shape. For example, quadratic functions must eventually either increase (or decrease) as X gets both really large or small. Cubic polynomials must move in opposite directions (i.e., increase and decrease) as X gets really large or really small, etc. Thus, there are some disadvantages to using polynomials to model non-linear relationships.

Splines allow us to gain further flexibility by combining polynomials applied separately to short segments of our data along with local constraints that ensure the different polynomials get “stitched” together in a reasonable way.

4.5 Splines

Splines are piecewise polynomials used in curve fitting. They may seem mysterious at first, but like polynomials, they can be understood as models that weight different basis vectors constructed from predictor variables included in the model.

4.5.1 Linear splines

Linear models are often a good approximation over small ranges of X. Thus, one option for modeling nonlinear relationships is to "piece together" several linear models fit to short segments of the data, taking care to make sure that the "ends meet" at the boundaries of each segment.

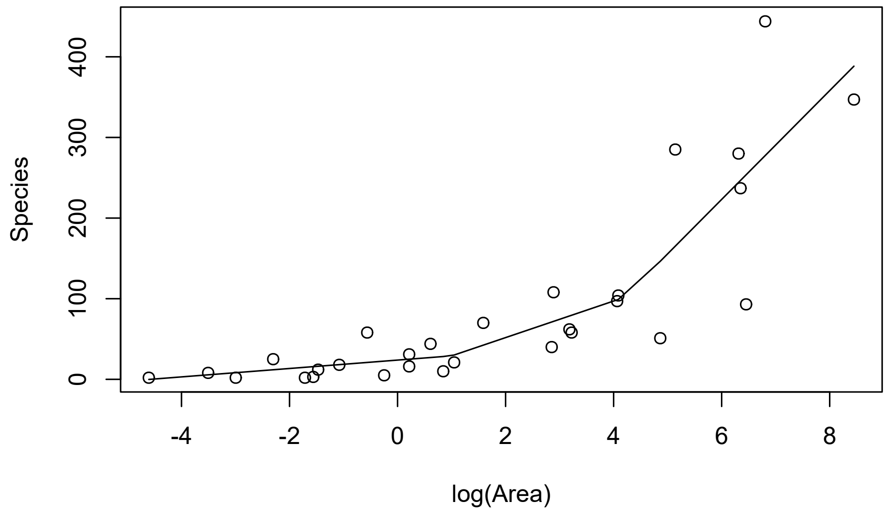


FIGURE 4.6 Piecewise linear model relating plant species richness to log(Area) for 29 islands in the Galapagos Islands archipelago. Data are from M. P. Johnson and Raven (1973).

The above plot was constructed by fitting a model with the effect of `logarea` modeled using a linear spline. A linear spline is a continuous function formed by connecting linear segments. The points where the segments connect are called the knots of the spline.

We chose 2 knots (located at `logarea` = 1 and 4.2, where the relationship between `Species` and `logarea` appeared to change) and formed two basis vectors by subtracting the knot from `logarea` and setting the value to 0 whenever the expression was negative:

- `(logarea-1)`$_+$
- `(logarea-4.2)`$_+$

where the $_+$ indicates that we set the value equal to 0 when the expression was negative. Here is how we created the basis vectors in R:

```
gala$logarea<- log(gala$Area)
gala$logarea.1<- ifelse(gala$logarea<1, 0, gala$logarea-1)
gala$logarea.4.2<- ifelse(gala$logarea<4.2, 0, gala$logarea-4.2)
```

We then fit our model using `lm`:

```
lm.sp<-lm(Species~logarea+logarea.1+logarea.4.2, data=gala)
summary(lm.sp)
```

```
##
## Call:
## lm(formula = Species ~ logarea + logarea.1 + logarea.4.2, data = gala)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -160.691  -16.547   -4.209   13.133  166.430
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   23.869     17.384   1.373   0.1819
## logarea        5.213      8.956   0.582   0.5658
## logarea.1     17.464     18.836   0.927   0.3627
## logarea.4.2   44.815     23.156   1.935   0.0643 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 58.97 on 25 degrees of freedom
## Multiple R-squared:  0.7695, Adjusted R-squared:  0.7418
## F-statistic: 27.82 on 3 and 25 DF,  p-value: 3.934e-08
```

As with the polynomial model, our predicted species richness at any given value of `logarea`, $E[Y_i|X_i] = \hat{Y}_i$, is formed by summing the weighted columns of our design matrix by their regression coefficients. The basis vectors, their weighted versions, and the fitted curve formed by summing the weighted basis vectors are depicted in Figure 4.7.

Linear splines can sometimes be useful for modelling abrupt changes in response variables or to depict relationships where we might expect a response variable to asymptote after crossing some threshold value of the predictor variable. For example, Thompson et al. (2015) used linear splines with a constraint that the slope be 0 after a single knot to model the effect of disturbance from oil and gas wells on bird abundance (Figure 4.8).

4.5.2 Cubic regression splines

Linear splines (D = 1) only ensure our function is continuous. The first derivative, which describes if the function is increasing or decreasing, is not constant. As a result, our fitted curve (Figure 4.6) looks somewhat jagged at the knot locations. In fact, the fitted model can be increasing before a knot and decreasing after it! Usually, we expect biological relationships to be smoother than this. Therefore, unless we have an expectation that Y will change drastically after crossing a threshold, we will generally benefit from approaches that result in a smoother fit to the data. One way to accomplish this goal is to use a higher order spline.

We can model non-linear relationships by connecting polynomial segments of degree D such that:

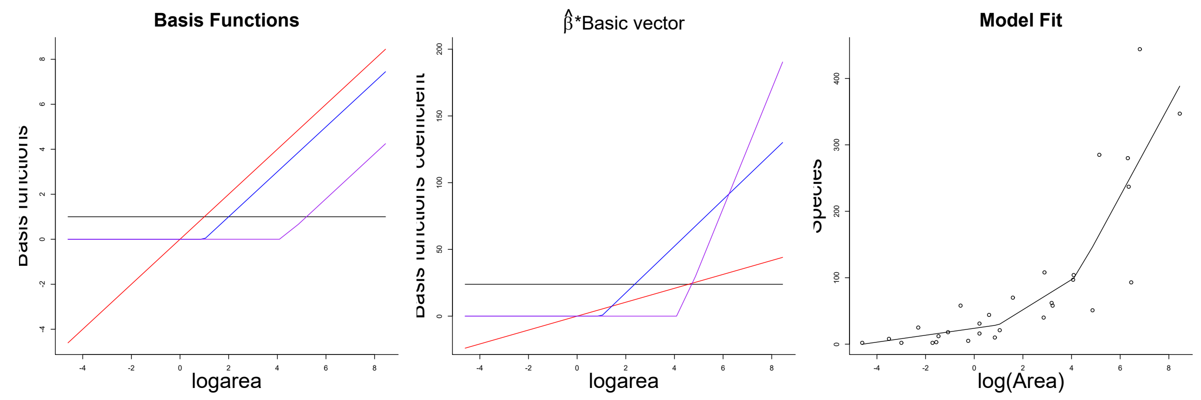


FIGURE 4.7 Basis vectors, weighted basis vectors, and predicted values formed by summing the weighted basis vectors in the linear spline model relating plant species richness to log(Area) for 29 islands in the Galapagos Islands archipelago. Data are from M. P. Johnson and Raven (1973).

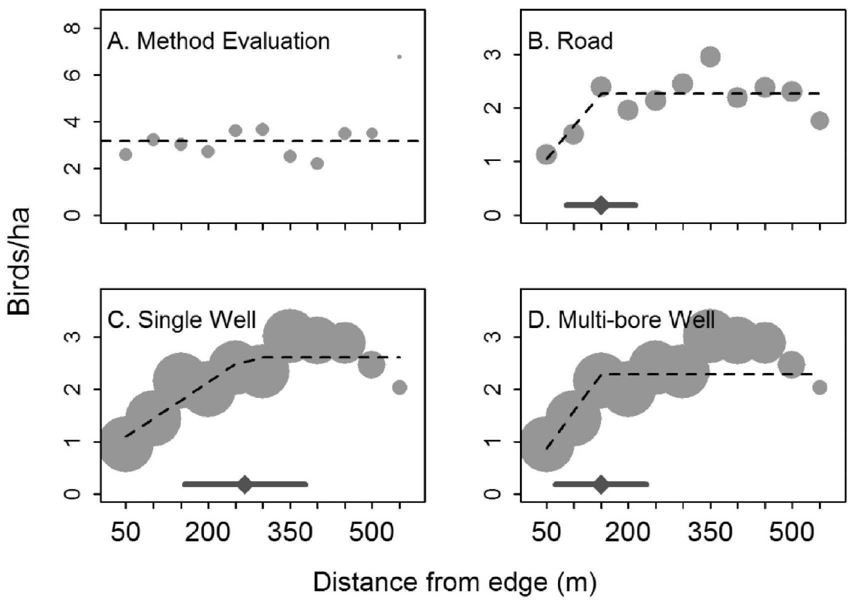


FIGURE 4.8 Figure 2 from Thompson et al. (2015) describing models used to explore the effect of oil and gas disturbance on grassland bird abundance in North Dakota.

- the function is continuous (i.e., it is not disjointed when it hits a knot)
- the function has D-1 continuous derivatives
- the Dth derivative is constant between the knots

Cubic splines (D = 3) are commonly used in model fitting, and ensure that $E[Y|X]$ and its first two derivatives are continuous everywhere (even at the knot locations). Remember, the first derivative (tells us if the function is increasing or decreasing), whereas the second derivative tells us about curvature – i.e., whether the function is concave up (positive second derivative), concave down (negative second derivative), or at a minimum or maximum (second derivative is equal to 0).

There are a number of different ways of creating basis vectors that lead to piecewise cubic polynomials with continuous first and second derivatives. The simplest approach is to use truncated polynomials, which is how we created our linear spline. We can create a truncated polynomial basic vector of degree D associated with a knot location ξ_k by setting its value equal to 0 to the left of ξ_k and equal to $(x-\xi_k)^D$ to the right of ξ_k.

$$(x-\xi_k)_+^D = \begin{cases} 0 & \text{if } x < \xi_k \\ (x-\xi_k)^D & \text{if } x \geq \xi_k \end{cases}$$

We could then fit a model that includes these basis vectors:

$$Y_i = \beta_0 + \sum_{d=1}^{D} \beta_D X_i^d + \sum_{k=1}^{K} b_k (X_i - \xi_k)_+^D + \epsilon_i$$

The design matrix for a cubic spline with K knots formed in this way would be given by:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & (x_1-\xi_1)_+^3 & \dots & (x_1-\xi_k)_+^3 \\ 1 & x_2 & x_2^2 & x_2^3 & (x_2-\xi_1)_+^3 & \dots & (x_2-\xi_k)_+^3 \\ 1 & x_3 & x_3^2 & x_3^3 & (x_3-\xi_1)_+^3 & \dots & (x_3-\xi_k)_+^3 \\ & & \vdots & & & & \\ 1 & x_n & x_n^2 & x_n^3 & (x_n-\xi_1)_+^3 & \dots & (x_n-\xi_k)_+^3 \end{bmatrix}$$

Although truncated power bases are relatively easy to understand and construct, they may lead to numerical problems due to scaling issues (Perperoglou et al. 2019). Other popular options are:

- Bsplines, which can be constructed using the `bs(x, df=)` functino in `splines` package
- Natural or restricted cubic splines constructed using the `ns(x, df=)` function in `splines` package

B-splines are numerically more stable than those based on the truncated power basis but can be poorly behaved near the edge of the data range. Natural or restricted cubic splines address this problem by assuming the model is linear before the first knot and after the last knot. A further advantage of these additional constraints is that it requires fewer basis vectors and thus, fewer model degrees of freedom than bsplines with the same number of knots. We will return to this point when we get to the section on model selection (Section 8).

In addition to b-splines and natural splines, cyclical splines can be used to model circular data like time of day or annual response patterns (e.g., Ditmer et al. 2015). In these cases, we would want our model to give similar predictions just before versus just after midnight or just before versus after a new year begins. Cyclical splines use basis vectors that are formed in a way that ensures these constraints are upheld.

4.5.3 Natural cubic regression splines

To fit a model with the effect of `logarea` on species richness modeled using a natural cubic regression spline, we use the `ns` function in the `splines` package. Using the second argument, `df`, we can specify the degrees of freedom that we want to use when modeling the effect of the predictor. This will equate to the number of coefficients estimated in the model.

```
lm.ns <- lm(Species ~ ns(logarea, df = 3), data = gala)
summary(lm.ns)
```

```
Call:
lm(formula = Species ~ ns(logarea, df = 3), data = gala)
```

```
Residuals:
     Min       1Q   Median       3Q      Max 
-156.173  -13.819   -5.998   13.922  170.555 

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)    
(Intercept)             1.468     43.542   0.034   0.9734    
ns(logarea, df = 3)1   47.790     45.957   1.040   0.3084    
ns(logarea, df = 3)2  276.125    102.146   2.703   0.0122 *  
ns(logarea, df = 3)3  381.743     45.084   8.467 8.22e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 59.48 on 25 degrees of freedom
Multiple R-squared:  0.7655,    Adjusted R-squared:  0.7374 
F-statistic: 27.21 on 3 and 25 DF,  p-value: 4.859e-08
```

We see from the output that we have estimated an intercept and 3 coefficients associated with `logarea`. Let's look at the design matrix for the first few observations and a plot of the different basis functions created by `ns` (i.e., the columns of the design matrix) versus `logarea`.

```
head(model.matrix(lm.ns))
```

```
##    (Intercept) ns(logarea, df = 3)1 ns(logarea, df = 3)2 ns(logarea, df = 3)3
## 17           1           0.00000000            0.0000000            0.0000000
## 8            1          -0.07289332            0.1993098           -0.1235349
## 5            1          -0.09952908            0.2856200           -0.1770311
## 4            1          -0.12204050            0.3907838           -0.2422131
## 9            1          -0.12443447            0.4653235           -0.2884138
## 3            1          -0.12199115            0.4821792           -0.2988612
```

The predicted species richness values are again formed by weighting the basis functions in Figure 4.9 along with the intercept. We can use `ggplot` to see the resulting fitted curve (Figure 4.10).

```
ggplot(gala, aes(x = logarea, y = Species)) + geom_point(size = 3) +
  geom_smooth(method = "lm", formula = y ~ ns(x, df = 3), se = TRUE)
```

4.5.4 Number of knots and their locations

The shape of a spline can be controlled by carefully choosing the number of knots and their exact locations in order to allow greater flexibility where the trend appears to be changing quickly, while avoiding overfitting in regions where the trend changes little. We could, in principle, compare models (e.g., using AIC) that have varying numbers of knots, or different knot locations. The danger here is that we would open the door to a wide range of possible models, increasing the chances we would overfit our data. Further, proper accounting of model-selection uncertainty associated with such data-driven processes can be challenging, a topic we will discuss in Chapter 8.

An alternative strategy is to choose a small number of knots (df), based on how much data you have and

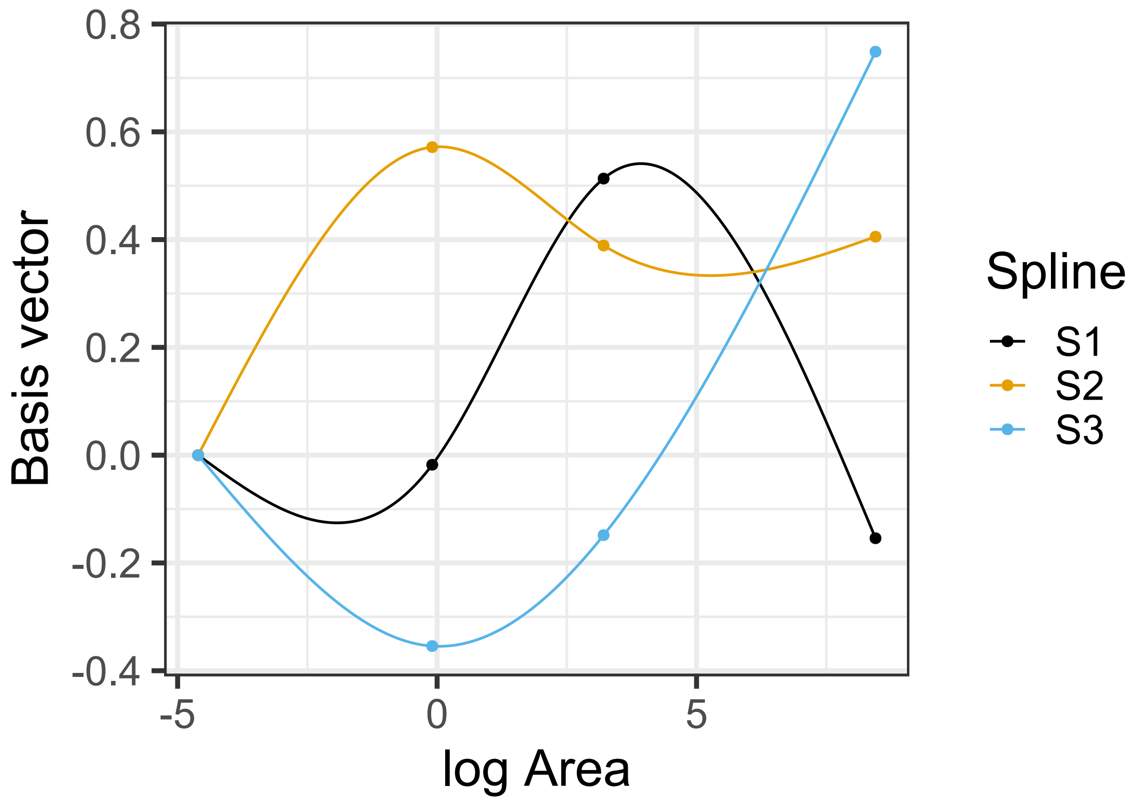


FIGURE 4.9 Basis vectors used in the fitting of the natural cubic regression spline model relating plant species richness to log(Area) for 29 islands in the Galapagos Islands archipelago. Data are from (M. P. Johnson and Raven 1973). Dots indicate knot locations. Boundary knots were chosen using the minimum and maximum values in the data set. Interior knots were chosen using the 0.33 and 0.67 quantiles of the data. For more on these choices, see Section 4.5.4.

how complex you expect the relationship to be *a priori* (Harrell Jr 2015). I have personally found that 2 or 3 internal knots are usually sufficient for small data sets. Also, Keele and Keele (2008), cited in Zuur et al. (2009), recommend 3 knots if $n < 30$ and 5 knots if $n > 100$. As for choosing the locations of the knots, we can do this using quantiles of the data (what **ns** does by default if you do not provide knot locations). Fortunately, models fit with cubic regression splines are *usually* not too sensitive to knot locations (Harrell Jr 2015). Alternatively, you can choose to place knots in places where you expect the relationship between X and Y to be changing rapidly based on familiarity with the system.

We can see the knots chosen by **ns** using:

```
attr(ns(gala$logarea,3), "knots")
```

```
[1] -0.09393711  3.20877345
```

```
attr(ns(gala$logarea,3), "Boundary.knots")
```

```
[1] -4.605170  8.448769
```

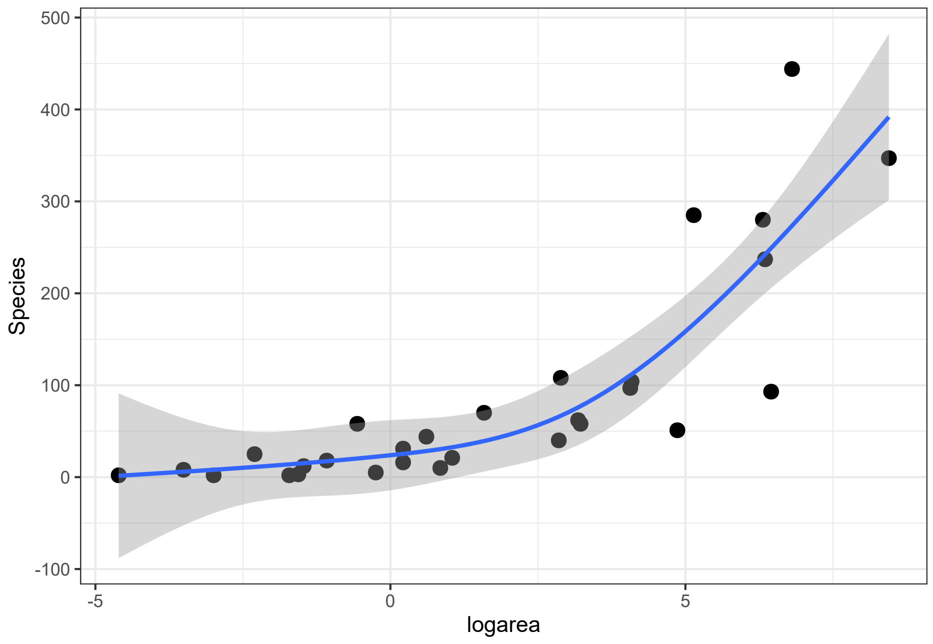


FIGURE 4.10 Predicted values from the natural cubic regression spline model relating plant species richness to log(Area) for 29 islands in the Galapagos Islands archipelago. Data are from M. P. Johnson and Raven (1973).

The first two knots are often referred to as internal knots, whereas the latter 2 knots are located at the outer limits of the data (remember, the model is assumed to be linear outside of this range):

```
range(gala$logarea)
```

```
## [1] -4.605170  8.448769
```

4.6 Splines versus polynomials

We have not yet talked about how we can compare competing models, but one popular method is to use the Akaike's Information Criterion or AIC (Akaike 1974; Anderson and Burnham 2004). Smaller values of AIC are generally preferable.

```
lmfit <- lm(Species ~ logarea, data = gala)
AIC(lmfit, lm.poly1.raw, lm.sp, lm.ns)
```

```
             df      AIC
lmfit         3 335.1547
lm.poly1.raw  4 324.4895
lm.sp         5 324.4646
lm.ns         5 324.9600
```

```
# lmfit = linear model
# lm.poly1.raw = model with quadratic
# lm.sp = linear spline model
# lm.ns = cubic regression spline model
```

Here, we see that all of the non-linear approaches we have so far considered fit better than a linear model. Although the natural cubic regression spline and polynomial models have similar AIC values, the spline model seems to provide a better fit for especially small values of `logarea` (Figure 4.11). In particular, the polynomial model appears to suggest species richness will stop decreasing and actually start increasing once `logarea` decreases past a certain point and approaches the smallest values in our data set, which does not seem reasonable. This is an inherent limitation of quadratic polynomials, which all have to eventually point in the same direction at the extremes of the data. By contrast, splines offer more "local control" when fitting deviations from linearity.

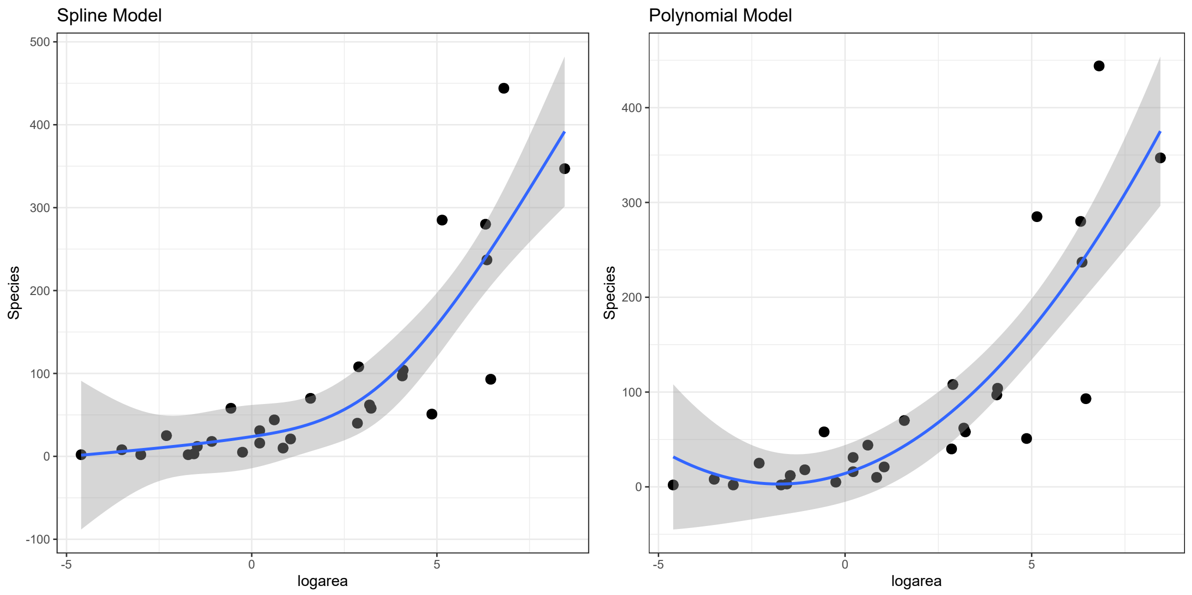


FIGURE 4.11 Comparison of predicted values from the natural cubic regression spline and polynomial models fit to plant species richness data collected from 29 islands in the Galapagos Islands archipelago. Data are from M. P. Johnson and Raven (1973).

4.7 Visualizing non-linear relationships

In the previous sections, we used `ggplot` to visualize our fitted models (e.g., Figure 4.10). This worked since our model only included a single predictor, `logarea`. More generally, we can use `termplot` to create a partial residual plot (see Section 3.14.2) depicting the effect of `logarea` while controlling for other predictors that may be included in the model:

```
termplot(lm.ns, se = T, partial = T, pch = 16, main = "Partial Residual Plot")
```

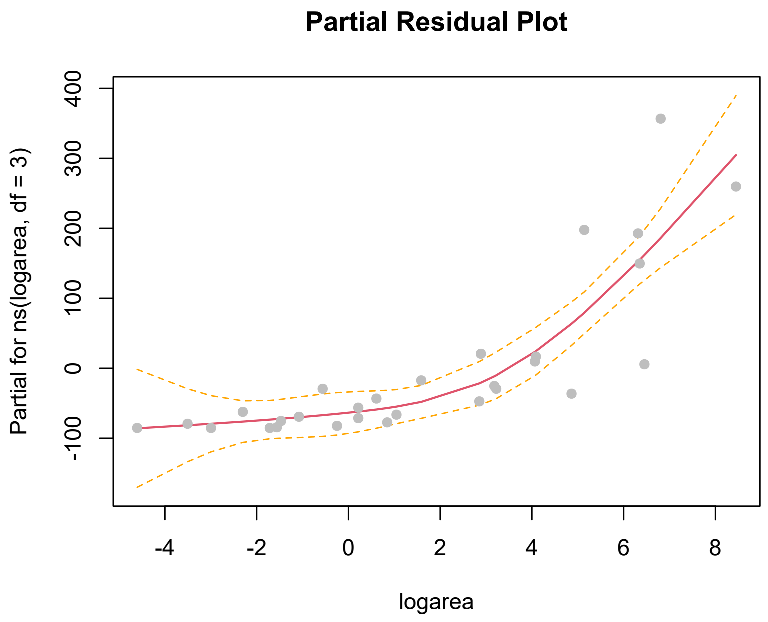


FIGURE 4.12 Partial residual plot depicting the effect of `logarea` on species richness in the natural cubic regression spline model fit to plant species richness data collected from 29 islands in the Galapagos Islands archipelago. Data are from M. P. Johnson and Raven (1973).

Another option is to use the `ggeffects` package (Lüdecke 2018) to produce effect plots (see Section 3.14.3; Figure 4.13).

```
library(ggeffects)
plot(ggpredict(lm.ns, terms = "logarea"), residuals = TRUE)
```

4.8 Generalized additive models (GAMS)

In general, non-linear models, including those that we have discussed so far with a single predictor, can be described via the following equation:

$$Y = \beta_0 + f(x_i) + \epsilon_i$$

where $f(x_i)$ represents an unknown function relating X to the expected value of Y. In this section, we will briefly explore an approach for estimating $f()$ using smoothing splines, since this approach is popular among ecologists and related to the regression spline approach we covered in the last section. A primary difference between these approaches is that with regression splines, we pre-specify a level of model complexity (using the

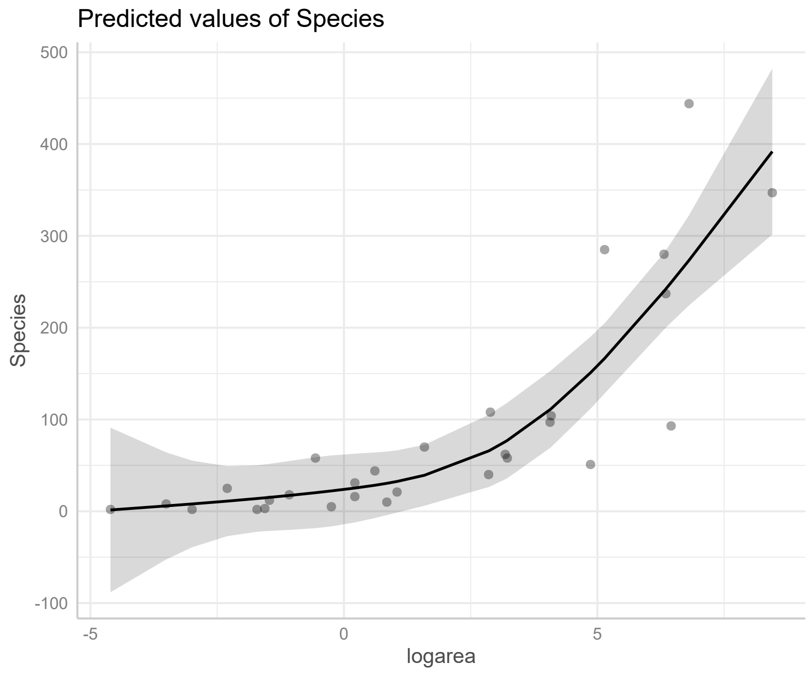


FIGURE 4.13 Partial residual plot depicting the effect of `logarea` on species richness in the natural cubic regression spline model fit to plant species richness data collected from 29 islands in the Galapagos Islands archipelago. Data are from M. P. Johnson and Raven (1973).

degrees of freedom), whereas smoothing splines use the data to determine an appropriate degree of flexibility needed to model $f()$. Smoothing splines are constructed using lots of knots/basis vectors (created using the same methods we saw in the last section), but the coefficients are "shrunk" towards 0 in an attempt to balance overfitting and smoothness. Although smoothing splines are perhaps easiest to understand philosophically when fitted in a Bayesian framework (they arise naturally from certain prior assumptions about the level of smoothness), they can also be implemented using penalized likelihoods, or since we have not yet covered maximum likelihood yet, penalized sums of squares:

$$PSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \int f''(x)dx$$

or, equivalently:

$$PSS = \sum_{i=1}^{n}(y_i - \beta_0 + f(x_i))^2 + \lambda \int f''(x)dx \tag{4.2}$$

Here, $f''(x)$ is the second derivative of $f(x)$ and describes how smooth the curve is (high values are indicative of a lot of 'wigglyness'), and λ is a tuning parameter that determines the penalty for 'smoothness' (larger values will result in less 'wigglyness', and as $\lambda \to \infty$, $f(x)$ will approach a straight line).

In this framework, how should we determine λ? Typically, this is accomplished using cross validation in which we seek to minimize the sum-of-squared prediction errors (see e.g., Section 8.8.1):

$$SSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{f}^{[-i]}(x_i))^2$$

where $\hat{f}^{[-i]}(x_i)$ is the fit of the model using all observations except for observation i. In practice, there are mathematical expressions/approximations that allow us to choose λ without fitting the model n times.

We can fit a GAM model, allowing for a non-linear relationship between species richness and log(Area) using the `gam` function in `mgcv` package (S. N. Wood 2004; S. N. Wood 2017):

```
library(mgcv)
sfit <- gam(Species ~ s(logarea), data = gala)
summary(sfit)
```

```

Family: gaussian
Link function: identity

Formula:
Species ~ s(logarea)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    87.34      11.11   7.864 2.76e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
             edf Ref.df     F p-value
s(logarea) 2.465  3.107 24.86  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.734   Deviance explained = 75.8%
GCV = 4062.7  Scale est. = 3577.3    n = 29
```

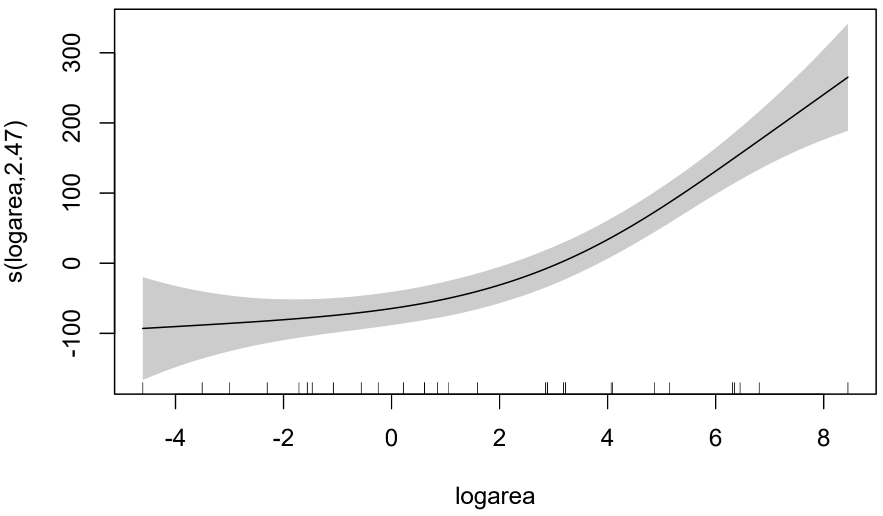
The small p-value for `s(logarea)` tells us that there is a relationship between `logarea` and `Species` and the associated `edf` tells us that this relationship required approximately `2.5` effective model degrees of freedom (`edf`) to describe it accurately. Note, however, because GAMs use the data to choose λ (and effective model degrees of freedom), the p-values can be misleading; S. N. Wood (2017) suggests p-values are often too small and should be multiplied by a factor of 2 (see also discussion in Ch 3.6 of Zuur et al. 2009).

To understand the model, it is best to explore the fit of the model using the associated `plot` function.

```
plot(sfit, scheme=1)
```

Besides the assumption of linearity, which we have relaxed, we otherwise have the same distributional assumptions as in any linear regression model, namely, that errors are normally distributed with constant variance. We can use the `gam.check` function to explore whether these assumptions are met.

```
par(mfrow=c(2,2))
gam.check(sfit)
```



```
Method: GCV   Optimizer: magic
Smoothing parameter selection converged after 6 iterations.
The RMS GCV score gradient at convergence was 0.0002855077 .
The Hessian was positive definite.
Model rank =  10 / 10

Basis dimension (k) checking results. Low p-value (k-index<1) may
indicate that k is too low, especially if edf is close to k'.

             k'  edf k-index p-value
s(logarea) 9.00 2.47    1.43    0.98
```

The plots produced by the `gam.check` function (Figure 4.14) suggest that the error variance increases with $\hat{Y}$ (top right panel), and the residuals do not appear to be Normally distributed (left panels). As with the RIKZ data, we are modeling a species count and would likely be better off using a distribution other than the Normal distribution (as we will see when we get to generalized linear models in Chapter 14).

As mentioned previously, the model is fit by creating a large number of basis vectors and then estimating the coefficients for those basis vectors using the penalized sum of squares in equation (4.2). The large p-value associated with the "Basis dimension check" suggests that the default choice for number of basis vectors (9 here) was sufficient for modeling the relationship between `logarea` and `Species.` If this p-value is small, then we would want to use more knots. If you are interested in learning more about GAMS, we highly recommend Noam Ross's online and free GAM class.

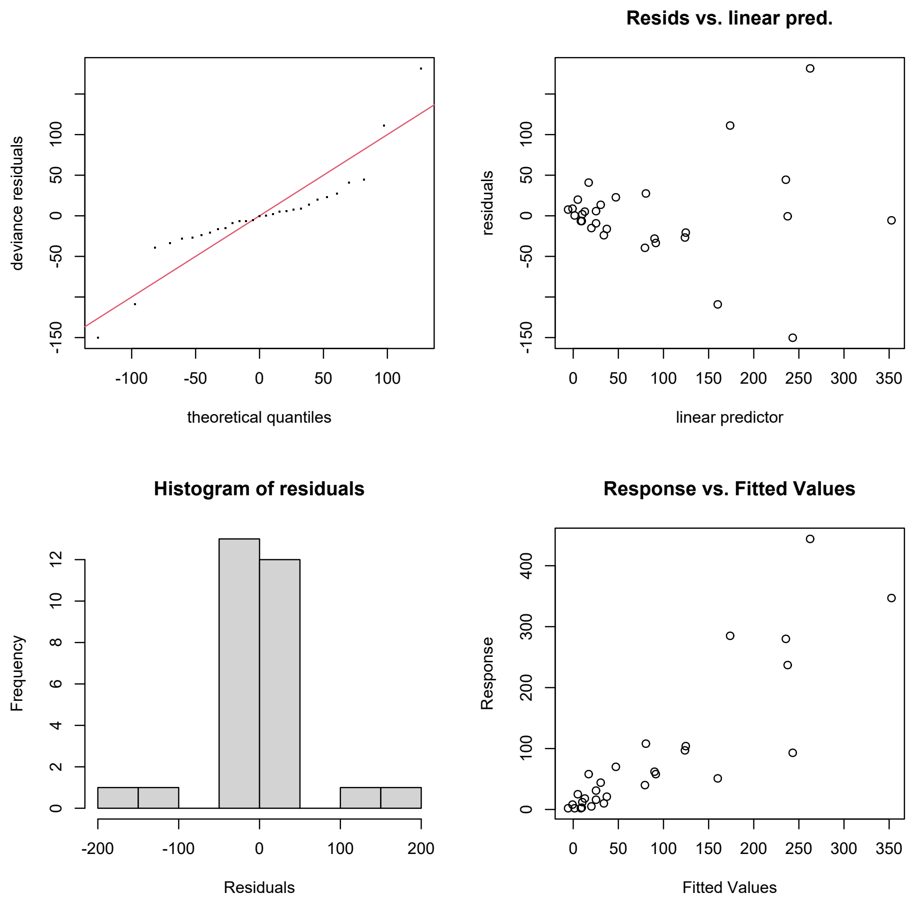


FIGURE 4.14 Model diagnostics for the fitted GAM model produced using the ‘gam.check‘ function.

4.9 Generalizations to multiple non-linear relationships

What if you want to allow for multiple non-linear relationships? We can add multiple `ns` terms to our model or use multiple smoothing splines (see Zuur et al. 2009, chap. 3; S. N. Wood 2017). We could also consider other basis functions to fit smooth surfaces (allowing for interactions between variables), including tensor splines, thin plate splines, etc... (see S. N. Wood 2017). It is also possible to include interactions with regression splines or to allow each level of a categorical variable to have its own smooth when using smoothing splines (see Zuur et al. 2009, chap. 3).

4.10 Non-Linear models with a mechanistic basis

Sometimes we may have a theoretical model linking X and Y. For example, we may want to assume $Y \sim f(x, \beta)$, where $f(x, \beta)$ is given by, for example:

- The Ricker model for stock-recruitment: $S_{t+1} = S_t e^{r(1-\beta S_t)}$
- A typical predator-prey relationship: $f(N) = \frac{aN}{1+ahN}$

Some of these models can be fit using non-linear least squares in R (e.g., using the `nls` function in base R). Alternatively, we will eventually learn how to fit custom models using Maximum likelihood and Bayesian methods (see Section 10.7).

4.11 Aside: Orthogonal polynomials

Standard polynomial-basis vectors can cause numerical issues when fitting linear models due to differences in scale. For example, if $X = 100$, then $X^3 = 1,000,000$. As a result, the coefficients weighting the X and X^3 terms can be on very different scales, which can make them hard to estimate accurately. In such cases, centering and scaling X prior to deriving polynomial terms (i.e., creating a new variable for use in the model by subtracting the mean of X and diving through by the standard deviation of X) can help.

Alternatively, we can use orthogonal polynomials created using `poly(raw=FALSE)` (the default). This will create a different set of basis functions for fitting a quadratic polynomial, but one that leads to an identical fit as the raw polynomial model. We can verify this by comparing predicted values and statistical tests for the effect of `logarea` modeled with both raw and orthogonal polynomials (Figure 4.15).

```
lm.poly.raw<-lm(Species~poly(logarea,2, raw=TRUE), data=gala)
lm.poly.orth<-lm(Species~poly(logarea,2), data=gala)
plot(fitted(lm.poly.raw), fitted(lm.poly.orth), xlab="Normal Polynomials",
     ylab="orthogonal Polynomials")
abline(0,1)
```

.

```
library(car)
Anova(lm.poly1.raw) #log(Area)+ I(log(Area)^2)
```

```
Anova Table (Type II tests)

Response: Species
                             Sum Sq Df F value    Pr(>F)    
poly(logarea, 2, raw = TRUE) 283970  2  39.596 1.285e-08 ***
Residuals                     93232 26                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

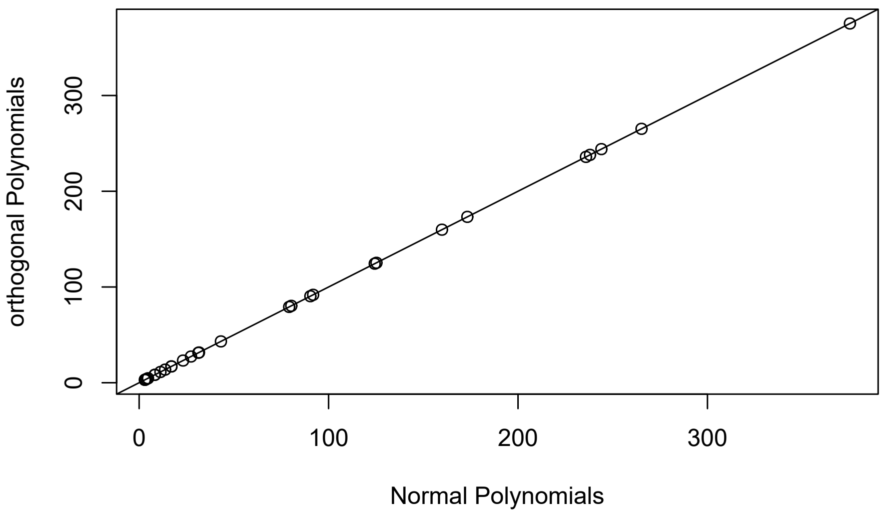


FIGURE 4.15 Predicted values for models using raw and orthogonal polynomials, demonstrating their equivalence.

```
Anova(lm.poly.orth) # poly(logarea,2)
```

```
Anova Table (Type II tests)

Response: Species
                 Sum Sq Df F value    Pr(>F)
poly(logarea, 2) 283970  2  39.596 1.285e-08 ***
Residuals         93232 26
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Let's look at the basis vectors used to model the effect of `logarea` in the two models (Figure 4.16):

```
head(poly(gala$logarea,2), 3)
```

```
##               1         2
## [1,] -0.3425801 0.4850314
## [2,] -0.2828395 0.2782420
## [3,] -0.2550617 0.1950582
```

```
head(cbind(gala$logarea, gala$logarea^2),3)
```

```
##           [,1]      [,2]
## [1,] -4.605170 21.207592
## [2,] -3.506558 12.295948
## [3,] -2.995732  8.974412
```

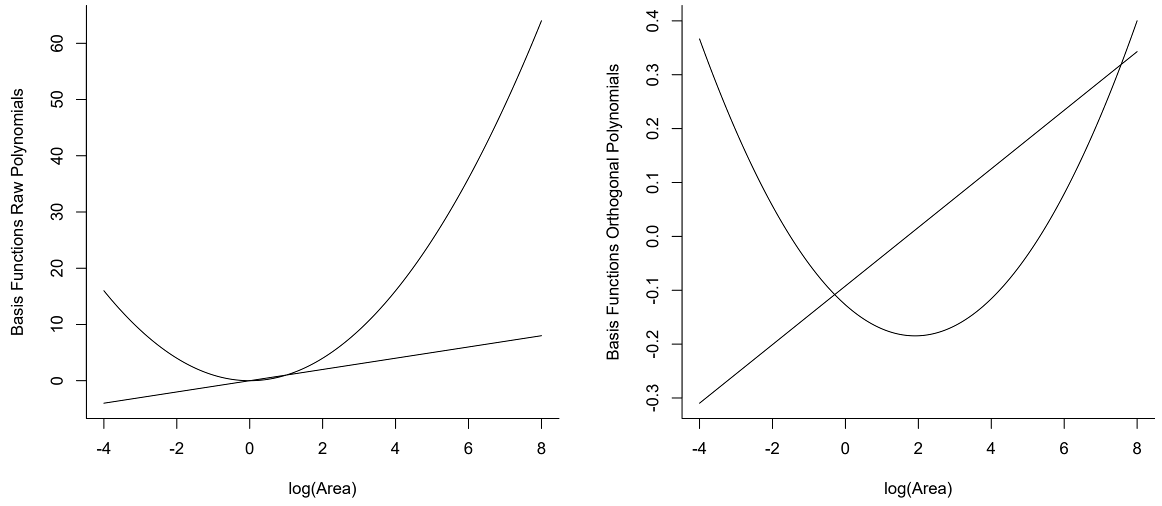


FIGURE 4.16 Basic vectors used to fit raw and orthogonal polynomial models.

We see that raw and orthogonal polynomials use different basis vectors. Both sets of vectors span the set of all quadratic polynomial functions, meaning that we can form any quadratic polynomial using a weighted sum of either set of basis vectors (for more, see Hefley et al. 2017). So, while they result in different parameter estimates, the models and fitted values are identical!

4.12 Aside: Segmented and piecewise regression

Segmented and piecewise regression offer a similar approach to splines in that one can construct models by splicing together polynomials fit to different segments of predictor space, but with the knots treated as free parameters that are to be estimated from the data (Toms and Lesperance 2003; Wolfson, Andersen, and Fieberg 2022). Although we won't cover these methods here, these models can be fit in a frequentist or Bayesian framework, e.g., using either the `segmented` (Muggeo et al. 2014) or `mcp` (Lindeløv 2020) packages in R, respectively.

5

Generalized least squares (GLS)

Learning Objectives

1. Learn how to use generalized least squares (GLS) to model data where $Y_i|X_i$ is normally distributed, but the variance of the residuals is not constant and may depend on one or more predictor variables.
2. Gain additional practice describing models using equations where model parameters are written as a function of explanatory variables and associated regression coefficients.

5.1 R Packages

We begin by loading a few packages:

```r
library(nlme) # for the gls function used to fit gls models
library(dplyr) # for data wrangling
library(ggplot2) # for plotting
theme_set(theme_bw()) # black and white background
library(ggthemes) # for colorblind palette
library(gridExtra) # for multi-panel plots
```

In addition, we will explore the `sockeye` data set from the `Data4Ecologists` package.

5.2 Why cover models fit using generalized least squares?

There are a few reasons why I like to cover this class of model:

1. There are many data sets where a linear model would be appropriate *except* for the assumption that the error variance is constant. Thus, I have found GLS models to be a very useful tool to have in one's toolbox.
2. The GLS framework can be used to model data where the residuals are not independent due to having multiple measurements on the same sample unit (see Section 18). In addition, it can be used to model data that are temporally or spatially correlated (though we will not cover those methods in this book).
3. Lastly, GLS models will give us additional practice with describing models using a set of equations. Linear models allow us to write the mean of the Normal distribution as a function of one or more

predictor variables. GLS models require that we also consider that the variance of the Normal distribution may depend on one or more predictor variables.

5.3 Generalized least squares (GLS): Non-constant variance

In Chapter 1, we considered models of the following form:

$$Y_i = \beta_0 + X_{1i}\beta_1 + X_{2i}\beta_2 + \dots X_{pi}\beta_p + \epsilon_i$$

where the ϵ_i were assumed to be independent, Normally distributed, and have constant variance. This allowed us to describe our model for the data using:

$$Y_i \sim N(\mu_i, \sigma^2)$$
$$\mu_i = \beta_0 + X_{1i}\beta_1 + X_{2i}\beta_2 + \dots X_{pi}\beta_p$$

In this chapter, we will consider models where the variance also depends on covariates, leading to the following model specification below:

$$Y_i \sim N(\mu_i, \sigma_i^2)$$
$$\mu_i = \beta_0 + X_{1i}\beta_1 + X_{2i}\beta_2 + \dots X_{pi}\beta_p$$
$$\sigma_i^2 = f(Z_i; \delta)$$

,

where σ_i is written as a function, $f(Z_i; \tau)$, of predictor variables, Z, and additional variance parameters, δ; Z may overlap with or be distinct from the predictors used to model the mean (i.e., X).

The `nlme` package in R (Jose Pinheiro et al. 2021) has several built in functions for modeling the variance, some of which are highlighted below:

- `varIdent`: $\sigma_i^2 = \sigma_g^2$ (allowing for different variances for each level of a categorical variable, g)
- `varFixed`, `varPower`, `varExp`, and `varConstPower`: $\sigma_i^2 = \sigma^2 X_i$, $\sigma^2|X_i|^{2\delta}$, $\sigma^2 e^{2\delta X_i}$, and $\sigma^2(\delta_1 + |X_i|^{\delta})^2$ (models where the variance scales with a continuous covariate, X)
- $\sigma_i^2 = \sigma^2 E[Y_i|X_i]^{2\theta} = \sigma^2(\beta_0 + X_{1i}\beta_1 + X_{2i}\beta_2 + \dots X_{pi}\beta_p)^{2\theta}$ (models where the variance depends on the mean)

We will explore these three different options in the next few sections. For a thorough description of available choices in `nlme`, see José Pinheiro and Bates (2006) and Zuur et al. (2009).

5.4 Variance heterogeneity among groups: T-test with unequal variances

To allow for unequal variances, let's begin by considering a really simple example, namely extending the linear model used to test for differences in the mean jaw lengths of male and female jackals from Section 3.6.1. Specifically, we will consider the following model:

$$Y_i \sim N(\mu_{sex_i}, \sigma^2_{sex_i}) \tag{5.1}$$

We can fit this model using the `gls` function in the `nlme` package (Jose Pinheiro et al. 2021). To do so, we specify the variance model using the `weights` argument:

```
gls_ttest <- gls(jaws ~ sex, weights = varIdent(form = ~ 1 | sex), data = jawdat)
```

The argument's name, `weights`, reflects the fact that the variance function is used to weight each observation when fitting the model. Specifically, estimates of regression parameters are obtained by minimizing:

$$\sum_{i=1}^{n} \frac{(Y - \mu_i)^2}{\sigma_i^2}$$

In other words, instead of minimizing the sum-of-squared errors, we minimize a weighted version where each observation is weighted by the inverse of its variance.

There are a number of different variance models that we can fit, each with its own variance function. In this case, we have used the `varIdent` variance function, which allows the variance to depend on a categorical variable, in this case, `sex`. The model is parameterized using multiplicative factors, δ_g, for each group g (i.e., each unique level of the categorical variable), with δ_g set equal to 1 for a reference group:

$$\sigma_i^2 = \sigma^2 \delta^2_{sex_i} \tag{5.2}$$

Let's use the `summary` function to inspect the output of our GLS model so that we can match our parameter estimates to the parameters in our model (eq. (5.1)):

```
summary(gls_ttest)
```

```
## Generalized least squares fit by REML
##   Model: jaws ~ sex
##   Data: jawdat
##        AIC      BIC    logLik
##   102.0841 105.6456 -47.04206
##
## Variance function:
##  Structure: Different standard deviations per stratum
##  Formula: ~1 | sex
##  Parameter estimates:
##         M         F
## 1.0000000 0.6107279
##
## Coefficients:
##             Value Std.Error   t-value p-value
## (Intercept) 108.6 0.7180211 151.24903  0.0000
## sexM          4.8 1.3775993   3.48432  0.0026
##
##  Correlation:
##      (Intr)
## sexM -0.521
```

```
##
## Standardized residuals:
##         Min          Q1         Med          Q3         Max
## -1.72143457 -0.70466511  0.02689742  0.76657633  1.77522940
##
## Residual standard error: 3.717829
## Degrees of freedom: 20 total; 18 residual
```

Think-pair-share: Use the output above to estimate the mean jaw length for males and females. Based on the output, do you think the jaw lengths are more variable for males or females?

Like the `lm` function, `gls` will, by default, use effects coding to model differences between males and females in their *mean* jaw length:

$$\mu_i = \beta_0 + \beta_1 \text{I(sex=male)}_i$$

If you have forgotten how to estimate the mean for males and females from the output of the model, I suggest reviewing Section 3.6.1.

To get estimates of the variances for males and females, note that the residual standard error will be used to estimate σ in equation (5.2), i.e., $\hat{\sigma} = 3.72$. Also, note that the estimates of the δ's are given near the top of the summary output. We see that whereas `female` serves as our reference group when estimating coefficients associated with the mean part of the model, `male` serves as our reference group in the variance model and is thus assigned a $\delta_m = 1$. For females, we have $\delta_f = 0.61$. Putting the pieces together and using equation (5.2), we have:

- for males: $\hat{\sigma}_i^2 = \hat{\sigma}^2\delta_m^2 = \hat{\sigma}^2 1 = 13.82$.
- for females: $\hat{\sigma}_i^2 = \hat{\sigma}^2\delta_f^2 = \hat{\sigma}^2 0.61^2 = 5.16$.

We can verify that these estimates are reasonable by calculating the variance of the observations for each group:

```
jawdat %>% group_by(sex) %>%
  dplyr::summarise(var(jaws))
```

```
## # A tibble: 2 x 2
##   sex   `var(jaws)`
##   <chr>       <dbl>
## 1 F            5.16
## 2 M           13.8
```

5.4.1 Comparing the GLS model to a linear model

Is the GLS model an improvement over a linear model? Good question. It may depend on the purpose of your model. It does allow us to relax the assumption of constant variance. And, it allows us to assign higher weights to observations that are associated with parts of the predictor space that are less variable (this will make more sense when we consider models that allow the variance to depend on continuous variables; Section 5.5). This may lead to precision gains when estimating regression coefficients if our variance model is a good one. Lastly, it may provide a better model of the data generating process, which can be particularly important when trying to generate accurate prediction intervals (see Section 5.6).

At this point, you may be wondering about whether or not we can formally compare the two models. The answer is that we can, but with some caveats that will be discussed in Section 18.12. One option is to use AIC, which will give a conservative test, meaning that it will tend to "prefer" simpler models. We won't formally define AIC until after we have learned about maximum likelihood. Yet, AIC is commonly used to compare models and can, for now, be viewed as a measure of model fit that penalizes for model complexity. In order to make this comparison, we will need to first fit the linear model using the `gls` function. By default, `gls` uses restricted maximum likelihood to estimate model parameters rather than maximum likelihood (these differences will be discussed when we get to Section 18.12). In this case, AIC is nearly identical for the two models.

```
lm_ttest <- gls(jaws ~ sex, data = jawdat)
AIC(lm_ttest, gls_ttest)
```

```
##           df      AIC
## lm_ttest   3 102.1891
## gls_ttest  4 102.0841
```

We will further discuss model selection strategies in the context of GLS models in Section 5.7 and more generally in Section 8.

5.4.2 Aside: Parameterization of the model

When fitting the model, the `gls` function actually parameterizes the variance model on the log scale, which is equivalent in this case to:

$$log(\sigma_i) = log(\sigma) + log(\delta)\text{I}(\text{sex} = \text{female})_i \tag{5.3}$$

We can see this by extracting the parameters that are actually estimated:

```
(varpars<-attr(summary(gls_ttest)$apVar, "Pars"))
```

```
##  varStruct     lSigma
## -0.4931038  1.3131400
```

If we exponentiate the second parameter, we get $\hat{\sigma}$:

```
exp(varpars[2])
```

```
##   lSigma
## 3.717829
```

If we exponentiate the first parameter, we get the multiplier of σ for females, δ_f:

```
exp(varpars[1])
```

```
## varStruct
## 0.6107279
```

And, we get the standard deviation for females by exponentiating the sum of the two parameters:

```
exp(varpars[1]+varpars[2]) #sigma_f
```

```
## varStruct
##  2.270582
```

```
(exp(varpars[1]+varpars[2]))^2 # sigma^2_f^2
```

```
## varStruct
##  5.155543
```

Why is this important? When fitting custom models, it often helps to have parameters that are not constrained - i.e., they can take on any value between $-\infty$ and ∞. Whereas σ must be > 0, $\log(\sigma)$ can take on any value. We will return to this point when we see how to use built in optimizers in R to fit custom models using maximum likelihood (Section 10.5.4).

5.5 Variance increasing with X_i or μ_i

In many cases, you may encounter data for which the variance increases as a function of one of the predictor variables or as a function of the predicted mean. We will next consider one such data set, which comes from a project I worked on when I was a Biometrician at the Northwest Indian Fisheries Commission.

Aside: Many of the tribes in the Pacific Northwest and throughout North America signed treaties with the US government that resulted in the cessation of tribal lands but with language that acknowledged their right to hunt and fish in their *usual and accustomed areas*. Many tribal salmon fisheries were historically located near the mouth of rivers or further up stream, whereas commercial and recreational fisheries often target fish at sea. Salmon spend most of their adult life in the ocean only coming back to rivers to spawn, and then, for most salmon species, to die. As commercial fisheries became more prevalent and adopted technologies that increased harvest efficiencies, and habitat became more fragmented and degraded, many salmon stocks became threatened and endangered in the mid 1900s. With fewer fish coming back to spawn, the State of Washington passed laws aimed at curtailing tribal harvests. The tribes fought for their treaty rights in court, winning a series of cases. The most well known court case, the Boldt Decision in 1974, reaffirmed that the treaty tribes, as sovereign nations, had the right to harvest fish in their *usual and accustomed areas*. Furthermore, it established that the tribes would be allowed 50% of the harvestable catch. The Northwest Indian Fisheries Commission was created to provide scientific and policy support to treaty tribes in Washington, and to help with co-management of natural resources within ceded territories.

In this section, we will consider data used to manage sockeye salmon (*Oncorhynchus nerka*) that spawn in the Fraser River system of Canada; the fishery is also important to the tribes in Washington State and US commercial fisherman. A variety of data are collected and used to assess the strength of salmon stocks, including "test fisheries" in the ocean that measure catch per unit effort, in-river counts of fish, and counts of egg masses ("reds") from fish that spawn. We will consider data collected in-river from a hydroacoustic counting station operated by the Pacific Salmon Commission near the town of Mission (Figure 5.1), as well as estimates of the number of fish harvested up-stream or that eventually spawn (referred to as the *spawning escapement*). Systematic differences between estimates of fish passing by Mission (X) and estimates of future harvest plus spawning escapement (Y) are attributed to in-river mortalities, which may correlate with environmental conditions (e.g., stream temperature and flow). Fishery managers rely on models that forecast future harvest and spawning escapement from in-river measurements of temperature/discharge and estimates of fish passing the hydroacoustic station at Mission (Cummings et al. 2011). If in-river mortalities are projected to be substantial, managers may reduce the allowable catch to increase the chances of meeting spawning escapement goals.

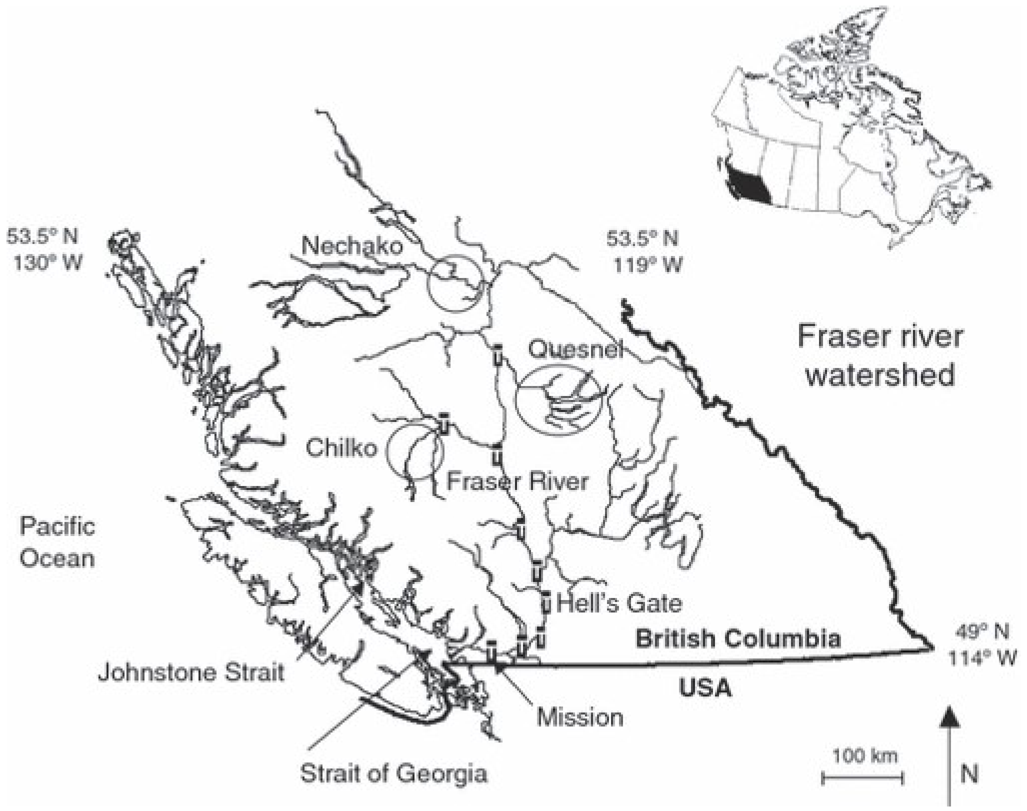


FIGURE 5.1 Map of Canada with an inset of the Fraser River Watershed of British Columbia from Cooke et al. (2009), published under a CC By 4.0 license. The number of salmon passing Mission is estimated using hydroacoustic counts.

Salmon in the Fraser river system spawn in several different streams and are typically managed based on stock aggregates formed by grouping stocks that travel through the river at similar "run" times. Typically, models would be constructed separately for each stock aggregate. For illustrative purposes, however, we will consider models fit to data from all stocks simultaneously.

```r
library(Data4Ecologists)
data(sockeye)
```

We will begin by considering a model without any in-river predictors and just focus on the relationship between the estimated number of salmon passing Mission (`MisEsc`) and the estimated number of fish spawning at the upper reaches of the river plus in-river catch above Mission (`SpnEsc`) (Figure 5.2). We see that: a) the two counts are highly correlated, b) there appear to be more observations below the 1-1 line than above it, as we might expect if there was significant in-river mortality, and c) the variability about the 1-1 line increases with the size of the counts. If we fit a least-squares regression line to the data and plot the residuals, we will see a "fan shape" indicating that the variance increases with the number of fish passing Mission (Figure 5.3). The Q-Q plot and histogram of residuals also suggests that there are several large residuals, more than you would expect if the residuals came from a Normal distribution with constant variance.

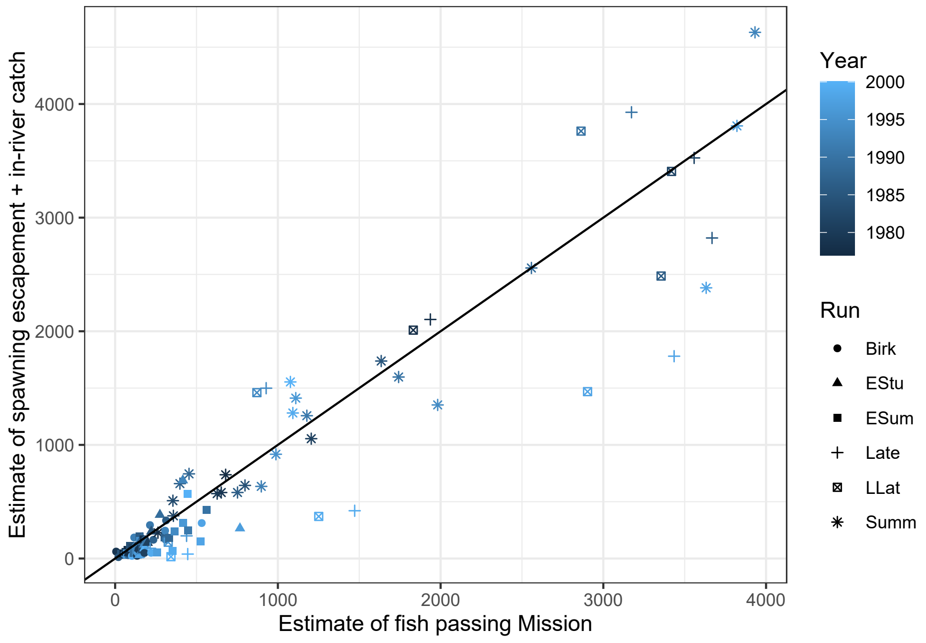


FIGURE 5.2 Summed catch and spawning escapement of Fraser Rivers sockeye salmon plotted against an estimate of the number of fish passing Mission. The black line is a 1-1 line which might be expected if there were no in-river mortalities above Mission.

```r
lmsockeye <- lm(SpnEsc ~ MisEsc, data = sockeye)
ggResidpanel::resid_panel(lmsockeye, plots = c("resid", "qq", "hist"), nrow = 1)
```

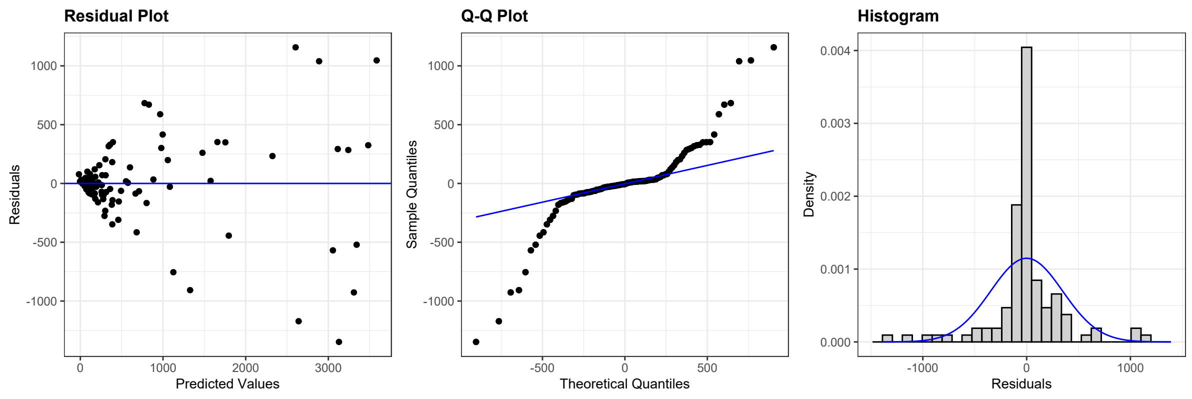


FIGURE 5.3 Residual plots for a linear model relating the estimated number of sockeye salmon passing Mission (‘MisEsc‘) to the estimated number of fish spawning at the upper reaches of the river + in-river catch above Mission (‘SpnEsc‘).

5.5.1 Variance increasing with X_i

One option for modeling data like these where the variance increases with the mean is to log both the response and predictor variable. Alternatively, we could attempt to model the variance as a function of the count at Mission. Some options include:

1. Fixed variance model: $\sigma_i^2 = \sigma^2 \text{MisEsc}_i$
2. Power variance model: $\sigma_i^2 = \sigma^2 |\text{MisEsc}_i|^{2\delta}$
3. Exponential variance model: $\sigma_i^2 = \sigma^2 e^{2\delta \text{MisEsc}_i}$
4. Constant + power variance model: $\sigma_i^2 = \sigma^2(\delta_1 + |\text{MisEsc}_i|^{\delta_2})^2$

Before exploring these different options, it is important to note a few of their characteristics. The *fixed variance model* does not require any additional parameters but will only work if the covariate takes on only positive values since the σ_i^2 must be positive. The other options do not have this limitation, but note that the *power variance model* can only be used if the predictor does not take on the value of 0 (as this would imply $\sigma_i^2 = 0$). José Pinheiro and Bates (2006) also suggest that the constant + power model tends to do better than the power model when the variance covariate takes on values close to 0. Lastly, note that several of the variance models are "nested". Setting $\delta = 0$ in the power variance model or the exponential variance model gets us back to the standard linear regression setting; setting $\delta = 0.5$ in the power variance model gets us back to the fixed variance model; setting $\delta_1 = 1$ and $\delta_2 = 0$ in the constant + variance model gets us back to a linear regression model.

Below, we demonstrate how to fit each of these models to the salmon data:

```
fixedvar <- gls(SpnEsc ~ MisEsc, weights = varFixed(~ MisEsc), data = sockeye)
varpow <- gls(SpnEsc ~ MisEsc, weights = varPower(form = ~ MisEsc), data = sockeye)
varexp <- gls(SpnEsc ~ MisEsc, weights = varExp(form = ~ MisEsc), data = sockeye)
varconstp <- gls(SpnEsc ~ MisEsc, weights = varConstPower(form = ~ MisEsc), data = sockeye)
```

We also refit the linear model using the `gls` function. We then compare the models using AIC:

```
lmsockeye <- gls(SpnEsc ~ MisEsc, data = sockeye)
AIC(lmsockeye, fixedvar, varpow, varexp, varconstp)
```

```
##           df      AIC
## lmsockeye  3 1614.925
## fixedvar   3 1459.525
## varpow     4 1446.861
## varexp     4 1482.668
## varconstp  5 1421.523
```

We find the constant + power variance function has the lowest AIC. Let's look at the summary of this model:

```
summary(varconstp)
```

```
## Generalized least squares fit by REML
##   Model: SpnEsc ~ MisEsc
##   Data: sockeye
##        AIC     BIC    logLik
##   1421.523 1434.98 -705.7614
```

```
##
## Variance function:
##  Structure: Constant plus power of variance covariate
##  Formula: ~MisEsc
##  Parameter estimates:
##     const     power
## 119.49037   1.10369
##
## Coefficients:
##                 Value Std.Error   t-value p-value
## (Intercept) -6.122849  8.644421 -0.708301  0.4803
## MisEsc       0.828173  0.052627 15.736618  0.0000
##
##  Correlation:
##        (Intr)
## MisEsc -0.666
##
## Standardized residuals:
##         Min          Q1         Med          Q3         Max
## -2.03554341 -0.68214833 -0.03890184  0.61977181  2.85039770
##
## Residual standard error: 0.172081
## Degrees of freedom: 111 total; 109 residual
```

We see that $\hat{\sigma} = 0.17$, $\hat{\delta}_1 = 119.49$ and $\hat{\delta}_2 = 1.10$. Thus, our model for the variance is given by:

$$\hat{\sigma}_i^2 = 0.17^2(119.49 + |\text{MisEsc}_i|^{1.10})^2$$

So, AIC suggests this is our best model, but how do we know that it is a reasonable model and not just the best of a crappy set of models? One thing we can do is plot *standardized* residuals versus fitted values. Specifically, if we divide each residual by $\hat{\sigma}_i$, giving $(Y_i - \hat{Y}_i)/\hat{\sigma}_i$, we should end up with standardized residuals that have approximately constant variance:

```
sig2i <- 0.172081*(119.49037 + abs(sockeye$MisEsc)^1.10369)
sockeye <- sockeye %>%
  mutate(fitted = predict(varconstp),
         stdresids  = (SpnEsc - predict(varconstp))/sig2i)
```

We can then plot standardized residuals versus fitted values to determine if we have a reasonable model for the variance (Figure 5.4). Although there appears to be more scatter in the left size of the plot, there are also more observations in this region.

```
ggplot(sockeye, aes(fitted, stdresids)) + geom_point() +
  geom_hline(yintercept = 0)
```

We could also look at a Q-Q plot of the standardized residuals (Figure 5.5), which doesn't look too bad as most observations fall on the qq-line.

```
ggplot(sockeye, aes(sample = stdresids)) +
  stat_qq() +
  stat_qq_line() +
  theme_bw()
```

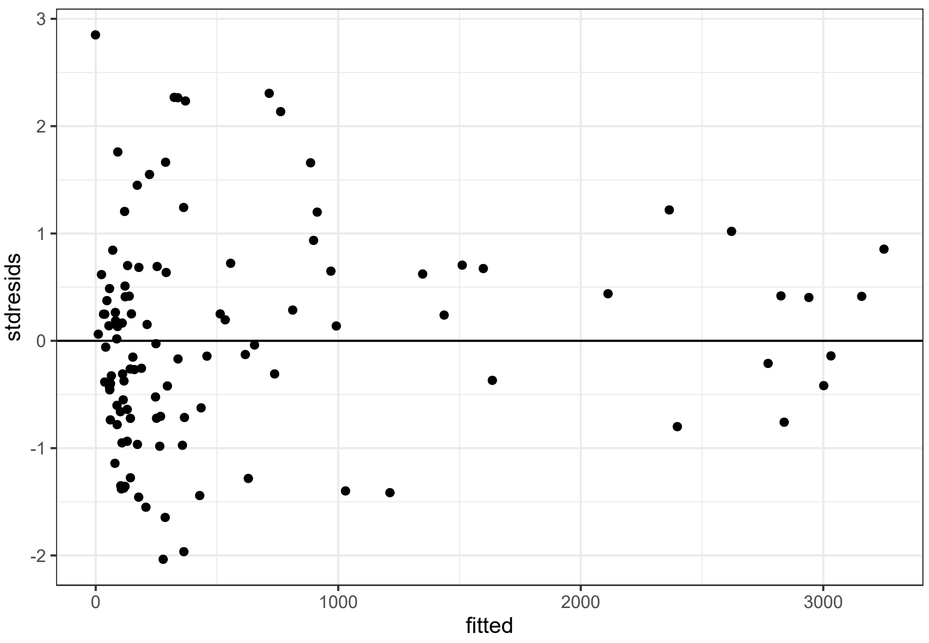


FIGURE 5.4 Plot of standardized residuals versus fitted values for the constant + power variance model relating the estimated number of sockeye salmon passing Mission (`MisEsc`) to the estimated number of fish spawning at the upper reaches of the river + in-river catch above Mission (`SpnEsc`).

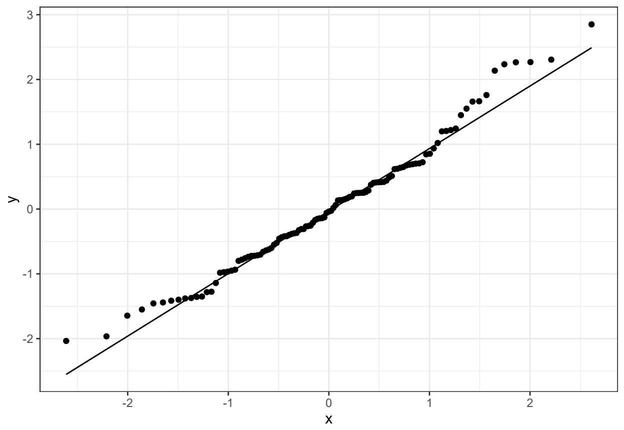


FIGURE 5.5 Q-Q plot of standardized residuals for the constant + power variance model relating the estimated number of sockeye salmon passing Mission (`MisEsc`) to the estimated number of fish spawning at the upper reaches of the river + in-river catch above Mission (`SpnEsc`).

Although it is instructive to calculate the standardized residuals "by hand", they can also be accessed via `resid(varconstp, type = "normlized")`, and a plot of standardized residuals versus fitted values can be created using `plot(modelobject)` (Figure 5.6)

```
cbind(resid(varconstp, type = "normalized"), sockeye$stdresids)[1:3,]
```

```
##          [,1]       [,2]
## 1 2.85039770 2.85039838
## 2 0.06174022 0.06174023
## 3 0.61705145 0.61705141
```

```
plot(varconstp)
```

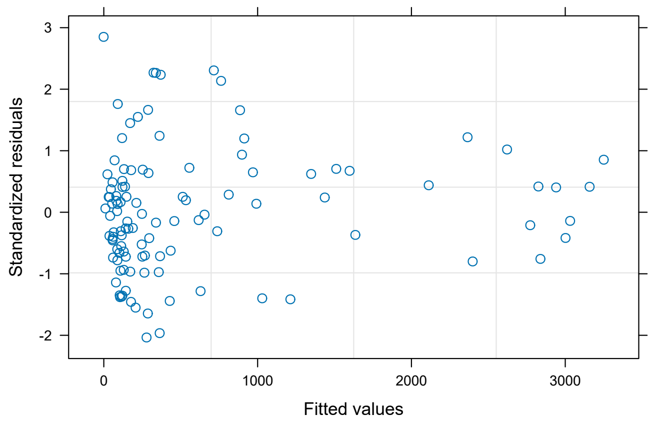


FIGURE 5.6 Standardized residuals versus fitted value plot created using the `plot` function.

5.5.2 Extension to multiple groups

The variance functions from the last section can also be extended to allow for separate variance parameters associated with each level of a categorical predictor. For example, we could specify a power variance model that is unique to each aggregated spawning run (`Run`) using `varPower(form = ~ MiscEsc | Run)`:

```
varconstp2 <- gls(SpnEsc ~ MisEsc, weights = varConstPower(form = ~ MisEsc | Run),
                  data = sockeye)
summary(varconstp2)
```

```
## Generalized least squares fit by REML
##   Model: SpnEsc ~ MisEsc
##   Data: sockeye
##        AIC      BIC   logLik
##   1422.534 1462.904 -696.267
##
## Variance function:
```

```
##  Structure: Constant plus power of variance covariate, different strata
##  Formula: ~MisEsc | Run
##  Parameter estimates:
##              Birk         ESum         EStu       LLat       Late        Summ
## const 145.5257480 2.255901e-05 5.574150e-07 0.01272132 831.497995 552.0327859
## power   0.9884558 1.090556e+00 1.121711e+00 1.11926641   1.031332   0.9761909
##
## Coefficients:
##                 Value Std.Error   t-value p-value
## (Intercept) -5.338102  6.264161 -0.852165   0.396
## MisEsc       0.846292  0.044241 19.128999   0.000
##
##  Correlation:
##        (Intr)
## MisEsc -0.646
##
## Standardized residuals:
##        Min         Q1        Med         Q3        Max
## -1.8507999 -0.7866592 -0.1026727  0.6805122  2.0956069
##
## Residual standard error: 0.2111633
## Degrees of freedom: 111 total; 109 residual
```

In this case, we see that we get separate variance parameters for each aggregated stock, and the parameters appear to differ quite a bit depending on the aggregation. Yet, AIC appears to slightly favor the simpler model:

```
AIC(varconstp, varconstp2)
```

```
##            df      AIC
## varconstp   5 1421.523
## varconstp2 15 1422.534
```

5.5.3 Variance depending on μ_i

With more complicated models that include multiple predictors that influence the mean, it is sometimes useful to model the variance not as a function of one or more predictor variables, but rather as a function of the mean itself (José Pinheiro and Bates 2006; Mech, Fieberg, and Barber-Meyer 2018). Specifically:

$$
\begin{gathered}
Y_i \sim N(\mu_i, \sigma_i^2) \\
\mu_i = \beta_0 + \beta_1 X_i \\
\sigma_i^2 = \mu_i^{2\delta}
\end{gathered}
$$

This model can be fit using `varPower(form = ~ fitted(.))`. I was unable to get this model to converge[1] when using the full data set, but demonstrate its use for modeling just the early summer aggregation:

[1]Parameters are estimated using an iterative numerical optimization method, which can sometimes fail. We will learn more about these methods when we get to Section 10 on Maximum Likelihood estimation.

```
varmean <- gls(SpnEsc ~ MisEsc, weights = varPower(form = ~ fitted(.)),
                  data = subset(sockeye, Run=="ESum"))
summary(varmean)
```

```
## Generalized least squares fit by REML
##   Model: SpnEsc ~ MisEsc
##   Data: subset(sockeye, Run == "ESum")
##        AIC      BIC    logLik
##   260.2083 264.5725 -126.1042
##
## Variance function:
##  Structure: Power of variance covariate
##  Formula: ~fitted(.)
##  Parameter estimates:
##    power
## 1.540548
##
## Coefficients:
##                 Value Std.Error  t-value p-value
## (Intercept) 21.632043  8.875419 2.437298  0.0233
## MisEsc       0.591687  0.100345 5.896532  0.0000
##
##  Correlation:
##        (Intr)
## MisEsc -0.823
##
## Standardized residuals:
##        Min         Q1        Med         Q3        Max
## -1.5456716 -0.6159292 -0.1371156  0.3850583  2.1805075
##
## Residual standard error: 0.02768571
## Degrees of freedom: 24 total; 22 residual
```

We see that our estimate of δ is equal 1.54058.

5.6 Approximate confidence and prediction intervals

The generic `predict` function works with models fit using `gls`, but it will not return standard errors as it does for models fit using `lm` when supplying the additional argument, `se.fit = TRUE`. In addition, for many applications, we may be interested in obtaining prediction intervals. This section will illustrate how we can use matrix multiplication to calculate confidence and prediction intervals for our chosen Sockeye salmon model.

Think-pair-share: Consider the role these models might play in management. Do you think managers of the sockeye salmon fishery will be more interested in confidence intervals or prediction intervals?

Recall the difference between confidence intervals and prediction intervals from Chapter 1.9. Confidence intervals capture uncertainty surrounding the average value of Y at any given set of predictors, whereas a prediction interval captures uncertainty regarding a *particular* value of Y at a given value of X. In the Sockeye Salmon example, above, managers are most likely interested in forming a prediction interval that is likely to capture the size of the salmon run in a *particular* year. Thus, we need to be able to calculate a prediction interval for $Y|X$. Managers may also be interested in describing average trends across years using confidence intervals for the mean.

For a confidence interval, we need to estimate:

$$Var(\hat{E}[Y_i|X_i]) = Var(\hat{\beta}_0 + \hat{\beta}_1 X_i)$$

i.e., we need to quantify our uncertainty regarding the regression line.

For a prediction interval, we need to estimate:

$$Var(\hat{Y}_i|X_i) = Var(\hat{\beta}_0 + \hat{\beta}_1 X_i + \epsilon_i)$$

i.e., we need to quantify both the uncertainty associated with the regression line and variability about the line.

Aside: to derive appropriate confidence and prediction intervals, we will need to leverage a bit of statistical theory, some of which was covered in Section 3.12. In particular, it helps to know that statistics such as means, sums, regression coefficients will be Normally distributed if our sample size, n, is "large" (thanks to the *Central Limit Theorem*!) Further, if the vector $(X, Y) \sim N(\mu, \Sigma)$, then for constants a and b, $aX + bY$ will be $N(\widetilde{\mu}, \widetilde{\Sigma})$ with:

- $\widetilde{\mu} = E(aX + bY) = aE(X) + bE(Y)$
- $\widetilde{\Sigma} = Var(aX + bY) = a^2Var(X) + b^2Var(Y) + 2abCov(X, Y)$, where $Cov(X, Y)$ describes how X and Y covary. If $Cov(X, Y)$ is positive, then we expect high values of X to be associated with high values of Y.

As we saw in Section 3.12, and we will further demonstrate below, matrix multiplication offers a convenient way to calculate means and variances of linear combinations of random variables (e.g., $aX + bY$).

We can begin by calculating predicted values for each observation in our data set by multiplying our design matrix by the estimated regression coefficients:

$$\hat{E}[Y|X] = \begin{bmatrix} 1 & \text{MscEsc}_1 \\ 1 & \text{MscEsc}_2 \\ \vdots & \vdots \\ 1 & \text{MscEsc}_n \end{bmatrix} \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{bmatrix} = \begin{bmatrix} \hat{\beta}_0 + \text{MscEsc}_1\hat{\beta}_1 \\ \hat{\beta}_0 + \text{MscEsc}_2\hat{\beta}_1 \\ \vdots \\ \hat{\beta}_0 + \text{MscEsc}_n\hat{\beta}_1 \end{bmatrix},$$

```r
# Design matrix for our observations
xmat <- model.matrix(~ MisEsc, data=sockeye)

# Regression coefficients
betahat<-coef(varconstp)

# Predictions
sockeye$SpnEsc.hat <- predict(varconstp)
cbind(head(xmat%*%betahat), head(sockeye$SpnEsc.hat))
```

```
##        [,1]      [,2]
## 1 -1.153813 -1.153813
## 2 10.440605 10.440605
## 3 23.691368 23.691368
## 4 31.973095 31.973095
## 5 36.942131 36.942131
## 6 36.942131 36.942131
```

The variance for the i^{th} prediction is given by:

$$Var(\hat{E}[Y_i|\text{MscEsc}_i]) = Var(\hat{\beta}_0 + \hat{\beta}_1\text{MscEsc}_i) = Var(\hat{\beta}_0) + \text{MscEsc}_i^2 Var(\hat{\beta}_1) + 2\text{MscEsc}_i Cov(\hat{\beta}_0, \hat{\beta}_0)$$

We can calculate this variance using matrix multiplication. Let:

$$\hat{\Sigma} = \begin{bmatrix} Var(\hat{\beta}_0) & Cov(\hat{\beta}_0, \hat{\beta}_1) \\ Cov(\hat{\beta}_0, \hat{\beta}_1) & Var(\hat{\beta}_1) \end{bmatrix}$$

Then, we have:

$$\begin{bmatrix} 1 & \text{MscEsc}_i \end{bmatrix} \hat{\Sigma} \begin{bmatrix} 1 \\ \text{MscEsc}_i \end{bmatrix} = Var(\hat{\beta}_0) + \text{MscEsc}_i^2 Var(\hat{\beta}_1) + 2\text{MscEsc}_i Cov(\hat{\beta}_0, \hat{\beta}_0)$$

Furthermore, if we calculate $X\hat{\Sigma}X'$, where X is our design matrix, we will end up with a matrix that contains all of the variances along the diagonal and covariances between the predicted values on the off-diagonal. We can then pull off the diagonal elements (the variances), take their square root to get standard errors, and then take $\pm 1.645SE$ to form point-wise 90% confidence intervals for the line at each value of X.

We can access $\hat{\Sigma}$ using the `vcov` function. Also, note that we can transpose a matrix in R using `t(matrix)`:

```
# Sigma^
Sigmahat <- vcov(varconstp)

# var/cov(beta0 + beta1*X)
varcovEYhat<-xmat%*%Sigmahat%*%t(xmat)

# Pull off the diagonal elements and take their sqrt to
# get SEs that quantify uncertainty associated with the line
SEline <- sqrt(diag(varcovEYhat))
```

We then plot the points, fitted line, and a confidence interval for the line (Figure 5.7)

```
# Confidence interval for the mean
sockeye$upconf <- sockeye$SpnEsc.hat + 1.645 * SEline
sockeye$lowconf <- sockeye$SpnEsc.hat - 1.645 * SEline
p1 <- ggplot(sockeye, aes(MisEsc, SpnEsc)) + geom_point() +
  geom_line(aes(MisEsc, SpnEsc.hat)) +
  geom_line(aes(MisEsc, lowconf), col = "red", lty = 2) +
  geom_line(aes(MisEsc, upconf), col = "red", lty = 2) +
  xlab("Estimate at Mission") + ylab("Estimate of C+Spawners") +
  theme_bw()
p1
```

For prediction intervals, we need to calculate:

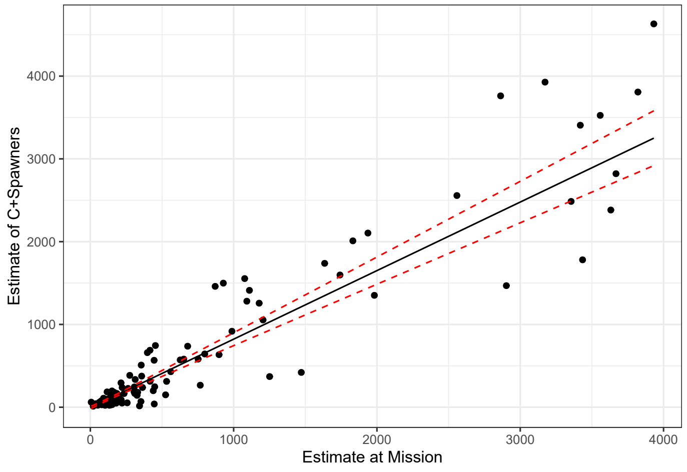


FIGURE 5.7 Predictions for the catch and spawning escapement based on the count of sockeye salmon at Mission. A 95% confidence interval for the mean is given by the red dotted lines.

$$Var(\hat{Y}_i|X_i) = Var(\hat{\beta}_0 + \hat{\beta}_1\text{MisEsc}_i + \epsilon_i)$$

We will approximate this expression with:

$$Var(\hat{Y}_i|X_i) \approx \widehat{var}(\hat{\beta}_0 + \hat{\beta}_1\text{MisEsc}_i) + \hat{\sigma}_i^2 = X\hat{\Sigma}X' + \hat{\sigma}_i^2,$$

where $\hat{\sigma}_i$ is determined by our variance function and is equal to $\hat{\sigma}_i^2 = 0.17^2(119.49 + |\text{MisEsc}_i|^{1.10})^2$ for the constant + power variance model. We have already calculated $\hat{\sigma}_i$ for all of the observations in our data set and stored it the variable `sig2i`. We put the different pieces together and plot the prediction intervals (Figure 5.8).

```
# Approx. prediction interval, ignoring the uncertainty in theta
varpredict <- SEline ^ 2 + sig2i ^ 2
sockeye$pupconf <- sockeye$SpnEsc.hat + 1.645 * sqrt(varpredict)
sockeye$plowconf <- sockeye$SpnEsc.hat - 1.645 * sqrt(varpredict)
ggplot(sockeye, aes(MisEsc, SpnEsc)) + geom_point() +
  geom_line(aes(MisEsc, SpnEsc.hat)) +
  geom_line(aes(MisEsc, lowconf), col = "red", lty = 2) +
  geom_line(aes(MisEsc, upconf), col = "red", lty = 2) +
  geom_line(aes(MisEsc, plowconf), col = "blue", lty = 2) +
  geom_line(aes(MisEsc, pupconf), col = "blue", lty = 2) +
  xlab("Estimate at Mission") + ylab("Estimate of C+Spawners")+
  theme_bw()
```

Before concluding, we note that these confidence and prediction intervals are approximate in that they rely on asymptotic Normality (due to the Central Limit Theorem), and they ignore uncertainty in the variance parameters.

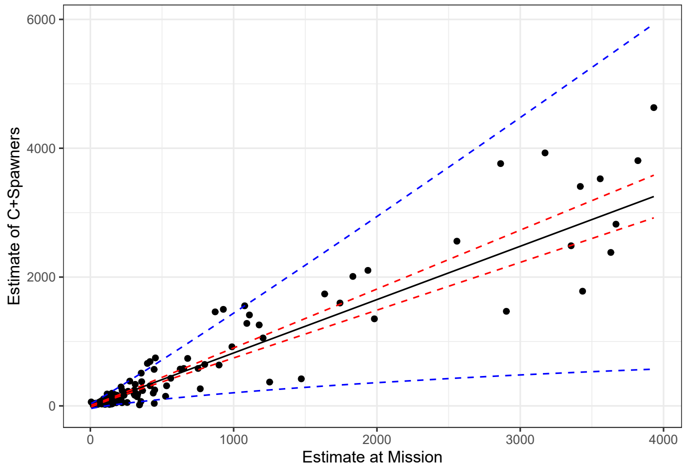


FIGURE 5.8 Predictions for the catch and spawning escapement based on the count of sockeye salmon at Mission. Blue and red lines depict 95% prediction and confidence intervals, respectively.

5.7 Modeling strategy

By considering models for both the mean and variance, we have just increased the flexibility (and potential complexity) of our modeling approach. I.e., we can fit models that relax the constant variance assumption of linear models, but doing so will require the estimation of additional parameters. Our simple examples involved very few predictors, but what if we are faced with a more difficult challenge of choosing among many potential predictors for the mean and also multiple potential models for the variance? It is easy to fall into a trap of more and more complicated data-driven model selection, the consequences of which we will discuss in Chapter 8. My general recommendation would be to include any predictor variables in the model for the mean that you feel are important for addressing your research objectives (see Chapters 6, 7, and 8 for important considerations). You might start by fitting a linear model and inspect the residuals to see if the assumptions of linear regression hold. Importantly, you could do this without even looking at the estimated coefficients and their statistical significance. Plotting the residuals versus fitted values and versus different predictors may lead you to consider a different variance function. You might then fit one or more variance models and inspect plots of standardized residuals versus fitted values to see if model assumptions are better met. You might also consider comparing models using AIC. Again, this could all be done without looking at whether the regression coefficients are statistically significant to avoid biasing your conclusions in favor of finding significant results.

If you find yourself gravitating towards more and more complex models and the assumptions still look problematic, I suggest going for a long walk. Then come back and consider other options. You might consider using bootstrapping (Chapter 2) or reverting to a simpler analysis but using more caution when interpreting results. You might also consider using simulations to evaluate the impact that assumption violoations may have on your inference.

Part II

What Variables to Include in a Model?

6

Multicollinearity

Learning objectives

1. Be able to describe and identify different forms of multicollinearity.
2. Understand the effects of multicollinearity on parameter estimates and their standard errors.
3. Be able to evaluate predictors for multicollinearity using variance inflation factors.
4. Understand some strategies for dealing with multicollinearity.

6.1 R Packages

We begin by loading the `dplyr` package:

```
library(dplyr) # for data wrangling
```

In addition, we will use data and functions from the following packages:

- `openintro` for the `mammals` data set
- `car` for the `vif` function used to calculate variance inflation factors
- `GGally` for creating a scatterplot matrix

6.2 Multicollinearity

As we noted in Section 3.14.1, predictor variables will almost always be correlated in observational studies. When these correlations are strong, it can be difficult to estimate parameters in a multiple regression model, leading to coefficients with large standard errors. This should, perhaps, not be too surprising given that we have emphasized the interpretation of regression coefficients in a multiple regression model as quantifying the expected change in Y as we change a focal predictor by 1 unit *while holding all other predictors constant*. When two variables are highly correlated, changes in one variable tend to be associated with changes in the other. Therefore, our data do not provide much information about how Y changes in response to changes in only one of the predictors, and the inflated standard errors associated with regression coefficients for collinear variables accurately reflect these uncertainties (Morrissey and Ruxton 2018).

If we are primarily interested in prediction, collinearity may be of little importance. We can use a model with collinear variables to obtain predictions, and these predictions can be unbiased and precise as long as our data are representative of the population for which we aim to obtain predictions (Michael H. Kutner et al. 2005, 5:283; Dormann et al. 2013). On the other hand, collinearity will make inference challenging since it will be difficult to distinguish the unique effects of collinear variables. Thus, we will need tools to identify collinear variables and to make inferences regarding their effects.

Ecologists commonly compute pairwise correlations between all predictor variables and drop one member of any pair when the absolute value of the associated correlation coefficient, $|r|$ is greater than 0.7 (Dormann et al. 2013). Dropping a predictor from a highly correlated pair will often result in more precise estimates but necessarily leads to some loss of information. Further, although pairwise correlations may identify pairs of variables that are collinear, ecologists also need to be able to diagnose situations in which the information in a predictor variable is largely redundant with the information contained in a suite of predictors (i.e., multicollinearity). Multicollinearity implies that one of the explanatory variables can be predicted by the others.

There are several types of collinearity/multicollinearity that can arise when analyzing data. In some cases, we may have multiple measurements that reflect some latent (i.e. unobserved) quantity; Dormann et al. (2013) refers to this as *intrinsic collinearity.* For example, biologists may quantify the size of their study animals using multiple measurements (e.g., length from head to tail, neck circumference, etc). It would not be surprising to find that pairwise correlations between these different body measurements are large because they all relate to one latent quantity, which is overall body size. In other cases, quantitative variables may be constructed by aggregating categorical variables (e.g., percent cover associated with different land-use categories in different pixels within a Geographical Information System [GIS] layer). These compositional variables must sum to 1, and thus, the last category is completely determined by the others. This is an extreme example of multicollinearity; statistical software will be unable to estimate separate coefficients for each compositional variable along with an overall intercept. Collinearity can also occur *structurally,* e.g., when fitting models with polynomial terms or with predictors that have very different scales. For example, the correlation between x and x^2 will often be large (see below).

```
x <- seq(0:100); x2 = x*x
cor(x,x^2)
```

```
## [1] 0.9688484
```

This type of collinearity can often be fixed by using various transformations of the explanatory variables (e.g., using orthogonal polynomials as discussed in Section 4.11 or by centering and scaling variables; Schielzeth 2010). Lastly, variables may be correlated due to other reasons that are harder to identify on their own - e.g., perhaps several predictors covary spatially leading to mild to severe multicollinearity. As an example, temperature, precipitation, and elevation may all covary along an elevation gradient. Lastly, collinearity can also happen by chance in small data sets; Dormann et al. (2013) calls this *incidental collinearity* and suggests it is best addressed through proper sampling or experimental design (e.g., to ensure that predictor variables do not substantially covary).

Some of the symptoms of multicollinearity include:

- regression coefficients may be statistically significant in simple linear regression models but not in regression models with multiple predictors as they compete to explain the same variability in Y.
- parameter estimates may have large standard errors, representing large uncertainty, despite large sample sizes.
- the p-values associated with t-tests for the individual coefficients in the regression model may be large (> 0.05), but a multiple degree-of-freedom F-test of the null hypothesis that all coefficients are 0 may have a p-value < 0.05. In essence, we can conclude that the suite of predictor variables explains significant variation in Y, but we are unable to identify the independent contribution of each of the predictors.
- lack of convergence when using numerical optimization routines to estimate parameters in complex statistical models.

- parameter estimates may be unstable; small changes in the data or model structure can change parameter estimates considerably (sometimes referred to as the "bouncing beta problem").
- parameter estimates may change in magnitude (and even sign) depending on what over variables are included in the model (e.g., Table 6.1). As we will see in Chapter 7, we can use causal diagrams to try to understand and leverage these types of behaviors when estimating direct and indirect effects of our explanatory variables.

6.3 Motivating example: what factors predict how long mammals sleep?

To demonstrate some of these issues, let's briefly explore a popular data set from the `openintro` package (Çetinkaya-Rundel et al. 2021) containing measurements of the average amount of time different mammals sleep per 24 hour period (Allison and Cicchetti 1976; Savage and West 2007).

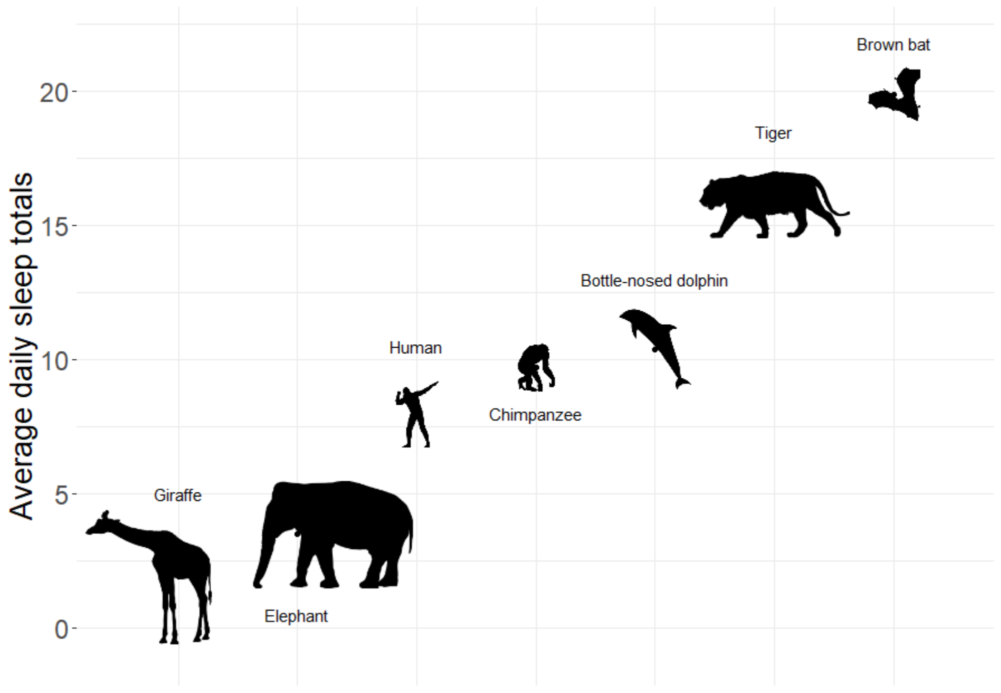


FIGURE 6.1 Average daily sleep totals for several different mammal species. Figure created using images available on PyloPic. Elephant by T. Michael Keesey - CC By 3.0.

Sleep is characterized by either slow wave (non-dreaming) or rapid eye movement (dreaming), with wide variability in the amount of both types of sleep among mammals. In particular, Roe Deer (*Capreolus capreolus*) sleep $<$ 3 hours/day whereas the Little Brown Bat (*Myotis lucifugus*) sleeps close to 20 hours per day.

The data set has many possible variables we could try to use to predict the amount of time different mammals sleep, including:

- Lifespan (years)
- Gestation (days)

TABLE 6.1 Estimates (p-values) for lifespan in a model with and without other explanatory variables.

	Model 1	Model 2
(Intercept)	12.312 (<0.001)	17.837 (<0.001)
life_span	−0.097 (0.004)	−0.008 (0.811)
danger		−1.728 (<0.001)
log(brain_wt)		−0.879 (<0.001)

- Brain weight (g)
- Body weight (kg)
- Predation Index (1-5; 1 = least likely to be preyed upon)
- Exposure Index [1-5: 1 = least exposed (e.g., animal sleeps in a den)]
- Danger Index (1:5, combines exposure and predation; 1= least danger from other animals)

It turns out that all of these variables are negatively correlated with sleep (see the correlations in the first row of Figure 6.2) - no wonder it is so difficult to get a good night's rest!

```
library(openintro)
data(mammals, package="openintro")
mammals <- mammals %>% dplyr::select(total_sleep, life_span, gestation,
                          brain_wt, body_wt, predation,
                          exposure, danger) %>%
  filter(complete.cases(.))
GGally::ggpairs(mammals)
```

We also see that our predictor variables are highly correlated with each other (Figure 6.2). Lastly, we see that the distribution of `brain_wt` and `body_wt` are severely right skewed with a few large outliers. To reduce the impact of these outliers, we will consider log-transformed predictors, `log(brain_wt)` and `log(body_wt)` when including them in regression models[1].

To see how collinearity can impact estimated regression coefficients and statistical hypothesis tests involving these coefficients, let's consider the following two models:

Model 1: $\texttt{Sleep} = \beta_0 + \beta_1\texttt{life\_span}$

Model 2: $\texttt{Sleep} = \beta_0 + \beta_1\texttt{life\_span} + \beta_2\texttt{danger} + \beta_3\texttt{log(brain\_wt)}$

```
model1<-lm(total_sleep ~ life_span, data=mammals)
model2<-lm(total_sleep ~ life_span + danger + log(brain_wt), data=mammals)
modelsummary(list("Model 1" = model1, "Model 2" = model2), gof_omit = ".*",
             estimate = "{estimate} ({p.value})",
             statistic = NULL,
             title="Estimates (p-values) for lifespan in a model with and
             without other explanatory variables.")
```

In the first model, `life_span` is statistically significant and has a coefficient of -0.097 (Table 6.1). However,

[1]Sometimes researchers will state that predictors were log-transformed to make them Normally distributed. Linear regression models do NOT assume that *predictors* are Normally distributed, so this reasoning is faulty. However, it is sometimes beneficial to consider log-transformations for predictor variables that have skewed distributions as a way to reduce the influence of individual outlying observations.

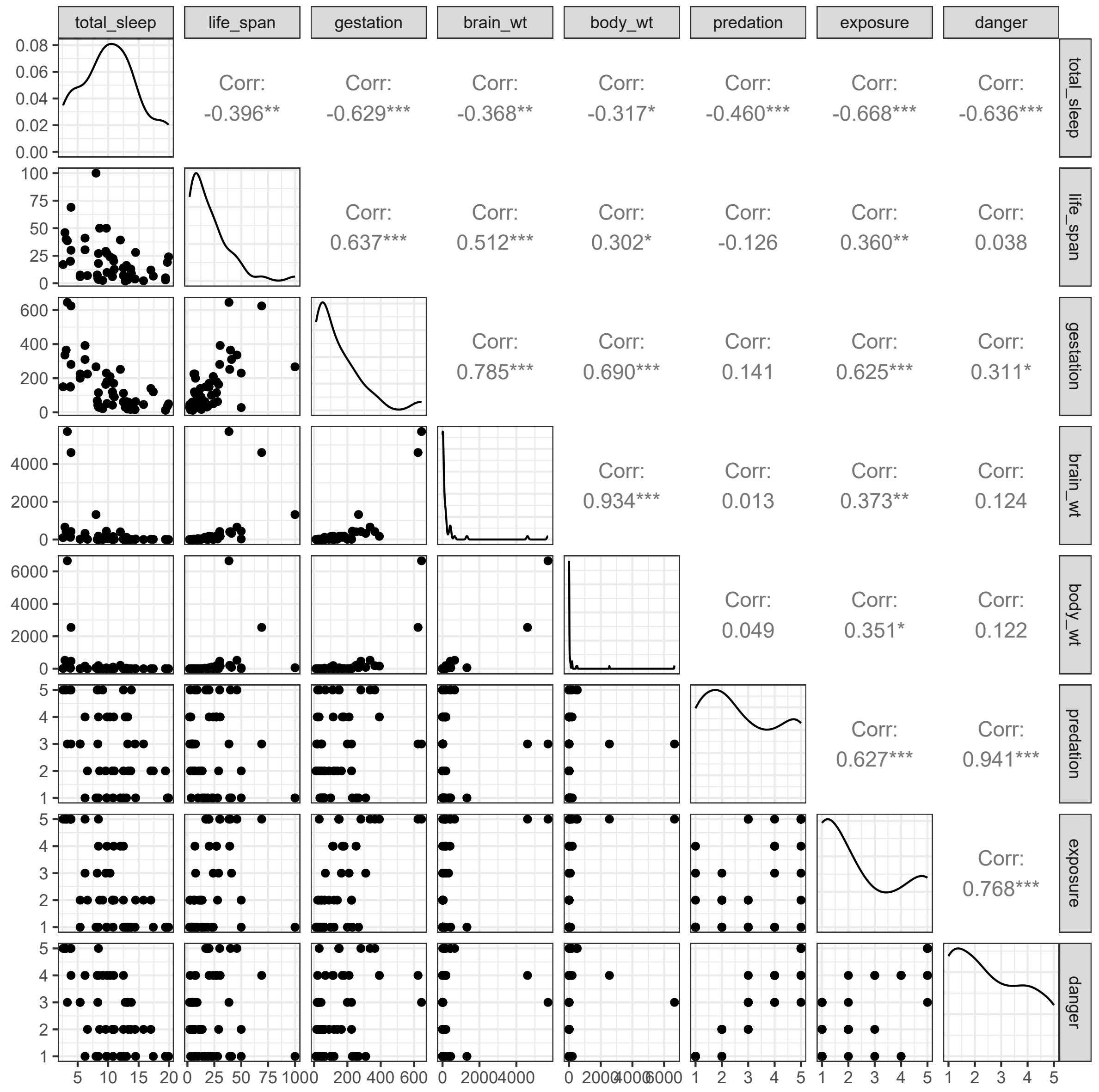


FIGURE 6.2 Scatterplot matrix of the predictors in the `mammals` data set.

TABLE 6.2 Estimates (p-values) for a model containing all predictors and one that only contains predation.

	Model 3	Model 4
(Intercept)	17.149 (<0.001)	14.465 (<0.001)
life_span	0.015 (0.647)	
gestation	−0.008 (0.094)	
log(brain_wt)	−0.841 (0.189)	
log(body_wt)	0.129 (0.781)	
predation	1.950 (0.046)	−1.448 (<0.001)
exposure	0.828 (0.147)	
danger	−4.193 (<0.001)	

`life_span` is no longer statistically significant once we include `danger` and `log(brain_wt)`, and its coefficient is also an order of magnitude smaller ($\hat{\beta}_{log.BrainWt} = -0.008$). What happened?

Because `life_span` and `log(brain_wt)` are positively correlated, they will "compete" to explain the same variability in sleep measurements.

```
mosaic::cor(life_span ~ log(brain_wt), data = mammals, use = "complete.obs")
```

```
[1] 0.7267191
```

If we try to fit a model with all of the explanatory variables, we find that only `danger` and `predation` have p-values < 0.05 (Table 6.2). Further, the coefficient for `predation` is positive despite our intuition that high predation pressure should be negatively correlated with sleep. And, indeed, the coefficient for `predation` is negative if it is the only explanatory variable in the model. In Sections 6.5 and 7, we will use causal diagrams to explore potential reasons why the magnitude and direction of a regression coefficient may change after adding or excluding a variable.

```
model3 <- lm(total_sleep ~ life_span + gestation + log(brain_wt) +
               log(body_wt) + predation + exposure + danger, data=mammals)
model4 <- lm(total_sleep ~ predation, data=mammals)
modelsummary(list("Model 3" = model3, "Model 4" = model4), gof_omit = ".*",
             estimate = "{estimate} ({p.value})",
             statistic = NULL,
             title="Estimates (p-values) for a model containing all
             predictors and one that only contains predation.")
```

6.4 Variance inflation factors (VIF)

Variance inflation factors (VIFs) were developed to quantify collinearity in ordinary least squares regression models. They measure the degree to which the variance (i.e., SE^2) of a regression parameter is inflated due to collinearity and can be calculated using:

$$VIF(\hat{\beta}_j) = \frac{1}{1 - R^2_{X_j|X_1,\ldots,X_{j-1},X_{j+1},X_p}} \tag{6.1}$$

where $R^2_{X_j|X_1,\ldots,X_{j-1},X_{j+1},X_p}$ is the multiple R^2 from a regression that predicts X_j using all other predictors in the model (i.e., $\text{lm}(X_j \sim X_1 + \ldots + X_{j-1} + X_{j+1} + X_p)$). Inspecting equation (6.1), we see that as this R^2 approaches 1, the VIF will approach ∞. The square root of the VIF can be interpreted as providing an indication of how much larger the standard error associated with X_j is compared to the case where X_j is uncorrelated with all other predictors (i.e., if $R^2_{X_j|X_1,\ldots,X_{j-1},X_{j+1},X_p} = 0$). There are several rules of thumb in the published literature that suggest VIFs ≥ 4 or 5 warrant further inspection and VIFs ≥ 10 are particularly problematic (Michael H. Kutner, Nachtsheim, and Neter 2004). In an influential paper published in Ecology, Graham (2003) highlighted that a VIF around 2 was high enough to impact the choice of predictor variables when applying common model selection algorithms (see also Zuur, Ieno, and Elphick 2010). In the sections that follow, we will briefly explore the impact of collinearity using a simulation study and then consider the data and examples from Graham (2003) .

Let's have a look at the VIFs for the sleep data if we were to include all of the predictors. We will use the `vif` function in the `car` package (Fox and Weisberg 2019) to calculate the VIFs:

```
car::vif(model3)
```

```
##     life_span     gestation log(brain_wt)  log(body_wt)     predation 
##      2.507122      2.971837     16.568907     13.903451     12.978738 
##      exposure        danger 
##      5.121031     16.732626
```

We can also use the `check_model` function in the performance package to create a nice visualization of the VIFS (Figure 6.3). We see that several of the VIFs are large and greater than 10, suggesting that several of our predictor variables are collinear. You will have a chance to further explore this data set as part of an exercise associated with this Section.

```
performance::check_model(model3, check = "vif")
```

6.5 Understanding confounding using DAGs: A simulation example

Let's consider a system of variables, represented using a directed acylical graph (DAG)[2] in which arrows represent causal effects (Figure 6.4).

The arrows between X_1 and y and X_2 and Y indicate that if we manipulate either X_1 or X_2, these changes will have a *direct* effect on Y. The link between X_1 and X_2 also highlights that when we manipulate X_1 we will also change X_2. Thus, X_1 also has an *indirect* effect on Y that is mediated by X_2 through the path $X_1 \rightarrow X_2 \rightarrow Y$.

Let's simulate data consistent with the DAG in Figure 6.4, assuming:

- $X_{1,i} \sim U(0, 10)$, where U is a uniform distribution, meaning that $X_{1,i}$ can take on any value between 0 and 10 with equal probability.
- $X_{2,i} = \tau X_{1,i} + \gamma_i$ with $\gamma_i \sim N(0, 4)$
- $Y_i = 10 + 3X_{1,i} + 3X_{2,i} + \epsilon_i$ with $\epsilon_i \sim N(0, 2)$

[2] We will talk more about DAGs in Chapter 7, but for now, note that a DAG displays causal connections among a set of variables

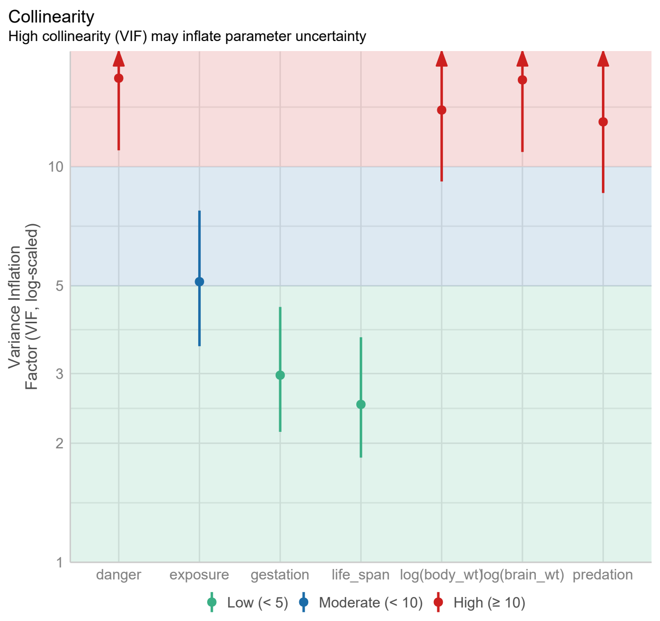


FIGURE 6.3 Variance inflation factors visualized using the `check_model` function in the `performance` package (Lüdecke et al. 2021).

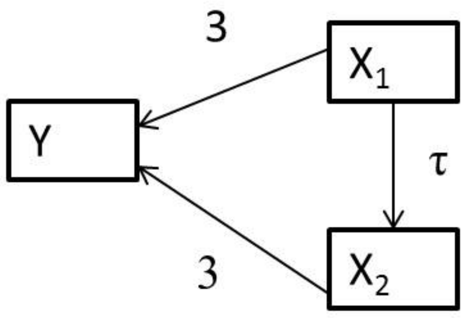


FIGURE 6.4 Directed acyclical graph (DAG) with magnitudes of coefficients depicting causal relationships between X1, X2, and Y.

Let's simulate data sets for values of τ ranging from 0 to 9 by 3, and for each data set, fit the following 2 models:

- `lm(Y ~ X1)`
- `lm(Y ~ X1 + X2)`

Looking at Figure 6.5, we see that:

- The coefficient for X_1 is biased whenever X_2 is not included in the model (unless $\tau = 0$, in which case X_1 and X_2 are independent; left panel of Figure 6.5)
- The magnitude of the bias increases with the correlation between X_1 and X_2 (i.e., with τ)

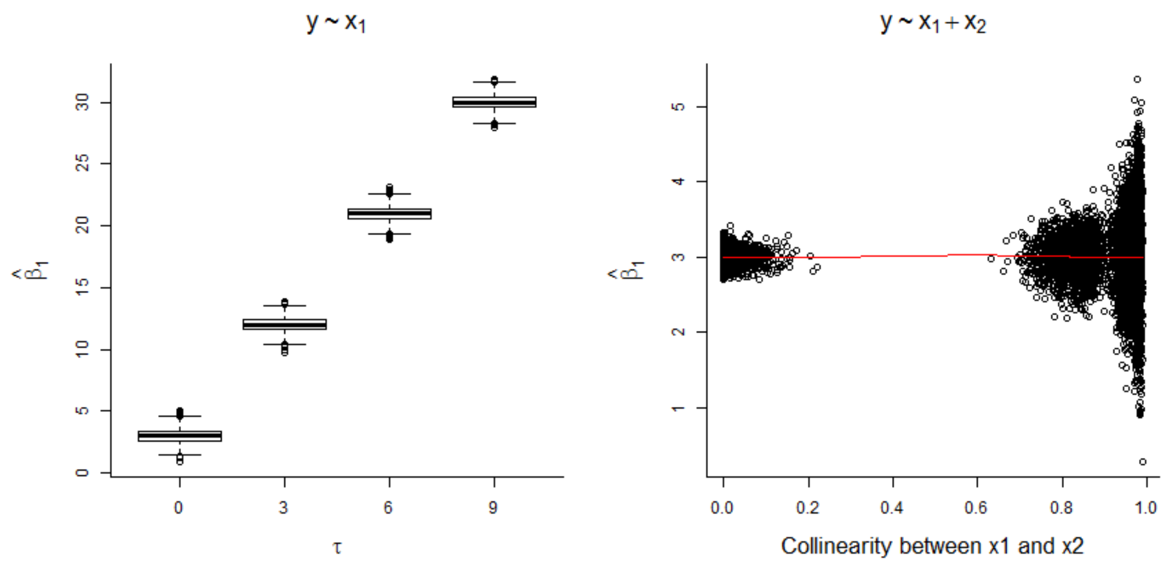


FIGURE 6.5 Results of fitting models with and without X_2 to data simulated using the DAG from Figure 6.4. The true $\beta_1 = 3$.

- The coefficient for X_1 is unbiased when X_2 is included, but estimates of $\hat{\beta}_1$ become more variable as the correlation between X_1 and X_2 increases (right panel of Figure 6.5).

To understand these results, note that:

$$\begin{aligned} Y_i = 10 + 3X_{1,i} + 3X_{2,i} + \epsilon_i \text{ and } X_{2,i} &= \tau X_{1,i} + \gamma_i \\ \Rightarrow Y_i &= 10 + 3X_{1,i} + 3(\tau X_{1,i} + \gamma_i) + \epsilon_i \\ \Rightarrow Y_i &= 10 + (3 + 3\tau)X_{1,i} + (3\gamma_i + \epsilon_i) \end{aligned}$$

Thus, when we leave X_2 out of the model, X_1 will capture both the direct effect of X_1 on Y as well as the indirect effect of X_1 on Y that occurs through the path $X_1 \rightarrow X_2 \rightarrow Y$. The relative strength of these two effects are dictated by the coefficients that capture the magnitude of the causal effects along each path. The magnitude for the path $X_1 \rightarrow Y$ is 3 since this is a direct path. For the $X_1 \rightarrow X_2 \rightarrow Y$ path, we have to multiply the coefficients along the path ($X_1 \rightarrow X_2$ and $X_2 \rightarrow Y$), giving 3τ. Thus, the *total effect* (direct and indirect effect) of manipulating X_1 is given by $3 + 3\tau$ in this example. We will further consider the implications of various causal diagrams in Chapter 7.

6.6 Strategies for addressing multicollinearity

As we saw in the last section, omitting important predictors can lead to biased regression parameter estimators.[3] Yet, it is common for researchers to drop one or more variables when they are highly correlated. For

[3]It may seem strange to see "parameter estimator" here rather than "parameter estimates". However, note that in statistics, bias is defined in terms of a difference between a fixed parameter and the *average* estimate across repeated samples (i.e., the

example, when faced with 2 highly correlated predictors, users may compare univariate regression models and select the variable that has the strongest association with the response variable. Alternatively, if one applies a stepwise model-selection routine (Section 8.4), it is likely that one of the highly correlated predictors will be dropped during that process because competing predictors will often look "underwhelming" when they are both included. What should we do instead?

As we will discuss in Chapter 8, it is paramount to consider how a model will be used when developing strategies for handling messy data situations, including multicollinearity. If our goal is to generate accurate predictions, then we may not care about the effect of any one variable in isolation (i.e., we may not be interested in how Y changes as we change X_1 while holding all other variables constant – we just want to be able to predict Y from the suite of available variables). In this case, we may choose to include all variables in our model even if their individual standard errors are inflated due to multicollinearity. We may want to consider regularization or penalization methods (see Section 8.7), such as ridge regression and the LASSO (Hoerl and Kennard 1970; Tibshirani 1996; Dormann et al. 2013). These methods trade off a small amount of bias to improve precision associated with regression parameter estimators. Yet, it is also important to recognize the challenges association with identifying the unique contributions of collinear variables; often, it will be best to simply consider their combined effects when performing inference. For example, if we have two variables that are highly collinear, we can use the methods from Section 3.12 to test whether both of their coefficients are 0 versus the alternative hypothesis that at least one of the two coefficients is non-zero.

Alternatively, one may create create new predictor variables that are orthogonal (i.e., not correlated), but these methods present their own challenges when it comes to interpretation. In the next sections, we will consider two such approaches suggested by Graham (2003):

- Residual and sequential regression
- Principal components regression

Graham (2003) also includes a short overview of structural equation models, which have a rich history, particularly in the social sciences and deserve their own treatment (by someone more well versed in their use than me!). One option would be Jarrett Byrnes's course. For a nice introduction, I also suggest reading Grace (2008).

6.7 Applied example: Modeling the effect of correlated environmental factors on the distribution of subtidal kelp

We begin by considering the data from Graham (2003), which are contained in the `Data4Ecologists` package.

```
library(Data4Ecologists)
data(Kelp)
```

The data consist of 38 observations with the following predictors used to model the shallow (upper) distributional limit of the subtidal kelp, *Macrocystis pyrifera*.

- `OD` = wave orbital displacement (in meters)
- `BD` = wave breaking depth (in meters)

mean of a *sampling distribution*). Thus, bias is associated with the *method* of generating estimates (i.e., the *estimator*), not the estimates themselves.

- `LTD` = average tidal height (in meters)
- `W` = wind velocity (in meters/s).

The distributional limit is contained in a variable named `Response`. We begin by calculating variance inflation factors for each of the predictor variables:

```
library(car)
vif(lm(Response~OD+BD+LTD+W, data=Kelp))
```

```
      OD       BD      LTD        W 
2.574934 2.355055 1.175270 2.094319 
```

Next, we use a pairwise scatterplot to explore the relationship among these predictors (Figure 6.6) using the `ggpairs` function in the `GGally` package (Schloerke et al. 2023). Importantly, we do not include the response variable in this plot because we hope to avoid having pairwise correlations between predictor and response variables influence our decisions regarding which predictors to include in our model. Although it is not uncommon to see this type of information used to eliminate some predictor variables from consideration, doing so increases the risk of overfitting one's data, leading to a model that fits the current data well but fails to predict new data in the future (see Fieberg and Johnson 2015, and Chapter 8).

```
library(GGally)
ggpairs(Kelp[,c("OD", "BD", "LTD", "W")],
        lower = list(continuous = "smooth"))
```

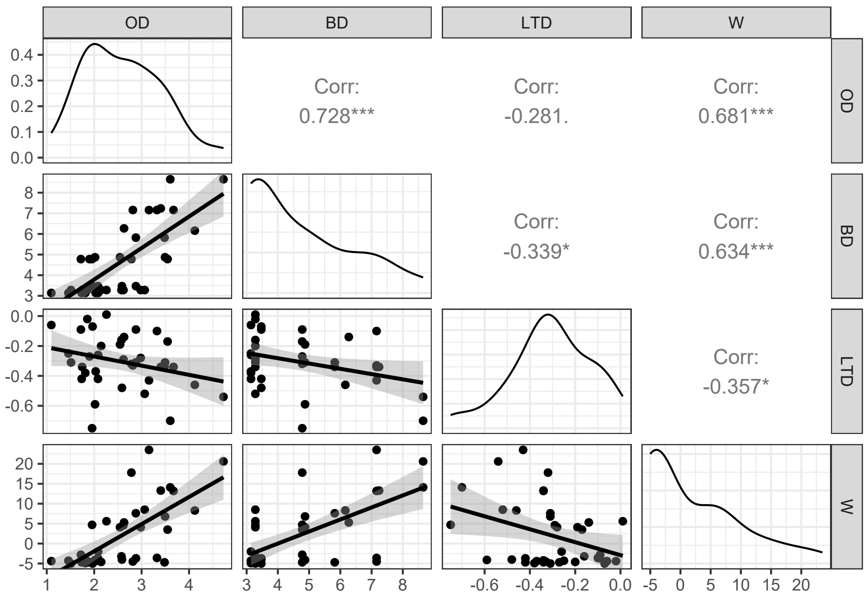


FIGURE 6.6 Pairwise scatterplot of predictor variables in the Kelp data set [@graham2003].

Although all of the variance inflation factors are < 5, the correlation between `OD` and `BD` is > 0.7 and the correlation between `W` and `OD` and `W` and `BD` are both > 0.6 (Figure 6.6).

6.8 Residual and sequential regression

Graham (2003) initially considers an approach he refers to as *residual and sequential regression*, in which variables are prioritized in terms of their order of entry into the model, with higher-priority variables allowed to account for unique as well as shared contributions to explaining the variability of the response variable, Y. Although I have not seen this approach applied elsewhere, it is instructive to consider how this approach partitions the variance of Y across a suite of correlated predictors.

Consider a situation in which you are interested in building a model that includes three correlated predictor variables. To apply this approach, we begin by ordering the variables based on their *a priori* importance, say X_1, X_2, and then X_3.[4] We then construct a multivariable regression model containing the following predictors:

- X_1
- $\tilde{X}_2 =$ the residuals of $\text{lm}(X_2 \sim X_1)$
- $\tilde{X}_3 =$ the residuals of $\text{lm}(X_3 \sim X_1 + X_2)$

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 \tilde{X}_{2i} + \beta_3 \tilde{X}_{3i} + \epsilon_i \tag{6.2}$$

In this case, β_1 will capture unique contributions of X_1 to the variance of Y as well as its shared contributions with X_2 and X_3, β_2 will capture contributions of X_2 to the variance of Y that are unique to X_2 or shared only with X_3 (the coefficient for X_1 is already accounting for any shared contributions of X_2 with X_1), and β_3 will capture the contributions of X_3 to the variance of Y that are not shared with X_1 or X_2 and thus unique only to X_3.

For the `Kelp` data set, Graham (2003) decided on the following order of greatest to least importance, wave orbital displacement (`OD`), wind velocity (`W`), average tidal height (`LTD`), and then wave breaking depth (`BD`). He then created the following predictors:

```
Kelp$W.g.OD<-lm(W~OD, data=Kelp)$resid
Kelp$LTD.g.W.OD<-lm(LTD~W+OD, data=Kelp)$resid
Kelp$BD.g.W.OD.LTD<-lm(BD~W+OD+LTD, data=Kelp)$resid
```

- `W.g.OD` = to capture the effect of `W` that is not shared with `OD`
- `LTD.g.W.OD` = to capture the effect of `LTD` that is not shared with `OD` or `W`
- `BD.g.W.OD.LTD` = to capture the effect of `BD` not shared with `OD`, `W`, or `LTD`

We can then fit a model that includes `OD` and all of these derived predictors:

```
seq.lm<-lm(Response~OD+W.g.OD+LTD.g.W.OD+BD.g.W.OD.LTD, data=Kelp)
summary(seq.lm)
```

```
Call:
```

[4] Graham (2003) suggests this ordering can be based on one's instincts, intuition, or prior or current data, but how to use this information when deciding upon an order is not discussed in detail.

```
lm(formula = Response ~ OD + W.g.OD + LTD.g.W.OD + BD.g.W.OD.LTD,
    data = Kelp)

Residuals:
      Min        1Q    Median        3Q       Max
-0.284911 -0.098861 -0.002388  0.099031  0.301931

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.747588   0.078192  35.139  < 2e-16 ***
OD             0.194243   0.028877   6.726 1.16e-07 ***
W.g.OD         0.008082   0.003953   2.045   0.0489 *
LTD.g.W.OD    -0.055333   0.141350  -0.391   0.6980
BD.g.W.OD.LTD -0.004295   0.021137  -0.203   0.8402
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1431 on 33 degrees of freedom
Multiple R-squared:  0.6006,    Adjusted R-squared:  0.5522
F-statistic: 12.41 on 4 and 33 DF,  p-value: 2.893e-06
```

To interpret the coefficients, we need to keep in mind how the predictors were constructed. For example, we would interpret the coefficient for `OD` as capturing unique contributions of `OD` as well as its shared contributions with `W`, `LTD` and `BD`. We would interpret the coefficient for `W.g.OD` as capturing the unique contributions of `W` as well as its shared contributions with `LTD` and `BD` that are not already accounted for by `OD`. The non-significant coefficients for `LTD.g.W.OD` and `BD.g.W.OD.LTD` could be interpreted as suggesting that the variables `LTD` and `BD` offer little to no new information about `Response` that is not already accounted for by `OD` and `W`.

Because of the way we created the variables, the variables are orthogonal - i.e., they are all uncorrelated. Thus, eliminating the lesser priority variables will not change the coefficients for the other variables.

```
seq.lm2<-lm(Response~OD+W.g.OD, data=Kelp)
summary(seq.lm2)$coef
```

```
               Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 2.747587774 0.076148241 36.082091 2.796476e-29
OD          0.194243475 0.028122800  6.906975 5.038589e-08
W.g.OD      0.008082141 0.003849614  2.099468 4.305538e-02
```

It is important to recognize, however, that we are still unable to identify the unique contributions of the different variables, and that the coefficients we estimate depend on how we prioritize the predictors.

6.9 Principal components regression

Another way to create orthogonal predictors is to perform a principal component analysis (PCA) on the set of predictor variables under consideration. A PCA will create new orthogonal predictors from linear combinations[5] of the original correlated variables:

[5]As a reminder, a linear combination is a weighted sum of the predictors.

$pca_1 = \lambda_{1,1}X_1 + \lambda_{1,2}X_2 + \ldots \lambda_{1,p}X_p$

$pca_2 = \lambda_{2,1}X_1 + \lambda_{2,2}X_2 + \ldots \lambda_{2,p}X_p$

$\ldots$

$pca_p = \lambda_{p,1}X_1 + \lambda_{p,2}X_2 + \ldots \lambda_{p,p}X_p,$

where p is the number of original correlated predictors and also the number of new orthogonal predictors that are created. These new predictors, pca_i, are often referred to as principal component scores. The λ's, often referred to as loadings, are unique to each pca_i and weight the contribution of each of the original variables when forming the new orthogonal variables.

Whereas the sequential regression approach from the last section created new orthogonal predictors through a prioritization and residual regression algorithm, a PCA creates new predictors such that pca_1 accounts for the greatest axis of variation in $(X_1, X_2, ..., X_p)$, pca_2 accounts for greatest axis of remaining variation in $(X_1, X_2, ..., X_p)$, not already accounted for by pca_1, etc[6]. Because of the way these variables are created, we might expect most of the information in the suite of variables will be contained in the first few principal components. PCAs are relatively easy to visualize in the 2-dimensional case (i.e., $p = 2$; see Figure 6.7)); the first principal component will fall upon the axis with the greatest spread, and the second principal component will fall perpendicular (orthogonal) to the first. This logic extends to higher dimensions but visualization is no longer so easy. When $p > 2$, it is common to plot just the first two principal components along with vectors that highlight the contributions of the original variables to these principal components (i.e., their loadings on the first two principal components); this type of plot is often referred to as a bi-plot (Figure 6.8).

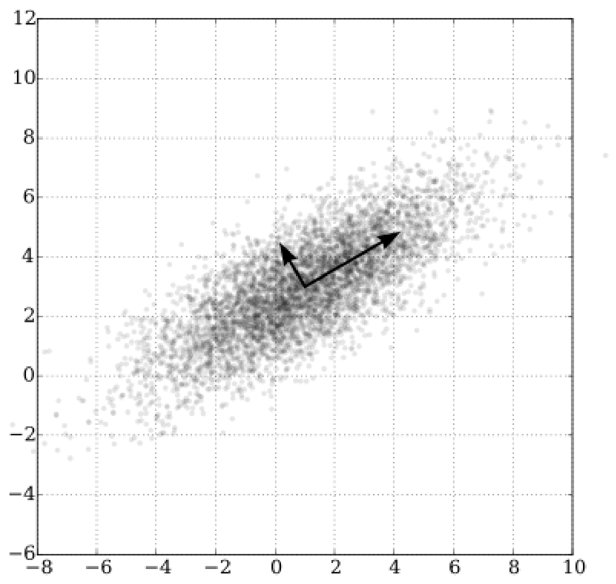


FIGURE 6.7 Principal component analysis (PCA) of a set of bivariate Normal random variables showing the axes associated with the first two principal components. From https://commons.wikimedia.org/wiki/File:GaussianScatterPCA.svg.

[6]For those readers that may have had a course in linear algebra, we can decompose Σ, the correlation matrix of $(X_1, X_2, ..., X_p)$ using $\Sigma = V\Lambda V^T$ where V is a $[n \times p]$ matrix of eigenvectors, equivalent to the principal components, and Λ is a diagonal matrix of eigenvalues.

There are lots of functions in R that can be used to implement a PCA. Here, we use the `princomp` function in R (R Core Team 2021). PCAs formed using the covariance matrix of the predictor data will depend heavily on how variables are scaled (remember, we are finding linear combinations of the original predictor data that explain the majority of the variance in our predictors; if we do not scale our predictors first, or use the correlation matrix, then our principal components will be dominated by the variables that are largest in magnitude as these will have the largest variance). Thus, to avoid these issues, we add the argument `cor=TRUE` to indicate that we want to form PCAs using the correlation matrix (which effectively scales all predictors to have mean 0 and standard deviation of 1) rather than use the covariance matrix of our original predictor variables.

```
pcas<-princomp(~OD+BD+LTD+W, data=Kelp, cor=TRUE, scores=TRUE)
summary(pcas)
```

```
Importance of components:
                          Comp.1    Comp.2     Comp.3     Comp.4
Standard deviation     1.6016768 0.8975073 0.60895215 0.50822166
Proportion of Variance 0.6413421 0.2013799 0.09270568 0.06457231
Cumulative Proportion  0.6413421 0.8427220 0.93542769 1.00000000
```

We see that the first principal component accounts for 64% of the variation in the predictors, the second principal component accounts for 20% of the variation, and the other two together account for approximately 15% of the variation. We can also look at the `loadings`, which give the λ's used to create our 4 principal components. The blank values are not 0 but just small (< 0.001), and the signs here are arbitrary (i.e., we could multiply all loadings in a column by -1 and not change the proportion of variance explained by the *pca*).

```
pcas$loadings
```

```
##
## Loadings:
##     Comp.1 Comp.2 Comp.3 Comp.4
## OD   0.548  0.290  0.159  0.768
## BD   0.545  0.179  0.581 -0.577
## LTD -0.338  0.934
## W    0.536  0.110 -0.795 -0.259
##
##                Comp.1 Comp.2 Comp.3 Comp.4
## SS loadings      1.00   1.00   1.00   1.00
## Proportion Var   0.25   0.25   0.25   0.25
## Cumulative Var   0.25   0.50   0.75   1.00
```

The new PCA variables are contained in `pca$scores`. We could also calculate these by "hand" if we first scaled and centered our original predictor data and then multiplied those values by the loadings as demonstrated below:

```
#scores by "hand" compared to scores returned by princomp
head(scale(as.matrix(Kelp[,2:5]))%*%pcas$loadings, n = 3)
```

```
         Comp.1    Comp.2      Comp.3      Comp.4
[1,] -0.1912783 -1.752736  0.66278941 -0.24694830
[2,]  0.6223409 -2.502387 -0.18091063 -0.46900655
[3,] -1.3326878 -0.919048  0.03361542  0.05590063
```

```
head(pcas$scores, n = 3)
```

```
      Comp.1     Comp.2      Comp.3      Comp.4
1 -0.1938459 -1.7762635  0.67168631 -0.25026319
2  0.6306949 -2.5359779 -0.18333907 -0.47530223
3 -1.3505770 -0.9313847  0.03406665  0.05665101
```

As mentioned previously, it is common to plot the first two sets principal component scores along with the loadings, e.g., using the `biplot` function in R (Figure 6.8). From this plot, we can see that the first principal component is largely determined by `OD`, `BD` and `W`, whereas the second principal component is mostly determined by `LTD`. We can also arrive at this same conclusion by inspecting the loadings on each principle component. For example, if we inspect the loading for the 2nd principal component, we see that the largest loading is associated with `LTD` (0.934) with all other loadings being less than 0.3. Furthermore, we can see that the 3rd predictor contributes little to the 3rd and 4th principal components.

```
biplot(pcas)
```

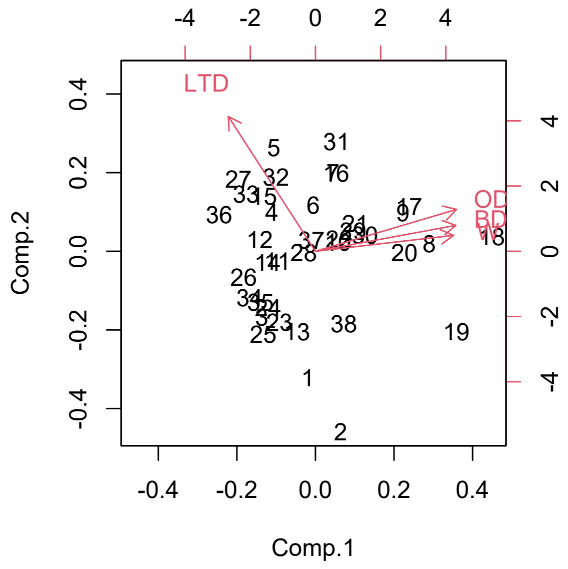


FIGURE 6.8 Bi-plot showing the first two principal components using the Kelp data set (Graham 2003), along with the loadings of the original variables.

It is important to recognize that pca_1 explains the greatest variation in $(X_1, X_2, ..., X_p)$ and not necessarily the greatest variation in Y (the same goes for the other $pcas$).[7] So although the last two principal components account for little variation in the original predictors, they may still account for significant variation in Y (Jolliffe 1982). Therefore, it is generally a good idea to include *all* principal components in a regression model (Graham 2003).

[7]There are a multitude of multivariate regression methods, such as partial least squares, that focus instead on creating orthogonal variables that explain covariation between explanatory and response variables; see e.g., Scott and Crone (2021).

```
Kelp<-cbind(Kelp, pcas$scores)
lm.pca<-lm(Response~ Comp.1 + Comp.2 + Comp.3 + Comp.4, data=Kelp)
summary(lm.pca)
```

```
Call:
lm(formula = Response ~ Comp.1 + Comp.2 + Comp.3 + Comp.4, data = Kelp)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.284911 -0.098861 -0.002388  0.099031  0.301931 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  3.24984    0.02321 140.035  < 2e-16 ***
Comp.1       0.09677    0.01449   6.678 1.33e-07 ***
Comp.2       0.02931    0.02586   1.134    0.265    
Comp.3      -0.03564    0.03811  -0.935    0.356    
Comp.4       0.07722    0.04566   1.691    0.100    
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 0.1431 on 33 degrees of freedom
Multiple R-squared:  0.6006,    Adjusted R-squared:  0.5522 
F-statistic: 12.41 on 4 and 33 DF,  p-value: 2.893e-06
```

Since the pca_i’s are orthogonal, the coefficients will not change if we drop one or more of them (as demonstrated below):

```
lm.pca2<-lm(Response ~ Comp.1, data = Kelp)
summary(lm.pca2)
```

```
## 
## Call:
## lm(formula = Response ~ Comp.1, data = Kelp)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.28269 -0.09537  0.00437  0.06927  0.35765 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  3.24984    0.02385 136.265  < 2e-16 ***
## Comp.1       0.09677    0.01489   6.499 1.51e-07 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
## 
## Residual standard error: 0.147 on 36 degrees of freedom
## Multiple R-squared:  0.5398, Adjusted R-squared:  0.527 
## F-statistic: 42.23 on 1 and 36 DF,  p-value: 1.507e-07
```

Yet, if we used the coefficients and loadings associated with the remaining *pca*'s (here, just pca_1) to estimate the effects of the original variables, we would find that they differ from those estimated using `lm(Response ~ OD + W + LTD + BD)`. In the end, as a data-reduction technique (Harrell Jr 2015), we might choose to capture the combined effect of `OD`, `W`, `LTD` and `BD` using a single principal component (and 1 model degree of freedom). A downside of this approach is that this principal component can be difficult to interpret as it is a function of all of the original predictor variables. In addition, it may be useful to consider methods, such as partial least squares, that consider covariation between predictor and response variables when constructing orthogonal predictors (Scott and Crone 2021).

6.10 Other methods

There are several other multivariate regression techniques that one could consider when trying to eliminate collinear variables. In particular, one might consider combining similar variables (e.g., "weather variables") using a PCA or variable clustering or scoring technique that creates an index out of multiple correlated variables (e.g., weather severity formed by adding a 1 whenever temperatures fall below, or snow depth falls above, pre-defined thresholds, DelGiudice et al. 2002; Dormann et al. 2013; Harrell Jr 2015). Scoring techniques are not that different from a multiple regression model, except rather than attempt to estimate optimal weights associated with each predictor (i.e., separate regression coefficients), variables are combined by assigning a +1, 0, -1 depending on whether the value of each predictor is expected to be indicative of larger or smaller values of the response variable, and then a single coefficient associated with the aggregated index is estimated.[8] There are also several statistical methods (e.g., factor analysis, partial least squares, structural equation models, etc) that use latent variables to represent various constructs (e.g., personality, size, etc) that can be informed by or are the products of multiple correlated variables. We will not discuss these methods here, but note that they are popular in the social sciences. Some of these methods are touched on in Dormann et al. (2013).

[8]Jacob Cohen, a famous statistician, argued that we might be better off using these indices directly for inference rather than regression models when faced with a small number of observations and many correlated predictors; see Cohen (1992)

7

Causal Inference

In this section, I will provide a very brief introduction to some of the concepts and tools used to evaluate evidence for causal effects from observational data.

Learning objectives

1. Gain a deeper appreciation for why *correlation (or association) is not the same as causation*
2. Discover basic rules that allow one to determine dependencies (correlations) among variables from an assumed causal network
3. Understand how causal networks can be used to inform the choice of variables to include in a regression model

Credit: this section was heavily influenced by Daniel Kaplan's chapter on Causation in his Statistics: A Fresh Approach (D. Kaplan 2009) and the excellent introduction to causal inference (Dablander 2020) from which many of the examples are drawn.

7.1 R Packages

We begin by loading a few packages upfront:

```
library(modelsummary) # for tables
library(kableExtra) # for tables
```

In addition, we will explore functions for evaluating conditional (in)dependencies in a Directed Acyclial Graph (DAG) using the `ggm` package (Marchetti, Drton, and Sadeghi 2020).

7.2 Introduction to causal inference

Most of the methods covered in introductory and even advanced statistics courses focus on methods for quantifying associations. For example, the regression models covered in this course quantify linear and non-linear associations between explanatory $(X_1, X_2, \dots, X_k)$ variables and a response variable (Y). Yet, most interesting research questions emphasize understanding the causes underlying the patterns we observe in nature, and information about causes and effects are critical for informing our thinking, decision making, government policies, etc. For example, an understanding of causation is needed to answer questions about:

- What will happen if we intervene in a system or manipulate one or more variables in some way (e.g., will we increase our longevity if we take a daily vitamin? Will we cause more businesses to leave the state if we increase taxes? How will a vaccine mandate influence disease transmission, unemployment rates, etc? Will we be able to reduce deer herds if we require hunters to shoot a female deer before they can harvest a male deer (sometimes referred to as an *earn-a-buck* management strategy[1])?
- Scenarios that we will never be able to observe (e.g., would George Floyd still be alive today had he been white? Would your female friend or neighbor have been promoted if she had been male? Would there have been fewer bird collisions with the Vikings stadium if it had been built differently?); These types of questions involve counterfactuals - and require considering what might have happened in an alternative world where the underlying conditions were different.

Importantly, we can't answer these questions using information on associations alone. For example, if we observe that X and Y are correlated, this could be due to X causing Y, Y causing X, or some other variable (or set of variables) causing both X and Y. Answers to questions like those above also may require consideration of both *direct* and *indirect effects*. For example, consider whether increasing taxes on businesses will increase the likelihood that they will locate to another state. Increasing taxes will almost surely have a *direct* negative effect on the likelihood of keeping a business in the state since no CEO wants to voluntarily pay more taxes to the government. On the other hand, taxes may have a positive *indirect* effect if the government spends its money wisely. For example, strong public schools, updated roads and bridges, and access to parks and trails may all help businesses attract skilled workers.

When you took a introductory statistics class, you probably heard your instructor say at least once, "*correlation is not causation.*" Furthermore, you probably learned about some of the challenges associated with inferring causation from observational data and the benefits of performing experiments whenever possible to establish causal linkages. A key challenge with inferring causation from observational data is that there are almost always confounding variables (variables correlated with both the predictor of interest and the response variable) that could offer an alternative explanation for why the predictor and response variables are correlated. One of my favorite examples of confounding is from R. H. Lock et al. (2020) (Figure 7.1), simply because it let's me talk about my father-in-law and the fact that he has collected a large number of old televisions. If you look at the average life expectancy versus the number of televisions (TVs) per person in different countries, there is a clear positive correlation (Figure 7.1). Does that mean my father-in-law will live forever thanks to his stockpiles of TVs? Should we all go out an collect old televisions that no one wants? Clearly no. That is not why my father-in-law has collected so many televisions – he just really likes to watch TV and doesn't like to throw away electronics that still work. While most students can quickly identify possible confounding variables in this case (e.g., access to health care that often accompanies the wealth that spurs TV consumption)[2], other cases may not be so clear. Thus, we need to be leery of inferring causality, especially from observational studies.

Evolution has equipped human minds with the power to explain patterns in nature really well, and we can easily jump to causal conclusions from correlations. Consider an observational study reporting that individuals taking a daily vitamin had longer lifespans. At first, this may seem to provide strong evidence for the protective benefits of a daily vitamin. Yet, there are many potential explanations for this observation – individuals that take a daily vitamin may be more risk averse, more worried about their health, eat better, exercise more, have more money, etc – or, perhaps a daily vitamin is actually beneficial to one's health. We can try to control or adjust for some of these other factors when we have the data on them, but in observational studies there will almost always be unmeasured variables that could play an important causal role in the relationships we observe. The reason experiments are so useful for establishing causality is that by randomly assigning individuals to treatment groups (e.g., to either take a vitamin or a placebo), we break

[1] https://www.realtree.com/deer-hunting/articles/earn-a-buck-was-it-the-greatest-deer-management-tool

[2] In most years, my students discover Tyler Vigen's web site (http://tylervigen.com/spurious-correlations), which offers many other silly examples of ridiculously strong correlations over time between variables that do not themselves share a causal relationship.

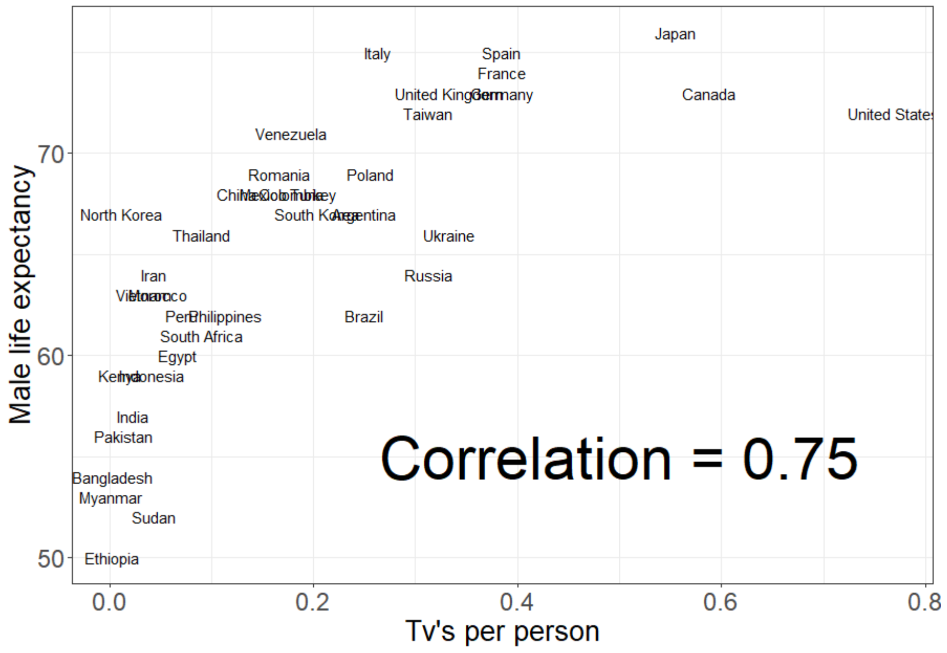


FIGURE 7.1 Association between male life expectancy and per capita ownership of televisions. Data from Rossman (1994).

any association between the treatment variable and possible confounders. Thus, if we see a difference between treatment groups, and our sample size is large, we can be much more assured that the difference is due to the treatment and not some other confounding variable.

One might walk away from an introductory statistics course thinking that the *only* way to establish causality is through experimentation. Yet, tools for inferring causality from observational data have also been around for a long time (e.g., Sewell Wright invented path analysis in the early 1920's; Wright 1921). Furthermore, computer scientists, econometricians, and statisticians have made a lot of progress over the past few decades in developing new theory and methods for inferring causality from observational data. Most of these approaches require assumptions about how the world works, encoded in a causal diagram or directed acyclical graph (DAG). In this section, I will briefly introduce DAGs and describe how they can be used to inform the choice of an appropriate regression model for estimating causal effects. Yet, this section will barely scratch the surface when it comes to causal inference. More in depth treatments can be found, e.g., in J. Pearl (2000), Glymour, Pearl, and Jewell (2016), Judea Pearl, Glymour, and Jewell (2016), and Judea Pearl and Mackenzie (2018).

7.3 Directed acyclical graphs and conditional independencies

Directed acyclical graphs (DAGs) represent causal pathways connecting nodes (either observed or unobserved variables) in a system, and thus, represent our understanding of how we think the world works. Connections between variables are *directed*, meaning that arrows are drawn so that one can distinguish cause from effect (cause $\rightarrow$ effect). We will only consider graphs that are acyclical, meaning that they do not eventually reconnect to form a closed loop (e.g., $X \rightarrow Z \rightarrow Y \rightarrow X$), though this framework could still be used in the context of time series data if we clearly specify the time of each observation (e.g., through a subscript, $X_t \to Y_t \to X_{t+1}$). Sometimes it may be beneficial to consider a correlation between two variables without an associated causal relationship. For example, $X \Leftrightarrow Z$ may be used to denote a non-causal connection between X and Z due to an unobserved variable, U ($X \leftarrow U \to Z$) that affects them both; strictly speaking, the graph will not be fully directed in this case.

As we will see, DAGs are central to understanding which predictor variables should be included in a regression model when attempting to estimate causal effects using observational data. At the most basic level, there are three types of causal connections between variables that need to be considered (arrows, below, indicate the direction of causal effects); these connections will help determine whether variables are dependent (i.e., associated) or independent after conditioning on one or more other variables in the system (Judea Pearl 1995; J. Pearl 2000):

- *chain*: $X \to Z \to Y$, with Z referred to as a mediator variable on the causal path from X to Y
- *fork*: $X \leftarrow Z \to Y$, with Z referred to as a common cause
- *inverted fork*: $X \to Z \leftarrow Y$, with Z referred to as a collider variable

Assume for now that X, Z, and Y are the only 3 variables that interact in a system, and we are interested in learning whether X causes Y (i.e., that the true relationship is $X \to Y$ or $X \to Z \to Y$). If correlation coefficients and regression models are our primary tools for inference, then it is important to know how they behave under these and alternative scenarios (i.e., connections involving a fork or inverted fork).

In the case of a chain, X and Y will be marginally dependent[3], meaning that we should expect the correlation between X and Y to be non-zero if we do not adjust for Z. However, adjusting for Z will make X and Y (conditionally) independent. In other words, once we know the value of Z, X adds no useful information for understanding Y. Using the *independence symbol*, $\perp\!\!\!\perp$, we can represent these two findings as: $X \not\perp\!\!\!\perp Y$ (X and Y are *not* independent) and $X \perp\!\!\!\perp Y|Z$ (X and Y are independent if we condition on Z). The top panels of Figure 7.2 provide an illustrative example when Z is a binary mediator variable.

Similarly, in the case of a *fork* or common cause, X and Y will be marginally correlated, but X and Y will be independent if we condition on Z. Thus, a common cause will result in a spurious correlation between two unrelated variables (e.g., ice cream sales and number of severe sun burns; Figure 7.3) unless we condition on their common cause. This is a classic example used to highlight the impact of a confounding variable, Z (summer temperature).

One might get the impression, based on the above example and simple discussions of confounding variables in introductory statistics courses, that it is always best to include or adjust for other variables when fitting regression models. However, we can also create a spurious correlation by conditioning on a collider variable in an inverted fork. Specifically, if $X \leftarrow Z \to Y$, then X and Y will be independent *unless* we condition on Z (i.e., $X \perp\!\!\!\perp Y$, but $X \not\perp\!\!\!\perp Y|Z$; see bottom row of panels in Figure 7.2). In essence, knowing X tells us nothing about Y, that is, unless we are also given information about Z. Once we have information about Z, then knowing something about X gives us additional insights into likely values of Y.

In summary, there are multiple DAGs that we could consider as representations of the causal connections between the variables X, Y and Z (Figure 7.4). The three DAGs on the left all result in the same set of (conditional) dependencies: $X \not\perp\!\!\!\perp Y$, but $X \perp\!\!\!\perp Y|Z$. Thus, it would be impossible to distinguish among them based on associations alone. The remaining DAG, where Z is a collider, results in the opposite set of (conditional) dependencies $X \perp\!\!\!\perp Y$, but $X \not\perp\!\!\!\perp Y|Z$.

[3]Marginal here refers to the *unconditional* relationship between X and Y rather than the *strength* of this relationship.

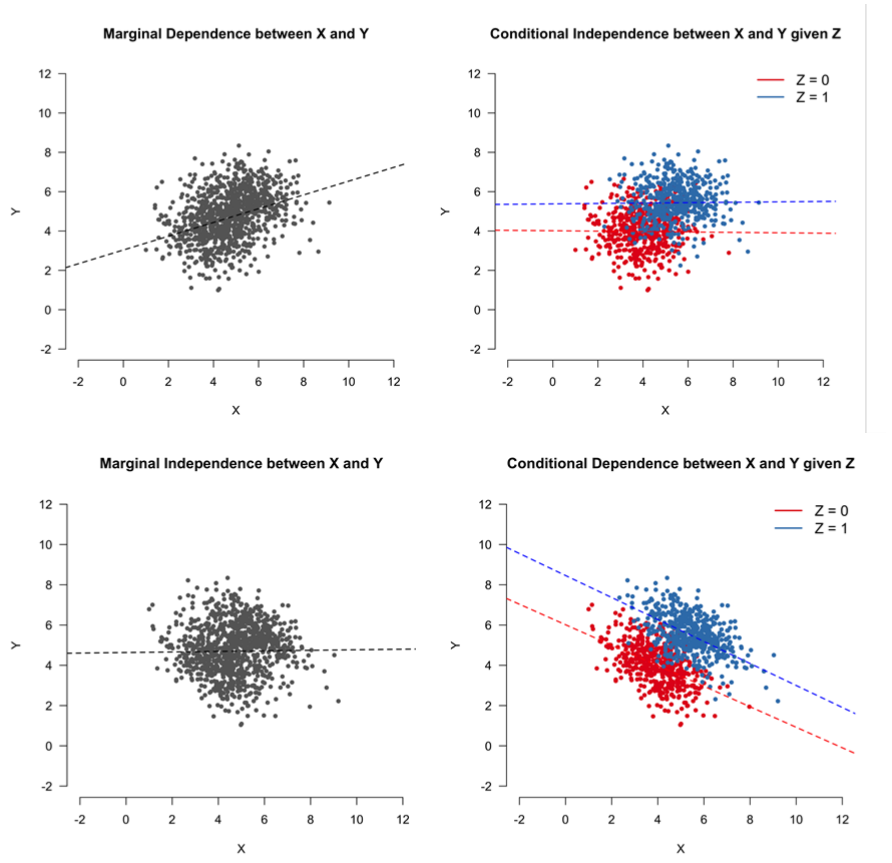


FIGURE 7.2 Two types of associations between X and Y, with and without conditioning on a third variable, Z. In the top panels, X and Y are marginally dependent (i.e., $X \not\perp\!\!\!\perp Y$), but become independent when we condition on Z (i.e., $X \perp\!\!\!\perp Y|Z$). This relationship could be due to a chain or fork. The opposite is true in the lower panels, which represent a scenario where Z is a collider variable. This figure was constructed by combining Figures 2 and 4 from Dablander (2020). CC BY 4.0.

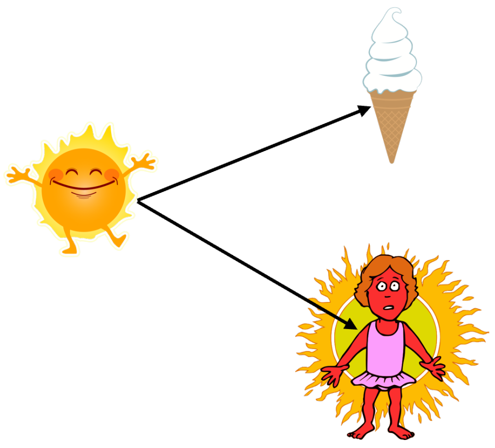


FIGURE 7.3 Example of a common cause or fork creating a spurious correlation between two unrelated variables (ice cream consumption and sunburns). Figure created by J. Fieberg using clipart in the public domain.

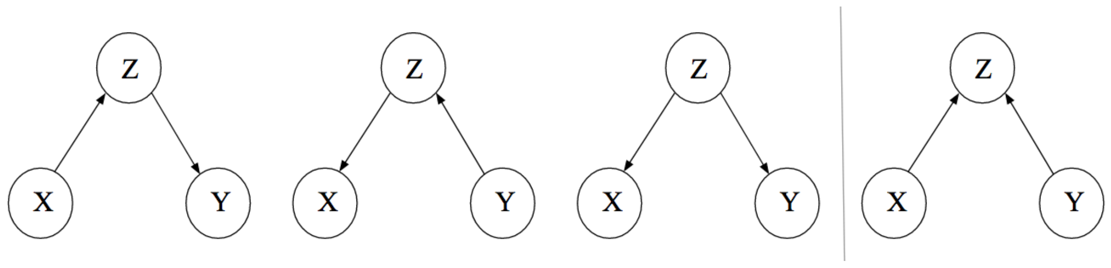


FIGURE 7.4 Figure 3 from Dablander (2020) CC BY 4.0 showing different directed acyclical graphs involving 3 variables, X, Y, and Z. The first three DAGs on the left result in the following set of (conditional) relationships: $X \not\perp\!\!\!\perp Y$, but $X \perp\!\!\!\perp Y|Z$. The DAG on the far right represents a case where Z is a collider variable, which implies $X \perp\!\!\!\perp Y$, but $X \not\perp\!\!\!\perp Y|Z$.

7.3.1 Collider bias

Although the bias caused by confounding variables (e.g., a common cause) is well known, many readers may be surprised to hear that a spurious correlation can be created when we adjust for a variable. Thus, we will demonstrate this issue with a simple simulation example. Consider a survey of students at the University of Minnesota, where students are asked whether they are taking one or more classes on the St. Paul campus ($Z_i = 1$ if yes and 0 otherwise), their level of interest in food science and nutrition (X on a 10-point scale) and how many days they spent fishing in the past 3 years (Y). Because the St. Paul campus hosts the Department of Food Science and Nutrition and the Department of Fisheries, Wildlife, and Conservation Biology, we might expect students with high values of X or high values of Y to be taking one or more courses on the St. Paul campus. It is also conceivable that X and Y would be marginally independent (i.e., there is no causal connection between a person's interest in nutrition X and how often they went fishing Y). We simulate data reflecting these assumptions[4]:

- We generate 10,000 values of X from a uniform distribution between 0 and 10.
- We generate 10,000 values of Y, **independent of** X, using a Poisson distribution with mean, λ, equal to 4.
- We determine the probability that each student is taking a class on the St. Paul campus, p_i, using:

$$p_i = \frac{exp(-5 + 2X + 2Y)}{1 + exp(-5 + 2X + 2Y)}$$

When X and Y are both 0, p_i will equal 0.007; p_i will increase as either X or Y increases. Thus, students that are really interested in nutrition and students that really like to fish are both likely to be taking classes on the St. Paul campus. Lastly, we generate Z for each student using p_i and a Bernoulli distribution (equivalent to flipping a coin with probability p_i of a student taking one or more classes on the St. Paul campus).

```r
# Set seed of random number generator
set.seed(1040)
# number of students
n <- 10000
# Generate X = interest in nutrition and food science
x <- runif(n, 0, 10)
# Generate number of days fishing
y <- rpois(n, lambda=4)
```

[4]We will learn more about the distributions used to simulate data when we get to Chapter 9.

TABLE 7.1 Estimates (p-values) of marginal and conditional relationships between unrelated variables, $X$ and $Y$, with and without adjusting for a collider variable, $Z$.

	Model 1	Model 2
(Intercept)	3.970 (<0.001)	1.457 (<0.001)
x	0.004 (0.535)	−0.036 (<0.001)
z		2.800 (<0.001)

```
# Generate whether students are taking classes on St. Paul campus
p <- exp(-5 + 2*x + 2*y)/(1+exp(-5 + 2*x + 2*y))
z <- rbinom(n, 1, prob=p)
```

We then explore marginal and conditional relationships between $X$ and $Y$ by fitting regression models with and without $Z$ (Table 7.1). We see that the coefficient for $X$ is near 0 and not statistically significant when we exclude $Z$ from the model (p = 0.535). This result is not surprising, given that we generated $X$ and $Y$ independently. However, $X$ becomes negatively correlated with $Y$ (and statistically significant) once we include $Z$. Thus, including the predictor $Z$ creates a spurious negative correlation between students' interest in fishing and their interest in nutrition (implying that liking one spurs dislike of the other), a phenomenon often referred to as collider bias.

```
mod1 <- lm(y ~ x)
mod2 <- lm(y ~ x + z)
```

Importantly, collider bias can also occur if we restrict the study population using information in the collider variable, $Z$ (Cole et al. 2010). For example, we also find that interest in nutrition and fishing are negatively correlated if we restrict ourselves to the population of students taking classes on the St. Paul campus:

```
collider.dat<-data.frame(x=x, y=y, z=z)
summary(lm(y ~ x, data=subset(collider.dat, z==1)))
```

```
##
## Call:
## lm(formula = y ~ x, data = subset(collider.dat, z == 1))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1936 -1.2010 -0.1128  1.0351  9.1015
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.251386   0.041596 102.208  < 2e-16 ***
## x           -0.035381   0.007101  -4.982 6.39e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.985 on 9701 degrees of freedom
## Multiple R-squared:  0.002553,   Adjusted R-squared:  0.00245
## F-statistic: 24.83 on 1 and 9701 DF,  p-value: 6.386e-07
```

In essence, an interest in nutrition or an interest in fishing might lead someone to be represented in the study population of students on the St. Paul campus. Those students that end up in St. Paul due to their high interest in fishing likely have an average (or slightly-below average) level of interest in nutrition (assuming that interest in fishing and nutrition are independent in the full population of students). Similarly, students studying nutrition likely have an average to slightly below average level of interest in fishing. When we combine these two sets of students (students studying nutrition and students studying fisheries, wildlife, or conservation biology), we end up with a negative association between interest in fishing and interest in nutrition in the study population.

Similar concerns have been raised when analyzing data from hospitalized patients. As one example, Sackett (1979) found an association between locomotor disease and respiratory disease in hospitalized patients but not in the larger population. This result could be explained by a DAG in which hospitalization serves as a collider variable (Figure 7.5).

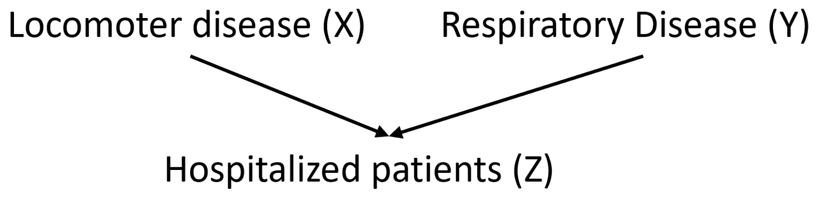


FIGURE 7.5 Example of association bias due to selecting cases using a collider variable, hospitalization (yes or no).

The importance of collider bias has also been recently highlighted in the context of occupancy models (MacKenzie et al. 2017), which are widely used in ecology. Using a simulation study, Stewart et al. (2023) showed that estimates of effect sizes associated with covariates influencing occupancy probabilities were biased when collider variables were considered for inclusion and information criterion (AIC and BIC; see Chapter 8) were used to select an appropriate model.

7.4 *d*-separation

We can use these same 3 basic rules to help determine, in larger causal networks, whether variables X and Y are independent after potentially conditioning on a set of other variables, Z. However, we must consider all paths connecting X and Y. We will refer to paths connecting variables as being either "open/correlating" or "closed/blocked". If you forget the 3 basic rules, it can often help to think of flowing electricity when determining if a path is open or closed (if you can get from X to Y, starting anywhere in the network then the path is open).

- *chain*: $X \rightarrow Z \rightarrow Y$: if we add electricity to X, it will flow to Y. Thus, the pathway is open unless we include Z which will "block" the path.
- *fork*: $X \leftarrow Z \rightarrow Y$: if we add electricity to Z, it will flow to both X and Y. Thus, the pathway between X and Y is open unless we include (i.e., condition on) Z which will block this path.
- *inverted fork*: $X \rightarrow Z \leftarrow Y$: if we add electricity to either X or Y it will flow to Z and get stuck. If we add electricity to Z it will remain there. Thus, there is no way to connect X and Y unless we condition on Z, which will open the pathway. It turns out, that conditioning on any of the descendants of Z will also open this pathway (Glymour, Pearl, and Jewell 2016). A descendant of Z is any variable that has an arrow (or set of arrows) that lead from Z into that variable.

Let's now consider a more complicated causal network (Figure 7.6 from Dablander 2020).

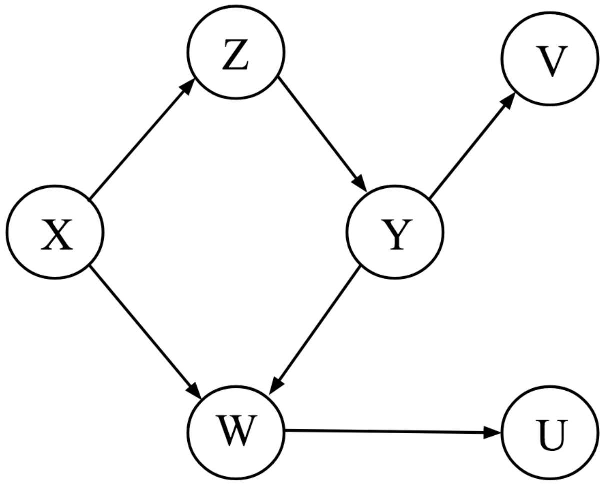


FIGURE 7.6 Figure 5 from Dablander (2020) representing a causal diagram for 6 variables. CC BY 4.0

To determine if two variables are dependent after conditioning on one or more variables, we will use the following steps:

1. Write down all paths connecting the two variables.
2. Determine if any of the paths are open/correlating. If any of the paths are open, then the two variables will be dependent. If all of the paths are blocked, then the two variables will be (conditionally) independent. When this occurs, we say that the variables are d-separated by the set of conditioning variables.

Let's consider variables X and Y in Figure 7.6. There are two paths connecting these variables:

- $X \rightarrow Z \rightarrow Y$
- $X \rightarrow W \leftarrow Y$

The first path is a chain and is correlating (unless we condition on Z). The second path is an inverted fork and will be closed (unless we condition on W or its descendant U). Thus, we can infer the following set of conditional (in)dependencies:

- $X \not\perp\!\!\!\perp Y$ (due to the first path being open)
- $X \perp\!\!\!\perp Y|Z$ (conditioning on Z will close this open path)
- $X \not\perp\!\!\!\perp Y|Z, W$ (conditioning on Z will close the first path, but conditioning on W will open up the second path)
- $X \not\perp\!\!\!\perp Y|Z, U$ (conditioning on Z will close the first path, but conditioning on U, which is a descendant of the collider Z, will open up the second path)

We can use functions in the `ggm` package to confirm these results (Marchetti, Drton, and Sadeghi 2020). We begin by constructing the DAG using the `DAG` function to capture all arrows flowing into our variables:

```
library(ggm)
dag1<-DAG(W ~ X + Y,
          Z ~ X,
          Y ~ Z,
          V ~ Y,
          U ~ W)
```

We can then use the `dSep` function to test whether two variables X and Y (`first` and `second` arguments of the `dSep` function) are d-separated after conditioning on one or more variables (`cond` argument). Here, we confirm the 4 results from before:

```
dSep(dag1, first = "X", second= "Y", cond=NULL)
```

```
## [1] FALSE
```

```
dSep(dag1, first = "X", second= "Y", cond="Z")
```

```
## [1] TRUE
```

```
dSep(dag1, first = "X", second= "Y", cond=c("Z","W"))
```

```
## [1] FALSE
```

```
dSep(dag1, first = "X", second= "Y", cond=c("Z","U"))
```

```
## [1] FALSE
```

7.5 Estimating causal effects (direct, indirect, and total effects)

As was mentioned in the introduction to this section, when we want to intervene in a system, we often find that there are both direct and indirect effects on other variables. Formal methods have been developed (e.g., do calculus for calculating effects of interventions; Judea Pearl, Glymour, and Jewell 2016; Dablander 2020). Although we will not go into detail about these methods here, it is useful to recognize that when we intervene in a system and set $X = x$, this will effectively eliminate all connections flowing into X. Consider all possible systems connecting variables X, Y, and Z (Figure 7.7). Setting $X = x$ will have no effect on Y in the second, third, or fourth cases, since there are no directed paths connecting X to Y in these DAGS. This basic understanding underlies the mathematics behind quantifying the effects of interventions. For more details, we refer to Judea Pearl, Glymour, and Jewell (2016) and Dablander (2020).

We can also consider how we can use DAGs to determine whether to include or exclude a variable from a regression model depending on whether we are interested in estimating a direct effect or total (sum of direct and indirect) effect. Let's start by considering two examples from Judea Pearl, Glymour, and Jewell (2016) and Dablander (2020) (Figure 7.8). Panel A depicts a situation in which a treatment (T) has, on average, a positive effect on patient's response (R). A patient's response will also depend on their sex (S). Lastly, the sex of the patient influences whether or not they will be treated. If we want to estimate the effect of the treatment, represented by $T \rightarrow R$, we need to determine whether or not to adjust for sex.

Let's start by writing down all paths that connect T to R:

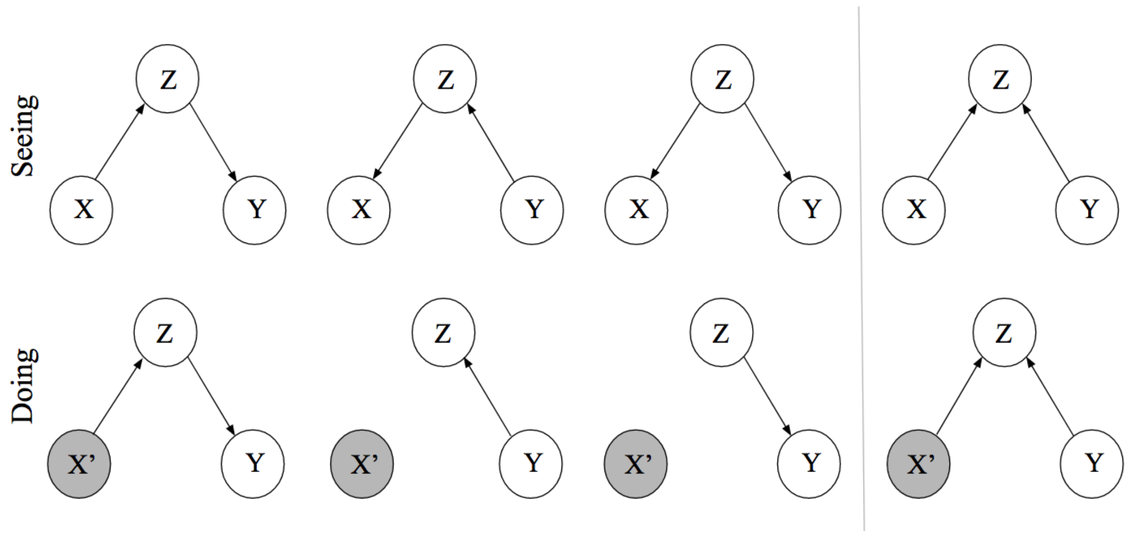


FIGURE 7.7 Figure 6 from Dablander (2020) representing the DAGs in Figure 7.4 after intervening in the system by setting $X = x$. CC BY 4.0

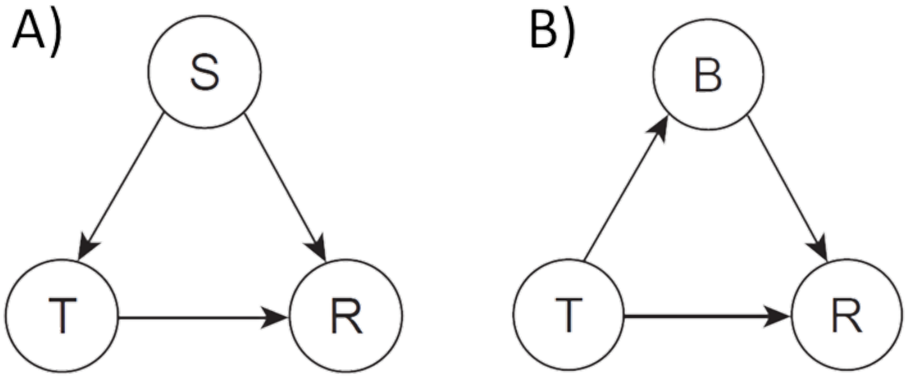


FIGURE 7.8 Two DAGS from Judea Pearl, Glymour, and Jewell (2016) and Dablander (2020) representing different causal networks from which we want to calculate the (total) effect of a treatment, T on an outcome R. CC BY 4.0

- $T \to R$
- $T \leftarrow S \to R$

When fitting a regression model, we want the coefficient for T to reflect the causal effect (i.e., the first path). However, the second path will also be correlating due to the common cause S. If we do not block this path, our coefficient for T will reflect associations arising from both of these paths. Thus, in this situation, we would want to include S in our regression model when estimating the effect of T on R.

Now, let's consider a second example where the treatment has both a direct effect on the response and an indirect effect through a variable B (Panel B in Figure 7.8). For example, a drug might have a direct effect on a patients recovery but also an indirect effect by increasing the patient's blood pressure. We again have two paths that connect T and R:

- $T \to R$

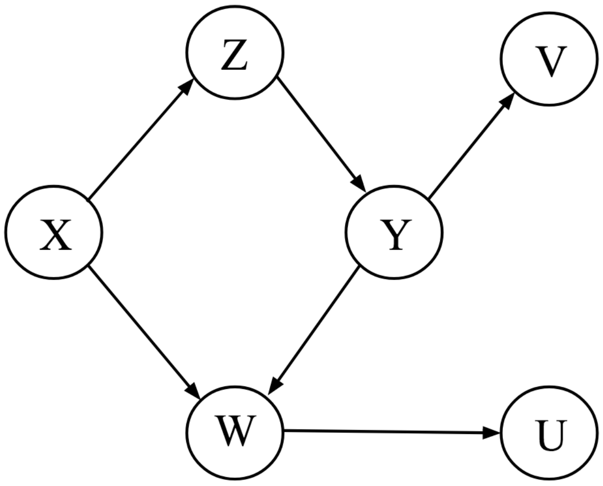
- $T \rightarrow B \rightarrow R$

Again, both of these paths are correlating. If we fit a model that includes only T, then the coefficient for T will reflect both its direct and indirect effects on R. Thus, if our interest is in knowing the overall (total) effect of the treatment on the patient's response, we would **not** want to include B as it would block the second path. On the other hand, if we wanted to estimate the *direct* effect of T on R, then we would want to block this path and include B in our model.

More generally, when we want to estimate a causal effect of X on Y, we need to consider the backdoor criterion (Judea Pearl, Glymour, and Jewell 2016). Specifically, we will need to

1) Block all spurious (non-causal) paths between X and Y
2) Leave all directed paths from X to Y unblocked (i.e., do not include mediator variables on the path between X and Y), and
3) Make sure not to create spurious correlations by including colliders that connect X to Y

Consider again Figure 7.6 (shown again, below). If we want to calculate the causal effect of Z on U, we can begin by writing down all paths connecting the two variables.



- $Z \rightarrow Y \rightarrow W \rightarrow U$
- $Z \leftarrow X \rightarrow W \rightarrow U$

The first path is our directed path connecting Z to U (and our effect of interest). The second is a correlating path due to a common cause, X. To satisfy the backdoor criterion, we need to 1) block the second path (e.g., by including X), and 2) make sure not to include Y or W as these would block our path of interest. See Laubach et al. (2021) for additional examples and discussion regarding how DAGs should be used to choose appropriate predictor variables when estimating causal effects.

7.6 Some (summary) comments

We have seen how simple causal diagrams can help with understanding if and when we should include variables in regression models. Hopefully, you will keep these ideas in mind when learning about other less thoughtful, data-driven methods for choosing a model (Chapter 8). In particular, it is important to recognize that a "best fitting" model may not be the most appropriate one for addressing your particular research question, and in fact, it can be misleading (Luque-Fernandez et al. 2019; Stewart et al. 2023; Arif and MacNeil 2022; Addicott et al. 2022).

One challenge with implementing causal inference methods is they rely heavily on assumptions (i.e., an assumed graph capturing causal relationships between variables in the system). And although it is sometimes possible to work backwards, using the set of observed statistical independencies in the data to suggest or rule out possible causal models, multiple models can lead to the same set of statistical independencies (Shipley 2002). In these cases, experimentation can prove critical for distinguishing between competing hypotheses.

Lastly, it is important to consider multiple lines of evidence when evaluating the the strength of evidence for causal effects. In that spirit, I end this section with Sir Austin Bradford Hill's[5] suggested criteria needed to establish likely causation (the list below is taken verbatim from https://bigdata-madesimple.com/how-to-tell-if-correlation-implies-causation/):

- *Strength*: A relationship is more likely to be causal if the correlation coefficient is large and statistically significant.
- *Consistency*: A relationship is more likely to be causal if it can be replicated.
- *Specificity*: A relationship is more likely to be causal if there is no other likely explanation.
- *Temporality*: A relationship is more likely to be causal if the effect always occurs after the cause.
- *Gradient*: A relationship is more likely to be causal if a greater exposure to the suspected cause leads to a greater effect.
- *Plausibility*: A relationship is more likely to be causal if there is a plausible mechanism between the cause and the effect
- *Coherence*: A relationship is more likely to be causal if it is compatible with related facts and theories.
- *Experiment*: A relationship is more likely to be causal if it can be verified experimentally.
- *Analogy*: A relationship is more likely to be causal if there are proven relationships between similar causes and effects.

[5]Sir Austin Bradford Hill was a famous British medical statistician.

8 Modeling Strategies

Learning objectives

1. Gain an appreciation for challenges associated with selecting among competing models and performing multi-model inference.
2. Understand common approaches used to select a model (e.g., stepwise selection using p-values, AIC, Adjusted R^2).
3. Understand the implications of model selection for statistical inference.
4. Gain exposure to alternatives to traditional model selection, including full model inference (df spending), model averaging, and penalized likelihood/regularization techniques.
5. Be able to evaluate model performance using cross-validation and model stability using the bootstrap.
6. Be able to choose an appropriate modeling strategy, depending on the goal of the analysis (*describe*, *predict*, or *infer*).

8.1 R Packages

We begin by loading a few packages upfront:

```r
library(kableExtra) # for creating tables
options(kableExtra.html.bsTable = T)
library(dplyr) # for data wrangling
library(broom) # for pulling off coefficients and SEs from lm
library(ggplot2) # for plotting
```

In addition, we will use data and functions from the following packages:

- `openintro` for the `mammals` data set
- `MASS` for the `stepAIC` function
- `abe` for augmented backwards selection
- `MuMin` for model averaging based on AIC weights
- `glmnet` for an implementation of the LASSO algorithm
- `caret` for performing cross-validation
- `rms` for evaluating model evaluation using the bootstrap

8.2 Goals of multivariable regression modeling

Before deciding on an appropriate modeling strategy, it is important to recognize that there are multiple possible objectives motivating the need for a model (Shmueli 2010; Kuiper and Sklar 2012; Tredennick et al. 2021). Kuiper and Sklar (2012) lists 3 general modeling objectives, to *describe*, *predict*, or *explain*, which closely mimics the goals outlined by Tredennick et al. (2021), *exploration*, *prediction*, and *inference*.

- **Describe**: we may just want to capture the main patterns in the data in a parsimonious way. This may be a first step in trying to understand which factors are *related* to the response variable. Note, describing associations between variables is a much less lofty goal than evaluating cause and effect.
- **Predict**: we may want to use data we have in hand to make predictions about future data. Here, we may not need to worry about causality and confounding (capturing associations may be just fine). We may be able to predict the average life expectancy for a country based on tvs per capita (Figure 7.1) or the number of severe sunburns from local ice creme sales (Figure 7.3). Yet these predictions do not capture causal mechanisms, and thus, we will often find that our model fails to predict when we try to apply it to a new situation where the underlying correlations among our predictor variables differs from the correlations among predictors in the data set we used to fit the model.
- **Explain/Infer**: we may have a biological hypothesis or set of competing hypotheses about how the world works. These hypotheses might suggest that certain variables, or combination of variables, are causally related to the response variable. When trying to infer causal relationships between predictor and response variables, we have to think more broadly about links among our observed predictor variables (not just links between predictors and the response). In addition, we must consider possible confounding by unmeasured or omitted variables. Essentially, we need to keep in mind everything from Chapter 7.

Clearly, the least lofty goal is to explore or describe patterns, but whether one views prediction or inference as the most challenging or lofty goal may depend on whether predictions are expected to match observations in novel situations or just under conditions similar to those used to train the model. If we want to be able to predict what will happen when the system changes, we will need to have a strong understanding of mechanisms driving observed patterns.

It is important to consider your goals when deciding upon a modeling strategy (Table 8.1, recreated Table 2 from Tredennick et al. 2021). I collaborate mainly with scientists that work with observational data. Often, they express their questions in broad terms, such as, "I want to know which predictor variables are most important." When pressed, they may indicate they are interested a bit in all of the above (describing patterns, explaining patterns, and predicting outcomes). The main challenge to "doing all of the above" is that it is all too easy to detect patterns in data that do not represent causal mechanisms, leading to models that fail to predict new data well. This can result in a tremendous waste of money and resources. We need to consider the impact of applying an overly flexible modeling strategy if our goal is to identify important associations among variables (**explain**) or if our goal is to develop a predictive model (**predict**). I learned of the dangers of overfitting the hard way while getting my Masters degree in Biostatistics at the University of North Carolina-Chapel Hill (see Section 8.3).

If we have competing models for how the world works (e.g., different causal networks; Chapter 7), then we can formulate *mechanistic* models that represent these competing hypotheses. In some cases you may have to write your own code to fit models using Maximum Likelihood (Chapter 10) and Bayesian machinery (Chapter 11); we will be learning about these methods soon. We can then compare predictions from these models or evaluate their ability to match qualitative patterns in data ("goodness-of-fit"), ideally across multiple study systems, to determine if our theories hold up to the scrutiny of data. If you find yourself in this situation (i.e., doing **inference**), consider yourself lucky (or good); you are doing exciting science.

TABLE 8.1 Modeling strategies should be tailored to specific modeling objectives (to explain, predict, or infer). Table 2 recreated from Tredennick et al. (2021) CC BY-NC 4.0.

Parameter	Exploration	Inference	Prediction
Purpose	generate hypotheses	test hypotheses	forecast the future accurately
Priority	thoroughness	avoid false positives	minimize error
A priori hypothesis	not necessary	essential	not necessary, but may inform model specification
Emphasis on model selection	important	minimal	important
Key statistical tools	any	null hypothesis significance tests	AIC; regularization; machine learning; cross-validation; out-of-sample validation
Pitfalls	fooling yourself with overfitted models with spurious covariate effects	misrepresenting exploratory tests as tests of a priori hypotheses	failure to rigorously validate prediction accuracy with independent data

8.3 My experience

In the first year of my Masters degree, I took a Linear Models class from Dr. Keith Muller. For our midterm exam, we were given a real data set and asked to develop a model for patients' Serum Albumin levels[1]. The data set contained numerous predictors, some with missing data. We were given a weekend to complete the analysis and write up a report on our findings. Importantly, we were instructed to take one of two approaches:

1. If we knew the subject area (or, could quickly get up speed), then we could choose to test a set of *a priori* hypotheses regarding whether specific explanatory variables were associated with Serum Albumin levels. We were encouraged to consider potential issues related to multiple comparisons and to adjust our α level to ensure an overall type I error rate of less than 5%[2].
2. Split the data into a "training" data set (with 80% of the original observations) and a "test" data set (with the remaining 20% of the observations). Use the training data set to determine an appropriate model. Then, evaluate the model's performance using the test data set. When determining an appropriate model, we were instructed to use various residual and diagnostic plots and to also consider the impact that an overly data-driven or flexible modeling approach might have on future model performance.

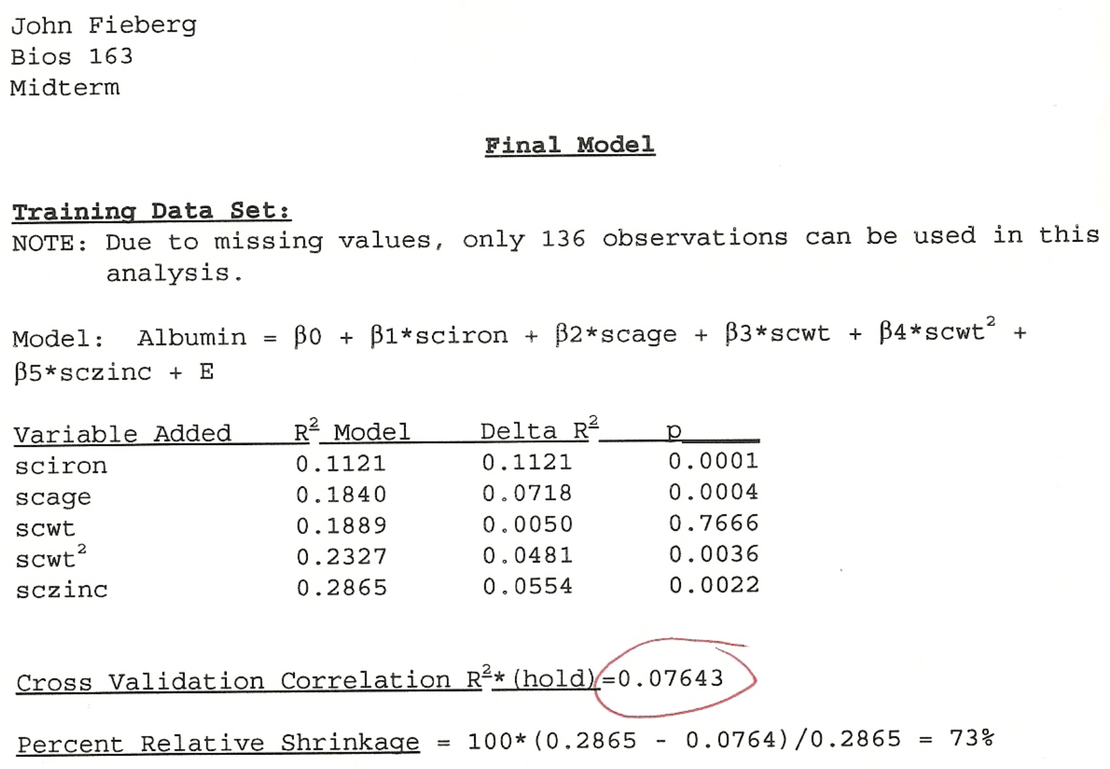
I don't think anyone from our class knew anything about blood chemistry or kidney failure, so I suspect everyone went with option 2. A few things I recall from my approach and the final outcome:

1. I grouped variables into similar categories (e.g., socio-economic status, stature [height, weight, etc], dietary intake variables) and examined variables within each group for collinearity. I then picked a single "best" variable in each category using simple linear regressions (i.e., fitting models with a single predictor to determine which variable within each group was most highly correlated with the response).
2. I fit a model that combined the "best" variables from each category and evaluated model assumptions (Normality, constant variance, linearity). Residual plots all looked good - no major assumption violations.

[1]Albumin is sometimes used as an indicator of kidney disease.
[2]The overall type I error is the probability of rejecting one or more null hypotheses when all of them are true

3. I allowed for a non-linear effect of weight using a quadratic polynomial (see Section 4.3). To be honest, I cannot remember why. The decision, however, was likely arrived at after looking at the data and noticing a trend in the residuals or finding that a quadratic term was highly significant when included in the model.
4. I applied various stepwise regression methods (see Section 8.4.1) and *always arrived at the same reduced model.*
5. I looked to see if I might have left off any important variables by adding them to this reduced model arrived at in step [4]. None of the added variables were statistically significant. This step gave me further confidence that I had found the best model for the job.
6. Lastly, I again looked at residual diagnostics associated with the reduced model. Everything looked hunky-dory.

I remember being highly confident that I found *THE BEST* model and that the assumptions were all reasonable. I had such good feelings about how things went initially, that when reflecting back on the assignment many years later, I was sure that my model had an R^2 near 90%. When I began teaching, I went back and found my actual test. The initial R^2 was only 28%. Then came the moment of truth. I evaluated the model's performance using the test data. My model only explained 7.6% of the variability (Figure 8.1)! I had a hard time believing it. Many other students had similar experiences. I had learned an important lesson that would stick with me for the rest of my professional life. It is way to easy to detect a signal in sea of noise if you allow for too flexible of a modeling approach.



John Fieberg
Bios 163
Midterm

Final Model

Training Data Set:
NOTE: Due to missing values, only 136 observations can be used in this analysis.

Model: Albumin = $\beta 0 + \beta 1{*}\text{sciron} + \beta 2{*}\text{scage} + \beta 3{*}\text{scwt} + \beta 4{*}\text{scwt}^2 + \beta 5{*}\text{sczinc} + E$

Variable Added	R^2 Model	Delta R^2	p
sciron	0.1121	0.1121	0.0001
scage	0.1840	0.0718	0.0004
scwt	0.1889	0.0050	0.7666
scwt^2	0.2327	0.0481	0.0036
sczinc	0.2865	0.0554	0.0022

Cross Validation Correlation R^2*(hold) = 0.07643

Percent Relative Shrinkage = 100*(0.2865 - 0.0764)/0.2865 = 73%

FIGURE 8.1 My mid-tem exam from my Linear Models class.

8.4 Stepwise selection algorithms

There are many ways to sift through model space to determine a set of predictors that are most highly associated with a response variable. At the extreme, one can specify a set of predictors and then fit "all possible models" that include 1 or more of these predictors along with a null model containing only an intercept (e.g., using the `dredge` function in the `MuMIn` package; Barton 2020). Having fit all possible models, one must specify a criterion for choosing a "best model." For linear models, one could use to use the adjusted-R^2 (Section 3.4). Alternatively, one could choose to use one of a number of various information criterion (e.g., AIC, BIC, WAIC, DIC). The Akaike information criterion, AIC (Akaike 1974), is probably the most well known and most frequently used information criterion:

$$AIC = -2\log l(\hat{\theta}) + 2p$$

where $\log l$ is the log-likelihood of the data evaluated at the estimated parameter values (Chapter 10), $\hat{\theta}$, and p is the number of parameters in the model. The likelihood will always increase as we add more parameters, but AIC may not due to the penalty, $2p$. Simpler models with smaller values of AIC are preferred, all else being equal.

Alternatively, one can try to successfully build bigger and better models (by adding 1 variable at a time), referred to as forward-stepwise selection, or try to build slimmer and more parsimonious models (by eliminating 1 variable at a time from an initial full model), referred to as backwards elimination. Either approach will result in many comparisons of nested models; two models are nested if you can get from the more complex model to the simpler model by by setting one or more parameters equal to 0. As one example, the following two models are nested (you can get from the first model to the second by setting β_2 to 0):

- Sleep $= \beta_0 + \beta_1$Danger $+ \beta_2$LifeSpan
- Sleep $= \beta_0 + \beta_1$Danger

whereas the two models below are not:

- Sleep $= \beta_0 + \beta_1$Danger
- Sleep $= \beta_0 + \beta_1$Lifespan

For nested models, we can use p-values from t-tests, F-tests, or likelihood-ratio tests to compare models (in addition to AIC or adjusted-R^2). In the next section, we will briefly consider stepwise-selection algorithms using AIC.

8.4.1 Backwards elimination and forward-stepwise selection

To apply backwards stepwise selection, we:

1. Fit a full model containing all predictors of interest.
2. Consider all possible models formed by dropping 1 of these predictors
3. Keep the current model, or drop the "worst" predictor depending on:
   - p-values from the individual t-tests (drop the variable with the highest p-value if it is greater than some threshold value, not necessarily 0.05)
   - Adjusted R^2 values (higher values are better)

   - AIC (lower values are better)

4. Rinse and repeat until you can no longer improve the model by dropping a predictor

The `stepAIC` function in the `MASS` library will do this for us. Let's explore this approach using the mammal sleep data set from Chapter 6. Before we begin, it is important to recognize that this data set has many missing values for several of the variables. This can create issues when comparing models since R will by default drop any observations where one or more of the variables are missing. Thus, the number observations will change depending on which predictors are included in the model, making comparisons using AIC or adjusted R^2 problematic. Ideally, we would consider a multiple imputation strategy to deal with this issue, but for now, we will just consider the data set with complete observations.

```
library(openintro)
data(mammals, package="openintro")
mammalsc<-mammals %>% filter(complete.cases(.))
MASS::stepAIC(lm(total_sleep ~ life_span + gestation + log(brain_wt) +
                     log(body_wt) + predation + exposure + danger, data=mammalsc))
```

```
## Start:  AIC=100.44
## total_sleep ~ life_span + gestation + log(brain_wt) + log(body_wt) +
##     predation + exposure + danger
## 
##                 Df Sum of Sq    RSS     AIC
## - log(body_wt)   1     0.412 313.99  98.491
## - life_span      1     1.218 314.79  98.598
## - log(brain_wt)  1    12.516 326.09 100.079
## - gestation      1    13.389 326.96 100.192
## <none>                       313.58 100.436
## - exposure       1    18.527 332.10 100.847
## - predation      1    31.079 344.65 102.405
## - danger         1    92.636 406.21 109.307
## 
## Step:  AIC=98.49
## total_sleep ~ life_span + gestation + log(brain_wt) + predation +
##     exposure + danger
## 
##                 Df Sum of Sq    RSS     AIC
## - life_span      1     0.977 314.96  96.621
## - gestation      1    13.113 327.10  98.209
## <none>                       313.99  98.491
## - exposure       1    19.446 333.43  99.014
## - predation      1    30.876 344.86 100.430
## - log(brain_wt)  1    31.497 345.48 100.506
## - danger         1    92.380 406.37 107.323
## 
## Step:  AIC=96.62
## total_sleep ~ gestation + log(brain_wt) + predation + exposure +
##     danger
## 
##                 Df Sum of Sq    RSS     AIC
## - gestation      1    12.206 327.17  96.218
## <none>                        314.96  96.621
## - exposure       1    18.987 333.95  97.080
## - predation      1    30.083 345.05  98.453
## - log(brain_wt)  1    34.465 349.43  98.983
## - danger         1    92.153 407.12 105.400
##
## Step:  AIC=96.22
## total_sleep ~ log(brain_wt) + predation + exposure + danger
##
##                 Df Sum of Sq    RSS     AIC
## <none>                        327.17  96.218
## - exposure       1    16.239 343.41  96.253
## - predation      1    41.155 368.32  99.194
## - log(brain_wt)  1    90.786 417.96 104.504
## - danger         1   108.584 435.75 106.255
```

```
##
## Call:
## lm(formula = total_sleep ~ log(brain_wt) + predation + exposure +
##     danger, data = mammalsc)
##
## Coefficients:
##   (Intercept)  log(brain_wt)      predation       exposure         danger
##       16.5878        -0.8800         2.2321         0.9066        -4.5425
```

We see a set of tables comparing the current "best" model (`<none>` indicating no variables have been dropped) to all possible models that have 1 fewer predictor, sorted from lowest (best fitting) to highest (worst fitting) as judged by AIC. For example, in the first table, we see that dropping any of (`log(body_wt)`, `life_span`, `log(brain_wt)`, `gestation`) will result in a model with a lower AIC than the full model containing all variables. On the other hand, dropping any of (`exposure`, `predation`, or `danger`) will result in a model with a worse AIC than the one associated with the full model. At this step, we drop `log(body_wt)` since dropping it results in a model with the lowest AIC (98.491) among all 6-variable models. We repeat the algorithm, dropping `lifespan`, and then once more, dropping `gestation`. At that point, dropping any of (`log(brain_wt)`, `predation`, `exposure`, `danger`) will result in a model that has a higher AIC, so we stop.

Forward stepwise selection moves in the opposite direction and can be implemented using the `step` function:

```
min.model <- lm(total_sleep ~ 1, data=mammalsc)
step(min.model, scope=( ~ life_span + gestation + log(brain_wt) + log(body_wt) +
                          predation + exposure + danger),
                          direction="forward", data=mammalsc)
```

```
## Start:  AIC=131.15
## total_sleep ~ 1
##
##                 Df Sum of Sq    RSS    AIC
## + log(brain_wt)  1    352.15 557.17 112.58
## + exposure       1    351.08 558.25 112.66
## + log(body_wt)   1    346.71 562.62 112.99
## + gestation      1    343.34 565.98 113.24
## + danger         1    332.07 577.25 114.07
## + predation      1    148.94 760.38 125.64
## + life_span      1    133.00 776.32 126.51
## <none>                      909.32 131.15
##
## Step:  AIC=112.58
## total_sleep ~ log(brain_wt)
##
##                Df Sum of Sq    RSS     AIC
## + danger        1   179.991 377.18  98.192
## + predation     1   118.745 438.43 104.512
## + exposure      1    83.765 473.41 107.736
## + gestation     1    38.295 518.88 111.588
## <none>                      557.17 112.579
## + life_span     1    11.296 545.88 113.718
## + log(body_wt)  1     6.190 550.98 114.109
##
## Step:  AIC=98.19
## total_sleep ~ log(brain_wt) + danger
##
##                Df Sum of Sq    RSS     AIC
## + predation     1    33.773 343.41  96.253
## + gestation     1    19.139 358.04  98.005
## <none>                      377.18  98.192
## + exposure      1     8.857 368.32  99.194
## + life_span     1     0.519 376.66 100.135
## + log(body_wt)  1     0.354 376.83 100.153
##
## Step:  AIC=96.25
## total_sleep ~ log(brain_wt) + danger + predation
##
##                Df Sum of Sq    RSS    AIC
## + exposure      1   16.2392 327.17 96.218
## <none>                      343.41 96.253
## + gestation     1    9.4578 333.95 97.080
## + log(body_wt)  1    0.6927 342.72 98.168
## + life_span     1    0.0103 343.40 98.251
##
## Step:  AIC=96.22
## total_sleep ~ log(brain_wt) + danger + predation + exposure
##
##                Df Sum of Sq    RSS    AIC
## <none>                      327.17 96.218
## + gestation     1   12.2056 314.96 96.621
## + log(body_wt)  1    0.0943 327.08 98.206
## + life_span     1    0.0696 327.10 98.209

##
## Call:
## lm(formula = total_sleep ~ log(brain_wt) + danger + predation +
##     exposure, data = mammalsc)
##
## Coefficients:
```

```
##    (Intercept)  log(brain_wt)          danger       predation        exposure
##        16.5878         -0.8800         -4.5425          2.2321          0.9066
```

In this case, we end up at the same place. This may seem reassuring as it tells us that we arrive at the same "best model" regardless of which approach we choose. Yet, what really matters is how these algorithms perform across multiple data sets (see Section 8.8.2). Therein lies many problems. As noted by Frank Harrell in his Regression Modeling Strategies book (Harrell Jr 2015), with stepwise selection:

1. R^2 values are biased high.
2. The ordinary F and χ^2 test statistics used to test hypotheses do not have the claimed distribution.
3. SEs of regression coefficients will be biased low and confidence intervals will be too narrow.
4. p-values will be too small and do not have the proper meaning (due to multiple comparison issues).
5. Regression coefficients will be biased high in absolute magnitude.
6. Rather than solve issues with collinearity, collinearity makes variable selection arbitrary.
7. Stepwise selection does not require that we think hard about our underlying problem/study system.

Issues with p-values are rather easy to understand in the context of multiple comparisons and multiple hypothesis tests, whereas the bias in regression coefficients may be more surprising to some readers. To understand why regression coefficient estimators will be biased high, it is important to recognize that inclusion in the final model depends not on the *true* relationship between the explanatory and response variables but rather on the *estimated* relationship determined from the original sample. A variable is more likely to be included if by chance its importance in the original sample was overestimated than if it was underestimated (Copas and Long 1991). These issues are more pronounced when applied to small data sets with lots of predictors; stepwise selection methods can easily select noise variables rather than ones that are truly important.

Issues with stepwise selection are well known to statisticians, who agree that these methods should be abandoned (Anderson and Burnham 2004; Whittingham et al. 2006; Hegyi and Garamszegi 2011; Giudice, Fieberg, and Lenarz 2012; Fieberg and Johnson 2015). Yet, they are still routinely used, likely because they appear to be objective and "let the data speak" rather than forcing the analyst to think. Stepwise selection methods can sometimes be useful for identifying important associations, but any results should be treated cautiously. I.e., it is best to consider these methods as useful for generating new hypotheses; these hypotheses should be tested with new, independent data.

If you are not convinced, try this simple simulation example posted by Florian Hartig here, that considers a response variable that is unrelated to 100 different simulated predictor variables. On average, we would expect to see $100 \times 0.05 = 5$ significant tests since the Null hypothesis is true for all 100 tests (the response is not influenced by any of the predictors).

```
set.seed(1)
library(MASS)

# Generate 200 observations with 100 predictors that are unrelated to the response variable
dat <- data.frame(matrix(runif(20000), ncol = 100))
dat$y <- rnorm(200)
```

And, if we fit a full model that contains all predictors and look at the p-values for the individual hypothesis tests, we find only 2 that have p-values less than 0.05

```
# Fit a full model containing all predictors and test for significance.
fullModel <- lm(y ~ . , data = dat)
summary(fullModel)
```

```
##
## Call:
## lm(formula = y ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.95280 -0.39983 -0.01572  0.46104  1.61967
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.19356    1.44518   1.518   0.1322
## X1           0.58079    0.32689   1.777   0.0787 .
## X2          -0.52687    0.32701  -1.611   0.1103
## X3           0.27721    0.33117   0.837   0.4046
## X4          -0.18342    0.30443  -0.602   0.5482
## X5          -0.18544    0.29011  -0.639   0.5242
## X6          -0.18382    0.31406  -0.585   0.5597
## X7          -0.46290    0.28349  -1.633   0.1057
## X8          -0.21527    0.29856  -0.721   0.4726
## X9           0.12216    0.30359   0.402   0.6883
## X10         -0.02594    0.33828  -0.077   0.9390
## X11          0.25669    0.29482   0.871   0.3860
## X12         -0.10183    0.30164  -0.338   0.7364
## X13          0.49507    0.33438   1.481   0.1419
## X14          0.16642    0.33659   0.494   0.6221
## X15          0.11402    0.32964   0.346   0.7302
## X16         -0.17640    0.31619  -0.558   0.5782
## X17         -0.03129    0.31830  -0.098   0.9219
## X18         -0.28201    0.29681  -0.950   0.3444
## X19          0.02209    0.29664   0.074   0.9408
## X20          0.25063    0.29855   0.839   0.4032
## X21         -0.02479    0.30556  -0.081   0.9355
## X22         -0.01187    0.31265  -0.038   0.9698
## X23         -0.58731    0.31491  -1.865   0.0651 .
## X24         -0.27343    0.32894  -0.831   0.4078
## X25         -0.22745    0.29223  -0.778   0.4382
## X26          0.18606    0.35755   0.520   0.6040
## X27         -0.26998    0.33302  -0.811   0.4195
## X28          0.09683    0.32235   0.300   0.7645
## X29          0.36746    0.32915   1.116   0.2670
## X30         -0.26027    0.31335  -0.831   0.4082
## X31         -0.07890    0.28822  -0.274   0.7849
## X32         -0.07879    0.32662  -0.241   0.8099
## X33         -0.27736    0.34542  -0.803   0.4239
## X34         -0.21118    0.34514  -0.612   0.5420
## X35          0.17595    0.30706   0.573   0.5679
## X36          0.17084    0.30423   0.562   0.5757
## X37          0.28246    0.29520   0.957   0.3410
## X38          0.01765    0.32873   0.054   0.9573
## X39          0.07598    0.27484   0.276   0.7828
## X40          0.09714    0.34733   0.280   0.7803
## X41         -0.16985    0.31608  -0.537   0.5922
## X42        -0.25184    0.33203  -0.758   0.4500
## X43        -0.08306    0.29306  -0.283   0.7774
## X44        -0.17389    0.31090  -0.559   0.5772
## X45        -0.30756    0.30995  -0.992   0.3235
## X46         0.61520    0.30961   1.987   0.0497 *
## X47        -0.61994    0.32461  -1.910   0.0591 .
## X48         0.62326    0.33822   1.843   0.0684 .
## X49         0.35504    0.30382   1.169   0.2454
## X50         0.09683    0.31925   0.303   0.7623
## X51         0.17292    0.30770   0.562   0.5754
## X52        -0.06560    0.30549  -0.215   0.8304
## X53        -0.29953    0.32318  -0.927   0.3563
## X54         0.06888    0.32289   0.213   0.8315
## X55         0.05695    0.32103   0.177   0.8596
## X56         0.26284    0.32914   0.799   0.4265
## X57         0.10457    0.29788   0.351   0.7263
## X58        -0.19239    0.30729  -0.626   0.5327
## X59         0.02371    0.29171   0.081   0.9354
## X60        -0.12842    0.32321  -0.397   0.6920
## X61         0.06931    0.30015   0.231   0.8179
## X62        -0.27227    0.31918  -0.853   0.3957
## X63        -0.17359    0.32287  -0.538   0.5920
## X64        -0.41846    0.33808  -1.238   0.2187
## X65        -0.37243    0.31872  -1.169   0.2454
## X66         0.36263    0.33034   1.098   0.2750
## X67        -0.10120    0.30663  -0.330   0.7421
## X68        -0.33790    0.33633  -1.005   0.3175
## X69        -0.05326    0.30171  -0.177   0.8602
## X70        -0.01047    0.33111  -0.032   0.9748
## X71        -0.46896    0.32387  -1.448   0.1508
## X72        -0.29867    0.33543  -0.890   0.3754
## X73        -0.32556    0.33183  -0.981   0.3289
## X74         0.21187    0.31690   0.669   0.5053
## X75         0.63659    0.31144   2.044   0.0436 *
## X76         0.13838    0.31642   0.437   0.6628
## X77        -0.18846    0.29382  -0.641   0.5227
## X78         0.06325    0.29180   0.217   0.8289
## X79         0.07256    0.30145   0.241   0.8103
## X80         0.33483    0.34426   0.973   0.3331
## X81        -0.33944    0.35373  -0.960   0.3396
## X82        -0.01291    0.32483  -0.040   0.9684
## X83        -0.06540    0.27637  -0.237   0.8134
## X84         0.11543    0.32813   0.352   0.7257
## X85        -0.20415    0.31476  -0.649   0.5181
## X86         0.04202    0.33588   0.125   0.9007
## X87        -0.33265    0.29159  -1.141   0.2567
## X88        -0.49522    0.31251  -1.585   0.1162
## X89        -0.39293    0.33358  -1.178   0.2417
## X90        -0.34512    0.31892  -1.082   0.2818
## X91         0.10540    0.28191   0.374   0.7093
## X92        -0.08630    0.30297  -0.285   0.7764
## X93         0.02402    0.32907   0.073   0.9420
## X94          0.51255    0.32139   1.595   0.1139
## X95         -0.19971    0.30634  -0.652   0.5160
## X96         -0.09592    0.34585  -0.277   0.7821
## X97         -0.18862    0.29266  -0.644   0.5207
## X98          0.14997    0.34858   0.430   0.6680
## X99         -0.08061    0.30400  -0.265   0.7914
## X100        -0.34988    0.31664  -1.105   0.2718
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9059 on 99 degrees of freedom
## Multiple R-squared:  0.4387, Adjusted R-squared:  -0.1282
## F-statistic: 0.7739 on 100 and 99 DF,  p-value: 0.8987
```

But, if we first perform stepwise selection, we end up with 15 out of 28 predictors that have p-values < 0.05.

```
# Perform model selection using AIC and then summarize the results
selection <- stepAIC(fullModel, trace=0)
summary(selection)
```

```
##
## Call:
## lm(formula = y ~ X1 + X2 + X3 + X5 + X7 + X13 + X20 + X23 + X30 +
##     X37 + X42 + X45 + X46 + X47 + X48 + X64 + X65 + X66 + X71 +
##     X75 + X80 + X81 + X87 + X88 + X89 + X90 + X94 + X100, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.04660 -0.50885  0.05722  0.49612  1.53704
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.0314     0.5045   2.044  0.04244 *
## X1            0.4728     0.2185   2.164  0.03187 *
## X2           -0.3809     0.2012  -1.893  0.06008 .
## X3            0.3954     0.1973   2.004  0.04668 *
## X5           -0.2742     0.1861  -1.473  0.14251
## X7           -0.4442     0.1945  -2.284  0.02359 *
## X13           0.4396     0.1980   2.220  0.02775 *
## X20           0.3984     0.1918   2.078  0.03924 *
## X23          -0.4137     0.2081  -1.988  0.04836 *
## X30          -0.3750     0.1991  -1.884  0.06125 .
## X37           0.4006     0.1989   2.015  0.04550 *
## X42          -0.3934     0.2021  -1.946  0.05325 .
## X45          -0.3197     0.2063  -1.550  0.12296
## X46           0.3673     0.1992   1.844  0.06690 .
## X47          -0.4240     0.2029  -2.090  0.03811 *
## X48           0.5130     0.1937   2.649  0.00884 **
## X64          -0.3676     0.2094  -1.755  0.08102 .
## X65          -0.2887     0.1975  -1.462  0.14561
## X66           0.2769     0.2107   1.315  0.19039
## X71          -0.5301     0.2003  -2.646  0.00891 **
## X75           0.5020     0.1969   2.550  0.01165 *
## X80           0.3722     0.2058   1.809  0.07224 .
## X81          -0.3731     0.2176  -1.715  0.08820 .
## X87          -0.2684     0.1958  -1.371  0.17225
## X88          -0.4524     0.2069  -2.187  0.03011 *
## X89          -0.4123     0.2060  -2.002  0.04691 *
## X90          -0.3528     0.2067  -1.707  0.08971 .
## X94           0.3813     0.2049   1.861  0.06440 .
## X100         -0.4058     0.2024  -2.005  0.04653 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.76 on 171 degrees of freedom
## Multiple R-squared:  0.3177, Adjusted R-squared:  0.2059
## F-statistic: 2.843 on 28 and 171 DF,  p-value: 1.799e-05
```

8.4.2 Augmented backward elimination

Heinze, Wallisch, and Dunkler (2018) provide a nice overview of methods for selecting explanatory variables, including stepwise approaches (Section 8.4.1), as well as other modeling strategies considered in this Chapter, including full model inference informed by an effective sample size (Section 8.5), AIC-based model weights (Section 8.6), and penalized likelihood methods (Section 8.7). They also highlight an *augmented backwards elimination* algorithm that adds a "change-in-estimate criterion" to the normal backwards selection algorithm covered in the last section. Variables that, when dropped, result in large changes in other coefficients are retained even if dropping them results in a lower AIC. This strategy is meant to guard against bias incurred by dropping important confounding variables while still allowing one to drop variables that do not aid in predicting the response as long as they do not influence other coefficients in the model.

This method is available in the `abe` package (Rok Blagus 2017). One can choose to force inclusion of variables that are of particular importance (e.g., treatment variables or known confounders), referred to as "passive" variables (using the `include` argument). Other variables will be "active" and considered for potential exclusion using significance tests or comparisons using AIC with the added condition that dropping them from the model does not result in a large change to coefficients association with passive variables (where "large" is controlled by a threshold parametervia the argument `tau`). We demonstrate the approach with the sleep data set, below:

```
library(abe)
fullmodel <- lm(total_sleep ~ life_span + gestation + log(brain_wt) +
                     log(body_wt) + predation + exposure + danger, data=mammalsc,
                     x = TRUE, y = TRUE)
abe(fullmodel, criterion="AIC", verbose = TRUE, exp.beta = FALSE, data=mammalsc)
```

```
##
##
## Model under investigation:
## lm(formula = total_sleep ~ life_span + gestation + log(brain_wt) +
##     log(body_wt) + predation + exposure + danger, data = mammalsc,
##     x = TRUE, y = TRUE)
## Criterion for non-passive variables: life_span : 98.5984 , gestation : 100.1916 , log(brain_wt) : 100.0
##    black list:  log(body_wt) : -1.9449, life_span : -1.8371, log(brain_wt) : -0.3562, gestation : -0.24
##            Investigating change in b or exp(b) due to omitting variable  log(body_wt)  ;  life_span : 0
##    updated black list: life_span : -1.8371, log(brain_wt) : -0.3562, gestation : -0.244
##            Investigating change in b or exp(b) due to omitting variable  life_span  ;  gestation : 0.01
##    updated black list: log(brain_wt) : -0.3562, gestation : -0.244
##            Investigating change in b or exp(b) due to omitting variable  log(brain_wt)  ;  life_span : 
##    updated black list: gestation : -0.244
##            Investigating change in b or exp(b) due to omitting variable  gestation  ;  life_span : 0.04
##
##
## Final model:
## lm(formula = total_sleep ~ life_span + gestation + log(brain_wt) +
##     log(body_wt) + predation + exposure + danger, data = mammalsc,
##     x = TRUE, y = TRUE)
```

```
##
## Call:
## lm(formula = total_sleep ~ life_span + gestation + log(brain_wt) +
##     log(body_wt) + predation + exposure + danger, data = mammalsc,
##     x = TRUE, y = TRUE)
##
## Coefficients:
##   (Intercept)      life_span      gestation  log(brain_wt)   log(body_wt)      predation       exposure
##      16.91558        0.01393       -0.00767       -0.85003        0.11271        2.00018        0.98176
```

In the output, above, variables that, when dropped, lead to increases in AIC are "blacklisted" and therefore not considered for potential exclusion (the same would be true for variables listed as passive using the `include` argument). The algorithm then looks to see whether dropping any of the variables that improve AIC result in significant changes to other non-passive variables. In this case, we see that we do not eliminate any of the explanatory variables as doing so results in large changes in one or more of the passive variables. We could relax the change-in-estimate criterion by specifying a larger value of `tau` than the default (0.05). If we use a value of 0.1, then we end up dropping `log(body_wt)` and `life_span`:

```
abe(fullmodel, criterion="AIC", verbose = TRUE, exp.beta = FALSE, tau= 0.1, data=mammalsc)
```

```
##
##
## Model under investigation:
## lm(formula = total_sleep ~ life_span + gestation + log(brain_wt) +
##     log(body_wt) + predation + exposure + danger, data = mammalsc,
##     x = TRUE, y = TRUE)
## Criterion for non-passive variables: life_span : 98.5984 , gestation : 100.1916 , log(brain_wt) : 100.0
##    black list:  log(body_wt) : -1.9449, life_span : -1.8371, log(brain_wt) : -0.3562, gestation : -0.24
##            Investigating change in b or exp(b) due to omitting variable  log(body_wt)  ;  life_span : 0
##
##
## Model under investigation:
## lm(formula = total_sleep ~ life_span + gestation + log(brain_wt) +
##     predation + exposure + danger, data = mammalsc, x = TRUE,
##     y = TRUE)
## Criterion for non-passive variables: life_span : 96.6211 , gestation : 98.2091 , log(brain_wt) : 100.50
##    black list:  life_span : -1.8695, gestation : -0.2816
##           Investigating change in b or exp(b) due to omitting variable  life_span  ;  gestation : 0.01
##
##
## Model under investigation:
## lm(formula = total_sleep ~ gestation + log(brain_wt) + predation +
##     exposure + danger, data = mammalsc, x = TRUE, y = TRUE)
## Criterion for non-passive variables: gestation : 96.218 , log(brain_wt) : 98.9825 , predation : 98.4525
##    black list:  gestation : -0.4031
##           Investigating change in b or exp(b) due to omitting variable  gestation  ;  log(brain_wt) : (
##
##
## Final model:
## lm(formula = total_sleep ~ gestation + log(brain_wt) + predation +
##     exposure + danger, data = mammalsc, x = TRUE, y = TRUE)


##
## Call:
## lm(formula = total_sleep ~ gestation + log(brain_wt) + predation +
##     exposure + danger, data = mammalsc, x = TRUE, y = TRUE)
##
## Coefficients:
##   (Intercept)      gestation  log(brain_wt)      predation       exposure         danger
##     16.759945      -0.007153      -0.658072       1.956738       0.985174      -4.257452
```

8.5 Degrees of freedom (df) spending: One model to rule them all

Given the potential for overfitting and issues with multiple testing and inference when using model selection algorithms, it can be advantageous at times to just fit a single model and use it for inference (Babyak 2004; Whittingham et al. 2006; Giudice, Fieberg, and Lenarz 2012; Harrell Jr 2015; Fieberg and Johnson 2015). In fact, this is the approach I suggest for those that feel they want to do a bit of everything (describe, predict, and infer). We can use the fitted coefficients and their signs to describe associations between our explanatory variables and the response variable. Because the model is pre-specified, confidence intervals and p-values would have their correct interpretation as long as the assumptions are not violated. In other words, rather than use model selection to determine whether associations are "significant" or not, we can judge significance using estimates of effect size (i.e., regression coefficients) and their uncertainty. If our goal is prediction, then we may lose out on some precision gains that could result from dropping unimportant predictors. However, this gain is usually minimal and probably not worth the cost (e.g., biased regression coefficients, p-values that are too small, etc).

Yet, this approach is not without its challenges, particularly when there are a large number of potential explanatory variables to choose from. Fitting a single model with too many predictors can be problematic for reasons discussed previously (e.g., collinearity, potential for overfitting). Thus, an important first step is usually to whittle down the number of explanatory variables to consider. During this step, one should also consider whether some variables are likely to have a non-linear relationship with the response variable (Chapter 4). The number of explanatory variables and the degree of flexibility allowed for modeling associations should ideally be informed by the effective sample size available for estimating model parameters, which will depend on the type of response variable (Table 8.2 recreated from Harrell Jr (2015)). For continuous response variables with independent observations, the effective sample size is simply the number of observations. For

TABLE 8.2 Limiting sample sizes for various response variables. Recreated from Harrell Jr (2015).

Response	Effective Degrees of Freedom
Continuous	n (total sample size)
Binary	$\min(n_0, n_1)$
Ordinal (k categories)	$n - \frac{1}{n^2}(\sum_{i=1}^{k} n_i^3)$
Failure (survival time)	Number of failures

binary data (0's and 1's), the minimum of the 0's and 1's is a better measure of effective sample size; we have no ability to discover important associations when all responses are either 0 or 1. General guidelines have been proposed in the literature based on theory and simulation studies that one should limit the number of model degrees of freedom (think "number of parameters") to $\leq n/10$ or $\leq n/20$, where n is the effective sample size; said another way, we should have 10-20 *events per variable* (EPV) considered.

When determining which variables to include, it is important that we do **not** consider the strength of the relationship between the explanatory variables and the response variable since this data-driven approach will potentially lead to overfitting and negate the benefits of avoiding model selection. Instead, we should consider subject matter knowledge (i.e., include variables likely to be important based on prior work), cost/feasibility of data collection, relevance to your research questions and potential to be a confounding variable. In addition, we might rule out variables that have a lot of missing data, that are highly correlated with one or more other variables, or that vary little. In some cases, we might consider using principle components or other indexing methods to reduce the number of predictors (Sections 6.9, 6.10). Lastly, Dormann et al. (2013) suggest:

> In any regression-style model, the results will be most informative if predictors that are directly relevant to the response are used, i.e. proximal predictors are strongly preferable over distal ones (Austin 1980, 2002). This general concept leads to careful consideration of candidate predictor sets in the light of ecological knowledge, rather than amassing whatever data can be found and challenging the model to make sense of it." and… "As a general rule of thumb, a good strategy is to select variables that a) are ecologically relevant, b) are feasible to collect data on and c) are closer to the mechanism (in the sequence resource-direct-indirect-proxy variables: Harrell 2001, Austin 2002). Then, if the statistical method suggests deleting an ecologically reasonable or important variable, prominence should be given to ecology.

To summarize this approach to modeling:

- Limit model df (number of parameters) to $\leq n/10$ or $\leq n/20$, where n is your effective sample size.
- Fit a "full model" without further simplification.
- Determine important associations between explanatory and response variables using measures of effect size and their uncertainty.

In other words, determine how many 'degrees of freedom' you can spend, spend them, and then don't look back. If you feel you must look at other predictors or models, do this as part of a secondary "exploratory analysis" or "sensitivity analysis", and treat any discoveries during this secondary phase with caution. If you find yourself modeling with fewer events per variable than recommended (e.g., EPV < 10-20), then you should consider evaluating model stability using a bootstrap (see Section 8.8.2) or consider using some form of model-averaging (Section 8.6) or shrinkage estimator like the LASSO (Section 8.7) (Heinze, Wallisch, and Dunkler 2018).

8.6 AIC and model-averaging

We have just considered two modeling strategies at the opposite ends of the spectrum, stepwise-selection algorithms using AIC and inference based on a single model. One of the main downsides that we highlighted with using a single model was that prediction errors may be larger than necessary due to including predictors that just add noise. We also remarked that there is usually little cost to including predictors that explain little variation in the response. The Akaike information criterion (AIC) (Akaike 1974) is often used to choose among competing models, not just nested models when applying a stepwise-selection algorithm. The AIC was derived to measure the quality of future predictions, and thus, it is not surprising that AIC tends to select bigger models than a selection algorithm that uses null hypothesis tests. For example, if we use AIC to distinguish between two nested models that differ by 1 parameter, this is equivalent to choosing the larger model whenever a likelihood ratio test has a p-value < 0.157 (Anderson and Burnham 2004)[3].

If we are truly interested in prediction, there are alternatives we might want consider, including model-averaging and various "regularization techniques" that effectively "shrink" estimates associated with weakly-informative parameters towards 0 (Dahlgren 2010; Hooten and Hobbs 2015; Lever, Krzywinski, and Altman 2016). Alternatively, there are a number of non-parametric approaches (e.g., boosted regression trees, random forests, etc) that tend to perform extremely well when it comes to prediction (Cutler et al. 2007; Elith, Leathwick, and Hastie 2008; Lucas 2020). The main downside of these latter approaches is that they can lack clearly interpretable parameters that describe relationships between explanatory and response variables.

Here, we will briefly consider model-averaging using AIC-based model weights. This approach become incredibly popular among wildlife ecologists in the early 2000's following the publication of Anderson and Burnham (2004) (and other papers and an earlier edition that preceded it). In fact, there was a time when it was really difficult to get anything published in the *Journal of Wildlife Management* if you choose to use a different strategy for analyzing your data. During the past 10 years or so, there have been many critiques of how AIC is often used by wildlife ecologists (Arnold 2010; Murtaugh 2014; Cade 2015; Brewer, Butler, and Cooksley 2016), and AIC-based model averaging is no longer so prominent.

Rather than choose a *best* model, we can choose to average predictions among multiple "good" models. Buckland, Burnham, and Augustin (1997) and Anderson and Burnham (2004) suggested weighting models using AIC. Specifically, they outlined the following approach:

1. Write down K biologically plausible models.
2. Fit these models and calculate AIC for each (possibly with a "small sample correction" which they refer to as $AIC_c = AIC + \frac{2p(p+1)}{n-p-1}$, where n is the sample size and p is again the number of parameters in the model).
3. Compute model weights, using the AIC values, reflecting "relative plausibility" of the different models:

$$w_i = \frac{\exp(-\Delta AIC_i)}{\sum_{k=1}^{K} \exp(-\Delta AIC_k)}.$$

[3]The likelihood ratio test statistic for two models that differ by a single parameter = -2(logL_1 - logL_2) = AIC_1-AIC_2 + 2. If the AIC's are the same, then our likelihood ratio test statistic = 2, which we compare to a χ^2 distribution with 1 degree of freedom to get the p-value. We can reproduce the p-value of 0.157 in R using `1-pchisq(2, df = 1)`.

where $\Delta AIC_i = min_k(AIC_k) - AIC_i$ (difference in AIC between the "best" model and model i)

4. Calculate a model-averaged parameter using the weights and parameter estimates from the K fitted models:

$$\hat{\theta}_{avg} = \sum_{k=1}^{K} w_k \hat{\theta}_k$$

5. Calculate a standard error (SE) that accounts for model and sampling uncertainty:

$$\widehat{SE}_{avg} = \sum_{k=1}^{K} w_k \sqrt{SE^2(\hat{\theta}_k) + (\hat{\theta}_k - \hat{\theta}_{avg})^2}$$

Typically, one would also form 95% CIs using $\hat{\theta}_{avg} \pm 1.96\widehat{SE}_{avg}$, assuming the sampling distribution of $\hat{\theta}_{avg}$ is Normal.

We can implement this approach using the `MuMin` package (Barton 2020) as demonstrated below with the sleep data set.

```r
library(MuMIn)
```

We begin by fitting all possible models including one or more candidate predictors from our full model fit to the sleep data using the `dredge` function.

```r
options(na.action = "na.fail")
fullmodel<-lm(total_sleep ~ life_span + gestation + log(brain_wt) +
                     log(body_wt) + predation + exposure + danger, data=mammalsc)
allsubsets <- dredge(fullmodel)
```

```
## Fixed term is "(Intercept)"
```

```r
allsubsets
```

```
## Global model call: lm(formula = total_sleep ~ life_span + gestation + log(brain_wt) +
##     log(body_wt) + predation + exposure + danger, data = mammalsc)
## ---
## Model selection table
##     (Int)    dng      exp       gst   lif_spn log(bdy_wt) log(brn_wt)      prd df   logLik  AICc delta w
## 98   16.40 -3.628                                              -0.6763  1.99200  5 -103.722 219.1  0.00
## 100 16.59 -4.543  0.90660                                      -0.8800  2.23200  6 -102.704 219.8  0.70
## 70   16.09 -3.742         -0.012360                                      2.11000  5 -104.384 220.4  1.33
## 34   17.44 -1.575                                              -0.9194           4 -105.692 220.5  1.35
## 102 16.54 -3.309         -0.006265                             -0.4665  1.73300  6 -103.135 220.7  1.56
## 38   17.44 -1.504         -0.008654                             -0.5858           5 -104.598 220.9  1.75
## 82   14.78 -3.837                            -0.495600                   2.23200  5 -104.658 221.0  1.87
## 104 16.76 -4.257  0.98520 -0.007153                             -0.6581  1.95700  7 -101.906 221.1  2.00
## 86   15.60 -3.384         -0.008064          -0.285800                   1.83600  6 -103.642 221.7  2.57
## 114 16.80 -3.663                              0.141400      -0.8359  2.00400  6 -103.679 221.8  2.65
## 106 16.39 -3.629                     0.001225                   -0.6836  1.99700  6 -103.721 221.8  2.73
## 40  17.72 -2.090  0.80650 -0.009634                              -0.7553           6 -103.822 222.0  2.93
## 36  17.67 -2.061  0.65970                                        -1.0890           5 -105.193 222.1  2.94
## 6   17.10 -1.558          -0.017460                                               4 -106.513 222.1  3.00
## 22  16.28 -1.459          -0.010860              -0.380000                        5 -105.249 222.2  3.06
## 84  14.51 -4.615  0.77180                        -0.638400           2.45400  6 -103.959 222.3  3.21
## 72  16.10 -4.273  0.46450 -0.013960                                  2.28800  6 -104.087 222.6  3.46
## 88  15.40 -4.238  0.90140 -0.009041              -0.427100           2.04700  7 -102.650 222.6  3.48
## 116 16.73 -4.546  0.89750                         0.052610   -0.9373  2.23400  7 -102.698 222.7  3.58
## 108 16.55 -4.548  0.90900            0.003185                   -0.8995  2.24600  7 -102.700 222.7  3.58
## 78  16.37 -3.675          -0.010680 -0.017440                            1.99500  6 -104.225 222.9  3.74
## 42  17.53 -1.602                    -0.008602                   -0.8641           5 -105.663 223.0  3.88
## 50  17.72 -1.592                                  0.101000   -1.0340           5 -105.672 223.0  3.90
## 18  15.29 -1.514                                 -0.715000                        4 -107.059 223.2  4.09
## 90  15.26 -3.768                    -0.018450   -0.419600           2.10600  6 -104.486 223.4  4.26
## 118 17.00 -3.345          -0.006350               0.164100   -0.6489  1.74300  7 -103.077 223.4  4.34
## 110 16.44 -3.303          -0.006559  0.008853                   -0.5092  1.75700  7 -103.103 223.5  4.39
## 54  17.84 -1.526          -0.008738               0.139500   -0.7415           6 -104.559 223.5  4.41
## 46  17.41 -1.493          -0.008772  0.003183                   -0.6018           6 -104.594 223.6  4.48
## 24  16.18 -1.955  0.70660 -0.011880              -0.499300                        6 -104.678 223.8  4.65
## 14  17.48 -1.646          -0.014130 -0.029600                                     5 -106.078 223.8  4.71
## 74  16.20 -4.615                    -0.051960                            2.66400  5 -106.147 224.0  4.85
## 112 16.63 -4.263  0.99890 -0.007569  0.012180                   -0.7196  1.99300  8 -101.841 224.0  4.94
## 120 16.95 -4.262  0.97320 -0.007178               0.070800   -0.7344  1.95800  8 -101.895 224.2  5.04
## 94  15.71 -3.381          -0.007776 -0.005091   -0.272300           1.81500  7 -103.629 224.6  5.44
## 8   17.14 -1.676  0.15800 -0.018150                                               5 -106.480 224.6  5.52
## 122 16.78 -3.669                     0.003473    0.151600   -0.8681  2.01900  7 -103.674 224.6  5.53
## 44  17.74 -2.085  0.65740           -0.008068                   -1.0360           6 -105.167 224.7  5.62
## 30  16.52 -1.506          -0.010070 -0.012620   -0.344000                        6 -105.178 224.8  5.65
## 52  17.76 -2.062  0.65380                         0.032970   -1.1250           6 -105.190 224.8  5.67
## 92  15.03 -4.560  0.79210           -0.020270   -0.558700           2.32100  7 -103.745 224.8  5.67
## 26  16.09 -1.623                    -0.032410   -0.559800                        5 -106.560 224.8  5.68
## 48  17.67 -2.075  0.81110 -0.009833  0.005268                   -0.7827           7 -103.810 224.9  5.80
## 56  17.89 -2.092  0.79610 -0.009657               0.060700   -0.8209           7 -103.814 224.9  5.81
## 66  14.89 -5.547                                                             3.61500  4 -107.948 225.0  5.87
## 80  16.45 -4.261  0.52590 -0.012110 -0.021400                            2.17100  7 -103.848 225.0  5.88
## 20  15.16 -1.847  0.47070                        -0.815400                        5 -106.821 225.3  6.20
## 96  15.51 -4.236  0.90250 -0.008729 -0.005529   -0.412600           2.02400  8 -102.635 225.6  6.52
## 58  17.74 -1.612                    -0.007476    0.079680   -0.9621           6 -105.651 225.7  6.59
## 124 16.72 -4.555  0.89850            0.004121    0.064630   -0.9756  2.25200  8 -102.691 225.7  6.64
## 101 17.35                 -0.010950                         -0.6853 -1.14400  5 -107.304 226.3  7.16
## 16  17.57 -1.858  0.27410 -0.015040 -0.032230                                     6 -105.981 226.4  7.25
## 62  17.83 -1.509          -0.008956  0.005644    0.156600   -0.7889           7 -104.547 226.4  7.28
## 126 16.96 -3.346          -0.006769  0.012080    0.201200   -0.7485  1.77700  8 -103.019 226.4  7.29
## 32  16.44 -2.011  0.71440 -0.011030 -0.013650   -0.461600                        7 -104.592 226.5  7.37
## 76  16.22 -4.797  0.13400           -0.054150                            2.73100  6 -106.122 226.6  7.53
## 97  17.24                                                       -1.1200 -1.17900  4 -108.851 226.8  7.67
## 28  15.99 -2.009  0.53310           -0.034600   -0.663000                        6 -106.250 226.9  7.79
## 128 16.92 -4.270  0.98180 -0.007670  0.013930    0.112700   -0.8500  2.00000  9 -101.813 227.3  8.14
## 68  14.97 -5.108 -0.26650                                                    3.40100  5 -107.847 227.4  8.25
## 85  15.90                 -0.012910              -0.472600          -1.08200  5 -107.858 227.4  8.27
## 60  17.77 -2.085  0.65570           -0.007926    0.010150   -1.0480           7 -105.166 227.6  8.52
## 10  17.79 -2.032                    -0.086870                                    4 -109.414 227.9  8.80
## 64  17.88 -2.074  0.79840 -0.009911  0.006495    0.080140     -0.8756           8 -103.798 228.0  8.85
## 69  16.81                 -0.021460                                  -1.14500  4 -109.601 228.3  9.17
## 109 17.23                 -0.011300  0.010940                 -0.7378 -1.10900  6 -107.263 228.9  9.82
## 103 17.32        -0.08041 -0.010770                           -0.6646 -1.09500  6 -107.295 229.0  9.88
## 117 17.42                 -0.010970              0.022330     -0.7105 -1.14700  6 -107.303 229.0  9.90
## 99  17.11        -0.31410                                     -1.0110 -0.98480  5 -108.723 229.1 10.00
## 81  14.52                                       -0.881400             -1.07300  4 -110.106 229.3 10.18
## 113 17.12                                       -0.044840     -1.0680 -1.17300  5 -108.848 229.4 10.25
## 105 17.28                           -0.002831                 -1.1030 -1.18800  5 -108.849 229.4 10.25
## 7   15.40        -1.23600 -0.014170                                             4 -110.220 229.5 10.41
## 71  16.66        -0.61020 -0.017660                                   -0.77220  5 -109.035 229.7 10.63
## 35  15.46        -1.18500                                     -0.7263           4 -110.463 230.0 10.90
## 87  15.92        -0.14950 -0.012540             -0.441300             -0.99540  6 -107.829 230.1 10.95
## 93  16.03                 -0.012520 -0.006672   -0.454700             -1.10600  6 -107.840 230.1 10.97
## 77  17.25                 -0.018410 -0.028810                         -1.23800  5 -109.256 230.2 11.07
## 12  17.66 -1.746 -0.33080           -0.079280                                    5 -109.275 230.2 11.11
## 39  15.48        -1.07100 -0.009029                           -0.4090           5 -109.535 230.7 11.63
## 19  13.87        -1.17600                       -0.568500                        4 -111.003 231.1 11.98
## 23  14.69        -1.05600 -0.010230             -0.283800                        5 -109.717 231.1 11.99
## 89  15.32                           -0.030120   -0.746100             -1.18200  5 -109.748 231.2 12.05
## 83  14.71        -0.47020                       -0.746700             -0.79970  5 -109.834 231.3 12.22
## 111 17.20        -0.06962 -0.011140  0.010670                 -0.7185 -1.06800  7 -107.257 231.8 12.70
## 125 17.38                 -0.011380  0.011900    0.058780     -0.8085 -1.11400  7 -107.257 231.8 12.70
## 107 17.14        -0.31490           -0.003154                 -0.9916 -0.99420  6 -108.720 231.8 12.73
## 115 17.11        -0.31450                        0.002015     -1.0130 -0.98480  6 -108.723 231.8 12.74
## 119 17.41        -0.08668 -0.010780              0.034100     -0.7014 -1.09600  7 -107.293 231.9 12.77
## 79  17.02        -0.54130 -0.015670 -0.022860                         -0.88800  6 -108.820 232.0 12.93
## 121 17.13                           -0.003633   -0.055190     -1.0340 -1.18300  6 -108.844 232.1 12.98
## 15  15.42        -1.23900 -0.013850 -0.002928                                    5 -110.217 232.1 12.99
## 43  15.37        -1.14800            0.012690                 -0.8154           5 -110.412 232.5 13.38
## 51  15.42        -1.18200                       -0.014570     -0.7104           5 -110.463 232.6 13.48
## 95  16.05        -0.14760 -0.012170 -0.006535   -0.424200             -1.02000  7 -107.813 232.9 13.81
## 47  15.31        -0.98250 -0.010040  0.026210                 -0.5573           6 -109.316 233.0 13.92
## 91  15.43        -0.42200           -0.027840   -0.635500             -0.92850  6 -109.527 233.5 14.34
## 55  15.51        -1.07300 -0.009033              0.010750     -0.4205           6 -109.535 233.5 14.36
## 27  14.07        -1.19800           -0.009217   -0.522100                        5 -110.968 233.6 14.49
## 21  12.73                 -0.012670             -0.546800                        4 -112.317 233.7 14.61
## 31  14.51        -1.01900 -0.010990  0.011680   -0.321300                        6 -109.664 233.7 14.62
## 37  14.13                 -0.012120                           -0.6856           4 -112.389 233.9 14.75
## 11  15.62        -1.71400           -0.048100                                    4 -112.564 234.2 15.10
## 33  13.90                                                     -1.1690           3 -113.885 234.4 15.29
## 3   15.16        -1.91600                                                        3 -113.925 234.5 15.37
## 4   16.18 -1.024 -1.18100                                                        4 -112.719 234.5 15.41
## 123 17.12        -0.31350           -0.003261   -0.007425     -0.9829 -0.99440  7 -108.720 234.7 15.62
## 45  14.01                 -0.013620  0.051600                 -0.9327           5 -111.566 234.8 15.69
## 17  11.40                                       -0.947400                        3 -114.089 234.8 15.70
## 127 17.38        -0.08129 -0.011200  0.011750    0.069360     -0.7987 -1.06600  8 -107.249 234.9 15.75
## 5   13.58                 -0.022640                                              3 -114.214 235.1 15.95
## 59  15.44        -1.15200            0.013020    0.023760     -0.8435           6 -110.411 235.2 16.11
## 75  16.81        -1.24800           -0.067100                         -0.65210  5 -111.881 235.4 16.32
## 29  12.42                 -0.014480  0.031160   -0.622500                        5 -111.966 235.6 16.49
## 2   16.16 -2.051                                                              3 -114.628 235.9 16.78
## 63  15.56        -0.99680 -0.010140  0.027660    0.095260     -0.6678           7 -109.303 235.9 16.79
## 41  13.79                            0.038270                 -1.3960           4 -113.455 236.0 16.88
## 53  13.32                 -0.011810             -0.330300     -0.3135           5 -112.228 236.1 17.01
## 49  12.89                                       -0.411100     -0.6900           4 -113.650 236.4 17.27
## 67  15.29        -1.86500                                              -0.08522 4 -113.911 236.9 17.79
## 25  11.30                            0.006779   -0.976300                       4 -114.072 237.2 18.12
## 61  13.68                 -0.013410  0.048970   -0.137800     -0.7649           6 -111.539 237.5 18.37
## 13  13.55                 -0.023240  0.005887                                   4 -114.202 237.5 18.37
## 73  17.38                           -0.108000                          -1.57300 4 -114.356 237.8 18.68
## 57  13.10                            0.033200   -0.288100     -1.0310           5 -113.345 238.4 19.25
## 65  14.53                                                              -1.31700 3 -120.415 247.5 28.35
## 9   12.37                           -0.088910                                   3 -120.850 248.3 29.22
## 1   10.64                                                                       2 -124.171 252.6 33.54
## Models ranked by AICc(x)
```

We see a list of all models, sorted by AIC from "best" to "worst." Interestingly, we find that the top model differs from the one chosen using forwards- or backwards-stepwise selection due to using the small sample size correction for AIC. In this case, the model containing `log(brain_wt)`, `predation`, and `danger` has the smallest AIC by 0.7 units and gets the largest weight of 0.102. Yet, several other models have similar AICc values and weights ranging from 0.001 to 0.072. Rather than average across all models, some of which have weights very close to 0, we could choose to average across all models with $\Delta AICc <$ some threshold (e.g., 4):

```r
modaverage <- model.avg(allsubsets, subset = delta < 4)
summary(modaverage)
```

```
##
## Call:
## model.avg(object = allsubsets, subset = delta < 4)
##
## Component model call:
## lm(formula = total_sleep ~ <23 unique rhs>, data = mammalsc)
##
## Component models:
##        df  logLik   AICc delta weight
## 167     5 -103.72 219.11  0.00   0.14
## 1267    6 -102.70 219.81  0.70   0.10
## 137     5 -104.38 220.44  1.33   0.07
## 16      4 -105.69 220.46  1.35   0.07
## 1367    6 -103.14 220.67  1.56   0.06
## 136     5 -104.60 220.86  1.75   0.06
## 157     5 -104.66 220.98  1.87   0.05
## 12367  7 -101.91 221.11  2.00   0.05
## 1357    6 -103.64 221.68  2.57   0.04
## 1567    6 -103.68 221.76  2.65   0.04
## 1467    6 -103.72 221.84  2.73   0.03
## 1236    6 -103.82 222.04  2.93   0.03
## 126     5 -105.19 222.05  2.94   0.03
## 13      4 -106.51 222.11  3.00   0.03
## 135     5 -105.25 222.17  3.06   0.03
## 1257    6 -103.96 222.32  3.21   0.03
## 1237   6 -104.09 222.57  3.46   0.02
## 12357  7 -102.65 222.59  3.48   0.02
## 12567  7 -102.70 222.69  3.58   0.02
## 12467  7 -102.70 222.69  3.58   0.02
## 1347   6 -104.23 222.85  3.74   0.02
## 146    5 -105.66 222.99  3.88   0.02
## 156    5 -105.67 223.01  3.90   0.02
##
## Term codes:
##        danger      exposure     gestation     life_span  log(body_wt) log(brain_wt)     predation
##             1             2             3             4             5             6             7
##
## Model-averaged coefficients:
## (full average)
##                Estimate Std. Error Adjusted SE z value Pr(>|z|)
## (Intercept)   16.5255571  1.4629993   1.4981005  11.031   <2e-16 ***
## danger        -3.2748330  1.5135224   1.5381705   2.129   0.0333 *
## log(brain_wt) -0.5227838  0.4728736   0.4785835   1.092   0.2747
## predation      1.4758462  1.2955748   1.3161523   1.121   0.2621
## exposure       0.2756132  0.5579838   0.5671662   0.486   0.6270
## gestation     -0.0043163  0.0064913   0.0065641   0.658   0.5108
## log(body_wt)  -0.0674230  0.2502241   0.2543647   0.265   0.7910
## life_span     -0.0004167  0.0115914   0.0119608   0.035   0.9722
##
## (conditional average)
##               Estimate Std. Error Adjusted SE z value Pr(>|z|)
## (Intercept)   16.525557  1.462999    1.498101  11.031   <2e-16 ***
## danger        -3.274833  1.513522    1.538171   2.129   0.0333 *
## log(brain_wt) -0.763824  0.377620    0.387988   1.969   0.0490 *
## predation      2.066175  1.063102    1.097919   1.882   0.0598 .
## exposure       0.841235  0.688824    0.711346   1.183   0.2370
## gestation     -0.009928  0.006419    0.006587   1.507   0.1318
## log(body_wt)  -0.271224  0.443391    0.452769   0.599   0.5492
## life_span     -0.004280  0.036925    0.038116   0.112   0.9106
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see a list of the models that we average over and also see two sets of averaged coefficients. The set of coefficients listed under `(full average)` uses 0's for coefficients when averaging over models that do not contain the focal predictor. By contrast, the set of coefficients under `(conditional average)` only averages coefficients in the models where the predictor is included. The former approach has a stronger theoretical foundation and results in a similar effect as various penalization strategies (see Section 8.7); namely, this approach to model averaging will shrink parameters associated with weak parameters towards 0 (Lukacs, Burnham, and Anderson 2010). We can see this effect if we plot the model-averaged coefficients against the coefficients from the full model. This "shrinkage" effect turns out to be a useful characteristic that often improves mean-squared prediction error, MSE:

$$MSE = \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)}{n}.$$

```
# pull off average coefficients and their SEs
avecoef <- (summary(modaverage))$coefmat.full
```

```r
#reorder terms so they appear the same as in fullmodelsum, below
avecoef <- avecoef[c(1,8,6,3,7,4,5,2),]
fullmodelsum <- broom::tidy(fullmodel)
combinedcoef <- data.frame(coefs = c(avecoef[,1], fullmodelsum$estimate),
                           se = c(avecoef[,3], fullmodelsum$std.error),
                           term = as.factor(rep(fullmodelsum$term, 2)),
                           method = rep(c("Model Average", "Full Model"), each =8)) %>%
                mutate(upci= coefs + 1.96*se,
                           loci = coefs - 1.96*se)
```

```r
ggplot(combinedcoef, aes(method, coefs)) + geom_point()+
    geom_errorbar(aes(ymin = loci, ymax = upci, width = 0.2)) +
    facet_wrap(~term, scales="free") +theme_bw()
```

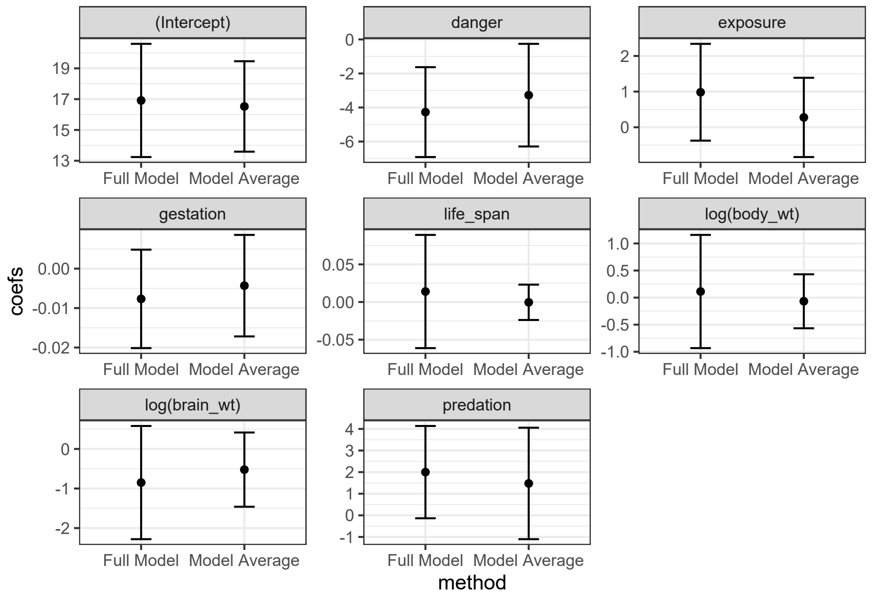


FIGURE 8.2 Comparison of full model and model-avaraged coefficient estimates and 95% confidence intervals.

8.7 Regularization using penalization

As mentioned in the previous section, if your goal is prediction, it may be useful to shrink certain coefficients towards 0. One way to accomplish this goal is to add a penalty, $\lambda \sum_{j=1}^{p} |\beta_j|^\gamma$, to the objective function used to estimate parameters (e.g., sum-of-squared errors or likelihood), where p is equal to the number of coefficients associated with our explanatory variables:

$$\sum_{i=1}^{n}(Y - \hat{Y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|^\gamma$$

Typically, γ is set to either 1 or 2, with $\gamma = 1$ corresponding to the LASSO (Least Absolute Shrinkage and Selection Operator) and $\gamma = 2$ corresponding to ridge regression. The two penalties can also be combined into what is referred to as the elastic net. When $\lambda = 0$, we get the normal least-squares (or Maximum Likelihood) estimator[4]. As λ gets larger, coefficients will shrink towards 0. The degree of shrinkage will differ between the LASSO and Ridge regression estimators due to the difference in the penalty functions. When using the LASSO, some coefficients may get shrunk to 0 depending on the value of λ, which allows some predictors to be dropped from the model. With Ridge regression, all coefficients will always be retained. Thus, the two approaches perform very differently when faced with strong collinearity. The LASSO will usually select one of a set of collinear variables, whereas Ridge regression will retain all variables but shrink their coefficients towards 0. Typically, a gridded set of λ values are considered and the value of λ that performs best (e.g., minimizes mean-squared prediction error) is selected

Why do shrinkage estimators improve predictive performance? Essentially, shrinkage can be viewed as a method for reducing model complexity; shrinkage reduces variability in the estimates of β at the expense of introducing some bias (Hooten and Hobbs 2015). This bias-variance tradeoff can be extremely effective in reducing prediction error, particularly when faced with too many predictors relative to the available sample size or in cases of extreme collinearity.

Below, we will estimate parameters using the LASSO and Ridge regression applied to the sleep data set. We will make use of the `glmnet` package (Friedman, Hastie, and Tibshirani 2010), which works with matrices rather than dataframes. Therefore, we will first need to create our logged predictors of body and brain weight. We can specify $\alpha = 1$ for the LASSO, $\alpha = 0$ for Ridge regression and values in-between for the elastic net which combines both penalties.

```
library(glmnet)
mammalsc <- mammalsc %>% mutate(logbrain_wt = log(brain_wt),
                                logbody_wt = log(body_wt))
x = as.matrix(mammalsc[, c("life_span", "gestation", "logbrain_wt",
                           "logbody_wt", "predation", "exposure", "danger")])
fitlasso <- glmnet(x = x,
                   y= mammalsc$total_sleep, alpha=1)
```

By default, `glmnet` will estimate parameters across a range of λ values chosen in a smart way, but users can change the defaults. We can then inspect how the coefficient estimates (and number of coefficients in the model) change as we increase λ using the `plot` function with argument `xvar = "lambda"` (Figure 8.3).

```
plot(fitlasso, xvar = "lambda", label = TRUE)
```

At this point, we have a number of different models, each with a different set of coefficient estimates determined by the unique set of λ values. We can see how the number of non-zero coefficients changes with λ and also the percent of the Deviance explained by the model (%Dev), which in the case of linear regression is equivalent to R^2, using the `print` function.

```
print(fitlasso)
```

```
##
## Call:  glmnet(x = x, y = mammalsc$total_sleep, alpha = 1)
##
##    Df  %Dev  Lambda
```

[4]If all of our assumptions of linear regression hold, then least squares will give us the Maximum Likelihood estimate - see Section 10.11.

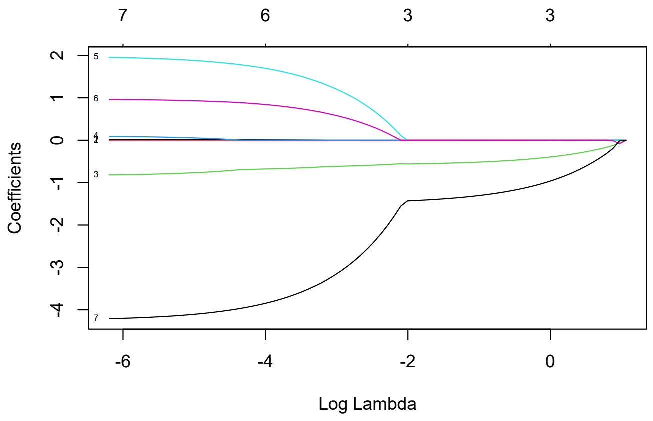


FIGURE 8.3 Coefficient estimates for predictors with the sleep data set when using the LASSO with different values of λ.

```
## 1   0  0.00 2.89600
## 2   4  8.32 2.63800
## 3   4 17.42 2.40400
## 4   3 24.79 2.19000
## 5   3 30.87 1.99600
## 6   3 35.92 1.81900
## 7   3 40.12 1.65700
## 8   3 43.60 1.51000
## 9   3 46.49 1.37600
## 10  3 48.89 1.25300
## 11  3 50.88 1.14200
## 12  3 52.54 1.04100
## 13  3 53.91 0.94820
## 14  3 55.05 0.86390
## 15  3 56.00 0.78720
## 16  3 56.78 0.71730
## 17  3 57.44 0.65350
## 18  3 57.98 0.59550
## 19  3 58.43 0.54260
## 20  3 58.80 0.49440
## 21  3 59.11 0.45050
## 22  3 59.37 0.41040
## 23  3 59.58 0.37400
## 24  3 59.76 0.34080
## 25  3 59.91 0.31050
## 26  3 60.03 0.28290
## 27  3 60.13 0.25780
## 28  3 60.21 0.23490
## 29  3 60.28 0.21400
## 30  3 60.34 0.19500
## 31  3 60.39 0.17770
## 32  3 60.43 0.16190
## 33  3 60.46 0.14750
## 34  3 60.49 0.13440
## 35  4 60.85 0.12250
## 36  5 61.60 0.11160
## 37  5 62.24 0.10170
## 38  5 62.76 0.09264
## 39  5 63.20 0.08441
## 40  5 63.57 0.07691
## 41  5 63.87 0.07008
## 42  5 64.12 0.06385
## 43  5 64.33 0.05818
## 44  5 64.51 0.05301
## 45  5 64.65 0.04830
## 46  5 64.77 0.04401
## 47  6 64.89 0.04010
## 48  6 64.99 0.03654
## 49  6 65.07 0.03329
## 50  6 65.13 0.03033
## 51  6 65.19 0.02764
## 52  6 65.24 0.02518
## 53  6 65.28 0.02295
## 54  6 65.31 0.02091
## 55  6 65.34 0.01905
## 56  6 65.36 0.01736
## 57  6 65.38 0.01582
## 58  6 65.39 0.01441
## 59  7 65.41 0.01313
## 60  7 65.43 0.01196
## 61  7 65.44 0.01090
## 62  7 65.45 0.00993
## 63  7 65.46 0.00905
## 64  7 65.47 0.00825
## 65  7 65.48 0.00751
## 66  7 65.48 0.00685
## 67  7 65.49 0.00624
## 68  7 65.49 0.00568
## 69  7 65.50 0.00518
## 70  7 65.50 0.00472
## 71  7 65.50 0.00430
## 72  7 65.50 0.00392
## 73  7 65.51 0.00357
## 74  7 65.51 0.00325
## 75  7 65.51 0.00296
## 76  7 65.51 0.00270
## 77  7 65.51 0.00246
## 78  7 65.51 0.00224
## 79  7 65.51 0.00204
```

Lastly, we can use cross-validation (see Section 8.8.1) to choose an optimal λ. Cross-validation attempts to evaluate model performance by training the model on subsets of the data and then testing its predictions on the remaining data. This can be accomplished using the `cv.glmnet` function:

```
cvfit.lasso <- cv.glmnet(x = x,
                   y= mammalsc$total_sleep, alpha=1)
```

We can use the associated `plot` function to see how our estimates of mean-squared error change with λ (Figure 8.4).

```
plot(cvfit.lasso)
```

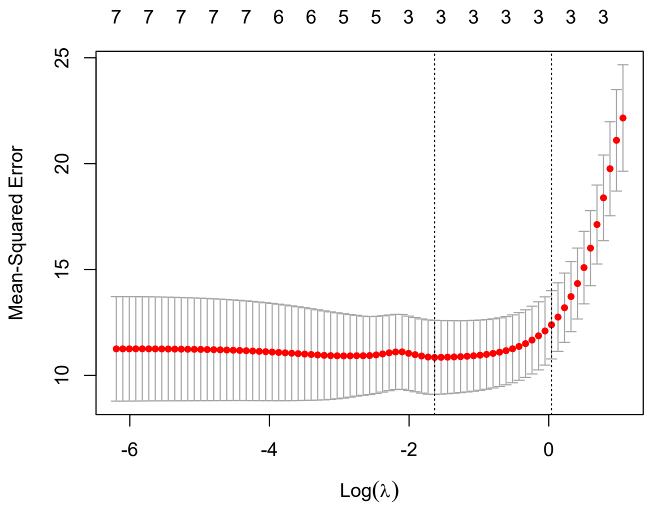


FIGURE 8.4 Mean-squared prediction error estimated using cross-validation with the sleep data set for different values of λ and $\alpha = 1$ (i.e., using the LASSO).

For each value of λ, the red dots and associated interval represent the estimated MSE along with $\pm$ 1 SE. The left-most dotted line corresponds to the values of λ that minimizes the mean-squared error. The rightmost line gives the largest value of λ that results in an estimated MSE within 1SE of the minimum observed. Lastly, we can inspect the value of λ that gives the smallest MSE and also the resulting coefficients using:

```
cvfit.lasso$lambda.min
```

```
## [1] 0.1949944
```

```
coef(cvfit.lasso, s="lambda.min")
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) 16.976166754
## life_span     .
## gestation   -0.008038413
## logbrain_wt -0.548365524
## logbody_wt    .
## predation     .
## exposure      .
## danger      -1.398245832
```

We see that we are left with the same 3 predictors as the model with the lowest AIC_C, but that their coefficients are also shrunk towards 0:

```
allsubsets[1,]
```

```
## Global model call: lm(formula = total_sleep ~ life_span + gestation + log(brain_wt) +
##     log(body_wt) + predation + exposure + danger, data = mammalsc)
## ---
## Model selection table
##    (Int)    dng log(brn_wt)   prd df   logLik  AICc delta weight
## 98  16.4 -3.628     -0.6763 1.992  5 -103.722 219.1     0      1
## Models ranked by AICc(x)
```

We can repeat the process using Ridge regression:

```
cvfit.ridge <- cv.glmnet(x = x,
                   y= mammalsc$total_sleep, alpha=0)
cvfit.ridge$lambda.min
```

```
## [1] 1.408017
```

```
coef(cvfit.ridge, s="lambda.min")
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) 16.225126140
## life_span   -0.008771936
## gestation   -0.007109805
## logbrain_wt -0.314594127
## logbody_wt  -0.181114581
## predation   -0.193136878
## exposure    -0.145969577
## danger      -0.948772311
```

In this case, all coefficients are non-zero. Lastly, if we compare these coefficients to those from model-averaging and from the full model (Figure 8.5), we see that the coefficients from Ridge regression are shrunk towards 0, in this case more so than model averaging.

One challenge with penalization-based methods is that it is not straightforward to calculate appropriate confidence intervals due to the bias introduced by the penalization, and therefore `glmnet` does not supply SEs. In many ways, a Bayesian treatment of regularization seems more natural in that the penalization terms can be viewed as arising from specific prior specifications (Tibshirani 2011; Casella et al. 2010). For a fuller treatment of regularization methods and connections between the LASSO, Ridge Regression, and Bayesian methods, see Hooten and Hobbs (2015).

8.8 Evaluating model performance

We have now seen a few different modeling strategies, identified some of the challenges inherent to them, including the potential of overfitting data, and discussed ways that we can improve predictive performance via

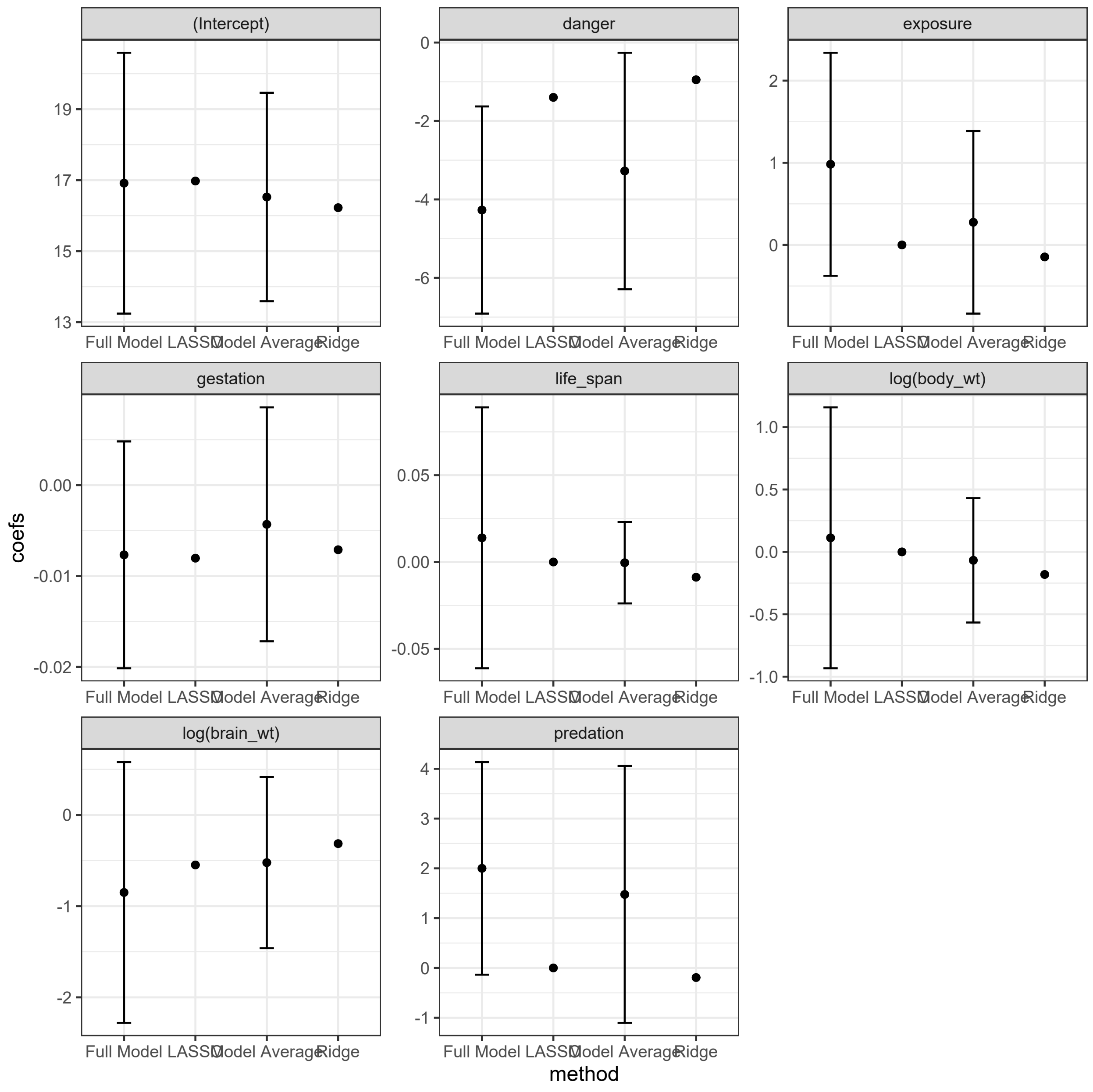


FIGURE 8.5 Comparison of full model, model-averaged and LASSO and Ridge-Regression coefficients when applied to the sleep data set.

shrinkage of coefficients towards 0. It is also important that we have tools for evaluating models and modeling strategies. Two powerful approaches are cross-validation, which was briefly mentioned in the previous section, and bootstrapping.

8.8.1 Cross-validation

One potential issue with evaluating models via data splitting (i.e., the approach I used during my Linear Regression midterm; Section 8.3) is that estimates of model performance can be highly variable depending on how observations get partitioned into training and test data sets. One way to improve upon this approach is to use k-fold cross validation in which the data are partitioned into k different subsets, referred to as folds (Mosteller and Tukey 1968). The model is then trained using all data except for 1 of the folds which serves as the test data set. This process is then repeated with each fold used as the test data set. The steps are as follows:

1. Split the data into many subsets ($D_i = 1, 2, \dots k$).
2. Loop over i:
   a. Fit the model using data from all subsets except i. We will use a $_{-i}$ subscript to refer to subsets that include all data except those in the i^{th} fold (e.g., D_{-i}).
   b. Predict the response for data in the i^{th} fold: $\hat{Y}_i$.
3. Pool results and evaluate model performance, e.g., by comparing Y_i to $\hat{Y}_i$ ($i = 1, 2, \dots, n$).

We will demonstrate this process using functions in the `caret` package (Kuhn 2021), though it is also easy to implement the approach on your own (e.g., using a `for` loop). The `caret` package allows one to implement various forms of data partitioning/resampling via its `trainControl` function. Here, we will specify that we want to evaluate the performance using cross-validation (`method = "cv"`) with 10 folds (`number = 10`). Let's evaluate the model containing our same set of sleep predictors.

```r
library(caret)
set.seed(1045) # to ensure we get the same result every time.
# defining training control
# as cross-validation and
# value of K equal to 10
train_control <- trainControl(method = "cv",
                              number = 10)

# training the model by assigning total_sleep column
# as target variable and rest other column
# as independent variable
model <- train(total_sleep ~ life_span + gestation + log(brain_wt) +
                 log(body_wt) + predation + exposure + danger,
               data = mammalsc,
               method = "lm",
               trControl = train_control)

# printing model performance metrics
# along with other details
print(model)
```

```
## Linear Regression
##
```

```
## 42 samples
##  7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 36, 38, 38, 38, 38, 38, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   3.598593  0.5286154  3.001091
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

We can repeat this approach multiple times (referred to as repeated k-fold cross validation) by changing the `method` argument of `trainControl` to `repeatedcv` and by specifying how many times we want to repeat the process using the `repeats` argument.

```
train_control <- trainControl(method = "repeatedcv",
                              number = 10, repeats=5)

# training the model by assigning sales column
# as target variable and rest other column
# as independent variable
model <- train(total_sleep ~ life_span + gestation + log(brain_wt) +
                 log(body_wt) + predation + exposure + danger,
               data = mammalsc,
               method = "lm",
               trControl = train_control)

# printing model performance metrics
# along with other details
print(model)
```

```
## Linear Regression
##
## 42 samples
##  7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 37, 38, 38, 39, 38, 36, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   3.328851  0.6342511  2.792544
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

We can compare these results to the estimate R^2 and RMSE (equivalent to $\hat{\sigma}$ when using linear regression) from the fit of our full model.

```
summary(fullmodel)$r.squared
```

```
## [1] 0.6551552
```

```
summary(fullmodel)$sigma
```

```
## [1] 3.036907
```

We see that the cross-validation estimate of R^2 is smaller and the estimate of $\hat{\sigma}$ is larger than the estimates provided by `lm`. Although the differences are small here, cross-validation can be particularly useful for estimating out-of-sample performance (i.e., performance of the model when applied to new data not used to fit the model) when model assumptions do not hold.

Most ecological data sets are subjected to some form of correlation due to repeated observations on the same sample unit or spatial or temporal autocorrelation, and thus, observations will not be independent. When applying cross-validation to data with various forms of dependencies, it is important to create folds that are independent of each other (Roberts et al. 2017); the data within folds, however, can be non-independent. For spatial or temporal data, this can often be accomplished by creating blocks of data that are clustered in space or time. Several R packages have been developed to facilitate spatial cross-validation, including `sperrorest` (Brenning 2012), `ENMeval` (Kass et al. 2021), and `blockCV` (Valavi et al. 2019).

8.8.2 Boostrapping to evaluate model stability

So far, we have looked at how we can evaluate model performance using cross-validation. This process attempts to evaluate model performance when applied to a new data set. Our examples in the last section assumed we had a pre-specified model. What if we wanted to also evaluate the impact of various modeling choices along the way – e.g., stepwise-selection procedures or tuning parameters (e.g., λ) used in regularization methods? We could repeat these steps (model selection, model tuning) with each fold. Similarly, we could evaluate model performance and also modeling strategies using bootstrap resampling. In this section, we will highlight a bootstrap approach implemented in the `rms` package (Harrell Jr 2021).

To evaluate model performance using a bootstrap, we will mimic the entire process from start to finish, including:

a. Creating training and test data sets using separate bootstraps.
b. Training the model using the training data (this step could include choosing tuning parameters or using stepwise-selection algorithms to determine a final model).
c. Evaluating the model using the test data.

To estimate model performance metrics (Figure 8.6A), we begin by estimating the average degree of *optimism* resulting from evaluating the model using the same data as was used to train the model. We then subtract this optimism from the same performance metrics calculated using the full data set. The steps here are:

1. Fit a model to the full data set and estimate R^2.
2. Form bootstrapped test and bootstrapped training data sets by resampling the original data set with replacement.
3. Fit the full model to the bootstrapped training data set and estimate R^2_{trian}.
4. Use the model from step 3 to form predictions for the test data set.
5. Use the predictions from step 4 and the test data to estimate R^2_{test}.

6. Estimate the degree of optimism, $\delta_i = R^2_{train} - R^2_{test}$.
7. Estimate an improved R^2 by subtracting the average optimism across the different bootstrap replicates from the R^2 in step 1.

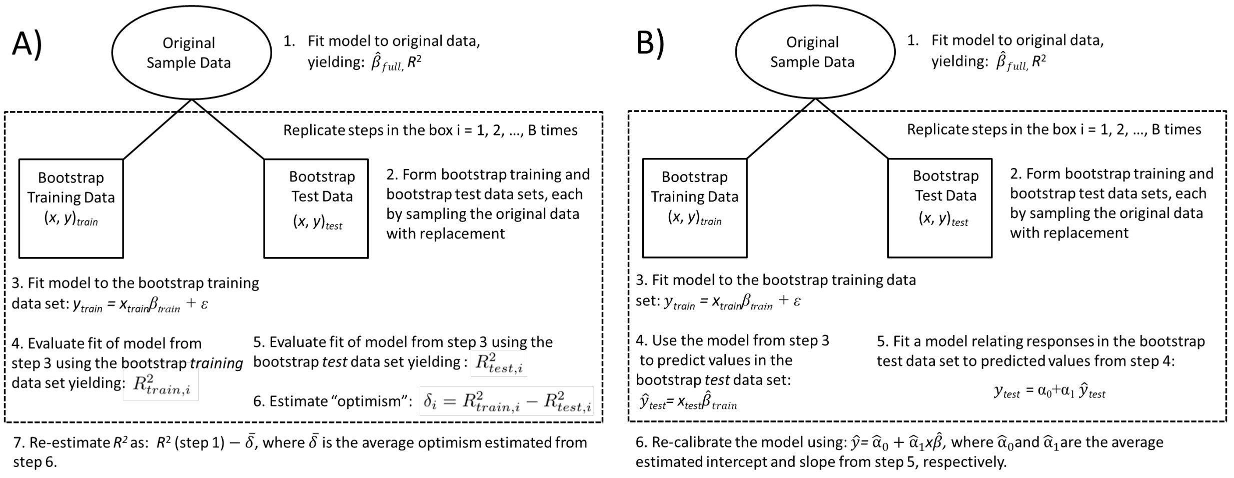

A)
Original Sample Data
1. Fit model to original data, yielding: $\hat{\beta}_{full}$, R^2
Replicate steps in the box i = 1, 2, …, B times
Bootstrap Training Data $(x, y)_{train}$
Bootstrap Test Data $(x, y)_{test}$
2. Form bootstrap training and bootstrap test data sets, each by sampling the original data with replacement
3. Fit model to the bootstrap training data set: $y_{train} = x_{train}\beta_{train} + \varepsilon$
4. Evaluate fit of model from step 3 using the bootstrap *training* data set yielding: $R^2_{train,i}$
5. Evaluate fit of model from step 3 using the bootstrap *test* data set yielding: $R^2_{test,i}$
6. Estimate "optimism": $\delta_i = R^2_{train,i} - R^2_{test,i}$
7. Re-estimate R^2 as: R^2 (step 1) $- \bar{\delta}$, where $\bar{\delta}$ is the average optimism estimated from step 6.
B)
Original Sample Data
1. Fit model to original data, yielding: $\hat{\beta}_{full}$, R^2
Replicate steps in the box i = 1, 2, …, B times
Bootstrap Training Data $(x, y)_{train}$
Bootstrap Test Data $(x, y)_{test}$
2. Form bootstrap training and bootstrap test data sets, each by sampling the original data with replacement
3. Fit model to the bootstrap training data set: $y_{train} = x_{train}\beta_{train} + \varepsilon$
4. Use the model from step 3 to predict values in the bootstrap *test* data set: $\hat{y}_{test} = x_{test}\hat{\beta}_{train}$
5. Fit a model relating responses in the bootstrap test data set to predicted values from step 4:
$y_{test} = \alpha_0 + \alpha_1 \hat{y}_{test}$
6. Re-calibrate the model using: $\hat{y} = \hat{\alpha}_0 + \hat{\alpha}_1 x\hat{\beta}$, where $\hat{\alpha}_0$ and $\hat{\alpha}_1$ are the average estimated intercept and slope from step 5, respectively.

FIGURE 8.6 A bootstrapping approach to evaluate model performance (Fieberg and Johnson 2015).

We can also use a bootstrap to recalibrate our predictions. If we have overfit the data, it may help to shrink our predictions back towards the mean response in the data set (Figure 8.6B). We can estimate an appropriate degree of shrinkage by comparing how well our predictions from models fit to the training data match observations in the test data set. Here the steps are similar to those outlined above:

1. Estimate regression parameters using the full data set.
2. Form bootstrapped test and bootstrapped training data sets by resampling the original data set with replacement.
3. Fit the full model to the bootstrapped training data set
4. Use the model from step 3 to form predicted responses for the bootstrapped test data set.
5. Fit a linear regression model using the test data responses (as the response variable) and the predicted values from step 4 as the only explanatory variable.
6. Repeat steps 2-5 many times. The average slope in step 5 can be used to proportionally reduce, or shrink, the regression parameters from step 1 towards 0; if no overfitting has occurred, the average slope in step 5 will be 1.

The `rms` package has functions for evaluating a variety of models (see below), including those obtained by backwards selection. However, users must fit their models using functions specific to the `rms` package:

- linear regression models (using `ols` rather than `lm`)
- generalized least squares (using `Gls` rather than `gls`)
- logistic regression models (using `lrm` rather than `glm`)
- cox proportional hazards models (using `cpm` rather than `cph`)

Unfortunately, methods for validating mixed effect models [see Chapter 18)] are not available in the `rms` package.

Let's explore this approach with the sleep data. We must first fit the model using `ols`. We can then evaluate model performance when also including backwards selection using the `validate` function with the argument `bw = TRUE`.

```
library(rms)
set.seed(130)
fullmod.ols<-ols(total_sleep ~ life_span + gestation + log(brain_wt) +
                   log(body_wt) + predation + exposure + danger,
               data = mammalsc, x = TRUE, y = TRUE)
  validate(fullmod.ols, bw = TRUE)
```

```
##
##      Backwards Step-down - Original Model
##
##  Deleted   Chi-Sq d.f. P      Residual d.f. P      AIC   R2
##  body_wt   0.04   1    0.8326 0.04     1    0.8326 -1.96 0.655
##  life_span 0.11   1    0.7448 0.15     2    0.9275 -3.85 0.654
##  gestation 1.32   1    0.2500 1.47     3    0.6883 -4.53 0.640
##  exposure  1.76   1    0.1845 3.23     4    0.5193 -4.77 0.622
##  predation 3.66   1    0.0557 6.90     5    0.2284 -3.10 0.585
##
## Approximate Estimates after Deleting Factors
##
##              Coef   S.E. Wald Z         P
## Intercept 17.4416 1.0680 16.331 0.000e+00
## brain_wt  -0.9194 0.1974 -4.658 3.199e-06
## danger    -1.5755 0.3566 -4.418 9.977e-06
##
## Factors in Final Model
##
## [1] brain_wt danger
```

```
##           index.orig training    test optimism index.corrected  n
## R-square      0.5852   0.6279  0.5205   0.1074          0.4778 40
## MSE           8.9805   7.2317 10.3815  -3.1498         12.1303 40
## g             4.1115   4.0430  3.9206   0.1223          3.9892 40
## Intercept     0.0000   0.0000  0.6651  -0.6651          0.6651 40
## Slope         1.0000   1.0000  0.9432   0.0568          0.9432 40
##
## Factors Retained in Backwards Elimination
##
##  life_span gestation brain_wt body_wt predation exposure danger
##            *         *                 *         *
##                               *                           *
##                      *                 *                  *
##            *                                              *
##                                        *                  *
##  *         *         *                 *                  *
##                               *
##                      *                                    *
##                               *                  *        *
##            *                           *                  *
##            *                                              *
##                                        *                  *
##                                        *                  *
##            *                          *         *        *
##                      *                                   *
##                                       *                  *
##                      *                                   *
##                      *                *                  *
##                                       *                  *
##                      *                                   *
##                      *                                   *
##  *                                    *                  *
##                               *
##                      *                *                  *
##            *                                             *
##                      *                                   *
##            *                          *                  *
##                                       *                  *
##                      *                                   *
##            *                                             *
##            *                                             *
##                      *                *         *        *
##            *                  *                          *
##            *                          *                  *
##                               *       *                  *
##  *                                                       *
##                                       *                  *
##                                       *                  *
##                               *                          *
##  *         *                  *                 *        *
##
## Frequencies of Numbers of Factors Retained
##
##  1  2  3  4  5
##  2 23 10  3  2
```

We see that backwards elimination results in a fair degree of optimism. Correcting for it decreases our estimate of R^2 and increases our estimate of the Mean-Squared Error. The estimates of intercept and slope also tell us that we can improve prediction error by recalibrating our model, shrinking our original estimates, $\hat{Y}_i^{naive}$, towards the overall mean using $\hat{Y}_i^C = 0.6651 + 0.9432\hat{Y}_i^{naive}$. The `validate` function also shows us which predictors are chosen when the backwards selection algorithm is applied to each bootstrap sample. We see that there is quite a bit of variability in the final model that is chosen. Thus, if we were to collect another data set of the same size and apply the same model-selection algorithm, we might vary well end up with a different set of variables in the reduced model. We may choose to communicate this uncertainty due to model selection by reporting bootstrap inclusion frequencies (how often variables are selected in final models) or by listing other "competitive models" (i.e, those models that are frequently chosen across the bootstrap replicates) (Heinze, Wallisch, and Dunkler 2018).

8.9 Summary

In this section, we have seen several modeling strategies, including:

1. Stepwise selection algorithms (possibly with change-in-estimate criterion)
2. Full model inference (df spending)
3. Model averaging (using multiple models for inference)
4. Penalized estimation methods (LASSO, Ridge regression)

For describing associations, we may consider any of these methods including the use of a full model or stepwise-selection algorithms. Note, however, that interpretation of coefficients can be more challenging when using model-averaging or regularization/penalization due to the bias introduced when estimating regression coefficients with shrinkage. These latter methods are geared towards improving predictions. For inference, we are likely best off fitting a small number of informed models (e.g., representing different causal mechanisms) which we can then compare. Lastly, we have seen how cross-validation and bootstrap resampling methods can help us quantify how (possibly tuned) models will likely perform when applied to new data. There is tremendous demand for *analysts* that can develop good predictive models, and this has fueled the development of open-source software for tuning and evaluating models. In addition to the tools available in the `caret` package (Kuhn 2021), the `tidymodels` and `workflows` packages offer powerful options for evaluating models using resampling-based techniques (Kuhn and Wickham 2020; Vaughan 2021). Useful tutorials for these packages can be found here and here.

Part III

Frequentist and Bayesian Inferential Frameworks

9

Introduction to probability distributions

Learning objectives

1. Understand and be able to work with basic rules of probability. We will need these rules to understand Bayes Theorem, which is fundamental to Bayesian statistics.
2. Gain a basic familiarity with concepts related to mathematical statistics (e.g., random variables, probability mass and density functions, expected value, variance, conditional and marginal probability distributions).
3. Understand how to work with different statistical distributions in R.
4. Understand the role of random variables and common statistical distributions in formulating modern statistical regression models.
5. Be able to choose an appropriate statistical distribution for modeling your data.

9.1 Statistical distributions and regression

Up until now, our focus has been on linear regression models, which can be represented as:

$$\begin{aligned} Y_i &= \beta_0 + X_i\beta_1 + \epsilon_i \\ \epsilon_i &\sim N(0, \sigma^2) \end{aligned} \tag{9.1}$$

or

$$Y_i \sim N(\beta_0 + X_i\beta_1, \sigma^2)$$

or

$$\begin{aligned} Y_i &\sim N(\mu_i, \sigma^2) \\ \mu_i &= \beta_0 + X_i\beta_1 \end{aligned} \tag{9.2}$$

Moving forward, we are going to use the last formulation to emphasize the role of the Normal distribution as our data-generating model. This formulation allows us to easily see that we are:

1. Modeling the mean, μ_i, of the Normal distribution as a linear combination of our explanatory variables; GLS models (Chapter 5) also allow us to model σ_i as a function of explanatory variables.

- The Normal distribution describes the variability of the observations about μ_i.

The Normal distribution has several characteristics that make it a poor data generating model for certain types of data (e.g., count or binary data):

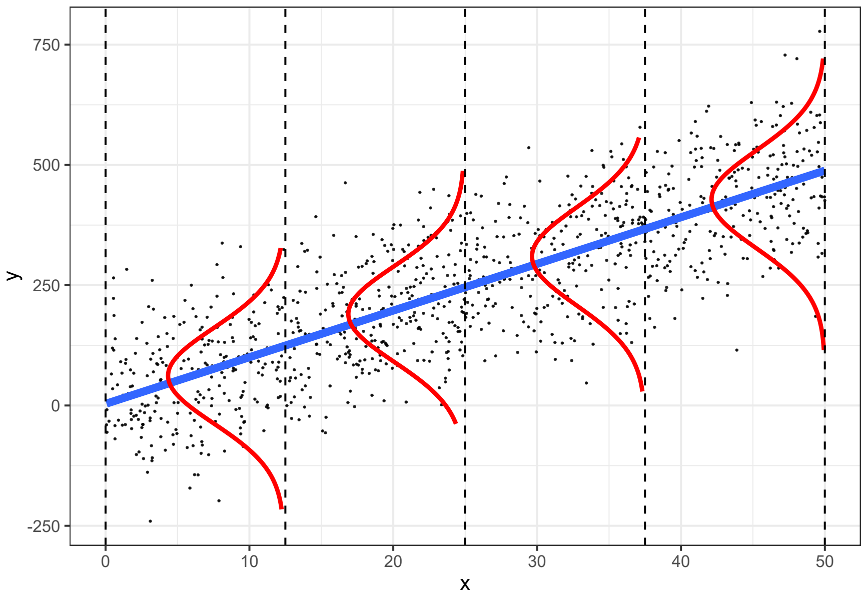


FIGURE 9.1 The role of the Normal distribution as a data-generating model. This figure was produced using modified code from an answer on stackoverflow by Roback and Legler (2021) (https://bookdown.org/roback/bookdown-bysh/).

- the distribution is bell-shaped and symmetric
- observations can take on any value between $-\infty$ and ∞

In this chapter, we will learn about other statistical distributions and ways to work with them in R. Along the way, we will learn about discrete and continuous random variables and their associated probability mass functions (discrete random variables) and probability density functions (continuous random variables). We will also learn a few basic probability rules that can be useful for understanding Bayesian methods and for constructing models using conditional probabilities (e.g., occupancy models; MacKenzie et al. 2017).

9.2 Probability rules

We will begin by highlighting a few important probability rules that we will occasionally refer back to:

1. Multiplicative rule: $P(A \text{ and } B) = P(A)P(B \mid A) = P(B)P(A \mid B)$
2. Union: $P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$
3. Complement rule: $P(\text{not } A) = 1 - P(A)$
4. Conditional probability rule: $P(A \mid B) = \frac{P(A \text{ and } B)}{P(B)}$

If you forget these rules, it is often helpful to create a visualization using a Venn diagram (i.e., an illustration with potentially overlapping circles representing different events). For example, consider the Venn diagram representing numbers between 1 and 10 along with whether the numbers are prime and whether they are even (Figure 9.2).

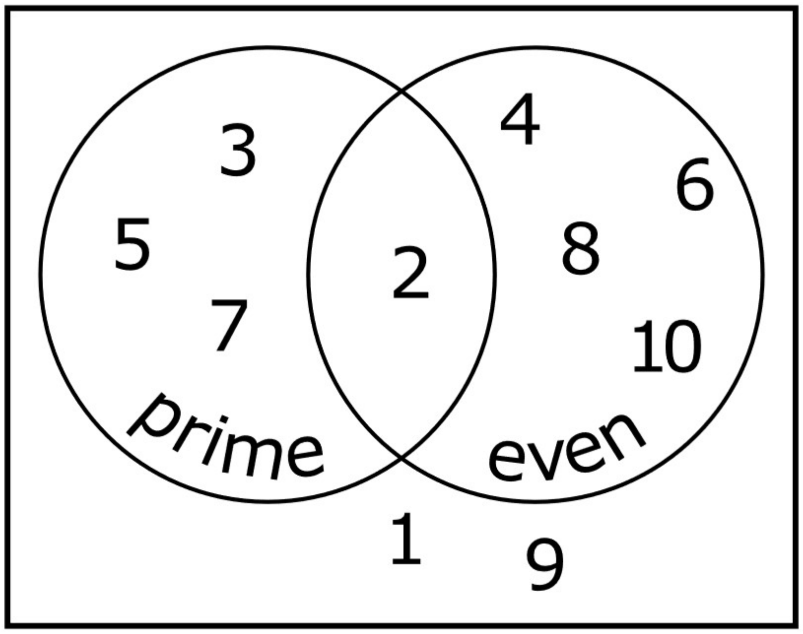


FIGURE 9.2 Venn diagram with even and prime numbers from 1 to 10. Guy vandegrift / Wikimedia Commons / CC-BY-SA 4.0 International.

Let A represent the set of prime numbers and B represent the set even numbers. We see that $P(A \text{ and } B)$ is determined by the intersection of the two circles and is equal to 1/10 (i.e., there is only one number, 2, in the intersection out of ten possible numbers). To understand the *Union rule*, we can consider that $P(A) = 4/10$ and $P(B) = 5/10$. To determine $P(A \text{ or } B)$, we can by sum these two probabilities, but then we need to subtract $P(A \text{ and } B)$. Otherwise, we would be including the number 2 twice.

It is also helpful to know of two special cases having to do with mutually exclusive or independent events. If events A and B are mutually exclusive (meaning that they both cannot happen), then:

- $P(A \text{ or } B) = P(A) + P(B)$
- $P(A \text{ and } B) = 0$

If events A and B are independent:

- $P(A \mid B) = P(A)$ (intuitively, if A and B are independent, knowing that event B happened does not change the probability that event A happened)
- $P(A \text{ and } B) = P(A)P(B)$ (we will use this rule when we construct likelihoods)

Lastly, it will be helpful to know the Law of Total Probability (Figure 9.3). If we partition the sample space into a set of k mutually exclusive events, $B_1, B_2, \ldots, B_k$, that together make up all possibilities, then:

$$P(A) = \sum_{i=1}^{k} P(A|B_i)P(B_i)$$

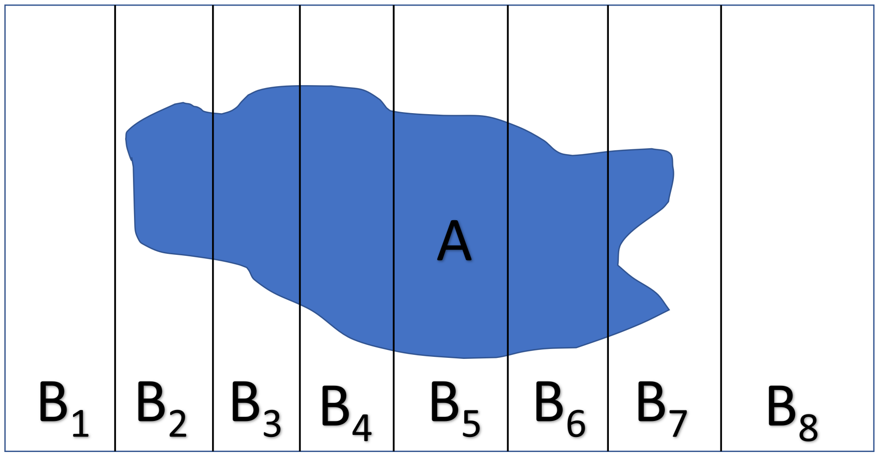


FIGURE 9.3 The total law of probability, $P(A) = \sum_{i=1}^{k} P(A|B_i)P(B_i)$, can be understood by partitioning the sample space into a discrete number of mutually exclusive events that together make up all possibilities.

A special case of the total law of probability is:

$$P(A) = P(A \text{ and } B) + P(A \text{ and } (\text{not } B))$$

Lastly, we can combine several of these probably rules to form Bayes Theorem:

$$P(A \mid B) = \frac{P(A \text{ and } B)}{P(B)} \tag{9.3}$$

$$= \frac{P(B \mid A)P(A)}{P(B \text{ and } A) + P(B \text{ and not } A)} \tag{9.4}$$

$$= \frac{P(B \mid A)P(A)}{P(B \mid A)P(A) + P(B \mid \text{not } A)P(\text{not } A)} \tag{9.5}$$

We can also write Bayes rule more generally as:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{\sum_{i=1}^{k} P(B \mid A_i)P(A_i)} \tag{9.6}$$

9.2.1 Application of Bayes theorem to diagnostic screening tests

Bayes theorem is the foundation of Bayesian statistics, but it is also often used for frequentist inference, for example, to understand results of various diagnostic tests for which we often have poor intuition (Bramwell, West, and Salmon 2006). As an example, consider the following question posed by Bramwell, West, and Salmon (2006):

The serum test screens pregnant women for babies with Down's syndrome. The test is a very good one, but not perfect. Roughly 1% of babies have Down's syndrome. If the baby has Down's syndrome, there is a 90% chance that the result will be positive. If the baby is unaffected, there is still a 1% chance that the result will be positive. A pregnant woman has been tested and the result is positive. What is the chance that her baby actually has Down's syndrome?

The answer, using the information above, is $\approx$ 48%. Surprised? You are not alone – Bramwell, West, and Salmon (2006) found that only 34% of obstetricians, 0% of midwives, and 9% of pregnant females arrived at this correct answer (and many of these participants benefited from an alternative presentation in terms of frequencies); See also Weber, Binder, and Krauss (2018) for a similar investigation.

The 1% prevalence rate assumed by Bramwell, West, and Salmon (2006) is close to what the CDC reports for women that are 40 years old or older. The prevalence rate in the general population is actually much lower (closer to 0.1%). As a result, the probability that a baby has Down's syndrome given that a younger mother receives a positive test is actually much lower than 48%. Let's repeat the analysis with the lower prevalence rate of 0.1%.

Let:

- $P(D)$ = the probability that a randomly chosen newborn will have Down's syndrome = 0.001.
- $P(+|D)$ = the probability of a positive test, given the baby has Down's syndrome = 0.90.
- $P(+|\text{ not } D)$ = the probability of a positive test given the baby does not have Down's syndrome = 0.01

Our goal is to use Bayes' theorem to calculate the probability a baby will have Down's syndrome given that the mother tests positive, $P(D|+)$. We can figure out the different pieces of information we need by writing down Bayes' rule in the context of our diagnostic testing example:

$$P(D|+) = \frac{P(+ \mid D)P(D)}{P(+ \mid D)P(D) + P(+ \mid \text{ not } D)P(\text{not } D)} \tag{9.7}$$

Thus, we need the following:

- $P(\text{not } D) = 1 - P(D)$ (using the complementary rule) $= 0.999$
- $P(-|D) = 1 - P(+|D) = 0.10$

Plugging these values in to equation (9.7), we get:

$$P(D|+) = \frac{(0.90)(0.001)}{(0.90)(0.001) + (0.01)(0.999)} = 0.083 \text{ or } \approx 8\% \tag{9.8}$$

Although this is an order of magnitude larger than the prevalence rate in the population, the probability of the child having Down's is still low – only 8% – even with a positive test! I remember working through the math when my wife was first pregnant with our oldest child, Zoe, and concluding that we had no interest in the diagnostic test. Many diagnostic tests suffer a similar same fate when prevalence rates in the population are really low.

9.2.2 Bayes rule and problems with multiple testing

Another really useful application of Bayes rule is in understanding problems associated with conducting multiple null hypothesis tests, particularly when hypotheses do not have strong *a priori* support or when the tests have low power (i.e., probability of detecting an effect when one exists). Let's consider a scenario in which we test 1000 hypotheses. Assume:

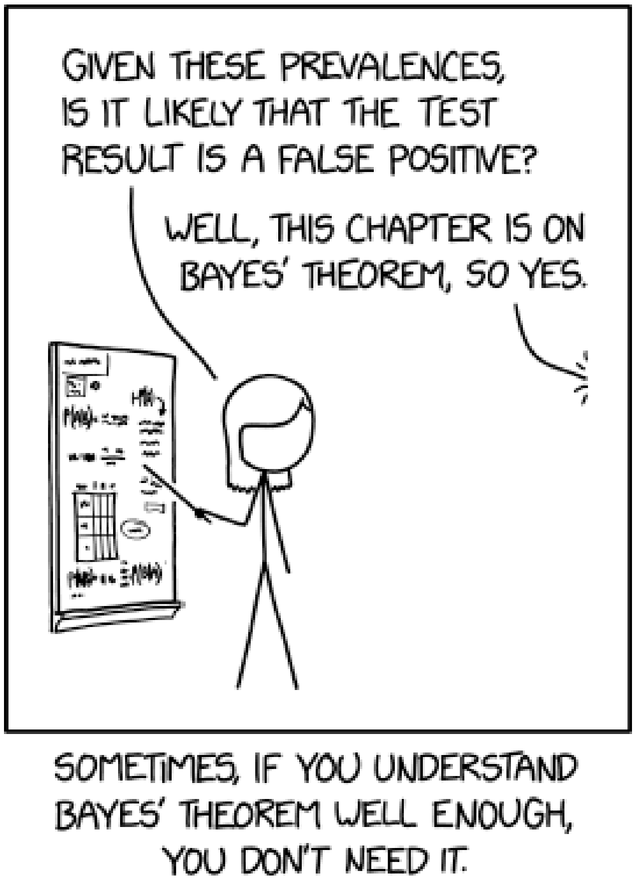

GIVEN THESE PREVALENCES, IS IT LIKELY THAT THE TEST RESULT IS A FALSE POSITIVE?
WELL, THIS CHAPTER IS ON BAYES' THEOREM, SO YES.
SOMETIMES, IF YOU UNDERSTAND BAYES' THEOREM WELL ENOUGH, YOU DON'T NEED IT.

FIGURE 9.4 As highlighted by this comic, many diagnostic tests suffer from extremely high false positive rates due to low prevalence rates in the population. Image from https://xkcd.com/2545/. CC BY-NC 2.5

- 10% of the hypotheses are truly false, $P(H_0 = true) = 0.1$.
- We have an 80% chance of rejecting the hypothesis given that it is false, $\beta = P(Reject|H_0 = false) = 0.80$.
- We use an $\alpha = 0.05$ to conduct all tests so that the probability of rejecting a hypothesis when it is true is 5%, $P(Reject|H_0 = true) = 0.05$.

Given these assumptions, we should expect to report a total of 125 significant findings, but 36% of these significant findings will be incorrect (i.e., rejections when the null hypothesis is true); 64% will be correct. We can use Bayes rule to verify this result. Let's calculate the probability the null hypothesis is false, given that we rejected it:

$$P(H_0 = false|Reject) = \frac{P(Reject|H_0 = false)P(H_0 = false)}{P(Reject|H_0 = false)P(H_0 = false) + P(Reject|H_0 = true)P(H_0 = true)} \tag{9.9}$$

$$= \frac{(0.8)(0.1)}{(0.8)(0.1) + (0.05)(0.9)} = 0.64 \tag{9.10}$$

Ioannidis (2005) used this framework to explain why studies that attempt to replicate research findings so often fail. The situation may be even worse in ecology, given that the power of most ecological studies is well below 80% and often closer to 20% (see Forstmeier, Wagenmakers, and Parker 2017 and references therein).

If we repeat the above calculations but for studies with only 20% power, we find $P(H_0 = false|Reject)$ is only around 0.30. I.e., the majority of the time that we think we are discovering something interesting, we are actually being misled.

9.3 Probability Trees

When working through compound probabilities (i.e., probabilities associated with 2 or more independent events), it can often be useful to draw a tree diagram that outlines all of the different possibilities calculated by repeatedly applying the conditional probability rule, $P(A \text{ and } B) = P(A)P(B|A)$. We show the tree diagram for the diagnostic testing problem below:

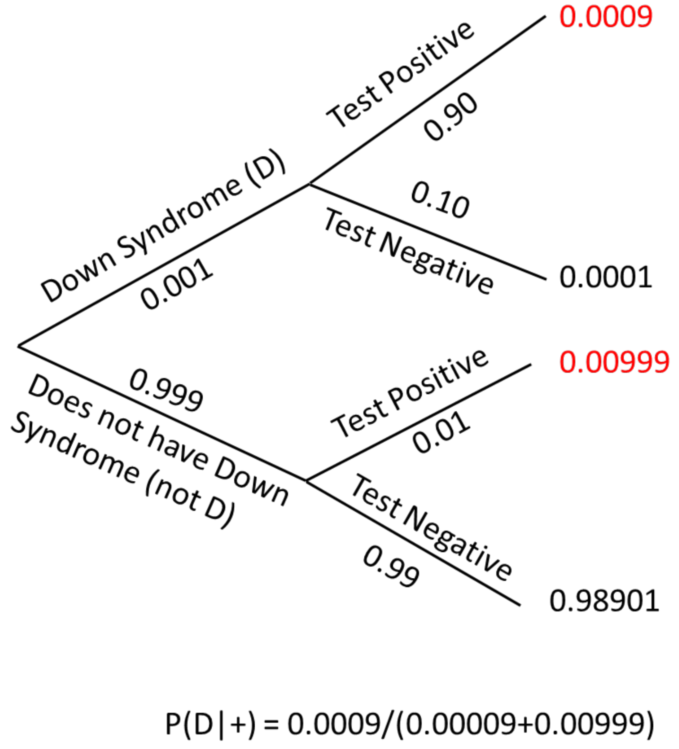


FIGURE 9.5 Probability tree for calculating the probability that a child will have Downs Syndrome given the mother tests positive. Probability trees represent all possible outcomes associated with multiple events using a branching process. Probabilities associated with each outcome can be determined by multiplying successive probabilites along each branch.

9.3.1 Just for fun: Monte Hall problem

Many of you may have seen a version of the Monte Hall problem before. One of the main reasons I like this problem is that, like the diagnostic testing problem, it can highlight how poor our intuition is at times when working with probabilities. The problem goes like this: suppose you are on a game show where the host asks you to choose from one of 3 doors. Behind one of the doors is a brand new car. Behind the other two are a goat. You are asked to choose a door (if you select the door with the car, you win the car). After you reveal your choice, the host opens one of the doors that you did not choose and shows you a goat. The game show host then asks whether you want to change your initial decision. If you are like most people, you will likely stick with your initial choice, thinking that you have a 50-50 chance of winning either way. But, it turns out that you increase your chances of winning by switching doors. If you switch, you have a 2/3 chance of winning versus 1/3 if you stick with your initial choice!

One way to understand this result is to construct a probability tree representing the initial event (choosing a door with or without a goat) and then the final outcome, conditional on your initial choice and whether you choose to change your decision or not after seeing the game show host reveal one of the 2 goats. If you initially chose a door with a goat, then you will end up with the car if you switch doors. If you initially chose the door with the car, then you will end up with a goat if you switch. The thing is, you are more likely to initially choose a door with a goat than the door with the car. So, in the end, you are best off switching doors once you can eliminate the door where the goat has been revealed.

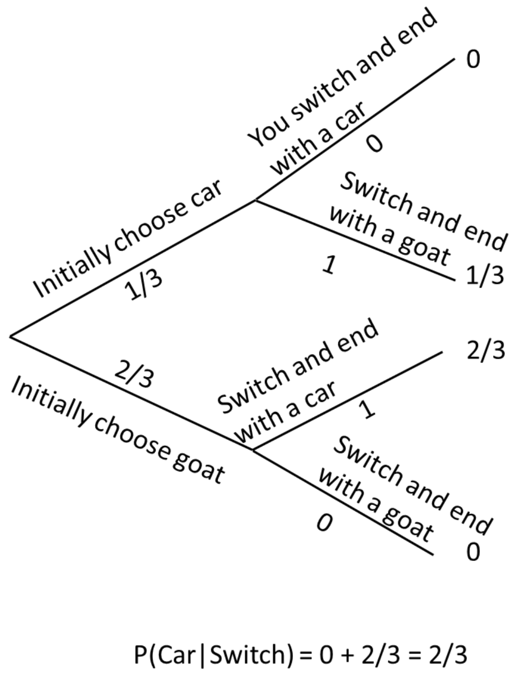

Initially choose car
1/3
You switch and end with a car
0
0
Switch and end with a goat
1
1/3
Initially choose goat
2/3
Switch and end with a car
1
2/3
Switch and end with a goat
0
0
P(Car|Switch) = 0 + 2/3 = 2/3

FIGURE 9.6 Probability tree for the Monte Hall problem.

If this result is surprising to you, you are not alone. For a really interesting read that highlights just how many (mostly overconfident male) PhD scientists not only got this answer wrong, but also choose to publicly deride Marilyn vos Savant (better known for her "Ask Marilyn" column in *Parade Magazine*) for providing the correct answer see this aticle. You might also be interested in knowing that pigeons (*Columba livia*) have performed better than humans at solving the Monte Hall problem (Herbranson and Schroeder 2010)!

9.4 Sample space, random variables, probability distributions

In frequentist statistics, probabilities are defined in terms of frequencies of events. Specifically, the probability of event A, $P(A)$, is defined as the proportion of times the event would occur in an infinite set of random trials with the same underlying conditions. To define a probability distribution, we first need to define the sample space of all possible outcomes that could occur and a random variable that maps these events to a set of real numbers. We can then map the real numbers to probabilities using a probability distribution.

A key consideration is whether we are interested in modeling the distribution of a *discrete* or *continuous* random variable. Discrete variables take on a finite or countably infinite number of values[1]. Examples include:

- age classes = (fawn, adult)
- the number of birds counted along a transect
- whether or not a moose calf survives its first year
- the number of species counted on a beach in the Netherlands

Continuous variables, by contrast, can take on any real number. Examples include:

- (lat, long) coordinates of species in an INaturalist database
- the age at which a randomly selected adult white-tailed deer dies
- mercury level (ppm) in a randomly chosen walleye from Lake Mille Lacs

9.4.1 Discrete random variables

Discrete random variables are easier to consider than continuous random variables because we can directly map events to probabilities. Let's consider a specific example. Consider sampling 2 white-tailed deer for chronic wasting disease. Assume our 2 individuals are randomly chosen from a population where the prevalence rate of chronic wasting disease is p.

In this case, the possible events are: (-, -), (-, +), (+, -), (+, +). We could define a random variable, Y = the number of positive tests. Then, our sample space for $Y = \{0, 1, 2\}$. Note, it is customary to write random variables using capital letters (Y) and realizations of random variables using lower case letters (y). Thus, Y would describe a random quantity (not yet observed) having a statistical distribution, whereas y would describe a specific observed value of the random variable Y.

We next define a probability mass function, $f(y; p)$, that maps the sample space of Y to probabilities (Table 9.1). When referring to probability mass or probability density functions that include one or more parameters, we will often use the notation, $f(variable; parameters)$ (here, $f(y; p)$). Sometimes I may forget to include the parameter(s) (or just get lazy) and use $f(y)$ instead.

TABLE 9.1 Probability distribution for the number of positive tests for chronic wasting disease in a random sample of 2 deer in a population with prevalence rate p.

Y	$f(y)$
0	(1-p)(1-p)
1	2p(1-p)
2	p^2

[1]A countably infinite set of values can be put in a 1-1 correspondence with the set of non-negative integers.

These probabilities can be easily derived by noting the probability of a positive test is p, the probability of a negative test is $1 - p$, and if events A and B are independent, then P(A and B) = P(A)P(B); see Section 9.2. This example is a special case of the binomial distribution with $n = 2$, which we will learn more about in Section 9.10.2.

9.4.2 Continuous random variables

For continuous random variables, there are an infinite number of values that Y can take on. Thus, we no longer assign probabilities to specific values of Y, but rather use probability density functions, $f(y;\theta)$, to determine the probability that Y takes on a value within an interval (a, b), $P(a \le Y \le b)$:

$$P(a \le Y \le b) = \int_a^b f(y;\theta)dy \tag{9.11}$$

You may recall from calculus that integrating $f(y;\theta)$ is equivalent to finding the area under the curve (Figure 9.7).

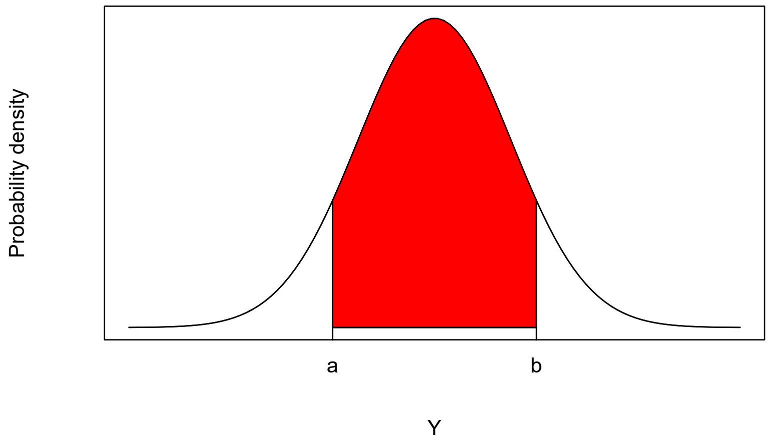


FIGURE 9.7 For continuous random variables, probabilities are defined as the areas under the curve of a probability density function.

A few other things to note about continuous random variables are:

- The probability associated with any point is 0; $P(Y = y) = 0$ for all y. Thus, we can choose to write $P(a \le Y \le b)$ or equivalently, $P(a < Y < b)$.
- $\int_{\infty}^{\infty} f(y;\theta) = 1$ (the total area under any probability density function is equal to 1).

Also, note that unlike probability mass functions, probability density functions can take on values greater than 1. To understand why this must be the case, consider representing the distribution of wing-lengths of houseflies in meters using a Normal distribution with mean = 0.00455 m and standard deviation = 0.000392 m. The range of wing-lengths is much less than 1, so the probability density function must be much greater than 1 for many values of Y to ensure that the total area under the curve is 1 (Figure 9.8).

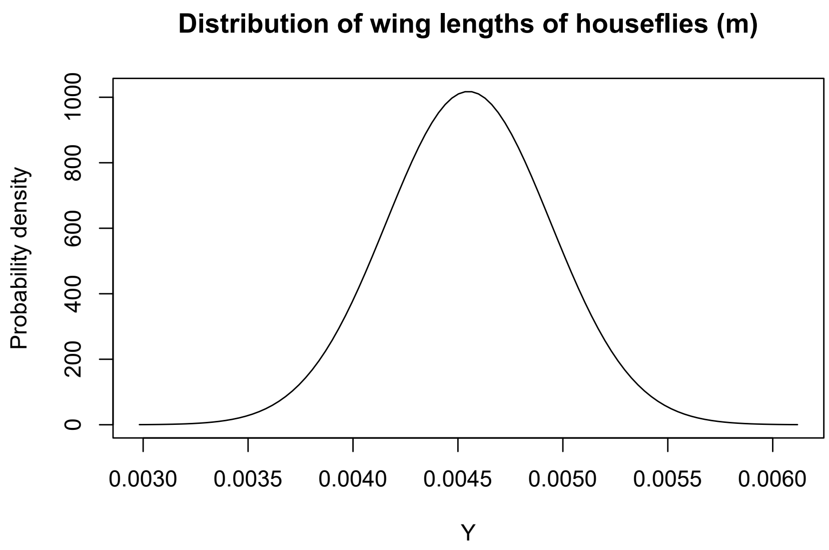


FIGURE 9.8 Distribution of wing-lengths for common houseflies in meters.

9.5 Cumulative distribution functions

We will often want to determine $F(Y) = P(Y \leq a)$ or its complement, $1 - P(Y \leq a) = P(Y > a)$; $P(Y \leq a)$ is called the cumulative distribution function (CDF). The CDF can be written as:

$$F(Y) = P(Y \leq y) = \int_{-\infty}^{x} f(y;\theta)dy$$

The cumulative distribution functions for our chronic wasting disease example assuming $p = 0.5$ and the housefly wing length example are depicted in (Figure 9.9). Note, that $F(-\infty) = 0$ and $F(\infty) = 1$ for all probability distributions.

9.6 Expected value and variance of a random variable

For Discrete random variables, we define the expected value and variance as:

- $E[Y] = \sum yf(y;\theta)$
- $Var[Y] = \sum(y - E[Y])^2 f(y;\theta)$

For continuous random variables, we define the expected value and variance as:

- $E[Y] = \int_{-\infty}^{\infty} yf(y;\theta)dy$

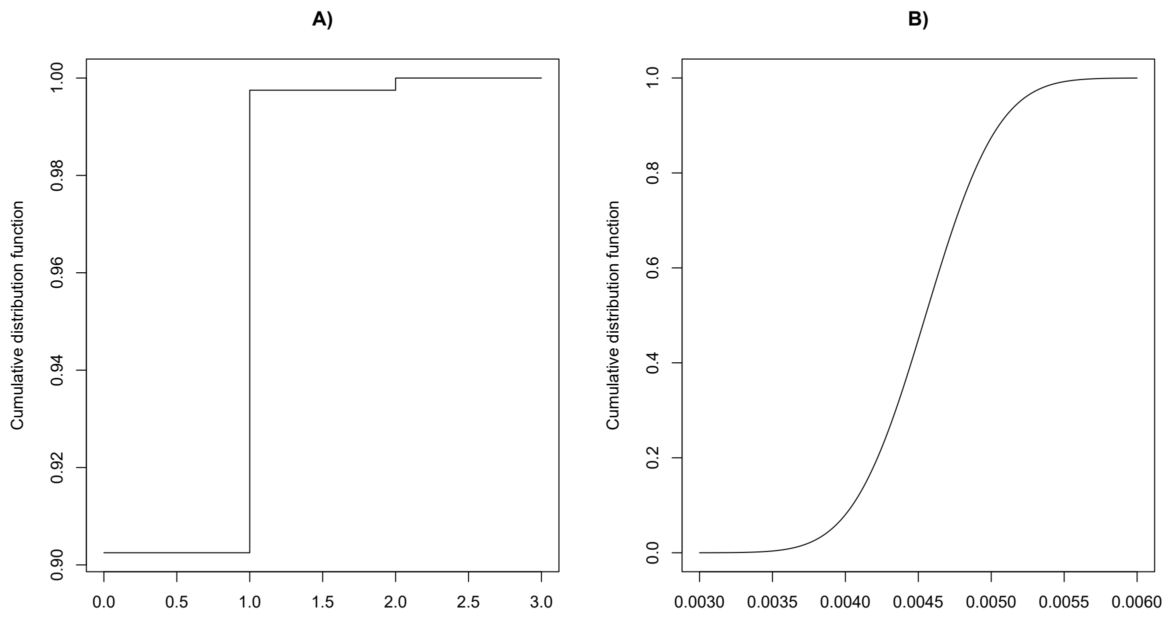


FIGURE 9.9 Cumulative distribution functions for the chronic wasting disease example (binomial distribution) and the housefly example (Normal distribution).

- $Var[Y] = \int_{-\infty}^{\infty}(y - E[Y])^2 f(y;\theta)dy$

The expected value gives the average value you would see if you generated a large number of random values from the probability distribution. Also, note that the standard deviation of a random variable is equal to $\sqrt{Var[Y]}$.

Example: Let's calculate the mean and variance of our random variable describing the number of positive tests for chronic wasting disease when sampling 2 random deer:

$$E[Y] = 0[(1-p)^2] + 1[2p(1-p)] + 2[p^2] = 2p - 2p^2 + 2p^2 = 2p$$
$$Var[Y] = (0-2p)^2[(1-p)^2] + (1-2p)^2[2p(1-p)] + (2-2p)^2[p^2] = 2p(1-p)$$

Deriving the final expression for the variance requires a lot of algebra (but, can be simplified using online calculators – e.g., here). Note, it is often easier to derive the variance using an alternative formula:

- $Var[Y] = E[Y^2] - (E[Y])^2$ where $E[Y^2] = \sum y^2 f(y;\theta)$

If you were to take a mathematical statistics course, you would likely derive this alternative expression for the variance and also derive expressions for the mean and variance associated with several common probability distributions.

9.7 Expected value and variance of sums and products

We will often need to calculate the expected value or variance of a sum or product of random variables. For example, in Section 3.12, we needed to be able to determine the variance of a sum of regression parameter estimates when calculating the uncertainty associated with pairwise differences between means for different values of a categorical variable. Similarly, in Section 5, we needed to calculate the mean and variance for a linear combination of our regression parameters in order to plot $E[Y_i|X_i]$. Here, we state three useful results for the case of 2 random variables, X and Y:

- $E[X+Y] = E[X] + E[Y]$
- $Var[X+Y] = Var[X] + Var[Y] + 2Cov[X,Y]$
- If X and Y are independent, $E[XY] = E[X]E[Y]$

9.8 Joint, marginal, and conditional distributions

There are many situations where we may be interested in considering more than 1 random variable at a time. In these situations, we may consider:

- joint probability distributions describing probabilities associated with all random variables simultaneously.
- conditional distributions describing probabilities for one or more of the random variables, given a realization of the other random variables.
- marginal distributions describing the probability of one of the random variables, which can be determined by averaging over all possible values of the other random variables.

Again, these concepts are easiest to understand when modeling discrete random variables since we can use a joint probability mass function to assign probabilities directly to unique combinations of the values associated with the two random variables. Consider the following simple example with two random variables, X and Y:

- X takes on a value of 1 when a randomly chosen site is occupied by a species of interest and 0 otherwise
- Y takes on a value of 1 when the species of interest is detected at a randomly chosen site and 0 otherwise

We will assume that the probability that a randomly chosen site is occupied by the species of interest is given by Ψ. Thus, the probability mass function for X, $f_x(x)$ is given by:

- $f_x(0) = 1 - \Psi$
- $f_x(1) = \Psi$

We will further assume that the probability that a species is detected at a randomly chosen site is equal to p if the site is occupied and 0 otherwise. Thus, the conditional probability mass function for $Y|X$ ("Y given X"), $f_y(y|x)$, is given by:

- $f_y(0|x=0) = 1$
- $f_y(1|x=0) = 0$

when $X = 0$ and

- $f_y(0|x = 1) = 1 - p$
- $f_y(1|x = 1) = p$

when $X = 1$.

Joint distribution from a conditional and marginal distribution

We can derive the *joint probability mass function* for (X, Y), $f_{x,y}(x, y)$, using the multiplicative probability rule: $f_{x,y}(x, y) = f_y(y|x)f_x(x)$:

- $f_{x,y}(0, 0) = 1 - \Psi$ (site is unoccupied and therefore the species is not detected)
- $f_{x,y}(0, 1) = 0$ (the site is unoccupied but the species is detected; equal to 0 when there are no false positives)
- $f_{x,y}(1, 0) = \Psi(1 - p)$ (the site is occupied, but we do not detect the species)
- $f_{x,y}(1, 1) = \Psi p$ (the site is occupied, and we detect the species)

We can verify that this is a valid joint probability distribution by noting that $0 \leq f(x, y) \leq 1$ for all values of x and y, and $\sum_{(x,y)} f(x, y) = 1 - \Psi + \Psi(1 - p) + \Psi p = 1$.

Marginal distribution from the joint distribution

We can derive the *marginal probability mass function* of Y, $f_y(y)$, using the total law of probability: $f_y(y) = \sum_x f_{x,y}(x, y)$:

- $f_y(0) = f_{x,y}(0, 0) + f_{x,y}(1, 0) = 1 - \Psi + \Psi(1 - p) = 1 - \Psi p$ (probability of not detecting the species)
- $f_y(1) = f_{x,y}(0, 1) + f_{x,y}(1, 1) = \Psi p$ (probability of detecting the species)

Conditional distribution from the joint and marginal distribution

We can then derive the distribution of $X|Y$, $f_x(x|y)$, using the conditional probability rule: $f_x(x|y) = \frac{f_{x,y}(x,y)}{f_y(y)}$. When $Y = 0$, we have:

- $f_x(0|y = 0) = \frac{1-\Psi}{1-\Psi p}$
- $f_x(1|y = 0) = \frac{\Psi(1-p)}{1-\Psi p}$

Importantly, this provides us with an estimate of the probability a site is occupied given that we did not detect the species at the site. The conditional distribution when $Y = 1$ is less interesting (given our assumptions, we know the site is occupied if we detect the species):

- $f_x(0|y = 1) = \frac{0}{\Psi p} = 0$
- $f_x(1|y = 1) = \frac{\Psi p}{\Psi p} = 1$

For continuous distributions, we can replace sums with integrals to specify relationships between joint, conditional, and marginal probability density functions:

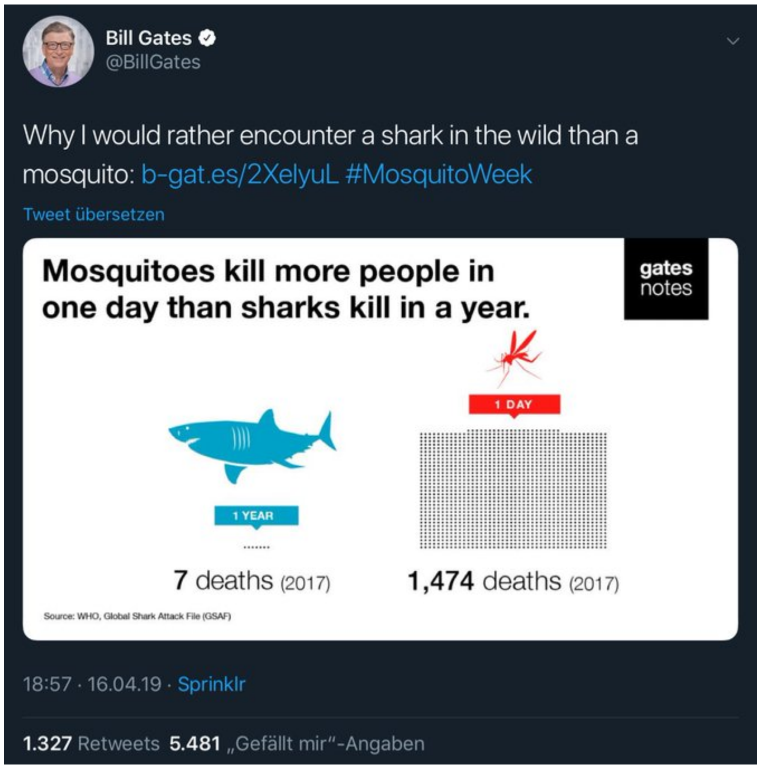
- $f_{x,y}(x, y) = f_y(y|x)f_x(x) = f_x(x|y)f_y(y)$ (joint distribution formulated in terms of a conditional and marginal distribution)
- $f_y(y) = \int_{-\infty}^{\infty} f_{x,y}(x, y)dx$ (marginal distribution derived from the joint distribution)
- $f_y(y|x) = \frac{f_{x,y}(x,y)}{f_x(x)}$ (conditional distribution specified from the joint distribution and marginal distribution)

These concepts are critical for deriving appropriate Monte Carol Markov Chain (MCMC) algorithms when fitting models in a Bayesian framework, and they are relevant to many of the models that are popular among ecologists, some of which we will explore in later chapters:

- Models with random effects (Sections 18, 19.5). We will specify these models in terms of a distribution of a response variable conditional on a set of random effects along with a distribution of the random effects.
- Models with latent (unobserved) variables, including occupancy models (MacKenzie et al. 2017). Models for Zero-inflated data (Chapter 17) can also be formulated in terms of a latent variable.

When fitting models with random effects or other latent variables in a frequentist framework, we must first derive (or approximate) the marginal distribution of the response variable after integrating or summing over the possible values of the unobserved random variables. Although we do not have to derive the marginal distribution of the response variable when fitting similar models in a Bayesian framework, doing so will often help speed up the model fitting process (Ponisio et al. 2020; Yackulic et al. 2020, see also this blog)).

Think-pair-share: explain why Bill Gate's reasoning is flawed in Figure 9.10 (thanks to Richard McElreath for tweeting this out!):



Bill Gates
@BillGates

Why I would rather encounter a shark in the wild than a mosquito: b-gat.es/2XelyuL #MosquitoWeek

Tweet übersetzen

Mosquitoes kill more people in one day than sharks kill in a year.

7 deaths (2017)

1,474 deaths (2017)

Source: WHO, Global Shark Attack File (GSAF)

18:57 · 16.04.19 · Sprinklr

1.327 Retweets 5.481 „Gefällt mir"-Angaben

FIGURE 9.10 Tweet from Bill Gates that mixes up marginal and conditional probabilities.

9.9 Probability distributions in R

This section closely follows Jack Weiss's lecture notes from his various classes. There are 4 basic probability functions associated with each probability distribution in R. These functions start with either - d, p, q, or r - and are then followed by a shortened name that identifies the distribution :

- d is for *density* and returns the value of f(y), i.e., the value of the probability density function (continuous distributions) or probability mass function (discrete distributions).
- p is for *probability*; returns a value of F(y), the cumulative distribution function.
- q is for *quantile*; returns a value from the inverse of F(y) and is also known as the quantile function.
- r is for *random*; generates a random value from the given distribution.

Let's consider these 4 functions in the context of the Normal distribution with mean 0 and standard deviation 1. The functions are named dnorm, pnorm, qnorm, rnorm and are depicted in Figure 9.11.

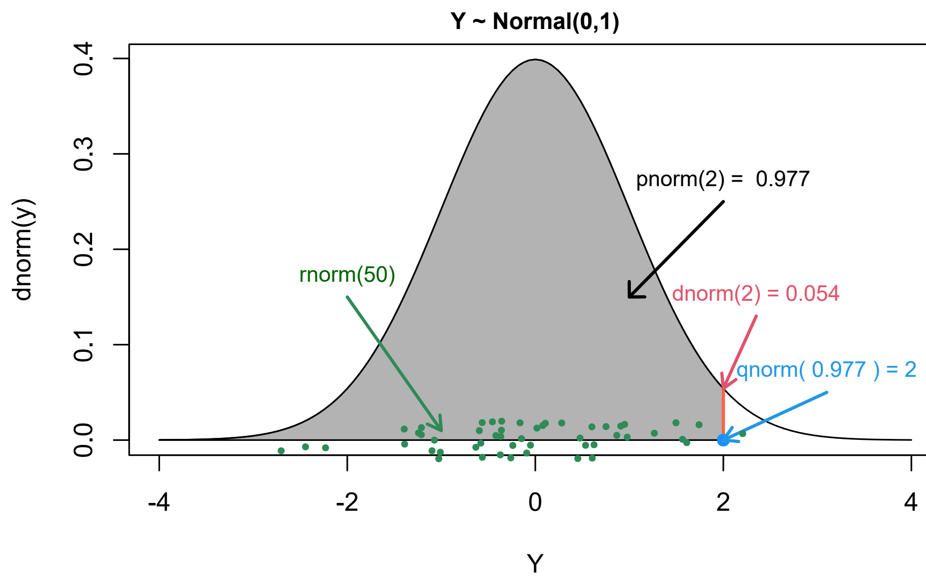


FIGURE 9.11 Figure created using code written by Jack Weiss and made available in his course notes depicting the 4 basic probability functions for working with the Normal distribution in R.

- dnorm(2)[2] or returns the value of the probability density function at $Y = 2$, i.e., $f(2)$. Because the Normal distribution is continuous, this is not a probability; rather, we can integrate the probability density function over an interval to determine the probability that Y falls in the interval. To visualize a probability density function or probability mass function, we can plot d* over a range of Y values.

[2]Note, we have ignored the other arguments to this function, which include `mean` and `sd`, allowing them to take on their default values of 0 and 1.

- pnorm(2) returns the value of the cumulative distribution function, $F(2) = P(Y \leq 2) = 0.977$ for the Normal distribution. This probability is equal to the area under the Normal density curve to the left of $Y = 2$ and is shaded gray in Figure 9.11.
- qnorm(0.977 returns the quantile function and determines the value of Y such that $P(Y \leq y) = 0.977$.
- rnorm(50) generates 50 random values from a standard normal distribution, represented by the green points in Figure 9.11.

9.10 A sampling of discrete random variables

In the next few sections, we will consider several discrete and continuous probability distributions that are commonly used to model data or to test hypotheses in statistics.

9.10.1 Bernoulli distribution: $Y \sim$ Bernoulli(p)

The Bernoulli distribution is used to model discrete random variables that can take on only two possible values, 1 or 0, with probabilities p and $1-p$, respectively. The probability mass function of a Bernoulli distribution is given by:

$$f(y;p) = P(Y = y) = p^y(1-p)^{1-y} \tag{9.12}$$

Characteristics:

- Parameter: p, often referred to as the probability of 'success' $= P(Y = 1)$, with $0 \leq p \leq 1$
- Support (i.e., range of values that Y can take on): $Y \in \{0,1\}$
- $E[Y] = \sum yf(y;p) = 0(1-p) + 1p = p$
- $Var[Y] = \sum(y - E[y])^2 f(y;p) = (0-p)^2(1-p) + (1-p)^2 p = p(1-p)$
- R does not contain functions specific to the Bernoulli distribution; rather, we have to work with functions for the Binomial distribution, which we will see next.
- JAGS and WinBugs: dbern

9.10.2 Binomial distribution $Y \sim$ Binomial(n, p)

A binomial random variable counts the the number of *successes* in a set of n independent and identical Bernoulli trials, each with probability of success, p. IF

$$\begin{aligned} X_i &\sim Bernoulli(p)(i = 1, 2, \dots n) \\ Y &= X_1 + X_2 + \dots X_n \end{aligned} \tag{9.13}$$

then, $Y \sim Binomial(n, p)$. The probability mass function for the Binomial distribution is given by:

$$P(Y = y) = \binom{n}{y} p^y (1-p)^{n-y} \tag{9.14}$$

The combinatoric term, $\binom{n}{y}$ ("n choose y"), arises from considering all possible trial sequences that give y successes regardless of when those successes occur and is given by:

$$\binom{n}{y} = \frac{n!}{y!(n-y)!} \text{ with } n! = n(n-1)(n-2)\cdots(2)1 \tag{9.15}$$

In our chronic wasting disease testing example (with $n = 2$), we could end up with 1 positive test if either the first deer or the second deer tested positive and the other tested negative. This led us to calculate $P(Y = 1) = p(1-p) + (1-p)p = \binom{2}{1}p^1(1-p)^1 = 2p(1-p)$.

Characteristics:

- Typically, the number of trials is assumed fixed and known, leaving a single parameter p with $0 \le p \le 1$. In ecology, however, there are many instances where n (usually written as N) is also an unknown parameter that we hope to estimate (e.g., population size in mark-recapture studies).
- Support: $Y \in [0, 1, \dots n]$
- $E[Y] = np$
- $Var[Y = np(1-p)$
- R: dbinom, pbinom, qbinom, rbinom with arguments `size` $= n$ and `prob` $= p$
- JAGS: dbin(p, n)

Uses: a Bernoulli or binomial distribution is often used to model presence/absence and detection/non-detection data from wildlife surveys. It can also be used to model survival (Y/N), the number of eggs in a clutch that will hatch (or number of chicks that will fledge), etc.

Example: Raymond Felton, a point guard on the University of North Carolina's 2005 National Championship team, shot free throws at a 70% success rate during the 2004-2005 season. If we assume each of his free throw attempts can be modeled as independent trials, what is the probability that he would hit **at least** 4 out of 6 free throws in the 2005 Championship Game (he hit 5)?[3]

There are several ways to compute this probability in R:

1. Using $P(X \ge 4) = P(X = 4) + P(X = 5) + P(X = 6)$ and the expression for the probability mass function:

```
choose(6,4)*(0.7)^4*(0.3)^2 +
  choose(6,5)*(0.7)^5*(0.3) +
  0.7^6
```

```
[1] 0.74431
```

2. Using R's built in probability mass function, `dbinom`:

```
sum(dbinom(4:6, size = 6, p = 0.70))
```

```
## [1] 0.74431
```

3. Using R's built in cumulative distribution function, `pbinom`, and recognizing that $P(Y \ge 4) = 1 - P(Y \le 3)$.

[3] I became a huge University of North Carolina basketball fan after going to graduate school in Chapel Hill and attending nearly every home basketball game for 6 years. My son is named after Raymond Felton, the point guard from the 2005 National Championship team and my youngest daughter is named Carolina Anne. I had plane tickets to join friends of mine in Chapel Hill to watch the 2005 Championship game, but ultimately decided not to go. My oldest daughter, Zoe, was only 2 weeks old. I had also witnessed 4 final four losses during my time in North Carolina. Ultimately, I decided to watch the game with my wife and daughter at home.

```
1- pbinom(3, size = 6, p = 0.7)
```

```
[1] 0.74431
```

4. We can also use `pbinom` with the argument `lower.tail = FALSE`, which will give us $P(Y > y) = P(Y \geq y+1)$ for discrete random variables.

```
pbinom(3, size = 6, p = 0.7, lower.tail= FALSE)
```

```
## [1] 0.74431
```

9.10.3 Geometric distribution

The Geometric distribution arises from considering the number of failures until you get your first success in a set of Bernoulli random trials. To derive the probability mass function for the Geometric distribution, note that we have to have y failures (each with probability $1-p$) followed by a single success with probability p:

$$f(y;p) = P(Y = y) = (1-p)^y p \tag{9.16}$$

- Parameter: p (probability of success), with $0 \leq p \leq 1$
- Support: $Y \in [0, 1, 2, \ldots, \infty]$
- $E[Y] = \frac{1}{p} - 1$
- $Var[Y] = \frac{(1-p)}{p^2}$
- R: *geom with parameter `prob` $= p$
- JAGS: dnegbin(p, 1)

Other notes:

Sometimes the Geometric distribution is defined as the number of trials until the first success. In this case, the probability mass function is defined as:

$$f(y) = P(Y = y) = (1-p)^{(y-1)} p \tag{9.17}$$

with support: $Y \in [1, 2, \ldots, \infty]$ and $E[Y] = \frac{1}{p}$, $Var[Y] = \frac{(1-p)}{p^2}$

9.10.4 Multinomial distribution $Y \sim$ Multinomial$(n, p_1, p_2, \ldots, p_k)$

The multinomial distribution is a generalization of the binomial distribution, allowing for k possible outcomes with probabilities $(p_1, p_2, \ldots, p_k;\ p_k = 1 - \sum_{i-1}^{k-1} p_i)$[4]. Let n be the number of independent trials, $Y = (Y_1, Y_2, \ldots, Y_k)$ be a multivariate random variable recording the number of events in each category, and $(n_1, n_2, \ldots, n_k)$ give the observed number of events in each category, then the probability mass function is defined as:

$P((Y_1, Y_2, \ldots, Y_k) = (n_1, n_2, \ldots, n_k)) = \frac{n!}{n_1! n_2! \cdots n_k!} p_1^{n_1} p_2^{n_2} \cdots p_k^{n_k}$

Characteristics:

[4]Once we know $k-1$ of the p_i's, we can determine the last one since they must sum to 1.

- Parameters: $p_1, p_2, \ldots, p_k$ with $0 \leq p_i \leq 1$ for all i. As with the binomial distribution, n can also be viewed as an unknown parameter in various mark-recapture and wildlife abundance estimation frameworks (e.g., Otis et al. 1978; Pollock et al. 1990).
- R: dmultinom, pmultinom, qmultinom, rmultinom with parameters `size` $= n$ and `prob` $= p$.
- JAGS: dmulti(p,n)

Uses:

The multinomial distribution is often appropriate for modeling a categorical response variable with more than 2 unordered categories. The multinomial distribution frequently arises when modeling capture histories in mark-recapture studies (e.g., Otis et al. 1978; Pollock et al. 1990).

9.10.5 Poisson distribution: $Y \sim Poisson(\lambda)$

The Poisson distribution is often used to model counts of events randomly distributed in space or time, occurring at a rate, λ. Let $Y_t =$ the number of events occurring in a time interval of length t. The probability mass function in this case is given by:

$$P(Y_t = y) = \frac{\exp(-\lambda t)(\lambda t)^y}{y!} \tag{9.18}$$

Similarly, let Y_A the number of events occurring in a spatial unit of area A. Then,

$$P(Y_A = y) = \frac{\exp(-\lambda A)(\lambda A)^y}{y!} \tag{9.19}$$

Oftentimes, observations are recorded over constant time intervals or spatial units all of the same size. In this case, the probability mass function would be defined simply as:

$$P(Y = y) = \frac{\exp(-\lambda)(\lambda)^y}{y!} \tag{9.20}$$

Characteristics:

- Parameter: $\lambda \geq 0$
- Support: $Y \in [0, 1, 2, \ldots, \infty]$
- $E[Y] = Var[Y] = \lambda$
- R: dpois, ppois, qpois, and rpois with parameter `lambda`.
- JAGS: dpois(lambda)

Uses: the Poisson distribution is often used as a null model in spatial statistics. Although it can be used as a general model for count data, most ecological data are “overdispersed”, meaning that $Var[Y] \gg E[Y]$, and thus, the Poisson distribution often does not fit well.

Example: suppose a flower bed receives, on average, 10 visits by Monarchs a day in mid-August. Further, assume the number of daily visits closely follows a Poisson distribution. What is the probability of seeing 15 monarchs on a random visit to the garden?

```
dpois(15, lambda=10)
```

```
## [1] 0.03471807
```

```
10^15*exp(-10)/(factorial(15))
```

```
## [1] 0.03471807
```

9.10.6 Negative Binomial

9.10.6.1 Classic parameterization $Y \sim$ NegBin(p, r)

There are 2 different parameterizations of the Negative Binomial distribution, which we will refer to as the "classic" and "ecological" parameterizations. Consider again a set of independent Bernoulli trials. Let $Y =$ number of failures before the r^{th} success. To derive the probability mass function for the classic parameterization of the Negative Binomial distribution, consider:

- We will have a total of $n = y + r$ trials
- The last trial will be a success (with probability p)
- The preceding $y + r - 1$ trials will have had y failures and can be described by the binomial probability mass function with $n = y + r - 1$ and p.

$P(Y = y) = \binom{y+r-1}{y}p^{r-1}(1-p)^y p = \binom{y+r-1}{y}p^r(1-p)^y$

Characteristics:

- Parameters: r must be integer valued and > 0, $0 \le p \le 1$
- Support: $Y \in [0, 1, 2, \ldots, \infty]$
- $E[Y] = \frac{r(1-p)}{p}$
- $Var[Y] = \frac{r(1-p)}{p^2}$
- In R: *nbinom, with parameters (`prob` = p, `size` = n)

9.10.6.2 Ecological parameterization $Y \sim$ NegBin(μ, θ)

To derive the ecological parameterization of the Negative Binomial distribution, we express p in terms of the mean, μ and r:

$$\begin{aligned} \mu = \frac{r(1-p)}{p} \Rightarrow p = \frac{r}{\mu + r} and \\ 1 - p = \frac{\mu}{\mu + r} \end{aligned} \tag{9.21}$$

Plugging these values in to $f(y)$ and changing r to θ, we get:

$P(Y = y) = \binom{y+\theta-1}{y}\left(\frac{\theta}{\mu+\theta}\right)^\theta\left(\frac{\mu}{\mu+\theta}\right)^y$

Then, let $\theta =$ dispersion parameter take on any positive number (not just integers as in the original parameterization).

Characteristics:

- Parameters: $\mu \ge 0$, $\theta > 0$
- $E[Y] = \mu$

- $Var[Y] = \mu + \frac{\mu^2}{\theta}$
- in R: *nbinom, with parameters `mu` $= \mu$, `size` $= \theta$
- JAGS: dnegbin with parameters (p, θ)

Uses: the Negative Binomial model is often used to model overdispersed count data where the variance is greater than the mean. Note that as $\theta \rightarrow \infty$, the Negative Binomial distribution approaches the Poisson. In my experience, the Negative Binomial model almost always provides a better fit to ecological count data than the Poisson.

Other notes: if $Y_i \sim \text{Poisson}(\lambda_i)$, with $\lambda_i \sim \text{Gamma}(\alpha, \beta)$, then Y_i will have a negative binomial distribution.

9.11 A sampling of continuous probability distributions

9.11.1 Normal Distribution $Y \sim N(\mu, \sigma^2)$

The probability density function for the normal distribution is given by:

$$f(y; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right) \tag{9.22}$$

Characteristics:

- Parameters: $\mu \in (-\infty, \infty)$ and $\sigma > 0$
- $E[Y] = \mu$
- $Var[Y] = \sigma^2$.
- Support: $Y \in (-\infty, \infty)$.
- R: dnorm, pnorm, qnorm, rnorm; these functions have arguments for the `mean` and standard deviation, `sd`, of the Normal distribution
- JAGS: also has a dnorm function, but it is specified in terms of the precision, $\tau = 1/\sigma^2$, rather than standard deviation

Uses: the Normal distribution forms the backbone of linear regression. The Central Limit Theorem tells us that as n gets large, $\bar{y}$ and $\sum y$ become Normally distributed. Thus, the Normal distribution serves as a useful model for response variables influenced by a large number of factors that act in an additive way. The Normal distribution is also often used to define prior distributions in Bayesian analysis.

Other notes: one of the really unique characteristics of the Normal distribution is that the mean and variance are independent (knowing the mean tells us nothing about the variance and vice versa). By contrast, most other probability distributions have parameters that influence both the mean and variance (e.g., the mean and variance of a binomial random variable are np and $np(1-p)$). Thus, although it is possible to define linear regression models in terms of a model for the mean + and independent error term, this will not be the case for other probability distributions. The error (or variability about the mean) will usually depend on the mean itself! We will revisit this point when we learn how to write expressions for regression models formulated using statistical distributions other than the Normal distribution.

Special cases: setting $\mu = 0$ and $\sigma = 1$ gives us the **standard normal distribution** $Y \sim N(0, 1)$.

9.11.2 log-normal Distribution: $Y \sim$ Lognormal(μ, σ)

Y has a log-normal distribution of if $\log(Y) \sim N(\mu, \sigma^2)$. The probability density function for the log-normal distribution is given by:

$$f(y;\mu,\sigma) = \frac{1}{y\sqrt{2\pi}\sigma}\exp\left(-\frac{(log(y)-\mu)^2}{2\sigma^2}\right) \tag{9.23}$$

Characteristics:

- Parameters: μ and $\sigma > 0$ are the mean and variance of $\log(Y)$ not Y
- Support: $Y \in (0, \infty)$
- $E[Y] = \exp(\mu + 1/2\sigma^2)$
- $Var(Y) = \exp(2\mu + \sigma^2)(\exp(\sigma^2) - 1)$
- $Var(Y) = kE[Y]^2$
- R: dlnorm, plnorm, qlnorm, rlnorm with parameters `meanlog` and `sdlog`
- JAGS: dlnorm(meanlog, 1/varlog)

Uses: whereas the Normal distribution provides a reasonable model for response variables formed by adding many different factors together, the lognormal distribution serves as a useful model for response variables formed by multiplying many independent factors together. This follows again from the Central Limit Theorem since $log(Y_1Y_2\cdots Y_n) = log(Y_1) + log(Y_2) + \ldots log(Y_n)$, and hence, as n gets large, $\sum log(y)/n)$ will converge in distribution to a Normal. The lognormal distribution arises naturally in the case of stochastic population models formulated in discrete time: $N_t = \lambda_t N_{t-1}$, where N_t is the population size at time t (Morris et al. 1999).

9.11.3 Continuous Uniform distribution: $Y \sim U(a, b)$

If observations are equally likely within an interval (a, b):

$$f(y;a,b) = \frac{1}{b-a} \tag{9.24}$$

Characteristics:

- Parameters: (a, b) which together defined the support of $Y \in (a, b)$
- $E[Y] = (a+b)/2$
- $Var[Y] = \sqrt{(b-a)^2/12}$
- R: *unif with `min` = a, `max` = b
- JAGS: dunif(lower, upper)

Uses: the uniform distribution is often used as a model of ignorance for prior distributions. Also, p-values across replicated experiments in which the null hypothesis is true should follow a uniform distribution.

9.11.4 Beta Distribution: $Y \sim$ Beta(α, β)

The Beta distribution is unique in that it has support only on the (0,1) interval, which makes it useful for modeling probabilities (or as a prior distribution for p when modeling binary data). The probability density function for the Beta distribution is given by:

$$f(y;\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} y^{\alpha-1}(1-y)^{\beta-1} \tag{9.25}$$

This is the first time we have seen the gamma function (Γ), which is defined as:

$$\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt \tag{9.26}$$

The gamma function generalizes factorials ($x!$) to non-integer values; for any integer, n, $\Gamma(n) = (n-1)!$ In R, the gamma function can be executed using `gamma(n)`, e.g., `gamma(4)` $= 3! = 3*2*1 = 6$:

```
gamma(4)
```

```
## [1] 6
```

Characteristics:

- Parameters: $\alpha > 0$ and $\beta > 0$

- Support: $Y \in (0,1)$
- $E[Y] = \frac{\alpha}{\alpha+\beta}$
- $Var[Y] = \frac{\alpha}{(\alpha+\beta)^2(\alpha+\beta+1)}$
- R: *beta with `shape1` $= \alpha$ and `shape2` $= \beta$
- JAGS: dbeta

Uses: the Beta distribution is sometimes used to model probabilities or proportions (in cases where the number of trials is not known). It is also frequently used as a prior distribution for p when modeling binary data.

Special cases: when $\alpha = \beta = 1$, we get a continuous uniform(0,1) distribution.

9.11.5 Exponential: $Y \sim \text{Exp}(\lambda)$

If we have a random (Poisson) event process with rate parameter, λ, then the wait time between events follows an exponential distribution. The probability density function for the exponential distribution is given by:

$$f(y;\lambda) = \lambda\exp(-\lambda y) \tag{9.27}$$

Characteristics:

- Parameter: $\lambda > 0$
- Support: $Y \in (0,\infty)$
- $E[Y] = \frac{1}{\lambda}$
- $Var[Y] = \frac{1}{\lambda^2}$
- R: *exp with `rate` $= \lambda$
- JAGS: dexp

Uses: often used to model right-skewed distributions, particularly in time-to-event models.

9.11.6 Gamma Distribution: $Y \sim$ Gamma(α, β)

If we have a random (Poisson) event process with rate parameter, λ, then the waiting time before α events occur follows a Gamma distribution. The probability density function for the Gamma distribution is given by:

$$f(y; \alpha, \beta) = \frac{1}{\Gamma(\alpha)} y^{\alpha-1} \beta^{\alpha} \exp(-\beta y) \tag{9.28}$$

- Parameters: $\alpha > 0$ and $\beta > 0$
- Support: $Y \in (0, \infty)$
- $E[Y] = \frac{\alpha}{\beta}$
- $Var[Y] = \frac{\alpha}{\beta^2}$
- R: *gamma with `shape` $= \alpha$ and `rate` $= \beta$, or `shape` $= \alpha$ and `scale` $= 1/\beta$
- JAGS: dgamma(alpha, beta)

Uses: the Gamma distribution can be used to model non-negative continuous random variables. In movement ecology, the Gamma distribution is often used to model step-lengths connecting consecutive telemetry locations Δt units apart (e.g., Signer, Fieberg, and Avgar 2019; Fieberg et al. 2021).

Special cases: when $\alpha = 1$, we get an exponential distribution with rate parameter β. A Gamma distribution with $\alpha = k/2$ and $\beta = 1/2$ is a chi-squared distribution with k degrees of freedom, χ^2_k.

9.12 Some probability distributions used for hypotheses testing

9.12.1 χ^2 distribution: $Y \sim \chi^2_{df=k}$

The χ^2 distribution is often encountered in introductory statistics classes where it is used for goodness-of-fit testing for a single categorical variable and for constructing tests of association between two categorical variables (Agresti 2018). We will also encounter the χ^2 distribution when comparing nested models using likelihood ratio tests. Lastly, the χ^2 distribution is used to construct confidence intervals for the variance of Normally distributed random variables since $(n-1)s^2/\sigma^2$ will follow a χ^2 distribution with $n-1$ degrees of freedom in this case.

The probability density function for the χ^2 distribution with k degrees of freedom is given by:

$$f(y; k) = \frac{y^{k/2-1} \exp(-y/2)}{2^{k/2}\Gamma(k/2)} \tag{9.29}$$

Characteristics:

- Parameter: $k > 0$
- Support: $Y \in (0, \infty)$
- $E[Y] = k$
- $Var[Y] = 2k$
- R: *chisq with `df` $= k$
- JAGS: dchisqr(k)

9.12.2 Student's t distribution $Y \sim t_{df=k}$

We encountered the t-distribution in Section 1, which we used to test hypotheses involving individual coefficients in our regression models. The probability density function for the student t distribution with k degrees of freedom is given by:

$$f(y;k) = \frac{\Gamma(\frac{k+1}{2})}{\Gamma(\frac{k}{2})\sqrt{k\pi}}(1+\frac{y^2}{k})^{-(\frac{k+1}{2})} \tag{9.30}$$

Characteristics:

- Parameter: $k > 0$; can also include a non-centrality parameter (e.g., used when calculating power of hypothesis tests)
- Support: $Y \in (-\infty, \infty)$
- $E[Y] = 0$
- $Var[Y] = k/(k-2)$ for $k > 2$
- R: *t with `df` $= k$, `ncp` $=$ non-centrality parameter
- JAGS: dt(mu, tau, k), with `tau=1`, then `mu` is equal to the non-centrality parameter in R and `k` is the degrees of freedom

9.12.3 F distribution, $Y \sim F_{df_1=k_1, df_2=k_2}$

We saw the F-distribution when testing whether multiple coefficients associated with a categorical predictor were simultaneously 0 in Section 3.10. The probability density function for the F-distribution with k_1 and k_2 degrees of freedom is given by:

$$f(y;k_1,k_2) = \frac{\Gamma((k_1+k_2)/2)(k_1/k_2)^{k_1/2}y^{k_1/2-1}}{\Gamma(k_1/2)\Gamma(k_2/2)[(k_1/k_2)y+1]^{(k_1+k_2)/2}} \tag{9.31}$$

Characteristics:

- Parameter: $k_1, k_2 > 0$; can also include a non-centrality parameter (e.g., used when calculating power of hypothesis tests)
- Support: $Y \in (0, \infty)$
- $E[Y] = \frac{k_2}{k_2-2}, k_2 > 0$
- $Var[Y] = 2\left(\frac{k_2}{k_2-2}\right)^2 \frac{(k_1+k_2-2)}{k_1(k_2-4)}, k_2 > 4$
- R: *f with `df1` $= k_1$, `df2` $= k_2$, `ncp` $=$ non-centrality parameter
- JAGS: df

9.13 Choosing an appropriate distribution

How do we choose an appropriate distribution for our data? This may seem like a difficult task after having just been introduced to so many new statistical distributions. In reality, you can usually whittle down your choices to a few different distributions by just considering the range of observations and the support of available distributions. For example:

- If your data are continuous and can take on any value between $(-\infty, \infty)$, or if the range is small relative to the mean, then the Normal distribution is usually a good default.
- If your data are continuous, non-negative, and right-skewed, then a log-normal or gamma distribution is likely appropriate; if your data also contain zeros, then you may need to consider a mixture distribution (see Chapter 17).
- If you are modeling binary data that can take on only 1 of 2 values (e.g., alive/dead, present/absent, infected/not-infected), then a Bernoulli distribution is appropriate.
- If you have a count associated with a fixed number of "trials" (e.g., Y out of n individuals are alive, infected, etc.), then a Binomial distribution is a good default.
- If you have counts associated with temporal or spatial units, then you might consider the Poisson or Negative Binomial distributions (or zero-inflated versions that we will see later).
- If you are modeling time-to-event data (e.g., as when conducting a survival analysis), then you might consider the exponential distribution, gamma distribution, or a generalization of the gamma called the Weibull distribution.
- If you have circular data (e.g., your response is an angle or a time between 0 and 24 hours or 0 and 365 days), then you would likely want to consider specific distributions that allow for periodicities (e.g., 0 is the same as 2π; 0:00 is equivalent to 24:00 hours). Examples include the von Mises distribution or various "wrapped distributions" like the wrapped Normal distribution (Pewsey, Neuhäuser, and Ruxton 2013).

9.14 Summary of statistical distributions

Table 9.2 briefly details most of the random variables discussed in this chapter and was modified from https://github.com/proback/BeyondMLR/blob/master/03-Distribution-Theory.Rmd.

In a few places, we have highlighted connections among different statistical distributions, but there are many other connections that we did not mention (e.g., see Figure 9.12).

TABLE 9.2 Review of mentioned random variables.

Distribution Name	pmf / pdf	Parameters	Possible Y Values	Description
Bernouli	$p^Y(1-p)^{1-Y}$	p	$0, 1$	Success or failure
Binomial	$\binom{n}{y}p^y(1-p)^{n-y}$	$p,\ n$	$0, 1, \ldots, n$	Number of successes after n trials
Geometric	$(1-p)^y p$	p	$0, 1, \ldots, \infty$	Number of failures until the first success
Multinomial	$\frac{n!}{n_1!n_2!\cdots n_k!}p_1^{n_1}p_2^{n_2}\cdots p_k^{n_k}$	$p_i, i = 1, 2, \ldots k$	$0, 1, \ldots, n$ for each Y_i	Number of successes associated with each of k categories
Poisson	$e^{-\lambda}\lambda^y\big/y!$	λ	$0, 1, \ldots, \infty$	Number of events in a fixed time interval or spatial unit
Negative Binomial (classic)	$\binom{y+r-1}{y}(p)^r(1-p)^y$	r, p	$0, 1, \ldots, \infty$	Number of failures before r successes
Negative Binomial (ecological)	$\binom{y+\theta-1}{y}\left(\frac{\theta}{\mu+\theta}\right)^\theta\left(\frac{\mu}{\mu+\theta}\right)^y$	μ, θ	$0, 1, \ldots, \infty$	A model for overdispersed counts
Normal	$\dfrac{e^{-(y-\mu)^2/(2\sigma^2)}}{\sqrt{2\pi\sigma^2}}$	$\mu,\ \sigma$	$(-\infty, \infty)$	Model for when a large number of factors act in an additive way
log-Normal	$\dfrac{e^{-(log(y)-\mu)^2/(2\sigma^2)}}{y\sqrt{2\pi}\sigma}$	$\mu,\ \sigma$	$(0, \infty)$	Model for when a large number of factors act in an multiplicative way
Uniform	$\frac{1}{b-a}$	a, b	$[a, b]$	Useful for specifying vague priors
Beta	$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}y^{\alpha-1}(1-y)^{\beta-1}$	$\alpha,\ \beta$	$(0, 1)$	Useful for modeling probabilities
Exponential	$\lambda e^{-\lambda y}$	λ	$(0, \infty)$	Wait time for one event in a Poisson process
Gamma	$\dfrac{\lambda^r}{\Gamma(r)}y^{r-1}e^{-\lambda y}$	$\alpha,\ \beta$	$(0, \infty)$	Wait time for r events in a Poisson process
χ^2	$\frac{y^{k/2-1}e^{(-y/2)}}{2^{k/2}\Gamma(k/2)}$	k	$(0, \infty)$	Distribution for goodness-of-fit tests, likelihood ratio test
Students t	$\frac{\Gamma(\frac{k+1}{2})}{\Gamma(\frac{k}{2})\sqrt{k\pi}}(1+\frac{y^2}{k})^{-(\frac{k+1}{2})}$	k	$(-\infty, \infty)$	Distribution for testing hypotheses involving means, regression coefficients
F	$\frac{\Gamma((k_1+k_2)/2)(k_1/k_2)^{k_1/2}y^{k_1/2-1}}{\Gamma(k_1/2)\Gamma(k_2/2)[(k_1/k_2)y+1]^{(k_1+k_2)/2}}$	k_1, k_2	$(0, \infty)$	Distribution for testing hypotheses involving regression coefficients

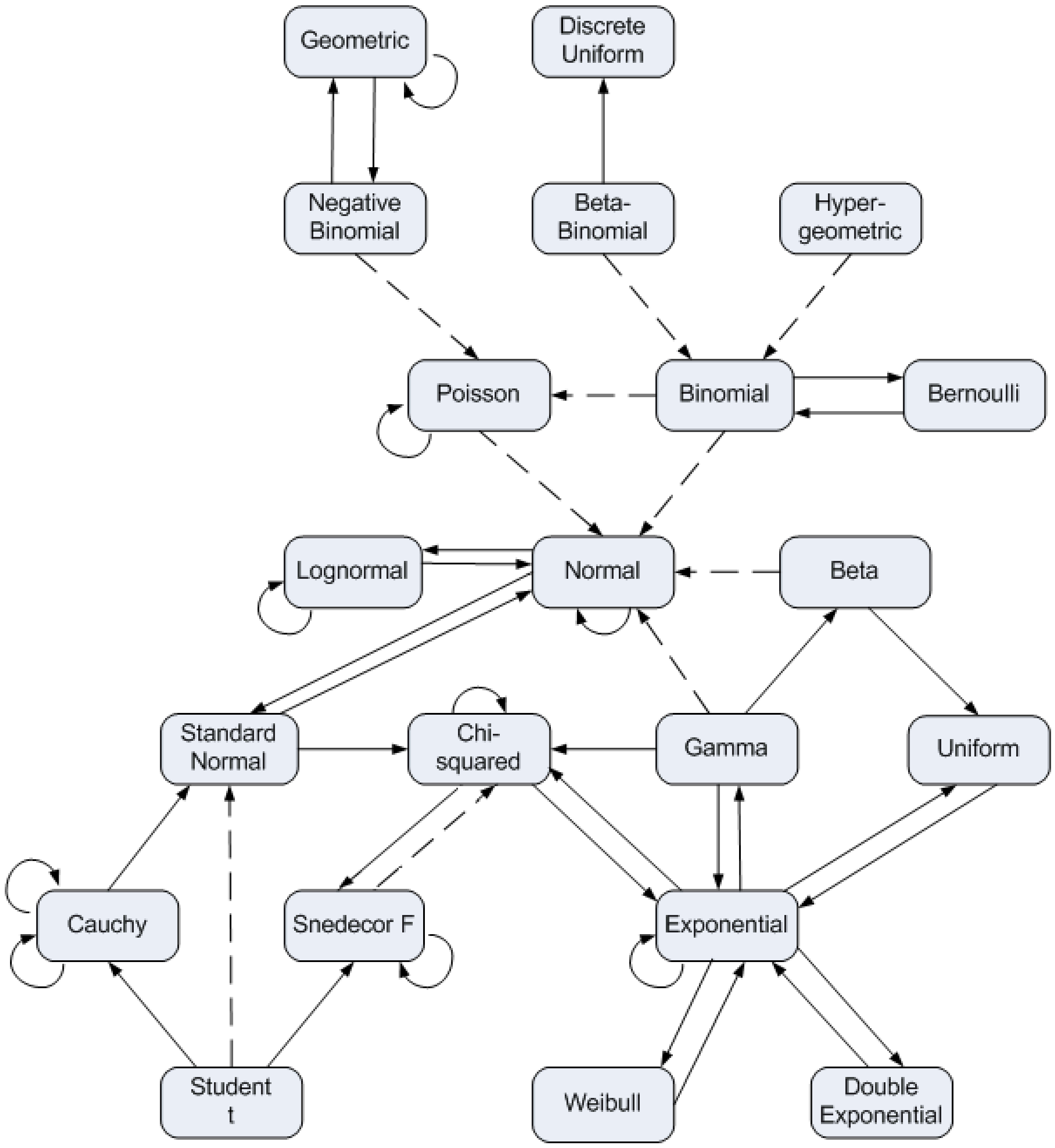


FIGURE 9.12 Relationships among some of univariate probability distributions. Figure from https://www.johndcook.com/blog/distribution_chart/.

10

Maximum likelihood

Learning objectives

1. Understand how to use maximum likelihood to estimate parameters in statistical models.
2. Understand how to create large-sample confidence intervals for parameters estimated using maximum likelihood.
3. Be able to construct large-sample confidence intervals for functions of parameters using the delta method.
4. Be able to conduct likelihood ratio tests for nested models and understand that profile-likelihood intervals can be formed by inverting the likelihood ratio test.

Credit: much of this material in this section was originally developed by Jack Weiss and presented online as part of his Statistical Ecology courses at the University of North Carolina.

10.1 R packages

We begin by loading a few packages upfront:

```
library(kableExtra) # for creating tables
options(kableExtra.html.bsTable = T)
library(dplyr) # for data wrangling
library(ggplot2) # for plotting
library(patchwork) # for creating multi-panel plots
```

In addition, we will use data and functions from the following packages:

- `Data4Ecologists` for `beargrowth` and `slugs` data sets
- `emdbook` (B. Bolker 2023) for the `deltavar` function used to calculate standard errors via the delta method.

10.2 Parameter estimation

In the previous section, we learned about a number of statistical distributions, described by a small set of parameters. We might now ask:

- How do we determine appropriate values of the parameters?
- How do we incorporate the effects of covariates on these parameters and thus, the distributions of our data?
- How do we calculate confidence intervals or conduct hypotheses tests associated with these parameters?

Although there are alternative approaches (e.g., least squares, methods of moments), we will focus on two primary methods for estimating parameters:

- Maximum likelihood estimators
- Bayesian inference

Most statistical software packages use maximum likelihood to estimate parameters, including for example, the `glm` function in R for fitting logistic and Poisson regression models that will be the subject of later Sections of this book. Although users do not necessarily need to know the details of how maximum likelihood works in order to apply these methods, this knowledge will be helpful for understanding the inferential framework associated with commonly used statistical models fit in a frequentist framework (e.g., SEs and confidence intervals usually rely on an assumption that the sampling distribution is Normal, which will be reasonable for parameters estimated using maximum likelihood as long as our sample sizes are large or the data are Normally distributed). Typically, numerical analysis methods are needed to estimate parameters using maximum likelihood. You will be introduced to these methods, which may seem like a bit of a diversion. Yet, this knowledge will allow you to formulate and fit custom models that may otherwise not be available in current software. Lastly, likelihoods are also a key component of Bayesian inferential techniques. Thus, this section will help to prepare you for your journey into Bayesian inference.

To help motivate this section, consider the weight-at-age relationship for female black bears monitored in Minnesota depicted in Figure 10.1, constructed using the code below.

```r
library(Data4Ecologists)
theme_set(theme_bw())
data(beargrowth)
beargrowth <- beargrowth %>% filter(sex=="Female")
ggplot(beargrowth, aes(age, wtkg)) +
  geom_point() + geom_smooth() + ylab("Weight (kg)") + xlab("Age (years)")
```

We see that the pattern is non-linear and also that the variance of the observations increases with the age of the bears. We could try to model these data using polynomials or splines (see Section 4) combined with a variance model from Section 5. However, there are also many non-linear functions that have "built-in" asymptotes, and these are frequently used to model growth rates. Here, we will consider the von Bertalanffy growth function (Beverton and Holt 2012):

$$E[Y_i] = L_\infty(1 - e^{-K(Age_i - t_0)}) \tag{10.1}$$

An advantage of using a growth model like the von Bertalanffy growth curve versus a phenomenological model (e.g., using polynomials or splines) is that the model's parameters often have useful biological interpretations. For example, L_∞ represents the asymptotic weight of individuals in the population.

Although there are R packages (e.g., `FSA`; Ogle et al. 2021) that allow one to fit the von Bertalanffy growth curve to data, by writing our own code and using built-in optimizers, we will be able to tailor our model to specific features of our data (e.g., we can relax the assumption that the variance about the mean growth curve is constant). We could relax the constant variance assumption using a GLS-style model or by using an alternative probability distribution (e.g., log-normal) that allows for non-constant variance. To accomplish this goal, we will need to be able to write some of our own code and also use built-in optimizers in R. Keep this example in mind when we go over numerical methods for estimating parameters using maximum likelihood.

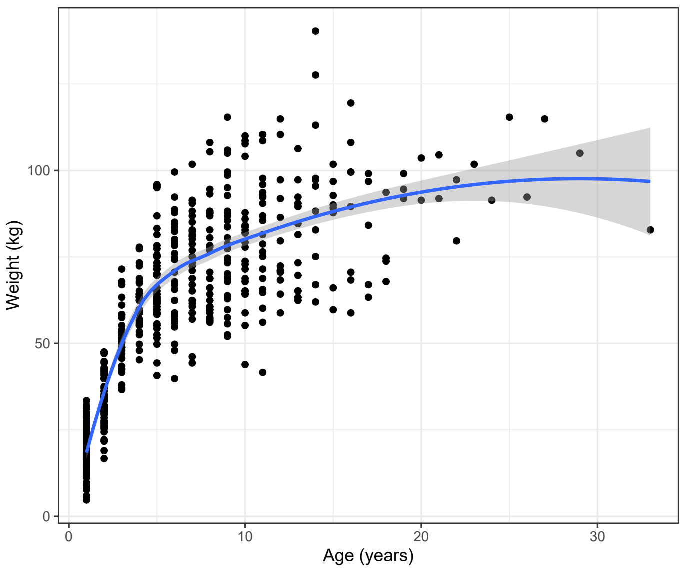


FIGURE 10.1 Weight versus age depicted for black bears monitored in Minnesota.

10.3 Introductory example: Estimating slug densities

Crawley (2012) describe a simple data set consisting of counts of the number of slugs found beneath 40 tiles in each of two permanent grasslands. The data are contained within the `Data4Ecologists` library.

```
library(Data4Ecologists)
data(slugs)
str(slugs)
```

```
'data.frame':   80 obs. of  2 variables:
 $ slugs: int  3 0 0 0 0 1 0 3 0 0 ...
 $ field: chr  "Nursery" "Nursery" "Nursery" "Nursery" ...
```

Let's begin by looking at the distribution of slug counts under the tiles in these two fields (Figure 10.2).

```
ggplot(slugs, aes(slugs, fill=field))+
  geom_bar(position=position_dodge())+
  theme(text = element_text(size=20))+
   scale_fill_manual(values=c("red", "blue"))+
   scale_x_continuous(breaks=seq(0,11,1))
```

We see that there are more slugs, on average, under the tiles located at the Rookery site than the Nursery site. Also note that the data take on only integer values $(0, 1, \ldots, 10)$ and that the distribution of slug counts is highly right skewed; this shape is typical of many count data sets. Thus, the data are not likely to be well described by a Normal distribution.

What if we wanted to conduct a formal hypothesis test to see whether the mean number of slugs per tile

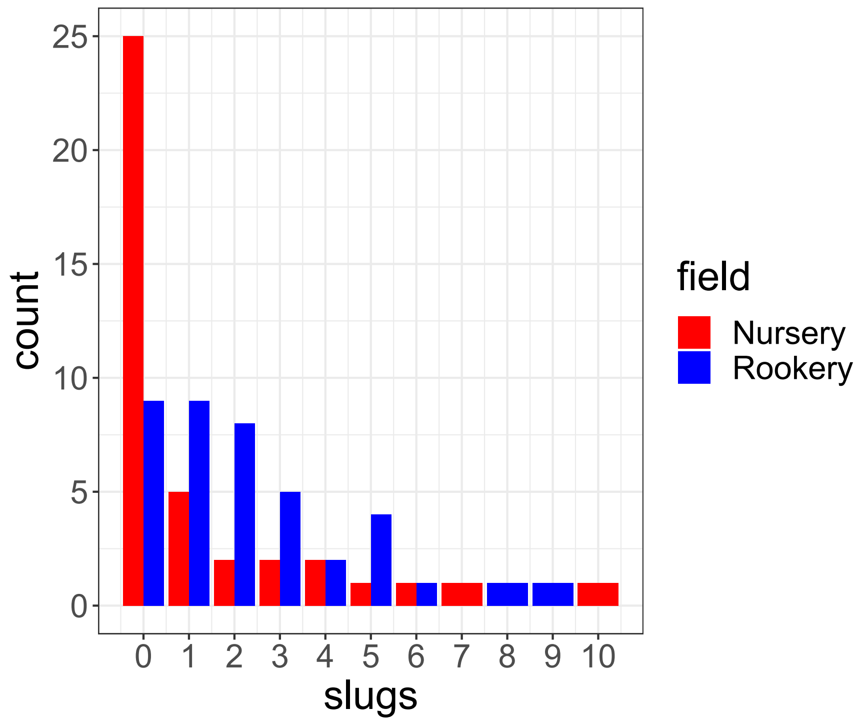


FIGURE 10.2 Distribution of slug counts under tiles placed in two fields.

differed between the two sites? I.e., what if we wanted to test the Null Hypothesis, $H_0 : \mu_{rookery} = \mu_{nursery}$ versus the alternative hypothesis $H_A : \mu_{rookery} \neq \mu_{nursery}$?[1]

We could consider using a two-sample t-test for a difference in means. In this case, what would we have to assume? At a minimum, we would need to assume that the observations are:

1. Independent
2. Normally distributed or that the sample size is "large enough" for the Central Limit Theorem (CLT) to apply

A common guideline is that $n \geq 30$ for the CLT to apply, with larger sample sizes needed when the population distributions are heavily skewed. Thus, we may be OK in this case due to having 40 observations in both fields. Nonetheless, we might consider whether there are more appropriate statistical distributions that we could use (instead of a Normal distribution) to describe the data from both fields. Given that we have count data, we might consider a Poisson or Negative Binomial distribution since these distributions also only take on integer values.

Let's begin by assuming the counts at each site follow a Poisson distribution. We could then represent our null and alternative hypotheses using the models, below:

[1]As an aside, if you came to me with this question, I would probably press you to justify why you think a hypothesis test is useful. As my friend and colleague Doug Johnson would point out, this null hypothesis is almost surely false (D. H. Johnson 1999). If we exhaustively sampled both fields, we almost surely would find that the densities differ, though perhaps not by much. The only reason why we might fail to reject the null hypothesis is that we have not sampled enough tiles. Thus, a p-value from a null hypothesis test adds little value. We would be better off framing our question in terms of an *estimation problem*. Specifically, we might want to estimate *how* different the two fields are in terms of their density. I.e., we should focus on estimating $\mu_{rookery} - \mu_{nursery}$ and its uncertainty.

Null model:

$Y_i \sim Poisson(\lambda_0)$ $(i = 1, 2, \ldots, 80)$

Alternative model:

$Y_{ij} \sim Poisson(\lambda_j)$ where $i = 1, 2, \ldots, 40$ and $j = 1, 2$ for the Nursery and Rookery site, respectively.

OK, great - but, how do we estimate λ_0 and (λ_1, λ_2)? And, how can we compare the two models to determine whether there is evidence to suggest the λ's differ at the two sites?

10.4 Probability to the likelihood

To estimate parameters using Maximum Likelihood, we begin by writing down a probability statement reflecting an assumed data-generating model. If all of our observations are independent, then:

$$P(Y_1 = y_1, Y_2 = y_2, \ldots, Y_n = y_n) = P(Y_1 = y_1)P(Y_2 = y_2)\cdots P(Y_n = y_n) = \prod_{i=1}^{n} P(Y_i = y_i)$$

(remember our probability rules from Chapter 9). In the examples considered in this textbook, we will specify $P(Y_i = y_i)$ in terms of a specific probability density or probability mass function with parameters θ, $f(y_i; \theta)$:

$$P(Y_1 = y_1, Y_2 = y_2, \ldots, Y_n = y_n) = \prod_{i=1}^{n} f(y_i; \theta)$$

In writing down this statement, we have treated the data as random and determined by a probability distribution with fixed but unknown parameters. For estimation purposes, we instead think of this expression as a function of the unknown parameters conditional on the observed data. We then refer to this expression as the likelihood of our data:

$$L(\theta; y_1, y_2, \ldots, y_n) = \prod_{i=1}^{n} f(y_i; \theta)$$

Let's build the likelihood of the slug data using the null data-generating model, $Y_i \sim Poisson(\lambda_0)$. We can begin by writing down the probability statement for just the first observation, $Y_1 = 3$. Assuming all tiles are the same size, the probability mass function for the Poisson distribution is given by:

$$f(y_i; \lambda) = \frac{e^{-\lambda_0}(\lambda_0)^{y_i}}{y_i!}$$

Thus, we have that:

$$P(Y_1 = 3) = \frac{e^{-\lambda_0}(\lambda_0)^3}{3!}$$

If, alternatively, we had chosen a negative binomial distribution as our data-generating model, we would have:

$$P(Y_i = y_i) = \binom{y_i + \theta - 1}{y_i} \left(\frac{\theta}{\mu + \theta}\right)^{\theta} \left(\frac{\mu}{\mu + \theta}\right)^{y_i}$$

We can write down similar expressions for all of the other observations in the data set and multiply them together to give us the likelihood:

$$L(\lambda; y_1, y_2, \ldots, y_{80}) = \frac{e^{-\lambda_0}(\lambda_0)^3}{3!}\frac{e^{-\lambda_0}(\lambda_0)^0}{0!}\cdots\frac{e^{-\lambda_0}(\lambda_0)^1}{1!} \tag{10.2}$$

Or, in more general notation, the likelihood for n independent observations from a Poisson distribution is given by:

$$L(\lambda; y_1, y_2, \ldots, y_n) = \prod_{i=1}^{n}\frac{e^{-\lambda_0}\lambda_0^{y_i}}{y_i!}$$

For discrete distributions, the likelihood gives us the probability of obtaining the observed data, given a set of parameters (in this case, λ_0). This probability will usually be extremely small. That is okay and to be expected. Consider, for example, the probability of seeing a specific set of 12 coin flips, `Head, Tail, Tail, Head, Head, Head, Tail, Tail, Head, Tail, Tail, Head`. The probability associated with this specific set of coin flips is $0.5^{12} = 0.000244$ (extremely small) because there are so many possible realizations when flipping a coin 12 times. If p were unknown, however, we could still use these data to evaluate evidence for whether we have an unfair coin (e.g., with probability of a Head, $p \neq 0.5$) and also to determine the most likely value of p given our observed data. What matters is the *relative probability* of the data across different potential values of p. This is the key idea behind maximum likelihood – we use the likelihood to determine the value of our parameters that make the data *most* likely.

10.5 Maximizing the likelihood

The likelihood of our slug data under the null data-generating model, which assumes the observations are independent realizations from a common Poisson distribution with parameter λ (dropping the "0" subscript as it is not needed since we only have 1 parameter), is given by:

$$L(\lambda; y_1, y_2, \ldots, y_n) = \prod_{i=1}^{n}\frac{e^{-\lambda}\lambda^{y_i}}{y_i!}$$

Using the result that $a^b a^c = a^{b+c}$, we can write this expression as:

$$L(\lambda; y_1, y_2, \ldots, y_n) = \frac{e^{-n\lambda}(\lambda)^{\sum_{i=1}^{n} y_i}}{\prod_{i=1}^{n} y_i!} \tag{10.3}$$

As previously mentioned, often the probability of the observed data (i.e., the likelihood) will be extremely small. This can create lots of numerical challenges when trying to find the parameters that maximize the likelihood. For this, and for other reasons[2], it is more common to work with the log-likelihood. Because the relationship between the likelihood and the log-likelihood is monotonic (i.e., increases in the likelihood always coincide with increases in the log-likelihood), the same value of λ will maximize both the likelihood and log-likelihood. We can write the log-likelihood as[3]:

[2]The log-likelihood for independent data is expressed as a sum of independent terms, which makes it relatively easy to derive asymptotic properties of maximum likelihood estimators using the Central Limit Theorem; see Section 10.6.

[3]Using the rule that $log(xy) = log(x) + log(y)$

$$\log L(\lambda; y_1, y_2, \ldots, y_n) = \log(\prod_{i=1}^{n} f(y_i; \lambda)) = \sum_{i=1}^{n} \log(f(y_i; \lambda)) \tag{10.4}$$

The log-likelihood for the Poisson model is thus given by[4]:

$$\log L(\lambda; y_1, y_2, \ldots, y_n) = -n\lambda + \log(\lambda)\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} \log(y_i!) \tag{10.5}$$

We will use this expression to find the value of the λ that makes the likelihood of observing the data as large as possible. We refer to this λ, $\hat{\lambda}$, as the maximum likelihood estimate or MLE.

How can we find the value of λ that maximizes eq (10.5)? We have several options, which we will explore in the following subsections:

- calculus
- using a grid search
- using functions in R or other software that rely on numerical methods to find an (approximate) maximum or minimum of a function

10.5.1 Calculus

To find the value of λ that maximizes the likelihood, we can take the derivative of the log-likelihood (eq. (10.5)) with respect to λ, set the derivative to 0, and then solve for λ. If you are like me, your calculus skills may be a bit rusty (though, mine improved while writing this book since my oldest daughter was taking calculus at the time). Fortunately, there are many websites that can perform differentiation if you get stuck[5]. We will need to remember the following rules in order to take the derivative of the log-likelihood:

- $\frac{d(ax)}{dx} = a$
- $\frac{d\log(x)}{dx} = \frac{1}{x}$
- $\frac{da}{dx} = 0$
- $\frac{d(f(x)+g(x))}{dx} = \frac{df(x)}{dx} + \frac{dg(x)}{dx}$

Putting things together, we have:

$$\frac{d\log L(\lambda; y_1, y_2, \ldots, y_n)}{d\lambda} = -n + \frac{\sum_{i=1}^{n} y_i}{\lambda}$$

Setting this expression to 0 and solving gives us:

$$\hat{\lambda} = \frac{\sum_{i=1}^{n} y_i}{n} = \bar{y}$$

To make sure we have found a maximum (and not a minimum), we should ideally take the second derivative of the log-likelihood and evaluate it at $\hat{\lambda}$. If this expression is negative, then we know the likelihood and

[4]Using the rules that $\log(\exp(x)) = x$, $\log(x^a) = a\log(x)$, and $\log(1/x) = -log(x)$

[5]One example is: https://www.symbolab.com/solver/derivative-calculator

log-Likelihood functions are concave down at $\lambda = \hat{\lambda}$, and we have thus found a maximum. Recalling that $\frac{d}{dx}\left(\frac{1}{x}\right) = \frac{-1}{x^2}$, and noting that all y values are non-negative, we have:

$$\frac{d^2 \log L(\lambda; y_1, y_2, \ldots, y_n)}{d\lambda^2} = -\frac{\sum_{i=1}^n y_i}{\lambda^2} < 0 \text{ for all values of } \lambda$$

.

Thus, we have verified that the maximum likelihood estimator for λ is equal to the sample mean, $\hat{\lambda}_{MLE} = \bar{y}$. This result makes intuitive sense since the mean of the Poisson distribution is equal to λ. For our slug data, the sample mean is given by:

```
mean(slugs$slugs)
```

```
## [1] 1.775
```

10.5.2 Grid search

What if your likelihood is more complicated (e.g., it involves more than 1 parameter), you do not have access to the internet for a derivative calculator, or, more likely, there is no analytical solution available when you set the derivatives with respect to the different parameters equal to 0?[6] In these cases, you could find an approximate solution using numerical methods, e.g., by using a grid search. Usually, I have students explore this concept using Excel, where they can enter equations for calculating the probability mass function associated with each observation and then multiply these values together to get the likelihood (eq. (10.2)). Then, we transition to R to see how we can accomplish the same task using functions.

Functions are incredibly useful for organizing code that needs to be repeatedly executed, while also allowing for flexibility depending on user specified options passed as *arguments*. We have been using functions throughout this course (e.g., `lm`, `plot`, etc). We will need to create our own function, which we will call `Like.s` (for likelihood slugs; you can of course use another name). This function will calculate the likelihood of the slug data under the null data-generating model for any specific value of λ. Before we go into the details of how this is done, let's consider what we need this function to do:

1. Calculate $P(Y_i = y_i)$ using the probability mass function of the Poisson distribution and a given value of λ for all observations in the data set.
2. Multiply these different probabilities together to give us the likelihood of the data for a particular value of λ.

Recall that R has built in functions for working with statistical distributions (Chapter 9). For the Poisson distribution, we can use the `dpois` function to calculate $P(Y_i = y_i)$ for each observation. For example, if we want to evaluate $P(Y_i = 3)$ for $\lambda = 2$, we can use:

```
dpois(3, lambda=2)
```

```
## [1] 0.180447
```

To verify that this number is correct, we could calculate it using the formula for the probability mass function of the Poisson distribution:

[6] Most problems cannot be solved analytically. For example, generalized linear models with distributions other than the Normal distribution will require numerical methods when estimating parameters using maximum likelihood.

```
exp(-2)*2^3/factorial(3)
```

```
## [1] 0.180447
```

So, we have accomplished our first objective. If, instead, we wanted to use a negative binomial distribution, we would use the `dnbinom` function, passing it parameter values for `mu` and `size`. Next, we need to multiply these probabilities together. R has a built in function called `prod` for taking the product of elements in a vector. For example, we can calculate $P(Y_1 = 3)P(Y_2 = 0)$ when $\lambda = 2$ using either of the expressions below:

```
dpois(3,lambda=2)*dpois(2, lambda=2)
```

```
## [1] 0.0488417
```

```
prod(dpois(c(3,2), lambda=2))
```

```
## [1] 0.0488417
```

The latter approach will be easier to work with when we have a large (and possibly unknown) number of observations. Putting everything together, we write our likelihood function so that it includes 2 arguments, `lambda` (the parameter of the Poisson distribution) and `dat` (a data set of observations):

```
Like.s<-function(lambda, dat){
      prod(dpois(dat, lambda))
  }
```

We can now evaluate the likelihood for a given value of λ, say $\lambda = 2$ using:

```
Like.s(lambda=2, dat=slugs$slugs)
```

```
## [1] 5.532103e-78
```

We see that this probability is really small, but again, what we are interested in is finding the value of λ that makes the probability as large as possible. We can evaluate the likelihood for a large number of equally spaced λ values between 0.1 and 5 using a loop:

```
# create a grid of lambda values
lambda.test <- seq(0.1, 5, length=1000)

# Create matrix to store likelihood values
L.vals <- matrix(NA, 1000, 1)
for(i in 1:1000){
  L.vals[i]<-Like.s(lambda=lambda.test[i], dat=slugs$slugs)
}
```

Alternatively, we could use `sapply` in base R or `map_dbl` in the `purrr` package (Henry and Wickham 2020) to vectorize the calculations, which will speed things up (though the loop is really fast in this case). Both `sapply` and `map_dbl` will evaluate a function for all values in a vector or list and return the output as a new vector. To use `sapply` or `map_dbl`, we need to pass a vector of λ values as the first argument and our `Like.s` function as the second argument. We also need to pass the slug observations so that `Like.s` can calculate the likelihood for each value of λ. If you are unfamiliar with these functions, do not fret. A loop will work just fine. The syntax for `sapply` or `map_dbl` is given by:

```
sapply(lambda.test, Like.s, dat=slugs$slugs)
purrr::map_dbl(lambda.test, Like.s, dat=slugs$slugs)
```

Let's look at how the likelihood changes as we vary λ (Figure 10.3):

```
plot(lambda.test, L.vals,
     xlab = expression(lambda),
     ylab = "Likelihood",
     type = "l")
```

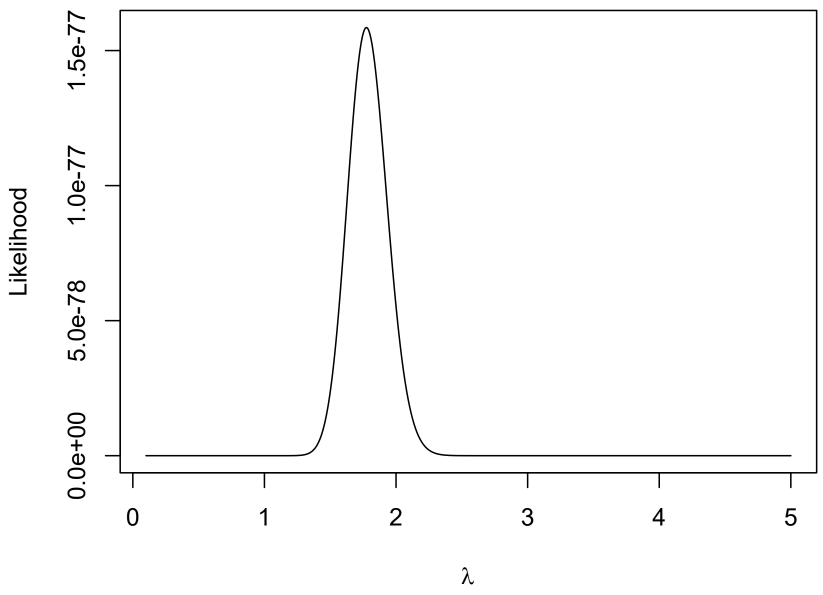


FIGURE 10.3 Likelihood of the slug data as a function of λ, assuming the slug counts are Poisson distributed with constant λ.

We can find the value of value of λ that makes the likelihood of our data as large as possible using the `which.max` function in R[7]:

```
# lambda that maximizes the likelihood
lambda.test[which.max(L.vals)]
```

[7]If you see a function that you are unfamiliar with and want to learn more, a good starting point is to access the help page for the function by typing `?functionname`. Although R help files can be rather difficult to digest sometimes, it can be helpful to scroll down to the bottom and look for examples of how to use the functions can be used.

```
## [1] 1.772573
```

This value is identical to the sample mean out to 2 decimal places (we could get a more accurate answer if we used a finer grid).

As previously mentioned, the likelihood values are really small for all values of λ. This can lead to numerical challenges associated with representing small numbers (referred to as underflow), making it difficult to find the parameters that maximize the likelihood. Therefore, it is more common to work with the log-likelihood, which will be maximized at the same value of λ. To verify, we create a function for calculating the log-likelihood of the data and repeat our grid search. We will again use the `dpois` function, but we add the argument `log = TRUE` so that this function returns $\log(f(y_i;\lambda))$ rather than $f(y_i;\lambda)$. In addition, we sum the individual terms (using the `sum` function) rather than taking their product. Instead of writing another loop, we will take advantage of the `map_dbl` function in the purrr library to calculate the log-likelihood values:

```
#log Likelihood
logL.s<-function(lambda, dat){
     sum(dpois(dat,lambda, log = TRUE))
}
# evaluate at the same grid of lambda values
logL.vals <- purrr::map_dbl(lambda.test, logL.s, dat=slugs$slugs)

# find the value that maximizes the log L and plot
lambda.test[which.max(logL.vals)]
```

```
## [1] 1.772573
```

We see that the same value of $\lambda = 1.77$ maximizes both the likelihood and log-likelihood (Figure 10.4).

```
plot(lambda.test, logL.vals, xlab=expression(lambda), ylab="Log Likelihood",type="l")
abline(v=lambda.test[which.max(logL.vals)])
```

10.5.3 Using numerical optimizers

Most software packages have built in functions that can find parameter values that either maximize or minimize a specified function. R has several functions available for this purpose, including `optim`, `nlm`, `nlminb`, among others. In this section, we will explore the use of `optim`, which can be used to find the parameters that minimize any function. It has the following arguments that we must supply:

- `par` = a vector of "starting" parameters (first guesses).
- `fn` = a function to minimize. Importantly, the first argument of this function must correspond to the vector of parameters that we want to optimize (i.e., estimate).

These two arguments are expected to occur in this order with `par` first and `fn` second. But, as with any function in R, we can change the order of the arguments as long as we use their names. Consider, for example, `dpois`. Its first two arguments are `x` and `lambda`. Normally, I am lazy and just pass the arguments in this order without supplying their names. For example,

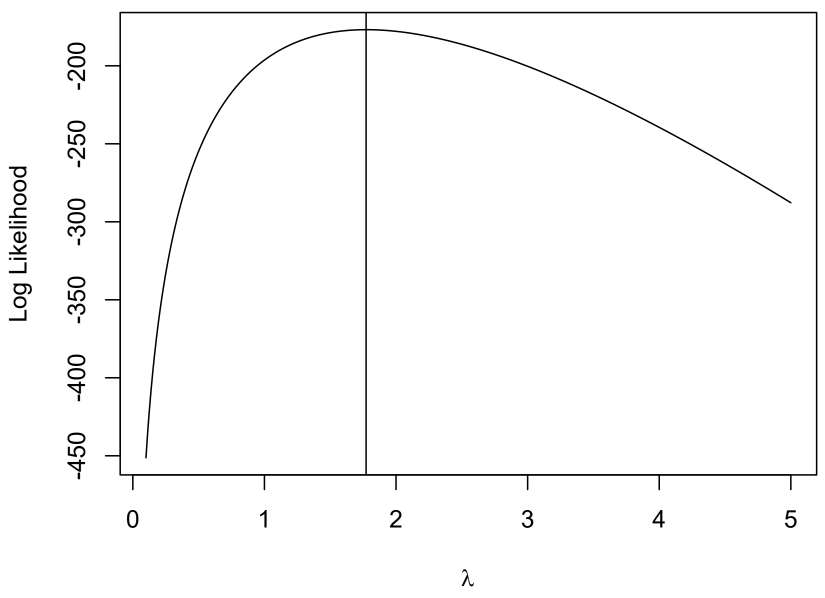


FIGURE 10.4 Log-likelihood of the slug data as a function of λ, assuming the slug counts are Poisson distributed with constant λ.

```
dpois(3, 5)
```

```
## [1] 0.1403739
```

We can, however, reverse the order if we use the argument names:

```
dpois(lambda = 5, x = 3)
```

```
## [1] 0.1403739
```

We can use the `method` argument to specify a particular optimization method (we will use `method = "BFGS"`). We can also pass other arguments to `optim` (e.g., `dat`, which is needed to calculate the value of the log-likelihood). Importantly, by default, `optim` tries to find a minimum. Thus, we create a new function, `minus.logL.s` that calculates the negative log-likelihood. The value that minimizes the negative log-likelihood will maximize the log-likelihood.

```
minus.logL.s <- function(lambda, dat){
   -sum(dpois(dat,lambda, log = TRUE))
}
```

Most numerical optimization routines are iterative. They begin with a starting value (or guess), and then use information about the function (and its derivatives) at that guess to derive the next value to try. This results in a sequence of iterative guesses that hopefully converge to the true minimum. If there is only 1 minimum (or maximum), most optimizers will work well. But, if there are multiple minima or maxima, then optimizers will often converge to the one closest to the starting value. Thus, it can be important to try different starting values to help determine if there are multiple minima. If you find that the results from using `optim` depend on the starting value, then you should be cautious, try multiple starting values, and select the one that results in the smallest negative log-likelihood. You might also consider trying alternative estimation approaches (e.g., Bayesian methods that rely on Monte Carlo Markov Chain [MCMC] sampling may be more robust; Subhash R. Lele, Dennis, and Lutscher 2007)

For the slug example, there is only 1 minimum (Figure 10.3, 10.4), and therefore, most starting values for λ will converge to the correct value (though, we must supply a starting value > 0 since $\lambda > 0$ in the Poisson distribution). Here, we supply a value of 2 to begin the optimization routine:

```
mle.fit<-optim(par = 2, fn = minus.logL.s,  method = "BFGS",  dat = slugs$slugs)
```

```
## Warning in dpois(dat, lambda, log = TRUE): NaNs produced
```

Note that `optim` outputs a warning here about NaNs being produced. The parameter, λ of the Poisson distribution, has to be positive. We have not told `optim` this, and therefore, it is possible that it will try negative values when attempting to evaluate the log-likelihood. In this case, `dpois` will return an `NaN`, giving us the warning. Yet, if we inspect the output, we will see that it was able to continue and find the value of λ that maximizes the likelihood:

```
mle.fit
```

```
## $par
## [1] 1.775
##
## $value
## [1] 176.8383
##
## $counts
## function gradient
##       15        6
##
## $convergence
## [1] 0
##
## $message
## NULL
```

We see that `optim` returns:

- `par` = the value of λ that minimizes the negative log-likelihood; this is the maximum likelihood estimate!
- `value` = the value of the function (i.e., the negative log-likelihood) at its minimum value
- `counts` = information about how hard it had to work to find the minimum. We see that it had to calculate the likelihood for 15 different values of λ and it calculated $\frac{d \log L}{d\lambda}$ 6 times (`function` and `gradient`, respectively)
- `convergence` = an indicator of whether the optimizer converged; a value of 0 suggests the optimizer was successful, whereas a value of 1 would indicate that it hit a limit based on the maximum number of iterations allowed (which can be changed from a default value of 100)

In summary, we have used 3 different methods for finding the value of λ that maximizes the log-likelihood. In each case, we ended up with $\hat{\lambda} = 1.77$.

10.5.4 Fitting the alternative data-generating model (site-specific Poisson distributions)

Consider our alternative data-generating model:

$Y_{ij} \sim Poisson(\lambda_j)$ where $i = 1, 2, \ldots, 40$ and $j = 1, 2$ for the Nursery and Rookery site, respectively.

We could estimate λ_1 and λ_2 in a number of different ways. We could split the data by field and then use each of the methods previously covered (calculus, grid search, numerical optimizer) but applied separately to each data set. Nothing new there. We could apply a 2-dimensional grid search, which could be instructive for other situations where we have more than 1 parameter (e.g., when fitting a negative binomial data-generating model to the data). Or, we could continue to get familiar with `optim` as it offers a more rigorous way of finding parameters that maximize the likelihood and will also facilitate estimation of parameter uncertainty, as we will see later. Using `optim` will require rewriting our negative log-likelihood function to allow for two parameters. We will also need to pass to the function the data that identifies from which field the observations come from so that it uses the correct λ for each observation. One option is given below, where we now pass a full data set rather than just the slug counts:

```
minus.logL.2a<-function(lambdas, dat){
     -sum(dpois(dat$slugs[dat$field=="Nursery"], lambdas[1], log=TRUE))-
      sum(dpois(dat$slugs[dat$field=="Rookery"], lambdas[2], log=TRUE))
  }
```

Earlier, we noted that we did not tell `optim` that our parameters must be > 0 (i.e., $\lambda_1, \lambda_2 > 0$). We could use a different optimizer that allows us to pass constraints on the parameters. For example, we could change to `method = "L-BFGS-B"` and supply a lower bound for both parameters via an extra argument, `lower = c(0, 0)`. Alternatively, we could choose to specify our likelihood in terms of parameters estimated on the log-scale and then exponentiate these parameters to get $\hat{\lambda}$. This is often a useful strategy for constraining parameters to be positive. For example, we could use:

```
minus.logL.2b<-function(log.lambdas, dat){
     -sum(dpois(dat$slugs[dat$field=="Nursery"], exp(log.lambdas[1]), log=TRUE))-
      sum(dpois(dat$slugs[dat$field=="Rookery"], exp(log.lambdas[2]), log=TRUE))
  }
```

In this way, we estimate parameters that can take on any value between $-\infty$ and ∞, but the parameters used in `dpois` will be > 0 because we exponentiate the optimized parameters using `exp()`. In fact, this is what R does when it fits the Poisson model using its built in `glm` function.

Let's use `optim` with `minus.logL.2b` to estimate λ_1 and λ_2 in our alternative data-generating model (note, you will need to supply a vector of starting values when using `optim`, say `par = c(2,2)`).

```
mle.fit.2lambdas <- optim(par = c(2, 2), fn = minus.logL.2b, dat = slugs, method = "BFGS")
  mle.fit.2lambdas
```

```
## $par
## [1] 0.2429461 0.8219799
##
```

```
## $value
## [1] 171.1275
##
## $counts
## function gradient
##       29       10
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Remember, our parameters now reflect log(λ), so to estimate λ, we have to exponentiate these values:

```
exp(mle.fit.2lambdas$par)
```

```
## [1] 1.275 2.275
```

These are equivalent to the sample means in the two fields:

```
slugs %>% group_by(field) %>%
  dplyr::summarize(mean(slugs))
```

```
## # A tibble: 2 x 2
##   field   `mean(slugs)`
##   <chr>           <dbl>
## 1 Nursery          1.27
## 2 Rookery          2.28
```

At this point, to test your understanding, you might consider the following two exercises:

Exercise: modify the likelihood function to fit a model assuming the slug counts were generated by a negative binomial distribution (ecological version; Section 9.10.6.2).

Exercise: modify the two parameter Poisson model so that it is parameterized using reference coding. Your estimates should match those below:

```
glm(slugs~field, data=slugs, family=poisson())
```

```
##
## Call:  glm(formula = slugs ~ field, family = poisson(), data = slugs)
##
## Coefficients:
##  (Intercept)  fieldRookery
##       0.2429        0.5790
##
## Degrees of Freedom: 79 Total (i.e. Null);  78 Residual
## Null Deviance:       224.9
## Residual Deviance: 213.4     AIC: 346.3
```

10.6 Properties of maximum likelihood estimators

Let θ be a parameter of interest and $\hat{\theta}$ be the maximum likelihood estimator of θ. Maximum likelihood estimators (MLEs) have many desirable properties (assuming the choice of distribution is correct). In particular,

- MLEs are consistent, meaning that as the sample size increases towards ∞, they will converge to the true parameter value.
- MLEs are asymptotically unbiased, i.e., $E(\hat{\theta}) \to \theta$ as $n \to \infty$. Maximum likelihood estimators are not, however, guaranteed to be unbiased in small samples. As one example, the maximum likelihood estimator of σ^2 for Normally distributed data is given by $\hat{\sigma}^2_{MLE} = \sum_{i=1}^n (y_i - \bar{y})^2/n$, which has as its expected value, $E[\hat{\sigma}^2_{MLE}] = \frac{n-1}{n}\sigma^2$. The bias of the maximum likelihood estimator becomes negligible as sample sizes become large. Yet, you are probably more familiar with the unbiased estimator, $\hat{\sigma}^2 = \sum_{i=1}^n (y_i - \bar{y})^2/(n-1)$.
- MLEs will have the minimum variance among all estimators as $n \to \infty$.

In addition, for large sample sizes, we can approximate the sampling distribution of maximum likelihood estimators using:

$$\hat{\theta} \sim N(\theta, I^{-1}(\theta)),$$

where $I(\theta)$ is called the Information matrix. We will use this last property repeatedly as it provides the basis for constructing hypothesis tests and confidence intervals for maximum likelihood estimators.

Before we can define the Information matrix, we will need to formally define a Hessian; the Hessian is a matrix of second-order partial derivatives of a function, in this case, derivatives of the log-likelihood function with respect to the parameters $(\theta_1, \theta_2, \ldots, \theta_p)$:

$$\text{Hessian}(\theta)_{p\times p} = \begin{bmatrix} \frac{\partial^2 logL(\theta)}{\partial\theta_1^2} & \frac{\partial^2 logL(\theta)}{\partial\theta_1\theta_2} & \cdots & \frac{\partial^2 logL(\theta)}{\partial\theta_1\theta_p} \\ \frac{\partial^2 logL(\theta)}{\partial\theta_1\theta_2} & \frac{\partial^2 logL(\theta)}{\partial\theta_2^2} & \cdots & \frac{\partial^2 logL(\theta)}{\partial\theta_2\theta_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 logL(\theta)}{\partial\theta_1\theta_p} & \frac{\partial^2 logL(\theta)}{\partial\theta_2\theta_p} & \cdots & \frac{\partial^2 logL(\theta)}{\partial\theta_p^2} \end{bmatrix}$$

If there is only 1 parameter in the likelihood, then the Hessian will be a scaler and equal to $\frac{d^2 \log L(\theta)}{d\theta^2}$, i.e., the second derivative of the log-likelihood with respect to the parameter.

The information matrix is defined in terms of the Hessian and takes one of two forms:

- The observed information matrix is the negative of the Hessian evaluated at the maximum likelihood estimate:

$$\text{observed } I(\theta)_{p\times p} = -\left.\begin{bmatrix} \frac{\partial^2 logL(\theta)}{\partial\theta_1^2} & \frac{\partial logL(\theta)}{\partial\theta_1\theta_2} & \cdots & \frac{\partial^2 logL(\theta)}{\partial\theta_1\theta_p} \\ \frac{\partial^2 logL(\theta)}{\partial\theta_1\theta_2} & \frac{\partial^2 logL(\theta)}{\partial\theta_2^2} & \cdots & \frac{\partial^2 logL(\theta)}{\partial\theta_2\theta_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 logL(\theta)}{\partial\theta_1\theta_p} & \frac{\partial^2 logL(\theta)}{\partial\theta_2\theta_p} & \cdots & \frac{\partial^2 logL(\theta)}{\partial\theta_p^2} \end{bmatrix}\right|_{\theta=\hat{\theta}}$$

- The expected information matrix is given by the expected value of the negative of the Hessian evaluated at the maximum likelihood estimate:

$$\text{expected } I(\theta)_{p\times p} = -E\left[\begin{bmatrix} \frac{\partial^2 logL(\theta)}{\partial\theta_1^2} & \frac{\partial logL(\theta)}{\partial\theta_1\theta_2} & \cdots & \frac{\partial^2 logL(\theta)}{\partial\theta_1\theta_p} \\ \frac{\partial^2 logL(\theta)}{\partial\theta_1\theta_2} & \frac{\partial^2 logL(\theta)}{\partial\theta_2^2} & \cdots & \frac{\partial^2 logL(\theta)}{\partial\theta_2\theta_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 logL(\theta)}{\partial\theta_1\theta_p} & \frac{\partial^2 logL(\theta)}{\partial\theta_2\theta_p} & \cdots & \frac{\partial^2 logL(\theta)}{\partial\theta_p^2} \end{bmatrix}\right]\Bigg|_{\theta=\hat{\theta}}$$

Note that the Hessian describes the curvature of the log-likelihood surface – the higher the curvature, the more quickly the log-likelihood decreases as we move away from the maximum likelihood estimate (Figure 10.5). If the curvature is large, then the asymptotic variance of $\hat{\theta}$ will be small. By contrast, when $\left[\frac{\partial^2 logL(\theta)}{\partial\theta^2}\right]$ is close to 0, this suggests that the log-likelihood is "flat" with many different parameter values giving similar log-likelihoods. In this case, the inverse of the Hessian, and hence the variance, will be large.

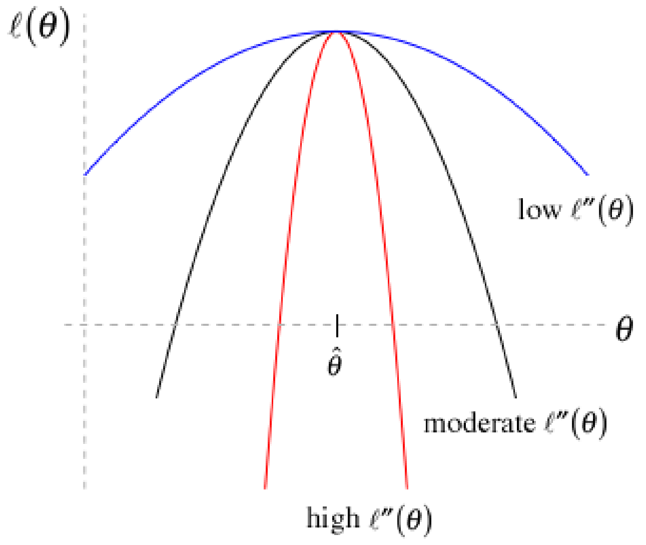


FIGURE 10.5 Large values of the Hessian (red) correspond to high levels of Information, high curvature of the log-likelihood surface, and low levels of uncertainty. Figure from Jack Weiss's online course notes.

Importantly, the Hessian is often used by numerical optimization methods (e.g., `optim`) to find parameter values that minimize an objective function (e.g., the negative log-likelihood). We can ask `optim` to return the Hessian by supplying the argument `hessian = TRUE`, which will be equivalent to the observed information matrix (since we defined our function as the negative of the log-likelihood, the minus sign will already be taken care of). We can then calculate the asymptotic variance-covariance matrix of $\hat{\theta}$ by taking the inverse of the Hessian.

Let's demonstrate with our slug example, where we fit the alternative data-generating model that assumed separate Poisson distributions for each field. We begin by asking `optim` to return the Hessian by supplying an extra argument, `hessian = TRUE`:

```
mle.fit.2lambdas <- optim(par = c(2, 2),
                          fn = minus.logL.2b,
                          dat = slugs,
                          method = "BFGS",
                          hessian = TRUE)
mle.fit.2lambdas
```

```
## $par
## [1] 0.2429461 0.8219799
##
## $value
## [1] 171.1275
##
## $counts
## function gradient
##       29       10
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##               [,1]          [,2]
## [1,]  5.100001e+01 -1.421085e-08
## [2,] -1.421085e-08  9.100001e+01
```

We can calculate asymptotic standard errors for our parameters by finding the inverse of the Hessian. To find the inverse of a matrix, H, in R, we can use `solve(H)`. Thus, we can find the variance/covariance matrix associated with parameter estimates using:

```
(var.cov.lambda <- solve(mle.fit.2lambdas$hessian))
```

```
##              [,1]         [,2]
## [1,] 1.960784e-02 3.062023e-12
## [2,] 3.062023e-12 1.098901e-02
```

The diagonal elements of this matrix correspond to our estimates of the variance of our parameters; the square root of the diagonals thus give us their standard errors[8]

```
sqrt(diag(var.cov.lambda))
```

```
## [1] 0.1400280 0.1048285
```

The off diagonals of our variance-covariance matrix tell us about how our parameters covary in their sampling distribution (i.e., do high values for $\hat{\lambda}_1$ tend to occur with high or low values of $\hat{\lambda}_2$). In this simple model, the parameter estimates are independent (data from the Nursery field does not inform the estimate of λ for the Rookery field and data from the Rookery field do not inform our estimate of λ for the Nursery field). If instead, we fit the model using effects coding, we will see that our parameters will covary in their sampling distribution.

Lastly, we can compare our estimates and their standard errors to those that are returned from R's `glm` function, which we will see in Chapter 15:

[8]Recall, $\sqrt{var}$ = standard deviation and a standard error is just a special type of standard deviation – i.e., a standard deviation of a sampling distribution!

```
summary(glm(slugs ~ field -1, data = slugs, family = "poisson"))
```

```
##
## Call:
## glm(formula = slugs ~ field - 1, family = "poisson", data = slugs)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## fieldNursery   0.2429     0.1400   1.735   0.0827 .
## fieldRookery   0.8220     0.1048   7.841 4.46e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 263.82  on 80  degrees of freedom
## Residual deviance: 213.44  on 78  degrees of freedom
## AIC: 346.26
##
## Number of Fisher Scoring iterations: 6
```

We see that we have obtained exactly the same estimates and standard errors as reported by the `glm` function. Also, note that at the bottom of the summary output, we see `Number of Fisher Scoring iterations: 6`. This comparison highlights:

1. `glm` is fitting a model parameterized in terms of the log mean (i.e., log λ)
2. `glm` is using numerical routines similar to `optim` and also reports standard errors that rely on the asymptotic results discussed in the above section, namely, $\hat{\theta} \sim N(\theta, I^{-1}(\theta))$.

10.7 A more complicated example: Fitting a weight-at-age model

Let us now return to our motivating example involving a weight-at-age model for female black bears in Minnesota. We will use `optim` to fit the following model:

$$\begin{aligned} Weight_i &\sim N(\mu_i, \sigma_i^2) \\ \mu_i &= L_\infty(1 - e^{-K(Age_i - t_0)}) \\ \sigma_i^2 &= \sigma^2 \mu_i^{2\theta} \end{aligned} \tag{10.6}$$

This allows us to model the mean using the von Bertalanffy growth function and model the variability about this mean using the variance function from Section 5.5.3.

We begin by constructing the likelihood function with the following parameters $L_\infty, K, t_0, \log(\sigma), \theta$:

```
logL<-function(pars, dat){
    linf <- pars[1]
    k <- pars[2]
    t0 <- pars[3]
    sig <- exp(pars[4])
    theta <- pars[5]
    mu<-linf*(1 - exp(-k*(dat$age - t0)))
    vari<-sig^2*abs(mu)^(2*theta)
    sigi<-sqrt(vari)
    ll<- -sum(dnorm(dat$wtkg, mean = mu, sd = sigi, log = T))
    return(ll)
  }
```

To get starting values, which we denote using a superscript s, we revisit our plot of the data (Figure 10.1), noting:

- The smooth curve asymptotes at around 100, so we use $L^s_\infty = 100$
- K tells us how quickly we approach this asymptote as bears age. If we set $\mu = L_\infty/2$ and solve for K, this gives $K = log(2)/(t - t_0)$. Then, noting that μ appears to be at $L_\infty/2 = 50$ at around 3 years of age, we set $K^s = log(2)/3 = 0.23$
- t_0 is the x-intercept (i.e., the Age at which $\mu = 0$); an extrapolation of the curve to $x = 0$ suggests that t_0 should be slightly less than 0, say $t^s_0 = -0.1$
- Most of the residuals when age is close to 0 are within approximately ± 10 kg. For Normally distributed data, we expect 95% of the observations to be within $\pm 2\sigma$ so we set $\log(\sigma)^s = log(10/2) = 1.61$
- We set $\theta^0 = 1$ (suggesting the variance increases linearly with the mean)

```
fitvb <- optim(par = c(100, 0.23, -0.1, 1.62, 1),
               fn = logL,dat = beargrowth,
               method = "BFGS", hessian = TRUE)
fitvb
```

```
## $par
## [1] 87.1070613  0.2853005  0.1530582 -0.1744772  0.6607753
##
## $value
## [1] 2156.353
##
## $counts
## function gradient
##      108       25
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##            [,1]       [,2]        [,3]        [,4]       [,5]
## [1,]   1.633524   200.5033   -31.11917    8.632616    33.2212
## [2,] 200.503252 36487.7264 -7444.52556 1325.372227  4676.6273
## [3,] -31.119171 -7444.5256  2101.48826 -271.183081  -841.5901
## [4,]   8.632616  1325.3722  -271.18308 1138.010281  4383.5226
## [5,]  33.221200  4676.6273  -841.59012 4383.522578 17355.9438
```

Let's inspect the fit of our model by adding our predictions from the model to our data set (Figure 10.6).

```
pars<-data.frame(Linf = fitvb$par[1], K = fitvb$par[2], t0 = fitvb$par[3],
                 sigma = exp(fitvb$par[4]), theta = fitvb$par[5])
beargrowth <- beargrowth %>%
  mutate(mu.hat = pars$Linf*(1 - exp(-pars$K*(age - pars$t0))))
ggplot(beargrowth, aes(age, wtkg)) +
  geom_point() + geom_line(aes(age, mu.hat), col = "red", linewidth = 1) +
  ylab("Weight (kg)") + xlab("Age (years)") +
  theme_bw()
```

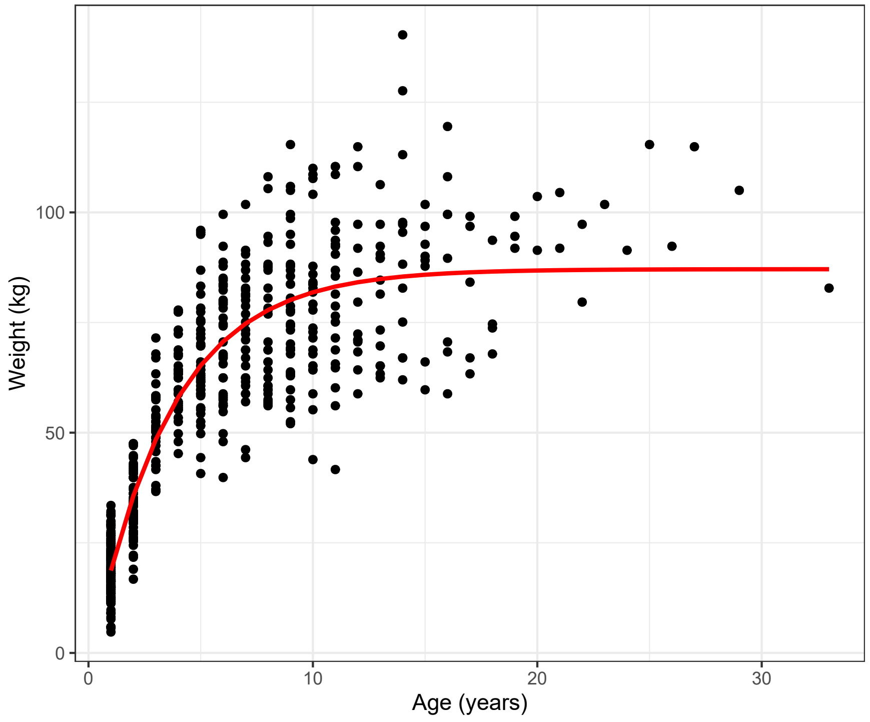


FIGURE 10.6 Fitted von Bertalanffy growth curve to weight-at-age data for black bears in Minnesota.

Next, let's create and inspect plots using standardized residuals (Figure 10.7). In particular, we can plot standardized residuals versus fitted values to evaluate whether the von Bertalanffy growth curve adequately describes how the mean weight varies with age. We can also plot the sqrt root of absolute values of these residuals versus fitted values to evaluate whether our variance model is appropriate.

```
beargrowth <- beargrowth %>%
  mutate(sig.hats = pars$sigma^2*abs(mu.hat)^(2*pars$theta)) %>%
  mutate(stdresids = (wtkg - mu.hat)/(sig.hats),
         sqrt.abs.resids = sqrt(abs(stdresids)))
p1<-ggplot(beargrowth, aes(mu.hat, stdresids)) + geom_point() +
  geom_hline(yintercept=0) + geom_smooth() + theme_bw()
p2<-ggplot(beargrowth, aes(mu.hat, sqrt.abs.resids)) + geom_point() +
   geom_smooth() + theme_bw()
p1+p2
```

```
## ‘geom_smooth()‘ using method = ’loess’ and formula = ’y ~ x’
## ‘geom_smooth()‘ using method = ’loess’ and formula = ’y ~ x’
```

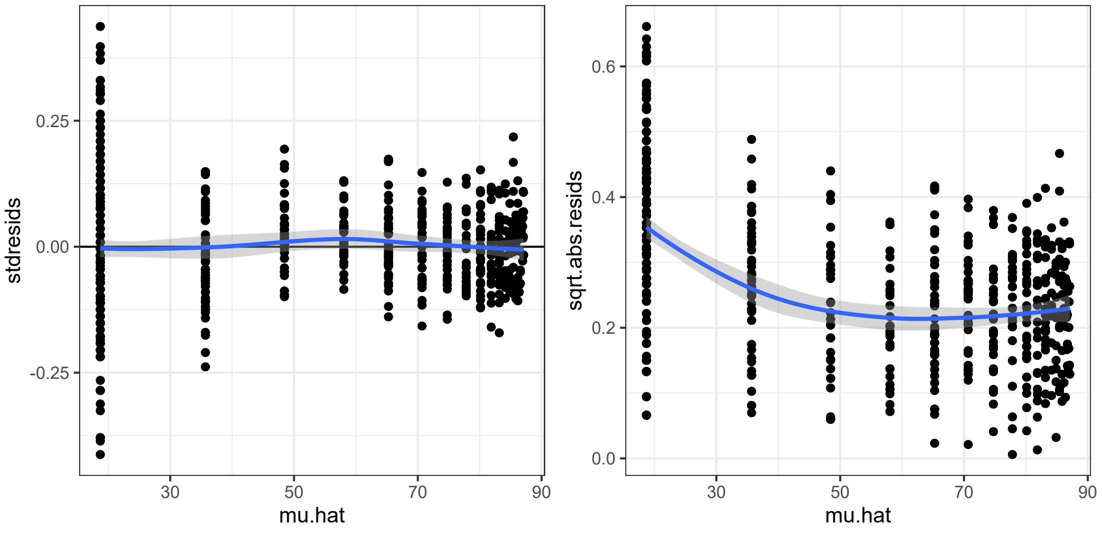


FIGURE 10.7 Standardized residuals (left panel) and square-rooted absolute values of these residuals (right panel) versus fitted values from the von Bertalanffy growth model applied to weight-at-age data from black bears in Minnesota.

The standardized residuals do not exhibit any trend as we move from left to right in the residual versus fitted value plot (left panel of Figure 10.7), but residuals of the smallest bears appear to be slightly more variability (right panel of Figure 10.7). At this point, we could try to fit alternative growth models for the mean weight at age or alternative models for the variance about the mean to see if we could better match the data. Nonetheless, the fit of the von Bertalanffy growth curve (Figure 10.6) provides a nice summary of the weight-at-age relationship. What if we wanted to add confidence or prediction intervals to our plot?

10.8 Confidence intervals for functions of parameters

The results from Section 10.6 provide a way for us to calculate uncertainty associated with our model parameters. I.e., if we let $\hat{\Psi} = (\hat{L}_\infty, \hat{K}, \hat{t}_0, \widehat{\log(\sigma)}, \hat{\theta})$, then we know the asymptotic variance-covariance matrix of $\hat{\Psi}$ is given by $I^{-1}(\Psi)$, which we can estimate from the returned Hessian:

```
(vcov.psi <- solve(fitvb$hessian))
```

```
##              [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  2.737807861 -2.424357e-02 -0.0455615458  0.0052606707 -2.245891e-03
## [2,] -0.024243567  3.165257e-04  0.0007662437  0.0001665308 -4.378909e-05
## [3,] -0.045561546  7.662437e-04  0.0025635157  0.0016454637 -4.105409e-04
## [4,]  0.005260671  1.665308e-04  0.0016454637  0.0346870394 -8.735923e-03
## [5,] -0.002245891 -4.378909e-05 -0.0004105409 -0.0087359227  2.260206e-03
```

The diagonals of this matrix tell us about the expected variability of our parameter estimates (across repetitions of data collection and analysis), with the square root of these elements giving us our standard errors. Thus, we can form 95% confidence intervals for our model parameters using the assumption that the sampling distribution is Normal. For example, a 95% confidence interval for L_∞ is given by:

```
SE.Linf<-sqrt(diag(vcov.psi))[1]
fitvb$par[1] + c(-1.96, 1.96)*rep(SE.Linf,2)
```

```
## [1] 83.86398 90.35014
```

All good, but again, what if we want a confidence interval for $E[Y_i|Age_i] = \mu_i = L_\infty(1-e^{-K(Age_i-t_0)})$, which is a function of more than 1 of our parameters. To quantify the uncertainty associated with $E[Y_i|Age_i]$, we have to consider the variability associated with each parameter. We also have to consider how the parameters *covary* – i.e., we need to consider the off-diagonal elements of $I^{-1}(\hat{\Psi})$. For example, the (2, 1) element of $I^{-}(\hat{\Psi})$ is negative, suggesting that large estimates of L_∞ tend to be accompanied by small estimates of K and vice versa.

Recall, in Section 5, we learned how to use matrix multiplication to get predicted values and their variance for linear models, possibly with non-constant variance. We briefly review those results here. Recall, if $\hat{\beta} \sim N(\beta, \Sigma)$, then:

$$X\hat{\beta} \sim N(X\beta, X\Sigma X^T),$$

and we can estimate the mean and variance using matrix multiplication, replacing β with $\hat{\beta}$ and Σ with $\hat{\Sigma}$, where $\hat{\Sigma}$ is the estimated variance covariance matrix of $\hat{\beta}$. This works if we are interested in a linear combination of our parameters (i.e., a weighted sum). But, what if we are interested in non-linear functions of parameters? I.e., $E[Y_i|Age_i] = L_\infty(1-e^{-k(Age_i-t_0)})$. We have 3 options:

- Use a bootstrap (see Section 2)
- Use the Delta method, which we will see shortly
- Switch to Bayesian inference and use posterior distributions

10.8.1 Delta Method

To understand how the delta method is derived, you have to know something about Taylor's series approximations. Taylor's series approximations are incredibly useful (e.g., they also serve to motivate some of the numerical methods we have used, including the `BFGS` method implemented by `optim`). We won't get into the gory details here, but rather will focus on implementation. To implement the delta method, we need to be able to take derivatives of our function with respect to each of our parameters.

Let $f(L_\infty, K, t_0, \log(\sigma), \theta; Age) = L_\infty(1-e^{-K(Age-t_0)})$ be our function of interest (i.e., the mean weight at a given age). To calculate $Var[f(\hat{L}_\infty, \hat{K}, \hat{t}_0, \log(\hat{\sigma}), \hat{\theta}; Age)]$, we need to determine:

$f'(L_\infty, K, t_0, \log(\sigma), \theta; Age) = (\frac{\partial\mu}{\partial L_\infty}, \frac{\partial\mu}{\partial K}, \frac{\partial\mu}{\partial t_0}, \frac{\partial\mu}{\partial\log(\sigma)}, \frac{\partial\mu}{\partial\theta})$

Then, we can approximate the variance of $f(\hat{L}_\infty, \hat{K}, \hat{t}_0, \log(\hat{\sigma}), \hat{\theta}; Age) = \hat{E}[Y_i|Age_i]$ for Age_i using matrix multiplication:

$$\hat{E}[Y_i|Age_i] \approx f'(L_\infty, K, t_0, \log(\sigma), \theta)I^{-1}(\Psi)f'(L_\infty, K, t_0, \log(\sigma), \theta)^T|_{L_\infty=\hat{L}_\infty, K=\hat{K}, t_0=\hat{t}_0, \log(\sigma)=\log(\hat{\sigma}), \theta=\hat{\theta}}$$

Because our function does not involve $\log(\sigma)$ or θ, we can simplify things a bit and just work with $f'(L_\infty, K, t_0)$ and the first 3 rows and columns of $I^{-1}(\Psi)$[9]:

[9]Again, if calculus is not your forte, online derivative calculators may prove useful

- $\frac{\partial \mu}{\partial L_\infty} = (1 - e^{-K(Age - t_0)})$
- $\frac{\partial \mu}{\partial K} = L_\infty (Age - t_0) e^{-K(Age - t_0)}$
- $\frac{\partial \mu}{\partial t_0} = -L_\infty K e^{-K(Age - t_0)})$

To calculate the variance for a range of ages (e.g., $Age = 1, 2, ..., 35$), we create a 35 x 3 dimension matrix with a separate row for each Age and the columns giving f' for each Age:

```
age <- 1:35
mu.hat <- pars$Linf*(1-exp(-pars$K*(age-pars$t0)))
fprime<-as.matrix(cbind(1-exp(-pars$K*(age-pars$t0)),
                         pars$Linf*(age-pars$t0)*exp(-pars$K*(age-pars$t0)),
                         -pars$Linf*pars$K*exp(-pars$K*(age-pars$t0))))
```

We then determine the variance using matrix multiplication, pulling off the diagonal elements (the off-diagonal elements hold the covariances between observations for different ages):

```
var.mu.hat <- diag(fprime %*% solve(fitvb$hessian)[1:3,1:3] %*%t(fprime))
```

Alternatively, the `emdbook` package (B. Bolker 2023) has a function, `deltavar`, that will do the calculations for you if you supply a function for calculating $f()$ (via argument `meanval`) and you pass the asymptotic variance covariance matrix, $I^{-1}(\Psi)$ (via the `Sigma` argument).

```
library(emdbook)
var.mu.hat.emd <- deltavar(linf*(1-exp(-k*(age-t0))),
                      meanval=list(linf=fitvb$par[1], k=fitvb$par[2], t0=fitvb$par[3]),
                      Sigma=solve(fitvb$hessian)[1:3,1:3])
```

We can take the square root of the diagonal elements of `var.mu.hat` to get SEs for forming pointwise confidence intervals for $E[Y_i|Age_i]$ at each Age.

```
muhats <- data.frame(age=age,
                     mu.hat=mu.hat,
                     se.mu.hat = sqrt(var.mu.hat))
ggplot(beargrowth, aes(age, wtkg)) +
  geom_point() + geom_line(aes(age, mu.hat), col="red", size=1) +
  ylab("Weight (kg)") + xlab("Age (years)") + theme_bw() +
  geom_ribbon(data=muhats, aes(x=age,
                               ymin=mu.hat-1.96*se.mu.hat,
                               ymax=mu.hat+1.96*se.mu.hat),
                 inherit.aes = FALSE, fill = "blue", alpha=0.5)
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

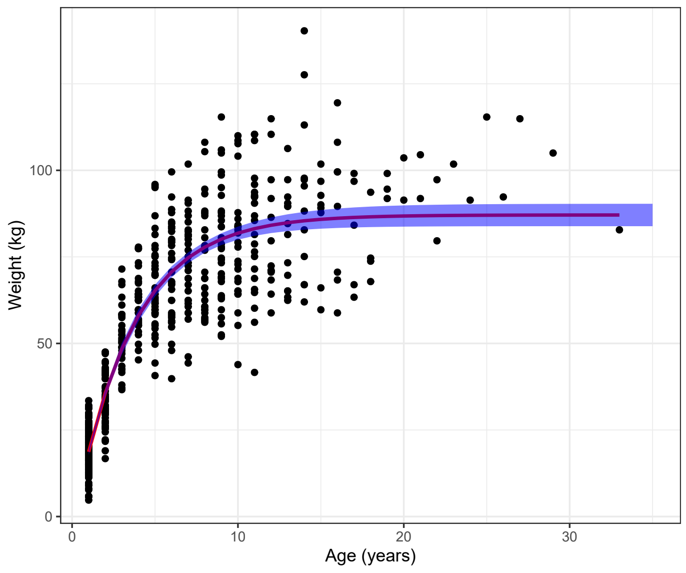


FIGURE 10.8 Fitted von Bertalanffy growth curve and 95% confidence interval estimated from weight-at-age data for black bears in Minnesota.

Lastly, we could calculate prediction intervals, similar to the example in Section 5 by adding +/- $2\sigma_i$, where $\sigma_i = \sqrt{\sigma^2\mu_i^{2\theta}}$ (Figure 10.9).

```
muhats <- muhats %>%
  mutate(pi.up = mu.hat + 1.95*se.mu.hat + 2*sqrt(pars$sigma^2*mu.hat^(2*pars$theta)),
         pi.low = mu.hat - 1.95*se.mu.hat - 2*sqrt(pars$sigma^2*mu.hat^(2*pars$theta)))
ggplot(beargrowth, aes(age, wtkg)) +
  geom_point() + geom_line(aes(age, mu.hat), col="red", size=1) +
  ylab("Weight (kg)") + xlab("Age (years)") + theme_bw() +
  geom_ribbon(data=muhats, aes(x=age,
                               ymin=mu.hat-1.96*se.mu.hat,
                               ymax=mu.hat+1.96*se.mu.hat),
                 inherit.aes = FALSE, fill = "blue", alpha=0.2)+
  geom_ribbon(data=muhats, aes(x=age,
                               ymin=pi.low,
                               ymax=pi.up),
                 inherit.aes = FALSE, fill = "red", alpha=0.2)
```

10.9 Likelihood ratio test

A likelihood ratio test can be used to compare nested models with the same probability generating mechanism (i.e., same statistical distribution for the response variable). Parameters must be estimated using maximum likelihood, and we must be able to get from one model to the other by setting a subset of the parameters to specific values (typically 0).

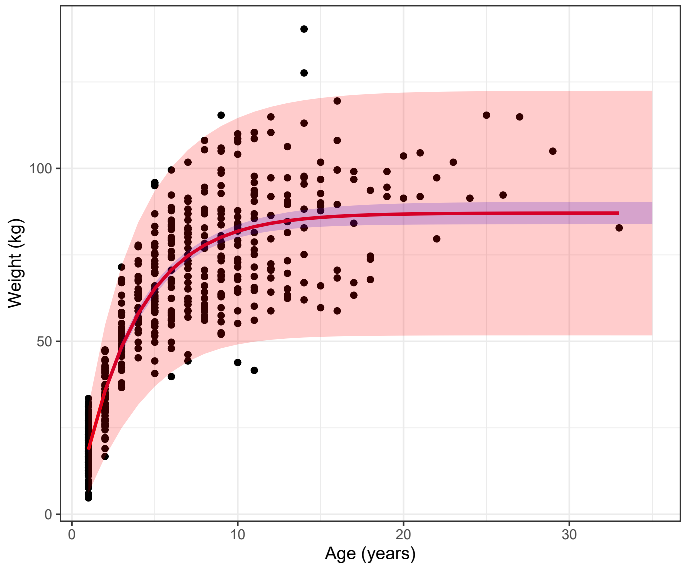


FIGURE 10.9 Fitted von Bertalanffy growth curve and 95% confidence and prediction intervals estimated from weight-at-age data for black bears in Minnesota.

In our slug example, we could use a likelihood ratio test to compare the following 2 models:

Model 1:

- $Y_{ij} \sim Poisson(\lambda_j)$ with $j = 1$ for the Nursery and $j = 2$ for the Rookery.

Model 2:

- $Y_i \sim Poisson(\lambda)$ (i.e., $\lambda_2 = \lambda_1$)

The test statistic in this case is given by:

$LR = 2\log\left[\frac{L(\lambda_1,\lambda_2;Y)}{L(\lambda;Y)}\right] = 2[\log L(\lambda_1,\lambda_2;Y) - \log L(\lambda;Y)]$

Asymptotically, the sampling distribution of the likelihood ratio test statistic is given by a χ^2 distribution with degrees of freedom equal to the difference in the number of parameters in the two models (i.e., df = 1 in our case). We can easily implement this test using the output from `optim` (recall that the negative log-likelihood values at the maximum likelihood estimates are stored in the `value` slot of `mle.fit.2lambdas` and `mle.fit`):

```
(LR<-2*(-mle.fit.2lambdas$value + mle.fit$value))
```

```
## [1] 11.42156
```

This gives us a test statistic. To calculate a p-value, we need to compute the probability of obtaining a test statistic this large or larger, if the null hypothesis is true. If the null hypothesis is true, then the likelihood ratio statistic approximately follows a χ^2 distribution with 1 degree of freedom. The p-value for a χ^2 test is always calculated using the right tail of the distribution. We can use the `pchisq` function to calculate this probability:

```
1-pchisq(LR, df=1); # or
```

```
## [1] 0.0007259673
```

```
pchisq(LR, df=1, lower.tail = FALSE)
```

```
## [1] 0.0007259673
```

We can verify this result by comparing it to what we would get if we used the `anova` function in R to compare the two models:

```
full.glm<-glm(slugs~field, data=slugs, family=poisson())
reduced.glm<-glm(slugs~1, data=slugs, family=poisson())
anova(full.glm, reduced.glm, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: slugs ~ field
## Model 2: slugs ~ 1
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1        78     213.44
## 2        79     224.86 -1  -11.422 0.000726 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

10.10 Profile likelihood confidence intervals

Typically, confidence intervals associated with maximum likelihood estimates are formed using the Normal approximation, $\hat{\theta} \sim N(\theta, I^{-1}(\theta))$. This will work well when our sample size is large but may fail to include the true parameter $(1-\alpha)\%$ of the time when n is small and the log-likelihood surface is not symmetric. As an alternative, we can "invert" the likelihood-ratio test to get a profile likelihood-based confidence interval. Here, "inverting" the test means creating a confidence interval using the parameter values for which we would not reject the null hypothesis when conducting a likelihood ratio test. The `confint` function in R will calculate profile-likelihood based intervals by default for some models (e.g., mixed effect models fit using `lmer`; Chapter 18). Thus, it is a good idea to have some familiarity with this approach. Let's demonstrate with a simple example, calculating a profile-likelihood based confidence interval for λ in our null data-generating model.

A likelihood ratio test could be used to evaluate $H_0 : \lambda = \lambda_0$ vs. $H_A : \lambda \neq \lambda_0$ for several different values of λ_0 (e.g., the values we created earlier when using a grid search to find the maximum likelihood estimator which we previously stored in an object called `lambda.test`). Our test statistic (eq. (10.7)) would, in this case, compare the negative log-likelihoods when:

- λ is set to the maximum likelihood estimator, $\hat{\lambda}$ (our unconstrained model)
- λ is set to a specific value, λ_0 (our constrained model)

$$LR = 2\log\left[\frac{L(\hat{\lambda}|Y)}{L(\lambda_0|Y)}\right] \sim \chi^2_1 \tag{10.7}$$

We would reject $\lambda = \lambda_0$ at $\alpha = 0.05$ if $LR > \chi^2_1(0.95)$, where $\chi^2_1(0.95)$ is the 95^{th} percentile of the χ^2_1 distribution. We would fail to reject the null hypotheses if $LR < \chi^2_1(0.95)$. Values of λ for which we would not reject the null hypothesis are plausible, given the data, and we will therefore want to include them in our confidence interval (Figure 10.10).

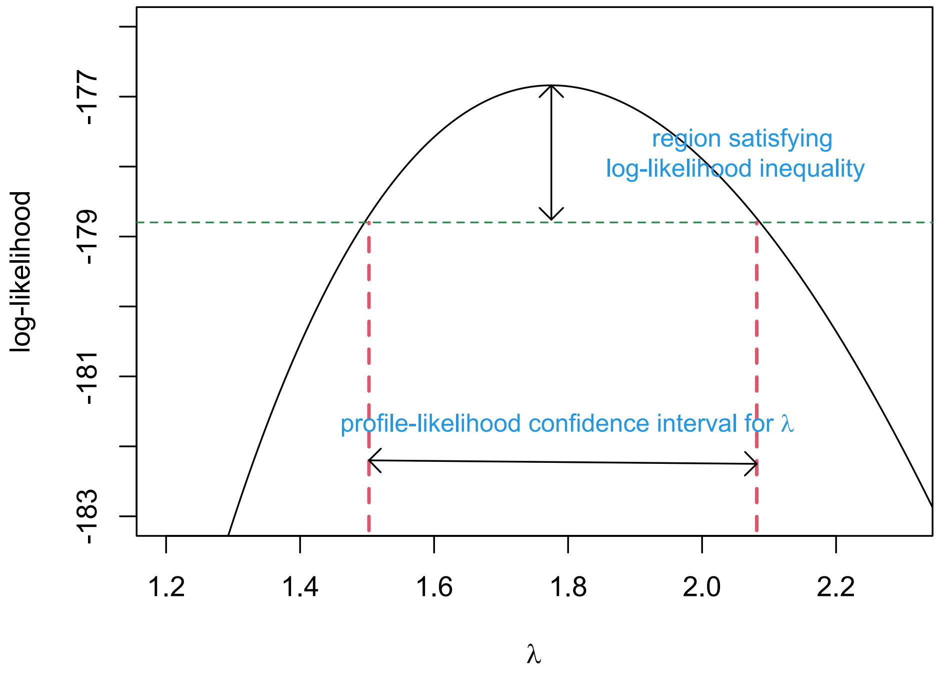


FIGURE 10.10 Profile-likelihood confidence interval for the single λ model fit to the slug data. Figure derived from code originally posted by Jack Weiss for his Statistical Ecology course.

To calculate the profile likelihood confidence interval, remember that the negative log-likelihood value at the maximum likelihood estimate is stored as `value` in the output returned by `optim` (and specifically, for this problem, as `mle.fit$value`). In addition, we already calculated the log-likelihood for several values of λ_0 in Section 10.5.2. Thus, we just need to calculate twice the difference in log-likelihood values (using $\hat{\lambda}_{MLE}$ and λ_0) and identify the values of λ_0 where $LR < \chi^2_1(0.95)$.

```
 # finds all values of lambda that are not rejected by the hypoth test
ind<-I(2*(-mle.fit$value - logL.vals) < qchisq(0.95, df=1))
(CI.95<-range(lambda.test[ind]))
```

```
## [1] 1.502803 2.081582
```

Let's compare this interval to a Normal-based confidence interval. To do so, we need to repeat the optimization but request that `optim` also return the Hessian.

```
mle.fit <-   mle.fit<-optim(par = 2,
                          fn = minus.logL.s,
                          method = "BFGS",
                          dat = slugs$slugs,
                          hessian = TRUE)
```

```
## Warning in dpois(dat, lambda, log = TRUE): NaNs produced
```

```
SE<-sqrt(solve(mle.fit$hessian))
mle.fit$par + c(-1.96, 1.96)*rep(SE,2)
```

```
## [1] 1.483049 2.066951
```

We see that, unlike the Normal-based interval, the profile likelihood interval is not symmetric and it includes slightly larger values of λ. Yet, the two confidence intervals are quite similar as might be expected given that the log-likelihood surface is nearly symmetric (Figure 10.10).

We can extend this basic idea to multi-parameter models, but calculating profile-likelihood-based confidence intervals becomes much more computationally intensive. In order to calculate a profile-likelihood confidence interval for a parameter in a multi-parameter model, we need to:

1. Consider a range of values associated with the parameter of interest (analogous to `lambda.test` in our simple example).
2. For each value in [1], we fix the parameter of interest and then determine maximum likelihood estimates for all other parameters (i.e., we need to optimize the likelihood several times, each time with the focal parameter fixed at a specific value).
3. We then use the likelihood-ratio test statistic applied to the likelihoods in [2] (our constrained model) and the likelihood resulting from allowing all parameters, including our focal parameter, to be estimated (our unconstrained model).

If we want to get confidence intervals for multiple parameters, we have to repeat this process for each of them. Alternatively, it is possible to calculate confidence regions for multi-parameter sets by considering more than 1 focal parameter at a time and a multi-degree-of-freedom likelihood-ratio χ^2 test. See e.g., B. M. Bolker (2008) for further examples and discussion and the `bbmle` package for an implementation in R (B. Bolker and R Development Core Team 2020).

10.11 Aside: Least squares and maximum likelihood

It is interesting and sometimes useful to know that the least-squares and maximum likelihood are equivalent when data are Normally distributed. To see this connection, note that the likelihood for data that are Normally distributed is given by:

$$L(\mu, \sigma^2; y_1, y_2, \ldots, y_n) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i-\mu_i)^2}{2\sigma^2}}$$

With simple linear regression, we assume $Y_i \sim N(\mu_i, \sigma^2)$ with $\mu_i = \beta_0 + X_i\beta_1$. Thus, we can write the likelihood as:

$L(\beta_0, \beta_1, \sigma; y_1, y_2, \ldots, y_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i - \beta_0 - X_i\beta_1)^2}{2\sigma^2}} = \frac{1}{(\sqrt{2\pi}\sigma)^n} e^{-\sum_{i=1}^{n} \frac{(y_i - \beta_0 - X_i\beta_1)^2}{2\sigma^2}}$

Thus, the log-likelihood is given by:

$\log L = -n\log(\sigma) - \frac{n}{2}\log(2\pi) - \sum_{i=1}^{n} \frac{(y_i - \beta_0 - X_i\beta_1)^2}{2\sigma^2}$

If we want to maximize the log-likelihood with respect to β_0 and β_1, we need to make $\sum_{i=1}^{n}(y_i - \beta_0 - X_i\beta_1)^2$ as small as possible (i.e., minimize the sum of squared errors).

11

Introduction to Bayesian statistics

Learning objectives

1. Understand differences in how probability is defined in frequentest and Bayesian statistics.
2. Understand how to estimate parameters and their uncertainty using Bayesian methods.
3. Compare Bayesian and frequentest inference, starting with a simple problem where we can solve for the Bayesian posterior distribution analytically.

As an aside, you may be surprised to learn that there are multiple ways to define probability. In fact, there are many, many more ways than two; philosophers and mathematicians have long debated the definition and meaning of probability (for an interesting discussion, see: Lyon 2010).

11.1 R packages

We begin by loading a few packages upfront:

```
library(kableExtra) # for tables
options(kableExtra.html.bsTable = T)
library(tidyverse) # for data wrangling
library(knitr) # for changing options when knitting
library(ggplot2) # for plotting
```

In addition, we will use functions from the `rgl` package (Murdoch and Adler 2021) for creating a 3-D plot that can be interacted with in the html version of the book.

11.2 Review of frequentist statistics

In frequentist statistics, we define probability in terms of the relative frequency of some event across an infinite sequence of random, repeatable experiments or random trials. This definition of probability is central to key concepts in frequentist statistics:

- a sampling distribution describes the distribution of parameter estimates across repeated events (where an event is represented by the process of collecting and analyzing data using the same methods each time).

- a null distribution is the distribution of parameter estimates across repeated events in the special situation where *the null hypothesis is true.*
- a confidence interval is an interval that should capture the true unknown parameter for a specified proportion of all events (where now, the event also includes calculating the confidence interval from the sample data).
- a p-value is the chance of obtaining a sample statistic as extreme (or more extreme than) an observed sample statistic, if the null hypothesis is true. Again, we consider repeated events of collecting data and calculating the same sample statistic under a situation where the null hypothesis is true.

In frequentist statistics,

- Parameters are generally assumed to be fixed quantities that are unknown.
- Data are random and used to estimate parameters (e.g., using Maximum Likelihood).
- Data are used to test hypotheses which are either TRUE or FALSE.

The goal is to make 'good' decisions with high probability (across potential repeated experiments).

11.3 Bayesian statistics

Bayesians also aim to make good decisions with a high probability, but as we will see, probability represents something a little different when it comes to assessing uncertainty in estimation and inference. Specifically, for Bayesians, probability refers to one's *belief* given observed data and any prior assumptions about unknown parameters. Although some have criticized Bayesian statistics on the grounds that it is subjective, it is important to stress that Bayesian analyses result in *informed beliefs* (i.e., *informed* by data), and in most cases, Bayesian and frequentist analyses will result in similar inferences. Furthermore, although prior beliefs can sometimes make a difference, they are less influential with large data sets.

In Bayesian statistics, all parameters are unknown and their uncertainty is represented by probability distributions (prior distributions before any data are collected and posterior distributions after data have been collected and analyzed).

In Bayesian statistics:

- Random variables are used to model all sources of uncertainty. Because parameters are uncertain, they will have a distribution!
- Data are treated as fixed and inference is performed conditional on the data.
- Prior assumptions are combined with a likelihood, leading to a posterior distribution that describes one's beliefs about parameters and hypotheses, conditional on the observed data.

Whereas a frequentist will tell you that hypotheses are either true or false (with probability = 1), Bayesians are comfortable quantifying P(Hypothesis | data), which will range between 0 to 1. Bayesians report credible intervals (similar to confidence intervals). These constructs are similar, but Bayesians can and will refer to the probability that a parameter is in a confidencet interval, whereas frequentists will insist that the parameter is in the interval or not (i.e., the probability is 0 or 1).

11.4 Comparing frequentist and Bayesian inference for a simple model

Let's compare Bayesian and frequentist inference using a simple example. Suppose we are interested in estimating the probability, p, that a biologist working with the Minnesota Department of Natural Resources (MN DNR) will detect a moose on the landscape when flying above it in a helicopter. We also want to provide an interval that represents our uncertainty in what p is.

From 2004-2007, the MN DNR conducted a series of detection trials using radiocollared moose to estimate p. In each trial, they flew systematic transects within a 4.0 km by 4.3 km area centered around the general vicinity of where a radiocollared moose had been located during the previous week (Figure 11.1). They recorded all moose observed within the plot and in cases where they did not locate the radiocollared moose, they went back and located the individual to determine if it was in the plot when they flew. This resulted in 124 trials during which moose were detected in 59 of them (Giudice, Fieberg, and Lenarz 2012).

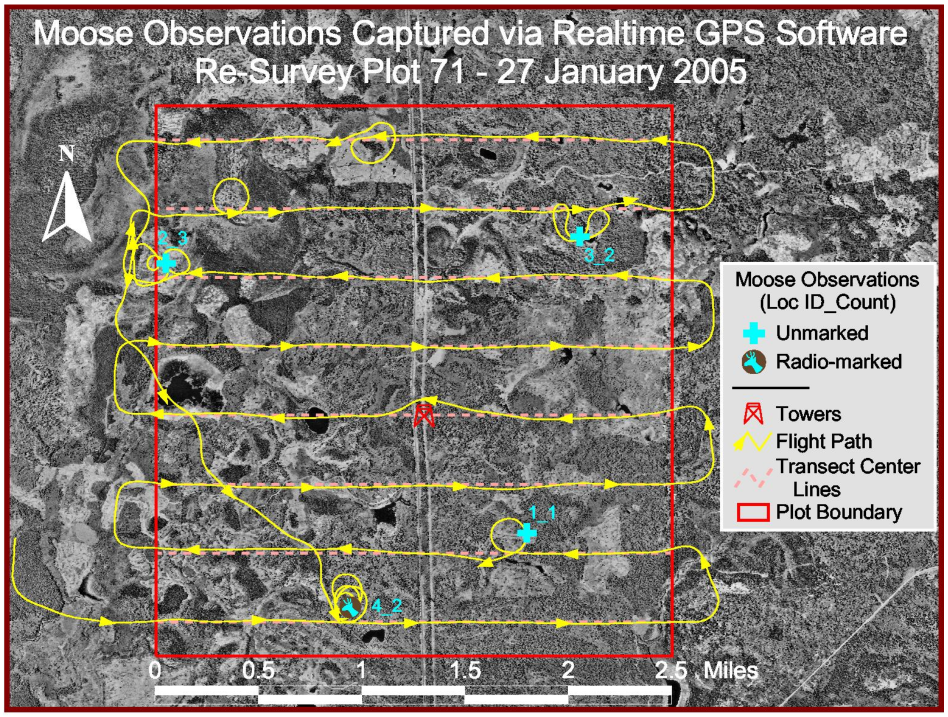

Moose Observations Captured via Realtime GPS Software
Re-Survey Plot 71 - 27 January 2005
N
2_3
3_2
1_1
4_2
Moose Observations (Loc ID_Count)
Unmarked
Radio-marked
Towers
Flight Path
Transect Center Lines
Plot Boundary
0
0.5
1
1.5
2
2.5
Miles

FIGURE 11.1 Survey flights used to estimate detection probabilities of radiocollared moose in Minnesota. Flight lines are in yellow. The moose in the lower left was not initially observed. At the end of the flight, this individual was located using the VHF signal on the collar and found to be within the flown plot. Figure created by Robert Wright, Minnesota Department of Natural Resources.

The probability of detecting a moose will depend on where the moose is located (e.g., the amount of cover that shields the moose from view, termed *visual obstruction*). We will eventually consider ways to model the probability of detection as a function of the amount of visual obstruction (Chapter 16). For now, however, we will assume that the probability of detection is constant and that all detection trials are mutually independent.

11.4.1 Maximum likelihood

If each trial is independent with constant probability of detection, then we can represent the data-generating process as a Binomial random variable:

$$Y \sim Binomial(124, p),$$

with $p =$ probability of a success (i.e., probability of seeing a moose).

The likelihood is thus given by:

$L(p; y) = \frac{n!}{y!(n-y!)}p^y(1-p)^{n-y}$

$\Rightarrow \log(L(p; y)) = log(\frac{n!}{y!(n-y!)}) + y\log(p) + (n-y)\log(1-p)$

To use Maximum Likelihood, we consider the likelihood as a function of p and choose the value of p that makes the data as likely as possible. We can maximize $\log[L(p; y)]$ with respect to p by taking the derivative of $\log(L)$ with respect to p, set the resulting expression equal to 0, and then solve for p:

$\frac{d\log(L(p;y))}{dp} = \frac{y}{p} - \frac{n-y}{1-p} = 0$

Solving for p, we obtain the Maximum Likelihood estimator, which turns out to be the sample proportion:

$\hat{p} = y/n$

We could calculate the variance directly using the result that: $Var(ay) = a^2Var(y) \Rightarrow Var(\hat{p}) = Var(y/n) = Var(y)/n^2 = p(1-p)/n$

Alternatively, we could use the asymptotic results from the section on maximum likelihood (Section 10.6) to derive an expression for $var(\hat{p})$. If we were to take second derivatives of $\log(L(p; y))$ with respect to p, we would find that:

$I(p) = E\left(-\frac{d^2\log L(p;n,y)}{dp^2}\right) = \frac{n}{p(1-p)}$. Thus, $var(\hat{p}) = I^{-1}(p) = \frac{p(1-p)}{n}$ (i.e., we end up at the same place as before!).

When our sample size, n, is large, we can form a 95% CI using:

$$\hat{p} \pm 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}.$$

Think-Pair-Share: Where does this expression come from?

In the section on maximum likelihood estimators (Section 10.6), we learned that for large n, the sampling distribution of $\hat{p}$ will be Normal, with: $\hat{p} \sim N(p, I^{-1}(p))$, and we just showed that $I^{-1}(p) = \frac{p(1-p)}{n}$. Further:

$$\hat{p} \sim N\left(p, \frac{p(1-p)}{n}\right) \implies \frac{\hat{p}-p}{\sqrt{\frac{p(1-p)}{n}}} \sim N(0,1) \tag{11.1}$$

$$\implies P(-1.96 \le \frac{\hat{p}-p}{\sqrt{var(\hat{p})}} \le 1.96) = 0.95 \tag{11.2}$$

$$\implies P\left(\hat{p} - 1.96\sqrt{\frac{p(1-p)}{n}} < p < \hat{p} + 1.96\sqrt{\frac{p(1-p)}{n}}\right) = 0.95 \tag{11.3}$$

Lastly, we approximate $\sqrt{\frac{p(1-p)}{n}}$ with $\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$, giving us the large sample confidence interval you may have seen from an introductory statistics course.

$$P(\hat{p} + 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} > p > \hat{p} - 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}) \approx 0.95$$

Let's calculate the large sample, Normal-based confidence interval in R for our moose data:

```
# Estimate and SE
y <- 54
n <- 124
(p.hat <- y/n)
```

```
## [1] 0.4354839
```

```
(se.p.hat <- sqrt(p.hat*(1 - p.hat)/124))
```

```
## [1] 0.04452597
```

```
# Confidence Interval
round(rep(p.hat, 2)+ c(-1.96, 1.96)*se.p.hat, 2)
```

```
## [1] 0.35 0.52
```

Thus, our 95% confidence interval for p is given by $(0.39, 0.56)$.

11.4.2 Aside: how well do these frequentist intervals work

The above confidence interval is based on a large-sample Normal approximation for maximum likelihood estimators. You may have learned in an introductory statistics class that these intervals work well only when $np \geq 10$ and $n(1-p) \geq 10$. Thus, these intervals may not work well with small sample sizes, particularly when p is close to 0 or 1.

We can use simulation to explore how well these intervals perform. Specifically, we will:

1. Simulate 10,000 binomial random variables using: `x <- rbinom(10,000, size = n, p)` with a range of values for `p` and `n`.
2. Generate 10,000 estimates of p for each combination of n and p, $\hat{p}$ = (`x/n`).
3. Generate 10,000 95% CIs for p.
4. Determine how many of these CIs include the true p used to generate the data.

```
# Create empty object to collect data
simulCIs = data.frame()
# Loop over values of p and n
for (n in seq(5, 200, 1))  {
    for (p in seq(.05, .95, .05)) {
      # Simulate "moose detections" (i.e. how many moose you see in the n trials)
      ys <- rbinom(10000, size = n, prob = p)
      p.hats <- ys/n
      se.p.hats <- sqrt(p.hats*(1-p.hats)/n)

      # Calculate 95% confidence intervals
      up.CIs <- p.hats+1.96*se.p.hats
      low.CIs <- p.hats-1.96*se.p.hats

      # Determine the percentage of confidence intervals that contain p
      perc.in.CI <- sum(I(low.CIs < p & up.CIs > p))/10000

      # Save n, p, and % of CI containing p into the dataframe
      simulCIs <- rbind(simulCIs,c(n,p,perc.in.CI))
    }
  }
  #name the columns in the simulation data
  names(simulCIs) <- c("n", "p", "perc.in.CI")
```

We will use the `plot3d` function in the `rgl` library (Murdoch and Adler 2021) to create a 3-D scatterplot of the results, below (if you are viewing the html version of the book, you should be able to rotate this plot):

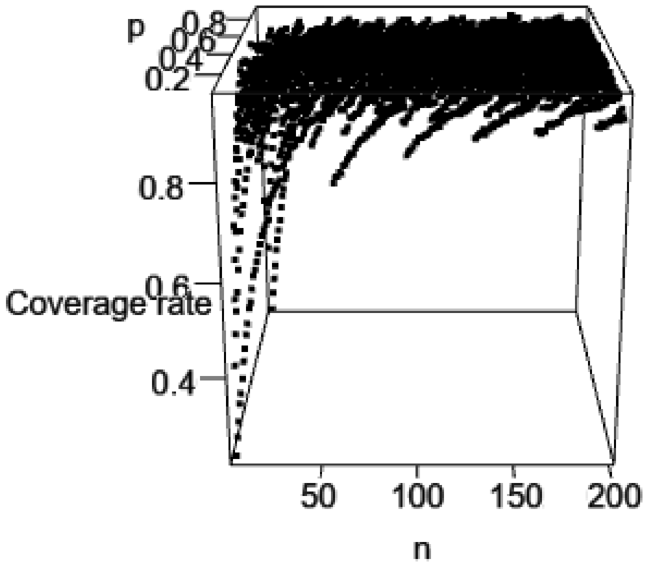


FIGURE 11.2 Coverage of large-sample, Normal-based 95 percent confidence intervals for different values of n and p when the data follow a binomial distribution.

We see that the confidence intervals perform well (close to 95% of the intervals contain the true p) except when n is really small and p is near 0 or 1.

11.4.3 Bayesian inference for p

Let's see how Bayesian Inference might differ. Here are the steps involved:

1. Specify a likelihood for the data, $L(y|p)$. Note, in Bayesian formulations, it is common to write the likelihood as a function of the data, y, conditional on the set of parameters, p. By contrast, in frequentist applications, you may see the likelihood written as a function of the parameters conditional on the data, $L(p|y)$.
2. Specify a prior distribution for the parameters, $\pi(p)$, reflecting our *a priori* belief about p.
3. Use Bayes rule (eq. (11.4)) to determine the posterior distribution of p given the data, $p(p|y)$:

$$p(p|y) = \frac{L(y|p)\pi(p)}{p(y)} = \frac{L(y|p)\pi(p)}{\int L(y|p)\pi(p)dp} \tag{11.4}$$

The posterior distribution, $p(p|y)$, captures our belief about the parameters conditional on the data we observe.

In equation (11.4), $p(y) = \int L(y|p)\pi(p)dp$ is a constant that ensures that the posterior distribution is a proper probability distribution that integrates to 1. $p(y)$ is also referred to as the *marginal* distribution of y formed by integrating $L(y|p)\pi(p)$ over p (see Section 9.8 for an introduction to marginal distributions). This integral is the continuous version of the *total law of probability* formula we saw previously in Section 9.2. In most applications, this integral will not have a closed form solution[1], so Markov Chain Monte Carlo (MCMC) will be used generate summaries of the posterior distribution (Chapter 12).

Let's now work through these three steps when analyzing the moose detection data.

1. Specify a likelihood for the data – we can again use the binomial likelihood:

$$L(y|p) = \frac{124!}{59!(124-59!)}p^{59}(1-p)^{124-59}$$

2. Specify a prior distribution for p. What should we use here? A simple way to narrow our choices is to consider the support for p (the values that p can take on), which in this case is all values between 0 and 1. This leads us naturally to one of two choices:

- a uniform distribution on the (0, 1) interval
- a beta distribution, since it is the only distribution we have learned about with support on the (0, 1) interval.

The beta distribution has two parameters, α and β, with probability density function given by:

$$\pi(p) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}p^{\alpha-1}(1-p)^{\beta-1} \tag{11.5}$$

Recall from Chapter 9, $\Gamma(z) = \int_0^\infty x^{z-1}e^{-x}dx$ generalizes the factorial function to all complex numbers. For integer values of z, $\Gamma(z) = (z-1)! = (z-1)(z-2)\cdots 3\times 2\times 1$.

The beta distribution can take on a number of different shapes depending on the values of α and β (see Fig. 11.3), and is equivalent to the uniform distribution when $\alpha = \beta = 1$ (you can verify this by plotting the Beta distribution with $\alpha = \beta = 1$ in R using `curve(dbeta(x, 1, 1), from = 0, to = 1)`).

Let's use a $Beta(1,1) = 1$ (i.e., uniform distribution) for $\pi(p)$, along with the binomial likelihood. We will then use Bayes Theorem to calculate the posterior distribution of p, $p(p|y)$.

[1]It is difficult to give a precise yet easily digested definition for *closed form solution*. I like the definition given at https://www.statisticshowto.com/closed-form-solution/, which suggests that the solution should only include: 1) a finite number of symbols, only $+ - *$ and $\div$ operators; and basic functions like exp(), log(), cos(), sin(), etc.

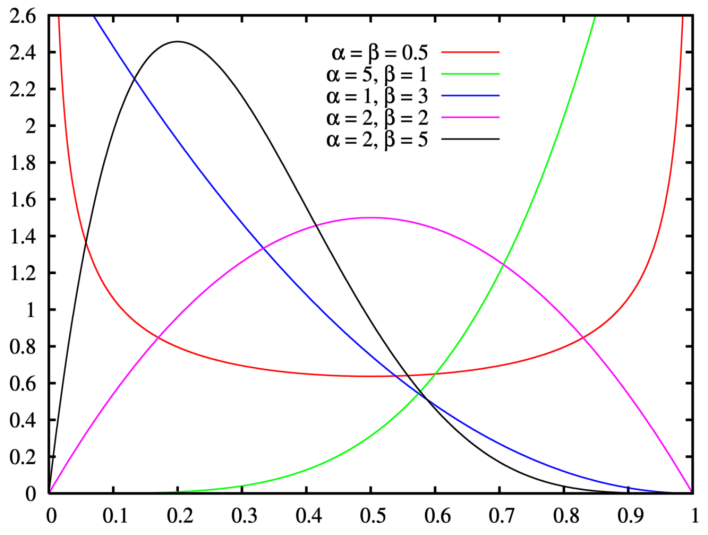


FIGURE 11.3 Probability density functions for beta distributions with different parameters, α and β.

Rather than write down the full equation for the posterior distribution, including the integral in the denominator, we will use a common trick that works when we have what is called a conjugate prior. A conjugate prior is one that results in a posterior distribution that is from the same family as the prior distribution (in this case, we will see that the posterior distribution is again a beta distribution). To see how this works, we begin by writing down an expression that is proportional to the posterior distribution:

$$p(p|y) \propto L(y|p)\pi(p) \tag{11.6}$$
$$\propto p^{59}(1-p)^{124-59}\cdot 1 \tag{11.7}$$
$$\propto p^{60-1}(1-p)^{66-1} \tag{11.8}$$

You may or may not immediately recognize that this expression is proportional to a beta distribution with parameters ($\alpha = 60$, $\beta = 66$) (eq. (11.9)):

$$p(p|y) = \frac{\Gamma(60+66)}{\Gamma(60)\Gamma(66)}p^{60-1}(1-p)^{66-1} \tag{11.9}$$

The constant, $\frac{\Gamma(60+66)}{\Gamma(60)\Gamma(66)}$, ensures that right-hand side of eq. (11.9) integrates to 1 and is therefore a proper probability distribution. This demonstrates a useful trick – if the part of the posterior distribution involving our parameters (i.e., the right-hand side of eq. (11.8)) is proportional to a known distribution, then we can safely assume that this known distribution is in fact our posterior distribution. In the above example, our posterior distribution is a Beta(60, 66) distribution. We can then use this distribution to represent our belief about the likely values for p, given the data we observed and our prior assumption that p has a uniform $(0, 1)$ distribution.

Let's use `curve` to plot the posterior distribution = beta(60, 66):

```
par(bty = "L", mar=c(2, 4.1, 1, 2.1))
# Plot the Posterior Distribution of theta
plot(curve(dbeta(x, shape1 = 60, shape2 = 66), from = 0, to = 1),
     type = "l",
     xlab = expression(theta),
     ylab = c(expression(p(group("", theta, "|") * y))))
```

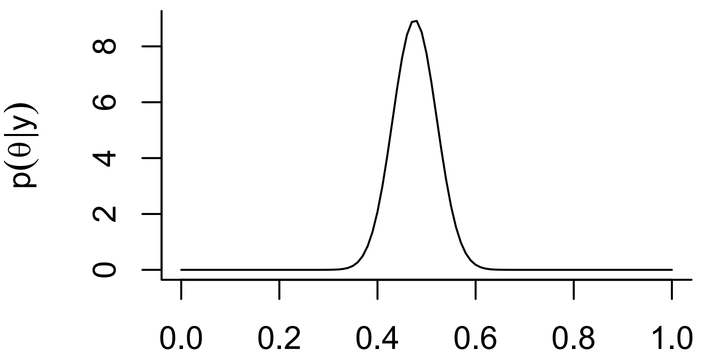


FIGURE 11.4 Posterior distribution for p, the probability of detecting a moose from a helicopter in MN surveys.

We can use this posterior distribution to form a 95% credible interval. To do so, we need to find the endpoints, x_1 and x_2 such that $P(p \leq x_1) = 0.025$ and $P(p \geq x_2) = 0.975$.

Think-Pair-Share: What function in R can we use to calculate these endpoints?

Remember, we can find quantiles of a distribution using `q*` (in this case, `qbeta`):

```
# 95% credible interval
round(qbeta(c(0.025, 0.975), shape1 = 60, shape2 = 66), 2)
```

```
[1] 0.39 0.56
```

We get the exact same endpoints as our frequentist confidence interval! Only now, we are free to say that the probability that p is between 0.39 and 0.56 is 0.95. This statement summarizes our *belief* about reasonable values of p given the observed data.

11.4.4 Comparing Interpretations: frequentist vs. Bayesian

For this simple problem, we found that our intervals were identical:

- frequentist 95% confidence interval for p = (0.39, 0.56)
- Bayesian 95% credible interval for p = (0.39, 0.56)

With the Bayesian interval, we can say that the probability that p is in the interval (0.39, 0.56) is 0.95, since this probability represents our belief about p given the data we have collected. By contrast, a frequentist will say that this particular interval either does contain p or it does not contain p (with probability = 1). The parameter itself is a fixed but unknown quantity, and the interval is random (it depends on the observed data). To interpret the frequentist interval in terms of a probability statement, we need to consider creating many such intervals across repeated events (collect data, form interval). We expect that close to 95% of these intervals would contain the fixed and unknown p (see Section 11.4.2). Thus, we say that we are 95% sure that our interval contains the true parameter p.

For many readers, the interpretation of the Bayesian interval will seem more natural and this often adds to the appeal of Bayesian methods[2]. Nonetheless, it is still important to consider the performance of Bayesian methods across repeated experiments. For example, we might use simulations to determine if Bayesian 95% credible intervals contain fixed parameters used to simulate data 95% of the time, what statistician Robert Little refers to as calibrated Bayes (Roderick J. Little 2006).

Often we will find that Bayesian and frequentist methods will lead to a similar conclusions. So, it is helpful to ask, why or when should you prefer to use Bayesian methods? Dorazio (2016) suggests Bayesian inference is most useful in the case of:

- Hierarchical models of data that "link a submodel of sampling processes with a submodel of ecological processes."
- Inference for latent (i.e., unobserved) state variables
- Missing data problems
- Intractable likelihood functions
- Complex models of different sources and types of data (with shared parameters); this is currently a hot topic in ecology where *integrated population models* (Schaub and Kéry 2021) and *integrated species distribution models* (Isaac et al. 2020) are frequently used to combine several sources of data when estimating demographic parameters and species distributions, respectively.

In addition, some of you may prefer Bayesian methods in general because:

- It is relatively easy to characterize uncertainty for functions of model parameters (no need for the bootstrap or delta method).
- The interpretation of credible intervals is appealing.
- Bayesian statistics is philosophically appealing since all inferences come from the posterior distribution. There is no need for separate theories for estimation, hypothesis testing, multiple comparisons, etc.

On the other hand, some concerns that arise when using Bayesian statistics include:

- With small samples, priors can make a big difference, and therefore, Bayesian methods can be criticized from a standpoint of perceived subjectivity. Furthermore, there are situations where even "vague" priors can end up having a significant influence on the results of an analysis (e.g., Subhash R. Lele 2020.).
- MCMC methods for estimating parameters can be computationally demanding and difficult to implement reliably (though software like JAGS (Plummer et al. 2003) and STAN (Carpenter et al. 2017) help).

These days, many applied statisticians (including me) will use a combination of Bayesian and frequentist methods. Yet, this has not always been the case, and there are still some statistical ecologists that feel strongly that Bayesian methods should not be generally adopted (see e.g., Subhash R. Lele and Dennis 2009). For example, consider the quote from Dennis (1996) (p. 1095-1103), below:

> Ecologists should be aware that Bayesian methods constitute a radically different way of doing science. Bayesian statistics is not just another tool to be added into ecologists' repertoire of statistical methods. Instead, Bayesians categorically reject various tenets of statistics and the scientific method that are currently widely accepted in ecology and other sciences. The Bayesian

[2] The same can be said for conclusions from hypothesis tests. Contrast these two conclusions: a) If H_0 is true, we would get a result as extreme as the data we saw only 3.2% of the time. Since that is smaller than 5%, we would reject H_0 at the 5% level. These data provide significant evidence for the alternative hypothesis; b) The odds in favor of H_0 against H_A are 1 to 3. This example is pulled from https://www.austincc.edu/mparker/stat/nov04/talk_nov04.pdf.

approach has split the statistics world into warring factions (ecologists' "density independence" vs "density dependence" debates of the 1950s pale by comparison), and it is fair to say that the Bayesian approach is growing rapidly in influence

12

A Brief introduction to MCMC sampling and JAGS

Learning objectives

1. Gain insights into how Markov chain Monte Carol (MCMC) sampling works.
2. Be able to implement your first Bayesian model using Just Another Gibbs Sampler (JAGS) software.

12.1 R packages

We begin by loading a few packages upfront:

```r
library(dplyr) # for data wrangling
library(knitr) # for options when knitting
library(ggplot2) # for plotting
theme_set(theme_bw()) # black and white background
library(ggthemes) # for colorblind palette
```

In addition, we will use data and functions from the following packages:

- `R2jags` package (Y.-S. Su and Masanao Yajima 2021) for calling JAGS from R
- `mcmcplots` (Curtis 2018) and `MCMCvis` (Youngflesh 2018) packages to help with visualizing MCMC results

12.2 Introduction to MCMC sampling

In our moose detection example from Section 11, we could determine the posterior distribution analytically. Usually, we will not be so lucky. In particular, in most cases, there will be no closed form solution to:

$$p(\theta|y) = \frac{L(y|\theta)\pi(\theta)}{\int_{-\infty}^{\infty} L(y|\theta)\pi(\theta)} \tag{12.1}$$

Instead, we will generate samples that we can use to summarize the posterior distribution, $p(\theta|y)$. There are many ways to generate a sample from the posterior distribution. In this section, we will briefly consider one possible way to generate a Markov chain (an autocorelated sequence of parameter values) that converges in distribution to $p(\theta|y)$. This approach, called Metropolis, is one of many possible Markov Chain Monte Carlo (MCMC) approaches used to generate samples from a posterior distribution.

12.3 Metropolis algorithm

MCMC approaches generate a sequence of parameter values using a set of rules that allow the sampler to explore the full range of support of the posterior distribution, while spending most of the time in areas where the posterior distribution is at its highest point (Figure 12.1).

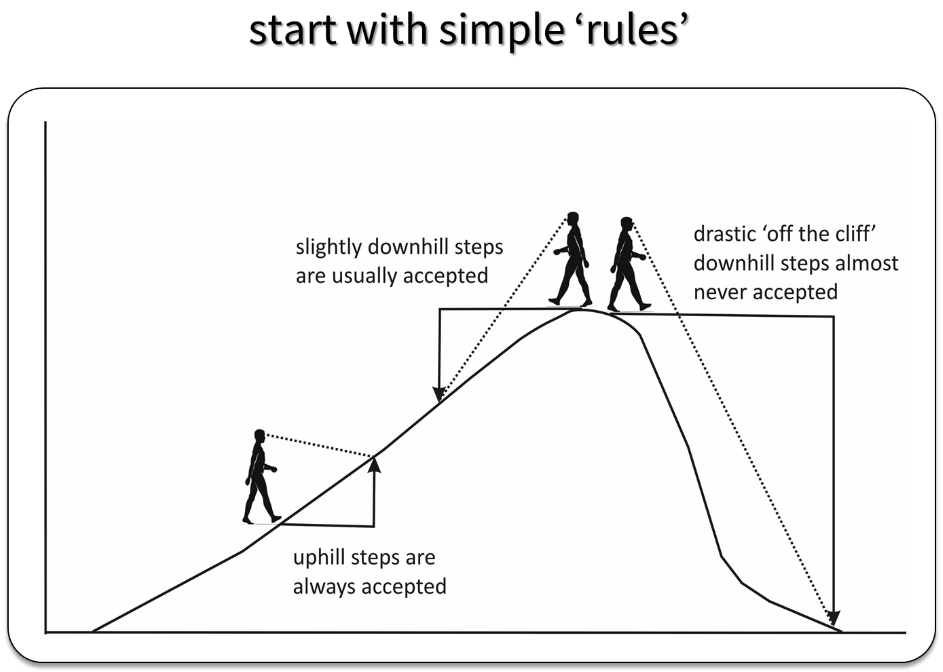

start with simple 'rules'
slightly downhill steps are usually accepted
drastic 'off the cliff' downhill steps almost never accepted
uphill steps are always accepted

FIGURE 12.1 Depiction of how the Metropolis Algorithm works. Figure created by Evan Cooch, Cornell University.

Remember, the posterior distribution is given by:

$$p(\theta|y) = \frac{L(y|\theta)\pi(\theta)}{\int_{-\infty}^{\infty} L(y|\theta)\pi(\theta)} \tag{12.2}$$

Consider two possible values of $\theta = \{\theta_1 \text{ and } \theta_2\}$. Without the denominator, we cannot evaluate $p(\theta_1|y)$ or $p(\theta_2|y)$. We can, however, evaluate the relative likelihood, R of θ_1 and θ_2 since the denominator will cancel out:

$$R = \frac{p(\theta_2|y)}{p(\theta_1|y)} = \frac{L(y|\theta_2)\pi(\theta_2)}{L(y|\theta_1)\pi(\theta_1)} \tag{12.3}$$

The Metropolis algorithm uses this relative likelihood to determine a sequence of parameter values using the following algorithm:

1. Initiate the Markov chain with an initial starting value, θ_0.
2. Generate a new, proposed value of θ from a symmetric distribution centered on θ_0 (e.g., $\theta' =$ `rnorm(mean =` $\theta_0, sd = \sigma$`))`.

3. Decide whether to accept or reject θ':
   - If $R = \frac{p(\theta'|y)}{p(\theta_0|y)} > 1$, we move up the probability hill in Figure 12.1 and accept the proposed value, setting $\theta_1 = \theta'$.
   - If $R < 1$, the proposal moves us down the probability hill. In this case, we accept θ' with probability $= R$. To do so, we generate a uniform random number, u, between 0 and 1. If $u \leq R$, then we accept the new value, setting $\theta_1 = \theta'$. Otherwise, we reject the new value and keep θ_1 at θ_0.
4. Return to step 2, replacing θ_0 with θ_1.

Note that step 3 ensures that the proposed parameter values that move us up the probability hill (towards higher values of the posterior distribution) are always accepted, but also that we do not get stuck at the top of the hill (i.e., we continue to sample from the full range of parameters supported by the posterior distribution). This strategy is not too difficult to program - if you want to see what R code would look like for a Metropolis sampler, see: https://theoreticalecology.wordpress.com/2010/09/17/metropolis-hastings-mcmc-in-r/.

We will continue to sample until:

1. The distribution of $\theta_1, \theta_2, \ldots, \theta_M$ appears to have reached a steady state (i.e., the chain has converged to an equilibrium distribution). In other words, if you were to continue to generate another M samples and then look at their distribution, it should be similar to the distribution of the first M samples.
2. The MCMC sample, $\theta_1, \theta_2, \ldots, \theta_M$, is large enough that we can summarize characteristics of the posterior distribution, $p(\theta|data)$ (e.g., its mean, median, 2.5th and 97.5th quantiles) to our desired level of precision.

An example Markov Chain may look something like Figure 12.2. This plot is called a traceplot, and it depicts the series of parameter values generated using MCMC sampling for 4 different parameters (represented in the different panels of the plot). In this case, two different Markov chains (shown in red and blue) were initiated at different starting values. Eventually, the two chains converge in distribution as evidenced by the red and blue plots largely overlapping after the first set of 500-1000 samples. Usually, we will discard the initial parameter values where the chains are clearly sampling different regions of the parameter space, which we label as a "burn-in" period. Once we have these samples, we can estimate θ by the mean (or median) of the samples, and we can compute credible intervals using quantiles of the sampled values.

How do we know if we have sampled long enough - i.e., that our sample has converged in distribution to $p(\theta|y)$? The short answer is that there is no foolproof method for detecting convergence. Some things that we can and will do, however, include:

- Run multiple chains (starting in different places) and see if they converge on similar distributions
- Discard the first n_{burnin} iterations (where the sampler has not yet converged)
- Consider the Gelman-Rubin Statistic (Gelman, Rubin, et al. 1992), Rhat, which is output by JAGS. Rhat compares the variance of parameter values between chains to the variance of parameter values within chains. Values near 1 suggest likely convergence. As a general rule, Rhat should be less than 1.1.

12.4 Aside: Sampler performance

With Metropolis or Metropolis-Hastings (the latter allows for non-symmetric proposal distributions), we need to consider how to generate good proposals for our parameters (i.e., "candidate" parameter values during

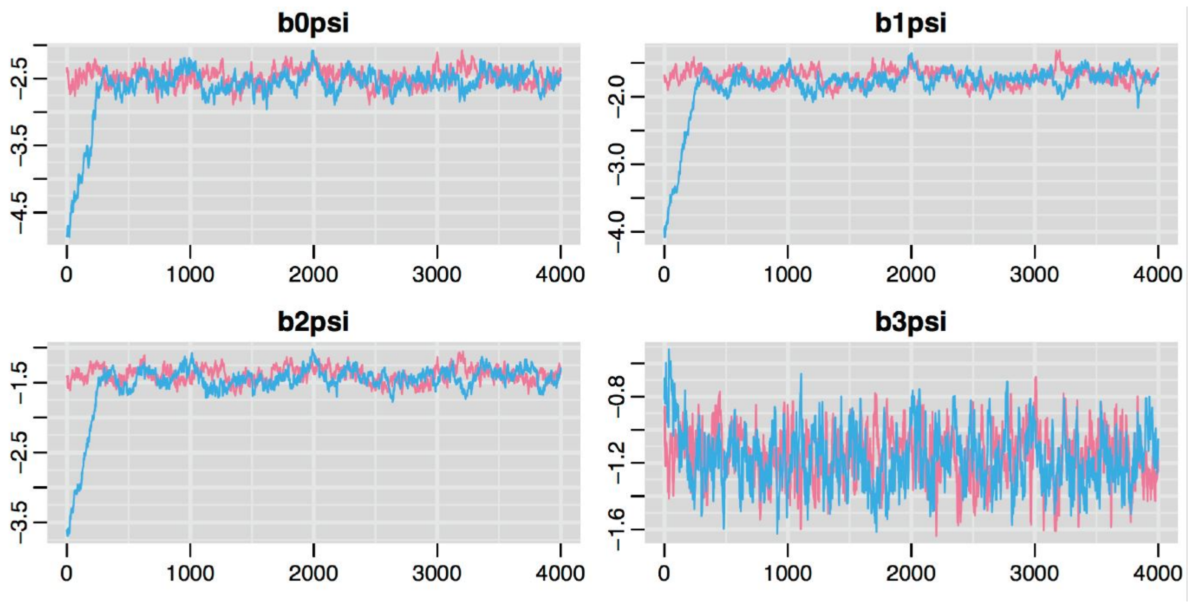


FIGURE 12.2 Traceplot depicting the series of parameter values generated using MCMC sampling. Two different Markov chains are shown in red and blue. These chains were initiated at different starting values, but converge in distribution after an initial burn in period.

step 2 of the process). This may require setting one or more "tuning" parameters to ensure the sampler does not get stuck at the top of the hill or jump too far away from the current value that we miss the hill altogether. For example, when using the normal distribution, we can try different values of σ and pick a value that ensures steps are big enough that we efficiently sample all areas of the posterior distribution without being so big that we completely "fall off the cliff" when we are near the top of the hill (Figure 12.1).

In this book, we will be using JAGS (Plummer et al. 2003) to implement Bayesian models. There are a number of packages available to run JAGS from within R (e.g., `rjags` , `runjags`, `R2jags`, `rube`, etc); we will generally use `R2Jags` (Y.-S. Su and Masanao Yajima 2021). JAGS has a variety of different MCMC algorithms at its disposal, and it will attempt to determine the best algorithm depending on the likelihood and set of prior distributions (one for each parameter) that we give it. Although JAGS is quite powerful, there are other samplers developed in recent years that are not available in JAGS but are more efficient (i.e., they converge in distribution quicker and their samples are less autocorrelated). For example, STAN (Carpenter et al. 2017), a popular alternative to JAGS, employs a "No-U-Turn" (NUTS) sampler that is typically more efficient at sampling the posterior distribution (Hoffman and Gelman 2014); for a visual demonstration of different samplers, see http://elevanth.org/blog/2017/11/28/build-a-better-markov-chain/. STAN can also be run from R (Stan Development Team 2019), but the syntax is a little more challenging to learn.

12.5 Specifying a model in JAGS

To run a model using JAGS, we will need to:

1. Specify prior distributions for all model parameters.
2. Specify the likelihood of the data.

3. Fit the model by calling JAGS from R to generate samples using its built in MCMC routines.
4. Evaluate whether or not we think the samples have converged in distribution to $p(\theta|y)$.
5. Use our samples to characterize the posterior distribution, $p(\theta|y)$.

Let's revisit our linear model for comparing the mean jaw lengths of male and female golden jackals from Section 3.6.1. The data are again provided, below:

```
males<-c(120, 107, 110, 116, 114, 111, 113, 117, 114, 112)
females<-c(110, 111, 107, 108, 110, 105, 107, 106, 111, 111)
```

As before, we will assume the mean jaw lengths for males and females are normally distributed, with sex-specific means, μ_m and μ_f. We will also, for now, assume that the jaw lengths are equally variable for males and females (Exercise 11.1 will have you will relax this assumption). This leads to the following likelihood:

- $y_{males} \sim N(\mu_m, \sigma^2)$
- $y_{females} \sim N(\mu_f, \sigma^2)$

This model has 3 parameters: μ_m, μ_f, and σ. We will need to specify prior distributions for each of these parameters. Before doing so, it is important to note that JAGS and WinBugs represent a normal distribution as $N(\mu, \tau = 1/\sigma^2)$. τ is called the precision parameter of the Normal distribution. Rather than specify a prior for the precision, it will be easier to specify a prior for σ (this will require thinking about the standard deviation rather than 1/variance). We will then use the prior for the σ to "induce" (i.e., determine) the prior for τ.

We want to choose priors that are "dispersed" (meaning they allow for a wide range of possible values). One way to do this is to use Normal distributions with large variance parameters (small precision parameters). Alternatively, we can use uniform distributions that allow for a wide range of values. We will use a mix of these strategies, below:

Priors:

- $\mu_m \sim N(100, 0.001)$
- $\mu_f \sim N(100, 0.001)$
- $\sigma \sim Uniform(0, 30)$

JAGS code (see `jaw.mod` below) looks just like R code, but with some differences:

- JAGS code is not executed (it just defines the model)
- It does not matter how we order our code (we can define the prior before likelihood or the likelihood after prior - it will not make a difference).

There are 6 types of objects in JAGS:

1. **Modeled data** specified using a $\sim$ (which can be read as "distributed as"). For example, we will use y $\sim$ followed by a probability distribution to specify the distribution of our response variable, y, in our regression model.
2. **Unmodeled data**: these are objects that are not assigned probability distributions. Examples of unmodeled data include predictors, constants, and index variables used in loops.
3. **Fixed-effects parameters**: these are parameters that are assigned uninformative priors.

4. **Random-effect parameters**: these are parameters associated with different groups that are linked together by a common prior distribution. This common prior distribution will include additional parameters called hyperparameters that are also assigned prior distributions sometimes referred to as hyperpriors. We will not see these until later in the course (Chapter 18).
5. **Derived quantities**: these are objects determined using the assignment arrow, `<-`. For example, we might be interested in obtaining samples for a function of parameters (e.g., $E[Y_i|Age_i] = L_\infty(1-e^{-k(Age_i-t_0)})$ in our bear growth example from Section 10.7). We can accomplish this goal by defining $E[Y_i|Age_i]$ as a derived quantity, depending on unmodeled data (`Age`) and modeled parameters (L_∞, k, t_0).
6. **Looping indicies**: i, j, used in loops (e.g., over our observations in a data set).

Below, we define our model for the jaw data using a `function` (by contrast, Kéry (2010) and others often write the model out to a file).

```
jaw.mod<-function(){

  # Priors
    mu.male ~ dnorm(100, 0.001) # mean of male jaw lengths
    mu.female ~ dnorm(100, 0.001) # mean of female jaw lengths
    sigma ~ dunif(0, 30) # common sigma
    tau <- 1/(sigma*sigma) #precision

  # Likelihood (Y | mu.male, mu.female, sigma) = Normal(mu[sex], sigma^2)
    for(i in 1:nmales){
      males[i] ~ dnorm(mu.male, tau)
    }
    for(i in 1:nfemales){
      females[i] ~ dnorm(mu.female, tau)
    }

  # Derived quantities:  difference in means
    mu.diff <- mu.male - mu.female
}
```

Here, we can identify the following types of objects in our model:

1. **Modeled data** = `males, females`(holding our jaw lengths)
2. **Unmodeled data** = `nmales, nfemales`
3. **Fixed-effects parameters** = `mu.male, mu.female, sigma`
4. **Random-effects parameters** (none in this example)
5. **Derived quantities** = `tau, mu.diff`
6. **Looping indexes**: `i` (used twice)

12.6 Fitting a model using JAGS

We begin by loading the `R2jags` package as well as the `mcmcplots` (Curtis 2018), `MCMCvis` (Youngflesh 2018), and `ggplot2` (Wickham 2016) packages to help with visualizing the results.

```
library(R2jags)
library(mcmcplots)
library(MCMCvis)
```

To begin sampling, JAGS will need a set of initial values (our θ_0's, for example, in step 1 of the Metropolis algorithm). Usually, JAGS can use the prior distributions to generate initial values. This will happen by default unless you supply a set of initial values. Alternatively, you can supply a function to generate initial values (this is the approach Kéry (2010) takes in his book). I tend to be lazy and allow JAGS to generate its own initial values, but this can sometimes get you in trouble (we will see this in a later section).

The function, `init.vals`, below can be used to generate initial values:

```
# Function to generate initial values
init.vals<-function(){
    mu.male <- rnorm(1, 100, 100)
    mu.female <- rnorm(1, 100, 100)
    sigma <- runif(1, 0, 10)
    out <- list(mu.male = mu.male, mu.female = mu.female, sigma = sigma)
}
```

We will also create objects to hold some of the unmodeled data that we will pass to JAGS, namely the number of males and females in the data set.

```
#' Create rest of the data for the model
nmales<-length(males)
nfemales<-length(females)
```

We then use the `jags` function to call JAGS and generate our MCMC sample:

```
t.test.jags <- jags(data=c("males", "females", "nmales",  "nfemales"),
                      inits = init.vals,
                      parameters.to.save = c("mu.male", "mu.female", "sigma", "mu.diff"),
                      progress.bar = "none",
                      n.iter = 10000,
                      n.burnin = 5000,
                      model.file = jaw.mod,
                      n.thin = 1,
                      n.chains = 3)
```

Note the following arguments:

- `data` will contain all modeled and unmodeled data (here, `males` and `females` containing the jaw lengths and `nmales` and `nfemales` containing the number of males and females).

- `parameters.to.save` is a list of parameters for which we want to save the MCMC samples. Here, we specify that we want to keep track of μ_m, μ_f, σ, and $\mu_m - \mu_f$. By contrast, we do not save τ since we do not intend to examine the posterior distribution of the precision parameter.
- `progress.bar = "none"` (for Windows users, you can also try progress.bar = `gui`). If we don't supply this argument, you will get a lot of output in your html file when using Rmarkdown.
- `n.iter = 10000` specifies the total number of samples we want to generate
- `n.burnin = 5000` specifies that we want to throw away the first 5000 samples
- `model.file = jaw.mod` specifies the function containing the model specification
- `n.thin = 1` specifies that we want to keep all of the samples. We can save memory by saving say every other sample if we change this to `n.thin = 2`. If the chains are highly autocorrelated, we won't loose much information by keeping every other sample.
- `n.chains = 3` specifies that we want to generate 3 Markov chains, each generated with a different set of starting values.

Alternatively, we could use `jags.parallel` to take advantage of parallel processing using:

```
t.test.jags <- jags.parallel(data = c("males", "females", "nmales",  "nfemales"),
                             inits = init.vals,
                             parameters.to.save = c("mu.male", "mu.female", "sigma", "mu.diff"),
                             n.iter = 10000,
                             n.burnin = 5000,
                             model.file = jaw.mod,
                             n.thin = 1,
                             n.chains = 3)
```

Let's inspect the output from jags by typing `t.test.jags` (the name of the object we created).

```
t.test.jags
```

```
## Inference for Bugs model at "jaw.mod", fit using jags,
##  3 chains, each with 10000 iterations (first 5000 discarded)
##  n.sims = 15000 iterations saved
##           mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat n.eff
## mu.diff     4.791   1.497   1.862   3.809   4.795   5.769   7.735 1.001 15000
## mu.female 108.586   1.068 106.492 107.887 108.579 109.272 110.699 1.001 15000
## mu.male   113.376   1.056 111.303 112.680 113.380 114.064 115.503 1.001  9900
## sigma       3.314   0.593   2.389   2.897   3.240   3.648   4.683 1.001 15000
## deviance  103.054   2.733  99.903 101.055 102.374 104.344 110.165 1.001 15000
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 3.7 and DIC = 106.8
## DIC is an estimate of expected predictive error (lower deviance is better).
```

We see that we get a summary of the posterior distribution for each saved parameter:

- `mu.vect` = the mean of the posterior distribution
- `sd.vect` = the standard deviation of the posterior distribution
- `2.5%` to `97.5%` = quantiles of posterior distribution
- `Rhat` is the Gelman-Rubin statistic for evaluating convergence of the MCMC samples.
- `n.eff` = an estimate of the effective sample size. As we will see in a bit, our MCMC samples will typically be autocorrelated. Thus, they will contain less information than if we were able to somehow generate a set of *independent* samples from the posterior distribution.

JAGS also returns an estimate of `DIC` (an information theoretic quantity like AIC that we will briefly discuss in Section 15.8.4).

If we only wanted to view the output for a few select parameters, we can use the `MCMCsummary` function in the `MCMCvis` package (Youngflesh 2018):

```
MCMCsummary(t.test.jags, params = c("mu.male", "mu.female"))
```

```
##               mean       sd     2.5%      50%    97.5% Rhat n.eff
## mu.male   113.3763 1.056445 111.3029 113.3805 115.5031    1 15206
## mu.female 108.5855 1.068213 106.4917 108.5786 110.6986    1 15237
```

Before interpreting the output, we should verify that we have sampled long enough that our samples have converged in distribution. Inspecting the values of `Rhat`, we see that they are all close to 1. It is also a good idea to inspect posterior density plots and trace plots to provide further reassurance that the different chains have ended up in the same spot (see Section 12.7).

We can use the `MCMCpstr` function in the `MCMCvis` package to pull off the posterior means to serve as our point estimates and to pull off quantiles of the posterior distribution to form 95% credible intervals.

```
# Use MCMCsummary to pull off posterior means
bayesests <- MCMCpstr(t.test.jags, params = c("mu.male", "mu.female"), func = mean)
bayesests
```

```
## $mu.male
## [1] 113.3763
##
## $mu.female
## [1] 108.5855
```

```
# Use MCMCsummary to pull of upper and lower 95% credible interval limits
bayesci <-  MCMCpstr(t.test.jags, params = c("mu.male", "mu.female"),
                        func = function(x) quantile(x, probs = c(0.025, 0.975)))
bayesci
```

```
## $mu.male
##             2.5%    97.5%
## mu.male 111.3029 115.5031
##
## $mu.female
##               2.5%    97.5%
## mu.female 106.4917 110.6986
```

We then fit the same model in a frequentist framework using `lm` and create a data set with our estimates and credible/confidence intervals.

```
# Fit linear model in frequentist framework
jawdat<-data.frame(jaws=c(males, females),
                    sex=c(rep("Male", length(males)),
                          rep("Female", length(females))))
lm.jaw<-lm(jaws~sex-1, data=jawdat)
# Store results
betaf <- coef(lm.jaw)
CIf <-confint(lm.jaw)
ests<-data.frame(estimate = c(bayesests$mu.female, bayesests$mu.male, betaf),
                 LCL = c(bayesci$mu.female[1],   bayesci$mu.male[1], CIf[,1]),
                 UCL = c(bayesci$mu.female[2],   bayesci$mu.male[2], CIf[,2]),
                 param = c("Mean females", "Mean males"),
                 Method = rep(c("Bayesian", "Frequentist"), each = 2))
```

Lastly, we plot the results, showing that they are nearly identical (Figure 12.3)!

```
ggplot(ests, aes(param,estimate, col = Method)) +
  geom_point(position = position_dodge(width = 0.2))+
  geom_pointrange(aes(ymin = LCL, ymax= UCL), position = position_dodge(width = 0.2))+
  ylab("Estimate") + xlab("") +
  scale_x_discrete(labels = c('Mean females' = expression(mu[f]),
                              'Mean males'   = expression(mu[m]))) +
  scale_colour_colorblind()+
  theme(text = element_text(size = 20))
```

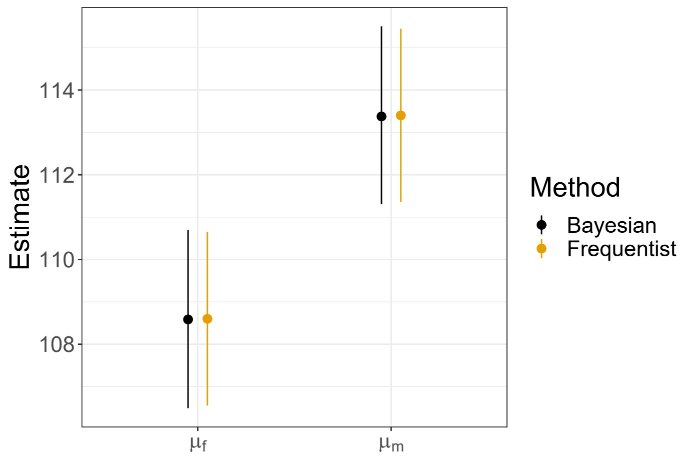


FIGURE 12.3 Comparison of frequentist and Bayesian estimates and confidence/credible intervals for the mean jaw length of male and female jackals.

12.7 Density and traceplots for assessing convergence

When running code in Rstudio, I often like to use the `mcmcplot` function to visualize output. This function will create a separate html file that you can use to visualize posterior distributions, autocorrelation functions, and traceplots for the different parameters. This function does not work well with the `bookdown` package Xie (2022) used to create this book, however, so we will explore these different plots separately.

We can visualize posterior densities using `denplot`:

```
denplot(t.test.jags, ask = FALSE)
```

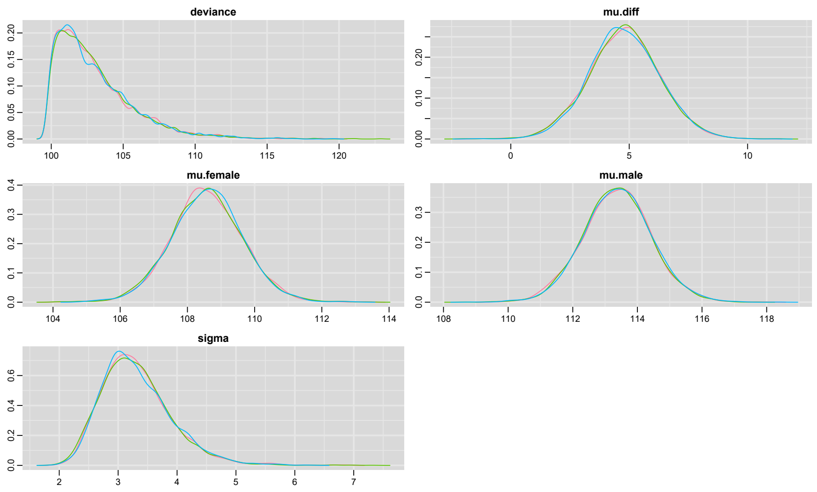


FIGURE 12.4 Posterior density plots for each parameter in our model fit to the jackal jaw length data set. Colors in the different panels correspond to different chains.

In each panel, we see density plots for each chain associated with a particular parameter as well as density plots for the deviance (we can also pool results across the different chains if we add `collapse = TRUE`). For this simple model, it is clear that all three chains ended up in the same place, providing further evidence that we have reached convergence. If we had a large number of parameters, then we could have added the `parms` argument to provide a list containing a subset of parameters to visualize. For example, we could visualize the posteriors for μ_m and μ_f using `denplot(t.test.jags, parms = c("mu.male", "mu.female"))`[1].

We can also look at traceplots showing the sequence of posterior samples using:

```
traplot(t.test.jags, ask=FALSE)
```

Again, the fact that the different chains largely overlap is reassuring. Sometimes, traceplots will not look so noisy - but, rather you will see a slow moving trend over time. When this happens, the samples are highly autocorrelated, and we will have to sample for a long time in order to explore the full range of the posterior distribution. By contrast, these traceplots suggest that the sample is doing a great job quickly exploring the full range of the posterior distribution, and we say the chains are "well mixed".

[1]Note, the functions in the `mcmcplots` package include the argument `parms`, whereas the functions in the `MCMCvis` package include the argument `params`. It is easy to get these confused. If you use the tab key in Rstudio after typing "par", you can have the function self-populate with the correct argument.

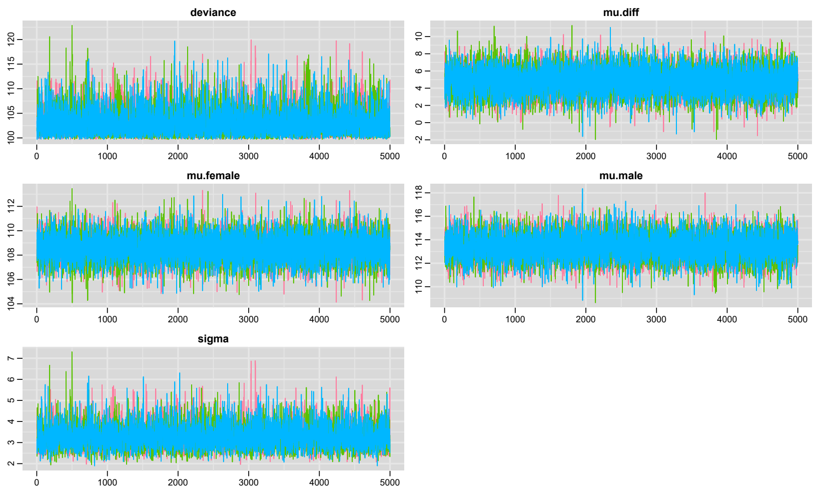


FIGURE 12.5 Trace plots for each parameter in our model fit to the jackal jaw length data set. Colors in the different panels correspond to different chains.

We can look to see if our parameters are correlated with each other using `parcorplot`.

```
parcorplot(t.test.jags)
```

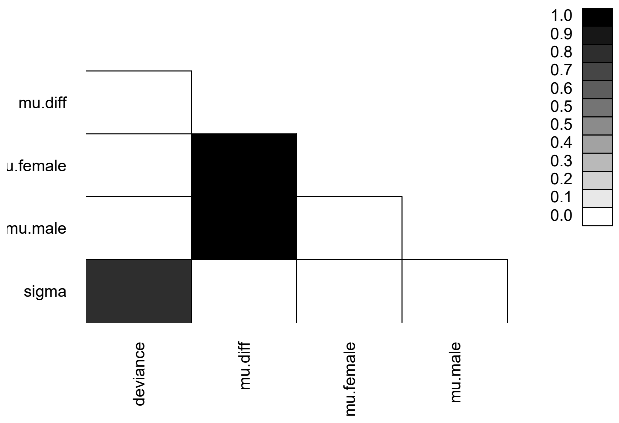


FIGURE 12.6 Correlation plot formed using the `parcorplot` function for inspecting whether the posterior distributions are correlated for different parameters (Figure @ref(fig:(ref:corplotbayes)).

Lastly, we can plot estimates with 68% and 95% credible intervals using the `caterplot` function (Figure 12.7)[2].

```
  caterplot(t.test.jags, ask = FALSE,
            parms = c("mu.male", "mu.female"))
```

Congrats - you have just fit your first JAGS model and learned how to work with and summarize samples from the posterior distribution!

[2]Note, for a Normal distribution, a 68% interval is given by ±1SD.

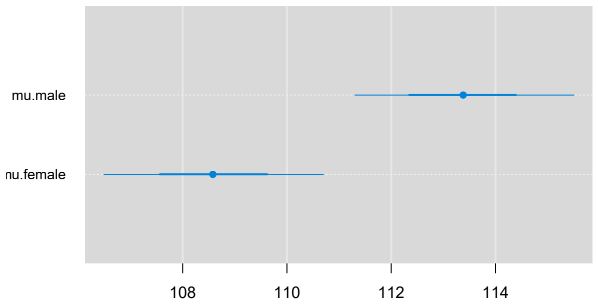


FIGURE 12.7 Visualization of 68% and 95% credible intervals using the `caterplot` function applied to the Bayesian model fit to the jaw length data.

12.8 Tips on running models in JAGS

Kéry (2010) offers lots of good tricks and tips in the Appendix of his book. First and foremost: *start simple, then build up towards more complex models!* If you have his book, we encourage you to look at his tips numbered: 2, 3, 4, 9, 11, 12 (use `%T%` in JAGS), 14, 16, 17, 20, 23,24, 25, 26, 27.

Also, as always, googling error messages can be useful for diagnosing problems, and it can help to work together with your peers when first learning to program in JAGS. It is extremely rare to specify everything correctly the first time when fitting a model using JAGS. That is OK. It is important to have failures followed by successes to give future you the confidence needed to stick with it!

13

Bayesian linear regression

Learning objectives:

- Understand how to fit a linear model using JAGS
- Learn how to evaluate goodness-of-fit of models using Bayesian p-values
- Be able to calculate credible and prediction intervals for linear models using JAGS

13.1 R packages

We begin by loading a few packages upfront:

```r
library(R2jags) # for interfacing with R
library(MCMCvis) # for summarizing MCMC output
library(mcmcplots) # for plotting MCMC output
library(ggplot2) # for other plots
library(patchwork) # for multi-panel plots
library(dplyr) # for data wrangling
```

In addition, we will use data and functions from the following packages:

- `abd` (Middleton and Pruim 2015) to access the `LionNoses` data set

13.2 Bayesian linear regression

Let's begin by reconsidering the regression model from Section 1 relating the age of lions to the amount of black pigmentation in their noses:

$$\begin{aligned} Age_i &\sim N(\mu_i, \sigma^2) \\ \mu_i &= \beta_0 + \beta_1 Proportion.black_i \end{aligned} \tag{13.1}$$

Think-pair-share: How many priors will we need to specify when fitting this model?

In this simple model, we have three parameters for which we need to specify priors: β_0, β_1, and σ^2. You might be temped to also list μ_i, but μ_i can be derived from β_0, β_1, and our data (i.e., `proportion.black`), and therefore, we do not need a prior for it. In this book, we will try to specify "non-informative" priors for our parameters – meaning that we do not want our choice of prior to have a large influence on our resulting parameter estimates and their uncertainty. This may not always be simple to do, and in general, it will be wise to:

1. Give careful consideration to the potential values that the parameters may take on and ensure your prior distribution covers this range of values.
2. Compare prior and posterior distributions to determine if your prior may have been overly constraining or informative (see Exercise 11.3 here).
3. Try multiple prior distributions as a form of sensitivity analysis (see Exercise 12.2 here).

Using non-informative priors will allow us to generally arrive at similar conclusions when analyzing data using Bayesian and frequentist methods, which will hopefully allow you to gain confidence in using both approaches. Nonetheless, there are certainly other ways to approach Bayesian modeling. In some situations, you may have additional data that you want to have inform your prior distributions (many Bayesians would certainly advocate for such an approach). Historically, priors were sometimes chosen to "play well" with the likelihood of the data – e.g., to give a closed form solution for the posterior distribution. This leads to choosing a conjugate prior – as we did in our initial example in Section 11.4.3. Priors can also influence the choice of MCMC sampling algorithm and the performance of MCMC samplers. We will not cover these topics here, but they can be of extreme importance, especially when constructing more complex models.

For regression parameters in linear regression models, we will usually use "vague" Normal priors[1], setting the mean equal to 0 and the precision to a small value (e.g., 0.01 or 0.001, which is equivalent to setting the variance to 100 or 1000). We will generally use uniform $(0, b)$ distributions as a simple and easy way to specify a prior for σ. As in our "Bayesian t-test" example from Section 12.5, this prior distribution for σ will induce a prior distribution for τ, the precision parameter of our Normal distribution.

Below, we specify our JAGS model for the lion data. For now, ignore the code below the likelihood function. We will discuss it when describing a method for performing a goodness-of-fit test for our model (Section 13.4).

```
# Write model
linreg<-function(){

# Priors
 beta0 ~ dnorm(0,0.001)
 beta1 ~ dnorm(0,0.001)
 sigma ~ dunif(0, 100)

# Derived quantities
 tau <- 1/ (sigma * sigma)

# Likelihood
 for (i in 1:n) {
    age[i] ~ dnorm(mu[i], tau)
    mu[i] <- beta0 + beta1*proportion.black[i]
 }

# Assess model fit using a sums-of-squares-type discrepancy
 for (i in 1:n) {
```

[1]The meaning of vague here is that the prior distribution allows for a wide range of potential values.

```
    residual[i] <- age[i]-mu[i]       # Residuals for observed data
    sq[i] <- pow(residual[i], 2)    # Squared residuals for observed data

# Generate replicate data and compute fit stats for them
    y.new[i] ~ dnorm(mu[i], tau) # one new data set at each MCMC iteration
    sq.new[i] <- pow(y.new[i]-mu[i], 2)  # Squared residuals for new data
 }
 fit <- sum(sq[])           # Sum of squared residuals for actual data set
 fit.new <- sum(sq.new[])       # Sum of squared residuals for new data set
}
```

We then bundle our data and call JAGS:

```
#Load the data
library(abd)
data("LionNoses")

# Bundle data
jags.data <- list(age = LionNoses$age,
                  proportion.black = LionNoses$proportion.black,
                  n = nrow(LionNoses))

# Parameters to estimate
params <- c("beta0", "beta1", "sigma", "mu", "y.new", "fit", "fit.new", "residual")

# MCMC settings
nc = 3
ni = 2200
nb = 200
nt = 1

# Start Gibbs sampler
lmbayes <- jags(data = jags.data,  parameters.to.save = params, model.file = linreg,
n.thin = nt, n.chains = nc, n.burnin = nb, n.iter = ni, progress.bar="none")
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 32
##    Unobserved stochastic nodes: 35
##    Total graph size: 295
##
## Initializing model
```

Let's take a look at the output and compare our results to a frequentist analysis using `lm`:

```
lmnoses <- lm(age ~ proportion.black, data = LionNoses)
summary(lmnoses)
```

```
##
## Call:
## lm(formula = age ~ proportion.black, data = LionNoses)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5449 -1.1117 -0.5285  0.9635  4.3421
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.8790     0.5688   1.545    0.133
## proportion.black  10.6471     1.5095   7.053 7.68e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.669 on 30 degrees of freedom
## Multiple R-squared:  0.6238, Adjusted R-squared:  0.6113
## F-statistic: 49.75 on 1 and 30 DF,  p-value: 7.677e-08
```

```
MCMCsummary(lmbayes, params = c("beta0", "beta1", "sigma"))
```

```
##            mean        sd       2.5%        50%     97.5% Rhat n.eff
## beta0  0.868424 0.5959412 -0.3139261  0.8691696  2.051789    1  5975
## beta1 10.655666 1.5827939  7.5489726 10.6555606 13.830992    1  6233
## sigma  1.737433 0.2291343  1.3457192  1.7130554  2.247658    1  3342
```

We see that we get similar results using both methods. That should be reassuring.

13.3 Testing assumptions of the model

As we saw in Section 1.5, there are lots of different R packages and functions available for creating residual diagnostic plots for linear regression models. We can build similar plots using the output from JAGS, but it is important to recognize that each residual has its own posterior distribution; we will usually want to summarize these posterior distributions using their mean or median values before creating diagnostic plots.

Think-Pair-Share: Why does each residual have a posterior distribution?

Consider this line of code in our JAGS model:

`residual[i] <- age[i]-mu[i]`

With each chain, we generated `ni = 2200` samples from the posterior distribution of `beta0` and `beta1`. Thus, with each chain, we end up with 2200 samples from the posterior distribution of each `mu[i]` and 2200 samples from the posterior distribution of each `residual[i]`.

The `MCMCpstr` function in the `MCMCvis` package (Youngflesh 2018) provides a convenient way to access the mean (or other summaries) of the posterior distribution for any parameter. Below, we show how to access the posterior mean of the residuals (`residual`) and fitted values (`mu`) so that we can create residual versus fitted value plots, QQ-plots for assessing Normality, and a scale-location plot for evaluating homogeneity of variance (Figure 13.1).

```
lmresids <- MCMCpstr(lmbayes, params = "residual", func = mean)
lmfitted <- MCMCpstr(lmbayes, params = "mu", func = mean)
jagslmfit <- data.frame(resid = lmresids$residual, fitted = lmfitted$mu)
jagslmfit$std.abs.resid <- sqrt(abs(jagslmfit$resid/sd(jagslmfit$resid)))
```

And, here we create our plots:

```
p1 <- ggplot(jagslmfit, aes(fitted, resid)) + geom_point() +
         geom_hline(yintercept = 0) + geom_smooth()
p2 <- ggplot(jagslmfit, aes(sample = resid)) + stat_qq() + stat_qq_line()
p3 <- ggplot(jagslmfit, aes(fitted, std.abs.resid)) + geom_point() +
         geom_smooth() + ylab("sqrt(|Standardized Residuals|)")
p1 + p2 + p3
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

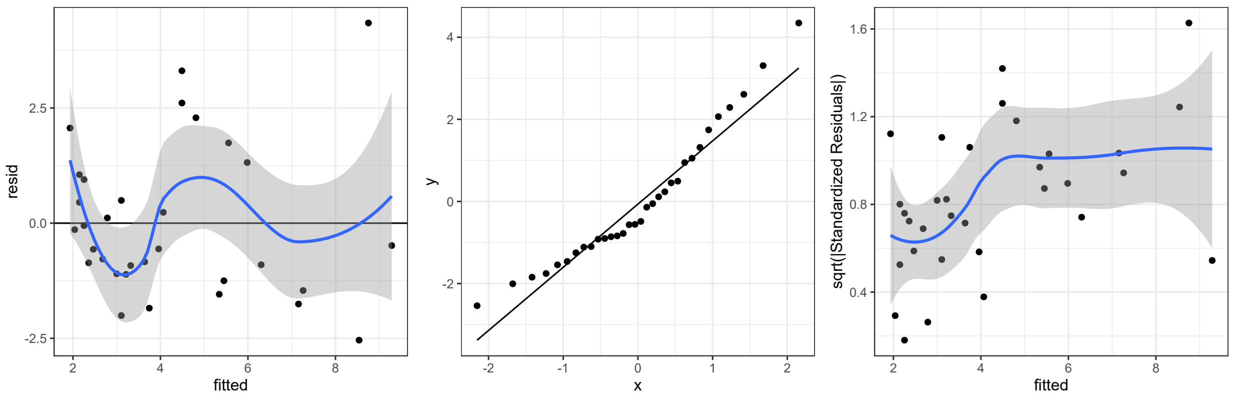


FIGURE 13.1 Residual diagnostic plots for a linear model relating the age of lions to the proportion of their nose that is black fit in a Bayesian framework. Plots depict posterior means of the residual and fitted values.

We can compare Figure 13.1 to similar plots constructed using the output from the frequentist regression using `lm` (Figure 13.2):

```
ggResidpanel::resid_panel(lmnoses, plots = c("resid", "qq", "ls"), nrow = 1, smoother = TRUE)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```

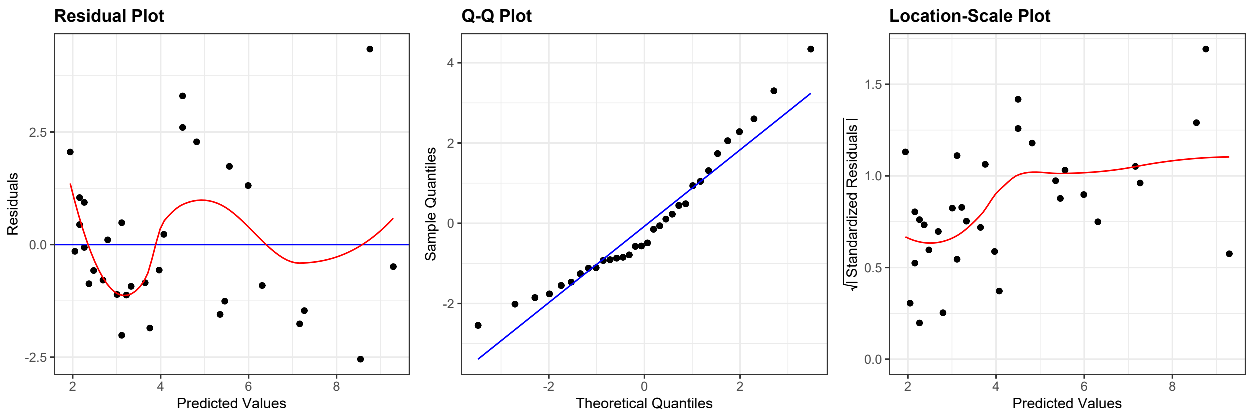


FIGURE 13.2 Residual diagnostic plots associated with a linear model relating the age of lions to the proportion of their nose that is black fit in a frequentist framework.

13.4 Goodness-of-fit testing: Bayesian p-values

In addition to standard residual plots, it is often useful to conduct a formal goodness-of-fit test when evaluating a fitted model. In this section, we will demonstrate a common goodness-of-fit test used to evaluate Bayesian models[2]. Let's consider the steps involved in any hypothesis test as outlined in Section 1.10.3.

1. First, we need to state null and alternative hypotheses. As with any goodness-of-fit test, our hypotheses may be stated as:

H_0 : the data are consistent with the assumed linear model (eq. (13.1)).

H_A : the data are not consistent with the assumed model.

2. Next, we need a *test statistic*, T, which will measure the degree to which our data are consistent with the null hypothesis. You are free to choose any test statistic that you want, but ideally the statistic should capture deviations from the null hypothesis that you deem to be important. For example, one might choose to use the number of zeros in a data set as the test statistic when concerned that data may be zero-inflated (Section 17.3). Here, we will use the sum of squared residuals as a general measure of overall fit:

$$T_{obs} = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

3. Next, we need to determine the *sampling distribution* of the test statistic when the null hypothesis is true. We can generate data sets from our assumed model and calculate T for each of these data sets to form this sampling distribution.
4. Lastly, we compare our test statistic calculated using the observed data, T_{obs}, to the sampling distribution of T when when the null hypothesis is true. The p-value is the probability, when the null hypothesis is true, of getting a test statistic, T_{H_0}, that is as extreme or more extreme than

[2]A similar approach can be implemented for frequentist models as we will see in Section 15.4.5.

T_{obs}. Ideally, this p-value will be large, suggesting that we do not have much evidence to conclude our assumed model of the data generating process is a poor one.

Now, let's go back and look at key parts of our specification of the JAGS model, highlighted below:

```
# Assess model fit using a sums-of-squares-type discrepancy
 for (i in 1:n) {
    residual[i] <- age[i]-mu[i]       # Residuals for observed data
    sq[i] <- pow(residual[i], 2)    # Squared residuals for observed data

# Generate replicate data and compute fit stats for them
    y.new[i] ~ dnorm(mu[i], tau) # one new data set at each MCMC iteration
    sq.new[i] <- pow(y.new[i]-mu[i], 2)  # Squared residuals for new data
 }
 fit <- sum(sq[])           # Sum of squared residuals for actual data set
 fit.new <- sum(sq.new[])       # Sum of squared residuals for new data set
```

Think-Pair-Share: Which lines of code implement steps 2 and 3 of the hypothesis test?

- We calculate T_{obs} using: `fit <- sum(sq[])`.
- We generate new data sets where the null hypothesis is true using: `y.new[i] ~ dnorm(mu[i], tau)`
- We calculate T_{H_0} using: `fit.new <- sum(sq.new[])`

There is one new wrinkle in all of this, namely that $\hat{Y}_i$ has a posterior distribution, which means that T_{obs} also has a posterior distribution. Nonetheless, we can calculate a Bayesian p-value by comparing the posterior distributions of T_{obs} and T_{H_0}. Specifically, we calculate the Bayesian p-value as:

$$\text{p-value} = \sum_{k=1}^{M} \frac{T^k_{H_0} \geq T^k_{obs}}{M}$$

where k indexes the M different posterior samples generated using JAGS. Note, if our model fits poorly, then we should expect that the sums-of-squared residuals associated with our observed data (T_{obs}) will be large relative to the sums-of-squared residuals for data generated under the assumed model (T_{H_0}). In this case, $T^k_{H_0}$ will rarely be greater than T^k_{obs}, we will end up with a small p-value, and we will reject the null hypothesis that the data are consistent with the assumed model. If you get a p-value near 0.5, that is ideal as it suggests that the observed data look very similar to the data generated under the assumed model (at least when it comes to our chosen test statistic). You might wonder at what point (i.e., what threshold for the p-value) should you become concerned about the reliability of your model. I hesitate to give hard and fast rules, but suggest that a p-value less than 0.05 or 0.1 should give you pause[3].

Below, we calculate this p-value for our linear regression model applied to the `LionNoses` data set by:

1. Extracting the MCMC samples using the `MCMCpstr` function in the `MCMCvis` package.
2. Creating an indicator variable that is equal to 1 whenever $T^k_{H_0} \geq T^k_{obs}$ and 0 otherwise.
3. Calculating the mean of this indicator variable (note, a mean of a 0-1 variable is the same as the proportion of times the variable is equal to 1).

[3]Note that it is possible, though fairly unlikely, that a model will fit the observed data much better than data generated from the assumed model, resulting in a Bayesian p-value that is close to 1. In this case, it is also worth digging a little deeper to see why this may have occurred.

```r
fitstats <- MCMCpstr(lmbayes, params = c("fit", "fit.new"), type = "chains")
T.extreme <- fitstats$fit.new >= fitstats$fit
(p.val <- mean(T.extreme))
```

```
## [1] 0.528
```

We see that we have a large p-value, suggesting that we do not have strong evidence that our data are inconsistent with the assumed model. That is great news, though we need to keep in mind that our power to detect deviations from the null hypothesis may be low due to our small sample size. In addition, because the data are used twice (once to inform the posterior distribution of our model parameters and then to evaluate fit), these tests tend to produce larger p-values than are generally warranted (see e.g., Conn et al. 2018). I.e., we would expect the model to perform more poorly when evaluated against independent data.

Instead of focusing on the numeric p-value, a former student in one of my courses suggested visualizing the posterior distribution of $T_{H_0} - T_{obs}$ (Figure 13.3), which I think is instructive. To create this posterior distribution, we begin by storing the chains in matrix form using the `MCMCchains` function. We then use the `mutate` function to create a new variable representing the difference in goodness-of-fit values, with positive values indicating that the model provides a better fit to observed data than the simulated data. We then visualize the p-value (Figure 13.3) using the `ShadedDensity` function in the `WVPlots` package (Mount and Zumel 2021).

```r
#' Posterior for the difference in fits
fitstats <- MCMCchains(lmbayes, params = c("fit", "fit.new"))
fitstats <- as.data.frame(fitstats) %>%
  mutate(postdiff = fit.new-fit)
WVPlots::ShadedDensity(frame = fitstats,
                       xvar = "postdiff",
                       threshold = 0,
                       title = "Posterior distribution: SSE(sim data)-SSE(obs data)",
                       tail = "right")+
  annotate("text", x=85, y = 0.005,
           label="Better fit to observed data")+
  annotate("text", x=-40, y = 0.005,
           label="Better fit to simulated data")
```

The p-value is equal to the area associated with values > 0, which corresponds to situations where the simulated data have larger sum-of-squared residuals than the observed data.

13.5 Credible intervals

One of the nice things about Bayesian inference is that we can calculate uncertainty for nearly any quantity using posterior distributions of our model parameters. For example, if we want to get predictions of the mean age of lions at a set of values for `proportion.black`, we can use the posterior distributions for β_0 and β_1. Below, we demonstrate how one can accomplish this task using a `for` loop to get predictions for 10 equally spaced values of `proportion.black` over the range of the observed data.

First, we pull off and store the MCMC samples of β_0 and β_1 using the `MCMCpstr` function with `type = "chains"` in the `MCMCvis` package.

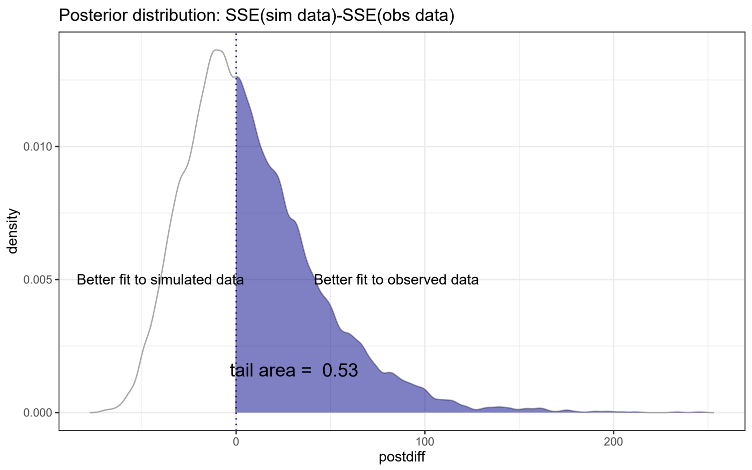


FIGURE 13.3 Posterior distribution of $T_{H_0} - T_{obs}$, where T_{H_0} represents the sum-of-squared residuals for data simulated from the assumed model and T_{obs} represents the sum-of-squared residuals for the observed data. Values greater than 0 indicate a better fit to the observed than simulated data.

```
betas <- MCMCpstr(lmbayes, params = c("beta0", "beta1"), type = "chains")
```

We see that we have a total of 6000 MCMC samples from the posterior distribution of (β_0, β_1) (i.e., 3 chains x 2000 samples per chain):

```
dim(betas$beta0)
```

```
## [1]    1 6000
```

Next, we create a vector of values for `proportion.black` for which we desire predictions:

```
prop.black <-seq(from = min(LionNoses$proportion.black),
                 to = max(LionNoses$proportion.black),
                 length = 10)
```

We then loop over the 10 different values of `prop.black`, estimate the mean age using each of our MCMC samples of β_0 and β_1, and then find the 2.5 and 97.5 quantiles of the resulting estimates to form our 95% credible interval for each value of `prop.black`:

```
# Number of MCMC samples
nmcmc <- dim(betas$beta0)[2]

# Number of unique values of proportion.black where we want predictions
nlions <- length(prop.black)

# Set up a matrix to hold 95% credible intervals for each value of prop.black
conf.int <- matrix(NA, nlions, 2)

# Loop over values for proportion.black
```

```r
for(i in 1:nlions){
  # Estimate the mean age for the current value of prop.black and for
  #   each MCMC sample of beta0 and beta1. Note: rep(prop.black[i], nmcmc)
  #   creates a vector of the current prop.black values that is equal in
  #   length to the vector of betas from the MCMC sampler.
  age.hats <- betas$beta0 + rep(prop.black[i], nmcmc)*betas$beta1
  conf.int[i,] <- quantile(age.hats, prob = c(0.025, 0.975))
}
```

Next, we combine our credible interval with predictions for each age using the mean of the posterior distribution of β_0 and β_1 and plot the results (Figure 13.4).

```r
beta.hat <- MCMCpstr(lmbayes, params = c("beta0", "beta1"), func = mean)
phats <- data.frame(est = rep(beta.hat$beta0, nlions) + rep(beta.hat$beta1, nlions)*prop.black,
                    LCL = conf.int[,1],
                    UCL = conf.int[,2], proportion.black = prop.black)
ggplot(phats, aes(proportion.black, est)) +
  geom_ribbon(aes(ymin = LCL, ymax = UCL), fill = "grey70", alpha = 2) +
  geom_line() +ylab("Age") +
  geom_point(data = LionNoses, aes(proportion.black, age))
```

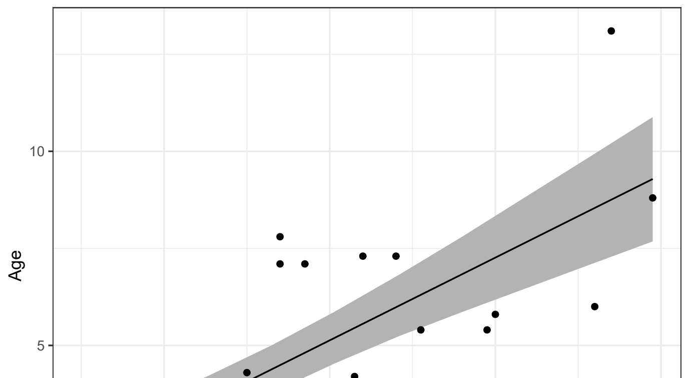


FIGURE 13.4 95% credible interval for the mean age as a function of `proportion.black`.

Think-Pair-Share: How could we modify the code in this section to produce a 95% prediction interval?

13.6 Prediction intervals

Recall that prediction intervals need to consider not only the uncertainty associated with the regression line but also the variability of the observations about the line. This variability is modeled using a Normal distribution with standard deviation σ. This presents a simple way forward - we just need to simulate added variability about the line using `rnorm` and can use our posterior distribution for σ when generating these values.

```
# Pull off the samples from the posterior distribution of sigma
sigmap <- MCMCpstr(lmbayes, params = "sigma", type = "chains")$sigma

# Set up a matrix to hold 95% prediction intervals for each value of prop.black
pred.int <- matrix(NA, nlions, 2)

# Loop over values for proportion.black
for(i in 1:nlions){
  # Estimate the mean age for the current value of prop.black and for
  #    each MCMC sample of beta0 and beta1
  age.inds <- betas$beta0 + rep(prop.black[i], nmcmc)*betas$beta1 +
                   rnorm(nmcmc, mean = 0, sd = sigmap)
  # Now for prediction intervals using the quantiles of the age values
  pred.int[i,] <- quantile(age.inds, prob = c(0.025, 0.975))
}
```

Now, we can add the prediction intervals to our plot (Figure 13.5).

```
beta.hat <- MCMCpstr(lmbayes, params = c("beta0", "beta1"), func = mean)
phats <- data.frame(phats,
                    LPL = pred.int[,1],
                    UPL = pred.int[,2])
ggplot(phats, aes(proportion.black, est)) +
  geom_line(aes(proportion.black, LCL), col = "red", lty = 2) +
  geom_line(aes(proportion.black, UCL), col = "red", lty = 2) +
  geom_line(aes(proportion.black, LPL), col = "blue", lty = 2) +
  geom_line(aes(proportion.black, UPL), col = "blue", lty = 2) +
  geom_line() +ylab("Age") +
  geom_point(data = LionNoses, aes(proportion.black, age))
```

13.7 Credible and prediction intervals the easy way

The above examples are nice in that they demonstrate how you can easily calculate derived quantities using samples from the posterior distributions. However, there is an easier way to get both credible intervals and prediction intervals, and that is, to have JAGS do all the calculations for us and save the necessary output.

If we look at our JAGS code, below, we actually had everything we needed to calculate credible intervals and prediction intervals.

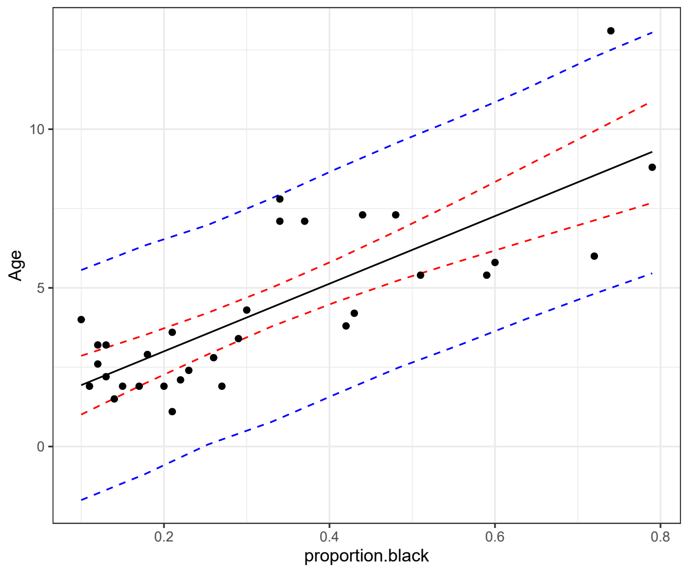


FIGURE 13.5 95% credible (red) and prediction (blue) intervals for age as a function of `proportion.black`.

Think-Pair-Share: Can you think of an easier way to calculate credible intervals and prediction intervals from quantities that JAGS is already computing?

```
linreg<-function(){

# Priors
 beta0 ~ dnorm(0,0.001)
 beta1 ~ dnorm(0,0.001)
 sigma ~ dunif(0, 100)

# Derived quantities
 tau <- 1/ (sigma * sigma)

# Likelihood
 for (i in 1:n) {
    age[i] ~ dnorm(mu[i], tau)
    mu[i] <- beta0 + beta1*proportion.black[i]
 }

# Assess model fit using a sums-of-squares-type discrepancy
 for (i in 1:n) {
    residual[i] <- age[i]-mu[i]       # Residuals for observed data
    sq[i] <- pow(residual[i], 2)    # Squared residuals for observed data

# Generate replicate data and compute fit stats for them
    y.new[i] ~ dnorm(mu[i], tau) # one new data set at each MCMC iteration
    sq.new[i] <- pow(y.new[i]-mu[i], 2)  # Squared residuals for new data
 }
 fit <- sum(sq[])            # Sum of squared residuals for actual data set
 fit.new <- sum(sq.new[])       # Sum of squared residuals for new data set
}
```

Note that:

- JAGS has already sampled from the posterior distribution of the mean age for the values of `proportion.black` in our data set – i.e., `mu[i]`. These can be used to form credible intervals.
- JAGS has already generated new observations from our assumed model – i.e., `y.new[i]`. These can be used to form prediction intervals.

Below, we demonstrate that these outputs could be used to form credible and prediction intervals.

```
CI <- MCMCpstr(lmbayes, params = c("mu"),
               func = function(x) quantile(x, probs = c(0.025, 0.975)))$mu
PI <- MCMCpstr(lmbayes, params = c("y.new"),
               func = function(x) quantile(x, probs = c(0.025, 0.975)))$y.new
```

Plotting these results (Figure 13.6), we see that we get nearly identical intervals as in Figure 13.5:

```
phats2 <- data.frame(est =MCMCpstr(lmbayes, params = c("mu"), func = mean)$mu,
                     proportion.black = LionNoses$proportion.black,
                    LCL = CI[,1],
                    UCL = CI[,2],
                    LPL = PI[,1],
                    UPL = PI[,2])
ggplot(phats2, aes(proportion.black, est)) +
  geom_line(aes(proportion.black, LCL), col = "red", lty = 2) +
  geom_line(aes(proportion.black, UCL), col = "red", lty = 2) +
  geom_line(aes(proportion.black, LPL), col = "blue", lty = 2) +
  geom_line(aes(proportion.black, UPL), col = "blue", lty = 2) +
  geom_line() +ylab("Age") +
  geom_point(data = LionNoses, aes(proportion.black, age))
```

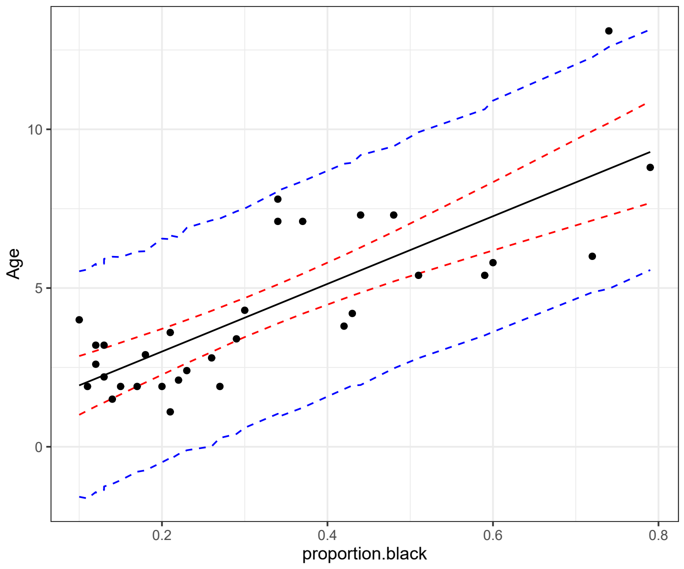


FIGURE 13.6 95% credible (red) and prediction (blue) intervals for age as a function of `proportion.black`.

Part IV

Models for Non-Normal Data

14

Introduction to generalized linear models (GLMs)

Learning objectives

1. Understand the role of random variables and common statistical distributions in formulating modern statistical regression models.
2. Be able to describe a variety of statistical models and their assumptions using equations and text.

14.1 R packages

We begin by loading a few packages upfront:

```r
library(knitr) # for setting options when knitting
library(ggplot2) # for plots
library(patchwork) # for multi-panel plots
library(gridExtra)  # another option for multi-panel plots
library(dplyr) # for data wrangling
```

14.2 The Normal distribution as a data-generating model

In introductory statistics courses, it is customary to see linear models written in terms of an equation that decomposes a response variable into a signal + error as below:

$$y_i = \underbrace{\beta_0 + x_i\beta_1}_{\text{Signal}} + \underbrace{\epsilon_i}_{\text{error}}, \text{ with} \qquad (14.1)$$
$$\epsilon_i \sim N(0, \sigma^2)$$

This decomposition is made possible because the Normal distribution has separate parameters that describe the mean (μ) and variance (σ^2). Yet, as we saw in Chapter 9, the mean and variance for many commonly used statistical distributions are linked together by the same set of parameters. For example, the mean and variance of a binomial random variable are given by np and $np(1-p)$, respectively. Similarly, the mean and variance of a Poisson random variable are both λ. Thus, moving forward, we will want to use a little different notation to describe the models we fit.

Instead of writing the linear model in terms of signal + error, we can write the model down by specifying the probability distribution of the response variable and indicating whether any of the parameters in the model are functions of predictor variables. For example, the model in equation (14.1) can be written as:

$$Y_i|X_i \sim N(\mu_i, \sigma^2)$$
$$\mu_i = \beta_0 + \beta_1 X_{1,i} + \dots \beta_p X_{p,i}$$

This description highlights that: 1) the distribution of Y_i depends on a set of predictor variables X_i, 2) the distribution of the response variable, conditional on predictor variables is Normal, 3) the mean of the Normal distribution depends on predictor variables (X_1 through X_p) and a set of regression coefficients (β_1 through β_p), and 4) the variance of the Normal distribution is constant and given by σ.

As usual, I think it is helpful to also visualize the model (depicted in Figure 14.1) for a model containing a single predictor, X. At any given value of X, the mean of Y falls on the line $E[Y|X] = \beta_0 + \beta_1 X$. The variability about the mean (i.e., the line) is described by a Normal distribution with constant variance σ.

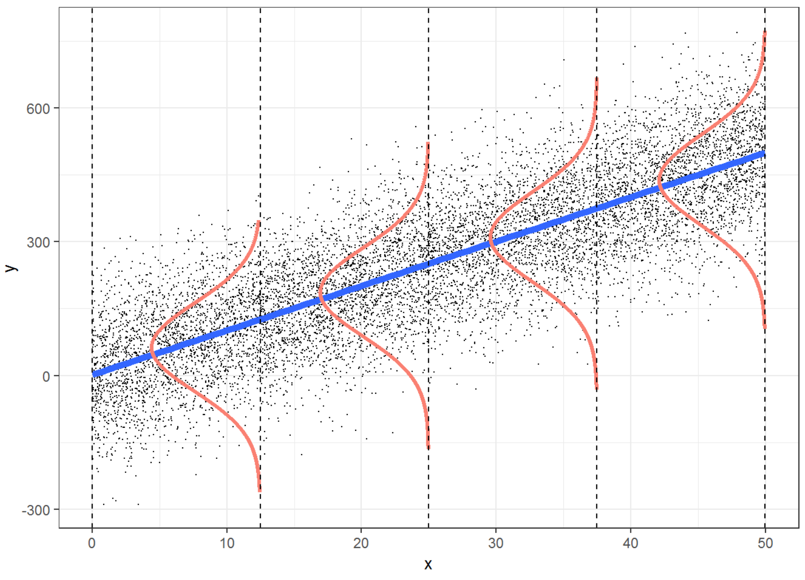


FIGURE 14.1 Figure illustrating the simple linear regression model. From https://bookdown.org/roback/bookdown-bysh/

14.3 Generalized linear models

14.3.1 Nomenclature

One thing that is sometimes confusing is the difference between General Linear Models and Generalized Linear Models. The term General Linear Model is used to acknowledge that several statistical methods, listed below, can all be viewed as fitting a linear model with the same underlying assumptions (linearity, constant variance, normal residuals, and independence):

- A t-test is a linear model with a single categorical predictor with 2 levels.

- ANOVA is a linear model with a single categorical predictor with ≥ 2 levels.
- ANCOVA is a linear model containing a continuous predictor and a categorical predictor, but no interaction between the two (thus, the different groups have a common slope) .
- More generally, models may include a mix of continuous and categorical variables and possible interactions between variables.

Generalized** linear models, by contrast, refers to a broader class of models that allow for alternative probability distributions in addition to the Normal distribution. Historically, the development of generalized linear models focused on statistical distributions in the exponential family (McCullagh and Nelder 1989), which includes the normal distribution with constant variance, Poisson, Binomial, Gamma, among others. These distributions can all be written in a specific form (eq. (14.2)), which facilitated the development of common methods for model fitting and a unified theory for interpreting model parameters (McCulloch and Neuhaus 2005):

$$f(y|\theta,\tau) = h(y,\tau)\exp\left(\frac{b(\theta)T(y) - A(\theta)}{d(\tau)}\right) \tag{14.2}$$

If you were to take a class on generalized linear models in a statistics department, you would likely be required to show that various statistical distributions (e.g., those listed above) can be written in this form. You might also learn how derive the mean and variance of each distribution from this general expression (see e.g., Table 2.1 from McCullagh and Nelder 1989; Section 8.7 of Zuur et al. 2009). The fact that it is possible to derive general results for distributions in the exponential family is really quite cool! At the same time, you do not need to work through those exercises to be able to use generalized linear models to analyze your data. Furthermore, it is relatively easy to extend the basic conceptual approach (constructing models where we express parameters of common statistical distributions as a function of predictor variables) to allow for a much richer class of models - i.e., there is no need to limit oneself to distributions from the exponential family.

14.4 Describing the data-generating mechanism

With distributions other than the Normal distribution, we will no longer be able to write our models in terms of a signal + error. Yet, it may still be conceptually useful to consider how these models capture systematic and random variation in the data. Systematic variation is captured by modeling some transformation of the mean, $g(\mu_i)$, as a linear function of predictor variables. In addition, we will assume that the response $Y_i|X_i$ comes from a particular statistical distribution.

Systematic component:

- $\eta_i = g(\mu_i) = \beta_0 + \beta_1 X_{1,i} + \ldots \beta_p X_{p,i}$; $g()$ is called the link function; η_i is called the linear predictor.
- $\Rightarrow E[Y_i|X_i] = \mu_i = g^{-1}(\beta_0 + \beta_1 X_{1,i} + \ldots \beta_p X_{p,i})$; $g^{-1}()$ is the inverse link function.

Random component:

- We write $Y_i|X_i \sim f(y_i|x_i), i = 1, \ldots, n$
- $f(y_i|x_i)$ is a probability density or probability mass function of a statistical distribution and describes unmodeled variation about $\mu_i = E[Y_i|X_i]$

Examples of generalized linear models are given below.

Linear Regression:

- $f(y_i|x_i) = N(\mu_i, \sigma^2)$
- $E[Y_i|X_i] = \beta_0 + \beta_1 X_1 + \dots \beta_p X_p$
- $g(E[Y_i|X_i]) = \mu_i$, the identity link
- $E[Y_i|X_i] = g^{-1}(\eta_i) = \eta_i = \beta_0 + \beta_1 X_{1,i} + \dots \beta_p X_{p,i}$

Poisson regression:

- $f(y_i|x_i) \sim Poisson(\lambda_i)$
- $E[Y_i|X_i] = \lambda_i$
- $g(E[Y_i|X_i]) = \eta_i = log(\lambda_i) = \beta_0 + \beta_1 X_{1,i} + \dots \beta_p X_{p,i}$
- $\mu_i = g^{-1}(\eta_i) = exp(\eta_i) = exp(\beta_0 + \beta_1 X_{1,i} + \dots \beta_p X_{p,i})$

Logistic regression:

- $f(y_i|x_i) \sim \text{Bernoulli}(p_i)$
- $E[Y_i|X_i] = p_i$
- $g(E[Y_i|X_i]) = \eta_i = \text{logit}(p_i) = \log(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 X_{1,i} + \dots \beta_p X_{p,i}$
- $\mu_i = g^{-1}(\eta_i) = \frac{\exp^{\eta_i}}{1+\exp^{\eta_i}} = \frac{\exp^{\beta_0+\beta_1 X_{1,i}+\dots\beta_p X_{p,i}}}{1+\exp^{\beta_0+\beta_1 X_{1,i}+\dots\beta_p X_{p,i}}}$

14.4.1 Link functions, $g()$

Link functions allow the systematic component $(\beta_0 + \beta_1 X_1 + \dots \beta_p X_p)$ to live on $(-\infty, \infty)$ while keeping the μ_i consistent with the range of the response variable. For example, $g^-1(\eta) = exp(\eta)$ ensures that the mean in Poisson regression models, λ_i, is greater than or equal to 0. Similarly, $g^{-1}(\eta) = \frac{exp(\eta)}{1+exp(\eta)}$ ensures the mean of logistic regression models, p_i, is between 0 and 1 (Figure 14.2).

```
eta<-seq(-10,10, length=1000)
mean.Poisson<-exp(eta)
mean.Logistic<-exp(eta)/(1+exp(eta))
pdat<-data.frame(eta, mean.Poisson, mean.Logistic)
p1<-ggplot(pdat,aes(eta, mean.Poisson))+geom_line()+
  xlab(expression(eta))+ylab(expression(exp(eta)))+ggtitle("Poisson Regression") + theme_bw()
p2<-ggplot(pdat,aes(eta, mean.Logistic))+geom_line()+
  xlab(expression(eta))+ylab(expression(exp(eta)/(1+exp(eta))))+
  ggtitle("Logistic Regression") + theme_bw()
gridExtra::grid.arrange(p1, p2, nrow=1)
```

Poisson Regression:

- $g(\mu_i) = \eta_i = log(\lambda_i) = \beta_0 + \beta_1 x_1 + \dots \beta_p x_p$, range $= (-\infty, \infty)$
- $\Rightarrow \mu_i = exp(\beta_0 + \beta_1 x_1 + \dots \beta_p x_p)$ has range $= [0, \infty]$

Logistic regression:

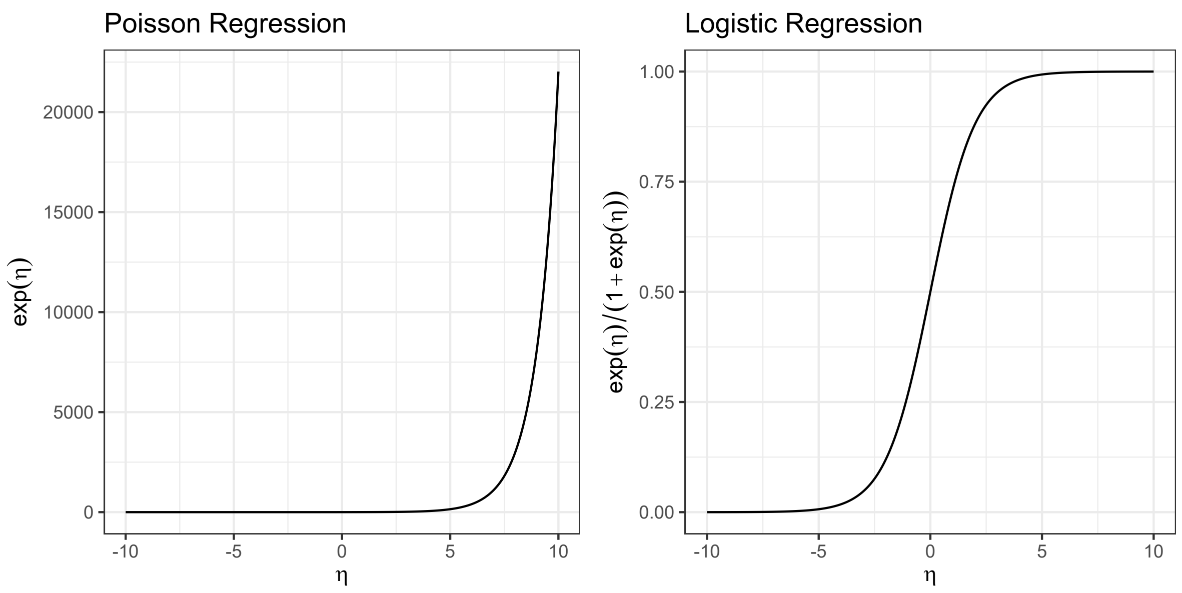


FIGURE 14.2 Mean response versus linear predictor for Poisson and Logistic regression models.

- $g(\mu_i) = \eta_i = log(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 x_1 + \dots \beta_p x_p$, range $= (-\infty, \infty)$
- $\mu_i = g^{-1}(\eta_i) = \frac{\exp^{\eta_i}}{1+\exp^{\eta_i}} = \frac{\exp^{\beta_0+\beta_1 x_1 + \dots \beta_p x_p}}{1+\exp^{\beta_0+\beta_1 x_1+\dots\beta_p x_p}}$, range $= (0, 1)$

Now that we have defined terms and seen how these definitions apply to specific distributions, let's consider a hypothetical example using the Poisson regression.

14.5 Example: Poisson regression for pheasant counts

Consider a wildlife researcher that is interested in modeling how pheasant abundance varies with the amount of grassland cover. She divides the landscape up into a series of plots. She and several colleagues walk transects in these plots with their hunting dogs to locate and flush pheasants, leading to a count in each plot, Y_i. They also use a Geographic Information System to capture the percent grassland cover in each plot, X_i.

We could fit a Poisson regression model (we will see how to do this in R in Section 15) in which the mean count depends on grassland abundance. To describe the model using equations, we would write:

- $Y_i|X_i \sim Poisson(\lambda_i)$
- $log(\lambda_i) = \beta_0 + \beta_1 X_i$

Because the mean of the Poisson distribution is λ, we have assumed:

- $E[Y_i|X_i] = \lambda_i = g^{-1}(\beta_0 + \beta_1 X_i) = \exp(\beta_0 + \beta_1 X_{1,i})$

In English, we model the log of the mean as a linear function of grassland abundance (this is the systematic component of the model). Further, we assume the counts, $Y_1, Y_2, \dots, Y_n$, are independent Poisson random variables, with means that depend on the amount of grassland cover (X_i) (this is the random component).

FIGURE 14.3 Picture of a pheasant by Gary Noon - Flickr, CC BY-SA 2.0.

14.5.1 Poisson regression assumptions

These next two subsections were slightly modified from the online version of Roback and Legler (2021), which is licensed under Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Much like with linear regression, we can list the underlying assumptions of Poisson regression:

1. **Poisson Response**: The response variable is a count per unit of time or space, described by a Poisson distribution.
2. **Independence**: The observations must be independent of one another.
3. **Mean=Variance**: By definition, the mean of a Poisson random variable must be equal to its variance.
4. **Linearity**: The log of the mean rate, $\log(\lambda)$, must be a linear function of x.

Figure 14.4 illustrates a comparison of linear regression and Poisson regression models.

1. For the linear regression model (left panel of Fig. 14.4), the responses appear to be approximately normal for any given value of X. For the Poisson regression model (right panel of Fig. 14.4), the responses follow a Poisson distribution for any given value of X. For Poisson regression, small values of λ are associated with a distribution that is noticeably skewed with lots of small values and only a few larger ones. As λ increases, the distribution of the responses begins to look more and more like a normal distribution.
2. In the linear regression model, the variation in Y at each level of X, σ^2, is the same. For Poisson regression, the variance increases with the mean (remember, the variance is equal to the mean for the Poisson distribution).
3. In the case of linear regression, the mean responses for each level of X falls on a line, $E[Y_i|X_i] = \mu_i = \beta_0 + \beta_1 X_i$. In the case of the Poisson model, the mean values of Y at each level of X fall on the curve $\lambda_i = \exp(\beta_0 + \beta_1 X_i)$.

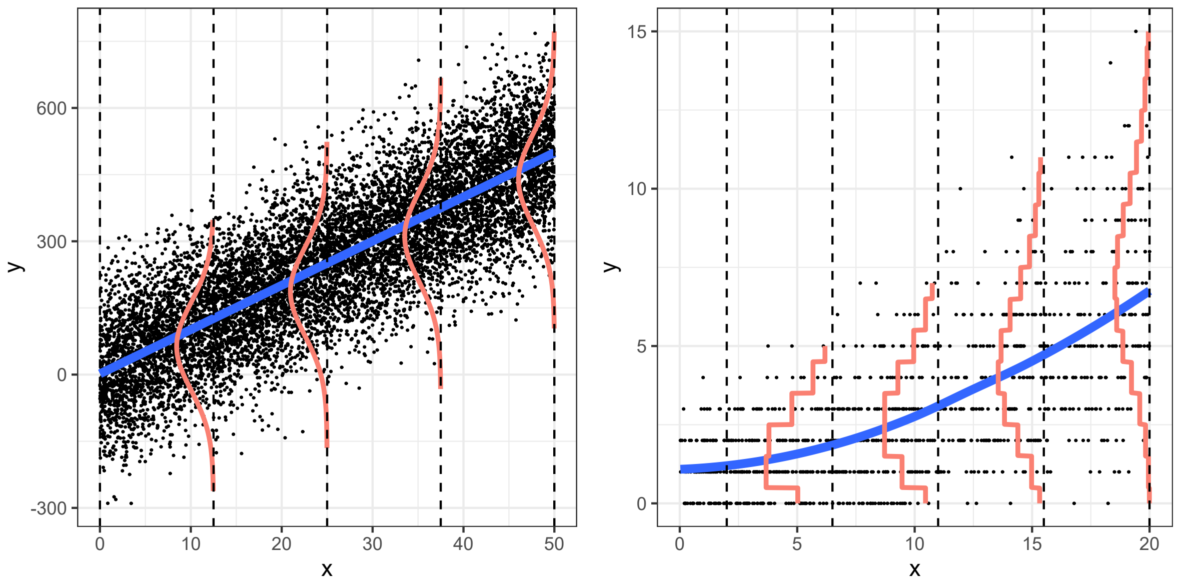


FIGURE 14.4 Regession Models: Linear Regression (left) and Poisson Regression (right)

15

Regression models for count data

Learning objectives

1. Understand how maximum likelihood is used to fit modern statistical regression models.
2. Be able to fit regression models appropriate for count data in R and JAGS
    - Poisson regression models
    - Quasi-Poisson (R only)
    - Negative Binomial regression.
3. Interpret estimated coefficients and describe their uncertainty using confidence and credible intervals.
4. Use simple tools to assess model fit
    - Residuals (deviance and Pearson)
    - Goodness-of-fit tests.
5. Use deviances and AIC to compare models.
6. Use an offset to model rates and densities, accounting for variable survey effort.
7. Be able to describe statistical models and their assumptions using equations and text and match parameters in these equations to estimates in R output.

15.1 R packages

We begin by loading a few packages upfront:

```r
library(kableExtra) # for tables
options(kableExtra.html.bsTable = T)
library(sjPlot) # for summarizing output
library(modelsummary) # for creating tables with regression output
library(performance) # for checking model assumptions
library(ggeffects) # for partial residual plots
library(ggthemes) # for colorblind pallete
library(car) # for residual plots
library(dplyr) # for data wrangling
library(MASS) # for glm.nb
```

In addition, we will use data and functions from the following packages:

- `Data4Ecologists` for the `slugs` and `longnosedace` data sets.
- `DHARMa` (Hartig 2021) for creating residual plots and testing goodness of fit
- `R2Jags` (Y.-S. Su and Masanao Yajima 2021) for running JAGS code
- `MCMCvis` (Youngflesh 2018) for summarizing MCMC output

15.2 Introductory examples: Slugs and Fish

15.2.1 Maximum likelihood: revisiting the slug data

Let's briefly remind ourselves how we can estimate parameters in a generalized linear model using maximum likelihood. Consider again the slug data from Section 10.3.

FIGURE 15.1 An orange slug to spice things up (this is not likely the type of slug that was counted). Guillaume Brocker/Gbrocker, CC BY-SA 3.0.

We saw how, using `optim`, we could fit a model where:

$$Y_i|X_i \sim Poisson(\lambda_i)$$

$$log(\lambda_i) = \beta_0 + \beta_1 X_i$$

$$\Rightarrow E[Y_i|X_i] = \mu_i = g^{-1}(\eta) = \exp(\beta_0 + \beta_1 X_i)$$

To do this, we relied on the `dpois` function in R to specify the likelihood. Remember, the general likelihood for the Poisson distribution is given by:

$$L(\beta; y_1, \ldots, y_n, x_1, \ldots, x_n) = \prod_{i=1}^{n} \frac{\lambda_i^{y_i} \exp(-\lambda_i)}{y_i!}$$

Allowing the λ_i to depend on predictors and regression parameters,$\lambda_i = exp(\beta_0 + \beta_1 x_i)$, we can write:

$$L(\beta; y_1, \ldots, y_n, x_1, \ldots, x_n) = \prod_{i=1}^{n} \frac{\exp(\beta_0 + \beta_1 x_i)^{y_i} \exp(-\exp(\beta_0 + \beta_1 x_i))}{y_i!}$$

And, we can write the log-likelihood as:

$$\log L(\beta; y_1, \dots, y_n, x_1, \dots, x_n) = \sum_{i=1}^{n} \Big(y_i(\beta_0 + \beta_1 x_i) - \exp(\beta_0 + \beta_1 x_i) + log(y_i!) \Big)$$

We could take derivatives with respect to β_0 and β_1 and set them equal to 0. However, we would not be able to derive analytical solutions for $\hat{\beta_0}$ and $\hat{\beta_1}$. Instead, we (and `glm`) will use numerical optimization techniques to find the values of β_0 and β_1 that maximize the log likelihood.

Lets look at how we can use `glm` to fit this model:

```
fit.pois <- glm(slugs ~ field, data = slugs, family = poisson())
summary(fit.pois)
```

```

Call:
glm(formula = slugs ~ field, family = poisson(), data = slugs)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    0.2429     0.1400   1.735 0.082744 .
fieldRookery   0.5790     0.1749   3.310 0.000932 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 224.86  on 79  degrees of freedom
Residual deviance: 213.44  on 78  degrees of freedom
AIC: 346.26

Number of Fisher Scoring iterations: 6
```

Note, by default, R will use a log link when modeling data using the Poisson distribution. We could, however, use other links by specifying the link argument to the `family` function as below (for the identity link):

```
fit.pois <- glm(slugs ~ field, data = slugs, family = poisson(link = "identity"))
summary(fit.pois)
```

```

Call:
glm(formula = slugs ~ field, family = poisson(link = "identity"),
    data = slugs)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.2750     0.1785   7.141 9.24e-13 ***
fieldRookery   1.0000     0.2979   3.357 0.000789 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 224.86  on 79  degrees of freedom
Residual deviance: 213.44  on 78  degrees of freedom
AIC: 346.26

Number of Fisher Scoring iterations: 3
```

Also, note that by default, R uses effects coding (recall Section 3.7.1). We could, instead, use means coding as follows:

```
fit.pois <- glm(slugs ~ field - 1, data = slugs, family = poisson())
summary(fit.pois)
```

```

Call:
glm(formula = slugs ~ field - 1, family = poisson(), data = slugs)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
fieldNursery   0.2429     0.1400   1.735   0.0827 .
fieldRookery   0.8220     0.1048   7.841 4.46e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 263.82  on 80  degrees of freedom
Residual deviance: 213.44  on 78  degrees of freedom
AIC: 346.26

Number of Fisher Scoring iterations: 6
```

Think-Pair-Share: See if you can demonstrate the relationships between the different parameterizations of the Poisson model (means and effects coding and using log and identity links).

15.2.2 Modeling the distribution of longnose dace

FIGURE 15.2 Picture of a longnose dace. Smithsonian Environmental Research Center, CC BY 2.0.

Let's consider another count data set collected by the Maryland Biological Stream Survey. The data set includes counts of longnose dace in different sampled areas along with the following predictors:

- **stream**: name of each sampled stream (i.e., the name associated with each case).
- **longnosedace**: number of longnose dace (*Rhinichthys cataractae*) per 75-meter section of stream.
- **acreage**: area (in acres) drained by the stream.
- **do2**: the dissolved oxygen (in mg/liter).
- **maxdepth**: the maximum depth (in cm) of the 75-meter segment of stream.
- **no3**: nitrate concentration (mg/liter).
- **so4**: sulfate concentration (mg/liter).
- **temp**: the water temperature on the sampling date (in degrees C).

We can load the data using:

```
library(Data4Ecologists)
data("longnosedace")
```

We begin by fitting a linear model to the data and explore whether the underlying assumptions are reasonable using residual plots constructed using the `check_model` function in the `performance` package (Lüdecke et al. 2021) (Figure 15.3).

```
lmdace <- lm(longnosedace ~ acreage + do2 + maxdepth + no3 + so4 + temp,
             data = longnosedace)
check_model(lmdace, check = c("linearity", "homogeneity", "qq", "normality"))
```

Examining the plots in the first row, we see that the variance of the residuals increases with fitted values. A reasonable next step might be to consider a Poisson regression model since:

1. Our data are counts and can thus only take on integer values.
2. The variance of the Poisson distribution is equal to the mean (and thus, the variance will increase linearly with the fitted values).

We can fit the Poisson model[1], below, using the `glm` function in R:

$$Dace_i \sim Poisson(\lambda_i) \tag{15.1}$$
$$\log(\lambda_i) = \beta_0 + \beta_1 Acreage_i + \beta_2 DO2_i + \beta_3 maxdepth_i + \beta_4 NO3_i + \beta_5 SO4_i + \beta_6 temp_i$$

```
glmPdace <- glm(longnosedace ~ acreage + do2 + maxdepth + no3 + so4 + temp,
              data = longnosedace, family = poisson())
summary(glmPdace)
```

```
##
## Call:
## glm(formula = longnosedace ~ acreage + do2 + maxdepth + no3 +
```

[1]Note that I have chosen to simplify the equation for the model slightly by writing $Y_i \sim Poisson(\lambda_i)$ rather than $Y_i|X_i \sim Poisson(\lambda_i)$. I like the former notation because it makes it clear that the distribution of the response variable depends on the predictor variables included in the model. With so many predictors, however, the equation would get a bit unwieldy if we listed them all. Also, note that we are assuming that the predictor variables are known quantities, so it is less important that we write down an equation for the distribution of Y_i as being conditional on X_i. When consider models that include latent variables (Chapter 17) or random effects (Chapter 18), it will be important to explicitly acknowledge when our distribution of Y_i is conditional on specific values of these random quantities. I.e., we need to be careful to indicate if the distribution for Y_i is a conditional distribution or marginal distribution. The latter will require averaging over other random variables (i.e., the latent variables or random effects).

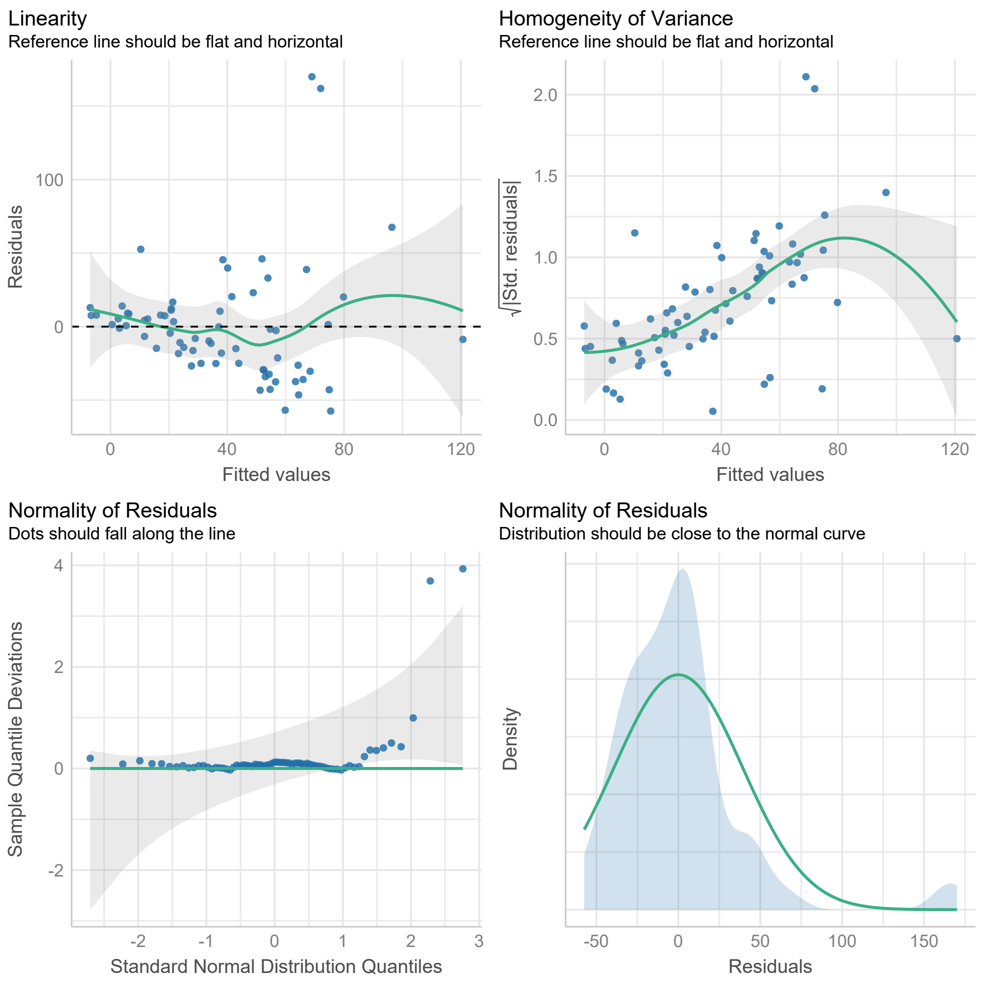


FIGURE 15.3 Residual diagnostic plots for a linear model fit to the longnosedace data.

```
##     so4 + temp, family = poisson(), data = longnosedace)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.564e+00  2.818e-01  -5.551 2.83e-08 ***
## acreage      3.843e-05  2.079e-06  18.480  < 2e-16 ***
## do2          2.259e-01  2.126e-02  10.626  < 2e-16 ***
## maxdepth     1.155e-02  6.688e-04  17.270  < 2e-16 ***
## no3          1.813e-01  1.068e-02  16.974  < 2e-16 ***
## so4         -6.810e-03  3.622e-03  -1.880   0.0601 .
## temp         7.854e-02  6.530e-03  12.028  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 2766.9  on 67  degrees of freedom
```

TABLE 15.1 Parameter estimates (SE) from fitted linear and Poisson regression models.

	Linear regression	Poisson regression
(Intercept)	−127.587 (66.417)	−1.564 (0.282)
acreage	0.002 (0.001)	0.000 (0.000)
do2	6.104 (5.384)	0.226 (0.021)
maxdepth	0.354 (0.178)	0.012 (0.001)
no3	7.713 (2.905)	0.181 (0.011)
so4	−0.009 (0.774)	−0.007 (0.004)
temp	2.748 (1.694)	0.079 (0.007)

```
## Residual deviance: 1590.0  on 61  degrees of freedom
## AIC: 1936.9
##
## Number of Fisher Scoring iterations: 5
```

At this point, we might ask:

- How do we interpret the parameters in the model?
- How can we evaluate the assumptions and fit of the model?
- What tools might we use to select an appropriate set of predictor variables?

15.3 Parameter interpretation

Let's compare the interpretation of regression coefficients in linear and Poison regression models fit to the `dace` data (Table 15.1), focusing on the effect of temperature (`temp`). For the linear regression model, the regression coefficient for `temp` (equal to 2.75) indicates that the mean number of fish changes by 2.75 as the temperature changes by 1 °C while holding everything else constant. To understand how to interpret the coefficient in the Poisson model, remember that the model is parameterized on the log scale. Thus, a first interpretation that we can offer is that the *log mean* number of fish changes 0.079 as the temperature changes by 1°C while holding everything else constant. Ideally, however, we would like to interpret the temperature effect on the scale of our observations – i.e., we want to know how temperature relates to the average number of fish in a stream segment, not the log mean.

It turns out that the Poisson model captures *multiplicative* effects, which are estimated by exponentiating the regression parameters. To see this, let's write down our model in terms of the mean, $E[Y_i|X_i] = \lambda_i$:

$$\lambda_i = \exp(\beta_0 + \beta_1 Acreage_i + \beta_2 DO2_i + \beta_3 maxdepth_i + \beta_4 NO3_i + \beta_5 SO4_i + \beta_6 temp_i)$$

Remembering how exponents work, we can alternatively write:

$$\lambda_i = e^{(\beta_0)} e^{(\beta_1 Acreage_i)} e^{(\beta_2 DO2_i)} e^{(\beta_3 maxdepth_i)} e^{(\beta_4 NO3_i)} e^{(\beta_5 SO4_i)} e^{(\beta_6 temp_i)}$$

Now, consider the ratio of means for stream segments that differ only in `temp` (and by 1 degree C):

$$\frac{\lambda_1}{\lambda_2} = \frac{e^{(\beta_0)} e^{(\beta_1 Acreage_1)} e^{(\beta_2 DO2_1)} e^{(\beta_3 maxdepth_1)} e^{(\beta_4 NO3_1)} e^{(\beta_5 SO4_1)} e^{(\beta_6 (temp_1+1))}}{e^{(\beta_0)} e^{(\beta_1 Acreage_1)} e^{(\beta_2 DO2_1)} e^{(\beta_3 maxdepth_1)} e^{(\beta_4 NO3_1)} e^{(\beta_5 SO4_1)} e^{(\beta_6 temp_1)}} = e^{\beta_6}$$

TABLE 15.2 Rate Ratios = $\exp(\beta)$ (95 percent confidence intervals).

	Poisson regression
(Intercept)	0.209 (0.120, 0.363)
acreage	1.000 (1.000, 1.000)
do2	1.253 (1.202, 1.307)
maxdepth	1.012 (1.010, 1.013)
no3	1.199 (1.174, 1.224)
so4	0.993 (0.986, 1.000)
temp	1.082 (1.068, 1.096)

Thus, if we increase `temp` by 1 unit, while keeping everything else constant, we increase λ by a factor of e^{β_6}.

It is common to report *rate ratios* formed by exponentiating regression parameters as a measure of effect size (Table 15.2)[2]. What about confidence intervals for these effect sizes? Again, we can rely on the theory for maximum likelihood estimators from Section 10.6. Specifically, we could use the result that for large samples $\hat{\beta} \sim N(\beta, I^{-1}(\beta))$ along with delta method (Section 10.8.1) to calculate a confidence interval for e^{β} (for really large samples, it can also be shown that $e^{\hat{\beta}}$ will also be Normally distributed). It is customary, however, to calculate confidence intervals for β (using the Normal distribution) and then calculate confidence intervals for e^{β} by exponentiating these confidence limits. This is because the sampling distribution for $\hat{\beta}$ will typically converge to a Normal distribution faster than the sampling distribution for $\exp(\beta)$ converges to a Normal distribution. Thus, we would calculate the confidence interval for $\exp(\beta_6)$ associated with temperature using: $\exp(\hat{\beta}_6 \pm 1.96\widehat{SE}(\hat{\beta}_6))$.

To calculate these confidence intervals, we need to pull off the variance-covariance matrix for our estimated coefficients using `vcov(glmPdace)`. Taking the square root of the diagonal of this matrix will give us our standard errors.

```
# store coefficients and their standard errors
beta.hats <- coef(glmPdace)
ses <- sqrt(diag(vcov(glmPdace)))
# Calculate CIs
round(cbind(exp(beta.hats - 1.96*ses), exp(beta.hats + 1.96*ses)), 3)
```

```
##              [,1]  [,2]
## (Intercept) 0.120 0.363
## acreage     1.000 1.000
## do2         1.202 1.307
## maxdepth    1.010 1.013
## no3         1.174 1.224
## so4         0.986 1.000
## temp        1.068 1.096
```

[2]There are several R packages that can be used to create tables summarizing regression output. The `tab_mod` function in the `sjPlots` library will, by default, report rate ratios for Poisson regression models. Further, the `modelsummary` function in the `modelsummary` package (Arel-Bundock 2022) has an argument, `exponentiate = TRUE` for calculating rate ratios and their confidence intervals and was used to create Table 15.2.

15.4 Evaluating assumptions and fit

15.4.1 Residuals

As with linear models, there are multiple types of residuals that we might consider:

Standard residuals: $r_i = Y_i - \hat{E}[Y_i|X_i] = Y_i - \hat{Y}_i$. We can access standard residuals using `residuals(model, type="response")`. If the Poisson distribution is appropriate, we would expect the magnitude of these residuals to increase linearly with λ_i and thus, with $\hat{E}[Y_i|X_i] = \hat{Y}_i = \hat{\lambda}_i$.

Pearson residuals: $r_i = \frac{Y_i - \hat{E}[Y_i|X_i]}{\sqrt{\widehat{Var}[Y_i|X_i]}}$. We can access Pearson residuals using `residuals(model, type="pearson")`. If our model is correct, Pearson residuals should have constant variance. You may recall that we considered Pearson residuals when evaluating Generalized Least Squares models in Section 5.

Deviance residuals: $r_i = sign(y_i - \hat{E}[Y_i|X_i])\sqrt{d_i}$, where d_i is the contribution of the i^{th} observation to the residual deviance (as defined in Section 15.4.2) and $sign = 1$ if $y_i > \hat{E}[Y_i|X_i]$ and -1 if $y_i < \hat{E}[Y_i|X_i]$. We can access deviance residuals using `residuals(model, type="deviance")`.

As with linear models, it is common to inspect plots depicting:

- Residuals versus predictors
- Residuals versus fitted values
- Residuals over time or space (to diagnose possible spatial/temporal correlation)

These plots may be useful for identifying whether we should consider allowing for a non-linear relationship between the response and predictor (on the link scale) or if we are missing an important predictor that may be related to temporal or spatial patterning. On the other hand, residual plots for GLMs often exhibit strange patterning due to the discrete nature of the data (e.g., counts can take on only integer values and binary responses have to be either 0 or 1). Thus, it can help to add a smooth curve to the plot to diagnose whether there is a pattern in residuals (e.g., as we increase from small to large fitted values). The `residualPlots` function in the `car` package (Fox and Weisberg 2019) will do this for us (Figure 15.4). It also returns goodness-of-fit tests aimed at detecting whether the relationship between predictor and response is non-linear on the transformed scale (i.e., log in the case of Poisson regression models).

```
residualPlots(glmPdace)
```

```
##            Test stat Pr(>|Test stat|)
## acreage      50.3494        1.287e-12 ***
## do2          40.9062        1.597e-10 ***
## maxdepth     12.8087         0.000345 ***
## no3           0.0670         0.795691
## so4          22.0158        2.704e-06 ***
## temp          1.2443         0.264652
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The goodness-of-fit tests suggest there are problems with this model, and indeed, my experience has been that the Poisson model rarely provides a good fit to ecological data. We will further consider goodness-of-fit testing in Section 15.4.5.

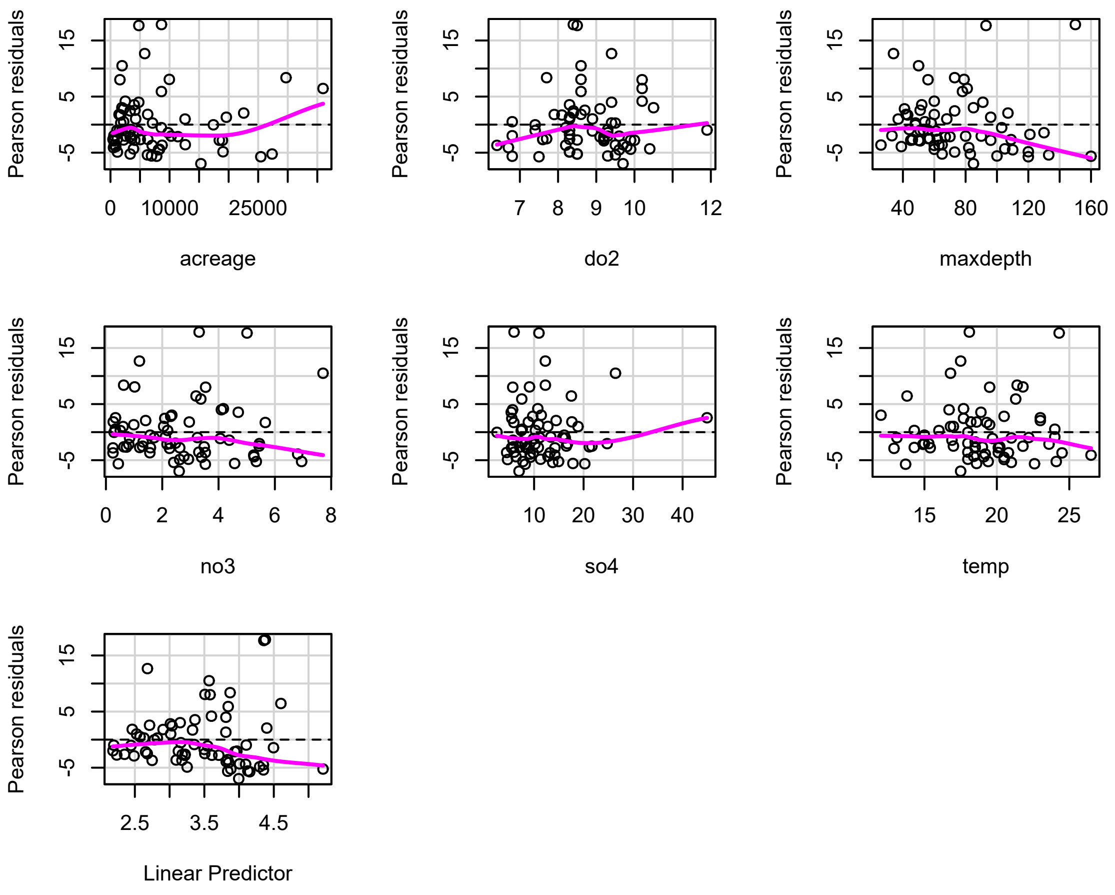


FIGURE 15.4 Plots of residuals versus each predictor and against fitted values using the `residualPlots` function in the `car` package (Fox and Weisberg 2019).

Another popular alternative for evaluating fit of generalized linear models, including models with random effects (Sections 18 and 19), is to use simulation-based approaches available in the `DHARMa` package (Hartig 2021). `DHARMa` creates simulated residuals that are transformed to lie between 0 and 1 (Figure 15.5) via the following steps:

1. New response data are simulated from the fitted model for each observation (similar to the process we used to conduct the goodness-of-fit test in Section 13.4).
2. For each observation, the cumulative distribution function of the simulated data (associated with that observation's predictor values) is used to determine a standardized residual under the assumed model. If the observation is smaller than all of its corresponding simulated data points, then the residual will be 0. If it is larger than 50% of the simulated data points, it will have a residual of 0.5. And if it is larger than all of the simulated data points, it will have a residual of 1.

If the model is appropriate for the data, then we should expect the distribution of the residuals to be uniform (all values should be equally likely). A nice thing about this approach is that it can work with any type of model (linear model, generalized linear model, etc).

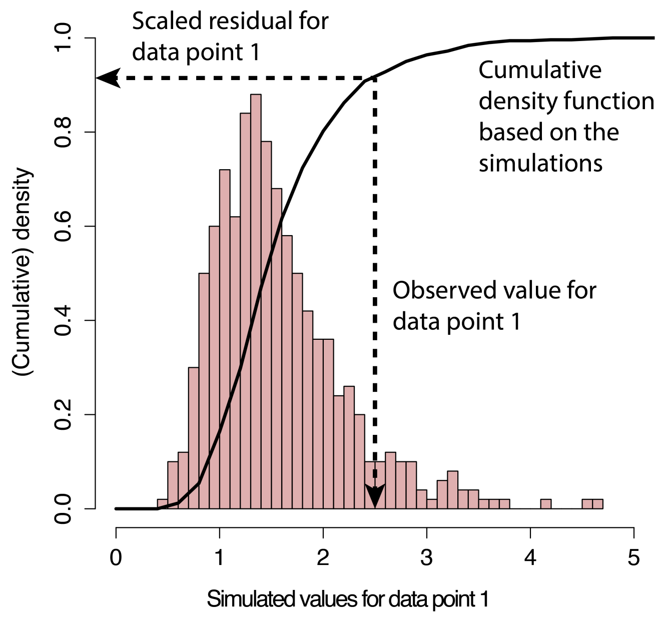


FIGURE 15.5 Simulation-based process for creating standardized residuals via the `DHARMa` package (Hartig 2021). Figure is copied from the DHARMa vignette.

The main workhorse of the `DHARMa` package is the `simulateResiduals` function, which implements the algorithm above.

```
library(DHARMa)
simulateResiduals(glmPdace, plot = TRUE)
```

```
## Object of Class DHARMa with simulated residuals based on 250 simulations with refit = FALSE . See ?DHARM
##
## Scaled residual values: 0.01291999 0 0.9107332 0.004 0.9422699 0 1 1 0 0 0.139003 0.8586797 0 0.2893398
```

By adding `plot = TRUE`, the function also provides us with two plots (Figure 15.6). The first plot on the left is a Quantile-Quantile (QQ) plot for a uniform (0, 1) distribution. If the model is appropriate, the points should fall on the 1-1 line. `DHARMa` also provides test statistics and p-values associated with tests for whether the distribution is uniform, whether the residuals are overdispersed, and whether their are outliers. In this case, all three tests suggest the model provides a poor fit to the data. The second plot (on the right) is a residuals versus fitted values plot. Observations that fall outside of the range of simulated values are highlighted as outliers (red stars). `DHARMa` also performs a quantile regression to compare empirical and theoretical 0.25, 0.5 and 0.75 quantiles and provides a p-value for a test of whether these differ from each other.

15.4.2 Aside: Deviance and pseudo-R^2

Let θ be a set of parameters to be estimated by maximum likelihood (in the context of our above example, θ is the set of regression coefficients, β_0 through β_6). The deviance is a measure of model fit when using maximum likelihood.

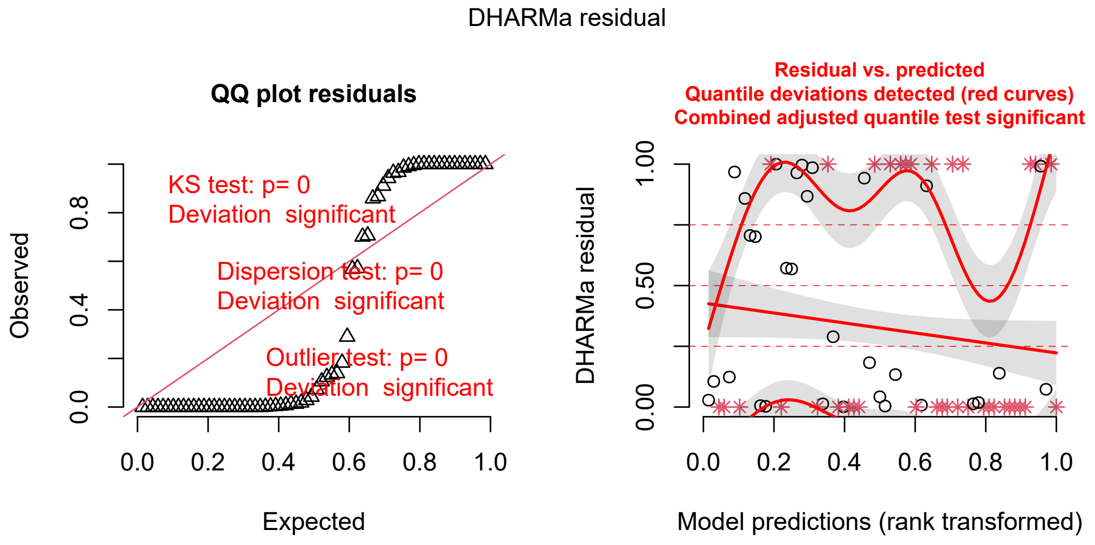


FIGURE 15.6 DHARMa residual plots for the Poisson regression model fit to the longnose Dace data.

The residual deviance, D_0, is given by:

$$D_0 = 2(logL(y|\hat{\theta}_s) - logL(y|\hat{\theta}_0)), \text{where}$$

- $logL(\hat{\theta}_s|y)$ = log-likelihood for a "saturated" model (one with a parameter for each observation).
- $logL(\hat{\theta}_0|y)$ = log-likelihood for our model of interest evaluated at the maximum likelihood estimates ($\hat{\theta}_0$).

The null deviance is the residual deviance for a model that only contains an intercept. It may be helpful to think of the null deviance and residual deviance as maximum likelihood equivalents of total and residual sums of squares, respectively.

There is no R^2 for generalized linear models, but it is not uncommon to see percent deviance explained in regression output and also reported as a (pseudo R^2) measure (Faraway 2016):

$$\text{pseudo } R^2 = 100 \times \frac{\text{null deviance - residual deviance}}{\text{null deviance}} \tag{15.2}$$

This is easy to calculate from the fitted model:

```
1-(glmPdace$deviance/glmPdace$null.deviance)
```

```
## [1] 0.4253322
```

Like R^2 for linear models, the above pseudo-R^2 will range between 0 and 1, with higher values indicating better fit. On the other hand, it should not be interpreted in the same way as the R^2 in a linear model (as the proportion of variance in the response explained by the model), and its values may be considerably lower than users of R^2 in linear models have come to expect (e.g., McFadden 1977 suggests values of 0.2-0.4 describe *excellent* fit in the context of logistic regression). Also, note there have been many other pseudo-R^2 measures proposed, and none are without controversy. For more information regarding other measures of pseudo R^2, how to calculate them, and potential considerations when using them, see this FAQ for pseudo-R^2 in the context of logistic regression. In addition, Nakagawa and Schielzeth (2013) reviews several pseudo-R^2 metrics proposed for generalized linear models before proposing new metrics for use with mixed models that are the subject of Chapters 18 and 19. Lastly, we note that the `performance` package has an `r2()` function for calculating various R^2 metrics for a variety of models, including generalized linear models and mixed-effect models. It will return a list containing values for what the authors deem to be the "most appropriate" R^2 for a given model type. Below, we apply this function to our Poisson regression model, which returns a value of 1 (which I find to be quite surprising).

```
r2(glmPdace)
```

```
## # R2 for Generalized Linear Regression
##   Nagelkerke's R2: 1.000
```

If you try other options, you will get very different results. For example, we get a much lower value when using the `r2_mcfadden()` function.

```
r2_mcfadden(glmPdace)
```

```
## # R2 for Generalized Linear Regression
##        R2: 0.380
##   adj. R2: 0.379
```

Personally, I do not find all that much value in these different R^2 measures. I would much rather inspect effect plots to get a feel for the amount of unexplained variation (e.g., Figure 15.7). These plots will be explored in more detail in Section 16.6.

```
sjPlot::plot_grid(plot(ggpredict(glmPdace), add.data = TRUE, one_plot = FALSE))
```

15.4.3 Overdispersion: What is it?

The Poisson distribution assumes $Var(Y_i|X_i) = E(Y_i|X_i)$. When the variance is greater than the mean, we say the data are overdispersed relative to the Poisson distribution. There are many reasons why data may be overdispersed:

- We may have omitted important predictor variables
- Explanatory and response variables may be measured with error
- Our model may be mis-specified (e.g., the relationship between $\log(\lambda)$ and X may be non-linear)
- Our data set may include one or more large outliers
- Our data may exhibit significant spatial, temporal, within-individual clustering (e.g., as may be the case when we have more than one observation on each sample unit)

Lastly, the response variable may result from a mixture of random processes. In wildlife monitoring data sets, for example, we might consider that:

- One set of environmental variables may determine whether a particular habitat is suitable for a given species, and thus, whether a site is occupied or not.

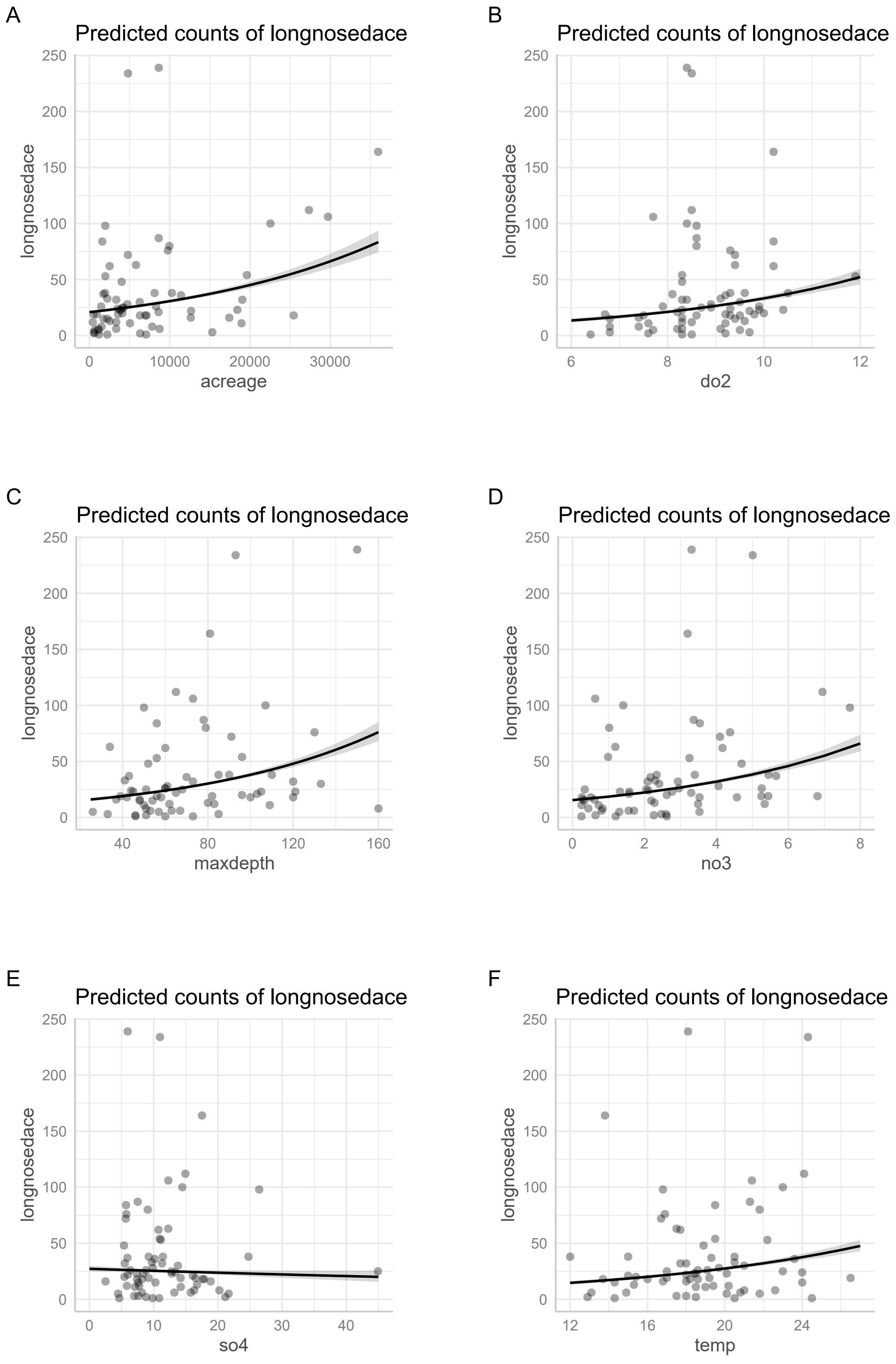


FIGURE 15.7 Partial residual plots for the Poisson regression model fit to the longnose dace data.

- A second, possibly overlapping set of variables, may determine the density in suitable habitat.

This situation can lead to data sets with an overabundance of zeros. Zero-inflated models (Section 17) have been developed to accommodate this situation.

15.4.4 Simple tests for overdispersion

Given that the mean and variance are likely to both depend on predictors, X, how should we diagnose possible overdispersion? It is not uncommon to compare the residual deviance D_0 or Pearson's χ^2 statistic $\sum_{i=1}^{n}\frac{(Y_i-E[Y_i|X_i])^2}{Var[Y_i|X_i]}$ to a χ^2 distribution with $n-p$ degrees of freedom to detect overdispersion. The `check_overdispersion` function in the `performance` package provides a test based on the Pearson χ^2 statistic.

```
check_overdispersion(glmPdace)
```

```
## # Overdispersion test
##
##        dispersion ratio =   29.403
##   Pearson's Chi-Squared = 1793.599
##                 p-value =  < 0.001

## Overdispersion detected.
```

Yet, the χ^2 distributional assumption may not be appropriate especially when the number of observations with a given suite of predictors is small (e.g. ≤ 5) as will typically be the case when the model contains continuous predictors. Thus, it is usually best to test for overdispersion using (predictive) simulation techniques discussed in Section 15.4.5.

15.4.5 Goodness-of-fit test using predictive simulation

In Section 15.4.1, we saw that the `residualPlots` function in the `car` package provided a lack-of-fit test for each predictor in the model. In this section, we will consider a general approach to assessing goodness-of-fit that is analogous to the Bayesian p-value we saw in Section 13.4.

First, it may be helpful to review the different steps involved in any statistical hypothesis test in the context of a goodness-of-fit test:

1. State Null and Alternative hypotheses:
   - H_0: the data are consisten with the assumed Poisson regression model given by eq. (15.1)
   - H_A: the data are not consistent with the assumed model
2. Calculate a test statistic that measures the discrepancy between the data and the null hypothesis.
3. Determine the distribution of the test statistic in step 2, *when the null hypothesis is true.*
4. Compare the statistic for the observed data [step 2] to the distribution of the statistic when the null hypothesis is true [step 3]. The p-value is the probability that the GOF statistic for the observed data is as or more extreme than GOF statistic for data collected under situations where the null hypothesis is true.

If p-value is small, then we have evidence that the model is not appropriate. If the p-value is not small, then we do not have enough evidence to suggest the model is not appropriate (whew!).

OK, let's begin by considering how to formulate a goodness-of-fit test for the Poisson model. Like the Bayesian p-value approach, we will want to account for our uncertainty due to having to estimate λ_i. Because `glm` uses maximum likelihood to estimate parameters, we know that the sampling distribution of $\hat{\beta}$, when n is large, is given by:

$$\hat{\beta} \sim N(\beta, cov(\hat{\beta})),$$

where $cov(\hat{\beta}) = I^{-1}(\hat{\beta})$, the inverse of the Hessian. For models fit using `glm`, we can access an estimate of the variance covariance matrix, $\widehat{cov}(\hat{\beta})$, using `vcov(model)`. Thus, we can account for parameter uncertainty by drawing random parameters from a multivariate normal distribution with mean $= \hat{\beta}$ and variance/covariance matrix $= \widehat{cov}(\hat{\beta})$. From these parameter values, we calculate $\hat{\lambda} = exp(X\hat{\beta})$ for each of our observations.

We also need the ability to simulate new data sets where the null hypothesis is true (in this case, our model is correct). We can do this using the `rpois` function. Lastly, we will need to calculate a measure of fit for both real and simulated data. There are many possibilities here depending on what sort of lack-of-fit you might want to be able to detect. One possibility is to use the Pearson χ^2 statistic (e.g., Kéry 2010, chap. 13):

$$\chi^2_{n-p} = \sum_{i=1}^{n} \frac{(Y_i - E[Y_i|X_i])^2}{Var[Y_i|X_i]}$$

.

The p-value for our test is given by the proportion of samples where $\chi^2_{sim} \geq \chi^2_{obs}$. If the model is appropriate, "goodness-of-fit" statistics for real and simulated data should be similar. If not, the model should fit the simulated data much better than the real data and our p-value will be small. A small p-value would lead us to reject the null hypothesis that our model is appropriate for the data.

We implement this approach using 10,000 simulated data sets below:

```r
# Number of simulations
  nsims <- 10000

# Set up matrix to hold goodness-of-fit statistics
  gfit.sim <- gfit.obs <- matrix(NA, nsims, 1)
  nobs <- nrow(longnosedace) # number of observations

# Extract the estimated coefficients and their asymptotic variance/covariance matrix
# Use these values to generate nsims new betas
  beta.hat <- MASS::mvrnorm(nsims, coef(glmPdace), vcov(glmPdace))

# Design matrix for creating new lambda^s
  xmat <- model.matrix(glmPdace)
  for(i in 1:nsims){
    # Generate lambda^s, note that %*% is used fo rmatrix mulitplication
    lambda.hat <- exp(xmat%*%beta.hat[i,])

    # Generate new simulated values
    new.y <- rpois(nobs, lambda = lambda.hat)

    # Calculate Pearson Chi-squared statistics
    gfit.sim[i,] <- sum((new.y-lambda.hat)^2/(lambda.hat))
    gfit.obs[i,] <- sum((longnosedace$longnosedace-lambda.hat)^2/lambda.hat)
  }
  # Calculate and print the p-value
  (GOF.pvalue <- mean(gfit.sim > gfit.obs))
```

```
## [1] 0
```

We see that the p-value is essentially 0, suggesting again that the Poisson model is not a good model for these data. The Pearson χ^2 statistic was always larger for the observed data than the simulated data.

15.5 Quasi-Poisson model for overdispersed data

What are the consequences of analyzing the data using Poisson regression when the data are overdispersed? If the model for the transformed mean, $\log(E[Y_i|X_i]) = \beta_0 + \beta_1 X_1 + \dots \beta_p X_p$ is appropriate, but the Poisson distribution is not, then our estimates of β may be unbiased but their standard errors may be too small. In addition, p-values for hypothesis tests will also be too small and may lead us to select overly complex models if we use hypothesis tests to select predictors to include in our models.

If we are happy with our model for the transformed mean, $\log(E[Y_i|X_i]) = \beta_0 + \beta_1 X_1 + \dots \beta_p X_p$, we might also consider two relatively simple approaches:

1. We can use a bootstrap to calculate standard errors and confidence intervals (see Chapter 2).
2. We can adjust our standard errors using a variance inflation parameter.

With both approaches, we relax the assumption that $Y_i|X_i \sim Poisson$. The second approach, however, adds an assumption regarding the variance of $Y_i|X_i$. In particular, we assume:

- $E[Y_i|X_i] = \mu_i = e^{\beta_0+\beta_1 X_{1,i}+\dots+\beta_p X_{p,i}}$
- $Var[Y_i|X_i] = \phi\mu_i = \phi e^{\beta_0+\beta_1 X_{1,i}+\dots+\beta_p X_{p,i}}$

Although we make assumptions about the mean and variance, we do not specify a probability distribution for the data, and therefore, we are not technically using likelihood-based methods. Rather, this approach has been termed quasi-likelihood (McCullagh and Nelder 1989)[3]. We can estimate the scale parameter, $\hat{\phi}$ by either:

- $\hat{\phi} = \frac{\text{Residual deviance}}{(n-p)}$
- $\hat{\phi} = \frac{1}{n-p}\sum_{i=1}^{n}\frac{(Y_i - E[Y_i|X_i])^2}{Var[Y_i|X_i]}$ (this is the approach R takes)

In R, we can fit the quasi-likelihood model as follows:

```
glmQdace<-glm(longnosedace ~ acreage + do2 + maxdepth + no3 + so4 + temp,
              data = longnosedace, family = quasipoisson())
summary(glmQdace)
```

[3]This approach is also similar to the Generalized Estimating Equation approach that we will see in Chapter 20.

```
##
## Call:
## glm(formula = longnosedace ~ acreage + do2 + maxdepth + no3 +
##     so4 + temp, family = quasipoisson(), data = longnosedace)
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.564e+00  1.528e+00  -1.024  0.30999
## acreage      3.843e-05  1.128e-05   3.408  0.00116 **
## do2          2.259e-01  1.153e-01   1.960  0.05460 .
## maxdepth     1.155e-02  3.626e-03   3.185  0.00228 **
## no3          1.813e-01  5.792e-02   3.130  0.00268 **
## so4         -6.810e-03  1.964e-02  -0.347  0.73001
## temp         7.854e-02  3.541e-02   2.218  0.03027 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 29.40332)
##
##     Null deviance: 2766.9  on 67  degrees of freedom
## Residual deviance: 1590.0  on 61  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

Here, the inflation parameter is estimated to be 29.4. Comparing the estimates and standard errors to our Poisson regression model, we see that $\hat{\beta}$ is unchanged, but the SEs are all larger by a factor of $\sqrt{\phi} = \sqrt{29.4}$.

When is it reasonable to use this approach? Zuur et al. (2009) suggest that a quasi-Poisson model may be appropriate when $\phi > 1.5$, but that other methods should be considered when ϕ is greater than 15 or 20. These other methods might include, Negative Binomial regression (see Section 15.6), zero-inflation models (see Chapter 17), or a Poisson-normal model (see Section 15.8.3).[4]

15.6 Negative Binomial regression

In my experience, most count data sets collected by ecologists are overdispersed relative to the Poisson distribution. The Negative Binomial distribution offers a useful alternative in these cases. We will consider the following parameterization of the Negative Binomial distribution:

$Y_i|X_i \sim \text{NegBin}(\mu_i, \theta)$, with $E[Y_i|X_i] = \mu_i$ and $var[Y_i|X_i] = \mu_i + \frac{\mu_i^2}{\theta}$

The Negative Binomial distribution is overdispersed relative to Poisson, with:

$$\frac{Var[Y_i|X_i]}{E[Y_i|X_i]} = 1 + \frac{\mu}{\theta}$$

When θ is large, the Negative Binomial model will converge to a Poisson with variance equal to the mean.

[4]Alternatively, one might try various transformations, including a $\log(y+1)$ transformation and then use a linear model. For a comparison of these approaches, see Warton et al. (2016).

We can use the `glm.nb` function in the `MASS` library to fit a Negative Binomial Regression model to the `dace` data. Like when using `glm` to fit Poisson regression models, the default is to model the log mean as a function of predictors and regression coefficients (eq. (15.6)):

$$\begin{gathered}
Dace_i \sim \text{Negative Binomial}(\mu_i, \theta) \\
\log(\mu_i) = \beta_0 + \beta_1 Acreage_i + \beta_2 DO2_i + \beta_3 maxdepth_i + \\
\beta_4 NO3_i + \beta_5 SO4_i + \beta_6 temp_i
\end{gathered}$$

```
glmNBdace <- MASS::glm.nb(longnosedace ~ acreage + do2 + maxdepth + no3 + so4 + temp,
                          data = longnosedace)
summary(glmNBdace)
```

```
##
## Call:
## MASS::glm.nb(formula = longnosedace ~ acreage + do2 + maxdepth +
##     no3 + so4 + temp, data = longnosedace, init.theta = 1.666933879,
##     link = log)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.946e+00  1.305e+00  -2.256 0.024041 *
## acreage      4.651e-05  1.300e-05   3.577 0.000347 ***
## do2          3.419e-01  1.050e-01   3.256 0.001130 **
## maxdepth     9.538e-03  3.465e-03   2.752 0.005919 **
## no3          2.072e-01  5.627e-02   3.683 0.000230 ***
## so4         -2.157e-03  1.517e-02  -0.142 0.886875
## temp         9.460e-02  3.315e-02   2.854 0.004323 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(1.6669) family taken to be 1)
##
##     Null deviance: 127.670  on 67  degrees of freedom
## Residual deviance:  73.648  on 61  degrees of freedom
## AIC: 610.18
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  1.667
##           Std. Err.:  0.289
##
##  2 x log-likelihood:  -594.175
```

The estimate of $\theta = 1.667$ suggests the data are overdispsersed relative to the Poisson. If we compare the Poisson and Negative Binomial models using AIC, we find strong evidence in favor of the Negative Binomial model:

```
AIC(glmPdace, glmNBdace)
```

```
##             df       AIC
## glmPdace     7 1936.8602
## glmNBdace    8  610.1754
```

Think-Pair-Share (Review): How could we use R's functions for probability distributions to calculate these AIC values?

```
df.model1 <- 7
df.model2 <- 8
-2*sum(dpois(longnosedace$longnosedace,
             lambda = predict(glmPdace, type = "resp"),
             log = TRUE)) + 2 * df.model1
```

```
## [1] 1936.86
```

```
-2*sum(dnbinom(longnosedace$longnosedace,
               mu = predict(glmNBdace, type = "resp"),
               size = glmNBdace$theta, log = TRUE)) + 2 * df.model2
```

```
## [1] 610.1754
```

Let's use the `residualPlot` function in the `car` library to construct a Pearson residual versus fitted value plot for both the Poisson and Negative Binomial regression models (Figure 15.8). Neither plot looks perfect, but the variance seems a little more constant for the Negative Binomial model.

```
residualPlot(glmPdace, main = "Poisson model")
residualPlot(glmNBdace, main = "Negative Binomial model")
```

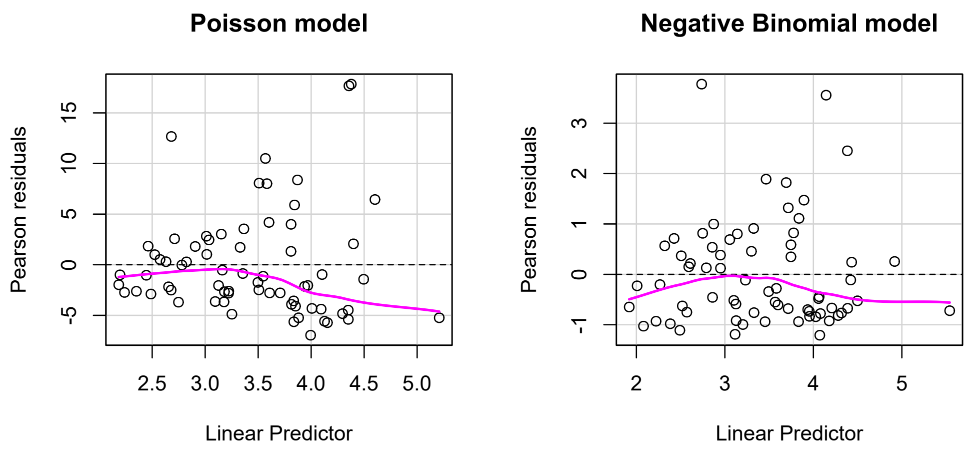


FIGURE 15.8 Plots of residuals versus each predictor and against fitted values using the `residualPlots` function in the `car` package (Fox and Weisberg 2019).

If we also inspect the lack-of-fit tests that the `residualPlots` function provides, we see that they are no longer significant when using an $\alpha = 0.05$.

```
residualPlots(glmNBdace, plot = FALSE)
```

```
##          Test stat Pr(>|Test stat|)
## acreage     0.0011          0.97373
## do2         3.5342          0.06012 .
## maxdepth    0.7947          0.37268
## no3         0.5994          0.43882
## so4         0.7818          0.37658
## temp        0.4307          0.51165
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The residual plots created using the `DHARMa` package also look considerably better (Figure 15.9).

```
simulateResiduals(glmNBdace, plot = TRUE)
```

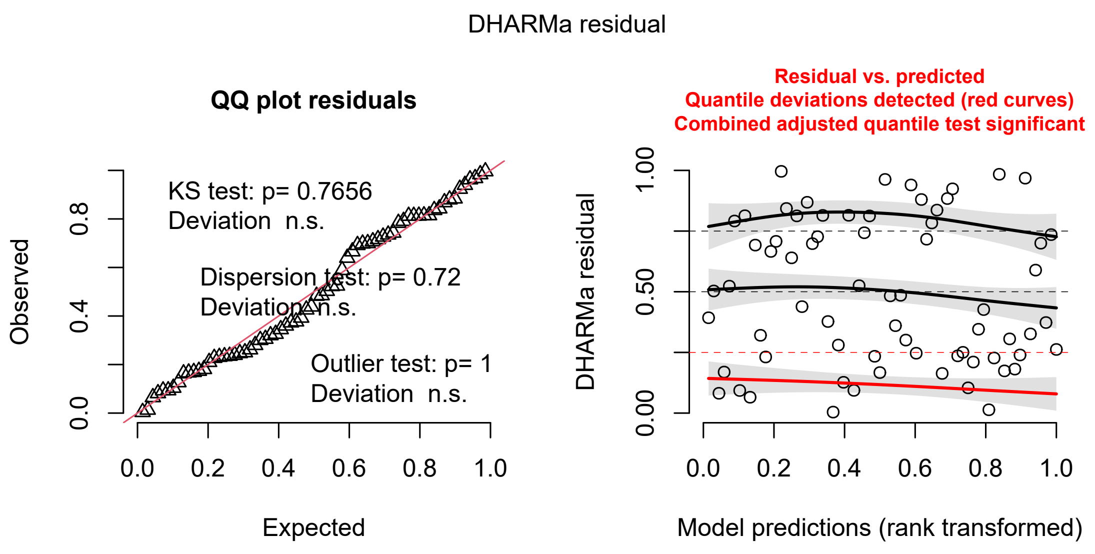


FIGURE 15.9 `DHARMa` residual plots for the Negative Binomial regression model fit to the longnose Dace data.

```
## Object of Class DHARMa with simulated residuals based on 250 simulations with refit = FALSE . See ?DHARM
##
## Scaled residual values: 0.3773173 0.1633754 0.7165224 0.3008422 0.7430748 0.09302904 0.8364012 0.735356
```

Lastly, we can adapt the goodness-of-fit test from Section 15.4.5 to the Negative Binomial model. To do so, we will assume the sampling distributions of $\hat{\beta}$ and $\hat{\theta}$ are independent. We can access the value of $\hat{\theta}$ and its estimated standard error using `glmNBdace$theta` and `glmNBdace$SE.theta`, respectively. We then just need to slightly modify the calculation of the Pearson χ^2 statistic to use the correct formulas for the mean and variance.

```r
# Number of simulations
  nsims <- 10000

# Set up matrix to hold goodness-of-fit statistics
  gfit.sim <- gfit.obs <- matrix(NA, nsims, 1)
  nobs <- nrow(longnosedace) # number of observations

# Extract the estimated coefficients and their asymptotic variance/covariance matrix
# Use these values to generate nsims new betas and thetas
  beta.hat <- MASS::mvrnorm(nsims, coef(glmNBdace), vcov(glmNBdace))
  theta.hat <- rnorm(nsims, glmNBdace$theta,  glmNBdace$SE.theta)

# Design matrix for creating new mu hats
  xmat <- model.matrix(glmNBdace)
  for(i in 1:nsims){

    # Generate E[Y|X] and var[Y|X], mu.hat and var.hat below
    mu.hat <- exp(xmat%*%beta.hat[i,])
    var.hat <- mu.hat + mu.hat^2/theta.hat[i]

    # Generate new simulated values
    new.y<-rnbinom(nobs, mu = mu.hat, size = theta.hat[i])

    # Calculate goodness-of-fit statistics
    gfit.sim[i,] <- sum((new.y - mu.hat)^2/(var.hat))
    gfit.obs[i, ] <- sum((longnosedace$longnosedace - mu.hat)^2/var.hat)
  }
  # Calculate and print the p-value
  (GOF.pvalue <- mean(gfit.sim > gfit.obs))
```

```
## [1] 0.2311
```

The p-value here is much larger, and suggests we do not have significant evidence to suggest that the Negative Binomial model is not appropriate for the data.

15.7 Model comparisons

Having settled on using a Negative Binomial model, we might now wonder if we can simplify our model at all by dropping one or more predictors. Again, my general modeling philosophy is to avoid model selection, preferring instead to report all coefficients and their standard errors in a full model (Giudice, Fieberg, and Lenarz 2012; Fieberg and Johnson 2015). On the other hand, there may be reasons for performing statistical hypothesis tests, particularly with experimental data.

For large samples, difference in deviances for nested models will be distributed according to a χ^2_{df} distribution, with df = difference in number of parameters:

$$D_2 - D_1 \sim \chi^2_{p_1 - p_2}$$

For quasi-Poisson models, we would need to divide the deviances by the overdispersion parameter, ϕ:

$$\frac{D_2 - D_1}{\phi} \sim \chi^2_{p_1 - p_2}$$

We can use `drop1(model, test="Chi")` (equivalent to a likelihood ratio test) or `Anova` in the `car` package to perform these tests. We can also use forward, backwards, stepwise selection (with the same potential issues that we discussed in Chapter 8) .

```
Anova(glmNBdace)
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: longnosedace
##          LR Chisq Df Pr(>Chisq)
## acreage   15.1095  1  0.0001015 ***
## do2        8.7214  1  0.0031449 **
## maxdepth   8.3963  1  0.0037598 **
## no3       15.5085  1  8.214e-05 ***
## so4        0.0227  1  0.8801254
## temp       7.5073  1  0.0061448 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The tests above are "Type II tests" of whether the coefficient with each variable is 0 in a model containing all of the other variables (Section 3.10). These tests suggest it would be reasonable to drop `sO4`. Alternatively, we can use AIC = -2(logL($\theta|y$)) + 2p for model selection with the advantage that it allows us to compare non-nested models. If we apply the `stepAIC` function to our Negative Binomial model, we see that it suggests dropping `so4` from the model. On the other hand, it may be preferable to stick with the full model and report confidence intervals associated with all of our predictor variables (see discussion in Chapter 8).

```
MASS::stepAIC(glmNBdace)
```

```
## Start:  AIC=608.18
## longnosedace ~ acreage + do2 + maxdepth + no3 + so4 + temp
##
##            Df    AIC
## - so4       1 606.20
## <none>        608.18
## - temp      1 613.35
## - maxdepth  1 614.13
## - do2       1 614.48
## - acreage   1 619.98
## - no3       1 620.32
##
## Step:  AIC=606.2
## longnosedace ~ acreage + do2 + maxdepth + no3 + temp
##
##            Df    AIC
## <none>        606.20
## - temp      1 611.35
## - do2       1 612.53
## - maxdepth  1 612.90
## - acreage   1 618.17
## - no3       1 618.57


##
## Call:  MASS::glm.nb(formula = longnosedace ~ acreage + do2 + maxdepth +
##     no3 + temp, data = longnosedace, init.theta = 1.666387265,
##     link = log)
##
## Coefficients:
## (Intercept)      acreage          do2     maxdepth          no3         temp
##  -2.977e+00    4.632e-05    3.422e-01    9.660e-03    2.069e-01    9.447e-02
##
## Degrees of Freedom: 67 Total (i.e. Null);  62 Residual
## Null Deviance:       127.6
## Residual Deviance: 73.65     AIC: 608.2
```

15.8 Bayesian implementation of count-based regression models

One of the main reasons to cover Bayesian methods is pedagogical. Fitting a model in JAGS requires that one clearly specify the model, including:

1. The likelihood, and hence the distribution of the response variable.
2. The relationship between the response and predictor variables.

In addition, as we saw in Chapter 13, it is easy to quantify uncertainty associated with average responses at a given set of predictor variables, $E[Y_i|X_i = x_i]$, as well as uncertainty associated with new observations (i.e., confidence intervals and prediction intervals, respectively). It is equally easy to estimate uncertainty associated with functions of parameters, whereas with maximum likelihood, we had to use the delta method and rely on large sample confidence intervals (see Section 10.7).

15.8.1 Poisson regression

The code, below, can be used to implement the Poisson regression model and also conduct the goodness-of-fit test based on the Pearson χ^2 statistic. We use a shortcut to specify the linear predictor. Specifically, we pass the matrix of predictor variables and use the `inprod` function to multiply this matrix by a set of regression coefficients, `beta`. I.e., we use `inprod(beta[], x[i,])` to calculate the linear predictor:

$$\log(\lambda_i) = \beta_0 + \beta_1 Acreage_i + \beta_2 DO2_i + \beta_3 maxdepth_i + \beta_4 NO3_i + \beta_5 SO4_i + \beta_6 temp_i$$

```r
library(R2jags)

#' Fit Poisson model using JAGS
pois.fit<-function(){
  # Priors for regression parameters
  for(i in 1:np){
    beta[i] ~ dnorm(0,0.001)
  }
  for(i in 1:n){
    log(lambda[i]) <- inprod(beta[], x[i,])
    longnosedace[i] ~ dpois(lambda[i])

  # Fit assessments
    Presi[i] <- (longnosedace[i] - lambda[i]) / sqrt(lambda[i]) # Pearson residuals
    dace.new[i] ~ dpois(lambda[i])    # Replicate data set
    Presi.new[i] <- (dace.new[i] - lambda[i]) / sqrt(lambda[i]) # Pearson residuals
    D[i] <- pow(Presi[i], 2)
    D.new[i] <- pow(Presi.new[i], 2)
 }

# Add up discrepancy measures
 fit <- sum(D[])
 fit.new <- sum(D.new[])
}


# Bundle data
jagsdata <- list(x = xmat, np = length(coef(glmPdace)),
                 n = nrow(longnosedace),
                 longnosedace = longnosedace$longnosedace)


# Parameters to estimate
params <- c("beta", "lambda", "Presi", "fit", "fit.new", "dace.new")


# Start Gibbs sampler
out.pois <- jags.parallel(data = jagsdata, parameters.to.save = params,
                 model.file = pois.fit, n.thin = 2, n.chains = 3, n.burnin = 5000,
                 n.iter = 20000)
```

We inspect posterior summaries for β_0 through β_6. This is made easier using the `MCMCvis` package.

```
library(MCMCvis)
# Look at results for the regression coefficients
MCMCsummary(out.pois, params="beta", round=3)
```

```
##           mean    sd   2.5%    50%  97.5% Rhat n.eff
## beta[1] -1.423 0.312 -2.037 -1.423 -0.822    1 14804
## beta[2]  0.000 0.000  0.000  0.000  0.000    1 18777
## beta[3]  0.188 0.024  0.140  0.188  0.236    1 15803
## beta[4]  0.015 0.001  0.013  0.015  0.016    1 16616
## beta[5]  0.191 0.011  0.170  0.191  0.214    1 17044
## beta[6] -0.004 0.004 -0.011 -0.004  0.003    1 14933
## beta[7]  0.084 0.007  0.071  0.084  0.097    1 17310
```

We can inspect the fit of the model by plotting posterior means of the Pearson residuals versus fitted values (Figure 15.10). We see a tendency for the Pearson residuals to increase as we move from left to right, again suggesting that we have not fully accounted for non-constant variance.

```
bresids <- MCMCpstr(out.pois, params = "Presi", func = mean)$Presi
bfitted <- MCMCpstr(out.pois, params = "lambda", func = mean)$lambda
jagsPfit <- data.frame(resid = bresids, fitted = bfitted)
ggplot(jagsPfit, aes(fitted, resid)) + geom_point() +
         geom_hline(yintercept = 0) + geom_smooth() +
     xlab("Fitted value") + ylab("Pearson Residual")
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

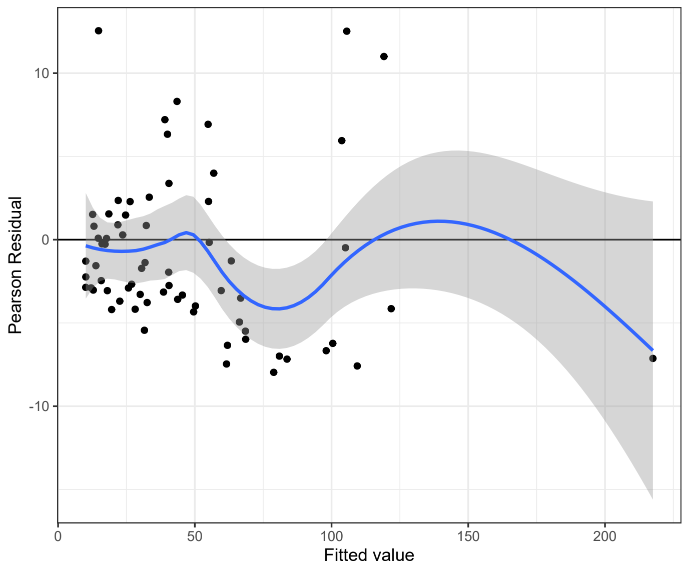


FIGURE 15.10 Pearson residuals versus fitted values for the Bayesian Poisson regression model fit to the longnose dace data.

The Bayesian p-value for the goodness-of-fit test is again ≈ 0. Thus, we come to the same conclusion as when fitting the model using `glm`; namely, the Poisson model does not fit the data very well.

```
fitstats <- MCMCpstr(out.pois, params = c("fit", "fit.new"), type = "chains")
T.extreme <- fitstats$fit.new >= fitstats$fit
(p.val <- mean(T.extreme))
```

```
## [1] 0
```

15.8.2 Negative Binomial regression

In order to fit the Negative Binomial model in JAGS, we have to remember that the negative binomial distribution is parameterized differently in JAGS than in R. In R, we have two possible parameterizations of the Negative Binomial distribution:

- dnbinom, with parameters (`prob`= p, `size` = n) or (`mu`= μ, `size`= θ)

In JAGS, the Negative Binomial model is parameterized as:

- dnegbin with parameters (p, θ)

When specifying the model in JAGS, we can write out the model in terms of μ and θ, but then we need to solve for $p = \frac{\theta}{\theta+\mu}$ and pass p and θ to the `dnegbin` function. We also need to modify the Pearson residuals to use the variance of the Negative Binomial distribution, $\mu + \frac{\mu^2}{\theta}$.

```
  #' Fit negative binomial model using JAGS
nb.fit<-function(){

  # Priors for regression parameters
  for(i in 1:np){
    beta[i] ~ dnorm(0,0.001)
  }
  theta~dunif(0,50)
  for(i in 1:n){
    log(mu[i]) <- inprod(beta[], x[i,])
    p[i] <- theta/(theta+mu[i])
    longnosedace[i] ~ dnegbin(p[i],theta)

  # Fit assessments
    Presi[i] <- (longnosedace[i] - mu[i]) / sqrt(mu[i]+mu[i]*mu[i]/theta) # Pearson residuals
    dace.new[i] ~ dnegbin(p[i],theta)    # Replicate data set
    Presi.new[i] <- (dace.new[i] - mu[i]) / sqrt(mu[i]+mu[i]*mu[i]/theta) # Pearson residuals
    D[i] <- pow(Presi[i], 2)
    D.new[i] <- pow(Presi.new[i], 2)
 }

# Add up discrepancy measures
 fit <- sum(D[])
 fit.new <- sum(D.new[])
}


# Parameters to estimate
params <- c("mu", "beta", "theta","Presi", "fit", "fit.new", "dace.new")


# Start Gibbs sampler
out.nb <- jags.parallel(data = jagsdata, parameters.to.save = params,
               model.file = nb.fit, n.thin = 2, n.chains = 3, n.burnin = 5000,
                 n.iter = 20000)
```

The residual plot looks a little bit better for the Negative Binomial model (Figure 15.11) and the p-value for the goodness-of-fit test is now much larger than 0.05.

```
nbresids <- MCMCpstr(out.nb, params = "Presi", func = mean)$Presi
nbfitted <- MCMCpstr(out.nb, params = "mu", func = mean)$mu
jagsnbfit <- data.frame(resid = nbresids, fitted = nbfitted)
ggplot(jagsnbfit, aes(fitted, resid)) + geom_point() +
         geom_hline(yintercept = 0) + geom_smooth() +
     xlab("Fitted value") + ylab("Pearson Residual")
```

```
## ‘geom_smooth()‘ using method = ’loess’ and formula = ’y ~ x’
```

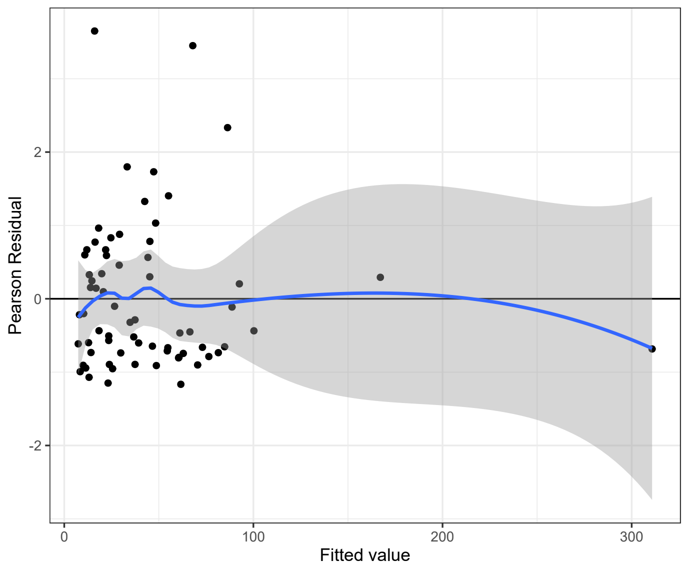


FIGURE 15.11 Pearson residuals versus fitted values for the Negative Binomial regression model fit to the longnose dace data.

```
fitstats <- MCMCpstr(out.nb, params = c("fit", "fit.new"), type = "chains")
T.extreme <- fitstats$fit.new >= fitstats$fit
(p.val <- mean(T.extreme))
```

```
## [1] 0.3171111
```

Lastly, for fun, we compare the frequentist and Bayesian 95% confidence/credible intervals for the Negative Binomial Model (Figure 15.12).

```
# Summary of the Bayesian model
Nbsum<-MCMCsummary(out.nb, params=c("beta","theta"), round=3)

# Confidence intervals Frequentist model
ci.nb<-confint(glmNBdace)

# Create data frame with results
est.dat<-data.frame(
  bhat = c(Nbsum[, 1], coef(glmNBdace), glmNBdace$theta),
  lowci = c(Nbsum[, 3], ci.nb[, 1], glmNBdace$theta - 1.96 * glmNBdace$SE.theta),
  upci = c(Nbsum[, 5], ci.nb[, 2], glmNBdace$theta + 1.96 * glmNBdace$SE.theta),
  Method = c(rep("Freq", 8), rep("Bayes", 8)),
  coefname = rep(c(names(coef(glmNBdace)), "theta"), 2))

ggplot(est.dat, aes(coefname, bhat, colour = Method)) +
  geom_point(size = 5, shape = 18, position = position_dodge(width = 0.5)) +
  geom_errorbar(aes(ymax = upci, ymin = lowci, colour = Method),
      position = position_dodge(width = 0.5)) + ylab("Point Estimate") +
  xlab("Coefficient") +  scale_colour_colorblind()
```

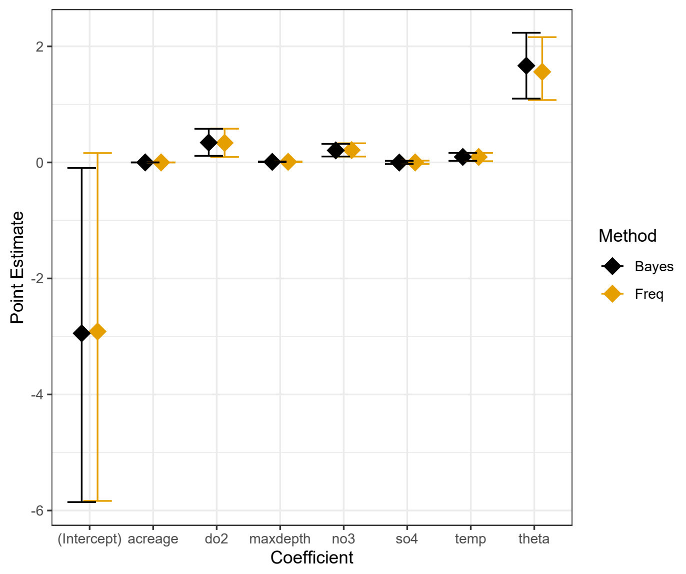


FIGURE 15.12 Comparison of frequentist and Bayesian Negative Binomial Models fit to longnose dace data.

15.8.3 Poisson-Normal model

Another alternative model that is sometimes considered for overdispersed data is a Poisson-Normal model (Harrison 2014):

$$Y_i|X_i, \epsilon_i \sim Poisson(\lambda_i)$$

$$log(\lambda_i) = \log(\mu_i) = \beta_0 + \beta_1 x_1 + \dots \beta_p x_p + \epsilon_i$$

$$\epsilon_i \sim N(0, \sigma^2)$$

In order to calculate the mean and variance of $Y_i|X_i$, we have to integrate over the distribution of ϵ_i, which leads us to the following expressions:

- $E[Y_i|X_i] = e^{\beta_0+\beta_1 X_{1,i}+\dots\beta_p X_{p,i}+\frac{1}{2}\sigma^2}$

- $Var[Y_i|X_i] = \mu_i + (e^{\sigma^2} - 1)\mu_i^2$

Note, that the expression for the mean is not the same as $E[Y|X, \epsilon = 0] = \exp^{\beta_0+\beta_1 X_1 + \dots \beta_p X_p}$. We will encounter similar complexities when we include random effects on the link scale to model correlation among repeated measures on the same sample units (Chapter 19).

Below, I fit this model to the dace data:

```
#' Fit Poisson model using JAGS
poisnorm.fit<-function(){

  # Priors for regression parameters
  for(i in 1:np){
    beta[i] ~ dnorm(0,0.001)
  }

  # Prior for sigma and tau
  sig ~ dunif(0, 3)
  tau <- 1/(sig*sig)


  for(i in 1:n){
  # Likelihood
    log(lambda[i]) <- inprod(beta[], x[i,]) + eps[i]
    longnosedace[i] ~ dpois(lambda[i])
    eps[i] ~ dnorm(0, tau)

  # Calculate mean and variance
    mu[i] <-  exp(inprod(beta[], x[i,]) + sig^2/2)
    vary[i] <- mu[i] + (exp(sig^2)-1)*mu[i]^2

  # Fit assessments
    Presi[i] <- (longnosedace[i] - mu[i]) / sqrt(vary[i]) # Pearson residuals
    dace.new[i] ~ dpois(lambda[i])    # Replicate data set
    Presi.new[i] <- (dace.new[i] - mu[i]) / sqrt(vary[i]) # Pearson residuals
    D[i] <- pow(Presi[i], 2)
    D.new[i] <- pow(Presi.new[i], 2)
 }

# Add up discrepancy measures
 fit <- sum(D[])
 fit.new <- sum(D.new[])
}

# Parameters to estimate
params <- c("beta", "mu", "Presi", "fit", "fit.new")


# Start Gibbs sampler
out.poisnorm <- jags.parallel(data = jagsdata, parameters.to.save = params,
                 model.file = poisnorm.fit, n.thin = 2, n.chains = 3, n.burnin = 5000,
                 n.iter = 20000)
```

The residual versus fitted value plot for the Poisson-Normal model looks reasonable (Figure 15.13).

```
pnresids <- MCMCpstr(out.poisnorm, params = "Presi", func = mean)$Presi
pnfitted <- MCMCpstr(out.poisnorm, params = "mu", func = mean)$mu
jagspnfit <- data.frame(resid = pnresids, fitted = pnfitted)
ggplot(jagspnfit, aes(fitted, resid)) + geom_point() +
         geom_hline(yintercept = 0) + geom_smooth() +
     xlab("Fitted value") + ylab("Pearson Residual")
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

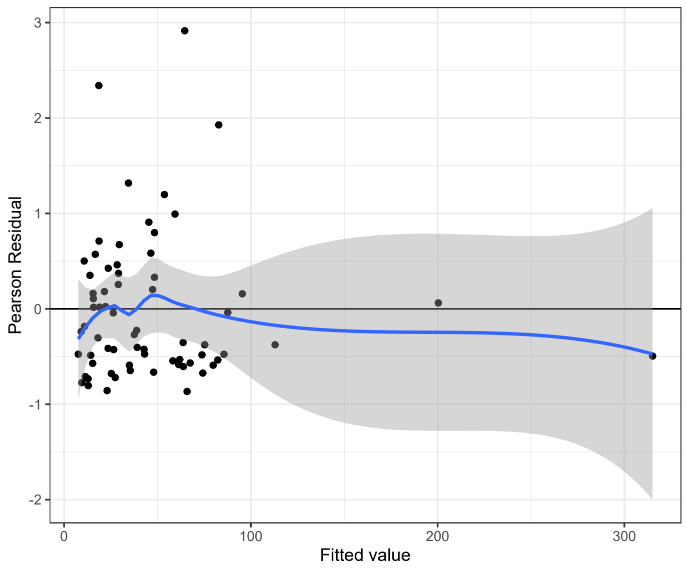


FIGURE 15.13 Pearson residuals versus fitted values for the Poisson-Normal regression model fit to the longnose dace data.

In addition, the Bayesian p-value is close to 0.5, suggesting the Poisson-Normal model is a reasonable data generating model for these data.

```
fitstats <- MCMCpstr(out.poisnorm, params = c("fit", "fit.new"), type = "chains")
T.extreme <- fitstats$fit.new >= fitstats$fit
(p.val <- mean(T.extreme))
```

```
## [1] 0.5211556
```

15.8.4 Aside: Bayesian model selection

There are several information criterion available for comparing models fit using Bayesian methods (see e.g., Hooten and Hobbs 2015 for a review). One of the most popular, though perhaps not the most appropriate, is DIC = Deviance Information Criterion.

Recall, the deviance, D = -2$logL(\theta|y)$. In Bayesian implementations, the fact that we have a posterior distribution for our parameters, θ, means that we also have a posterior distribution of the deviance. DIC is calculated from:

- $\overline{D}$ = the mean of the posterior deviance
- $\hat{D}$ = the deviance calculated using the posterior means of the different model parameters

DIC uses the difference between these two quantities to estimate the effective number of parameters in a model, $p_D = \overline{D} - \hat{D}$. DIC is then estimated using:

DIC $= \hat{D} + 2p_D = (\overline{D} - p_D) + 2p_D = \overline{D} + p_D$

JAGS reports DIC in its summary, which can also be accessed as follows:

```
out.pois$BUGSoutput$DIC
```

```
## [1] 2474.525
```

```
out.nb$BUGSoutput$DIC
```

```
## [1] 610.9902
```

```
out.poisnorm$BUGSoutput$DIC
```

```
## [1] 473.6137
```

These results mirror the AIC comparison between the Poisson and Negative Binomial model but also suggest that the Poisson-Normal model may offer the best fit to the data.

15.8.5 Cautionary statements regarding DIC

DIC is not without problems. Here is what Martyn Plummer, the creator of JAGS has to say:

> The deviance information criterion (DIC) is widely used for Bayesian model comparison, despite the lack of a clear theoretical foundation. . . .valid only when the effective number of parameters in the model is much smaller than the number of independent observations. In disease mapping, a typical application of DIC, this assumption does not hold and DIC under-penalizes more complex models. Another deviance-based loss function, derived from the same decision-theoretic framework, is applied to mixture models, which have previously been considered an unsuitable application for DIC.

And, Andrew Gelman had this to say:

> I don't really ever know what to make of DIC. On one hand, it seems sensible. . . On the other hand, I don't really have any idea what I would do with DIC in any real example. In our book

we included an example of DIC–people use it and we don't have any great alternatives–but I had to be pretty careful that the example made sense. Unlike the usual setting where we use a method and that gives us insight into a problem, here we used our insight into the problem to make sure that in this particular case the method gave a reasonable answer.

There are other options for comparing models fit in a Bayesian framework. Two popular options are LOO (Vehtari and Lampinen 2002) and WAIC (Watanabe and Opper 2010), which are available in the `loo` package (Vehtari et al. 2020). Again, I recommend reading Hooten and Hobbs (2015) to learn more about these metrics; see also Vehtari, Gelman, and Gabry (2017).

15.9 Modeling rates and densities using an offset

Count data are often collected over varying lengths of time or in sample units that have different areas. We can account for variable survey effort by modeling rates (counts per unit time) or densities (counts per unit area) using Poisson or Negative Binomial regression with an offset. Example applications include models of catch per unit effort (CPUE) in fisheries applications (e.g., Z. Su and He 2013), encounter rates in camera-trap studies (e.g., Scotson et al. 2017), and visitation rates in pollinator studies (Reitan and Nielsen 2016).

An offset is like a special predictor variable where we set its regression coefficient to `1`. To see how this works, consider a model for the log(rate) as a linear function of predictors:

$$\log(E[Y_i|X_i]/\text{Time}) = \beta_0 + \beta_1 X_{1,i} + \ldots + \beta_p X_{p,i}$$

$$\Rightarrow \log(E[Y_i|X_i]) - \log(\text{Time}) = \beta_0 + \beta_1 X_{1,i} + \ldots + \beta_p X_{p,i}$$

$$\Rightarrow \log(E[Y|X]) = \log(\text{Time}) + \beta_0 + \beta_1 X_{1,i} + \ldots + \beta_p X_{p,i}$$

We can include log(Time) as an **offset**, e.g., using R:

```
glm(y~x + offset(log(time)), data= , family = poisson())
```

This ensures that log(Time) is taken into account when estimating the other regression coefficients, but we do not estimate a coefficient for log(Time). We can think of this model as one for the rate, or alternatively, a model for the count where the mean count is determined by a rate parameter, λ_i, multiplied by the total effort, E_i, such that $E[Y_i|X_i] = \lambda_i E_i$.

What are the advantages to modelling a rate using a count distribution (Poisson or negative binomial) with an offset rather than modeling the rate directly using a continuous statistical distribution (e.g., log-normal)? First, the log-normal distribution cannot handle 0's, so many users will add a small constant, c, and model $\log(Y + c)$ (O'Hara and Kotze 2010), but results may be sensitive to the size of this constant. Furthermore, we lose some information by modeling the rates directly. Consider two observations that have a 0 count, but one observation has twice the effort of the other observation. If we model the count divided by effort, both observations are treated identically (since in both cases $y_i = 0$). The rate model, by contrast, will naturally account for the differing levels of effort.

Although modeling rates and densities is often appealing, it may be useful to consider including a measure of log(effort) as a covariate first. When the regression coefficient for log(effort) is close to 1, then it can safely be replaced with an offset. If the coefficient is far from 1, however, it may be preferable to retain the covariate.

To demonstrate this approach, we consider catch and effort data of black sea bass (*Centropristis striata*) from the Southeast Reef Fish Survey. The data are contained in `reeffish` within the `Data4Ecologists` package.

```
data(reeffish)
head(reeffish)
```

```
##   Year      Date  Time Doy Hour Duration Temperature Depth Seabass
## 1 1998 7/23/1998 11:50 204   12       97    9.495300    53       0
## 2 1998 7/23/1998 11:53 204   12      101    9.495300    52       0
## 3 1998 7/23/1998 13:56 204   14       96    9.548500    52       0
## 4 1998 7/23/1998 18:11 204   18       89    9.750867    52       0
## 5 1998 7/23/1998 18:18 204   18       91    9.750867    47       0
## 6 1998 7/23/1998 18:34 204   19       91    9.750867    52       0
```

The data set contains a count of sea bass (`Seabass`) captured in Chevron traps, along with the sampling duration (in minutes) associated with each observation (`Duration`) for surveys conducted between 1990 and 2021. Potential predictor variables include Julian date (`Doy`), hour of the day (`Hour`), temperature (`Temperature`), and water depth (`Depth`). We might also expect catch-per-unit effort to vary annually, and thus, we will include `Year` as a categorical variable[5]. We will consider a quadratic function to model a non-linear effect of `Temperature` on the log scale.

The data from the survey are highly variable, with many observations of 0 sea bass in the trap[6]. A Poisson model fit to the data will fail to pass standard goodness-of-fit tests, so we will go directly to fitting the following Negative Binomial model using `glm.nb` in the `MASS` package (Venables and Ripley 2002):

$$C_{i,t} \sim \text{Negative Binomial}(\lambda_{i,t}E_{i,t}, \theta)$$
$$\log(\lambda_i E_{i,t}) = \beta_{0,t} + \beta_1 Temperature_{i,t} + \beta_2 Temperature^2_{i,t} +$$
$$\beta_3 Depth_{i,t} + \beta_4 Doy_{i,t} + \beta_5 Hour_{i,t}$$

Here, $C_{i,t}$ and $E_{i,t}$ are the catch and effort (sampling duration) associated with the i^{th} observation in year t, $\lambda_{i,t}$ is the expected catch per unit effort in the survey, and θ is an overdispersion parameter of the Negative Binomial distribution. Note, we have written the model as though fit using a means parameterization for `Year` so that we can simply capture the different intercepts in each year with the term $\beta_{0,t}$. We can fit the model using:

```
reeffish <- reeffish %>% mutate(logduration = log(Duration))
fitoffset <- glm.nb(Seabass ~ as.factor(Year) + poly(Temperature,2) +
                      Depth + Doy + Hour + offset(logduration) -1, data = reeffish)
summary(fitoffset)
```

```
##
## Call:
## glm.nb(formula = Seabass ~ as.factor(Year) + poly(Temperature,
##     2) + Depth + Doy + Hour + offset(logduration) - 1, data = reeffish,
##     init.theta = 0.2529787278, link = log)
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## as.factor(Year)1990    2.259e+00  1.594e-01  14.174  < 2e-16 ***
```

[5]Alternatively, we could explore temporal *trends* in catch rates by modeling `Year` as a continuous variable. Further, we might consider modeling annual variation using random effects (see Chapters 18, 19).

[6]Zero-inflation models considered in Chapter 17 would offer another alternative.

```
## as.factor(Year)1991    1.949e+00  1.778e-01  10.960  < 2e-16 ***
## as.factor(Year)1992    2.010e+00  1.637e-01  12.280  < 2e-16 ***
## as.factor(Year)1993    1.723e+00  1.613e-01  10.687  < 2e-16 ***
## as.factor(Year)1994    1.614e+00  1.657e-01   9.740  < 2e-16 ***
## as.factor(Year)1995    1.426e+00  1.693e-01   8.426  < 2e-16 ***
## as.factor(Year)1996    1.686e+00  1.680e-01  10.038  < 2e-16 ***
## as.factor(Year)1997    1.700e+00  1.685e-01  10.087  < 2e-16 ***
## as.factor(Year)1998    1.728e+00  1.615e-01  10.698  < 2e-16 ***
## as.factor(Year)1999    2.201e+00  1.882e-01  11.701  < 2e-16 ***
## as.factor(Year)2000    2.015e+00  1.783e-01  11.301  < 2e-16 ***
## as.factor(Year)2001    2.433e+00  1.847e-01  13.174  < 2e-16 ***
## as.factor(Year)2002    1.229e+00  1.943e-01   6.326 2.51e-10 ***
## as.factor(Year)2003    6.698e-01  2.054e-01   3.261  0.00111 **
## as.factor(Year)2004    2.339e+00  1.795e-01  13.031  < 2e-16 ***
## as.factor(Year)2005    1.777e+00  1.745e-01  10.184  < 2e-16 ***
## as.factor(Year)2006    1.967e+00  1.772e-01  11.098  < 2e-16 ***
## as.factor(Year)2007    1.446e+00  1.739e-01   8.318  < 2e-16 ***
## as.factor(Year)2008    1.563e+00  1.789e-01   8.740  < 2e-16 ***
## as.factor(Year)2009    1.258e+00  1.622e-01   7.757 8.71e-15 ***
## as.factor(Year)2010    2.530e+00  1.462e-01  17.302  < 2e-16 ***
## as.factor(Year)2011    2.700e+00  1.446e-01  18.678  < 2e-16 ***
## as.factor(Year)2012    2.666e+00  1.331e-01  20.026  < 2e-16 ***
## as.factor(Year)2013    2.376e+00  1.303e-01  18.234  < 2e-16 ***
## as.factor(Year)2014    2.091e+00  1.276e-01  16.390  < 2e-16 ***
## as.factor(Year)2015    1.754e+00  1.288e-01  13.620  < 2e-16 ***
## as.factor(Year)2016    1.492e+00  1.355e-01  11.014  < 2e-16 ***
## as.factor(Year)2017    1.287e+00  1.280e-01  10.049  < 2e-16 ***
## as.factor(Year)2018    1.110e+00  1.245e-01   8.913  < 2e-16 ***
## as.factor(Year)2019    6.354e-01  1.276e-01   4.981 6.32e-07 ***
## as.factor(Year)2021   -1.408e-01  1.245e-01  -1.131  0.25811
## poly(Temperature, 2)1 -3.263e+01  2.959e+00 -11.028  < 2e-16 ***
## poly(Temperature, 2)2  1.818e+01  2.343e+00   7.760 8.48e-15 ***
## Depth                 -1.388e-01  1.535e-03 -90.451  < 2e-16 ***
## Doy                    5.426e-04  4.359e-04   1.245  0.21317
## Hour                   8.856e-03  4.679e-03   1.893  0.05839 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(0.253) family taken to be 1)
##
##     Null deviance: 43485  on 21039  degrees of freedom
## Residual deviance: 16826  on 21003  degrees of freedom
## AIC: 89995
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  0.25298
##           Std. Err.:  0.00361
##
##  2 x log-likelihood:  -89921.25400
```

If we look at the coefficients in the summary of the model, we see that we do not estimate a coefficient for `log(Duration)`. Yet, this information is accounted for in the model by including the `offset(log(Duration))` term, which allows us to model the catch rate rather than the count of sea bass observed.

We could also fit a model that includes `log(Duration)` as a predictor. We can examine the coefficient for `log(Duration)` and also compare the two models using AIC to determine if the latter model does a better job of fitting the data than the catch-rate model.

```
fitpredictor <- glm.nb(Seabass ~ as.factor(Year) + poly(Temperature,2) +
                       Depth + Doy + Hour + logduration -1, data = reeffish)
AIC(fitoffset, fitpredictor)
```

```
##              df      AIC
## fitoffset    37 89995.25
## fitpredictor 38 89996.88
```

```
# Profile-likelihood confidence interval
round(confint(fitpredictor, parm = "logduration"), 2)
```

```
## Waiting for profiling to be done...
```

```
##  2.5 % 97.5 %
##   0.65   1.19
```

In this case, we see that AIC slightly prefers the model with the offset. Further, the confidence interval for the coefficient associated with `log(Duration)` includes 1. Thus, we have support for using the simpler model, which also has the advantage of interpretation – i.e., we can simply state that we are modeling the catch per unit effort. We end by using effect plots, constructed using the `ggpredict` package, to visualize the importance of the predictors (Figure 15.14). These plots depict estimates of adjusted means formed varying a focal predictor (shown on the x-axis) while holding all other predictor variables at their sample means (Section 3.14.3). We see that the effects of depth and temperature far outweigh those of the other variables.

```
sjPlot::plot_grid(plot(ggpredict(fitoffset),  one_plot = FALSE))
```

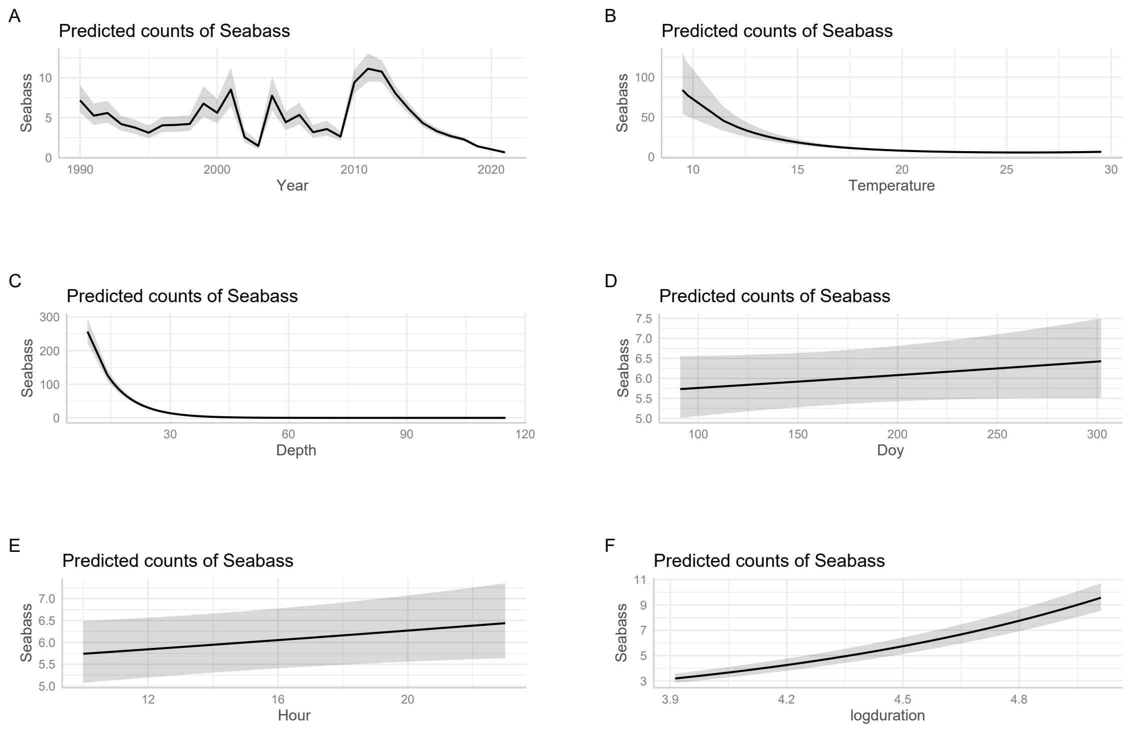


FIGURE 15.14 Effect plots depicting estimates of marginal means for the Negative Binomial regression model fit to the black sea bass data.

16

Logistic regression

Learning objectives

- Be able to formulate, fit, and interpret logistic regression models appropriate for binary data using R and JAGS
- Be able to compare models and evaluate model fit
- Be able to visualize models using effect plots
- Be able to describe statistical models and their assumptions using equations and text and match parameters in these equations to estimates in R output

16.1 R packages

We begin by loading a few packages upfront:

```
library(knitr) # for chunk options
library(kableExtra) # for tables
options(kableExtra.html.bsTable = T)
library(patchwork) # for multi-panel plots
library(ggplot2) # for plotting
library(dplyr) # for data wrangling
library(car) # for residual plots
library(R2jags) # for fitting models using JAGS
library(MCMCvis) # for visualizing MCMC output
library(modelsummary) # for summarizing models
library(MASS) # for generating multivariate random normal variables
library(ggeffects) # for summarizing fitted models
library(effects) # for summarizing fitted models
library(performance) # for residual plots
library(ggthemes) # for colorblind pallete
```

In addition, we will use data and functions from the following packages:

- `SightabilityModel` for the moose sightability data
- `sos` package to search for functions to accomplish specific tasks
- `ResourceSelection` for the `hoslem.test` function
- `LaplacesDemon` for student-t distribution
- `glmx` for the `BeetleMortality` data set
- `Data4Ecologists` for the `wells` data set
- `logistf` for fitting a penalized likelihood version of a logistic regression model

16.2 Introduction to logistic regression

Often, we are interested in understanding or predicting binary response variables (i.e., variables that can take on only 1 of two values, Yes/No, Alive/Dead, Infected/Not-infected). Logistic regression is used to relate a binary response variable, Y_i , to a set of explanatory variables, $X_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,p})$. The model assumes Y_i follows a Bernoulli distribution with probability of success, $P(Y_i = 1) = p_i$, that depends on predictors, X_i, via the following set of equations:

$$Y_i|X_i \sim \text{Bernoulli}(p_i)$$

$$\log\left(\frac{p_i}{1-p_i}\right) = \text{logit}(p_i) = \beta_0 + \beta_1 X_{1,i} + \ldots \beta_p X_{p,i}$$

The quantity $\frac{p}{1-p}$ is referred to as the odds of success, and the link function, $\log\left(\frac{p}{1-p}\right)$, is referred to as logit function. Thus, we can describe our model in the following ways:

- We are modeling $\log\left(\frac{p}{1-p}\right)$ as a linear function of $X_1, \ldots, X_p$.
- We are modeling the logit of p as a linear function of $X_1, \ldots, X_p$.
- We are modeling the log odds of success (defined as $Y_i = 1$) as a linear function of $X_1, \ldots, X_p$.

To determine $E[Y_i|X_i] = p_i$, we need to apply the inverse-logit transformation:

$$p_i = g^{-1}(\beta_0 + \beta_1 X_{1,i} + \ldots \beta_p X_{p,i}) = \frac{\exp(\beta_0 + \beta_1 X_{1,i} + \ldots \beta_p X_{p,i})}{1 + \exp(\beta_0 + \beta_1 X_{1,i} + \ldots \beta_p X_{p,i})}.$$

As the linear predictor, $\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_{1,i} + \ldots \beta_p X_{p,i}$, ranges between $-\infty$ and ∞, the odds $=$ $\exp\left(\frac{p}{1-p}\right)$ ranges between 0 and ∞, and p is constrained to be between 0 and 1 (Table 16.1).

TABLE 16.1 Relationship between probability (p), odds, and log odds.

log odds $= \log\left(\frac{p}{1-p}\right)$	-6.907	-1.386	0.0	1.386	6.907
odds $= \frac{p}{1-p}$	0.001	0.250	1.0	4.000	999.000
p	0.001	0.200	0.5	0.800	0.999

We can gain further insights into the model by plotting p_i as a function of X_i for a logistic regression model with a single continuous predictor and associated intercept and slope parameters, β_0 and β_1, respectively (Figure 16.1). The sign of β_1 determines whether p increases or decreases as X_i increases, and its magnitude controls how quickly p_i transitions from 0 to 1

16.3 Parameter Interpretation: Application to moose detection data

In Chapter 11, we considered data collected by the Minnesota Department of Natural Resoureces (MN DNR) to estimate the probability of detecting a moose when flying in a helicopter (Giudice, Fieberg, and Lenarz

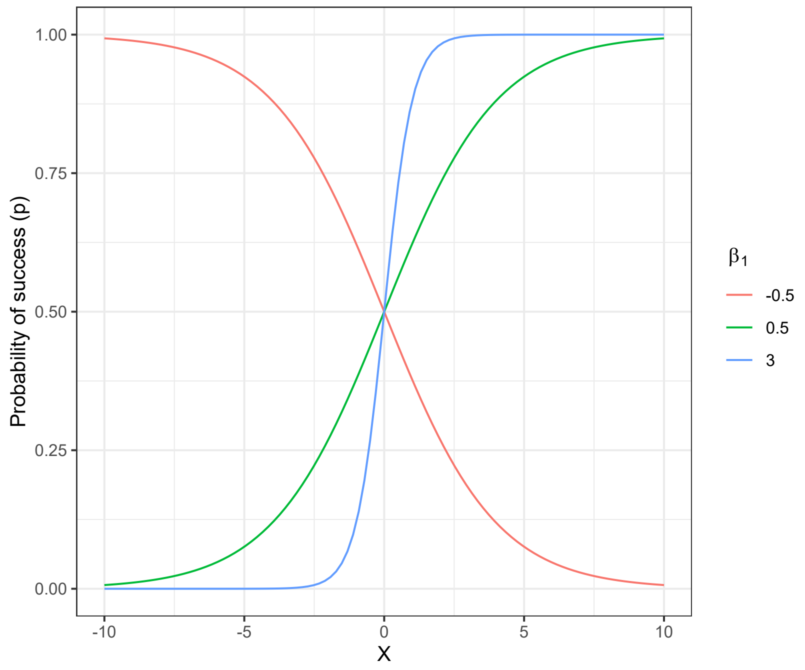


FIGURE 16.1 Probability of success for different slope coefficients in a logistic regression model with a single continuous predictor and $\beta_0 = 0$. The intercept β_0 controls the height of the curve when $X_i = 0$. For the curves depicted in Figure 16.1, $\beta_0 = 0$. Thus, $E[Y_i|X_i = 0] = \frac{exp(\beta_0)}{1+exp(\beta_0)} = 1/2$.

2012). The probability of detecting a moose will depend on where the moose is located (e.g., the amount of cover that shields the moose from view, termed *visual obstruction* (or VOC); Fig. 16.2). If we model the probability of detection as a function of VOC, we can then adjust counts of observed animals in future surveys that also record this information, providing a formal method for estimating moose abundance that accounts for imperfect detection (Fieberg 2012; Fieberg et al. 2013; ArchMiller et al. 2018). We might also want to consider whether detection probabilities vary by observer (a categorical variable) or whether detection probabilities varied among the four years of data collection.

Let's read in a data set with the observations recording whether each moose was observed or not (`observed`) along with potential covariates for modeling the probability of detection. The data are contained in the `SightabilityModel` package (Fieberg 2012) and can be accessed using:

```
library(SightabilityModel)
data(exp.m)
str(exp.m)
```

```
## 'data.frame':    124 obs. of  4 variables:
##  $ year    : int  2005 2005 2005 2005 2005 2005 2005 2005 2005 2005 ...
##  $ observed: int  1 1 0 0 0 0 1 1 1 1 ...
##  $ voc     : int  20 85 80 75 70 85 20 10 10 70 ...
##  $ grpsize : int  4 2 1 1 1 1 1 2 2 2 ...
```

The data set in this package contains a small subset of the variables that could be used to model detection probabilities (see Giudice, Fieberg, and Lenarz 2012 for a discussion regarding how these were chosen). We

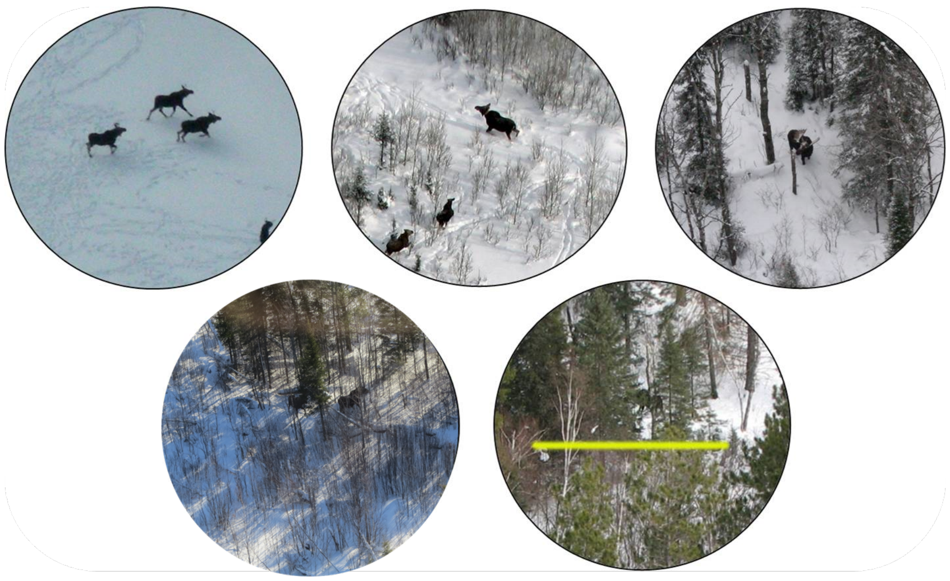


FIGURE 16.2 Observations of moose in Minnesota with a circular field of view used to measure the degree of visual obstruction. Photos by Mike Schrage, Fond du Lac Resource Management Division.

will focus on the relationship between detection and visual obstruction measurements using the variables `observed` and `voc`, respectively. Note that the variable `observed` is binary:

$$observed_i = \begin{cases} 1 & \text{if the radiocollared moose was detected} \\ 0 & \text{if the radiocollared moose was not detected} \end{cases}$$

In Figure 16.3, we use a smoother to visualize overall trends in detection probabilities relative to `voc`. As expected, we see that detection probabilities decrease as the amount of screening cover increases. We could fit a linear regression model to the data. However, this approach would fail to capture several of the characteristics of the data:

- Detection probabilities must necessarily fall between 0 and 1.
- The observations do not have have constant variance (recall the variance of a Bernoulli random variable = $p(1-p)$, which will be at its highest value when $p = 0.5$).

```
ggplot(exp.m, aes(voc,observed))+theme_bw()+
    geom_point(position = position_jitter(w = 2, h = 0.05), size=3) +
    geom_smooth(colour="red") + xlab("Visual Obstruction") +
    ylab("Detection = 1")
```

Letting $Y_i = observed_i$, and $X_i = voc_i$, we can specify a logistic regression model for the moose data using the following set of equations:

$$Y_i|X_i \sim Bernoulli(p_i)$$

$$logit(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 voc_i$$

We can fit this model using `glm` with `family =binomial()`.

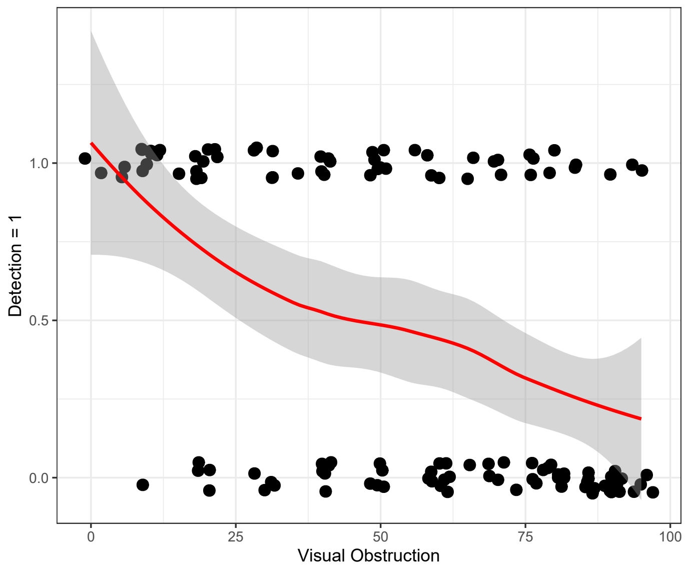


FIGURE 16.3 Detection of moose as a function of the amount of visual obstruction within an approximate 10m radius of the moose.

```
mod1 <- glm(observed ~ voc, data = exp.m, family = binomial())
```

By default, `glm` will use a logit link when specifying `family=binomial()`, but other links are possible. Similar to other regression models we've seen, we can use the `summary` function to view the estimated coefficients, their standard errors, and explore hypothesis tests for whether the coefficients are 0 (versus an alternative that they are not 0):

```
summary(mod1)
```

```
##
## Call:
## glm(formula = observed ~ voc, family = binomial(), data = exp.m)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.759933   0.460137   3.825 0.000131 ***
## voc         -0.034792   0.007753  -4.487 7.21e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 171.61  on 123  degrees of freedom
## Residual deviance: 147.38  on 122  degrees of freedom
## AIC: 151.38
##
## Number of Fisher Scoring iterations: 4
```

We see that the log-odds of detection decreases by -0.035 as we increase `voc` by 1 unit (i.e., for every increase

TABLE 16.2 Estimates of the effect of visual obstruction (β_1, $\exp(\beta_1)$, and 95% CIs) on moose detection probabilities.

	$\widehat{\beta}_1$	$SE(\widehat{\beta}_1)$	Lower 95% CL	Upper 95% CL	$\exp(\beta_1)$	Lower 95% CL	Upper 95% CL
voc	-0.035	0.008	-0.05	-0.02	0.966	0.951	0.981

of 1% visual obstruction). Alternatively, it is common to report effect sizes in terms of a change in odds. The odds of detection is given by:

$$\frac{p_i}{1-p_i} = \exp(\beta_0 + \beta_1 voc_i) = e^{\beta_0} e^{\beta_1 voc_i}$$

Thus, if we increase `voc` by 1 unit, we increase the odds by e^{β_1}. To calculate a confidence interval for the change in odds associated with a 1 unit increase in a predictor variable, it is common to calculate a confidence interval for β_1 and then exponentiate the limits (similar to what we saw for Poisson regression). Again, we can rely on large sample theory for Maximum Likelihood estimators, $\hat{\beta} \sim N(\beta, I^{-1}(\beta))$, and the output from the summary of the `glm` model to calculate a 95% confidence interval for β_1 and also for the change in odds (Table 16.2).

```
beta1 <- coef(mod1)[2]
SEbeta1 <- sqrt(diag(vcov(mod1)))[2]
oddsr <- exp(beta1)
CI_beta <- beta1 + c(-1.96, 1.96)*SEbeta1
CI_odds <- exp(CI_beta)
estdata <- data.frame(beta1 = beta1, SEbeta1 = SEbeta1,
           LCL_beta = CI_beta[1], UCL_beta = CI_beta[2], oddsratio  = oddsr,
           LCL_odds = CI_odds[1], UCL_odds = CI_odds[2])
colnames(estdata) <- c("$\\widehat{\\beta}_1$", "$SE(\\widehat{\\beta}_1)$",
                       "Lower 95\\% CL", "Upper 95\\% CL", "$\\exp(\\beta_1)$",
                       "Lower 95\\% CL", "Upper 95\\% CL")
kable(round(estdata,3), booktabs = TRUE, escape = FALSE,
      caption="Estimates of the effect of visual obstruction ($\\beta_1$, $\\exp(\\beta_1)$, and 95\\% CIs
```

Alternatively, we could calculate profile-likelihood based confidence intervals by inverting the likelihood ratio test (Section 10.10). Recall, profile-likelihood intervals include all values, $\widetilde{\beta}$, for which we would **not** reject the null hypothesis $H_0 : \beta = \widetilde{\beta}$. In small samples, profile likelihood intervals should perform better than the Normal-based intervals, and thus, this is the default approach used by the function `confint`.

```
(ci.prof <- confint(mod1))
```

```
## Waiting for profiling to be done...
```

```
##                   2.5 %      97.5 %
## (Intercept)  0.89794673  2.71375978
## voc         -0.05082098 -0.02025314
```

```
(exp(ci.prof))
```

```
##                  2.5 %     97.5 %
## (Intercept) 2.4545581 15.0858887
## voc         0.9504488  0.9799506
```

The Normal-based and profile-likelihood confidence intervals are similar in this case. If confidence limits for β do not include 0 or confidence limits for $\exp(\beta)$ do not include 1, then we might say that the results are statistically significant at the $\alpha = 0.05$ level.

We can also use `tab_model` in the `sjPlot` library or the `modelsummary` function in the `modelsummary` package to calculate a table of effect sizes for our model (Table 16.3). By default, `tab_model` reports effect sizes in terms of odds, labeled Odds Ratios. We can can also calculate odds ratios by adding the argument `exponentiate = TRUE` when calling the `modelsummary` function (Table 16.3).

```
modelsummary(list("Logistic regression" = mod1),
             exponentiate = TRUE, gof_omit = ".*", estimate  = "{estimate} ({std.error})",
             statistic=NULL,
             coef_omit = "SD",
             title = "Odds ratios (SE) from the logistic regression model fit
             to the moose detection data.") %>%
  kable_styling(latex_options = "HOLD_position")
```

TABLE 16.3 Odds ratios (SE) from the logistic regression model fit to the moose detection data.

	Logistic regression
(Intercept)	5.812 (2.674)
voc	0.966 (0.007)

To understand why $\exp(\hat{\beta})$ is referred to as an odds ratio, consider the ratio of odds for two observations that differ only in `voc` with $voc_2 = voc_1 + 1$. The ratio of odds for observation 2 relative to observation 1 is given by:

$$\frac{\frac{p_2}{1-p_2}}{\frac{p_1}{1-p_1}} = \frac{e^{\beta_0}e^{\beta_1 voc_2}}{e^{\beta_0}e^{\beta_1 voc_1}} = \frac{e^{\beta_1(voc_1+1)}}{e^{\beta_1 voc_1}} = e^{\beta_1}$$

What about the intercept? The intercept gives the log-odds of detection when `voc` $= 0$ (i.e., when moose are completely in the open). We can also use the `plogis` function in R to calculate the inverse-logit transform, $g^{-1}(\beta_0) = \frac{exp(\beta_0)}{1+exp(\beta_0)}$, which provides an estimate of the probability of success when all covariates are equal to 0 (or when all covariates are set to their means if we use centered and scaled predictors):

```
plogis(coef(mod1)[1])
```

```
## (Intercept)
##   0.8532013
```

Thus, the model predicts there is an 85% chance of detecting a moose when it is completely in the open (i.e., when `voc` $= 0$).

Lastly, we can visualize the model using `ggplot`, here with `stat_smooth` used to specify our regression model (Figure 16.4).

```r
ggplot(exp.m, aes(voc,observed)) + theme_bw() +
    geom_point(position = position_jitter(w = 2, h = 0.05), size=3) +
    xlab("Visual Obstruction") +
    stat_smooth(method="glm", method.args = list(family = "binomial") )+
    ylab("Detection (Yes = 1)")
```

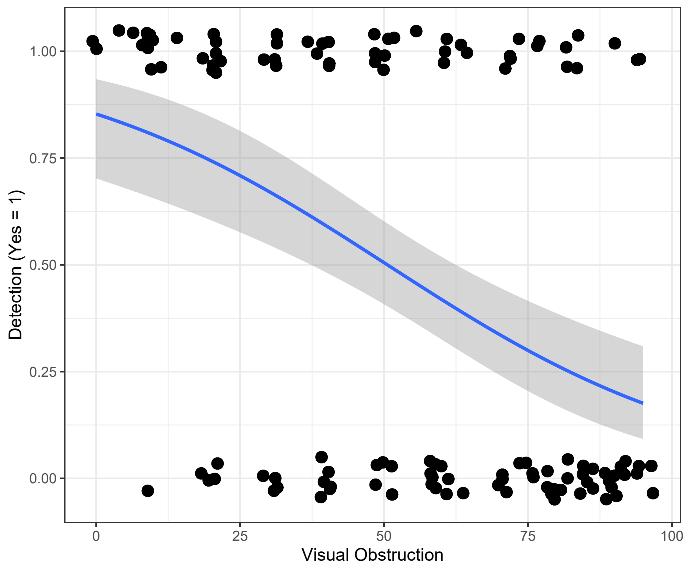


FIGURE 16.4 Fitted logistic regression model relating detection probabilities to the amount of visual obstruction.

16.3.1 Model with continuous and categorical variables

Let's now consider a second model, where we also include the year of the observation as a predictor. Rather than model a smooth trend in detection probabilities over time, we will use the `as.factor` function to create a categorical variable to represent the different years.

```r
exp.m$year <- as.factor(exp.m$year)
mod2 <- glm(observed ~ voc + year, data = exp.m, family = binomial())
summary(mod2)
```

```
Call:
glm(formula = observed ~ voc + year, family = binomial(), data = exp.m)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.453203   0.622248   3.942 8.06e-05 ***
voc         -0.037391   0.008199  -4.560 5.11e-06 ***
year2006    -0.453862   0.516567  -0.879   0.3796
year2007    -1.111884   0.508269  -2.188   0.0287 *
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 171.61  on 123  degrees of freedom
Residual deviance: 142.23  on 120  degrees of freedom
AIC: 150.23

Number of Fisher Scoring iterations: 4
```

Again, R uses reference coding by default. This provides another opportunity to revisit design matrices from Chapter 3 as well as further test our ability to interpret the regression coefficients when we include multiple predictor variables.

Think-Pair-Share: How can we interpret the coefficients in our model with `year` and `voc`?

Since we did not include an interaction between `voc` and `year`, we have assumed the effect of `voc` is the same in all years. We can interpret the effect of `voc` in the same way as in Section 16.3, except we must now tack on the phrase *while holding year constant*. I.e., the log odds of detection changes by 0.037 and the odds of detection by 0.96 for every 1 unit increase in `voc` while holding `year` constant.

What about the three coefficients associated with the year variables that R created? These coefficients provide estimates of *differences* in the log odds of detection between each listed year and 2005 (the reference level), *while holding `voc` constant*. And, if we exponentiate these coefficients, we get ratios of odds (i.e., odds ratios) between each year and 2005 (Table 16.4), *while holding `voc` constant*.

TABLE 16.4 Odds ratios = $\exp(\beta)$ (SE) from the fitted logistic regression model with voc and year.

	Logistic regression
(Intercept)	11.626 (7.234)
voc	0.963 (0.008)
year2006	0.635 (0.328)
year2007	0.329 (0.167)

Thus, for example, we can conclude that the odds of detection were $1/0.635 = 1.57$ times higher in 2005 than 2006 if we hold `voc` constant.

16.3.2 Interaction model

What if we were to include an interaction between `voc` and `year`?

```r
mod3 <- glm(observed ~ voc * year, data = exp.m, family = binomial())
summary(mod3)
```

```
##
## Call:
## glm(formula = observed ~ voc * year, family = binomial(), data = exp.m)
##
## Coefficients:
```

```
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     1.53466    0.83338   1.841   0.0656 .
## voc            -0.02224    0.01287  -1.728   0.0839 .
## year2006        0.35841    1.20786   0.297   0.7667
## year2007        0.55335    1.13559   0.487   0.6261
## voc:year2006 -0.01310    0.02002  -0.655   0.5127
## voc:year2007 -0.03151    0.01983  -1.589   0.1121
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 171.61  on 123  degrees of freedom
## Residual deviance: 139.60  on 118  degrees of freedom
## AIC: 151.6
##
## Number of Fisher Scoring iterations: 4
```

Think-Pair-Share: How can we interpret the coefficients in the above model?

The `year2006` and `year2007` terms provide estimates of differences in intercepts between each listed year and the reference level (2005 in this case) and the `voc:year2006` and `voc:year2007` terms provide estimates of differences in slopes between each listed year and the reference level year. If this is not clear, it would help to review Chapter 3. You might also consider trying to write out the design matrix for a few observations in the data set that come from different years. You could then check your understanding using the `model.matrix` function. Try writing out the design matrix for the following observations:

```
exp.m[c(1,37,53,74), c("voc", "year")]
```

```
##     voc year
## 37   20 2005
## 73   10 2007
## 89   60 2006
## 110  85 2007
```

Check your answer by typing:

```
model.matrix(mod3)[c(1,37,53,74),]
```

16.4 Evaluating assumptions and fit

We can use similar methods as seen in Section 15.4 to evaluate model fit.

16.4.1 Residual plots

As discussed in Section 15.4.1, there are multiple types of residuals (standard, Pearson, deviance) that we might consider using to evaluate model fit. Below, we use the `residualPlots` function in the `car` library to create plots of Pearson residuals versus model predictors and versus fitted values (Figure 16.5).

```
residualPlot(mod1)
```

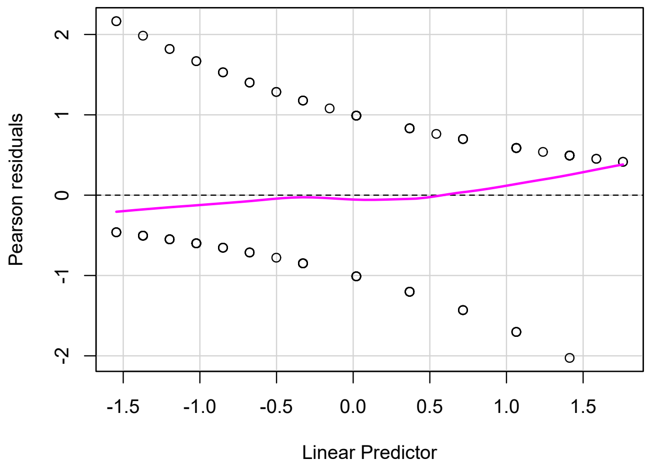


FIGURE 16.5 Residual plots for the logistic regression model fit to data from Moose in Minnesota.

The residuals for binary data *always* look weird since Y_i can only take on the values of 0 and 1. Thus, it is important to focus on the smooth line to see if there is any patterning. Alternatively, it can be beneficial to create binned residual plots by "dividing the data into categories (bins) based on their fitted values, and then plotting the average residual versus the average fitted value for each bin." (Gelman and Hill 2006). The `binned_residual` function in the `performance` package will do this for us (Figure 16.6). It also provides error bounds and colors binned residuals that fall outside these bounds to help easily visualize whether the model provides a reasonable fit to the data.

```
binplot<-binned_residuals(mod1)
plot(binplot)
```

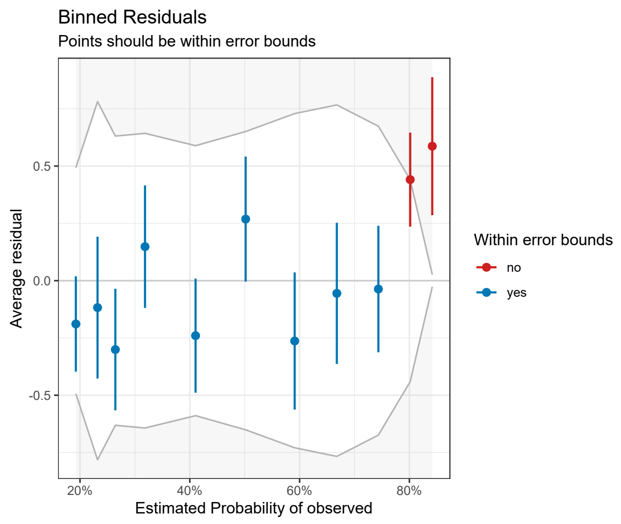


FIGURE 16.6 Binned residual plot for the logistic regression model containing only `voc`. The model looks OK, except that it is underpredicting those observations that have the highest probabilities of detection; these are the observations with the smallest values of `voc`.

16.4.2 Goodness-of-fit test

We can easily adapt the general approach for testing goodness-of-fit from Sections 13.4 and 15.4.5 to logistic regression models. Again, we will consider the sum of Pearson residuals as our goodness-of-fit statistic:

$$\chi^2_{n-p} = \sum_{i=1}^{n} \frac{(Y_i - E[Y_i|X_i])^2}{Var[Y_i|X_i]}$$

For binary data analyzed using logistic regression, we have:

- $E[Y_i|X_i] = p_i = \frac{exp(\beta_0+\beta_1 x_1+...\beta_k x_k)}{1+exp(\beta_0+\beta_1 x_1+...\beta_k x_k)}$
- $Var[Y_i|X_i] = p_i(1-p_i)$

Here, we demonstrate the approach using the simpler model containing only `voc`.

```
nsims<-10000
gfit.sim<-gfit.obs<-matrix(NA, nsims, 1)
nobs<-nrow(exp.m)
beta.hat<-mvrnorm(nsims, coef(mod1), vcov(mod1))
xmat<-model.matrix(mod1)
for(i in 1:nsims){
  # Form predictions using the randomly chosen betas
  ps<-plogis(xmat%*%beta.hat[i,])
  # Generate new y's
  new.y<-rbinom(nobs, size=1, prob=ps)
  # Calculate Pearson residuals
  gfit.sim[i,]<-sum((new.y-ps)^2/(ps*(1-ps)))
  gfit.obs[i,]<-sum((exp.m$observed-ps)^2/(ps*(1-ps)))
}
# GOF p-value
mean(gfit.sim > gfit.obs)
```

```
## [1] 0.4636
```

The p-value is much greater than 0.05, suggesting we do not have significant evidence to conclude that the logistic regression model with `voc` is inappropriate for our data.

16.4.3 Aside: alternative methods for goodness-of-fit testing

It is also common to see goodness-of-fit tests that ignore parameter uncertainty and that group observations to better meet the asymptotic χ^2 distributional assumption[1]. For example, observations may be grouped by deciles of their predicted values (similar to how the binned residual plot is constructed). The observed (O_i) and expected (E_i) number of successes and failures is then calculated for each group and compared to a χ^2 distribution:

$$\chi^2 = \sum_{i=1}^{n_g} \frac{(O_i - E_i)^2}{E_i} \sim \chi^2_{g-2}, \text{where}$$

$g =$ number of groups.

This approach is called the Hosmer-Lemeshow test (Hosmer Jr, Lemeshow, and Sturdivant 2013). There are many different implementations of this test in various R packages. To see a listing, we could use the `findFn` function from the `sos` package. If you run the code, below, you will see that there are over 40 packages that show up.

```
library(sos)
findFn("hosmer lemeshow")
```

One such implementation is in the `ResourceSelection` package (Subhash R. Lele, Keim, and Solymos 2019).

```
library(ResourceSelection)
hoslem.test(exp.m$observed, fitted(mod1), g = 8)
```

```

    Hosmer and Lemeshow goodness of fit (GOF) test

data:  exp.m$observed, fitted(mod1)
X-squared = 3.2505, df = 6, p-value = 0.7768
```

The `performance_hosmer` function in the `performance` package (Lüdecke et al. 2021) also implements this test.

[1]The distribution of the χ^2 goodness-of-fit test statistic will follow a χ^2 distribution when sample sizes are large and the null hypothesis is true.

```
performance_hosmer(mod1)
```

```
## # Hosmer-Lemeshow Goodness-of-Fit Test
##
##   Chi-squared: 6.674
##            df: 8
##       p-value: 0.572

## Summary: model seems to fit well.
```

The p-values differ in the two implementations due to using a different number of groups (8 versus 10), which highlights a potential limitation of this test. Note, if we change `g = 10` in the call to `hoslem.test`, we replicate the output from `performance_hosmer`.

16.5 Model comparisons

We can again use likelihood ratio tests to compare full and reduced models using either the `drop1` function or the `Anova` function in the `car` package (Fox and Weisberg 2019). Let's start with the model containing `voc` and `year` but not their interaction and consider whether we can simplify the model by dropping either `voc` or `year`.

```
drop1(mod2, test="Chisq")
```

```
Single term deletions

Model:
observed ~ voc + year
       Df Deviance    AIC     LRT  Pr(>Chi)
<none>      142.23 150.23
voc     1   168.20 174.20 25.9720 3.464e-07 ***
year    2   147.38 151.38  5.1558   0.07593 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The top row of this table reports the AIC for the full model containing both `voc` and `year`. The second and third rows report AICs for reduced models in which either `voc` (second row) or `year` (third row) are dropped. The last two columns report the test-statistic and p-value associated with the likelihood ratio test comparing full and reduced models. The small p-value for the `voc` row suggests we should prefer the `observed ~ voc + year` model relative to the `observed ~ year` model. This again confirms that `voc` is an important predictor of the probability of detection. The results are less clear when comparing the `observed ~ voc` and `observed ~ voc + year` models, with a p-value of 0.07.

We can get the same set of tests using the `Anova` function in the `car` package:

```
Anova(mod2)
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: observed
##      LR Chisq Df Pr(>Chisq)
## voc   25.9720  1  3.464e-07 ***
## year   5.1558  2    0.07593 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Lastly, we could compare nested or non-nested models using the AIC function. Here we compare:

- mod1 = `observed ~ voc`
- mod2 = `observed ~ voc + year`
- mod3 = `observed ~ voc * year`

```
AIC(mod1, mod2, mod3)
```

```
     df      AIC
mod1  2 151.3824
mod2  4 150.2266
mod3  6 151.6040
```

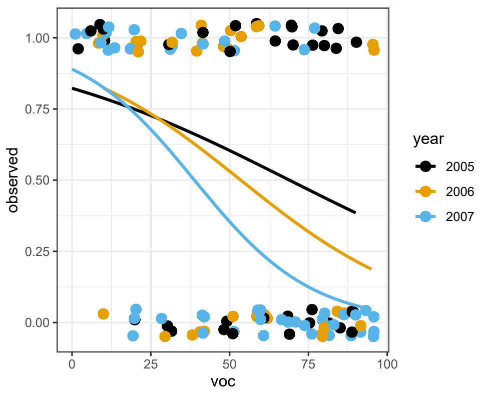
The model with `voc` and `year` but not their interaction has the lowest AIC.

16.6 Effect plots: Visualizing generalized linear models

It is easy to use `ggplot` to visualize models containing a single continuous variable as in Figure 16.4. We can also use `ggplot` to easily visualize regression models that include 1 categorical and 1 continuous predictor, along with their interaction – using a single plot or a multi-panel plot (e.g., Fig. 16.7).

```
p1<-ggplot(exp.m, aes(x=voc, y=observed, colour=year)) + theme_bw()+
   geom_point(position = position_jitter(w = 2, h = 0.05), size=3) +
   stat_smooth(method="glm", method.args = list(family = "binomial"),
               se=FALSE) +
  scale_colour_colorblind()

p2<-ggplot(exp.m, aes(voc,observed))+ theme_bw() +
    geom_point(position = position_jitter(w = 2, h = 0.05), size=3) +
    xlab("Visual Obstruction") +
    stat_smooth(method="glm", method.args = list(family = "binomial"))+
    ylab("Detection = 1") +facet_wrap("year")
(plot_spacer() + p1 + plot_spacer()) / p2
```

What if we fit more complicated models containing multiple predictor variables? We have a few options:



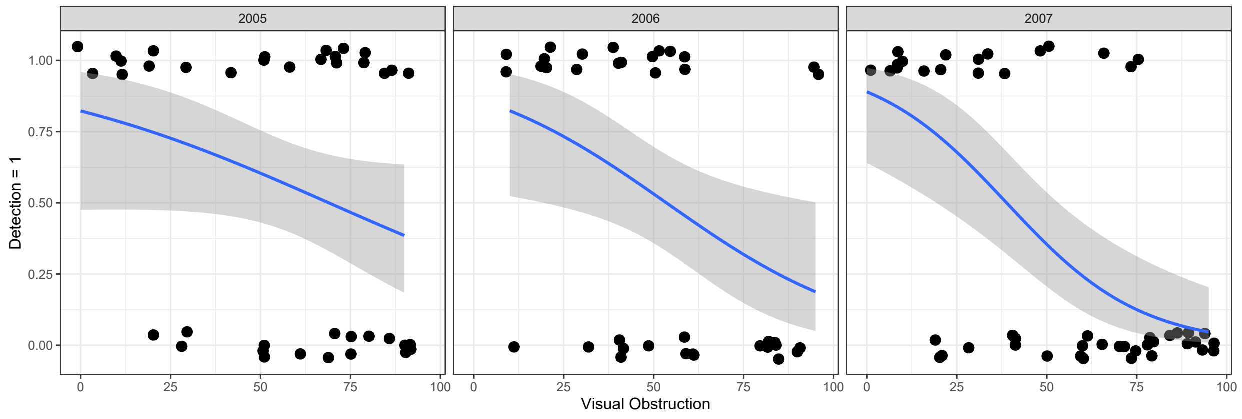


FIGURE 16.7 Fitted logistic regression model relating detection probabilities to the amount of visual obstruction, year, and their interaction.

a) We can create predicted values for different combinations of our explanatory variables, their standard errors, and then plot the results.
b) We can use one or more R packages to automate this process. In Section 16.6.4, we will explore the `effects` (Fox and Weisberg 2018) and `ggeffects` packages (Lüdecke 2018), which were also introduced in Section 3.14 when we discussed creating partial residual plots.

Let's begin by creating our own plots, which will give us insights into how these latter packages work.

16.6.1 Predictions and confidence intervals

Let's begin by considering our initial model containing only `voc`. We can summarize models in terms of: $\hat{E}[Y_i|X_i] = \hat{p}_i = \hat{P}(Y_i = 1|voc_i)$ for a range of `voc` values (Figure 16.8). The easiest way to accomplish this is to use the `predict` function in R. Specifically, we can generate predictions for a range of values of `voc` using:

```
voc.p<-seq(1, 99, length = 100)
p.hat <- predict(mod1, newdata = data.frame(voc = voc.p), type = "response")
```

```
plot(voc.p, p.hat, xlab = "VOC", ylab = "Pr(detect | voc)", type = "l")
```

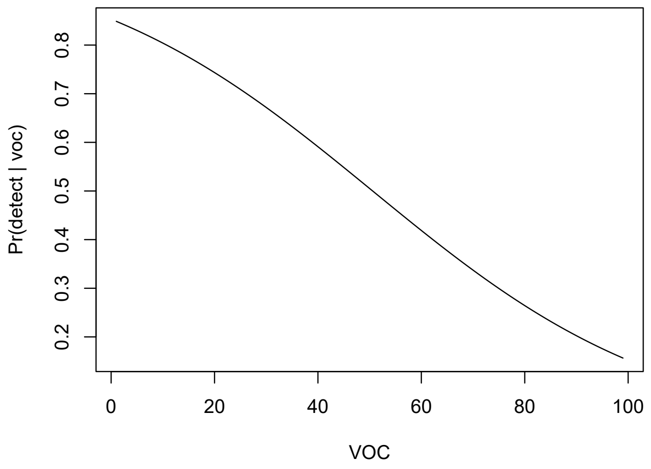


FIGURE 16.8 Predicted probability of detecting a moose as a function of visual obstruction (`voc`) only.

We could also ask the `predict` function to return standard errors associated with our predictions by adding the argument `se.fit=TRUE`. If we ask for SEs on the response scale, these standard errors will be calculated using the delta method (see Section 10.8.1). We could then use $\hat{p}_i \pm 1.96SE$ to form confidence intervals for our plot, depicting how p_i (and our uncertainty about p_i) depends on `voc`. Remember, that p_i has to be between 0 and 1, and there is no guarantee that these confidence intervals will remain ≤ 1 or ≥ 0. A better approach is to use `predict` to generate estimates on the link (i.e., logit) scale, along with standard errors on this scale. Using this information, we can then form a confidence interval for logit(p_i) and then back-transform this interval using the `plogis` function to get a confidence interval for p_i. This approach has several advantages:

1. The sampling distribution for logit$(\hat{p}_i) = \hat{\beta}_0 + \hat{\beta}_1 X_{1,i} + \ldots \hat{\beta}_p X_{p,i}$ tends to be more Normal than the sampling distribution for $\hat{p}_i$.
2. Confidence intervals will be guaranteed to live on the (0,1) scale. Also, note that the intervals will *not* be symmetric.

The code and plot below compares these two approaches (Figure 16.9):

```
# Predictions response scale
newdat <- data.frame(voc = seq(1, 99, length = 100))
phat <- predict(mod1, type = "resp", se.fit = T,newdata = newdat)
lcl.r <- phat$fit + 1.96 * phat$se.fit
ucl.r <- phat$fit - 1.96 * phat$se.fit

# Predictions link scale
phat2<-predict(mod1, type = "link", se.fit = T, newdata = newdat)
lcl.l <- plogis(phat2$fit + 1.96 * phat2$se.fit)
ucl.l <- plogis(phat2$fit - 1.96 * phat2$se.fit)
pe.l <- plogis(phat2$fit)

# Combine and plot
```

```
newdat <- cbind(newdat, phat = phat$fit, lcl.r, ucl.r, phat2 = phat2$fit, lcl.l, ucl.l, pe.l)
ggplot(newdat, aes(voc, phat)) + geom_line() +
  geom_line(aes(voc, lcl.l), lty = 2, col = "blue") +
  geom_line(aes(voc, ucl.l), lty = 2, col = "blue") +
  geom_line(aes(voc, lcl.r), lty = 2, col = "red") +
  geom_line(aes(voc, ucl.r), lty = 2, col = "red") +
  xlab("Visual Obstruction") + ylab(expression(hat(p)))
```

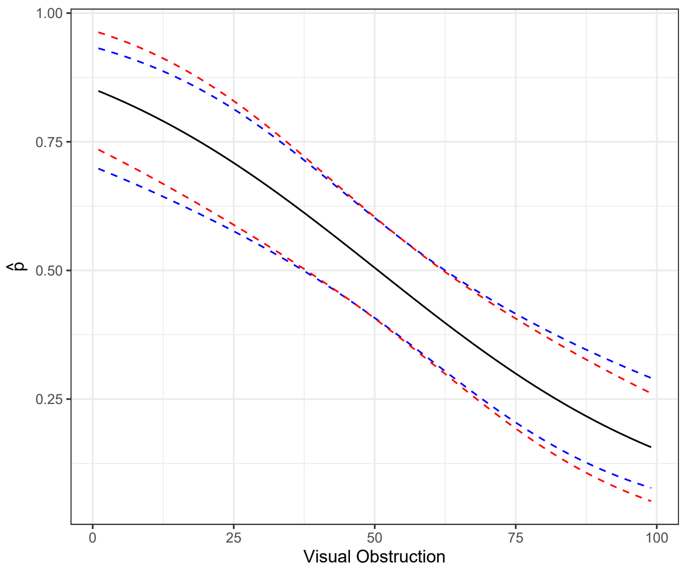


FIGURE 16.9 Comparison of methods for forming confidence intervals on the response scale in red versus link scale with back transformation in blue.

16.6.2 Aside: Revisiting predictions using matrix algebra

How does R and the `predict` function work? Here, we briefly revisit the ideas from Sections 5.6 and 3.12. Remember, `glm` estimates β using maximum likelihood, specifically, by maximizing:

$$L(\beta; y, x) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} \text{ with}$$

$$p_i = \frac{e^{\beta_0+\beta_1 X_{1,i}+\ldots\beta_p X_{p,i}}}{1+e^{\beta_0+\beta_1 X_{1,i}+\ldots\beta_p X_{p,i}}}$$

Also, remember, for large samples, $\hat{\beta} \sim N(\beta, I^{-1}(\beta))$. We can use this theory to conduct hypothesis tests (see the p-values using z-statistics in the output by the `summary` function) and to get confidence intervals.

Let X = our design matrix, in this case consisting of 2 columns (a column of 1's for the intercept and a second column containing the `voc` measurements). Again, we can use `model.matrix` to see what this design matrix looks like.

Let $\beta = (\beta_0, \beta_1)$, our vector of parameters (intercept and slope).

We can calculate $\text{logit}(\hat{p}_i|X_i)$ and its SE using matrix multiplication:

- $\text{logit}(\hat{p}_i|X_i) = X\hat{\beta}$ (use `%*%` in R to perform matrix multiplication)
- Variance/covariance of $\text{logit}(\hat{p}|X) = X(\hat{I}^{-1}(\beta))X^T$ (we access our estimate of $I^{-1}(\beta)$ using `vcov(mod1)`).

The code below demonstrates that we get the same results using matrix multiplication as when using the `predict` function.

```
newdata <- data.frame(observed = 1, voc = voc.p)
xmat <- model.matrix(mod1, data = newdata)
p.hat.check <- xmat %*% coef(mod1)
p.se.check <- sqrt(diag(xmat %*% vcov(mod1) %*% t(xmat)))
# Compare predictions
summary(predict(mod1, newdata = newdata, type = "link") - p.hat.check)
```

```
##        V1
##  Min.   :0
##  1st Qu.:0
##  Median :0
##  Mean   :0
##  3rd Qu.:0
##  Max.   :0
```

```
# Compare SEs
summary(predict(mod1, newdata = newdata, type = "link", se.fit = TRUE)$se.fit - p.se.check)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -5.551e-17  5.551e-17  5.551e-17  6.634e-17  1.110e-16  1.665e-16
```

16.6.3 Additive effects on logit scale $\neq$ additive effects on probability scale

Creating and interpreting effect plots is a little trickier than it might seem due to modeling the mean on the logit scale. Let's consider our second model:

$$Y_i|X_i \sim \text{Bernoulli}(p_i)$$
$$\text{logit}(p_i) = \beta_0 + \beta_1 voc_i + \beta_2 I(year = 2006)_i + \beta_3 I(year = 2007)_i$$

Here, the effects of `year` and `voc` are "additive" on the logit scale, and thus, differences in $\text{logit}(p_i)$ between years do not depend on `voc`. However, the effects of `voc` and `year` are multiplicative on the odds scale and even more complicated on the p scale (see e.g., the discussion related to parameter interpretation in Section 16.3). As a result, differences in p_i between years will depend on `voc` and the effect of `voc` on p will depend on the year (even though we do not have an interaction in the model!).

To see this in action, let's plot $\text{logit}(p_i)$ and p_i versus `voc` for the different years (Figure 16.10). Note that distances between the lines for the different years are constant when we look at $\text{logit}(p_i)$ but not when we plot p_i (the curves are closer together as p_i approaches 0 or 1). These differences result from having to back-transform from the logit to response scale.

```
  newdat<-expand.grid(voc=seq(0,100,5), year=unique(exp.m$year))
  newdat$p.hats<-predict(mod2, newdat=newdat, type="response")
  newdat$logit.p.hats<-predict(mod2, newdata=newdat, type="link")
```

```
plot.logitp<-ggplot(newdat, aes(voc, logit.p.hats, colour=year))+geom_line()+
  ylab("logit(p)")
plot.p<-ggplot(newdat, aes(voc, p.hats, colour=year))+geom_line()+ylab("p")
plot.logitp + plot.p
```

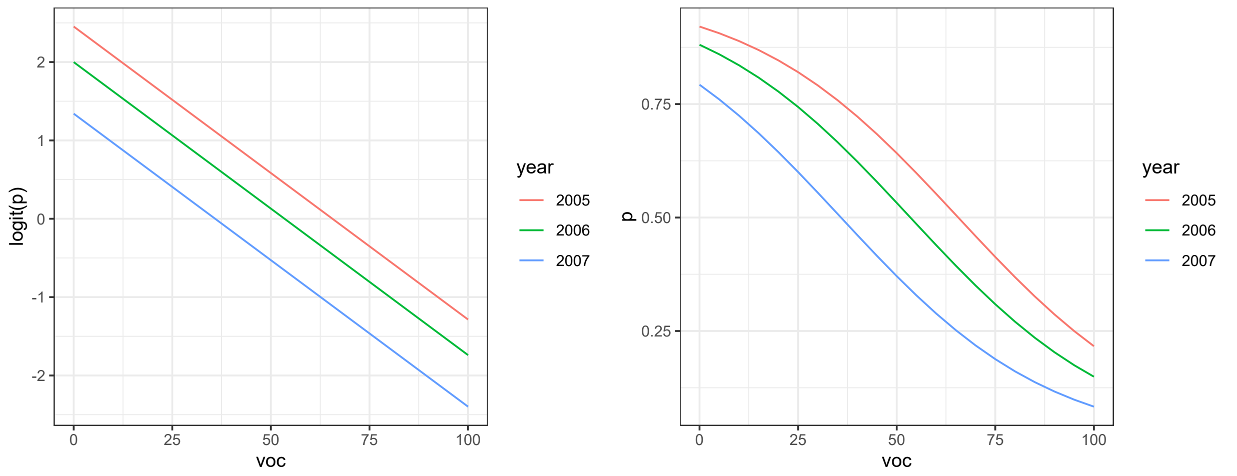


FIGURE 16.10 Relationships between the amount of visual obstruction (`voc`) and logit(p) (left panel) and p (right panel).

16.6.4 Effect plots for complex models: Using the effects and ggpredict packages

Recall from Section 3.14.3, we can visualize the effect of predictors using effect plots formed by varying 1 predictor while holding all other predictors at a common value (e.g., the mean for continuous variables and the modal value for categorical variables). The `expand.grid` function, used in the creation of Figure 16.10 can be a big help here as it makes it easy to create a data set containing all combinations of multiple predictor variables. Still, creating effect plots from scratch can be a little tedious. The `effects` and `ggeffects` packages can makes this process a little easier.

16.6.4.1 Effects package

The `effect` function in the `effects` package:

- Fixes all continuous variables (other than the one of interest) at their mean values
- For categorical predictors, it averages predictions on the link scale, weighted by the proportion of observations in each category

By default, plots are constructed on the link scale, but we can add `type="response"` to create a plot on response scale. Examples are given in Figures 16.11 and 16.12.

```
plot(effect("voc", mod2), type="response")
plot(effect("year", mod2), type="response")
```

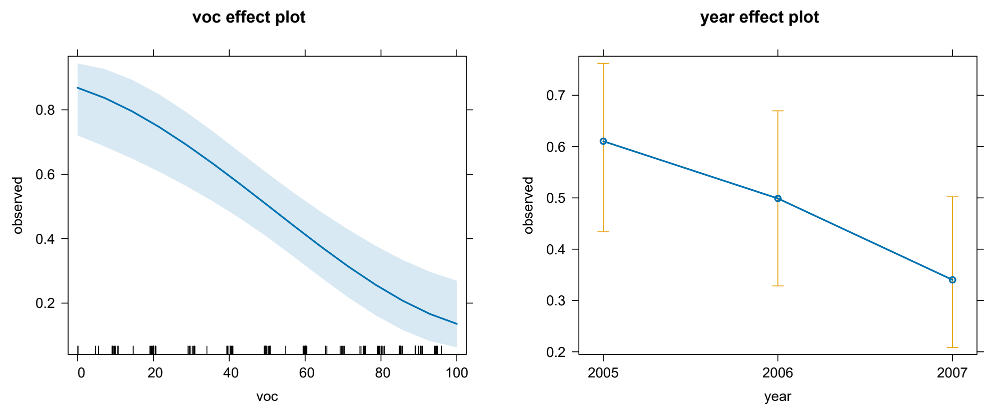


FIGURE 16.11 Effect plots showing how the probability of detection varies with `voc` and with `year` using the model with `voc` and `year`.

We can add partial residuals using adding `parital.residuals = TRUE` to the `effect` function (Figure 16.12). For binary data, it is difficult to interpret the raw residuals, but this option also overlays a smooth line capturing any trend.

```
plot(effect("voc", mod2, partial.residuals = TRUE), type="response")
```

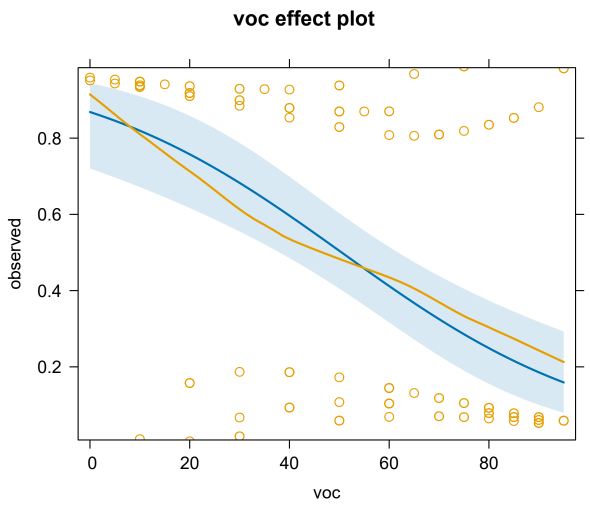


FIGURE 16.12 Effect plot with partial residuals added, showing how the probability of detection varies with `voc` using the model with `voc` and `year`.

We can also use `effect` and the `summary` functions to return numerical values (estimates and confidence intervals):

```
  summary(effect("year", mod2))
```

```
 year effect
year
     2005      2006      2007 
0.6105102 0.4988990 0.3401946 

 Lower 95 Percent Confidence Limits
year
     2005      2006      2007 
0.4338786 0.3283351 0.2086299 

 Upper 95 Percent Confidence Limits
year
     2005      2006      2007 
0.7622325 0.6697194 0.5020875 
```

Alternatively, we can use the `ggeffect` function in the `ggeffects` package to format the output as a list with an associated `print` function.

```
ggeffect(mod2)
```

```
## $voc
## # Predicted probabilities of observed
## 
## voc | Predicted |       95% CI
## ------------------------------
##   0 |      0.87 | [0.72, 0.94]
##  10 |      0.82 | [0.67, 0.91]
##  20 |      0.76 | [0.62, 0.86]
##  35 |      0.64 | [0.52, 0.74]
##  55 |      0.46 | [0.36, 0.56]
##  65 |      0.37 | [0.27, 0.47]
##  75 |      0.29 | [0.19, 0.41]
##  95 |      0.16 | [0.08, 0.29]

## 
## Not all rows are shown in the output. Use ‘print(..., n = Inf)‘ to show
##   all rows.

## 
## $year
## # Predicted probabilities of observed
## 
## year | Predicted |       95% CI
## -------------------------------
## 2005 |      0.61 | [0.43, 0.76]
## 2006 |      0.50 | [0.33, 0.67]
## 2007 |      0.34 | [0.21, 0.50]
## 
## attr(,"class")
## [1] "ggalleffects" "list"        
## attr(,"model.name")
## [1] "mod2"
```

There is also a `plot` function that we can use with the `ggeffect` function to visualize these effects (Figure 16.13).

```
p1 <- plot(ggeffect(mod2, "year"))
p2 <- plot(ggeffect(mod2, "voc"))
p1 + p2
```

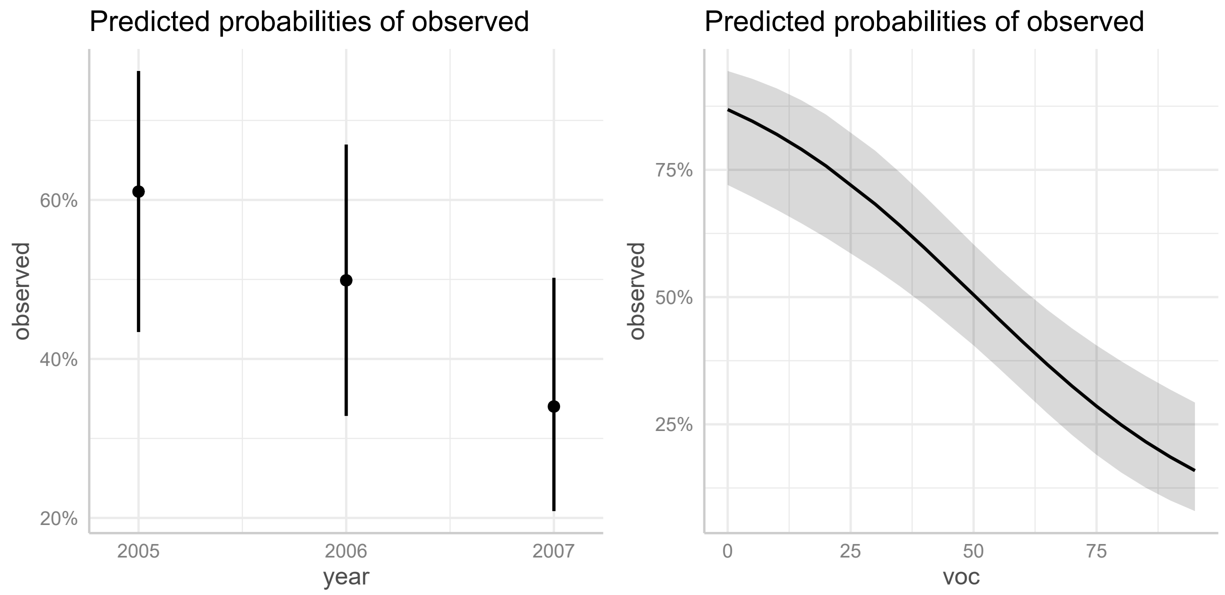


FIGURE 16.13 Effect plot for the model with `voc` and `year` created using the `ggeffects` package.

And again, we could add partial residuals and a smooth line to our plot (Figure 16.14).

```
plot(ggeffect(mod2, "voc"), residuals = TRUE, residuals.line = TRUE)
```

```
## Data points may overlap. Use the ‘jitter‘ argument to add some amount of
##   random variation to the location of data points and avoid overplotting.
```

```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```

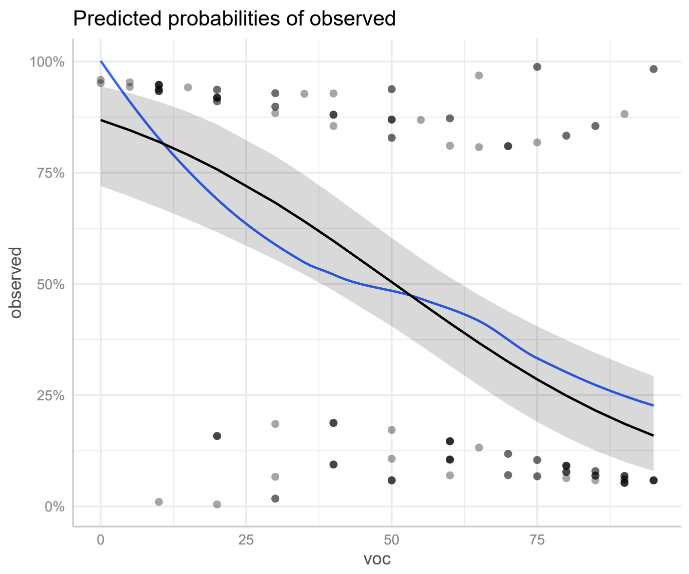


FIGURE 16.14 Effect plot created using the `ggeffects` package with partial residuals overlayed.

Lastly, we can specify specific terms for which we want estimates. In this case, `ggeffects` will return a data.frame from which we can create our own plot using `ggplot` (Figure 16.15).

```
effects.mod2 <- ggeffect(mod2, terms = c("voc", "year"))
ggplot(effects.mod2, aes(x, predicted, col = group)) + geom_line()
```

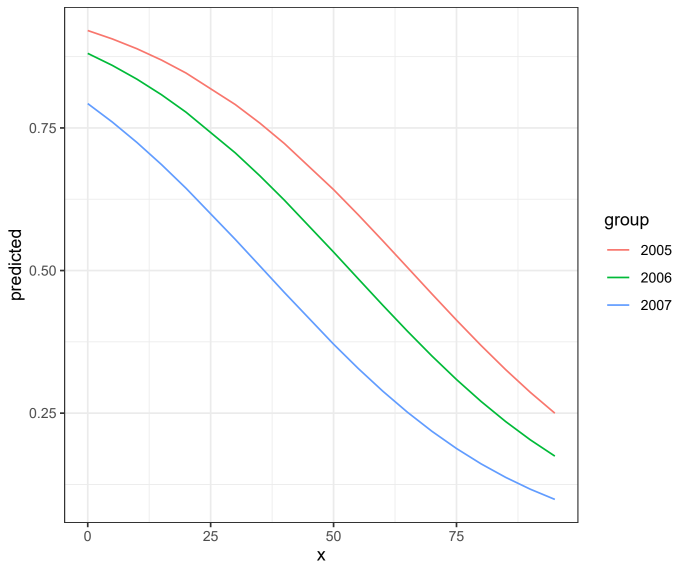


FIGURE 16.15 Effect plot created using the `ggeffects` package along with `ggplot`.

16.6.5 Understanding effect estimates from the effects and ggeffects packages

The goal of this section is to illuminate how the estimates and confidence intervals on the response scale are formed when using the `effects` package or the `ggeffects` function in the `ggeffects` package. Let's begin by considering the effects for `year`. These are formed by setting `voc` to its mean value, and estimating:

$P(Y_i = 1|year = year_i, voc = \overline{voc})$ for each of the years in the data set. These estimates are easy to understand and recreate using the `predict` function:

```
newdata<-data.frame(voc = rep(mean(exp.m$voc), 3),
                    year = c("2005", "2006", "2007"))
predict(mod2, newdata = newdata, type = "resp")
```

```
        1         2         3 
0.6105102 0.4988990 0.3401946 
```

```
effect("year", mod2, type = "response")
```

```
 year effect
year
     2005      2006      2007 
0.6105102 0.4988990 0.3401946 
```

Now, how does the `effects` package estimate the effect of `voc`? There is no "mean or average" year since `year` is a categorical variable. Instead, it calculates a weighted mean of the predictions for each year on the link scale, with weights given by the proportion of observations in each category. It then back-transforms this weighted mean.

```
effect("voc", mod2)
```

```
 voc effect
voc
        0        20        50        70       100 
0.8684553 0.7575962 0.5044524 0.3251918 0.1356677 
```

```
# Proportion of observations in each year (will be used to form weights)
weights <- data.frame(table(exp.m$year) / nrow(exp.m))
names(weights) = c("year", "weight")
weights
```

```
  year    weight
1 2005 0.3145161
2 2006 0.2983871
3 2007 0.3870968
```

```
# New data for predictions
newdata <- data.frame(expand.grid(year = c("2005", "2006", "2007"),
  voc = seq(0, 100, 20)))
newdata <- left_join(newdata, weights)

# Predictions on link scale
newdata$logit.phat <- predict(mod2, newdata = newdata, type = "link")

# Weight predictions and backtransform
newdata %>% group_by(voc) %>% dplyr::summarize(plogis(sum(logit.phat * weight)))
```

```
# A tibble: 6 x 2
    voc `plogis(sum(logit.phat * weight))`
  <dbl>                              <dbl>
1     0                              0.868
2    20                              0.758
3    40                              0.597
4    60                              0.412
5    80                              0.249
6   100                              0.136
```

16.6.6 Predictions using the ggpredict function

Whereas the `effects` function averages predictions across the different levels of a categorical predictor, producing what are sometimes referred to as *marginal effects*, the `ggpredict` function will provide *adjusted predictions* where all variables except a focal variable remain fixed at their mean, modal, or user-specified values. We first demonstrate this approach for our additive model with `voc` and `year`.

```
ggpredict(mod2)
```

```
## $voc
## # Predicted probabilities of observed
##
## voc | Predicted |       95% CI
## ------------------------------
##   0 |      0.92 | [0.77, 0.98]
##  10 |      0.89 | [0.73, 0.96]
##  20 |      0.85 | [0.67, 0.94]
##  35 |      0.76 | [0.58, 0.88]
##  55 |      0.60 | [0.42, 0.75]
##  65 |      0.51 | [0.34, 0.67]
##  75 |      0.41 | [0.25, 0.59]
##  95 |      0.25 | [0.12, 0.45]
##
## Adjusted for:
## * year = 2005

##
## Not all rows are shown in the output. Use `print(..., n = Inf)` to show
##   all rows.
```

```
##
## $year
## # Predicted probabilities of observed
##
## year | Predicted |       95% CI
## -------------------------------
## 2005 |      0.55 | [0.38, 0.71]
## 2006 |      0.44 | [0.28, 0.62]
## 2007 |      0.29 | [0.17, 0.45]
##
## Adjusted for:
## * voc = 60.00
##
## attr(,"class")
## [1] "ggalleffects" "list"
## attr(,"model.name")
## [1] "mod2"
```

As with the `ggeffects` function, if we specify terms from the model, the function will return a data.frame that can be easily plotted. This approach can be particularly useful when there are interactions between variables as in our `mod3` (Figure 16.16).

```
adjustpredict <- ggpredict(mod3, terms = c("voc", "year"))
plot(adjustpredict, grid = TRUE)
```

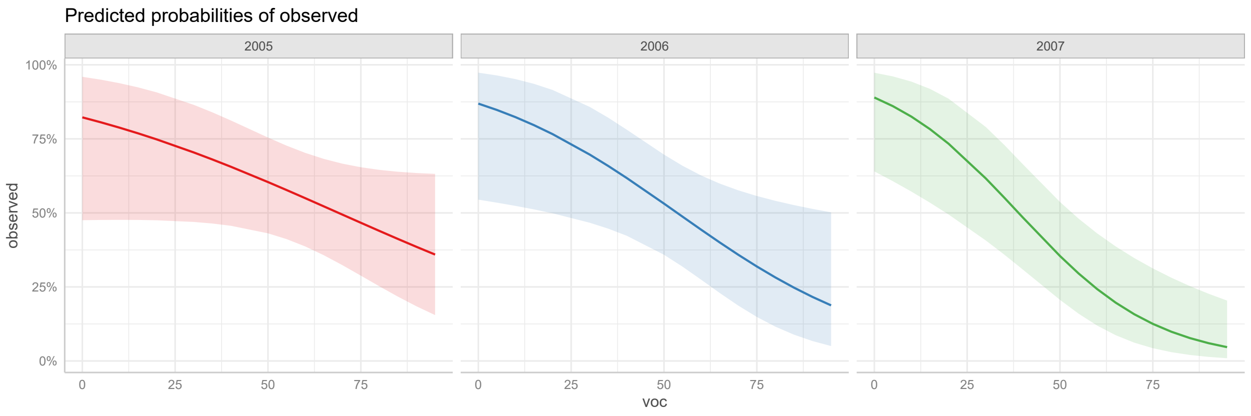


FIGURE 16.16 Adjusted prediction plot for the interaction model created using the ‘ggeffects‘ package.

16.7 Logistic regression: Bayesian implementations

Again, an advantage of implementing models in JAGS is that we will be forced to be very clear about the model we are fitting. Specifically, we will have to write down the likelihood with the expression relating logit(p) to our predictor variables. Contrast this process with fitting a model using `glm` where users may not even know that the mean is being modeled on a transformed scale.

In this section, we will fit the logistic regression model containing only `voc`, leaving the model with `year` and `voc` as an in-class exercise. Before we look at JAGS code for fitting this model, it is helpful to give some thought to priors for our regression parameters. It turns out that specifying priors can be a bit tricky. We can specify priors that are vague (meaning they take on a wide range of values, all equally likely) on the logit scale, but this may then imply something very specific when viewed on the p scale. Consider, for example, the model below, where p is constant and we specify our prior on the logit scale:

$$\begin{gathered} Y_i \sim \text{Bernoulli}(p) \\ \text{logit}(p) = \beta_0 \\ \beta_0 \sim N(0, 10^2) \end{gathered}$$

It turns out (see Fig. 16.17) that this prior is not at all vague when viewed on the p scale. In particular, this prior suggests that p is extremely likely to be very near 0 or 1 and unlikely to take on any value in between these two extremes.

```
par(mfrow=c(1,2))
beta0<-rnorm(10000,0,100)
p<-plogis(beta0)
hist(beta0, xlab=expression(beta[0]),
     main=expression(paste("Prior distribution for ", beta[0])),
     col="gray")
hist(p, xlab="p", main="Prior distribution for p", col="gray")
```

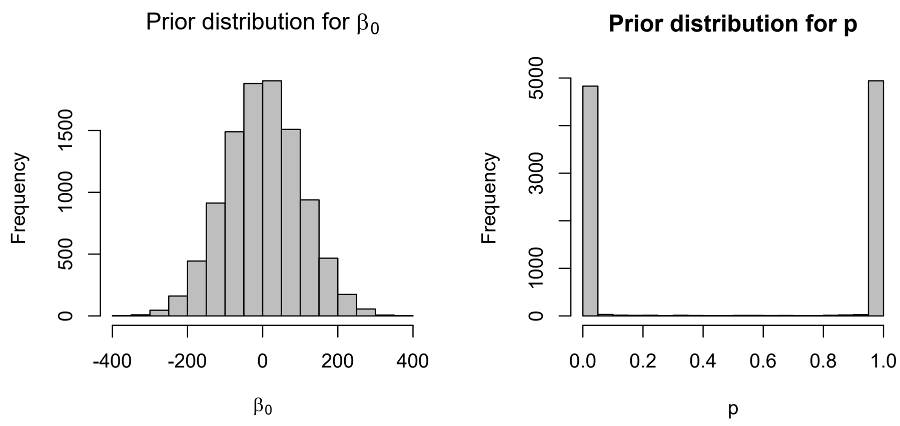


FIGURE 16.17 A N(0,100) prior distribution for logit(p) results in a non-vague prior on the p scale.

If the goal is to achieve a near uniform distribution for p, it turns out that a $N(0, \sqrt{3})$ distribution is a much better option (Fig. 16.18):

```
par(mfrow=c(1,2))
beta0<-rnorm(10000, 0, sqrt(3))
p<-plogis(beta0)
hist(beta0, xlab=expression(beta[0]),
     main=expression(paste("Prior distribution for ", beta[0])),
     col="gray")
hist(p, xlab="p", main="Prior distribution for p", col="gray")
```

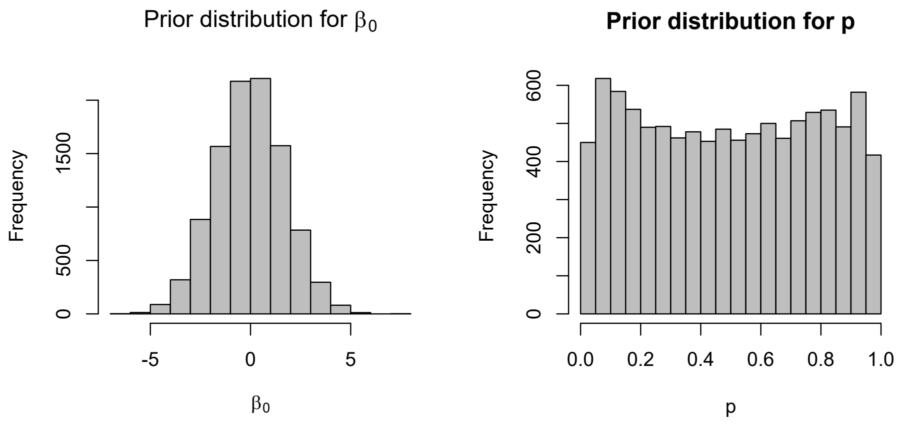


FIGURE 16.18 A $N(0, \sqrt{3})$ distribution for the prior on the logit scale, resulting in a near uniform prior distribution on the p scale.

Establishing recommendations for default priors is a high priority and active research area within the Bayesian community (see discussion here: https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations). For logistic regression models, Gelman et al. (2008) recommended:

1. Scaling continuous predictors so they have mean 0 and sd = 0.5.
2. Using a Cauchy prior, with precision parameter $= \frac{1}{2.5^2} = 0.16$; this distribution is equivalent to a Student-t distribution with 1 degree of freedom and can be specified in JAGS as: `dt(0, pow(2.5,-2), 1)`.

Both t- and Normal distributions assume most effect sizes are near 0, but the t-distribution allows for the possibility of more extreme values since a t-distribution has wider tails than a Normal distribution (Figure 16.19)[2].

```
library(LaplacesDemon)
curve(dnorm(x, mean=0, sd=2.5), from=-8, to=8, ylab="Density")
curve(dstp(x, mu=0, tau=1/2.5^2, n=1), from=-8, to=8, add=TRUE, lty=2)
legend(2, 0.15, c("Normal", "t"), lty=c(1,2), bty="n")
```

In general, we may seek a non-informative prior, which means that we want our answers to be determined primarily by the data and likelihood and *not* the prior. On the other hand, an advantage of a Bayesian approach is that we can potentially take advantage of previous knowledge and data when this information exists and we want it to influence our results. I.e., there may be times when using an *informative* prior can be beneficial. Furthermore, many would argue for weakly informative priors that "regularize" or shrink parameters towards 0 (e.g., McElreath 2020). This approach is often used to improve predictive performance, particularly in cases where one is faced with having too many explanatory variables relative to one's sample size (Section 8.7).

[2]Here we use the `dstp` function in the `LaplacesDemon` package (Statisticat and LLC. 2021) since it is parameterized in the same way as the dt function in JAGS

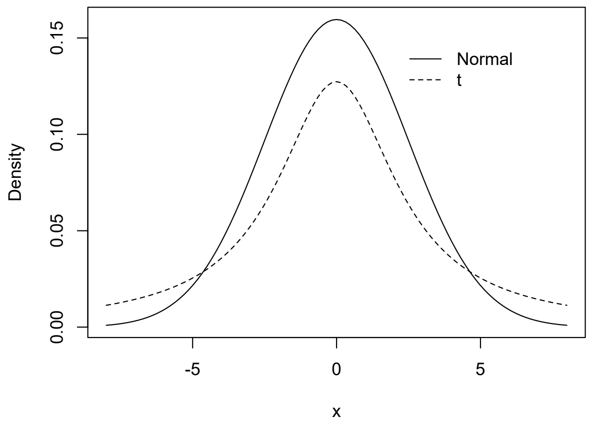


FIGURE 16.19 Comparison of Normal(0, 2.5^2) (solid curve) and Student-t priors (dashed curve) for logistic regression models.

16.7.1 Fitting the Bayesian logistic regression model to moose sightability data

Below, we use Gelman's suggested prior when analyzing the moose data in JAGS:

```
lrmod<-function(){

  # Priors
  # Note: Gelman recommends
  # - scaling continuous predictors so they have mean 0 and sd = 0.5
  # - using a non-informative Cauchy prior dt(0, pow(2.5,-2), 1)
  # see arxiv.org/pdf/0901.4011.pdf
  alpha ~ dt(0, pow(2.5, -2), 1)
  beta ~ dt(0, pow(2.5, -2), 1)

  # Likelihood
  for(i in 1:n){
    logitp[i] <- alpha + beta * voc[i]
    p[i] <- exp(logitp[i]) / (1 + exp(logitp[i]))
    observed[i] ~ dbin(p[i], 1)

    # GOF test
    presi[i] <- (observed[i] - p[i]) / sqrt(p[i] * (1 - p[i]))
    obs.new[i] ~ dbin(p[i], 1)
    presi.new[i] <- (obs.new[i] - p[i]) / sqrt(p[i] * (1 - p[i]))
    D[i] <- pow(presi[i], 2)
    D.new[i] <- pow(presi.new[i], 2)
  }
  fit <- sum(D[])
  fit.new <- sum(D.new[])
}
```

```
# Bundle data
voc <- (exp.m$voc - mean(exp.m$voc)) / (0.5 * sd(exp.m$voc))
jagsdata <- list(observed = exp.m$observed, voc = voc, n = nrow(exp.m))


# Parameters to estimate
params <- c("alpha", "beta", "p", "presi", "fit", "fit.new")

# MCMC settings
nc <- 3
ni <- 3000
nb <- 1000
nt <- 2

out.p <- jags.parallel(data = jagsdata, parameters.to.save = params,
                       model.file = lrmod, n.thin= 2, n.chains = 3,
                       n.burnin = 1000, n.iter = 3000)

#Goodness-of-fit test
fitstats <- MCMCpstr(out.p, params = c("fit", "fit.new"), type = "chains")
T.extreme <- fitstats$fit.new >= fitstats$fit
(p.val <- mean(T.extreme))
```

```
## [1] 0.4643333
```

Looking at the goodness-of-fit test (above), we again fail to reject the null hypothesis that our data are consisten with the assumed model. Below, we compare estimates of coefficients and 95% credible intervals to those obtained usin `glm` with scaled `voc`:

```
MCMCsummary(out.p, params = c("alpha", "beta"), round = 3)
```

```
##         mean    sd   2.5%    50%  97.5% Rhat n.eff
## alpha -0.101 0.197 -0.494 -0.099  0.277    1  2730
## beta  -0.497 0.112 -0.738 -0.494 -0.280    1  2759
```

```
exp.m$voc.scaled.gelman <- voc # scaled version
mod1b <- glm(exp.m$observed ~ voc.scaled.gelman, family = binomial(),
      data = exp.m)
cbind(coef(mod1b), confint(mod1b))
```

```
## Waiting for profiling to be done...
```

```
##                                   2.5 %     97.5 %
## (Intercept)       -0.1045001 -0.4980208  0.2864257
## voc.scaled.gelman -0.4870714 -0.7114791 -0.2835381
```

What if we had naively used extremely vague priors for our regression parameters? We can compare the results using the `MCMCplot` function in the `MCMCvis` package (Figure 16.20).

```
lrmodv<-function(){

  alpha~dnorm(0, 0.0001)
  beta~dnorm(0, 0.0001)

  # Likelihood
  for(i in 1:n){
    logitp[i]<-alpha+beta*voc[i]
    p[i]<-exp(logitp[i])/(1+exp(logitp[i]))
    observed[i]~dbin(p[i],1)
  }
}

params <- c("alpha", "beta")
out.p.vague <- jags.parallel(data = jagsdata, parameters.to.save = params,
                             model.file = lrmodv, n.thin= 2, n.chains = 3,
                             n.burnin = 1000, n.iter = 3000)
```

```
MCMCplot(object = out.p, object2=out.p.vague, params=c("alpha", "beta"),
         offset=0.1, main='Posterior Distributions')
```

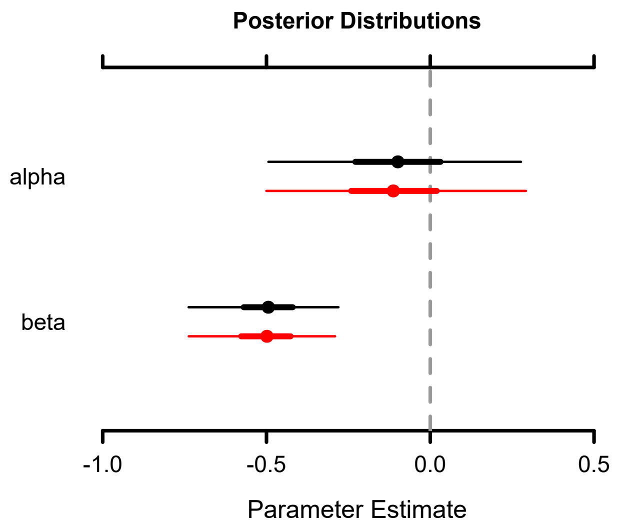


FIGURE 16.20 Comparison of logistic regression models fit using JAGS with different prior distributions for the regression parameters. Plot constructed with the `MCMCplot` function in the `MCMCvis` package.

In this case, the priors make little difference to our end results, but that may not always be the case.

16.8 Aside: Logistic regression with multiple trials

If we have multiple observations for each unique set of predictor variables, then we can write the model as:

$$Y_i|X_i \sim Binomial(n_i, p_i)$$

$$logit(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 X_{1,i} + \dots \beta_k X_{k,i}$$

This formulation can provide significant increases in speed when fitting large data sets (see e.g., Iannarilli et al. 2019). The `glm` function allows us to fit a logistic regression model to grouped data if we specify the number of trials and the number of successes. We will demonstrate the approach using a famous data set from Bliss (1935), which contains the number of adult flour beetles killed after 5 hours of exposure to gaseous carbon disulfide at various concentrations. The data are contained in a number of different R packages, including the `glmx` package. We access the data using:

```
library(glmx)
data("BeetleMortality")
BeetleMortality
```

```
##     dose died  n
## 1 1.6907    6 59
## 2 1.7242   13 60
## 3 1.7552   18 62
## 4 1.7842   28 56
## 5 1.8113   52 63
## 6 1.8369   53 59
## 7 1.8610   61 62
## 8 1.8839   60 60
```

Let Y_i be the number of individuals that died and n_i be the number of beetles exposed to each dosage level. We will assume:

$$Y_i|Dosage_i \sim Binomial(n_i, p_i)$$

$$logit(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 Dosage_i$$

We can fit the model using:

```
lrdose <- glm(cbind(died, n - died) ~ dose, data = BeetleMortality, family = binomial())
summary(lrdose)
```

```
##
## Call:
## glm(formula = cbind(died, n - died) ~ dose, family = binomial(),
##     data = BeetleMortality)
```

```
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -60.717      5.181  -11.72   <2e-16 ***
## dose          34.270      2.912   11.77   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 284.202  on 7  degrees of freedom
## Residual deviance:  11.232  on 6  degrees of freedom
## AIC: 41.43
##
## Number of Fisher Scoring iterations: 4
```

To demonstrate that we get the same result if we fit a model to data containing a separate record for each observation, we create a "long format" data set below:

```
BeetleLong <- NULL
uid <- unique(BeetleMortality$dose)
for(i in seq_along(uid)){
  tempdat <- BeetleMortality[i,]
  BeetleLong <- rbind(BeetleLong,
                      data.frame(died = c(rep(1, tempdat$died), rep(0, tempdat$n- tempdat$died)),
                                 dose = rep(tempdat$dose, tempdat$n)))
}
head(BeetleLong)
```

```
##   died   dose
## 1    1 1.6907
## 2    1 1.6907
## 3    1 1.6907
## 4    1 1.6907
## 5    1 1.6907
## 6    1 1.6907
```

```
#Show we get the same group summaries as the original data set
BeetleLong %>% group_by(dose) %>%
  dplyr::summarize(died = sum(died), trials = n())
```

```
## # A tibble: 8 x 3
##    dose  died trials
##   <dbl> <dbl>  <int>
## 1  1.69     6     59
## 2  1.72    13     60
## 3  1.76    18     62
## 4  1.78    28     56
## 5  1.81    52     63
## 6  1.84    53     59
## 7  1.86    61     62
## 8  1.88    60     60
```

We then fit the same regression model to the set of Bernoulli trials showing that we get the same result:

```
lrdose2 <- glm(died ~ dose, data = BeetleLong, family = binomial())
summary(lrdose2)
```

```
##
## Call:
## glm(formula = died ~ dose, family = binomial(), data = BeetleLong)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept)  -60.717      5.181  -11.72   <2e-16 ***
## dose          34.270      2.912   11.77   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 645.44  on 480  degrees of freedom
## Residual deviance: 372.47  on 479  degrees of freedom
## AIC: 376.47
##
## Number of Fisher Scoring iterations: 5
```

16.9 Aside: Complete separation

Occasionally, when fitting a logistic regression model, you may encounter a warning message that the `glm` fitting algorithm "did not converge" or that "fitted probabilities numerically 0 or 1 occurred." This is indicative of a problem referred to as complete separation – in which a predictor variable or set of predictor variables is able to fully discriminate between the 0's and 1's.

To demonstrate, we will use an example developed by Jack Weiss using data from Piegorsch and Bailer (2005) collected to evaluate water quality and potential contamination by an industrial solvent trichloroethylene (TCE). The data are included in the `Data4Ecologists` package and can be accessed using:

```
library(Data4Ecologists)
data(wells)
```

We will build a logistic regression model to describe the probability that a well will be contaminated with TCE as a function of the surrounding land use category (`land.use`) and whether or not sewers were used in the area surrounding the well (`sewer`). Each row in the data set contains a unique set of covariates, with `y` quantifying the number of wells contaminated with TSE and `n` giving the total number of wells with the same `land.use` and `sewer` covariates. Thus, we model the data using `glm` with a `cbind(successes, failures)` syntax as described in Section 16.8:

```
lr1 <- glm(cbind(y, n - y) ~ land.use + sewer, data = wells, family = binomial())
summary(lr1)
```

```
##
## Call:
## glm(formula = cbind(y, n - y) ~ land.use + sewer, family = binomial(),
##     data = wells)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -3.6568     0.7376  -4.957 7.14e-07 ***
## land.usecomm     1.8676     0.8044   2.322  0.02025 *
## land.useindus    2.2070     0.8053   2.741  0.00613 **
## land.useinst     0.7584     0.8395   0.903  0.36629
## land.userecr     0.6676     0.8435   0.791  0.42870
## land.useresH     1.7316     0.7784   2.225  0.02611 *
## land.useresL     0.6663     1.0501   0.635  0.52572
## land.useresM     1.0212     0.7809   1.308  0.19099
## land.usetrans    0.7933     0.8360   0.949  0.34267
## land.useundev  -18.3414  3033.9308  -0.006  0.99518
## seweryes         1.5980     0.2955   5.407 6.41e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 146.956  on 19  degrees of freedom
## Residual deviance:  15.201  on  9  degrees of freedom
## AIC: 82.476
##
## Number of Fisher Scoring iterations: 17
```

Although we didn't receive a warning in this case, we see the symptoms of complete separation if we inspect the model coefficients and their SEs; namely the coefficient for `land.useundev` is extremely large in absolute value (-18.34) and its SE is also large (3034).

Think-Pair-Share: Estimate the probability that a well placed in undeveloped land without sewers has TSE contamination. What do you think this indicates about the underlying data?

We can estimate the probability that a well placed in undeveloped land has TSE contamination by adding the intercept and the coefficient for `land.useundev` and then taking the inverse logit transformation:

```
betas.lr1 <- coef(lr1)
plogis(betas.lr1[1] + betas.lr1[length(betas.lr1)-1])
```

```
##  (Intercept)
## 2.794265e-10
```

We see that this probability is extremely small. Let's look at the observations broken down by land use category:

```
wells %>% mutate(nocontam = n-y) %>% group_by(land.use) %>%
  dplyr::summarize(contam = sum(y), noncontam = sum(nocontam))
```

```
## # A tibble: 10 x 3
##    land.use contam noncontam
##    <chr>     <int>     <int>
##  1 agri          2        53
##  2 comm         20        29
##  3 indus        20        25
##  4 inst          8        46
##  5 recr          7        57
##  6 resH         34        59
##  7 resL          2        24
##  8 resM         17       112
##  9 trans         8        51
## 10 undev         0        76
```

We see that there are no wells in undeveloped areas that were contaminated. Thus, the smaller the coefficient for `land.useundev` the better – essentially, the numerical optimizer keeps moving towards $-\infty$ but at some point the likelihood becomes extremely flat (e.g., consider that our current estimate already equates to a probability on the order fo 1^{-10}). Also, recall that flat likelihoods result in Hessians that are near 0, and as a result, large SEs (see Section 10.6).

There are several different approaches that we might take to deal with this problem. One simple solution would be to group land use categories so that we no longer have complete separation. Another popular approach is to use penalized likelihood or Bayesian approaches (Firth 1993; Fijorek and Sokolowski 2012). Let's consider a Bayesian model with $N(\mu = 0, \sigma^2 = 3)$ priors:

```
lrmod<-function(){

  # Priors
  for(i in 1:nlanduse){
    beta.landuse[i] ~ dnorm(0, 1/3)
  }
  beta.sewer ~ dnorm(0, 1/3)
  # Likelihood
  for(i in 1:nobs){
    logit(p[i]) <-beta.landuse[landuse[i]] + beta.sewer * sewer[i]
    y[i]~dbin(p[i], n[i])
  }
}



# Bundle data
jags.data <- list(y = wells$y,
                  landuse = as.numeric(as.factor(wells$land.use)),
                  sewer = as.numeric(as.factor(wells$sewer)),
                  n = wells$n, nobs = nrow(wells),
                  nlanduse = length(unique(wells$land.use)))



# Parameters to estimate
params <- c("beta.landuse", "beta.sewer")

# MCMC settings
```

```
nc <- 3
ni <- 3000
nb <- 1000
nt <- 2

lr.bayes <- jags.parallel(data = jags.data, parameters.to.save = params,
                        model.file = lrmod, n.thin = 1, n.chains = 3,
                        n.burnin = 1000, n.iter = 5000)
```

Let's look to make sure everything converged and inspect the SE's of the different coefficients:

```
MCMCvis::MCMCsummary(lr.bayes)
```

```
##                       mean         sd       2.5%        50%       97.5% Rhat
## beta.landuse[1]  -3.587814  0.6073940 -4.8247411 -3.5604594  -2.4832725 1.02
## beta.landuse[2]  -1.463457  0.4863417 -2.4085845 -1.4653444  -0.5082888 1.00
## beta.landuse[3]  -1.231906  0.4713098 -2.1471992 -1.2345322  -0.3064563 1.00
## beta.landuse[4]  -2.656346  0.4941036 -3.6411895 -2.6549084  -1.6979328 1.01
## beta.landuse[5]  -2.891551  0.4991950 -3.9187910 -2.8830222  -1.9718231 1.01
## beta.landuse[6]  -1.637646  0.4333541 -2.4760085 -1.6414424  -0.7911272 1.00
## beta.landuse[7]  -2.936314  0.6705834 -4.3194755 -2.8966989  -1.7119706 1.01
## beta.landuse[8]  -2.688568  0.3981482 -3.4558552 -2.6962779  -1.9153455 1.01
## beta.landuse[9]  -2.672110  0.4940992 -3.6768105 -2.6688090  -1.7112487 1.00
## beta.landuse[10] -4.726058  0.7601821 -6.3112348 -4.6971673  -3.3747042 1.02
## beta.sewer        0.599020  0.2092529  0.1864398  0.6002753   0.9993316 1.00
## deviance         87.062768 10.8333724 73.9455898 86.1707714 103.4192716 1.00
##                  n.eff
## beta.landuse[1]    342
## beta.landuse[2]   1348
## beta.landuse[3]   1547
## beta.landuse[4]    791
## beta.landuse[5]    582
## beta.landuse[6]   1486
## beta.landuse[7]    561
## beta.landuse[8]    800
## beta.landuse[9]    783
## beta.landuse[10]   232
## beta.sewer        1244
## deviance          1051
```

The last landuse coefficient is the one for undeveloped. We see that the N(0, 3) prior helped to stabilize the posterior distribution for this coefficient. We can estimate the probability of a well in an undeveloped area without sewers being contaminated along with a 95% credible interval for this probability using:

```
# Estimate
MCMCvis::MCMCpstr(lr.bayes, params = "beta.landuse[10]", ISB = FALSE,
                  func = function(x){mean(plogis(x))})
```

```
## $`beta.landuse[10]`
## [1] 0.01163286
```

```
# 95% CI
MCMCvis::MCMCpstr(lr.bayes, params = "beta.landuse[10]", ISB = FALSE,
                  func = function(x){quantile(plogis(x), probs = c(0.025, 0.975))})
```

```
## $`beta.landuse[10]`
##                          2.5%      97.5%
## beta.landuse[10] 0.001812499 0.03309544
```

The estimate is still very small, but non-zero. And, we are able to report an estimate of uncertainty to go with it.

The `logistf` package also provides a penalized likelihood approach developed by Firth (1993), which is equivalent to Bayesian logistic regression with a Jeffreys prior. The `logistf` function requires each observation to be binary (i.e., a Bernoulli response). Thus, we create a "long version" of our data set below.

```
library(logistf)
WellsLong <- NULL
for(i in 1:nrow(wells)){
  tempdat <- wells[i,]
  WellsLong <- rbind(WellsLong,
                     data.frame(contam = c(rep(1, tempdat$y), rep(0, tempdat$n- tempdat$y)),
                                land.use = rep(tempdat$land.use, tempdat$n),
                                sewer = rep(tempdat$sewer, tempdat$n)))
}
```

Then, we fit the model using means coding so that we can easily compare our results to the Bayesian approach:

```
lrfirth <- logistf(contam ~ land.use + sewer -1, data = WellsLong, family=binomial())
summary(lrfirth)
```

```
## logistf(formula = contam ~ land.use + sewer - 1, data = WellsLong,
##     family = binomial())
##
## Model fitted by Penalized ML
## Coefficients:
##                     coef  se(coef) lower 0.95 upper 0.95    Chisq            p
## land.useagri  -3.424344 0.6617365  -5.025055 -2.3136352 68.76325 1.110223e-16
## land.usecomm  -1.741409 0.3939727  -2.544436 -0.9788752 20.92229 4.782972e-06
## land.useindus -1.409143 0.3872128  -2.200166 -0.6614146 14.05506 1.775357e-04
## land.useinst  -2.811149 0.4483773  -3.760190 -1.9761494 56.48525 5.662137e-14
## land.userecr  -2.895559 0.4450532  -3.853023 -2.0799366 68.76325 1.110223e-16
## land.useresH  -1.878656 0.3381481  -2.574947 -1.2278191 35.33950 2.769582e-09
## land.useresL  -2.770159 0.6920825  -4.415058 -1.5807259 28.47004 9.515975e-08
## land.useresM  -2.580059 0.3157102  -3.244995 -1.9922454      Inf 0.000000e+00
## land.usetrans -2.777363 0.4372769  -3.707734 -1.9676855 60.79674 6.328271e-15
## land.useundev -5.632506 1.4246979 -10.478750 -3.6571844      Inf 0.000000e+00
## seweryes       1.555010 0.2842178   1.004490  2.1479905 33.46428 7.258474e-09
##               method
## land.useagri       2
## land.usecomm       2
## land.useindus      2
## land.useinst       2
## land.userecr       2
## land.useresH       2
## land.useresL       2
## land.useresM       2
## land.usetrans      2
## land.useundev      2
## seweryes           2
##
## Method: 1-Wald, 2-Profile penalized log-likelihood, 3-None
##
## Likelihood ratio test=347.4569 on 11 df, p=0, n=650
## Wald test = 168.1994 on 11 df, p = 0
```

We see again that the estimate of the coefficient for the undeveloped category is shrunk back towards 0, but not as much as in our Bayesian model using the N(0, 3) prior.

17

Models for zero-inflated data

Learning objectives

1. Be able to fit models to response data with lots of zeros (hurdle and zero-inflated models).
2. Be able to describe these models and their assumptions using equations and text and match parameters in these equations to estimates in computer output.

17.1 R packages

We begin by loading a few packages upfront:

```
set.seed(1288)
library(patchwork) # for creating multi-panel plots
library(ggplot2) # for plotting
library(R2jags) # for interfacing with JAGS
library(MCMCvis) # for summarizing MCMC output
library(mcmcplots) # for visualizing MCMC output
library(sjPlot) # for creating tables
library(modelsummary) # for creating tables
library(dplyr) # for data wrangling
library(knitr) # for creating reproducible reports
library(MASS) # for fitting negative binomial regression models
library(pscl) # for fitting zero-inflated models
```

In addition, we will use data and functions from the following packages:

- `Data4Ecologists` for the slugs data set
- `glmmTMB` for fitting zero-inflated models

17.2 Zero-inflation

Presence-absence and count data collected in ecological monitoring efforts are often analyzed using models that allow for zero-inflation – i.e., an overabundance of 0's. The first thing to note about zero-inflation is that it refers to response data, Y_i, not predictors, X_i. Although we will focus on zero-inflated count data, zero-inflation is also relevant to binary data (e.g., occupancy models as discussed in Chapter 20 of Kéry 2010), and continuous data (e.g., Friederichs et al. 2011).

There are many reasons why we may end up with an overabundance of zero counts in ecological monitoring data sets:

- sites may not be suitable for the species;
- the species may occur at low densities, and therefore suitable sites may not all be occupied;
- our sampling effort may not have been sufficient to detect individuals that use the site or we may have sampled during times when individuals are difficult to detect;
- we may have mistaken individuals for the wrong species.

Common statistical distributions used to model count data, including the Poisson and Negative Binomial distributions, allow for zeros:

- Poisson Distribution: $P(Y_i = 0) = \exp(-\lambda_i)$
- Negative Binomial Distribution: $P(Y_i = 0) = \left(\frac{\theta}{\mu+\theta}\right)^{\theta}$

Zero-inflation refers to situations where there are more zeros than expected given the assumed statistical distribution. In regression applications, we are allowing the mean of these distributions to depend on predictor variables. There may be some regions of "predictor space" where the mean is really small, and thus, where we might expect to find many zeros. So, a lot of zeros does not necessarily mean that data are zero-inflated. As the mean gets larger, however, we should expect fewer zeros. When certain regions of predictor space have both a large mean and many zeros, we might want to consider models that allow for zero-inflation.

17.3 Testing for excess zeros

How can we determine if we have excess zeros (i.e., zero-inflated data)? One option would be to modify our goodness-of-fit test (see Section 15.4.5) to use the number of zeros in our data set as the test statistic (rather than the sum of Pearson residuals). To demonstrate, let's revisit the slug data from Sections 10 and 15.

```
library(Data4Ecologists)
data(slugs)
```

We will use the total number of zeros as our test statistic: $T = \sum_{i=1}^{n} I(Y_i = 0)$. Let's work through the different steps involved in the hypothesis test:

1. State null and alternative hypotheses,

- H_0 : the data are consistent with a Poisson
- H_A : the data are not consistent with a model with separate λs for the field and Rookery sites; there are more 0's than expected if the data followed separate Poisson distributions for each site.

2. Calculate a statistic that measures the discrepancy between the data and the null hypothesis. Here, we will use $T =$ the number of zeros in the data set.

3. Determine the distribution of the statistic in step 2 *when the null hypothesis is true*. We will determine the distribution of T when the null hypothesis is true using simulations, allowing for uncertainty in our values of λ_i.
4. Compare the statistic for the observed data [step 2] to the distribution of the statistic when the null hypothesis is true [step 3]. The p-value = the probability of getting a value of T as big or bigger than the value for our observed data.

```
# Fitted model
  fit.pois<-glm(slugs~field, data=slugs, family=poisson())

 # Number of simulations
  nsims<-10000

# Set up matrix to hold goodness-of-fit statistics
  zeros.sim<-matrix(NA, nsims, 1)
  nobs<-nrow(slugs) # number of observations

# Extract the estimated coefficients and their asymptotic variance/covariance matrix
# Use these values to generate nsims new betas (beta.hat)
  beta.hat<-MASS::mvrnorm(nsims, coef(fit.pois), vcov(fit.pois))

# Design matrix for creating new lambda^'s
  xmat<-model.matrix(fit.pois)
  for(i in 1:nsims){
    # Generate lambda^'s
    lambda.hat<-exp(xmat%*%beta.hat[i,])
    # Generate new simulated values
    new.y<-rpois(nobs, lambda = lambda.hat)
    # Count the number of zeros
    zeros.sim[i]<-sum(new.y==0)
  }
```

We can then plot the distribution of T when the null hypothesis is true and overlay the number of zeros from our actual data set (Figure 17.1).

```
# T for our data
(T<-sum(slugs$slugs==0))
```

```
## [1] 34
```

```
hist(zeros.sim, xlab="# of zeros", ylab="Density", main="", col="gray", xlim=c(0, 35))
abline(v = T, col = "blue")
```

Lastly, we calculate the p-value as the probability of getting a value of $T \geq 34$ when the null hypothesis is true:

```
#pvalue
sum(zeros.sim>=T)/nsims
```

```
## [1] 0
```

Clearly, there are more zeros in the observed data than one might expect if the data were Poisson distributed with separate λs for each site.

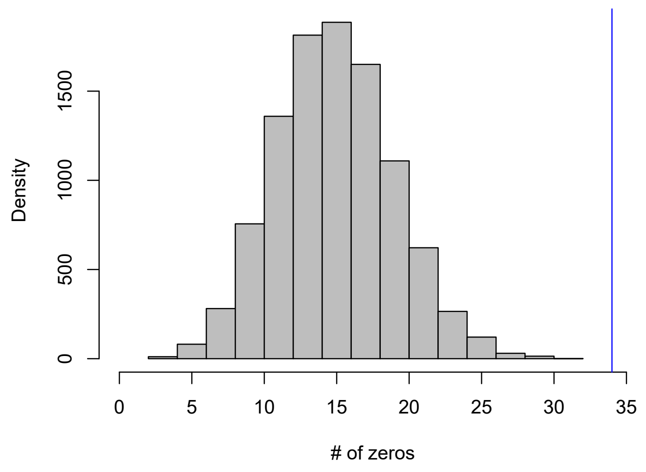


FIGURE 17.1 Distribution of zeros in data sets generated by the assumed Poisson regression model. Note that our test statistic (number of zeros in our original data set, which equaled 34) falls way out in the tail of the distribution, giving us a p-value of essentially 0.

17.4 Zeros and the Negative Binomial model

Before considering zero-inflated models, it is important to recognize that the Negative Binomial distribution often provides a reasonable fit to data with many zeros. This has been my experience, and is also the view that has been expressed by Paul Allison, a statistician with lots of experience analyzing real data - see his blog here.

For example, if we repeat the goodness-of-fit test from the last section with a negative binomial model, we find that we do not reject the null hypothesis (Figure 17.2).

```
# Fitted model
  fit.nb<-MASS::glm.nb(slugs~field, data=slugs)

 # Number of simulations
  nsims<-10000

# Set up matrix to hold goodness-of-fit statistics
  zeros.sim<-matrix(NA, nsims, 1)
  nobs<-nrow(slugs) # number of observations

# Extract the estimated coefficients and their asymptotic variance/covariance matrix
# Use these values to generate nsims new beta's
  beta.hat<-MASS::mvrnorm(nsims, coef(fit.nb), vcov(fit.nb))
  theta.hat<-rnorm(nsims, fit.nb$theta, fit.nb$SE.theta)

# Design matrix for creating new lambda^'s
```

```
  xmat<-model.matrix(fit.nb)
  for(i in 1:nsims){

    # Generate lambda^'s and thetas
    mus<-exp(xmat%*%beta.hat[i,])

    # Generate new simulated values
    new.y<-rnbinom(nobs, mu = mus, size=theta.hat[i])

    # Count the number of zeros
    zeros.sim[i]<-sum(new.y==0)
  }
```

```
Warning in rnbinom(nobs, mu = mus, size = theta.hat[i]): NAs produced
```

```
# T for our data
hist(zeros.sim, xlab="# of zeros", ylab="Density", main="", col="gray")
abline(v = T, col = "blue")
```

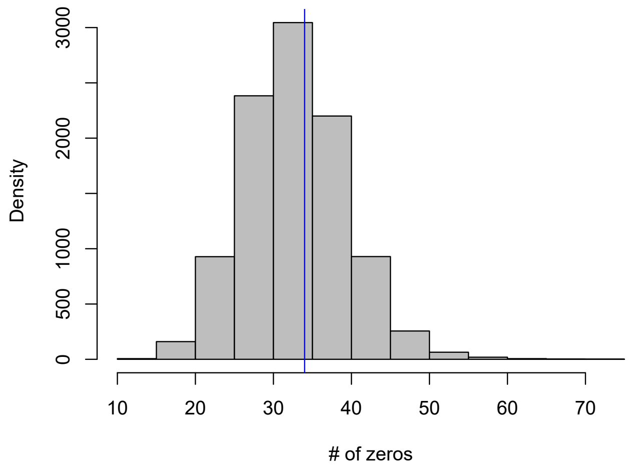


FIGURE 17.2 Distribution of zeros in data sets generated by the assumed negative binomial regression model. In this case, our test statistic (T = 34) falls near the mean of the distribution, and we fail to reject the null hypothesis that the data are consistent with the Negative Binomial regression model.

```
#pvalue
ind<-is.na(zeros.sim)!=TRUE # some NAs due to getting negative thetas
sum(zeros.sim[ind]>=T)/sum(ind)
```

```
[1] 0.4681468
```

17.5 Hurdle models

There are two general classes for modeling zero-inflated data, Hurdle models and zero-inflated mixture models. We will briefly introduce hurdle models here but then focus on the latter class of model for the rest of this section. Hurdle models contain two different sub-components:

1. A model for whether or not an observation is 0.
2. A truncated count model for the non-zero observations.

These models can be fit using a two-step approach:

1. Create a data set with new response variable, Z_i, indicating whether the original response is a 0 or not:

$$Z_i = \begin{cases} 1 & \text{if } Y_i = 0 \\ 0 & \text{if } Y_i \neq 0 \end{cases}$$

Model Z_i using logistic regression:

$$\begin{aligned} Z_i &\sim Bernoulli(p_i) \\ logit(p_i) &= \beta_0 + \beta_1 X_{i,1} + \ldots \beta_p X_{i,p} \end{aligned} \tag{17.1}$$

2. Create a second data set containing only the non-zero observations, $\widetilde{Y}_i$. Model $\widetilde{Y}_i$ using a truncated count distribution, usually either a truncated-Poisson or truncated Negative Binomial. A truncated distribution is one in which we modify the probability mass function to exclude values below a threshold (in this case we exclude the possibility of a zero; see Section 17.5.1).

Alternatively, hurdle models can be fit in one step using the `zeroinfl` function in the `pscl` library. Fitting the model in a single step can be accomplished using maximum likelihood with the following probability mass function:

$$P(Y = y) = f(y; \pi, \lambda) = \begin{cases} \pi & \text{if } y = 0 \\ (1-\pi) f_T(y; \lambda) & \text{if } y = 1, 2, 3, \ldots \end{cases}$$

where $f_T(y; \lambda)$ is a truncated count distribution.

17.5.1 Truncated distributions

Remember our rule for calculating conditional probabilities: P(A|B)=P(A and B)/P(B) (Chapter 9). We can use this rule to determine the probability mass function for a truncated count distribution (truncated at 0):

$f_T(y; \lambda) = P(Y = y | Y > 0) = \frac{P(Y=y)}{P(Y>0)} = \frac{f(y;\lambda)}{(1-f(0;\lambda))}$

where $f(y; \lambda)$ is the probability mass function for the non-truncated distribution and $f(0; \lambda) = P(Y = 0)$.

For example, the probability mass function for a truncated Poisson distribution with parameter λ would be given by:

$$P(Y=y|y>0) = \frac{\frac{e^{-\lambda}\lambda^y}{y!}}{1-e^{-\lambda}} = \frac{e^{-\lambda}\lambda^y}{y!(1-e^{-\lambda})}$$

We could write a function to calculate these probabilities as below. We compare non-truncated and truncated Poisson distributions with the same λ in Figure 17.3.

```
par(mfrow=c(1,2))
dtpois<-function(x, lambda){
    0 + I(x>0)*dpois(x, lambda)/(1-dpois(0, lambda))
}
```

```
x<-seq(0,20, by=1)
plot(x, dpois(x, lambda=2), type="h", ylim=c(0, 0.35), ylab="P(X=x)", main="Poisson")
plot(x, dtpois(x, lambda=2), type="h", ylim=c(0, 0.35), ylab="P(X=x)", main="Truncated Poisson")
```

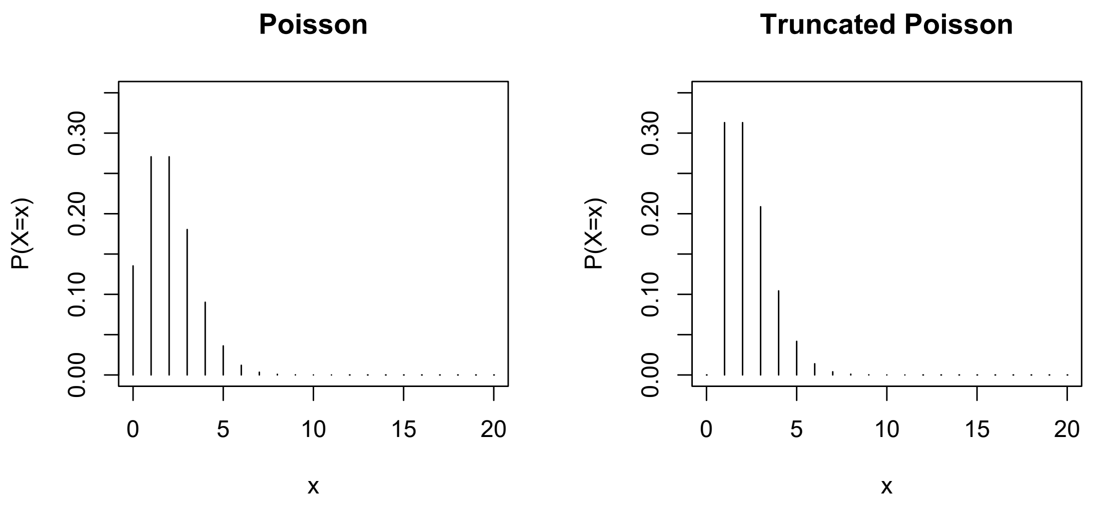


FIGURE 17.3 Comparison of truncated and non-truncated Poisson distributions with $\lambda = 2$.

It is also straightforward to allow the log mean of the non-zero observations to be modeled as a linear function of predictor variables:

$$\begin{aligned} Y_i|Y_i > 0 &\sim Poisson(\lambda_i) \\ \log(\lambda_i) &= \beta_0 + \beta_1 X_{1,i} + \dots \beta_p X_{p,i} \end{aligned} \tag{17.2}$$

A few important notes:

- The same (or different) variables may be included in the two model sub-components – i.e., the model for whether or not an observation is 0 (logistic component of the model; (17.1)) and the model for the mean of the non-zero observations (eq. (17.2)).
- λ_i describes the mean of the non-zero observations, not the overall mean of the observations, which is given by: $E[Y_i|X_i] = (1-p_i)\frac{\lambda_i}{1-e^{-\lambda_i}}$.

For more detail, along with expressions for $E[Y_i|X_i]$ and $Var[Y_i|X_i]$ for Poisson and Negative Binomial hurdle models, see page 288 of Zuur et al. (2009).

17.5.2 Hurdle models for continuous response data

The same general approach can be used to model continuous response data with many zeros. For continuous data, the non-zero observations can be modeled using log-normal or gamma distributions. Since these distributions already exclude zeros, there is no need to derive truncated versions of them. Alternatively, it is possible to use a truncated version of continuous probability distributions that allow for negative responses. For example, a truncated Normal distribution could be constructed using $P(Y|Y > 0) = \frac{f(y;\mu,\sigma^2)}{1-F(0;\mu,\sigma^2)}$ where $F(y) = P(Y \leq y)$. We could create a function for calculating the probability density function of a truncated Normal as follows:

```
dtnorm<-function(x, mean, sd){
    0+I(x>0)*dnorm(x, mean, sd)/(1-pnorm(0, mean, sd))
}
```

17.6 Zero-inflated mixture models

In the next section, we will model counts of the number of fish that were caught by visitors of a state park. Not everyone that provided data actually fished (and, there is no way of knowing who went fishing and who didn't since the park did not ask this question of survey respondents). It is easy to imagine two different ways to end up with 0 fish at the end of a camping trip: either you didn't fish at all or you fished but were not successful. Some predictors may be uniquely suited for modeling whether or not you fished, while others may predict both whether you fished and also how successful you were. Zero-inflated mixture models provide a formal way to model these separate processes.

Like hurdle models, zero-inflated mixture models also have two model sub-components, both of which allow for the possibility of a 0:

1. A model for whether or not an observation is an inflated 0.
2. A count model that also allows for the possibility of a 0.

17.6.1 Marginal distribution

Let's begin by considering a zero-inflated Poisson model without any covariates. Let Y_i be our zero-inflated response variable. We can think of Y_i as being determined by two random processes. First, a coin is flipped to determine if Y_i is an inflated zero; we get an inflated zero with probability π. If Y_i is not an inflated zero, then we generate a Poisson random number. This allows us to get a 0 in two ways:

- we get an inflated zero with probability π
- we get a non-inflated zero by: 1) not having an inflated zero (occurs with probability $1 - \pi$), and 2) the Poisson random variable is 0 (occurs with probability $f(0) = \exp(-\lambda_i)$). Combining these two probabilities using P(A and B) = P(A)P(B|A), we get the probability of a obtaining a 0 that is not inflated, which is equal to $(1 - \pi)e^{-\lambda}$.

To get any non-zero number, we have to: 1) not have an inflated zero (occurs with probability $1-\pi$), and 2) the Poisson random variable has to take on the specific value (this occurs with probability $f(y;\lambda)$). Thus, the probability that we get a non-zero response, y, is equal to $(1-\pi)f(y;\lambda)$.

Putting all of this together gives us the probability mass function for the zero-inflated Poisson model:

$$P(Y=y) = f(y;\pi,\lambda) = \begin{cases} \pi + (1-\pi)e^{-\lambda} & \text{if } y=0 \\ (1-\pi)\frac{e^{-\lambda}\lambda^y}{y!} & \text{if } y=1,2,3,\ldots \end{cases}$$

A zero-inflated version of the Negative Binomial can be constructed in much the same way, noting that the probability Mass Function for the Negative Binomial distribution is given by: $f(y;\mu,\theta) = \binom{y+\theta-1}{y}\left(\frac{\theta}{\mu+\theta}\right)^{\theta}\left(\frac{\mu}{\mu+\theta}\right)^{y}$ and $f(0;\mu,\theta) = \left(\frac{\theta}{\mu+\theta}\right)^{\theta}$.

Thus:

$$f(y;\mu,\theta) = \begin{cases} \pi + (1-\pi)\left(\frac{\theta}{\mu+\theta}\right)^{\theta} & \text{if } y=0 \\ (1-\pi)\binom{y+\theta-1}{y}\left(\frac{\theta}{\mu+\theta}\right)^{\theta}\left(\frac{\mu}{\mu+\theta}\right)^{y} & \text{if } y=1,2,3,\ldots \end{cases}$$

17.6.2 Zero-inflated mixture models defined using a latent variable

In my description of the zero-inflated mixture model in the previous section, I alluded to two random processes, the first of which was a coin flip to determine whether the response was an inflated 0. The second random process generated a Poisson random number (but only when the first process did not result in an inflated zero). It is often useful to formalize this idea by using a latent or unobserved variable, Z_i, to define the zero-inflation process. Specifically, we may define our zero-inflated Poisson model as:

$$Z_i \sim Bernouli(p_i)$$
$$Y_i|Z_i = 0 \sim Poisson(\lambda_i)$$

Alternatively, we may write:

$$Z_i \sim Bernouli(p_i)$$
$$Y_i|Z_i \sim Poisson((1-Z_i)\lambda_i)$$

We will use the latter approach to define zero-inflated models in JAGS. In my opinion, this is an area where Bayesian methods shine – they provide an easy way to fit models that can be formulated in terms of latent (unobserved) variables.

As with hurdle models, it is important to understand that λ_i (or μ_i if fitting a zero-inflated Negative Binomial) describes the mean, conditional on the observation not being an inflated zero ($E[Y_i|X_i, Z_i=0]$). The unconditional mean and variance are given by[1]:

- $E[Y_i|X_i] = (1-p_i)\lambda_i$ and $E[Y_i|X_i] = (1-p_i)\mu_i$ for the Poisson and Negative Binomial distributions, respectively.
- $Var[Y_i|X_i] = (1-p_i)(\lambda_i + p_i\lambda_i^2)$ and $Var[Y_i|X_i] = (1-p_i)(\mu_i + \frac{\mu_i^2}{\theta}) + \mu_i^2 p_i(1-p_i)$ for the Poisson and Negative Binomial distributions, respectively.

These expressions are needed when constructing Pearson residuals. As with hurdle models, it is also easy to incorporate predictors that influence the probability of getting an inflated-zero or the mean of our non-inflated observations. Let's take a look with an applied example.

[1]These expressions can be derived fairly easily using $E[Y] = E_X(E[Y|X])$ and $Var[Y] = Var_X(E[Y|X]) + E_X(Var[Y|X])$.

17.7 Example: Fishing success in state parks

Here, we will consider the `zeroinfl` function in the `pscl` package for fitting zero-inflated versions of the Poisson and Negative Binomial models (Jackman 2020). The `zeroinfl` function can also be used to fit hurdle models. We will consider a data set and example from UCLA's Statistical Consulting website. The data have also been incorporated into the `Data4Ecologists` package and can be accessed using:

```
data(fish)
```

The data set is constructed from surveys of 250 groups that visited a state park and contains the following variables:

- `count` = reported number of fish the group caught
- `child` = how many children were in the group
- `persons` = the number of people that were in the group
- `camper` = 1 if the group brought a camper to the campground and 0 otherwise

On the UCLA page, they considered the following zero-inflated model:

$$Z_i|X_i \sim Bernoulli(p_i) \quad (17.3)$$
$$\text{logit}(p_i) = \gamma_0 + \gamma_1 persons_i \quad (17.4)$$
$$Y_i|X_i, Z_i \sim Poisson((1-Z_i)\lambda_i) \quad (17.5)$$
$$\log(\lambda_i) = \beta_0 + \beta_1 child_i + \beta_2 camper_i \quad (17.6)$$

I.e., they assumed that the probability of an inflated zero depended on the number of people in the group and that the mean number of fish caught, given the observation was not an inflated zero, depended on the number of children in the group and whether the group brought a camper.

As someone who likes to fish and also camp with family members (3 of whom were once young children), I think a better model would be:

$$Z_i|X_i \sim Bernoulli(p_i) \quad (17.7)$$
$$logit(p_i) = \gamma_0 + \gamma_1 child_i \quad (17.8)$$
$$Y_i|X_i, Z_i \sim Poisson((1-Z_i)\lambda_i) \quad (17.9)$$
$$\log(\lambda_i) = \beta_0 + \beta_1 child_i + \beta_2 camper_i + \beta_3 person_i \quad (17.10)$$

We can fit this model as follows:

```
m.zip <- zeroinfl(count ~ child + camper +persons | child, data = fish, dist="poisson")
summary(m.zip)
```

```
##
## Call:
## zeroinfl(formula = count ~ child + camper + persons | child, data = fish,
##     dist = "poisson")
##
```

```
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -1.4052 -0.7178 -0.4505 -0.1199 26.9595
##
## Count model coefficients (poisson with log link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.05721    0.18123  -5.834 5.42e-09 ***
## child       -1.16755    0.09471 -12.327  < 2e-16 ***
## camper       0.77091    0.09384   8.215  < 2e-16 ***
## persons      0.88856    0.04663  19.056  < 2e-16 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.9150     0.2503  -3.655 0.000257 ***
## child         1.1857     0.2654   4.468 7.89e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 11
## Log-likelihood:  -766 on 6 Df
```

The variables on the right hand side of the "|" specify the variables that influence the probability of an inflated zero (in this case, `child`). We can interpret the coefficients of the zero-inflation part of the model in the same way as we did for logistic regression. We see that the odds of an inflated zero increases by a factor of exp(1.1857) = 3.27 for each additional child in the group. Furthermore, a group with 3 children is exp(1.1857*3) = 35 times more likely to be an inflated zero than a group with 0 children. Lastly, the probability that a group with 3 children is an inflated zero is given by $\frac{\exp(-0.9150+3\times1.1858)}{1+\exp(-0.9150+3\times1.1858)}$ and can be calculated using the `plogis` function:

```
plogis(-0.9150+3*1.1857)
```

```
## [1] 0.9335224
```

I am not too surprised – it is a lot of work to take 3 children fishing (even if it is incredibly rewarding)! Looking at the count part of the model, we see that groups with children catch fewer fish, whereas larger groups and groups that have a camper tend to catch more fish. Perhaps those with a camper also have better equipment (depth finders, a nice boat, etc). As with standard count models, we can estimate multiplicative effects on the mean number of fish caught, **given the observation is not an inflated zero**, by exponentiating these parameters. For example, for non-inflated observations, groups with a camper are expected to catch exp(0.77091) = 2.16 times as many fish as groups without a camper.

We can also fit a zero-inflated negative binomial model by changing the distribution argument (`dist`):

$$Z_i|X_i \sim Bernoulli(p_i) \tag{17.11}$$
$$logit(p_i) = \gamma_0 + \gamma_1 child_i \tag{17.12}$$
$$Y_i|X_i, Z_i \sim NegativeBinomial((1-Z_i)\mu_i, \theta) \tag{17.13}$$
$$\log(\mu_i) = \beta_0 + \beta_1 child_i + \beta_2 camper_i + \beta_3 person_i \tag{17.14}$$

```
m.zinb <- zeroinfl(count ~ child + camper +persons | child,
               data = fish, dist = "negbin")
summary(m.zinb)
```

```
##
## Call:
## zeroinfl(formula = count ~ child + camper + persons | child, data = fish,
##     dist = "negbin")
##
## Pearson residuals:
##      Min       1Q   Median       3Q      Max
## -0.71609 -0.54836 -0.34329 -0.02088 15.00807
##
## Count model coefficients (negbin with log link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.6600     0.3197  -5.192 2.08e-07 ***
## child        -1.2056     0.2715  -4.441 8.95e-06 ***
## camper        0.5834     0.2379   2.452   0.0142 *
## persons       1.0516     0.1110   9.477  < 2e-16 ***
## Log(theta)   -0.5824     0.1823  -3.194   0.0014 **
##
## Zero-inflation model coefficients (binomial with logit link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -4.4302     1.5159  -2.923 0.003472 **
## child         2.9263     0.8478   3.452 0.000557 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Theta = 0.5586
## Number of iterations in BFGS optimization: 26
## Log-likelihood: -399.9 on 7 Df
```

Note that the `zeroinfl` function parameterizes the overdispersion parameter, θ on the log-scale. This ensures that the estimate of θ, is positive (exp(`Log(theta)`) = exp(-0.5823) = 0.5586, which is also printed at the bottom of the summary output). The qualitative conclusions are similar to those of the zero-inflated Poisson model, but the coefficients and p-values have changed slightly.

We can compare the two zero-inflated models and also poisson and negative binomial models using AIC (below):

```
m.pois<- glm(count ~ child + camper + persons, family = poisson, data = fish)
m.nb <- glm.nb(count ~ child + camper + persons, data = fish)
AIC(m.pois, m.nb, m.zip, m.zinb)
```

```
##        df       AIC
## m.pois  4 1682.1450
## m.nb    5  820.4440
## m.zip   6 1544.0557
## m.zinb  7  813.8197
```

This comparison suggests that the zero-inflated negative binomial model provides the best fit to the data, with the non-zero inflated negative binomial model not far behind. The Poisson and zero-inflated Poisson models, by comparison, provide poor fits to the data. My experience, and that of others, is that a Negative Binomial model (without zero-inflation) often “wins” followed by the zero-inflated negative binomial distribution (see e.g., Gray 2005; Warton 2005; Sileshi 2008 for comparative studies).

Lastly, we exponentiate the parameters in our models to summarize effect sizes using incidence ratios for our count models (Table 17.1) and incidence and odds ratios for our zero-inflated models (Table 17.2).

TABLE 17.1 Odds and Incidence Ratios [95 Percent confidence Intervals]

	Poisson	Negative Binomial
	Incidence Ratio	**Incidence Ratio**
(Intercept)	0.138 [0.102, 0.185]	0.197 [0.103, 0.376]
child	0.185 [0.157, 0.215]	0.169 [0.114, 0.243]
camper	2.537 [2.137, 3.031]	1.861 [1.168, 2.947]
persons	2.978 [2.761, 3.220]	2.889 [2.306, 3.659]
RMSE	9.65	9.71

TABLE 17.2 Odds and Incidence Ratios [95 percent confidence Intervals]

	Zero-Inflated Poisson	Zero-Inflated Negative Binomial
Count model	**Incidence Ratio**	**Incidence Ratio**
count_(Intercept)	0.347 [0.244, 0.496]	0.190 [0.102, 0.356]
count_child	0.311 [0.258, 0.375]	0.300 [0.176, 0.510]
count_camper	2.162 [1.799, 2.598]	1.792 [1.124, 2.857]
count_persons	2.432 [2.219, 2.664]	2.862 [2.303, 3.558]
Zero-inflation model	**Odds Ratio**	**Odds Ratio**
zero_(Intercept)	0.401 [0.245, 0.654]	0.012 [0.001, 0.232]
zero_child	3.273 [1.946, 5.506]	18.658 [3.542, 98.294]

17.8 Goodness-of-fit

The AIC comparisons only tell us which of our models is “best”, but they do not tell us if any of the models are actually any good. We could modify our goodness-of-fit test based on Pearson residuals. This would require generating Bernoulli random variables for the zero-inflation process and either Poisson or Negative Binomial random variables for the non-inflated zero observations. We demonstrate this approach when fitting the zero-inflated negative binomial model in JAGS (see Section 17.9). We can also explore plots of the observed counts or Pearson residuals versus fitted values (Figure 17.4).

```
fish$predict.m.zinb<-predict(m.zinb, type="response")
fish$presiduals.m.zinb<-resid(m.zinb, type="pearson")
p1<-ggplot(fish, aes(predict.m.zinb, count))+geom_point()+
    geom_jitter()+geom_abline(intercept=0,slope=1)+
  xlab("Predicted")+ylab("Observed")
```

```
p2<-ggplot(fish, aes(predict.m.zinb, presiduals.m.zinb)) +
    geom_point() +
    geom_jitter()+
    geom_abline(intercept=0,slope=0) +
    geom_smooth() +
  xlab("Predicted")+ylab("Pearson Residual")
p1+p2
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

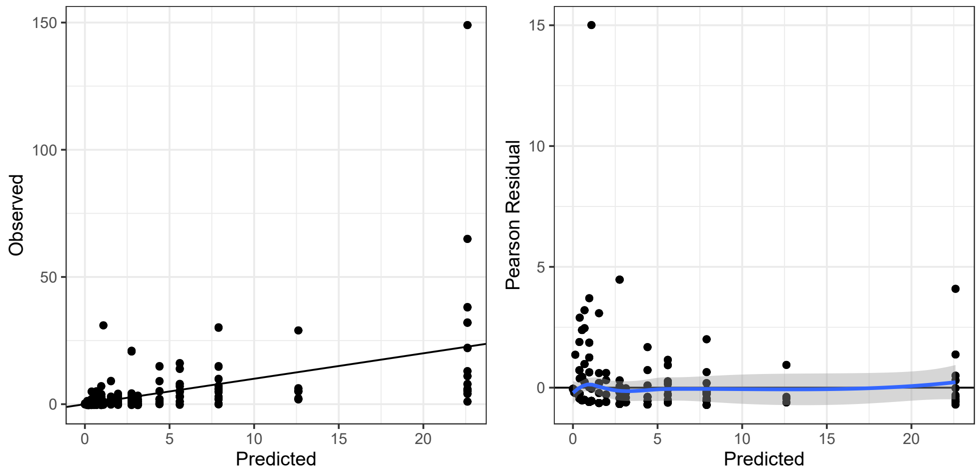


FIGURE 17.4 Observed counts (left panel) and Pearson residuals (right panel) plotted versus fitted values from the zero-inflated Negative Binomial model.

Inspecting these on these two plots (Figure 17.4), there are 2 observations we might want to look into further:

- the group that reported catching ~150 fish (is that even possible?)
- the observation with the large Pearson residual (~ 15).

We can use the `which.max` function to identify these observations:

```
ind1<-which.max(fish$count) # Max count
ind2<-which.max(fish$presiduals.m.zinb) # Max Pearson residual
fish[c(ind1, ind2),]
```

```
##     count livebait camper persons child predict.m.zinb presiduals.m.zinb
## 89    149        1      1       4     0      22.605575          4.091239
## 138    31        1      0       3     1       1.092746         15.008067
```

The group that caught 149 fish had 4 people with no children. The observation with the large Pearson residual was for a group of 3 people, one of which was a child, that caught 31 fish. Based on the Pearson residual, we might conclude that the latter observation is actually more of an outlier!

17.9 Bayesian implementation

Here, we follow the approach taken by Kéry (2010). Note that Kery parameterizes the model in terms of $\psi = 1 - \pi =$ the probability of a **non** zero-inflated response. In comparison to the formulation using `zerolinfl`, we have:

Parameterization used by `zeroinfl`:

$$P(Y = y) = f(y) = \begin{cases} \pi + (1-\pi)e^{-\lambda} & \text{if } y = 0 \\ (1-\pi)\frac{e^{-\lambda}\lambda^y}{y!} & \text{if } y = 1, 2, 3, \ldots \end{cases}$$

ZIP model (Kery):

$$P(Y = y) = f(y) = \begin{cases} 1 - \psi + \psi e^{-\lambda} & \text{if } y = 0 \\ \psi\frac{e^{-\lambda}\lambda^y}{y!} & \text{if } y = 1, 2, 3, \ldots \end{cases}$$

And, for the Negative Binomial model:

`zeroninfl` parameterization:

$$f(y) = \begin{cases} \pi + (1-\pi)\left(\frac{\theta}{\mu+\theta}\right)^{\theta} & \text{if } y = 0 \\ (1-\pi)\binom{y+\theta-1}{y}\left(\frac{\theta}{\mu+\theta}\right)^{\theta}\left(\frac{\mu}{\mu+\theta}\right)^{y} & \text{if } y = 1, 2, 3, \ldots \end{cases}$$

ZINB model (Kery):

$$f(y) = \begin{cases} 1 - \pi + \pi\left(\frac{\theta}{\mu+\theta}\right)^{\theta} & \text{if } y = 0 \\ \pi\binom{y+\theta-1}{y}\left(\frac{\theta}{\mu+\theta}\right)^{\theta}\left(\frac{\mu}{\mu+\theta}\right)^{y} & \text{if } y = 1, 2, 3, \ldots \end{cases}$$

The only difference is that the coefficients in the logistic part of the model will be of the opposite sign since we are now modeling the probability of a non-inflated zero rather than the probability of an inflated zero.

We use the latent variable approach to implement the Bayesian zero-inflated Negative Binomial model. Specifically, we will create a (partially) latent variable, `I.fish` (rather than Z_i), to indicate whether an observation is NOT an inflated 0. We will think of `I.fish` as an indicator of whether or not the group went fishing (if so, we do not have an inflated 0). We then model `count` as a Negative Binomial random variable with mean = `I.fish`$\times\mu_i$:

$$\begin{gathered} I.fish_i \sim Bernouli(p_i) \\ logit(p_i) = \gamma_0 + \gamma_1 child_i \\ count_i | I.fish_i \sim \text{Negative Binomial}(I.fish_i \times \mu_i, \theta) \\ \log(\mu_i) = \beta_0 + \beta_1 child_i + \beta_2 camper_i + \beta_3 person_i \end{gathered}$$

What makes this model different from others that we have considered so far is that we do not actually have a variable in our data set called `I.fish` that takes on a value of 1 when the group went fishing and 0 otherwise. We will create the `I.fish` variable, initially setting all of its values to NA (missing). We then set `I.fish` = 1 for all groups that caught a fish (these cannot be inflated zeros), leaving the value of `I.fish` = NA for all other observations. JAGS will impute values for `I.fish` for the observations with missing values during each MCMC iteration using information from the predictor variables for those individuals.

```
I.fish<-rep(NA, nrow(fish))
I.fish[fish$count>0]<-1 # these people caught fish and are NOT inflated zeros
```

Think-Pair-Share: How many priors will we need for this model?

```
znb<-function(){

# Priors for count model
 for(i in 1:4){
   beta.c[i] ~ dnorm(0,0.001)
 }

# Priors for zero-inflation model (fit on a logit scale)
 for(i in 1:2){
   beta.zi[i] ~ dnorm(0,1/3)
 }

# Overdispersion parameter
  theta ~ dunif(0,50)

# Likelihood
  for(i in 1:n){
  # zero-inflation part (logit prob NOT inflated 0, i.e., "went fishing"))
    logitpsi[i] <- beta.zi[1] + beta.zi[2]*child[i]
    psi[i] <- exp(logitpsi[i])/(1+exp(logitpsi[i])) # prob Not inflated 0
    I.fish[i] ~ dbern(psi[i]) # Not zero-inflated (i.e., "went fishing")

  # Count part
    log.mu[i] <- beta.c[1] + beta.c[2]*child[i] + beta.c[3]*camper[i] + beta.c[4]*persons[i]
    mu[i] <- exp(log.mu[i])
    mu.eff[i] <- mu[i]*I.fish[i]
    p[i] <- theta/(theta+mu.eff[i])
    count[i] ~ dnegbin(p[i], theta)

  # Mean and variances of the observations
  # Derived using:  Var(Count) = E[var(count|z)] + Var[E(count|z)]
    Ey[i] <- mu[i]*psi[i]
    Vary[i] <- psi[i]*(mu[i] + mu[i]^2/theta) + mu[i]^2*(psi[i]*(1 - psi[i]))

  # Generate "new" data
    I.fish.new[i] ~ dbin(psi[i],1)
    mu.eff.new[i] <- mu[i]*I.fish.new[i]
    p.new[i] <- theta/(theta + mu.eff.new[i])
    count.new[i] ~ dnegbin(p.new[i], theta)

  # Pearson residuals
    presi[i] <- (count[i] - Ey[i])/sqrt(Vary[i]) # Pearson Resid
    presi.new[i] <- (count.new[i] - Ey[i])/sqrt(Vary[i])

  # Discrepancy measures
    D[i] <- pow(presi[i], 2)
    D.new[i] <- pow(presi.new[i],2)
  }
  fit <- sum(D[])
  fit.new <- sum(D.new[])
}

# Bundle data
jagsdata <- list(count=fish$count,
                 child=fish$child,
                 camper = as.numeric(fish$camper) -1,
                 persons = fish$persons,
                 n = nrow(fish),
                 I.fish = I.fish )

# Parameters to estimate
params <- c("beta.zi", "beta.c", "Ey", "psi", "mu", "presi",
            "presi.new", "fit", "fit.new", "theta", "I.fish")


# Start sampler
out.znb <- jags.parallel(data = jagsdata, parameters.to.save = params,
                model.file = znb, n.thin = 10, n.chains = 3, n.burnin = 4000,
                n.iter= 15000)
MCMCsummary(out.znb, params = c("beta.zi", "beta.c", "theta"))
```

```
##                  mean        sd        2.5%        50%      97.5% Rhat n.eff
## beta.zi[1]  3.1979630 0.7343385  1.94629281  3.1324735  4.8302946 1.00   957
## beta.zi[2] -2.0270388 0.7593942 -3.16899035 -2.0911349 -0.2854357 1.06   427
## beta.c[1]  -1.0133695 0.3047952 -1.58937586 -1.0101328 -0.4157184 1.00  1387
## beta.c[2]  -1.2259605 0.2949835 -1.82877307 -1.2086964 -0.6647536 1.01  1253
## beta.c[3]   0.5735719 0.2424826  0.08064972  0.5742974  1.0443988 1.00  3300
## beta.c[4]   1.0475373 0.1112952  0.83646516  1.0471353  1.2684389 1.00  1482
## theta       0.5999159 0.1113954  0.41015527  0.5894695  0.8417613 1.01  2147
```

As always, we should explore traceplots and density plots to make sure that our sampler has converged (Figures 17.5, 17.6).

```
  denplot(out.znb, parms=c("beta.zi", "beta.c", "theta"), ask=FALSE)
```

```
  traplot(out.znb, parms=c("beta.zi", "beta.c", "theta"), ask=FALSE)
```

We can also calculate the Bayesian p-value for the goodness-of-fit test. Although the p-value is fairly small, the conclusions depend critically on the one outlier noted previously with Pearson residual ~ 15. If we drop that observation, the p-value increases substantially (results not shown).

```
fitstats <- MCMCpstr(out.znb, params = c("fit", "fit.new"), type = "chains")
T.extreme <- fitstats$fit.new >= fitstats$fit
(p.val <- mean(T.extreme))
```

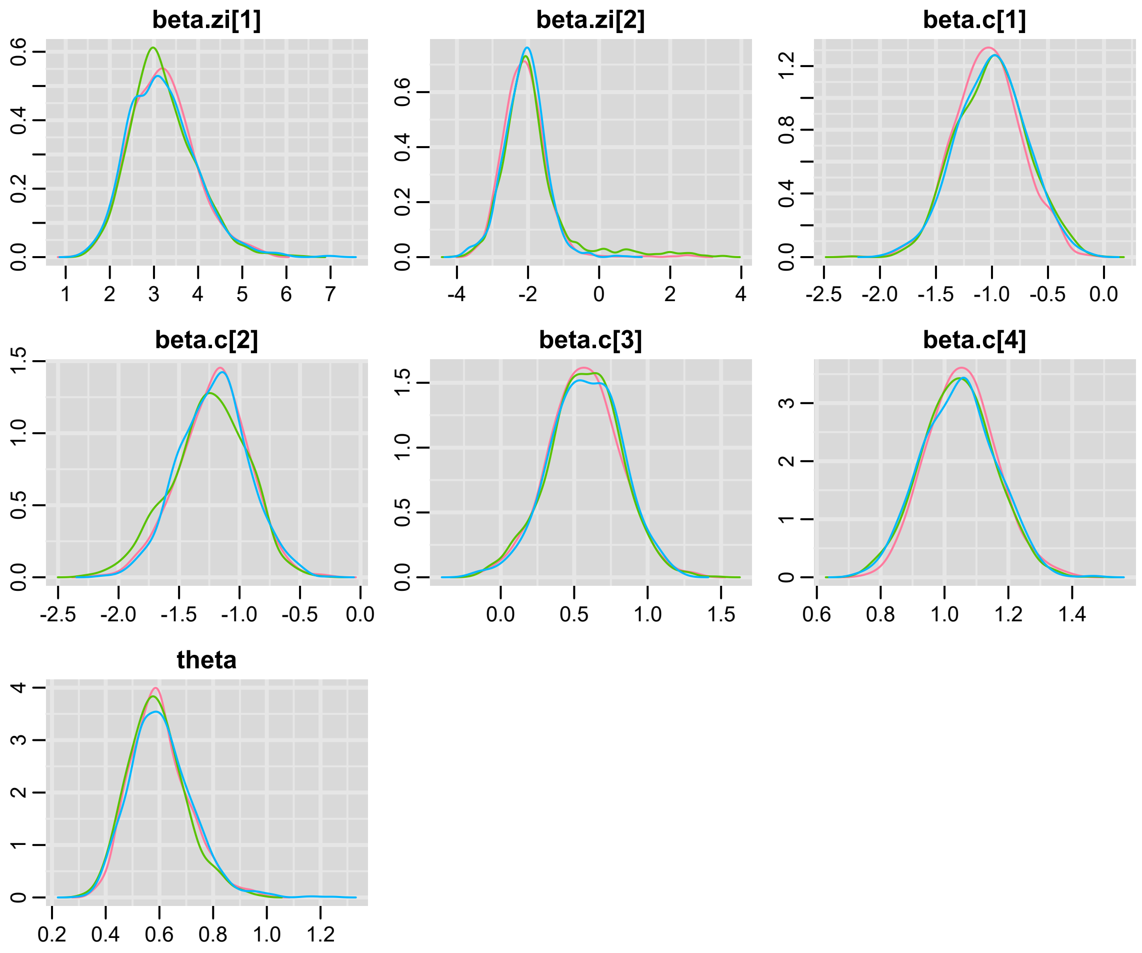


FIGURE 17.5 Posterior density plots from the Bayesian zero-inflated model fit to the fishing data set.

```
## [1] 0.06181818
```

Lastly, it may be interesting to look at the imputed values for `I.Fish` and how they relate to the number of children in each group. Below, we plot the distribution of posterior means for `I.fish` at each level of `child`.

```
fish$I.fish.hat<-out.znb$BUGSoutput$mean$I.fish
ggplot(fish, aes(x = as.factor(child), y = I.fish.hat)) +
    geom_boxplot() +
    xlab("Number of Children")+ylab("Went fishing?") +
    geom_jitter(color = "black", size = 1, alpha = 0.9)
```

We see that `I.fish` is rarely = 1 in groups with 2 or 3 children. This makes sense in light of the fact that no group with 3 children actually caught any fish and the mean number of fish caught for groups with 2 children was less than 1 fish!

```
fish %>% group_by(child) %>% dplyr::summarize(meanfish=mean(count))
```

```
## # A tibble: 4 x 2
```

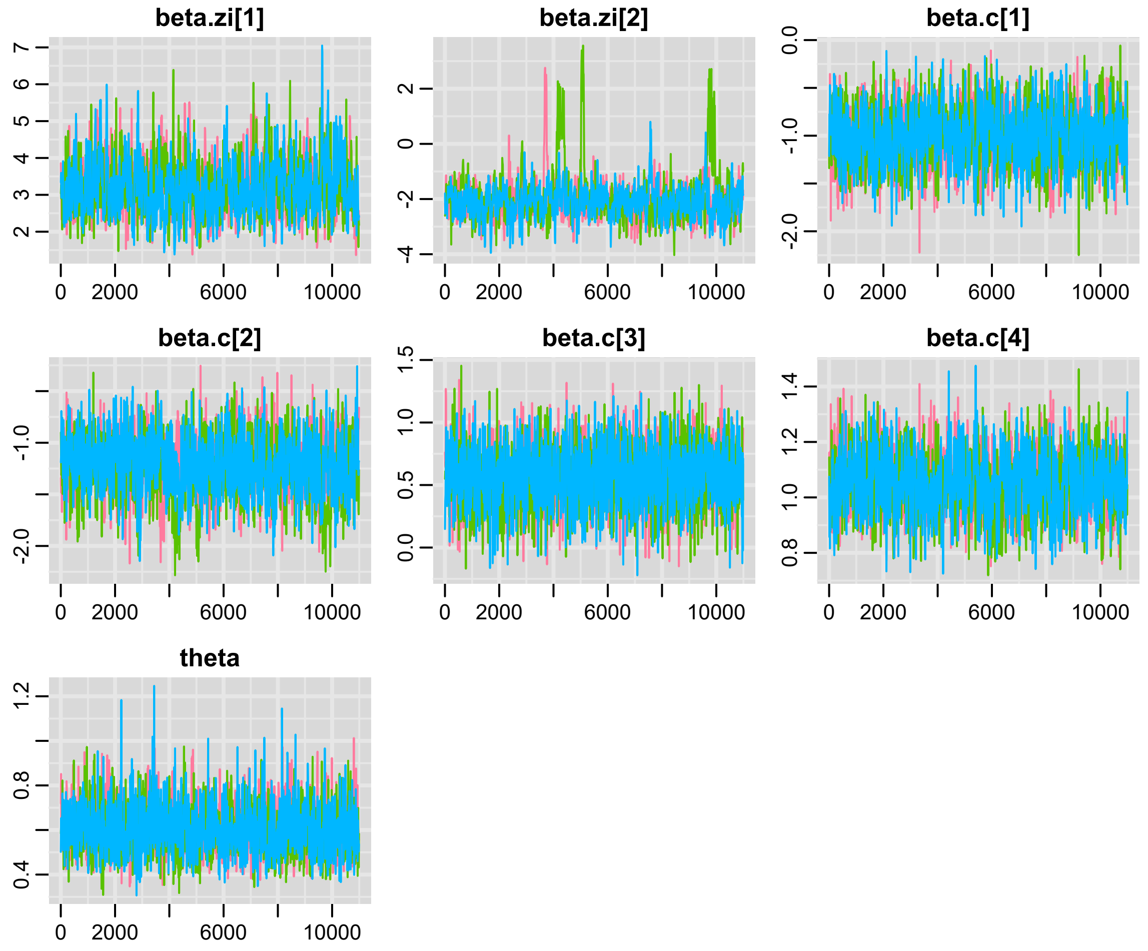


FIGURE 17.6 Trace plots from the Bayesian zero-inflated model fit to the fishing data set.

```
##   child meanfish
##   <int>    <dbl>
## 1     0    5.19
## 2     1    1.76
## 3     2    0.212
## 4     3    0
```

17.10 Implementation using glmmTMB

Although the `zeroinfl` function in the `pscl` package is probably the most widely used function for fitting zero-inflation models, it is also possible to fit zero-inflated Poisson and zero-inflated negative binomial models using the `glmmTMB` function in the `glmmTMB` package (Brooks et al. 2017). The main advantage to using `glmmTMB` is that it makes it possible to include random effects (e.g., when dealing with non-independent data; see Chapter 18). The `glmmTMB` function has a separate argument, `ziformula`, for specifying the zero-inflation

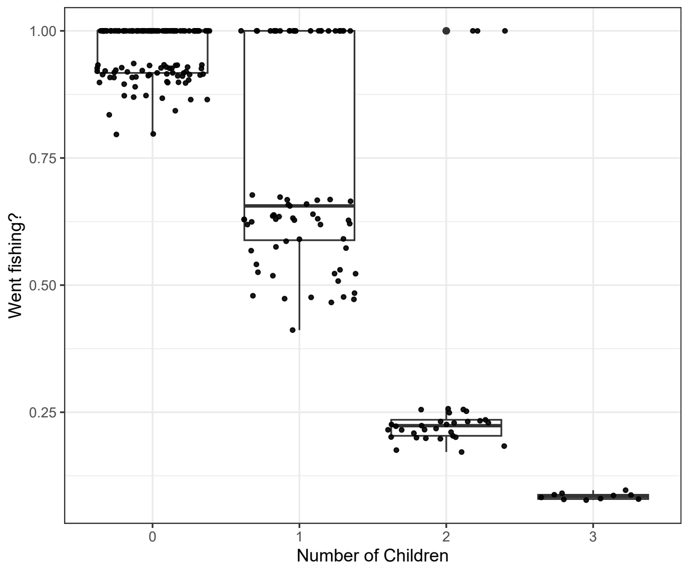


FIGURE 17.7 Boxplots depicting posterior means of `I.fish`, an indicator that the observation is not an inflated zero, as a function of `child`.

part of the model. It also uses the argument `family` to specify the distribution (either `poisson` or `nbinom2`). Below, we demonstrate the equivalence of zero-inflated Poisson models fit using `glmmTMB` and `zeroinfl`.

```
# Equivalent zero-inflated Poisson models
library(glmmTMB)
m.zip.TMB <- glmmTMB(count ~ child + camper + persons, ziformula = ~ child,
                     data = fish, family  = poisson)
m.zip
```

```
##
## Call:
## zeroinfl(formula = count ~ child + camper + persons | child, data = fish,
##     dist = "poisson")
##
## Count model coefficients (poisson with log link):
## (Intercept)        child       camper      persons
##     -1.0572      -1.1675       0.7709       0.8886
##
## Zero-inflation model coefficients (binomial with logit link):
## (Intercept)        child
##      -0.915        1.186
```

```
m.zip.TMB
```

```
## Formula:          count ~ child + camper + persons
## Zero inflation:          ~child
## Data: fish
##       AIC       BIC    logLik  df.resid
## 1544.0557 1565.1845 -766.0279       244
##
## Number of obs: 250
##
## Fixed Effects:
##
## Conditional model:
## (Intercept)        child       camper      persons
##     -1.0572      -1.1675       0.7709       0.8886
##
## Zero-inflation model:
## (Intercept)        child
##      -0.915        1.186
```

And, here, the equivalence between zero-inflated Negative Binomial models fit using `glmmTMB` and `zeroinfl`.

```
# Equivalent zero-inflated Negative Binomial Models
m.zinb.TMB <- glmmTMB(count ~ child + camper + persons, ziformula = ~ child,
                    data = fish, family = nbinom2)
m.zinb
```

```
##
## Call:
## zeroinfl(formula = count ~ child + camper + persons | child, data = fish,
##     dist = "negbin")
##
## Count model coefficients (negbin with log link):
## (Intercept)        child       camper      persons
##     -1.6600      -1.2056       0.5834       1.0516
## Theta = 0.5586
##
## Zero-inflation model coefficients (binomial with logit link):
## (Intercept)        child
##      -4.430        2.926
```

```
m.zinb.TMB
```

```
## Formula:          count ~ child + camper + persons
## Zero inflation:         ~child
## Data: fish
##       AIC       BIC    logLik  df.resid
##  813.8197  838.4700 -399.9099       243
##
## Number of obs: 250
##
## Dispersion parameter for nbinom2 family (): 0.559
##
## Fixed Effects:
##
## Conditional model:
## (Intercept)        child       camper      persons
##     -1.6600      -1.2056       0.5834       1.0516
##
## Zero-inflation model:
## (Intercept)        child
##      -4.430        2.926
```

Part V

Models for Correlated Data

18

Linear Mixed Effects Models

Learning objectives

1. Be able to identify situations when it is appropriate to use a mixed model.
2. Learn how to implement mixed models in R/JAGs for Normally distributed response variables.
3. Be able to describe models and their assumptions using equations and text and match parameters in these equations to estimates in computer output.

18.1 R packages

We begin by loading a few packages upfront:

```r
library(tidyverse) # for data wrangling
library(gridExtra) # for multi-panel plots
library(lme4)  # for fitting random-effects models
library(nlme) # for fitting random-effects models
library(glmmTMB) # for fitting random-effects models
library(sjPlot) # for visualizing fitted models
library(modelsummary) # for creating output tables
library(kableExtra) # for tables
options(kableExtra.html.bsTable = T)
library(ggplot2)# for plotting
library(performance) # for model diagnostics
```

In addition, we will use data and functions from the following packages:

- `Data4Ecologists` for the `Selake` and `HRData` data sets
- `lmerTest` for testing hypotheses using linear mixed effect models
- `tidy` in the `broom.mixed` package for accessing model output
- `R2jags` for fitting models using JAGS
- `MCMCvis` and `mcmcplots` for summarizing MCMC samples
- `cAIC` in the `cAIC4` package for calculating conditional AICs

18.2 Revisiting the independence assumption

A common assumption of all of the models we have fit so far is that our observations are independent. This assumption is unrealistic if we repeatedly record observations on the same sample units (i.e. we have repeated measures); this assumption may also be violated if our observations are measured close in space or close in time. Let's consider the implications of having non-independent data and also various strategies for accommodating non-independence using a simple simulated data set from Schwarz (2014). These data are contained in the `Data4Ecologists` package:

```
library(Data4Ecologists) # for data
data(Selake)
head(Selake)
```

```
##   Lake Log_Water_Se Log_fish_Se
## 1    a     -0.30103   0.9665180
## 2    a     -0.30103   1.1007171
## 3    a     -0.30103   1.3606978
## 4    a     -0.30103   1.0993405
## 5    a     -0.30103   1.2822712
## 6    b      0.83600   0.9713466
```

The data set contains selenium (Se) concentrations (on the log scale) measured in 83 fish sampled from 9 different lakes, along with log(Se) concentrations in those lakes. Selenium can leach from coal into surface waters when coal is mined, and therefore, this data set could conceivably have been collected during an environmental review to determine whether selenium in nearby lakes is bioaccumulating in fish. We begin by fitting a simple linear regression model to explore the relationship between logged concentrations in the water (`Log_Water_Se`) and log concentrations in fish (`Log_fish_SE`), assuming all locations are independent. We then plot the regression line overlaid on the observations, and inspect a residual versus fitted value plot; in both cases, we associate a unique color with the observations by lake (Figure 18.1.

```
fit.naive <- lm( Log_fish_Se ~ Log_Water_Se, data=Selake)
plot.naive <- ggplot(data = Selake,
    aes(x = Log_Water_Se, y = Log_fish_Se, col = Lake)) +
    ggtitle("Se in Fish vs Se Water with linear regression line") +
    xlab("log(Se) in the water\nPoints jittered") +
    ylab("log(Se) in fish") +
    geom_point(size = 3, position = position_jitter(width = 0.05)) +
    geom_abline(intercept = coef(fit.naive)[1], slope = coef(fit.naive)[2]) +
                    theme(legend.position="none")
residplot <- ggplot(broom::augment_columns(fit.naive, Selake),
                    aes(.fitted, .resid, col=Lake)) +
                    geom_point() +
                    geom_smooth() +
                    ggtitle("Residual vs. fitted values") +
                    xlab("Fitted values") +
                    ylab("Residuals")
ggpubr::ggarrange(plot.naive, residplot, ncol=2, common.legend=TRUE, legend="bottom")
```

Importantly, the number of fish sampled varies from lake to lake. Further, we see that the residuals for

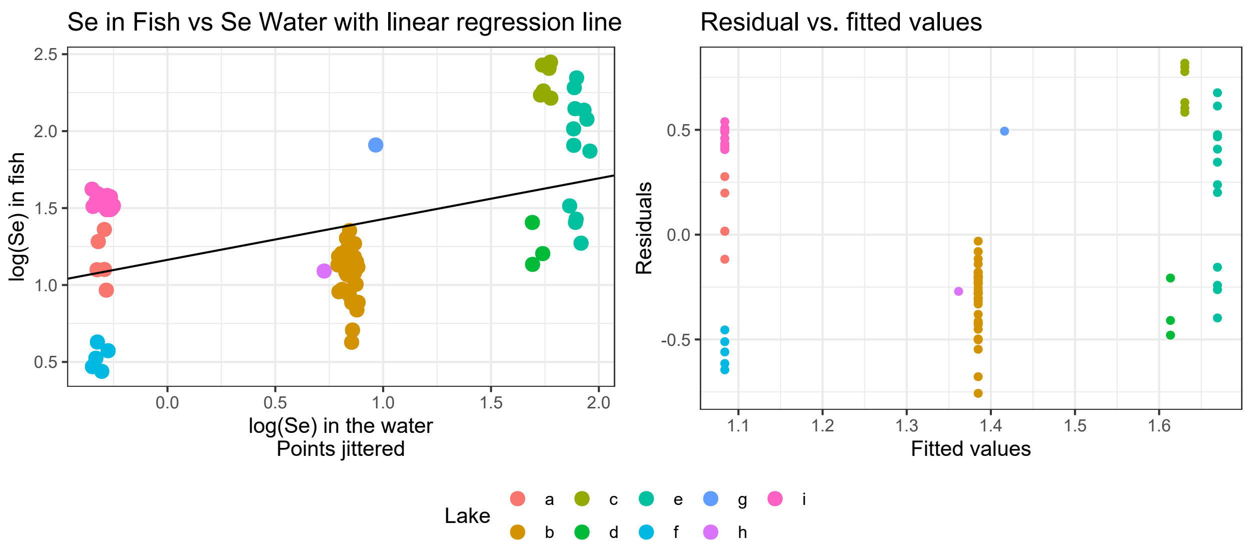


FIGURE 18.1 Plot of log(Selenium) concentrations in fish versus log(Selenium) concentration measured in nine lakes from which the fish were sampled (left panel). Residual versus fitted value plot for a linear regression model fit to these data, assuming all observations are independent.

different lakes tend to be clustered with most values being either negative or positive for any given lake. Thus, two randomly chosen residuals from the same lake are likely to be more alike than two randomly chosen residuals from different lakes. The data are clearly **not independent**!

Think-pair-share: What are the consequences of ignoring the fact that we have multiple observations from each lake?

Often when I pose this question to ecologists, they express concern that the estimates of the regression parameters will be *biased*. Here, it is important to recognize that bias has a specific definition in statistics. In particular, bias is quantified by the difference between the expected (or average) estimate across repeated random trials and the true parameter value. Again, by random trial, we mean the process of collecting and analyzing data in the same way. In this case, that means collecting data in a way that leads to an unbalanced data set (with unequal numbers of fish per lake) and then analyzing that data set using simple linear regression. When our data are not balanced and we treat observations as independent, it is clear that some lakes will contribute more information than others. But, does that imply we have a biased estimator? Well, it depends on the process that leads to the imbalance.

18.2.1 What if sample sizes are random?

There are clearly other factors that influence selenium concentrations in the water and in fish, but if the sample size in each lake is completely random (i.e., the number of fish sampled per lake does not depend on these other factors that are associated with our response variable), then our regression parameter estimators may be unbiased even if the data set is not balanced. On the other hand, we might expect naive estimates of uncertainty to be too small.

Let's explore the properties of the regression parameter estimators using simulations from the following data-generation model for log Se concentrations in fish:

$$log(SeFish)_{i,j} = \beta_0 + \beta_1 log(SeWater)_i + \tau_i + \epsilon_{ij}$$
$$\tau_i \sim N(0, \sigma^2_\tau)$$
$$\epsilon_{ij} \sim N(0, \sigma^2_\epsilon)$$

where $log(SeFish)_{i,j}$ is the log Se concentration of the j^{th} fish in the i^{th} lake, $log(SeWater)_i$ is the log Se concentration in i^{th} lake's water, and β_0 and β_1 are the regression parameters we are interested in estimating. Deviations about the regression line are due to two independent random factors represented by τ_i and ϵ_{ij}; τ_i is constant for all observations in a lake whereas ϵ_{ij} is unique to each observation. We can think of the τ_i as representing ecolgoical variables (other than the Se concentration in a lake) that cause all SeFish concentrations to be higher or lower, on average, for particular lakes[1], whereas the ϵ_{ij} represent other random factors that create variability among fish found in the same lake. Whenever writing down models for repeated measures data, it is critical that we use multiple subscripts (e.g., we use i to index each lake and j to index the different observations within each lake).

We will repeatedly generate a random number of observations from each of 9 randomly selected lakes. We will then fit a linear regression model to the data assuming all observations are independent and ignoring the lake-level random effects, τ_i. We will store the results to look at the sampling distribution of $\hat{\beta}_0$, $\hat{\beta}_1$, $\widehat{SE}(\hat{\beta}_0)$ and $\widehat{SE}(\hat{\beta}_1)$:

```
# Set.seed
set.seed(1024)

# Number of lakes and number of simulations
nlakes <- 10000 # Go Minnesota! :)
nsims <- 10000  # to approximate the sampling distribution

# Parameters used to simulate the data:
beta0 = 1.16 # intercept
beta1 = 0.26 # slope
sigma_tau = 0.45 # sd tau
sigma_epsilon = 0.20 #sd epsilon

# Matrix to hold results
beta.hat <- matrix(NA, nsims, 2)
standard.error.hat <- matrix(NA, nsims, 2)

# Generate lake Se values for each of 10,000 lakes.
# We will assume these are ~ N(1,1)
log_se_water<-rnorm(nlakes, 1, 1)

# Generate tau_i for 10,000 lakes
tau <- rnorm(nlakes, 0, sigma_tau)

# Simulation loop
for(i in 1:nsims){
  # pick a sample of 9 lakes
  lakeids <- sample(1:nlakes, 9)

  # determine a random sample size for each lake
```

[1]We will later refer to these as *random effects*, or more specifically, *random intercepts*.

```
  # using a negative binomial random number generator
  # with mean = mu and overdispersion parameter = theta
  mu <- 7 # mean number of observations per lake
  theta <- 1.15
  ni <- rnbinom(9, mu = mu, size = theta) # number of obs/lake
  log_se_fish <- beta0 + beta1*rep(log_se_water[lakeids], ni) +
                 rep(tau[lakeids], ni) + rnorm(sum(ni), 0, sigma_epsilon)
  dat <- data.frame(id = rep(lakeids, ni),
                    log_se_water = rep(log_se_water[lakeids], ni),
                    log_se_fish)
  lmtemp <- lm(log_se_fish~log_se_water, data=dat)
  beta.hat[i,] <- coef(lmtemp)
  standard.error.hat[i,]<-sqrt(diag(vcov(lmtemp)))
}
```

Let's begin by looking at the sampling distributions of our regression parameter estimators with the true parameters used to generate the data highlighted in red:

```
# plot
par(mfrow=c(1,2))
hist(beta.hat[,1], main="Intercept", xlab="Estimate", ylab="Sampling Distribution")
abline(v=beta0, col="red")
hist(beta.hat[,2], main="Slope", xlab="Estimate", ylab="Sampling Distribution")
abline(v=beta1, col="red")
```

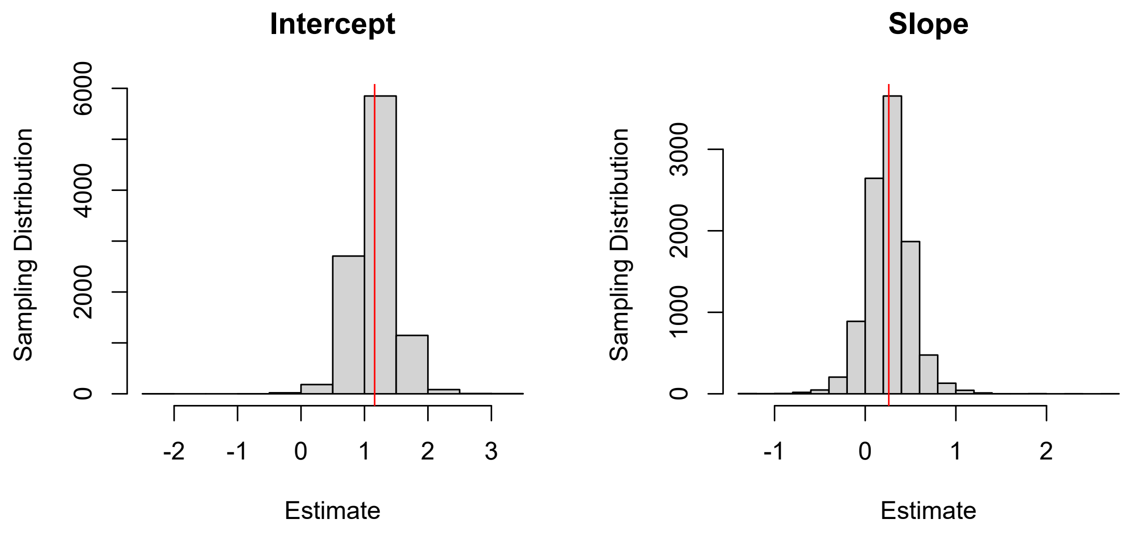


FIGURE 18.2 Sampling distribution for the regression parameter estimators when sample sizes for each lake are random.

There appears to be little bias when estimating β_0 and β_1 despite having incorrectly assumed the observations were independent, which is also confirmed, below:

```
# Estimate of Bias: Intercept
mean(beta.hat[,1])- beta0
```

```
## [1] -0.00434286
```

```
# Estimate of Bias: Slope
mean(beta.hat[,2])- beta1
```

```
## [1] 0.00175792
```

Next, we consider the impact that repeated measures have on our estimates of uncertainty when assuming observations are independent. If our estimators of the standard errors of $\hat{\beta}_0$ and $\hat{\beta}_1$ are unbiased, then we should expect the average estimated standard errors, $E[\widehat{SE}(\hat{\beta}_0)]$ and $E[\widehat{SE}(\hat{\beta}_1)]$, to be close to the standard deviation of $\hat{\beta}$ (remember, these SEs estimate the standard deviation of our sampling distribution!).

```
# Estimate of Bias: SE(intercept)
mean(standard.error.hat[,1]) - sd(beta.hat[,1])
```

```
## [1] -0.231249
```

```
# Estimate of Bias: SE(slope)
mean(standard.error.hat[,2]) - sd(beta.hat[,2])
```

```
## [1] -0.1746312
```

We see in both cases that the $E[\widehat{SE}(\hat{\beta}_i)] < sd(\hat{\beta}_i)$, suggesting our estimates of uncertainty are too small if we ignore the repeated measures and assume our observations are independent.

18.2.2 What if sample sizes are not completely random?

What if the sample sizes associated with each lake are not random? Maybe we are more likely to sample lakes where fish tend to have high levels of Se (more so than would be predicted by the lake's Se concentration). I.e., what if the sample sizes were correlated with the τ_i? Let's simulate data under this assumption (but for a sample of 100 lakes) and again fit the simple linear regression model. In addition, we will fit the data-generating model using the `lmer` function in the `lme4` package (Bates et al. 2015) (this is a mixed-effects model, which will be the subject of the remainder of this chapter):

```
# Set.seed
set.seed(1024)

# Matrix to hold results
beta.hat2 <- matrix(NA, nsims, 2)
beta.hat2.lme4 <- matrix(NA, nsims, 2)

# Simulation loop
for(i in 1:nsims){
  # pick a sample of 100 lakes
  lakeids <- sample(1:nlakes, 100)

  # determine random sample size for each lake
  # generate using negative binomial with mean = mu*exp(tau[i])
  # overdispersion parameter = theta = 1.15
  ni <- rnbinom(100, mu = exp(log(mu)+tau[lakeids]), size = theta)
  log_se_fish <- beta0 + beta1*rep(log_se_water[lakeids], ni) +
    rep(tau[lakeids], ni) + rnorm(sum(ni), 0, sigma_epsilon)
  dat <- data.frame(id = as.factor(rep(lakeids, ni)),
                    log_se_water = rep(log_se_water[lakeids], ni),
                    log_se_fish)
  beta.hat2[i,] <- coef(lm(log_se_fish~log_se_water, data=dat))
  beta.hat2.lme4[i,] <- fixef(lmer(log_se_fish~log_se_water + (1|id), data=dat))
}
```

We again look at the sampling distributions with the true parameters used to generate the data highlighted in red (the top row of panels correspond to the simple linear regression model and the bottom panels to the mixed-effects model used to generate the data):

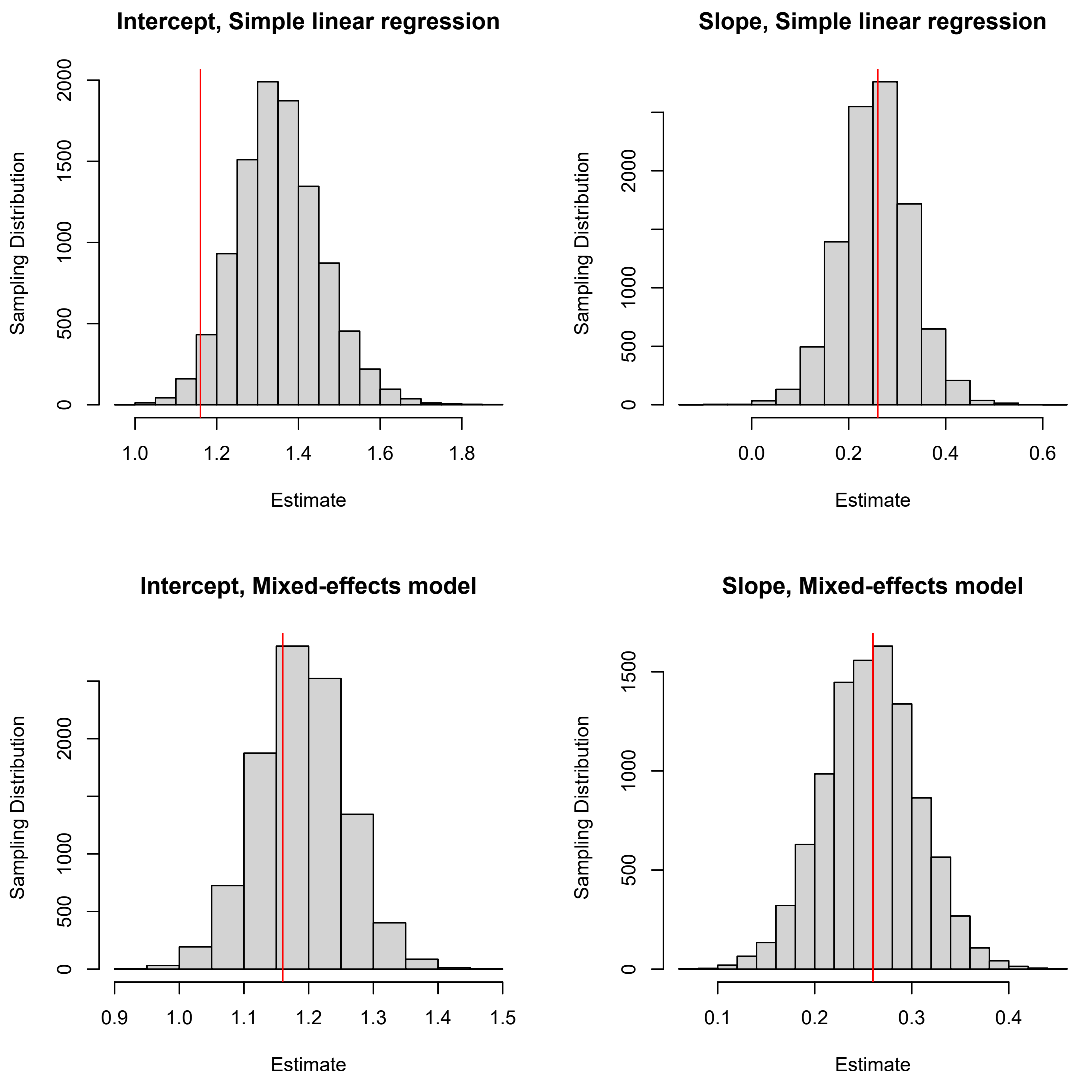


FIGURE 18.3 Sampling distribution for the regression parameter estimators when sample sizes for each lake are correlated with within-lake factors associated with Se concentrations in fish. Top row corresponds to estimators using linear regression assuming independent observations. The bottom row are for a mixed-effects model that includes lake-level random intercepts.

We see that when using simple linear regression, our estimator of the intercept is biased. Nonetheless, the estimator of the slope appears to be unbiased. The estimators of both parameters appear to be unbiased when fitting the mixed-effects (i.e., data-generating) model. Also, note that the estimates are less variable when using the mixed-effects model.

```
bias.betas<-data.frame(
  Method = rep(c("Simple Linear Regression", "Data-Generating (Mixed) Model")),
  Intercept = round(c(mean(beta.hat2[,1])- beta0,
                      mean(beta.hat2.lme4[,1])- beta0), 3),
  Slope = round(c(mean(beta.hat2[,2])- beta1,
                  mean(beta.hat2.lme4[,2])- beta1),3))
bias.betas %>% kable(., caption = "Bias associated with regression parameter estimators.",
                     booktabs = TRUE)%>%
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

TABLE 18.1 Bias associated with regression parameter estimators.

Method	Intercept	Slope
Simple Linear Regression	0.193	-0.003
Data-Generating (Mixed) Model	0.029	-0.003

18.3 Optional: Mixed-effect models versus simple alternatives

In the last section, we saw that if sample sizes are random, then our estimates of regression parameters will be unbiased, but estimates of their uncertainty will be too small. Thus, one simple strategy for analyzing repeated measures data in cases where sample sizes can be argued to be completely random or the data are balanced (i.e., contain equal numbers of observations for each cluster) is to assume independence when estimating regression parameters, but then use a method for estimating uncertainty that considers the cluster as the independent unit of replication. For example, in Section 2, we explored a cluster-level bootstrap when analyzing the `Rikzdat` data (resampling beaches with replacement). Similarly, in Chapter 20 will learn about Generalized Estimating Equations (GEE) that use robust standard errors calculated by considering variation among independent clusters.

Alternatively, we could consider aggregating our data to the lake level before fitting our regression model. Each lake has a single recorded value of log(Se) in the water. We could calculate the average log(Se) among fish for each lake and then fit a regression model with this average as our response variable. This strategy is often reasonable when the predictors of interest do not vary within a cluster (Murtaugh 2007). The code below shows how to accomplish this in R.

```
fishse.avg <- Selake %>% group_by(Lake) %>%
   dplyr::summarize(Log_Water_Se=mean(Log_Water_Se), fish.avg.se=mean(Log_fish_Se))
fit.avg <- lm(fish.avg.se ~ Log_Water_Se, data=fishse.avg)
```

Lastly, although we have not yet introduced mixed-effect models, we can fit the exact model that was used to generate the data using the following code:

```
fit.mixed <- lmer(Log_fish_Se ~ Log_Water_Se + (1|Lake), data=Selake)
```

Let's compare the estimated intercept and slope from these three approaches: a naive linear regression model, a regression model fit to lake averages, and the mixed-effect model fit using `lmer`, which was the model used to generate the data.

```
modelsummary(list("Naive linear model" = fit.naive,
                  "Linear model fit to averages" = fit.avg,
                  "Mixed-effects model" = fit.mixed),
             gof_omit = ".*", estimate  = "{estimate} ({std.error})", statistic=NULL,
             coef_omit = "SD",
             title="Regression coefficients (SE) from a naive linear regression model
             assuming independence, a regression model fit to lake averages, and a
             random-intercept model.")%>%
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

TABLE 18.2 Regression coefficients (SE) from a naive linear regression model assuming independence, a regression model fit to lake averages, and a random-intercept model.

	Naive linear model	Linear model fit to averages	Mixed-effects model
(Intercept)	1.164 (0.064)	1.131 (0.209)	1.132 (0.209)
Log_Water_Se	0.265 (0.059)	0.367 (0.181)	0.366 (0.179)

We see that the estimated standard errors for β_0 and β_1 are **much** narrower in the naive linear model, which is not a surprise since this approach treats the data as though they were independent (Table 18.2). On the other hand, the estimates of β_0, β_1, and their standard errors are nearly identical when fitting a linear model to the aggregated data or when fitting an appropriate mixed-effects model. An advantage of the former approach is its simplicity – it requires only methods you would likely have seen in an introductory statistics course.

Murtaugh (2007) provided a similar example in which he compared two analyses, both aiming to test for differences in mean zooplankton size between ponds that either contained or did not contain fish. In each pond, he measured the body length of thousands of zooplankters. Clearly, these observations will not be independent (zooplankton from the same lake are likely to be more similar to each other than zooplankton from different lakes, even after accounting for the effects of fish presence-absence). The first analysis, reported in Murtaugh (1989), required a full paragraph to describe:

> I ran a nested analysis of variance (ANOVA) on the lengths of zooplankton in the six ponds—the effect of pond on body size is nested within the effect of predation treatment. Note that, since the study ponds can be thought of as a sample from a larger population of ponds that might have been used, 'pond' is a random effect nested within the fixed effect of fish . . . Because of the complications involved in nested ANOVA's with unequal sample sizes (Sokal and Rohlf 1981), I fixed the number of replicates per pond at the value for the pond with the smallest number of length measurements available . . .; appropriately-sized samples were obtained from the larger sets of measurements by random sampling without replacement.' '

In the second analysis, he calculated pond-level means and then conducted a two-sample t-test. He points out that the two hypotheses tests relied on the exact same set of assumptions and the p-values were identical. The nested analysis of variance will sound more impressive, but the latter approach has the advantage of simplicity – again, requiring only methods learned in an introductory statistics class. Murtaugh (2007) makes a strong argument for the simpler analyses, highlighting that:

Simpler analyses are easier to explain and understand; they clarify what the key units in a study are; they reduce the chances for computational mistakes; and they are more likely to lead to the same conclusions when applied by different analysts to the same data.

On the other hand, there are potential advantages of fitting a mixed-effects model (e.g., we will see that we can study variability in `Log_Se_fish` measurements both among and within lakes). In addition, mixed-effects models will be more powerful when studying predictor-response relationships when the predictor variable varies within a lake. For example, if we also wanted to study how Se concentrations change as fish age, we would be better off studying this using a model that included the age of each fish rather than modeling at the lake level using the average age in each lake as a predictor. Thus, it is important to be able to identify when using a mixed-effects model will be beneficial and when it may be viewed as just *statistical machismo*.[2]

The rest of this chapter will be devoted to mixed-effect models that allow observational units or clusters to have their own parameters. We will learn how to write down a set of equations for mixed models (e.g., eq. (18.2.1)), match R output to the parameters in these equations, explore the assumptions of the model and how to evaluate them, and discover how these models *may or may not* properly account for non-independence resulting from having multiple observations on the same sampling unit or cluster.

18.4 What are random effects, mixed-effect models, and when should they be considered?

The model fit using the `lmer` function allowed each lake to have its own intercept. More generally, we can consider models that allow each cluster to have its own intercept and slope parameters. Actually, we already saw how we could allow for cluster-specific parameters in Chapter 3.

Think-pair-share: Using what we learned in Chapter 3, how could we allow each lake to have its own intercept? And, more generally, how could we allow each lake to have its own slope?

We could have included `Lake` in the model using `lm(Log_fish_Se ~ Log_Water_Se + Lake, data=Selake)`[3]. This would have allowed each lake to have its own intercept. Furthermore, if we had a predictor that varied within a lake (e.g., fish age), we could allow for lake-specific slopes by interacting `Lake` with `age`, e.g., using `lm(Log_fish_Se ~ Log_Water_Se + age + Lake + age:Lake, data=Selake)`. These approaches model among-cluster variability using fixed effects, parameters that we directly estimate when fitting the model. How do these parameters differ from random effects?

The defining characteristic of random effects is that they are assumed to come from a common statistical distribution, usually a Normal distribution. Thus, models that contain random effects add another assumption, namely that the cluster-specific parameters come from a common distribution. Specifically, if we examine eq. (18.2.1), we see the assumption that $\tau_i \sim N(0, \sigma^2_\tau)$.

---

[2]For some interesting viewpoints on this larger topic, see this dynamic ecology blog and counterpoints here and here.

[3]Because `Log_Water_Se` is constant within each lake, we won't be able to estimate both the coefficient for `Log_Water_Se` and separate coefficients for all lakes. If you try to fit this model, you will find that one of the estimated lake-level coefficients is `NA`

The assumption that the cluster-specific parameters follow a common distribution makes the model hierarchical. Note that most models with *random effects* also include *fixed effects*, and are thus referred to as mixed-effects models or mixed models. Thus, we may refer to the model we fit using `lmer` as any of the following:

- a mixed-effects (or mixed model)
- a random-effects model
- a hierarchical or multi-level model
- a random-intercept model.

Mixed-effect models are useful for analyzing data where:

- there are repeated measures on the same sample units (same site, same animal, same lake).
- the data are naturally clustered or hierarchical in nature (e.g., you are collecting data on individuals that live in packs within multiple populations).
- the experimental or sampling design involves replication at multiple levels of hierarchy (e.g. you have a split-plot design with treatments applied to plots and subplots, you have sampled multiple individuals within each of several households).
- you are interested in quantifying variability of a response across different levels of replication (e.g., among and within lakes, among and within sites, etc).
- you want to generalize to a larger population of sample units than just those that you have observed.

These latter two points will hopefully become clearer as we proceed through this Chapter.

18.5 Two-step approach to building a mixed-effects model

It can be useful to consider building hierarchical models in multiple steps. In the first step, we can build models for individual observations within a cluster (referred to as the level-1 model). In the second step, we can model how the cluster-specific parameters from stage 1 vary across clusters (referred to as the level-2 model). We will demonstrate this approach using data from a study investigating tradeoffs between maximum age of trees and their size in mountain pines (*Pinus montana*) in Switzerland (Bigler 2016). The data, originally downloaded from dryad here, record the (maximum) `age` and diameter at breast height (`dbh`) of 160 trees measured at 20 different sites (`site`); I have also appended a variable that records the predominate aspect (i.e., direction of the site's facing slope), `Aspect`. Although a two-step approach to model fitting has some drawbacks, it is a useful pedagogical tool that reinforces the key feature of mixed-effects models; namely, they include cluster-specific parameters that are assumed to arise from a common statistical distribution. A two-step approach can also serve as a useful tool for exploratory data analysis, particularly for data sets that have few clusters but lots of observations per cluster; a good example is wildlife telemetry studies that collect lots of location data on few individuals.

An important consideration when fitting mixed-effects models is whether available predictors vary within a cluster (referred to as level-1 predictors) or they are constant for each cluster (level-2 predictors). In the `pines` data set, `age` is a level-1 predictor, whereas `Aspect` is a level-2 predictor. We can see this by exploring the relationship between `age` and `dbh`, which varies by site (Figure 18.4). We also see that `Aspect` is constant for all observations from the same site.

```
data(pines)
ggplot(pines, aes(age, dbh, col = Aspect)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x) +
  facet_wrap(~ site)
```

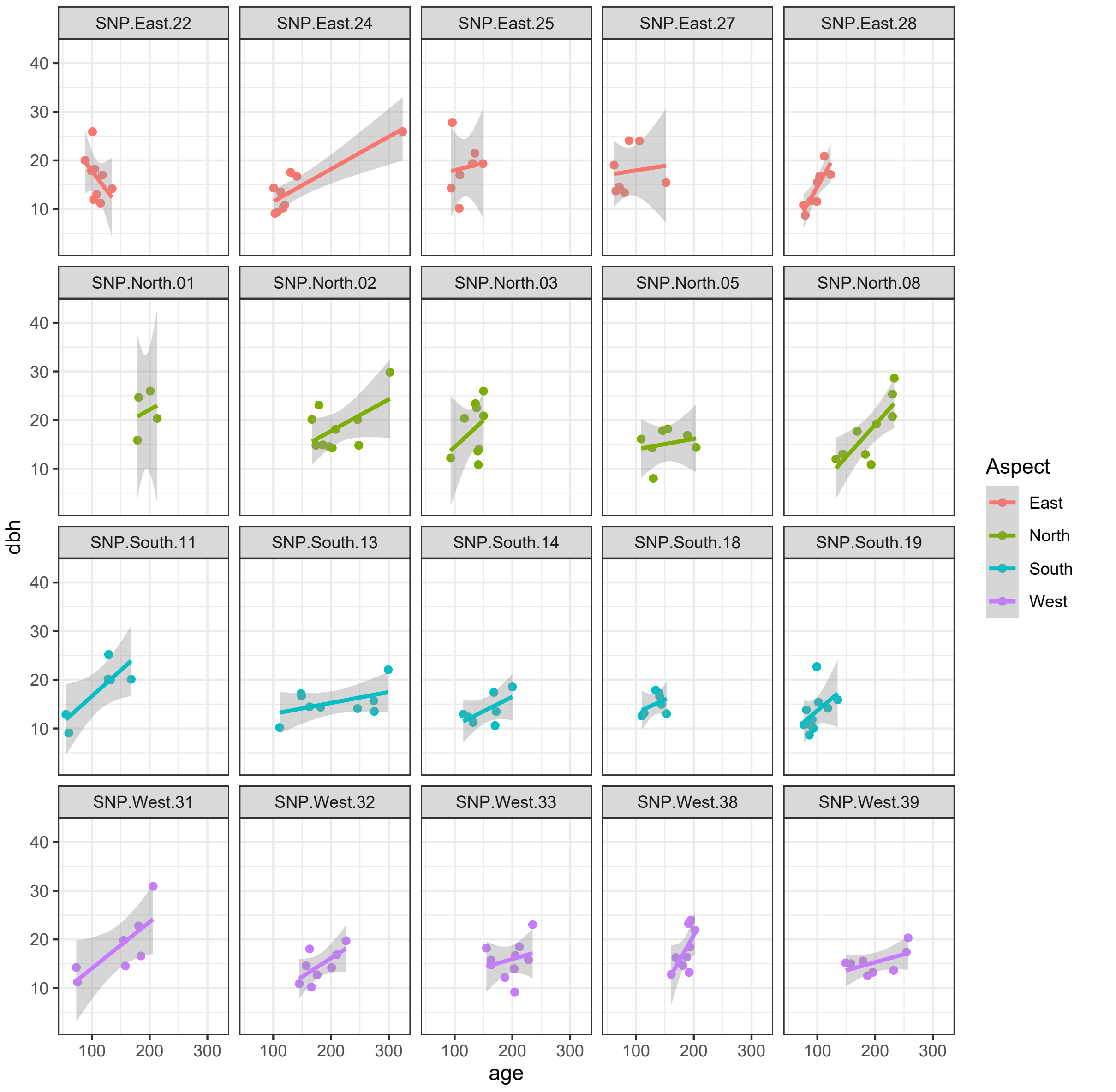


FIGURE 18.4 Diameter at breast height (dbh) versus longevity (age) for 160 pine trees measured at 20 different sites.

18.5.1 Stage 1 (level-1 model)

In the first stage (or step), we build a separate model for each cluster (i.e., `site`). When building this model, we can only consider variables that are *not* constant within a cluster. Thus, we can include `age` but not `Aspect`.

Level 1 model:

$$dbh_{ij} = \beta_{0i} + \beta_{1i} age_{ij} + \epsilon_{ij} \tag{18.1}$$

In this model, each site has its own intercept β_{0i} and slope β_{1i}. We can use a loop to fit this model to each site, and then pull off the coefficients, or we can accomplish the task in a few lines of code using functions in the `purrr` (Henry and Wickham 2020), `tidyr` (Wickham 2021), and `broom` packages (Robinson, Hayes, and Couch 2021). Before fitting models to each site, we will create a centered and scaled version of `age`, which will make it easier to fit mixed-effects models down the road (see Section 18.13.2). The `scale` function creates a new variable, which we name `agec`, by subtracting the mean `age` and diving by the standard deviation of `age` from each observation.

```
pines <- pines %>%
  mutate(agec = scale(age))
```

Below, we provide code for fitting a model to each site using a `for` loop:

```
usite <- unique(pines$site)
nsites <- length(usite)
Beta <- matrix(NA, nsites, 2) # to hold slope and intercept
Aspect <-  matrix(NA, nsites, 1) # to hold aspect (level- covariate)
for(i in 1:nsites){
  dataseti <- subset(pines, site == usite[i])
  LMi <- lm(dbh ~ agec, data = dataseti)
  Beta[i,] <- coef(LMi)
  Aspect[i] <- dataseti$Aspect[1]
}
betadat <- data.frame(site = usite,
                      intercept = Beta[,1],
                      slope = Beta[,2],
                      Aspect = Aspect)
betadat
```

```
##             site intercept    slope Aspect
## 1  SNP.South.18 16.097300 2.849878  South
## 2  SNP.North.03 20.309811 5.823510  North
## 3   SNP.East.27 18.953448 1.000244   East
## 4  SNP.North.01 19.069013 3.497746  North
## 5  SNP.North.02 14.697916 3.469895  North
## 6   SNP.East.25 19.580211 1.604887   East
## 7  SNP.North.05 15.136417 1.158623  North
## 8  SNP.South.11 22.318424 5.666733  South
## 9   SNP.West.31 19.140467 5.061144   West
## 10  SNP.West.33 14.366430 1.821074   West
## 11 SNP.South.13 14.189394 1.189321  South
## 12 SNP.South.19 19.034423  5.336379  South
## 13  SNP.East.24 15.108255  3.547825   East
## 14 SNP.South.14 13.727017  3.160221  South
## 15  SNP.East.28 26.010214 11.332107   East
## 16  SNP.West.39 13.760994  1.759141   West
## 17  SNP.West.38 11.105753 11.127715   West
## 18  SNP.West.32 12.578978  4.060440   West
## 19  SNP.East.22  9.500404 -8.309131   East
## 20 SNP.North.08 12.899520  7.010497  North
```

And, here is an option using functions in the `tidyverse` set of packages:

```
# Nest the data by site, then fit a model and pull off
# the coefficients and associated information
betadata <- pines %>% nest_by(site) %>%
  mutate(mod = list(lm(dbh ~ agec, data = data))) %>%
  dplyr::reframe(tidy(mod))

# Now, merge this information with Aspect using a left join
betadata <- left_join(betadata, unique(pines[,c("site", "Aspect")]))
```

```
## Joining with ‘by = join_by(site)‘
```

```
betadata
```

```
## # A tibble: 40 x 7
##    site        term        estimate std.error statistic     p.value Aspect
##    <chr>       <chr>          <dbl>     <dbl>     <dbl>       <dbl> <chr>
##  1 SNP.East.22 (Intercept)     9.50     5.49      1.73  0.127       East
##  2 SNP.East.22 agec           -8.31     6.20     -1.34  0.222       East
##  3 SNP.East.24 (Intercept)    15.1      0.953    15.9   0.000000960 East
##  4 SNP.East.24 agec            3.55     0.747     4.75  0.00209     East
##  5 SNP.East.25 (Intercept)    19.6      4.81      4.07  0.00962     East
##  6 SNP.East.25 agec            1.60     6.25      0.257 0.807       East
##  7 SNP.East.27 (Intercept)    19.0      4.65      4.08  0.00954     East
##  8 SNP.East.27 agec            1.00     3.50      0.286 0.786       East
##  9 SNP.East.28 (Intercept)    26.0      3.25      8.00  0.000203    East
## 10 SNP.East.28 agec           11.3      3.00      3.78  0.00916     East
## # i 30 more rows
```

18.5.2 Stage 2 (level-2 model)

In the second step, we treat the coefficients from stage 1 as "data", and we model the coefficients as a function of variables that are constant within a cluster. Let's begin by exploring, graphically, how much the coefficients vary across the different sites. We can also see if the coefficients co-vary with our level-2 predictor, `Aspect`.

```
ggplot(betadata, aes(Aspect, estimate)) + geom_point() + facet_wrap(~term)
```

There appears to be quite a bit of site-to-site variability but no strong effect of `Aspect` on the coefficient

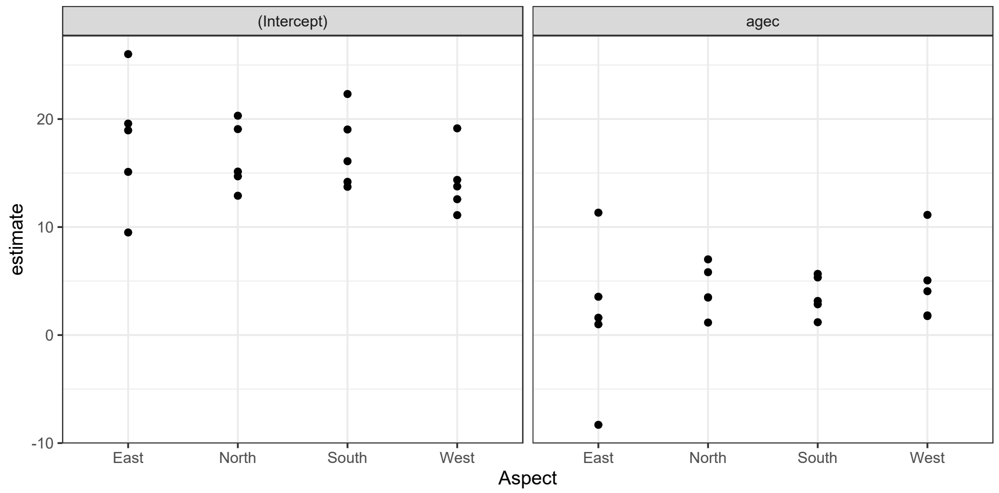


FIGURE 18.5 Distribution of site-specific intercepts and slopes relating ‘age‘ to diameter at breast height.

estimates. We could then fit regression models relating estimates of cluster-specific parameters to our level-2 predictors (e.g., `Aspect`). To facilitate further modeling, we will create a “wide” data set containing separate columns for the estimated intercepts and slopes:

```
betacoefswide <- betadata %>%
  dplyr::select(site, estimate, Aspect, term) %>%
  pivot_wider(names_from = "term", values_from = "estimate") %>%
  rename(intercept = `(Intercept)`, slope = "agec")
head(betacoefswide)
```

```
## # A tibble: 6 x 4
##   site         Aspect intercept slope
##   <chr>        <chr>      <dbl> <dbl>
## 1 SNP.East.22  East        9.50 -8.31
## 2 SNP.East.24  East       15.1   3.55
## 3 SNP.East.25  East       19.6   1.60
## 4 SNP.East.27  East       19.0   1.00
## 5 SNP.East.28  East       26.0  11.3
## 6 SNP.North.01 North      19.1   3.50
```

We then use this data set to explore how our cluster-specific intercepts and slopes vary and co-vary with our level-2 predictor, `Aspect`:

```
summary(lm(intercept ~ Aspect, data = betacoefswide))
```

```
##
## Call:
## lm(formula = intercept ~ Aspect, data = betacoefswide)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.3301 -2.7627 -0.7028  2.1325  8.1797
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  17.8305     1.8569   9.602 4.82e-08 ***
## AspectNorth  -1.4080     2.6260  -0.536    0.599
## AspectSouth  -0.7572     2.6260  -0.288    0.777
## AspectWest   -3.6400     2.6260  -1.386    0.185
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.152 on 16 degrees of freedom
## Multiple R-squared:  0.118,	Adjusted R-squared:  -0.04738
## F-statistic: 0.7135 on 3 and 16 DF,  p-value: 0.5581
```

```
summary(lm(slope ~ Aspect, data = betacoefswide))
```

```
##
## Call:
## lm(formula = slope ~ Aspect, data = betacoefswide)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.1443  -1.2390  -0.5873   1.7001   9.4969
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.835      1.905   0.964    0.350
## AspectNorth    2.357      2.694   0.875    0.395
## AspectSouth    1.805      2.694   0.670    0.512
## AspectWest     2.931      2.694   1.088    0.293
##
## Residual standard error: 4.259 on 16 degrees of freedom
## Multiple R-squared:  0.07676,	Adjusted R-squared:  -0.09635
## F-statistic: 0.4434 on 3 and 16 DF,  p-value: 0.7252
```

Taken together with Figure 18.5, we might conclude that the coefficients do not vary significantly with respect to `Aspect`. Thus, we could consider a level-2 model for the site-specific intercepts and slopes that allow these parameters to vary about their overall mean but not depend on `Aspect`.

Level-2 model:

$$\beta_{0i} = \beta_0 + b_{0i} \tag{18.2}$$
$$\beta_{1i} = \beta_1 + b_{1i} \tag{18.3}$$

In equation (18.3), the β_0 and β_1 represent the mean intercept and slope in the population of sites, respectively, and b_{0i} and b_{1i} represent deviations from these average parameters for `site` i.

18.5.3 Putting things together: Composite equation

We can combine the level-1 and level-2 models, substituting in the equation for the level-2 model (eq. (18.3)) into the equation for the level-1 model (eq. (18.1)) to form what is referred to as the composite equation:

$$dbh_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})age_{ij} + \epsilon_{ij} \tag{18.4}$$

This leads us to what is referred to as a random coefficient or random slopes model in which both the intercepts and slopes are site-specific. Typically, we also assume that the $(b_{0i}, b_{1i}) \sim N(0, \Sigma_b)$, where Σ_b is a 2 x 2 matrix.

Think-Pair-Share: Why is Σ_b a 2 x 2 matrix? What do the diagonal and non-diagonal entries represent? We will discuss the answer to this question when we explore output from fitting this model in R in Section 18.7.

18.6 Random-intercept versus random intercept and slope model

The 2-step approach of the last section led us to consider a model that included site-specific intercepts and slope parameters associated with our level-1 covariate, `agec`.[4] However, in a great many cases, users default to including only random intercepts without consideration of random slopes[5]; this choice often leads to p-values that are too small, confidence intervals that are too narrow, and overconfident conclusions when evaluating the importance of level-1 covariates (Schielzeth and Forstmeier 2009; Muff, Signer, and Fieberg 2020). As noted by Schielzeth and Forstmeier (2009), random slopes should generally be considered for predictors that vary within a cluster, especially when cluster-specific parameters exhibit a high degree of variation (Figure 18.6a), when within cluster variability is low (Figure 18.6b), and when there are lots of observations per cluster (Figure 18.6c). In addition to Schielzeth and Forstmeier (2009), Arnqvist (2020) and Silk, Harrison, and Hodgson (2020) also identify this issue as being one that is neglected by all too many users of mixed-effect models.

18.7 Fitting mixed-effects models in R

There are several R packages that can be used to fit mixed-effects models. In this book we will consider 4 of them:

- `nlme` (which only fits linear mixed effects models) (Jose Pinheiro et al. 2021)
- `lme4` (Bates et al. 2015)
- `glmmTMB` (Brooks et al. 2017)
- `GLMMadaptive` (Rizopoulos 2021)

We have already seen the `nlme` package in Chapter 5, which we used to fit linear models that relaxed the constant variance assumption using generalized least squares. The `nlme` package is older and less commonly

[4]In Section 18.13, we will discuss other approaches that are often recommended for choosing an appropriate mixed model.

[5]It is quite common for users to just say that they fit a "random effect" for variable x; almost always, this means they included a random intercept associated with a categorical variable, x

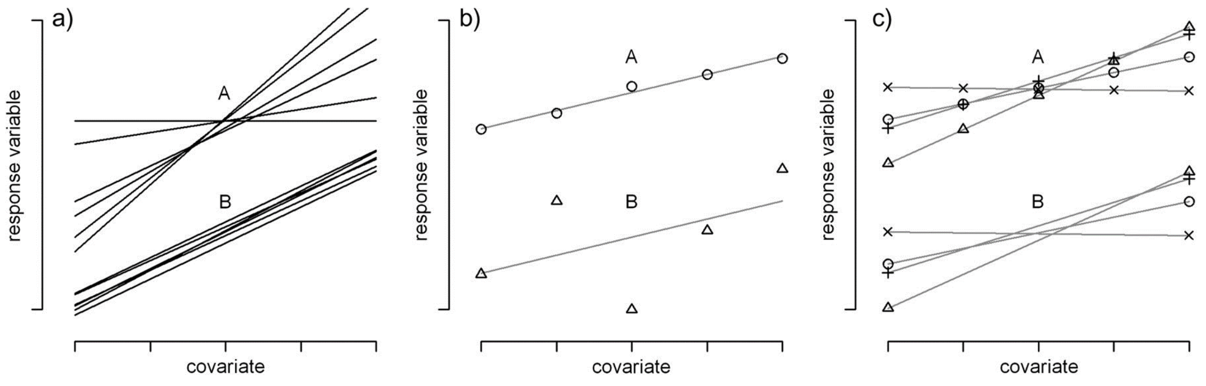


FIGURE 18.6 Figure from Schielzeth and Forstmeier (2009). CC By NC 2.0. Schematic illustrations of more (A) and less (B) problematic cases for the estimation of fixed-effect covariates in random-intercept models. (a) Regression lines for several individuals with high (A) and low (B) between-individual variation in slopes (σ_b)). (b) Two individual regression slopes with low (A) and high (B) scatter around the regression line (σ_r). (c) Regression lines with (A) many and (B) few measurements per individual (independent of the number of levels of the covariate).

used to fit mixed-effect models than other packages (e.g., `lme4` and `glmmTMB`). The primary advantage of the `nlme` package is that it can be used to fit mixed models that relax the assumption that within-cluster errors, ϵ_{ij}, are independent and have constant variance (i.e., one can account for temporal or spatial autocorrelation and non-homogeneous variance). Yet, the `nlme` package does not include methods for fitting generalized linear mixed effects models (Chapter 19), and it is slowly becoming outdated. The `lme4` package is arguably the most popular package for fitting mixed-effects models, and it will be the focus of this chapter. The `glmmTMB` package is newer on the scene and can be quicker when fitting generalized linear mixed effects models to response data that are not Normally distributed (Brooks et al. 2017). It uses syntax that is similar to `lme4` but allows for a wider range of models (e.g., one can fit models that account for zero-inflation, and `glmmTMB` can also fit mixed-effects models using a negative binomial distribution). Thus, we will explore `glmmTMB` when we get to Chapter 19. Lastly, the `GLMMadaptive` package has certain advantages when fitting generalized linear mixed effects models, particularly when it comes to estimating population-level response patterns. We will explore this package in Chapter 19. For now, we will focus on implementations using the `lmer` function in the `lme4` package.

18.7.1 Random intercept model

Let's first fit a model containing a random intercept for each `site` by adding `(1 | site)` to the model formula (recall, `1` is used to refer to the intercept).

```
lmer.ri <- lmer(dbh ~ agec + (1 | site), data = pines)
summary(lmer.ri)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: dbh ~ agec + (1 | site)
##    Data: pines
##
## REML criterion at convergence: 918.8
##
## Scaled residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -1.9218 -0.7456 -0.0585  0.6020  2.8824
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  site     (Intercept)  4.216   2.053
##  Residual             16.147   4.018
## Number of obs: 160, groups:  site, 20
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  16.4743     0.5608  29.375
## agec          2.4305     0.4100   5.928
##
## Correlation of Fixed Effects:
##      (Intr)
## agec 0.012
```

Think-Pair-Share: try to describe the model using a set of equations and link the parameters in the equations to the output. We will leave this exercise to the reader but provide this information for the random intercept and slope model below.

18.7.2 Random intercepts and slopes model

Let's next fit a model with random intercpets and random slopes for `agec`. To do so, we add `(1 + agec | site)` to the model formula, letting `lmer` know that we want each `site` to have its own intercept and slope for `agec`.

```
lmer.rc <- lmer(dbh ~ agec + (1 + agec | site), data = pines)
summary(lmer.rc)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: dbh ~ agec + (1 + agec | site)
##    Data: pines
##
## REML criterion at convergence: 917.4
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.90307 -0.72242 -0.08194  0.58100  2.93686
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  site     (Intercept)  3.740   1.934
##           agec         1.122   1.059    0.25
##  Residual             15.631   3.954
## Number of obs: 160, groups:  site, 20
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  16.5258     0.5601  29.506
## agec          2.4502     0.4934   4.966
##
## Correlation of Fixed Effects:
##      (Intr)
## agec 0.157
```

Note, we could have omitted the `1`, as `lmer` by default includes a random intercept whenever a random slope is included. Thus, `lmer(dbh ~ agec + (age | site), data = pines)` fits the same model as the one above.

Let's write down this model using a set of equations and find the estimated parameters in the R output:

$$dbh_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})age_{ij} + \epsilon_{ij} \sim N(0, \sigma^2_\epsilon)$$

$$\begin{bmatrix} b_{0i} \\ b_{1i} \end{bmatrix} \sim N(\mu, \Sigma_b), \text{ with} \tag{18.5}$$

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \Sigma_b = \begin{bmatrix} \sigma^2_{b_0} & Cov(b_0, b_1) \\ Cov(b_0, b_1) & \sigma^2_{b1} \end{bmatrix} \tag{18.6}$$

$$\epsilon_{ij}$$

$\sigma^2_{b_0}$ is the variance of b_{0i}, $\sigma^2_{b_1}$ is the variance of b_{1i}, and $Cov(b_0, b_1) = E[b_{0i}b_{1i}] - E[b_{0i}]E[b_{1i}]$is the covariance of b_{0i} and b_{1i}[6]. If you are unfamiliar with covariances, they quantify the degree to which two random variables co-vary (i.e., whether high values of b_{0i} are associated with low/high values of b_{1i}). We can also calculate the covariance from the correlation of two random variables and their standard deviations:

$$Cor(b_0, b_1) = \frac{Cov(b_0, b_1)}{\sigma_{b_0}\sigma_{b_1}} \tag{18.7}$$

$$\implies Cov(b_0, b_1) = Cor(b_0, b_1)\sigma_{b_0}\sigma_{b_1} \tag{18.8}$$

The estimates of β_0 and β_1 are under the label **`Fixed effects`** and are equal to 16.52 and 2.45, respectively. The estimates of $\sigma^2_{b_0}$, $\sigma^2_{b_1}$, and σ^2_ϵ are under the label **`Random effects`** and are equal to 3.74, 1.12, and 15.63, respectively. Note that the **`Std.dev`** column next to the **`Variance`** column just translates these variances to standard deviations (i.e., $sd(b_1) = \sqrt{\sigma^2_{b_1}} = 1.059$). The R output from the **`summary`** function provides $Cor(b_0, b_1)$ rather than $Cov(b_0, b_1)$. We can calculate $Cov(b_0, b_1)$ using eq. (18.8), $Cov(b_0, b_1) = (0.25)(1.934)(1.059) = 0.512$.

This is a good opportunity to highlight the difference between *variability* and *uncertainty*. We describe uncertainty using SEs and confidence intervals. For example, the SEs for the fixed-effects parameters (**`Std. Error`** column) quantify the extent to which our estimates of β_0 and β_1 would vary across repeated random trials[7]. Our random intercept and slope model also contains 4 variance/covariance parameters that describe *variability* in the response variable (within and among sites): $\sigma^2_{b_0}$, $\sigma^2_{b_1}$, $Cov(b_0, b_1)$, and σ^2_ϵ. It is important to understand that these parameters do **not** describe uncertainty in the estimated parameters like the SEs do for β_0 and β_1!

Think-Pair-Share: do $\hat{\sigma}^2_{b_0}$, $\hat{\sigma}^2_{b_0}$, $\widehat{cov}(b_0, b_1)$, and $\hat{\sigma}^2_\epsilon$ have sampling distributions?

You bet they do! We estimate these parameters, and therefore, they will have a sampling distribution. We are not given any information here about the uncertainty associated with these parameters, and it turns out that their sampling distribution is usually right skewed. Thus, Normal-based confidence intervals usually

[6]In general, $Cov(X, Y) = E[XY] - E[X]E[Y]$.

[7]Again, a random trial consists of collecting a data set of the same size as this one, generated in the same way, assuming all assumptions of the model are met, and then analyzed the resulting data using the same mixed-effects model

do not work well, and SEs do not provide an adequate measure of uncertainty. We can, however, calculate confidence intervals using the profile-likelihood approach introduced in Section 10.10, which is what the `confint` function does by default (see below):

```
confint(lmer.rc)
```

```
##                  2.5 %    97.5 %
## .sig01       0.4441978  3.247535
## .sig02      -1.0000000  1.000000
## .sig03       0.0000000  2.293833
## .sigma       3.5168693  4.498968
## (Intercept) 15.3979835 17.754390
## agec         1.2692533  3.529877
```

R returns confidence intervals for σ_{b0} (`.sigma01`), $Cor(b_{0i}, b_{1i})$ (`.sig02`), σ_{b1} (`.sig03`), and $\sigma_{epsilon}$ (`.sigma`). We see that the confidence intervals for the standard deviations are asymmetric and the confidence interval for $Cor(b_{0i}, b_{1i})$ spans the entire range from -1 to 1 (i.e., we do not estimate this parameter well). We can verify by adding the argument `oldNames = FALSE)` to the `confint` function.

```
confint(lmer.rc, oldNames=FALSE)
```

```
##                                   2.5 %    97.5 %
## sd_(Intercept)|site           0.4441978  3.247535
## cor_agec.(Intercept)|site    -1.0000000  1.000000
## sd_agec|site                  0.0000000  2.293833
## sigma                         3.5168693  4.498968
## (Intercept)                  15.3979835 17.754390
## agec                          1.2692533  3.529877
```

Lastly, note that $Var(\beta_0 + b_{0i}) = Var(b_{0i}) = \sigma^2_{b_0}$ because β_0 is a constant. Similarly $Var(\beta_1 + b_{1i}) = Var(b_{1i}) = \sigma^2_{b_1}$. Thus, we can think of these variance parameters as describing the variance of site-specific deviations from the mean intercept and slope or variability of the site-specific intercepts and slopes themselves.

18.8 Site-specific parameters (BLUPs)

When looking at the output in the last section, you might have wondered why R does not provide estimates of the site-specific parameters – i.e., the intercepts $(\beta_0 + b_{0i})$ and slopes $(\beta_1 + b_{1i})$ for each site. The deviations from the average intercept and slope, b_{0i} and b_{1i}, are also missing in action. What gives?

In this section, we will learn how to generate "estimates" of site-specific parameters from the mixed-effects model. However, in a frequentist framework, we will actually refer to the estimated quantities as "predictions", or more specifically, Best Linear Unbiased Predictions (BLUPs) in the context of linear mixed effects models. This distinction arises in frequentist inference because the b_{0i}s and b_{1i}s are considered to be random variables rather than parameters with fixed but unknown values. If this seems confusing, you are not alone. This is another situation where life just seems easier if you are a Bayesian, since all parameters are treated as random variables and assigned probability distributions. There is nothing philosophically different about random effects if you are a Bayesian, other than these parameters will all be assigned a common distribution.

We will then need to specify priors for parameters in that common distribution. We will see how to implement linear mixed effects models in JAGS in Section 18.16.

We can generate the BLUPs for b_{0i} and b_{1i} in R using the `ranef` function in the `lme4` package

```
ranef(lmer.rc)
```

```
## $site
##               (Intercept)         agec
## SNP.East.22    1.0581009 -0.539278703
## SNP.East.24   -0.9713254  0.511836750
## SNP.East.25    2.0985073 -0.195316866
## SNP.East.27    1.9471416 -0.659731780
## SNP.East.28    0.4036588  0.341754637
## SNP.North.01   1.5173623  0.510452398
## SNP.North.02  -0.5543028  0.087613361
## SNP.North.03   1.8144629  0.262796638
## SNP.North.05  -0.8777090 -0.282413026
## SNP.North.08  -0.5255991  0.754697359
## SNP.South.11   2.2017430  0.493258625
## SNP.South.13  -1.9043070 -0.960482225
## SNP.South.14  -1.7227241 -0.131886984
## SNP.South.18  -0.3500402  0.005026152
## SNP.South.19  -0.1677626  0.242216046
## SNP.West.31    1.6388877  0.882281861
## SNP.West.32   -1.9592144 -0.291839131
## SNP.West.33   -1.6116716 -0.556945196
## SNP.West.38   -0.1456621  0.214895844
## SNP.West.39   -1.8895462 -0.688935759
##
## with conditional variances for "site"
```

If we want an estimate of uncertainty to accompany the BLUPs, we can use the `tidy` function in the `broom.mixed` package (B. Bolker and Robinson 2021):

```
broom.mixed::tidy(lmer.rc, effects="ran_vals")
```

```
## # A tibble: 40 x 6
##    effect   group level        term        estimate std.error
##    <chr>    <chr> <chr>        <chr>          <dbl>     <dbl>
##  1 ran_vals site  SNP.East.22  (Intercept)    1.06       1.31
##  2 ran_vals site  SNP.East.24  (Intercept)   -0.971      1.10
##  3 ran_vals site  SNP.East.25  (Intercept)    2.10       1.32
##  4 ran_vals site  SNP.East.27  (Intercept)    1.95       1.45
##  5 ran_vals site  SNP.East.28  (Intercept)    0.404      1.40
##  6 ran_vals site  SNP.North.01 (Intercept)    1.52       1.36
##  7 ran_vals site  SNP.North.02 (Intercept)   -0.554      1.14
##  8 ran_vals site  SNP.North.03 (Intercept)    1.81       1.15
##  9 ran_vals site  SNP.North.05 (Intercept)   -0.878      1.18
## 10 ran_vals site  SNP.North.08 (Intercept)   -0.526      1.11
## # i 30 more rows
```

We can also easily plot the BLUPs for b_{0i} and b_{1i} using the `plot_model` function in the `sjPlot` library (Lüdecke 2021):

```
plot_model(lmer.rc, type="re")
```

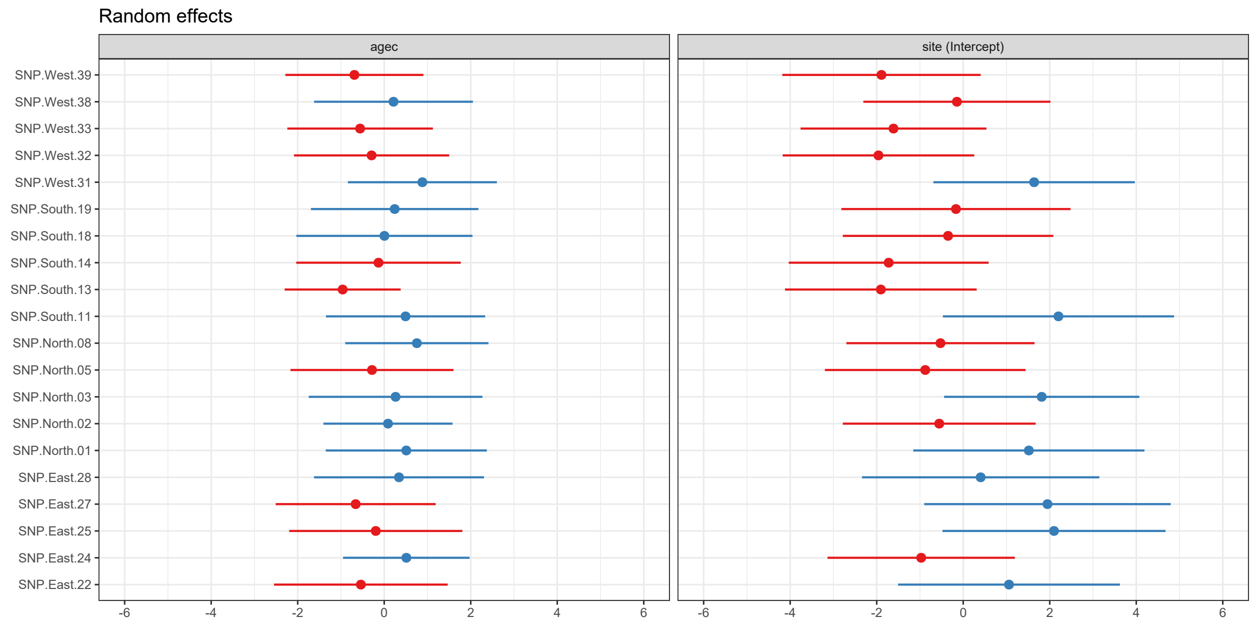


FIGURE 18.7 BLUPs for b_{0i} and b_{1i}, along with their uncertainty, plotted using the `plot_model` function in the `sjPlot` package.

Lastly, we can calculate predictions of the cluster-specific parameters by adding $\hat{\beta}_0$ and $\hat{\beta}_1$ to the BLUPs for b_{0i} and b_{1i}. These estimates can also be extracted directly using the `coef` function:

```
coef(lmer.rc)
```

```
## $site
##               (Intercept)     agec
## SNP.East.22      17.58393 1.910931
## SNP.East.24      15.55450 2.962047
## SNP.East.25      18.62434 2.254893
## SNP.East.27      18.47297 1.790478
## SNP.East.28      16.92949 2.791965
## SNP.North.01     18.04319 2.960662
## SNP.North.02     15.97153 2.537823
## SNP.North.03     18.34029 2.713007
## SNP.North.05     15.64812 2.167797
## SNP.North.08     16.00023 3.204907
## SNP.South.11     18.72757 2.943469
## SNP.South.13     14.62152 1.489728
## SNP.South.14     14.80310 2.318323
## SNP.South.18     16.17579 2.455236
## SNP.South.19     16.35807 2.692426
## SNP.West.31     18.16472 3.332492
## SNP.West.32     14.56661 2.158371
## SNP.West.33     14.91416 1.893265
## SNP.West.38     16.38017 2.665106
## SNP.West.39     14.63628 1.761274
##
## attr(,"class")
## [1] "coef.mer"
```

Note, however, that it is not straightforward to calculate uncertainty associated with the cluster-specific intercepts ($\beta_0 + b_{0i}$) and slopes ($\beta_1 + b_{1i}$) as this would require consideration of uncertainty in both $\hat{\beta}_0, \hat{\beta}_1$ and $\hat{b}_{0i}, \hat{b}_{1i}$, including how these estimates and predictions, respectively, co-vary (see this link on Ben Bolker's GLMM FAQ page).

18.8.1 Digging deeper, how are the BLUPS calculated?

To understand how how BLUPS are calculated from a fitted mixed-effects model, it is helpful to write the model down in terms of the *conditional distribution* of Y given the random effects (i.e., $f_{Y|b}(y)$) and the distribution of the random effects, $f_b(b)$. This way of writing down the model also makes it easy to generalize to other types of probability distributions (i.e., generalized linear mixed effect models), which we will see in Chapter 19.

We can describe our random intercept and slope model from the last section as:

$$\begin{aligned} dbh_{ij}|b_{0i}, b_{1i} &\sim N(\mu_i, \sigma^2_\epsilon) \qquad (18.9)\\ \mu_{ij} &= (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})age_{ij} \\ \begin{bmatrix} b_{0i} \\ b_{1i} \end{bmatrix} &\sim N(0, \Sigma_b), \text{ with} \qquad (18.10)\\ \Sigma_b &= \begin{bmatrix} \sigma^2_{b_0} & Cov(b_0, b_1) \\ Cov(b_0, b_1) & \sigma^2_{b1} \end{bmatrix} \qquad (18.11) \end{aligned}$$

And, more generally, we can write down any linear mixed effects model using:

$$\begin{aligned} Y_{ij} \mid b &\sim N(\mu_i, \sigma^2_\epsilon) \\ \mu_{ij} &= X\beta + Zb \qquad (18.12)\\ b &\sim N(0, \Sigma_b) \end{aligned}$$

Here, X and Z are design matricies associated with the fixed and random effects, respectively. For a random-intercept model, Z would contain a column of 1s. For our random intercept and slope model, Z would also contain a column with the values of `agec`.

The BLUPs, which are returned by the `ranef` function, are estimated by maximizing:

$$f_{b|Y}(b) = \frac{f_{Y|b}(y) f_b(b)}{f_Y(y)} \qquad (18.13)$$

where $f_{b|Y}(b)$ is the conditional distribution of the random effects given the observed data, $f_b(b)$ is the distribution of the random effects (given by eq. (18.10)), and $f_{Y|b}(y)$ is the conditional distribution of $Y|b$ (given by eq. (18.9)). The BLUPs are often referred to as conditional modes and have a Bayesian flavor to them. To calculate the BLUPs, we have to substitute in the estimated fixed effects and variance parameters into our equations for $f_{Y|b}(y)$ and $f_b(b)$, leading to what is sometimes referred to as an Empirical Bayes estimator.

18.9 Fixed effects versus random effects and shrinkage

Earlier, I hinted at the fact that we could estimate site-specific intercepts and slopes using a linear model without random effects. Let's go ahead and fit a fixed-effects model with site-specific intercepts and slopes and compare our results to those from the mixed model:

```
lm.fc <- lm(dbh ~ site + agec +  site : agec, data=pines)
summary(lm.fc)
```

```
##
## Call:
## lm(formula = dbh ~ site + agec + site:agec, data = pines)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.1059 -2.3715 -0.0893  2.0420  9.9533
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)               9.500      4.879   1.947  0.05383 .
## siteSNP.East.24           5.608      5.059   1.109  0.26985
## siteSNP.East.25          10.080      5.791   1.741  0.08430 .
## siteSNP.East.27           9.453      6.058   1.561  0.12127
## siteSNP.East.28          16.510      7.246   2.279  0.02446 *
## siteSNP.North.01          9.569      7.653   1.250  0.21361
## siteSNP.North.02          5.198      5.340   0.973  0.33233
## siteSNP.North.03         10.809      5.262   2.054  0.04212 *
## siteSNP.North.05          5.636      5.100   1.105  0.27137
## siteSNP.North.08          3.399      5.238   0.649  0.51762
## siteSNP.South.11         12.818      5.386   2.380  0.01890 *
## siteSNP.South.13          4.689      5.159   0.909  0.36524
## siteSNP.South.14          4.227      5.100   0.829  0.40889
## siteSNP.South.18          6.597      5.472   1.206  0.23038
## siteSNP.South.19          9.534      6.542   1.457  0.14760
## siteSNP.West.31           9.640      5.103   1.889  0.06128 .
## siteSNP.West.32           3.079      5.265   0.585  0.55980
## siteSNP.West.33           4.866      5.412   0.899  0.37040
## siteSNP.West.38           1.605      6.096   0.263  0.79273
## siteSNP.West.39           4.261      5.354   0.796  0.42775
## agec                     -8.309      5.500  -1.511  0.13351
## siteSNP.East.24:agec     11.857      5.600   2.117  0.03629 *
## siteSNP.East.25:agec      9.914      6.832   1.451  0.14933
## siteSNP.East.27:agec      9.309      6.129   1.519  0.13142
## siteSNP.East.28:agec    19.641      7.393   2.657  0.00897 **
## siteSNP.North.01:agec   11.807      9.197   1.284  0.20170
## siteSNP.North.02:agec   11.779      5.741   2.052  0.04238 *
## siteSNP.North.03:agec   14.133      6.835   2.068  0.04081 *
## siteSNP.North.05:agec    9.468      6.039   1.568  0.11955
## siteSNP.North.08:agec   15.320      5.845   2.621  0.00990 **
## siteSNP.South.11:agec   13.976      5.882   2.376  0.01909 *
## siteSNP.South.13:agec    9.498      5.605   1.695  0.09272 .
## siteSNP.South.14:agec   11.469      6.147   1.866  0.06450 .
## siteSNP.South.18:agec   11.159      7.677   1.454  0.14866
## siteSNP.South.19:agec   13.646      6.767   2.016  0.04600 *
## siteSNP.West.31:agec    13.370      5.730   2.333  0.02129 *
## siteSNP.West.32:agec    12.370      6.156   2.009  0.04674 *
## siteSNP.West.33:agec    10.130      6.046   1.676  0.09641 .
## siteSNP.West.38:agec    19.437      7.852   2.475  0.01470 *
## siteSNP.West.39:agec    10.068      5.817   1.731  0.08607 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.93 on 120 degrees of freedom
## Multiple R-squared:  0.4727, Adjusted R-squared:  0.3013
## F-statistic: 2.758 on 39 and 120 DF,  p-value: 1.301e-05
```

This model gives us direct estimates of intercepts and slopes, and estimates of their uncertainties, for each site. If we are predominately interested in these particular sites, then this model is much more straightforward to work with (e.g., we do not have to try to understand what BLUPs are!). On the other hand, this model requires fitting a lot of parameters (40 in total) and it does not allow us to generalize to other sites that we may not have sampled. In addition, we are unable to consider level-2 predictors in this model since they will be confounded with the site-specific intercepts (to see this, we can try adding `Aspect` to the model, which will result in `NA` for several of the coefficient estimates):

```r
lm.fc2 <- lm(dbh ~ site + agec +  site:agec + Aspect, data=pines)
coef(lm.fc2)
```

```
##           (Intercept)       siteSNP.East.24       siteSNP.East.25
##              9.500404              5.607851             10.079807
##       siteSNP.East.27       siteSNP.East.28      siteSNP.North.01
##              9.453044             16.509810              9.568609
##      siteSNP.North.02      siteSNP.North.03      siteSNP.North.05
##              5.197512             10.809407              5.636013
##      siteSNP.North.08      siteSNP.South.11      siteSNP.South.13
##              3.399116             12.818020              4.688990
##      siteSNP.South.14      siteSNP.South.18      siteSNP.South.19
##              4.226613              6.596896              9.534019
##       siteSNP.West.31       siteSNP.West.32       siteSNP.West.33
##              9.640063              3.078574              4.866026
##       siteSNP.West.38       siteSNP.West.39                  agec
##              1.605349              4.260590             -8.309131
##           AspectNorth           AspectSouth            AspectWest
##                    NA                    NA                    NA
##  siteSNP.East.24:agec  siteSNP.East.25:agec  siteSNP.East.27:agec
##            11.856956              9.914018              9.309375
##  siteSNP.East.28:agec siteSNP.North.01:agec siteSNP.North.02:agec
##            19.641238             11.806878             11.779026
## siteSNP.North.03:agec siteSNP.North.05:agec siteSNP.North.08:agec
##            14.132642              9.467754             15.319628
## siteSNP.South.11:agec siteSNP.South.13:agec siteSNP.South.14:agec
##            13.975865              9.498452             11.469352
## siteSNP.South.18:agec siteSNP.South.19:agec  siteSNP.West.31:agec
##            11.159010             13.645510             13.370275
##  siteSNP.West.32:agec  siteSNP.West.33:agec  siteSNP.West.38:agec
##            12.369572             10.130205             19.436846
##  siteSNP.West.39:agec
##            10.068272
```

Otherwise, how do our estimates of site-specific parameters from the fixed-effect model compare to the BLUPs from the mixed-effect model? Let's plot them together to see! To facilitate comparison, we will refit our fixed effects model using means coding. Then, we create a data set with the coefficients from both models for plotting.

```
lmmeans.fc <- lm(dbh ~ site + agec +  site : agec - 1 - agec, data=pines)
coef.fixed <- coef(lmmeans.fc)
coef.random <- coef(lmer.rc)$site
coefdat <- data.frame(intercepts = c(coef.fixed[1:20], coef.random[,1]),
                      slopes = c(coef.fixed[21:40], coef.random[,2]),
                      method = rep(c("Fixed", "Random"), each=20),
                      site = rep(names(coef.fixed)[1:20],2))
```

Figure 18.8 compares the estimates of site-specific intercept and slope parameters from the fixed-effects and mixed-effects models. The estimated mean intercept ($\hat{\beta}_0$) and slope ($\hat{\beta}_1$) in the population of sites is also depicted by a black horizontal line.

```
p1 <- ggplot(coefdat, aes(site, intercepts, col=method)) +
        geom_point(size=2)+ ggtitle("Intercepts")+
        theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
        geom_hline(yintercept=summary(lmer.rc)$coef[1,1])
p2 <- ggplot(coefdat, aes(site, slopes, col=method)) +
        geom_point(size=2)+ ggtitle("Slopes")+
        theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
        geom_hline(yintercept=summary(lmer.rc)$coef[2,1])

ggpubr::ggarrange(p1, p2, ncol=2, common.legend=TRUE, legend="bottom")
```

Alternatively, we could compare the two sets of coefficients by plotting the (intercept, slope) pairs and connecting them for the two methods (Figure 18.9).

```
ggplot(coefdat, aes(intercepts, slopes, group=site, col=method)) +
        geom_point(size=2) + geom_path()
```

We see that the coefficients for the random effects are *shrunk* back towards the estimated mean coefficients in the population ($\hat{\beta}_0$ and $\hat{\beta}_1$). The degree of shrinkage depends on the amount of information available for

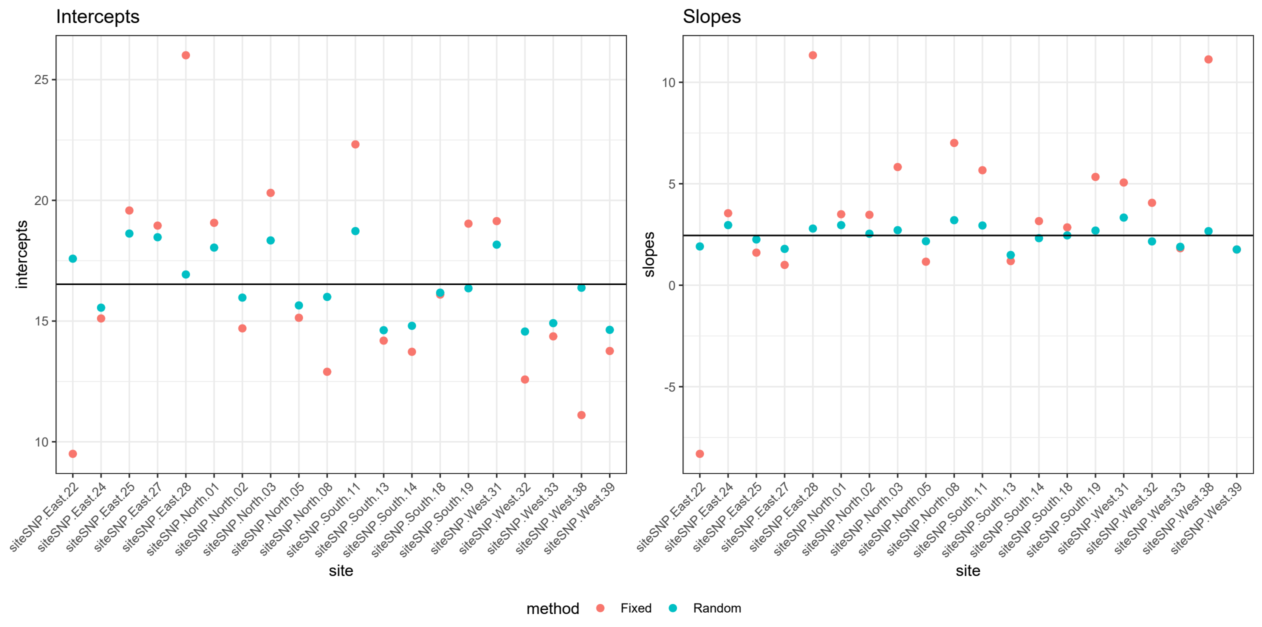


FIGURE 18.8 Comparison of fixed versus random effects parameters demonstrating the shrinkage property of random-effects.

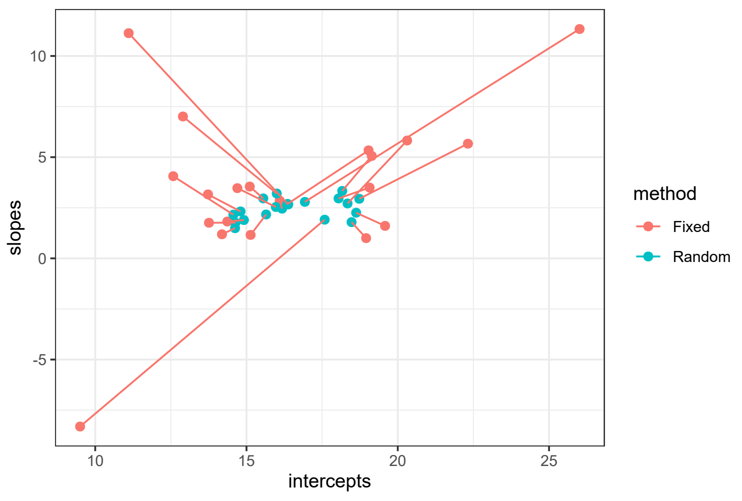


FIGURE 18.9 Comparison of fixed versus random effects parameters demonstrating the shrinkage property of random-effects.

each site and also on the relative magnitude of the variance among sites ($\hat{\sigma}^2_{b_0}, \hat{\sigma}^2_{b_1}$) relative to the variance within sites ($\hat{\sigma}^2_{\epsilon}$). Coefficients for sites that have few observations and that lie far from the population mean will exhibit more shrinkage than those with more observations and that lie closer to the population mean. Coefficients will also exhibit more shrinkage in applications where the within-site variability is large relative to the total (within- plus among-site) variability.

Recall that what makes random effects different from fixed effects is that we assume our random-effect parameters all come from a common distribution. If parameters come from a common distribution, then it makes sense to "borrow information" from other sites when estimating site-specific parameters. The extent to which information from other sites is relevant depends on how similar sites are to each other (i.e., the among-site variance). And the value of using other information from other sites increases in cases where the information content from within-site observations is low. Thus, mixed-effect models provide a sensible way to weight information coming from within and among sites.

18.10 Fitted/predicted values from mixed-effects models

Now that we have explored how to estimate (or predict) site-specific parameters, we may also consider two types of predictions when fitting mixed-effect models. Specifically, we may be interested in exploring predictor-response relationships within specific sites or within the population (averaging across different sites). These two types of predictions are often referred to as subject-specific and population-averaged response patterns (e.g., Fieberg et al. 2009) or conditional and marginal response patterns, respectively (Muff, Held, and Keller 2016).

Subject-specific mean response patterns (associated with each sampled site):

- $E[dbh|age, b_{0i}, b_{1i}] = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})age_{ij}$

Population-averaged mean response patterns (associated with the population of sites):

- $E[dbh|age] = E(E[dbh|X, b_{0i}, b_{1i}]) = \beta_0 + \beta_1 age$

We can generate these two types of predictions using the `predict` function. By default, this will return subject-specific responses: $\widehat{dbh}_{ij} = (\hat{\beta}_0 + \hat{b}_{0i}) + (\hat{\beta}_1 + \hat{b}_{1i})age_{ij}$. We compare site-specific response patterns from the mixed-effects model (red) to those obtained from the fixed-effects model (blue) in Figure 18.10).

```
# Subject-specific regression lines from the mixed-effects model
pines$sspred.rc<-predict(lmer.rc)

# Subject-specific regression lines from the fixed-effects model
pines$fepred<-predict(lmmeans.fc)

# Plot
ggplot(pines, aes(age, dbh)) +
  geom_point(size = 2) +
  geom_line(aes(age, sspred.rc), lwd = 1.3, col = "red") +
  geom_line(aes(age, fepred), lwd = 1.3, col = "blue") +
  facet_wrap( ~ site)
```

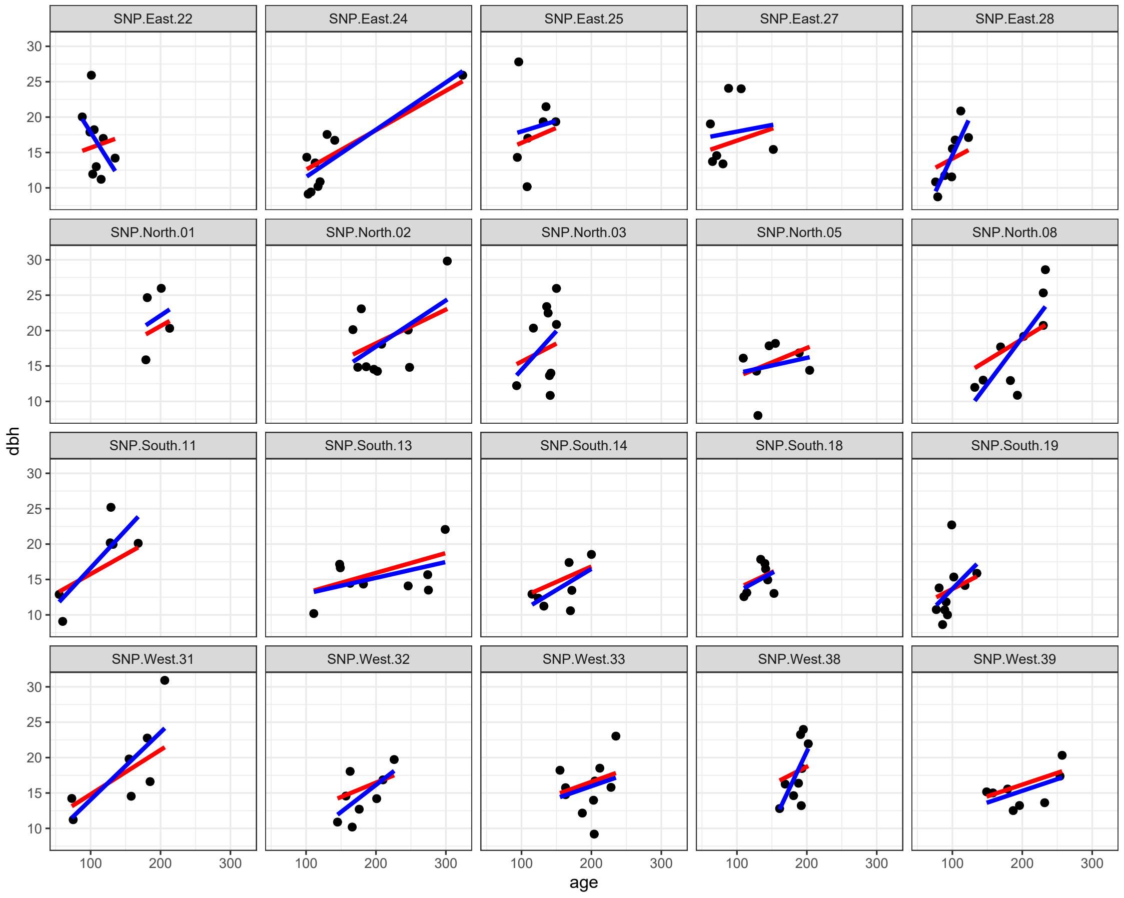


FIGURE 18.10 Fitted regression lines relating dbh to tree age using a fixed - effects (only) model (blue) and a model using random intercepts and slopes (red).

We see that the most of the fitted lines are similar between the two approaches (fixed-effects only and mixed-effects model). However, the estimated slope for `SNP.East.22` (upper left corner) was negative when estimated from the fixed-effects model and positive when estimated from the mixed-effects model. This result again demonstrates how random effects "borrow information" across sites. The mixed-effects model assumes that all of the slope parameters come from a common distribution. In the fixed-effects model, all of the other slope estimates were positive, and site `SNP.East.22` had a limited range of observed ages. Thus, the mixed-effect model infers that the slope for this site should look like the others and should be positive.

What if we had only included random intercepts, as many ecologists tend to do, or fit a fixed-effect model without the interaction between `site` and `agec`? Let's create a similar plot, but this time comparing the random intercept (only) model to the model containing both random intercepts and random slopes.

```
# Subject-specific lines from random intercept mixed-effects model
pines$sspred.ri <- predict(lmer.ri)

# Subject-specific regression lines from the fixed-effects model
pines$fpred2<-predict(lm(dbh ~ site + agec, data = pines))

# Plot
```

```
ggplot(pines, aes(age, dbh)) +
  geom_point(size = 2) +
  geom_line(aes(age, sspred.ri), lwd = 1.3, col = "red")  +
  geom_line(aes(age, fpred2), lwd = 1.3, col = "blue")  +
  facet_wrap( ~ site)
```

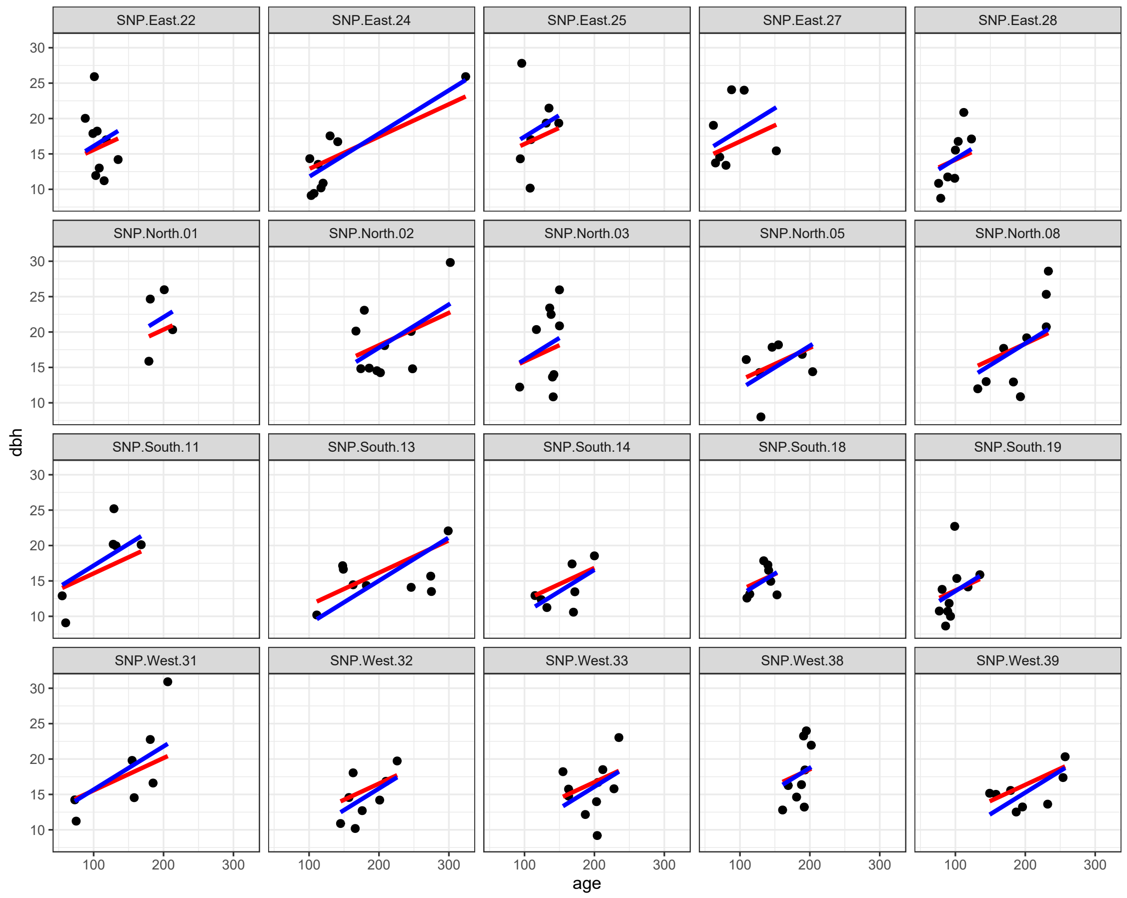


FIGURE 18.11 Fitted regression lines relating `dbh` to tree age using a fixed-effects (only) model (blue) and a mixed-effects model with random intercepts (red). In both cases, the effect of `age` is assumed to be constant across sites.

We see that both methods assume the effect of `age` is constant for all sites, but that each site has its own intercept. The slope parameters are similar for the two methods, but not identical. In addition, the intercepts differ in many cases due to the shrinkage noted previously.

We can also use the `predict` function to generate population-averaged response curves by supplying an extra argument, `re.form = ~ 0`, or equivalently, `re.form = ~ NA`.[8]

```
pines$papred <- predict(lmer.rc, re.form = ~0)
```

[8]Note, the `re.form` argument can also be used to include some, but not all, random effects when models are specified to have multiple random effects (see e.g., Section 18.15.

```
ggplot(pines, aes(age, dbh, col = site)) +
  geom_point(size = 2) +
  geom_line(aes(age, papred), lwd = 1.3, col = "red")
```

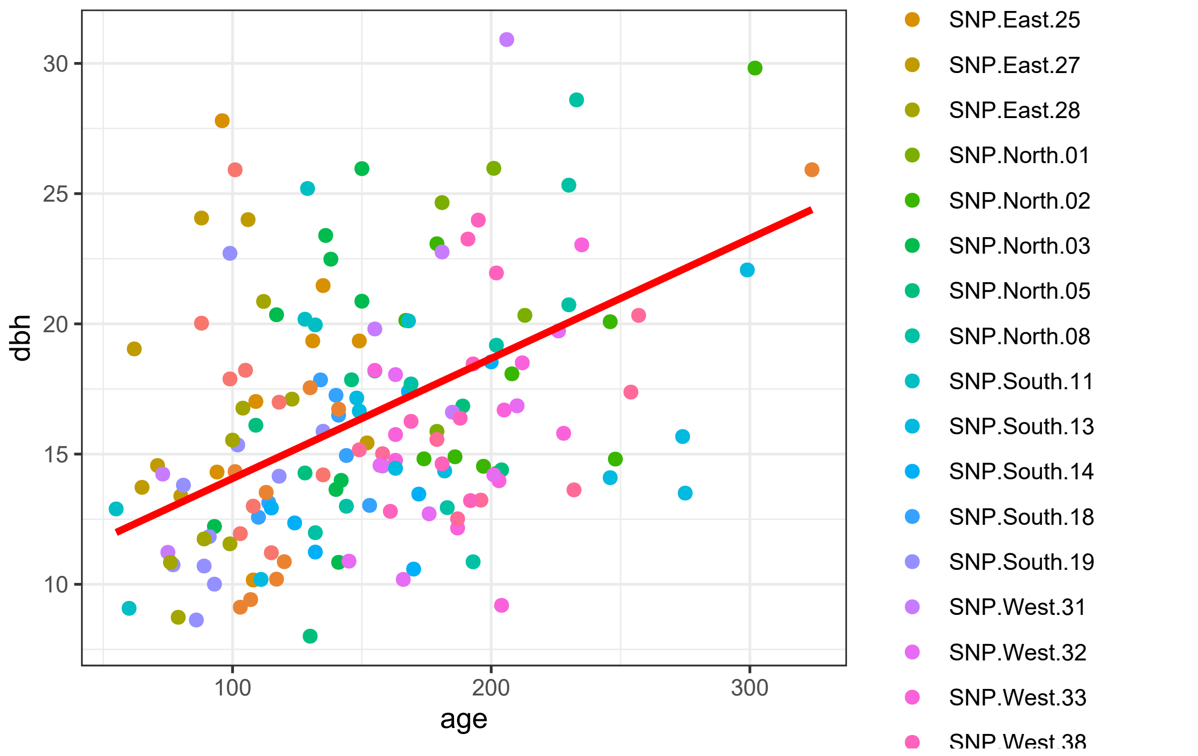


FIGURE 18.12 Population-averaged regression line relating `dbh` to `age` using a mixed model containing random intercepts and slopes.

The `predict` function can also be used to generate predictions for new data. Note, however, that when generating subject-specific predictions, the data associated with the `newdata` argument must include columns corresponding to all of the grouping variables used to specify the random effects in the model (in this case, `site`).

If you look at the help page for `predict.merMod`[9], you will see that it states:

> There is no option for computing standard errors of predictions because it is difficult to define an efficient method that incorporates uncertainty in the variance parameters; we recommend `bootMer` for this task.

---

[9]Note, `predict` is a generic function for generating predictions from different models; behind the scenes, when we use `predict` with a model fit using `lmer` or `glmer`, it actually calls the `predict.merMod` function

We illustrate the use of the `bootMer` function in the `lme4` package to calculate uncertainty in our subject-specific predictions, below[10]:

```
set.seed(101)
  boot.pred <- bootMer(lmer.rc,
              FUN=function(x){predict(x)},
              nsim=500)
```

The `bootMer` function will store the bootstrap statistics (i.e., our predictions for each observation) in a matrix named `t`. The number of rows will equal the number of bootstrap replicates and the number of columns is equal to the number of observations in our data set.

```
dim(boot.pred$t)
```

```
## [1] 500 160
```

```
nrow(pines)
```

```
## [1] 160
```

We can use the `apply` function to calculate quantiles of the bootstrap distribution for each observation, allowing us to construct 95% confidence intervals for the site-specific response patterns[11]:

```
predboot.CI<-apply(boot.pred$t, 2, quantile, c(0.025, 0.975))
```

We then append these values to our pines data set.

```
pines$lowCI <-predboot.CI[1,]
pines$upCI <-predboot.CI[2,]
```

Then, we re-create our plot with the site-specific predictions (Figure 18.13).

```
# Plot
ggplot(pines, aes(age, dbh)) +
  geom_point(size=2)+
  geom_line(aes(age, sspred.rc), lwd=1.3, col="red")  +
  geom_line(aes(age, lowCI), lwd=1.3, col="red", lty=2)  +
  geom_line(aes(age, upCI), lwd=1.3, col="red", lty=2)  +
  facet_wrap(~site)
```

One thing that sticks out in Figure 18.13 is that the point estimates do not always fall in the middle of the confidence interval for many of the sites. Recall that the mixed-effects models borrow information across sites. Like other statistical methods that borrow information by smoothing (e.g., generalized additive models, kernel density estimators), or induce shrinkage (e.g., the LASSO estimator from Section 8.7), the subject-specific predictions from the mixed-effects model trade off some level of bias to increase precision. This property makes it challenging to calculate proper confidence intervals for quantities that are a function of $\hat{b}_{0i}, \hat{b}_{1i}$. If you inspect the `boot.pred` object we created, you will see that `bootMer` returns several non-zero estimates of bias associated with the subject-specific predictions.

[10] Another option that might be worth exploring is the `predictInterval` function in the `merTools` package (Knowles and Frederick 2023).

[11] Note, the `apply` function applies a function to the margins of an array or matrix. Here, we apply the `quantile` function to the columns of `boot.pred$t` by specifying `2` for the second argument (using `1` instead would have applied the `quantile` function to the rows of `boot.pred$t`.

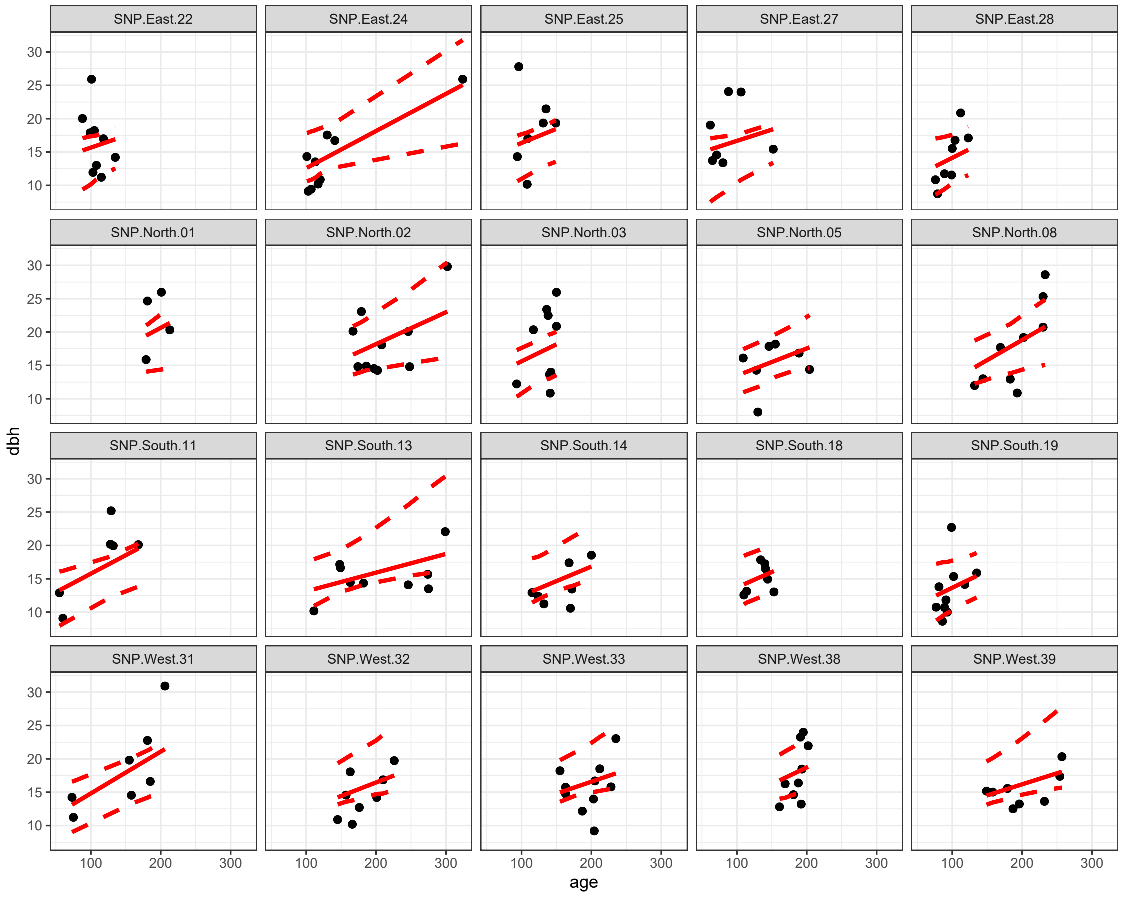


FIGURE 18.13 Fitted regression lines relating `dbh` to `age` using a mixed model containing random intercepts and slopes. A bootstrap was used to calculate pointwise 95-percent confidence intervals for site-specific lines.

18.11 Model assumptions

The assumptions of our mixed-effects model are similar to those of linear regression, except we have added another assumption that our random-effects come from a common Normal distribution. Below, we write these assumptions for the random intercept and slopes model; assumptions for the random intercept model can be inferred by setting b_{1i} to 0.

- **Constant within-cluster variance:** the within-cluster errors, ϵ_{ij}, have constant variance (i.e., there is constant scatter about the regression line for each cluster); $Var(\epsilon_{ij}) = \sigma^2_\epsilon$.
- **Independence:** within-cluster errors, ϵ_{ij}, are independent (both within clusters and across clusters). We also assume the within-cluster errors, ϵ_{ij}, are independent of the random effects, (b_{0i}, b_{1i}).
- **Linearity:** we may state this assumption in terms of the mean response pattern for a particular site, $E[Y_i \mid X_i, b_{0i}, b_{1i}] = (\beta_0 + b_{0,i}) + (\beta_1 + b_{1i})X_i$, or for the mean response pattern averaged across sites, $E[Y_i \mid X_i] = \beta_0 + \beta_1 X_i$ (see Section 18.10). We can derive the latter expression by noting that $E[b_{0i}] = E[b_{1i}] = 0$.

- **Normality:** the ϵ_{ij} follow a Normal (Gaussian) distribution; in addition, the random effects, (b_{0i}, b_{1i}) are normally distributed: $(b_{0i}, b_{1i}) \sim N(0, \Sigma_b)$

We can evaluate these assumptions using similar methods to those used to evaluate the assumptions of linear regression (Section 1.5):

- residual versus fitted values plots to evaluate the linearity and constant variance assumptions
- qqplots or histograms/density plots to evaluate assumptions of normality

18.11.1 Evaluating assumptions

There is a built-in `plot` function that works with mixed-effect models fit using `lme` or `lmer`. This function plots residuals versus subject-specific fitted values, which we explore below for the random-intercept and random intercept and slope models.

```
ri<-plot(lmer.ri, main = "Random intercept") # residual versus fitted
rc<-plot(lmer.rc, main = "Random intercept and slope")
grid.arrange(ri, rc, ncol=2)
```

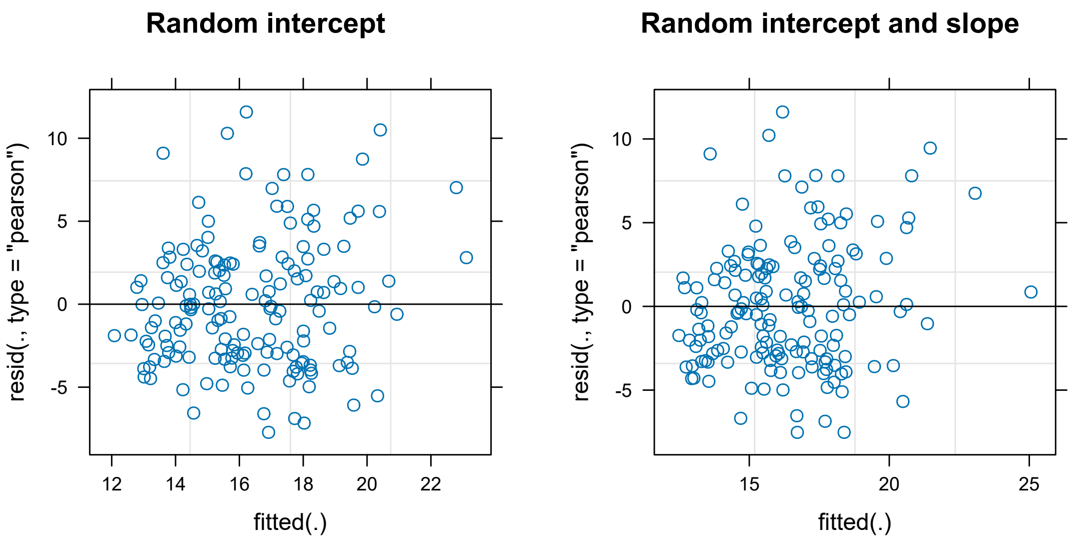


FIGURE 18.14 Residual versus fitted value plots for evaluating linearity and constant variance for the random intercept and random intercept and slope models fit to the pines data.

As with linear regression, we are looking to see whether the residuals have constant scatter as we move from left to right and that there are no trends that would indicate a missing predictor or the need to allow for a non-linear relationship with one of the predictors. These plots generally look OK to me.

We can evaluate the Normality assumption for the ϵ_{ij} and for the random effects using qqplots. The `check_model` function in the `performance` package (Lüdecke et al. 2021) will produce these plots, along with standard residual plots used to evaluate linearity and constant variance assumptions of linear regression (Figures 18.15, 18.16).

```
check_model(lmer.ri)
```

The top-right plot is our standard residual versus fitted value plot (the same one that we saw with the default `plot` method applied to our linear model object). Except for a few outlying points near the extreme left and right of the plot which cause the smooth (green) line to deviate from a horizontal line at 0, the residuals appear to have constant scatter and very little trend. Thus, the linearity assumption seems reasonable. The middle-left plot is the scale-location plot that we encountered previously for evaluating the constant variance assumption of linear regression models in Section 1.5. Here, we are hoping to see points that have constant scatter (and a flat green trend line), indicating the residuals have constant variability. There may be a slight increase in variance for larger fitted values, but overall, this plot doesn't look too bad to me. The middle-right plot shows standardized residuals plotted against leverage; observations with high values for leverage have predictor values that fall far from the sample mean of the predictor values and are potentially influential observations. We have largely ignored this plot when discussing linear models, but we might consider looking into the points that fall outside the green dotted lines to see if they are in error or if dropping them substantially changes our estimates. Of particular concern would be observations that have large residuals and high leverage (i.e., points in the upper or lower right corners of the plot). The next two plots are used to examine the Normality assumption of the within-group errors, ϵ_{ij}. There are some larger-than-expected residuals that fall far from the line in the QQ-plot (lower-left plot), but these observations largely fall with in the simulation bounds. Lastly, the lower-right plot can be used to evaluate whether the random intercepts are Normally distributed. We see that there are a few sites with intercepts that fall off the QQ-line, but the individual estimates have a lot of uncertainty, with confidence intervals that overlap the QQ-line. Lastly, the top left panel provides a posterior predictive check, which is similar in flavor to our Bayesian goodness-of-fit test that we have seen in other chapters. To create this plot, several data sets are simulated the fitted model and their distribution is plotted (light blue lines) along with the distribution of the original data (thicker green line). The distributions of simulated data appear similar to the distribution of the observed data, giving us further confidence that the model assumptions are met and the model is appropriate for the data.

We can create the same set of plots for the random intercept and slope model (Figure 18.16). In this case, we get one more QQ-plot for the random slopes (lower-right plot). Otherwise, the plots look similar to those from the random intercept model.

```
check_model(lmer.rc)
```

Somewhat subjectively, I would conclude that there are no major issues with the assumptions of either model.

18.12 Model comparisons, hypothesis tests, and confidence intervals for fixed-effects and variance parameters

In this section, we will explore methods available for constructing hypothesis tests and confidence intervals for model parameters, and for selecting an appropriate model.

18.12.1 Estimation using restricted maximum likelihood (REML) or maximum likelihood (ML)

One topic we have avoided so far, but will come up when discussing hypothesis tests for fixed and random-effects parameters, is the difference between restricted maximum likelihood (REML) and maximum likelihood

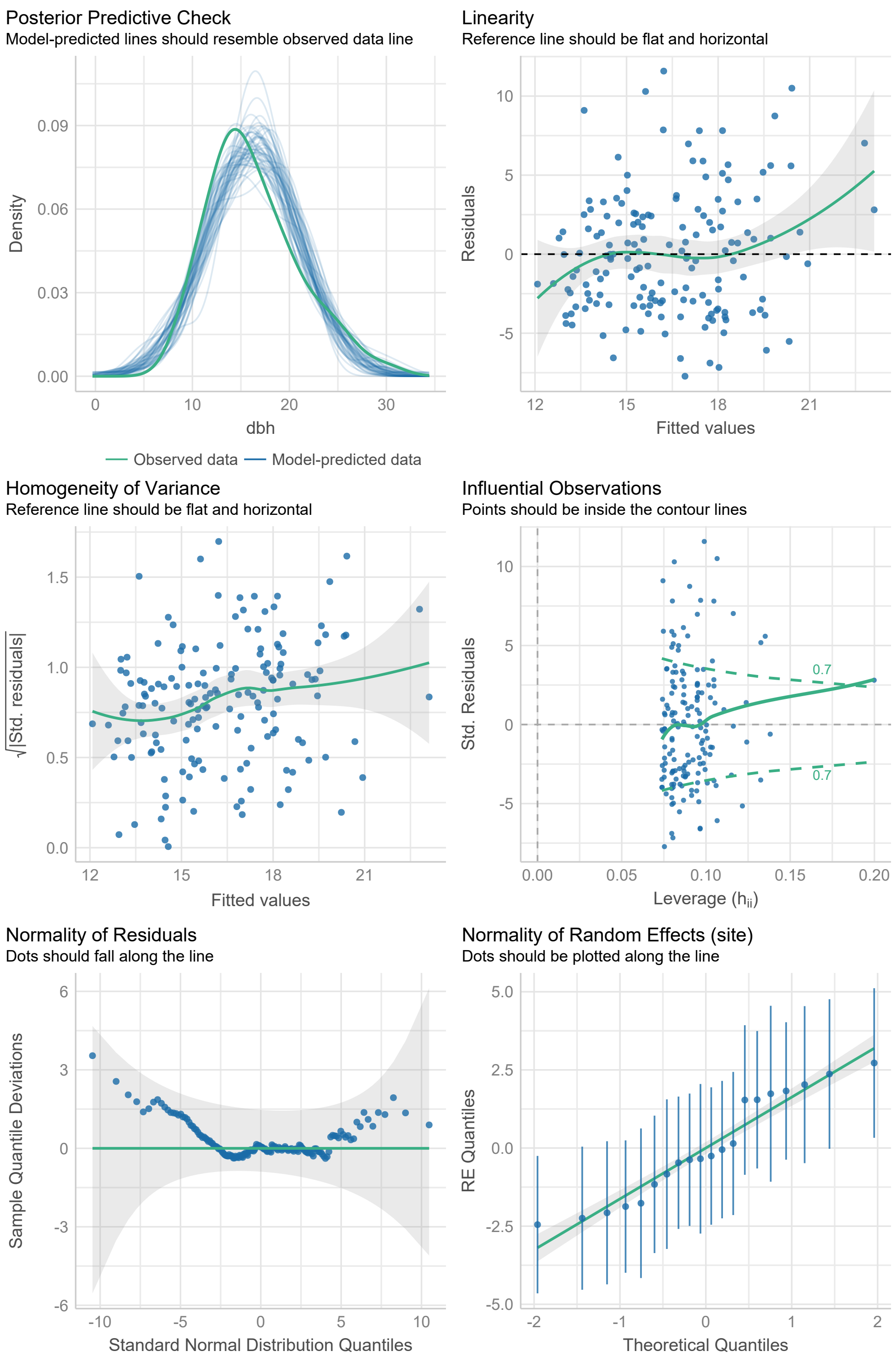


FIGURE 18.15 Residual diagnostic plots using the `check_model` function in the `performance` package (Lüdecke et al. 2021) for the random intercept model fit to the `pines` data set.

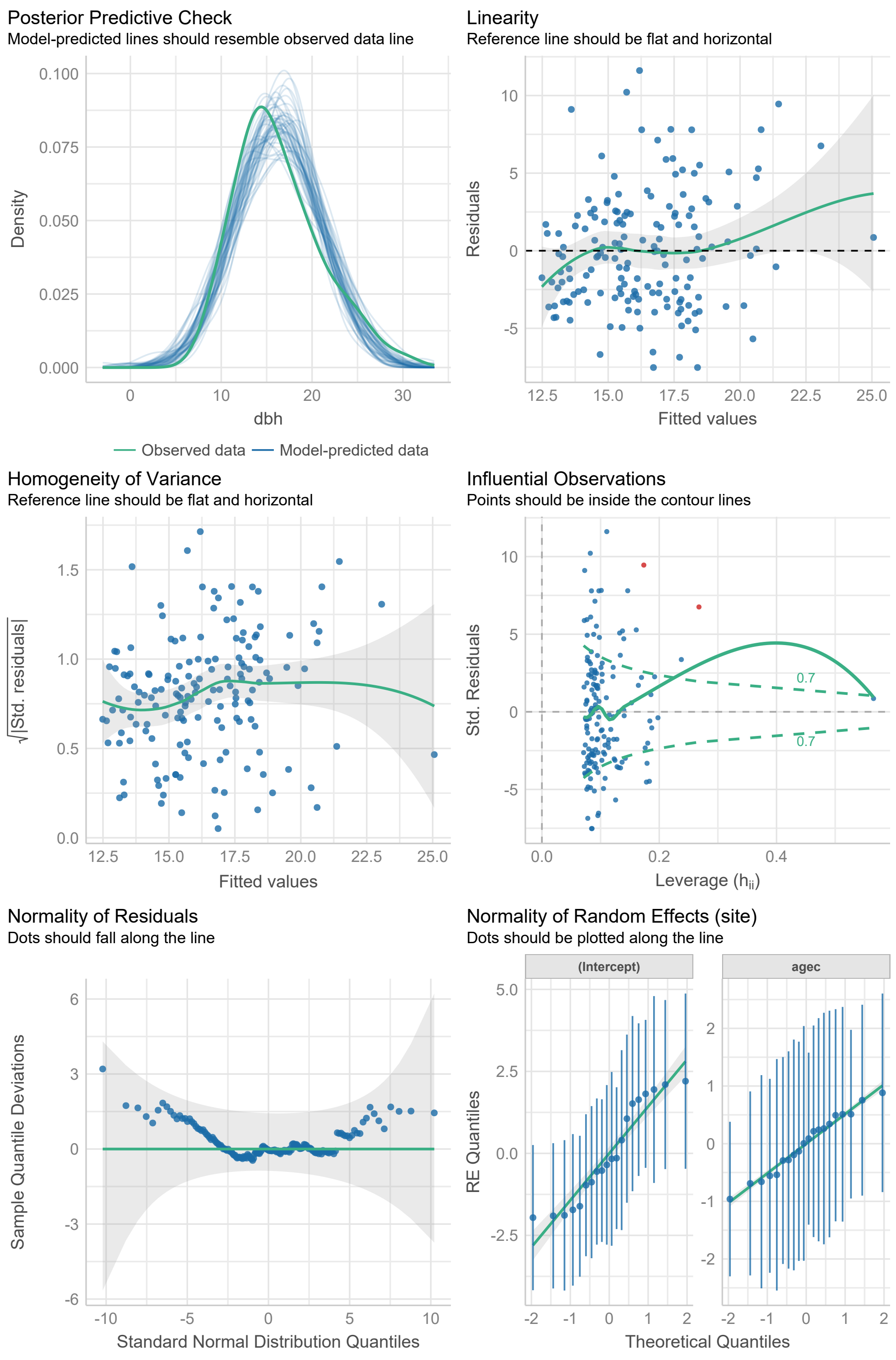


FIGURE 18.16 Residual diagnostic plots using the `check_model` function in the `performance` package (Lüdecke et al. 2021) for the random intercept and slope model fit to the `pines` data set.

(ML) methods for estimating parameters. We briefly mentioned in Section 10.6 that the maximum likelihood estimator of σ^2 for Normally distributed data is biased, and that we tend to use an unbiased estimator that divides by $n-1$ instead of n. It turns out that maximum-likelihood estimators of variance components in mixed-effect models are also biased. REML provides a similar correction by maximizing a modified form of the likelihood that depends on the fixed effects included in the model. Thus, REML is the default method used when fitting models using `lmer`. To estimate parameters using maximum likelihood, we need to add the argument, `method = "ML"`.

Why is this important? Comparisons of models with different fixed effects (e.g., using AIC or likelihood ratio tests) are not valid when using REML. Thus, a common recommendation is to use `method = ML` when comparing models (e.g., using likelihood ratio tests) that differ only in their fixed effects. However, it is generally accepted, and in fact recommended, to compare models that differ only in terms of their random effects using models fit using REML.

18.12.2 Hypothesis tests for fixed-effects parameters and degrees of freedom

To begin our journey to understand tests of fixed-effect parameters, we will explore output when fitting random intercept and random slope models using the `lme` function in the `nlme` package. The syntax will be a little different than what we saw for `lmer`. Instead of specifying the random intercepts associated with each site using `(1 | site)`, we will have a separate argument, `random = ~ 1 | site`. In both cases, we use `1 | site` to tell R that we want the intercepts (i.e., the column of 1s in the design matrix) to vary randomly by `site`.

```
lme.ri <- lme(dbh ~ agec, random=~1 | site, data = pines) # package nlme
summary(lme.ri)
```

```
## Linear mixed-effects model fit by REML
##   Data: pines
##        AIC      BIC    logLik
##   926.7807 939.0311 -459.3904
##
## Random effects:
##  Formula: ~1 | site
##         (Intercept) Residual
## StdDev:    2.053276 4.018271
##
## Fixed effects:  dbh ~ agec
##                 Value Std.Error  DF   t-value p-value
## (Intercept) 16.474305 0.5608232 139 29.375223       0
## agec         2.430471 0.4100212 139  5.927671       0
##  Correlation:
##      (Intr)
## agec 0.012
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -1.92182870 -0.74556286 -0.05850249  0.60196033  2.88237382
##
## Number of Observations: 160
## Number of Groups: 20
```

As with `lmer`, we end up with estimates of the mean intercept and slope in the population, $\hat{\beta}_0 = 16.47$ and $\hat{\beta}_1 = 2.43$, respectively; we also see that `lme` provides estimates of the standard deviations of the site-level intercepts ($\hat{\sigma}_{site} = 2.05$) and within-site errors ($\hat{\sigma}_\epsilon = 4.018$). Unlike with the output from `lmer`, however, we also get a test statistic and p-value for null hypothesis tests that $\beta_0 = 0$ and $\beta_1 = 0$ based on a t-distribution with $df = 139$. This test is only approximate and there are better ways to conduct hypothesis tests (e.g., using simulation based approaches as discussed in Section 18.12.5). Because of the approximate nature of the tests, the developers of the `lme4` package decided it would be best *not* to provide p-values when summarizing the output of a model fit using `lmer`. Nonetheless, the default method that `lme` uses to determine degrees of freedom is instructive, particularly if we fit models that include level-1 and level-2 covariates (i.e., variables that do and do not vary within each site):

```
lme.ri2 <- lme(dbh ~ agec + Aspect, random=~1 | site, data = pines)
summary(lme.ri2)
```

```
## Linear mixed-effects model fit by REML
##   Data: pines
##        AIC      BIC    logLik
##   920.0064 941.3103 -453.0032
##
## Random effects:
##  Formula: ~1 | site
##         (Intercept) Residual
## StdDev:    1.994415  3.99883
##
## Fixed effects:  dbh ~ agec + Aspect
##                 Value Std.Error  DF   t-value p-value
## (Intercept) 18.337679 1.1513720 139 15.926807  0.0000
## agec         2.679396 0.4435988 139  6.040134  0.0000
## AspectNorth -1.362490 1.6508010  16 -0.825351  0.4213
## AspectSouth -2.681490 1.5738823  16 -1.703742  0.1078
## AspectWest  -3.376375 1.6551813  16 -2.039883  0.0582
##  Correlation:
##             (Intr) agec   AspctN AspctS
## agec         0.310
## AspectNorth -0.732 -0.329
## AspectSouth -0.711 -0.160  0.514
## AspectWest  -0.741 -0.363  0.558  0.518
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -1.85312474 -0.71672171 -0.07700017  0.55987900  2.77709336
##
## Number of Observations: 160
## Number of Groups: 20
```

In this case, we see that the p-values associated with the coefficients for `Aspect` are based on a t-distribution with only 16 degrees of freedom. The exact formulas used to calculate these default degrees of freedom (provided below) are not important (again, there are now better ways to conduct these tests) – what is important is that *we have much less information about predictors that do not vary within a cluster than we do about predictors that do vary within a cluster*. For level-1 predictors, `lme` calculates the within-subjects degrees of freedom as the number of observations minus the number of clusters minus the number of level-1 predictors in the model. For level-2 predictors, `lme` calculates degrees of freedom using the number of clusters minus the number of level-2 predictors the model - 1 for the intercept.

```
nrow(pines) - length(unique(pines$site))- 1
```

```
[1] 139
```

```
length(unique(pines$site))- 3 -1
```

```
[1] 16
```

Again, these formula are not that important. What is important is that `lme` attempts to account for the data structure when carrying out statistical tests, recognizing that the `site` is the correct unit of replication for tests involving `Aspect`. The default degrees of freedom calculated by `lme` are essentially correct for balanced data (where you have equal numbers of observations within each cluster), assuming the model assumptions hold. For unbalanced data, the tests (and degrees-of-freedom) are only approximate.

18.12.3 Alternative methods for testing hypotheses about fixed effects

Various other methods for approximating degrees of freedom have been suggested, and these methods are implemented in the `lmerTest` and `pbkrtest` packages (Kuznetsova, Brockhoff, and Christensen 2017; Halekoh and Højsgaard 2014). If the `lmerTest` function has been loaded, you will automatically see tests using a t-distribution with Satterthwaite's degrees of freedom, a method you may have seen when learning about a t-test for a difference in means with unequal variances (Giesbrecht and Burns 1985; Hrong-Tai Fai and Cornelius 1996).

```
library(lmerTest)
lmer.ri <- lmer(dbh ~ agec + (1 | site), data = pines)
summary(lmer.ri)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: dbh ~ agec + (1 | site)
##    Data: pines
##
## REML criterion at convergence: 918.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.9218 -0.7456 -0.0585  0.6020  2.8824
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  site     (Intercept)  4.216   2.053
##  Residual             16.147   4.018
## Number of obs: 160, groups:  site, 20
##
## Fixed effects:
```

```
##             Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)  16.4743     0.5608  13.8574  29.375 7.02e-14 ***
## agec          2.4305     0.4100 108.5317   5.928 3.70e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr)
## agec 0.012
```

We see that the degrees of freedom are smaller (and p-values are larger) than those reported by default by `lme`, particularly for the intercept. Another popular method for calculating degrees of freedom is the method proposed by Kenward and Rodger (Kenward and Roger 1997). This approximation can be used to compare two nested models using the `KRmodcomp` function in the `pbkrtest` package (Halekoh and Højsgaard 2014). Alternatively, tests using Kenward-Rodger degrees of freedom can be calculated by adding `ddf = "Kenward-Roger"` when using the `summary` function with a model fitted using `lmer`:

```
summary(lmer.ri, ddf = "Kenward-Roger")
```

```
## Linear mixed model fit by REML. t-tests use Kenward-Roger's method [
## lmerModLmerTest]
## Formula: dbh ~ agec + (1 | site)
##    Data: pines
##
## REML criterion at convergence: 918.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.9218 -0.7456 -0.0585  0.6020  2.8824
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  site     (Intercept)  4.216   2.053
##  Residual             16.147   4.018
## Number of obs: 160, groups:  site, 20
##
## Fixed effects:
##             Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)  16.4743     0.5611  18.3642  29.358  < 2e-16 ***
## agec          2.4305     0.4208 118.5132   5.776 6.29e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr)
## agec 0.012
```

These alternative degrees of freedom have also been implemented in conjunction with the `anova` function (`lmerTest` also switches the default from sequential, added-in-order tests to backwards elimination tests – see Section 3.10). For example, we now have non-integer degrees of freedom if we consider testing for an effect of `Aspect` in a model that also contains `age`:

```
lmer.ri2 <- lmer(dbh ~ agec + Aspect +  (1 | site), data = pines)
anova(lmer.ri2)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##        Sum Sq Mean Sq NumDF   DenDF F value    Pr(>F)
## agec   583.39  583.39     1 144.061 36.4832 1.249e-08 ***
## Aspect  82.44   27.48     3  13.972  1.7186     0.209
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Further, the order of entry no longer matters:

```
lmer.ri2 <- lmer(dbh ~ Aspect + agec  +  (1 | site), data = pines)
anova(lmer.ri2)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##        Sum Sq Mean Sq NumDF   DenDF F value    Pr(>F)
## Aspect  82.44   27.48     3  13.972  1.7186     0.209
## agec   583.39  583.39     1 144.061 36.4832 1.249e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Luke (2017) compared type I error rates for likelihood ratio tests and tests using Kenward-Roger or Satterthwaite approximations and found the latter two approaches performed well.

Lastly, to reinforce the need to consider random slopes for predictors that vary within a cluster, let's compare the degrees of freedom and the p-values for the fixed effects parameters with and without the random slope:

```
lmer.ri2 <- lmer(dbh ~ agec + Aspect +  (1 | site), data = pines)
anova(lmer.ri2)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##        Sum Sq Mean Sq NumDF   DenDF F value    Pr(>F)
## agec   583.39  583.39     1 144.061 36.4832 1.249e-08 ***
## Aspect  82.44   27.48     3  13.972  1.7186     0.209
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
lmer.rc2 <- lmer(dbh ~ agec + Aspect +  (1  + agec | site), data = pines)
anova(lmer.rc2)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##        Sum Sq Mean Sq NumDF  DenDF F value    Pr(>F)
## agec   488.34  488.34     1 15.462 31.5029 4.418e-05 ***
## Aspect  74.26   24.75     3 13.777  1.5968    0.2355
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that including the random slope does little to change the degrees of freedom (or p-value) for the test associated with `Aspect`, but it has a big effect on the test for `age`, resulting in a much more conservative test with larger p-value!

18.12.4 Confidence intervals for fixed effects and variance parameters

We can use the `confint` function along with models fit using `lmer` to generate confidence intervals using the profile-likelihood method (the default; see Section 10.10), using bootstrapping (by specifying `method = "boot"`), or using a large-sample approximation based on assuming the sampling distribution is Normal (`method = "Wald"`); the latter approach is only available for fixed-effects coefficients since the sampling distribution of variance parameters tends to be right-skewed and thus, far from Normal. The profile-likelihood and bootstrap confidence intervals are to be preferred, but they can be computationally intensive to compute. By contrast, the Wald method is simple and fast; these intervals are calculated simply as $\hat{\beta} \pm 1.96 SE(\hat{\beta})$ When calculating profile-likelihood and bootstrap confidence intervals, you will often see warnings related to computational issues (e.g., non-convergence associated with one or more bootstrapped data sets). In those cases, it is useful to also compute Wald intervals to see how they compare.

We compare these three approaches for the random intercepts and slopes model, below. We have suppressed warnings and messages, but if you run the code on your own, you will be able to see them.

```
conf.pl <- confint(lmer.rc, method = "profile")
conf.boot <- confint(lmer.rc, method = "boot")
conf.norm <- confint(lmer.rc, method = "Wald")
confs <- data.frame(rbind(conf.pl, conf.boot, conf.norm))
names(confs) <- c("LCL", "UCL")
confs$term <- as.factor(rep(c("sigma[b0]","sigma[b1]", "Cov(b0, b1)",
                              "sigma[epsilon]", "beta[0]", "beta[1]"), 3))
confs$method <- rep(c("Profile likelihood", "conf.boot", "conf.norm"),
                    each = nrow(conf.pl))

# Note the estimates are given in a slightly different order than the confidence intervals
confs$estimate <- rep(c(as.data.frame(VarCorr(lmer.rc))$sdcor[c(1,3, 2, 4)],
                        fixef(lmer.rc)), 3)
ggplot(confs, aes(method, estimate)) + geom_point() +
  geom_errorbar(aes(ymin=LCL, ymax=UCL), width=0.2) +
  facet_wrap(~term, scales="free")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Examining Figure 18.17, we see that for this particular data set, the three intervals are all fairly similar.

18.12.5 Simulation-based testing: Comparing models with different random-effect structures

As mentioned in the previous section, the sampling distributions for variance parameters tend to be right skewed. Therefore, tests based on a Normal or t-distribution are not likely to be appropriate. We could consider using a likelihood ratio test to compare nested models (e.g., models with and without a random effects), thereby testing whether one or more variance parameters are 0. For example, we could compare our random intercept model with a random intercept and slope model to test:

$$H_0 : \sigma^2_{b1} = 0 \text{ versus } H_A : \sigma^2_{b1} \neq 0$$

However, this test will be conservative since, under the null hypothesis, σ^2_{b1} is "on the boundary" of the parameter space – i.e., $H_0 : \sigma^2_{b_1} = 0$, sets the parameter $\sigma^2_{b_1}$ to the boundary of what is possible for this parameter (variance parameters cannot go negative). When this is the case, p-values and comparisons using AIC need adjustment (see e.g., Self and Liang 1987). José Pinheiro and Bates (2006) note that the p-value for

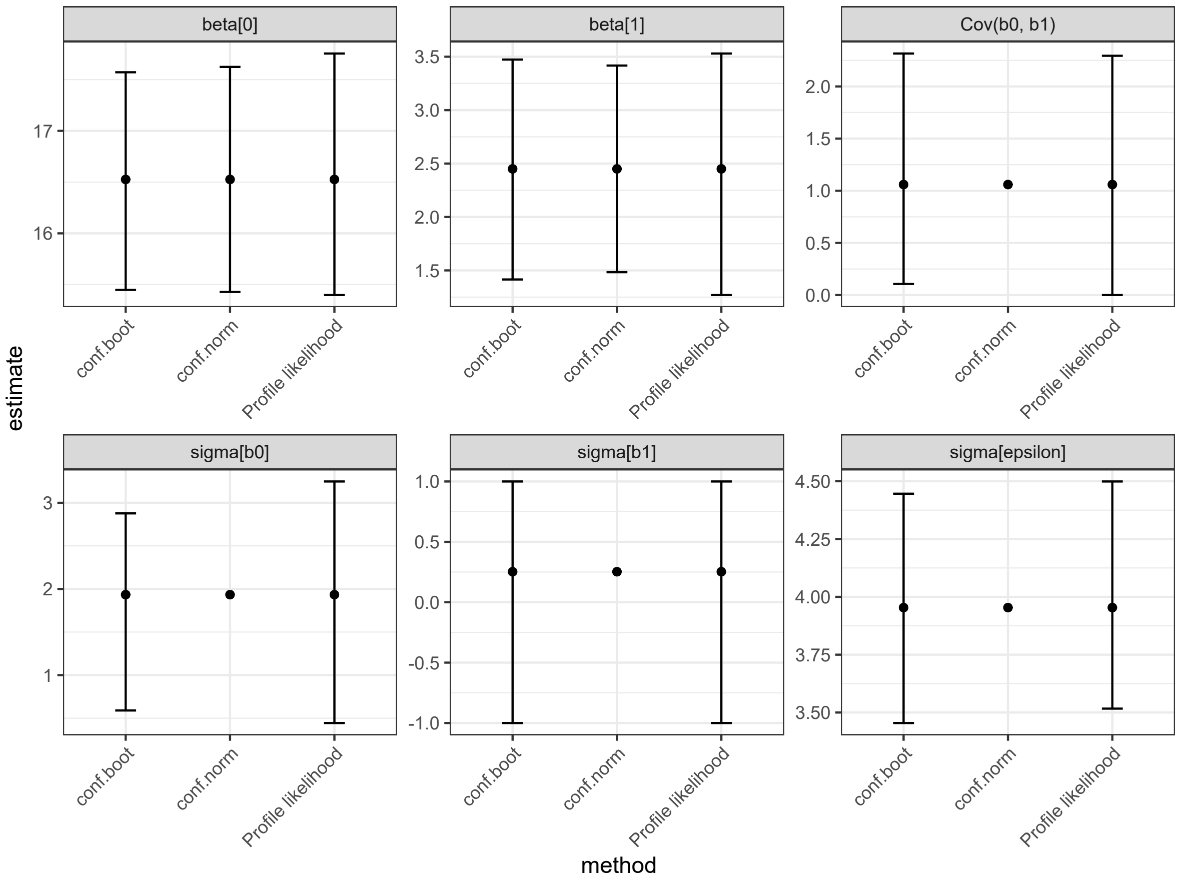


FIGURE 18.17 Comparison of different methods (profile likelihood, bootstrap, Wald) for computing confidence intervals for fixed-effects and variance parameters in mixed-effects models.

a likelihood ratio test involving a single variance parameter will be approximately twice as large as it should be. They suggest that the sampling distribution associated with the likelihood-ratio χ^2 statistic comparing models that differ by 1 parameter (set to the boundary of its parameter space) can be approximated as a 50:50 mixture of a χ^2_1 and a χ^2_2 distribution. In this case, we could calculate an approximate p-value using: `0.5 + 0.5qchisq(LR, 1)`, where LR is the likelihood-ratio statistic calculated by comparing our two models.

Alternatively, we can calculate a p-value using a simulation-based test. Let's consider comparing the random intercept and random intercept and slope models. Our null hypothesis would be that the simpler model containing only random intercepts is appropriate for the data. The alternative hypothesis is that the more complicated random intercepts and slope model fits the data better. Recall that with any null hypothesis test, we need a test statistic and a distribution of that test statistic when the null hypothesis is true. We can use the likelihood-ratio statistic as our test statistic, but rather than rely on an assumed χ^2_1 distribution, we can generate the distribution of the test statistic when the null hypothesis is true using simulations from the null model (Crainiceanu and Ruppert 2004). This will require that we:

1. Simulate data under the null (i.e., smaller) model; a parametric bootstrap can be used here to simulate data from the fitted random-intercept model.
2. Fit the two nested models (random intercept only model, random intercept and slope model) to the simulated data.
3. Calculate the likelihood ratio statistic comparing the two models using the simulated data.

Steps 1-3 will be repeated several times to generate the null distribution of the likelihood ratio statistic. We can then calculate a p-value for the test by comparing the observed likelihood ratio statistic using the actual data to the above null distribution.

The `RLRsim` (Scheipl, Greven, and Kuechenhoff 2008) and `pbkrtest` (Halekoh and Højsgaard 2014) packages provide functions for conducting simulation-based tests. Below, we demonstrate the `PBmodcomp` function in the `pbkrtest` package for conducting the hypothesis test.

```
lrsimtest <- pbkrtest::PBmodcomp(lmer.rc, lmer.ri, nsim = 500)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00227731 (tol = 0.002, component 1)
```

```
summary(lrsimtest)
```

```
## Bootstrap test; time: 11.82 sec; samples: 500; extremes: 178;
## Requested samples: 500 Used samples: 497 Extremes: 178
## large : dbh ~ agec + (1 + agec | site)
## dbh ~ agec + (1 | site)
##            stat      df   ddf p.value
## LRT      1.1054 2.00000        0.5754
## PBtest   1.1054                0.3594
## Gamma    1.1054                0.3619
## Bartlett 1.7840 2.00000        0.4098
## F        0.5527 2.00000 10.36  0.5915
```

`PBmodcomp` provides several tests using different test statistics, including the likelihood ratio test statistic (given by the `LRT` line). The p-values for all of these tests are large, suggesting there is not much evidence to support the more complicated model that also includes random slopes. Nonetheless, it is often best to choose the structure of the random effects to match the experimental design or structure of your data rather than base this choice on a null hypothesis test (Arnqvist 2020). We will further discuss methods for choosing an appropriate model in Section 18.13.

18.12.6 Comparisons using AIC

It is also tempting to use AIC to compare models – e.g., the random intercept model and the model that also includes a random slope:

```
AIC(lmer.ri, lmer.rc)
```

```
        df      AIC
lmer.ri  4 926.7807
lmer.rc  6 929.3578
```

The use of AIC for mixed models is, however, not straightforward. Consider counting the number of parameters in the random-intercept model. We might choose to count the 2 fixed effects parameters, the variance of the random intercepts, and the variance of the within-site errors, so 4 parameters. Sometimes referred to as the marginal AIC, this could be appropriate if we are mainly interested in making inference to the population of sites. On the other hand, we might be interested in estimating site-specific response patterns (and a conditional AIC). In that case, our parameters might be closer to 3 + the number of sites -1. These issues, related to the "level of focus", are discussed in Vaida and Blanchard (2005) and Greven and Kneib (2010). Conditional AICs can be calculated using the `cAIC` function in the `cAIC4` package (Säfken et al. 2021), which again suggests the random intercept model is preferred:

```
library(cAIC4)
cAIC(lmer.ri)
```

```
##
##                 Conditional log-likelihood:  -442.30
##                         Degrees of freedom:    16.55
##  Conditional Akaike information criterion:   917.70
```

```
cAIC(lmer.rc)
```

```
##
##                 Conditional log-likelihood:  -438.36
##                         Degrees of freedom:    21.18
##  Conditional Akaike information criterion:   919.08
```

Another complication is that, like likelihood ratio tests, the comparison between the random intercept and random intercept and slope model also involves setting a variance parameter to a value "on the boundary" of the parameter space (as discussed in Section 18.12.5). AIC comparisons can be conservative in this case, leading to simpler models (Greven and Kneib 2010). For more discussion of these topics, I highly recommend visiting Ben Bolker's GLMM FAQ page.

18.13 Modeling strategies revisited

We saw in Section 8 how challenging it can be to select an appropriate model when allowed freedom to choose among several predictors. These challenges are even more pronounced when fitting mixed-effects models, and again, we will find that there are different viewpoints on how best to go about choosing an appropriate model.

18.13.1 Fixed versus random effects

As a first step, analysts must choose whether to model the effect of categorical variables (and their interactions with other continuous variables) as fixed or random effects. Guidance here is often based on:

- how many levels (i.e., clusters or groups) are associated with the categorical variable. If there are few clusters (e.g., < 5 or 10), then it may be best to model the categorical variable using fixed effects since it can be really difficult to estimate variance parameters with so few replicates. On the other hand, it may be OK model the variable using random effects if the main reason to do so is to account for a feature of the data (e.g., an experimental design that results in hierarchical data or repeated measures). Although the variance component associated with the categorical variable may be imprecisely estimated, its inclusion as a random effect may still have the desired property of properly controlling the type I error rate when conducting hypothesis tests for other fixed effects. In fact, either approach (including a categorical variable as random or fixed) is likely going to control the type I error rate and is preferable to ignoring clustering in the data and treating the observations as though they were independent (see e.g., Oberpriller, Souza Leite, and Pichler 2021).

- whether the focus is on the specific clusters (e.g., sites) that were observed or one wants to make inference to a larger population. In the latter case, mixed-effects models are preferable since they estimate coefficients describing mean effects and their variance across different clusters.

Assuming one chooses to model using random effects, then one is faced with questions about the structure of the random effects as well as which variables to include in the fixed effects part of the model.

18.13.2 Maximal model

As we mentioned previously, too often ecologists default to fitting random-intercept only models. Doing so often results in models that fail to control type I error rates in the face of non-independent observations (Schielzeth and Forstmeier 2009). At the other extreme, is the recommendation to fit a *maximal model* with as many random coefficients as possible (i.e., for all predictors that vary within a cluster) (Barr et al. 2013). Although this sounds appealing from a type I error perspective, in reality, it is often difficult to fit a model with multiple random coefficients. When faced with convergence problems, this often leads to a haphazard model-selection process in which users drop various random coefficients in an attempt to find a simpler model that converges.

One simple thing you can do to improve the chances that the optimizer used to estimate parameters converges is to center and scale your continuous variables. We did this with `age` when fitting mixed models to the `pines` data. We would have encountered issues with fitting the random slope model if we had not done this (see below):

```
lmer(dbh ~ age + (age | site), data = pines)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 1.16634 (tol = 0.002, component 1)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unidenti
##  - Rescale variables?

## Linear mixed model fit by REML ['lmerModLmerTest']
## Formula: dbh ~ age + (age | site)
##    Data: pines
## REML criterion at convergence: 925.3739
## Random effects:
##  Groups   Name        Std.Dev. Corr
##  site     (Intercept) 2.67944
##           age         0.01718  -0.73
##  Residual             3.96886
## Number of obs: 160, groups:  site, 20
## Fixed Effects:
## (Intercept)          age
##     9.45435      0.04629
## optimizer (nloptwrap) convergence code: 0 (OK) ; 0 optimizer warnings; 2 lme4 warnings
```

18.13.3 Top-down model-selection

Largely following Diggle, Liang, and Zeger (1994), Zuur et al. (2009) suggests a "top-down" strategy for choosing an appropriate model. They note that tests for fixed effects will be wrong if the structure of the random effects is too simple. In addition, they note that random effects can soak up variation that could be due to the omission of important fixed effects. Thus, they suggest an iterative approach involving the following 5 steps:

1. Start with a "full model" that contains as many covariates of interest as possible.
2. Compare different random effects structures (e.g., models with random intercepts versus random slopes) using AIC or likelihood-ratio tests (see Section 18.12.5). During this step, they suggest using `method = "REML"` since it will give better estimates of variance parameters.
3. Once an appropriate random-effects structure is chosen, they then compare models with different fixed effects, again using AIC and likelihood-ratio tests but with `method = "ML"`. During this stage, they keep the random effects structure constant based on the results from step 2.
4. Once the fixed effects are chosen, they recommend refitting the final model but with `method = "REML"`. At this point, they also recommend evaluating the assumptions of the model (see Section 18.11)

18.13.4 A multi-level bottom-up approach to model selection

Others, including Jack Weiss and Singer et al. (2003) recommend a bottom up approach that begins by fitting the following models (Figure 18.18):

- Pooled model (assuming independence), include level-1 predictors (predictors that vary within clusters), `lm(y ~ x1)`
- Unconditional means model or variance components model (no predictors, just random intercepts), `lmer(y ~ 1 +(1 | site))`
- Random intercepts (with level 1 predictors), `lmer(y ~ x1 + (1 | site))`
- Random intercepts and slopes (with level 1 predictors), `lmer(y ~ x1 + (1 + x1 | site))`

After picking the best of these models, they then suggest adding level-2 predictors (predictors that are constant within clusters).

Let's apply this approach to our `pines` data. Here, we use AIC, not withstanding the concerns raised earlier (see Section 18.12.6).

```
pooled.model <- lm(dbh ~ agec, data = pines)
uncond.means.model <- lmer(dbh ~ 1 + (1 | site), data = pines)
randint.model <- lmer(dbh ~ 1 + agec + (1 | site), data = pines)
randcoef.model <- lmer(dbh ~ 1 + agec + (agec | site), data = pines)

# Marginal AIC
AIC(pooled.model, uncond.means.model, randint.model, randcoef.model)
```

```
##                    df      AIC
## pooled.model        3 932.4945
## uncond.means.model  3 953.2307
## randint.model       4 926.7807
## randcoef.model      6 929.3578
```

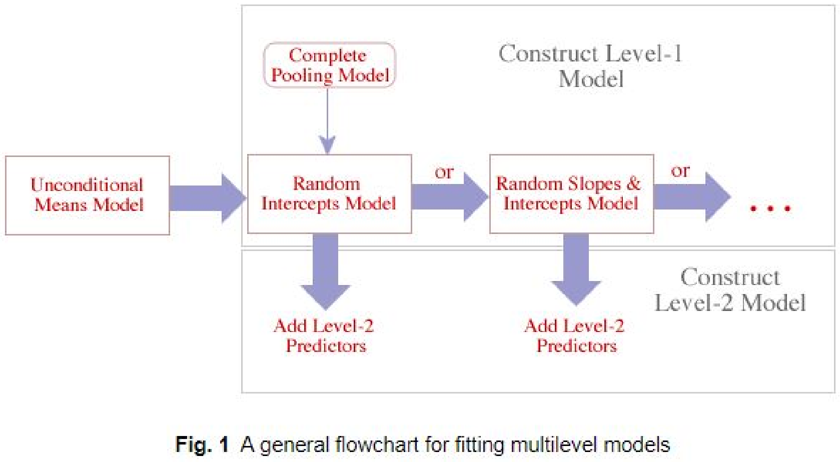


FIGURE 18.18 Multi-level bottum up approach to model selection (Figure from Jack Weiss's web pages).

```
# Conditional AIC
cAIC(pooled.model); cAIC(uncond.means.model); cAIC(randint.model); cAIC(randcoef.model)
```

```
##
##                Conditional log-likelihood:  -463.25
##                        Degrees of freedom:     3.00
##  Conditional Akaike information criterion:   932.49

##
##                Conditional log-likelihood:  -466.87
##                        Degrees of freedom:     9.53
##  Conditional Akaike information criterion:   952.80

##
##                Conditional log-likelihood:  -442.30
##                        Degrees of freedom:    16.55
##  Conditional Akaike information criterion:   917.70

##
##                Conditional log-likelihood:  -438.36
##                        Degrees of freedom:    21.18
##  Conditional Akaike information criterion:   919.08
```

Both sets of AIC comparisons lead to choosing the random intercept model. We also failed to find evidence for the random intercept and slope model using the simulation-based likelihood ratio test in Section 18.12.5. Thus, we might consider adding level-2 covariates (e.g., `Apect`) to the random intercept model.

```
lmer.ri2 <- lmer(dbh ~ 1 + agec + Aspect + (1 | site), data = pines)
```

We can then compare this model to the model without `Aspect` using either hypothesis tests or AIC.

```
anova(lmer.ri2)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##        Sum Sq Mean Sq NumDF   DenDF F value    Pr(>F)
## agec   583.39  583.39     1 144.061 36.4832 1.249e-08 ***
## Aspect  82.44   27.48     3  13.972  1.7186     0.209
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
AIC(lmer.ri, lmer.ri2)
```

```
##          df      AIC
## lmer.ri   4 926.7807
## lmer.ri2  7 920.0064
```

```
cAIC(lmer.ri); cAIC(lmer.ri2)
```

```
##
##                Conditional log-likelihood:  -442.30
##                        Degrees of freedom:    16.55
##  Conditional Akaike information criterion:   917.70

##
##                Conditional log-likelihood:  -441.08
##                        Degrees of freedom:    17.37
##  Conditional Akaike information criterion:   916.90
```

The evidence for the importance of `Aspect` is fairly week, despite resulting in a model with lower AIC.

18.14 Induced correlation, marginal model, generalized least squares

As we noted earlier, many ecologists add random intercepts mainly as a way to avoid issues related to "pseudoreplication" or non-independence (even though random slopes may be needed to guard against inflated type I error rates, Schielzeth and Forstmeier 2009). In this section, we briefly highlight how adding random effects induces correlation among observations that share the same values of the random effect. We will also show an equivalence between the random-intercept model and a marginal model with compound symmetry that can be fit using the `gls` function in the `nlme` package.

Recall from Section 9.7, that the variance of two random variables, X and Y is given by:

$Var(X+Y) = Var(X) + Var(Y) + 2Cov(X,Y)$

In addition, the covariance of X and Y is given by: $Cov(X,Y) = E[XY] - E[X]E[Y]$. If we consider the random intercepts model, we have 2 random terms, b_{0i} and ϵ_{ij}:

$$dbh_{ij} = \beta_0 + b_{0i} + \beta_1 age_{ij} + \epsilon_{ij} \qquad (18.14)$$

Thus, it is easy to derive the following:

- $Var(dbh_{ij}) = Var(b_{0i} + \epsilon_{ij}) = Var(b_{0i}) + Var(\epsilon_{ij}) = \sigma^2_{b_0} + \sigma^2_\epsilon$
- $Cov(dbh_{ij}, dbh_{ij'}) = \sigma^2_{b_0}$ (for any 2 observations collected at the same site since they will share b_{0i}); in addition, the correlation between any two observations from the same site $= Cor(dbh_{ij}, dbh_{ij'}) = \frac{\sigma^2_{b_0}}{\sigma^2_{b_0}+\sigma^2_\epsilon}$.
- $Cov(dbh_{ij}, dbh_{i'j}) = 0$ (for any 2 observations taken from 2 different sites since they will not share b_{0i} or ϵ_{ij})

We can also derive the marginal distribution of our response variable by averaging over the distribution of the random effects (see Section 9.8 for a discussion of conditional versus marginal distributions).

Let's start by writing down equations for the random intercept model in terms of the conditional distribution of $Y|b$ and the distribution of b as in Section 18.8.

$$dbh_{ij}|b_{0i} \sim N(\mu_i, \sigma^2_\epsilon) \tag{18.15}$$
$$\mu_{ij} = (\beta_0 + b_{0i}) + \beta_1 age_{ij} \tag{18.16}$$
$$b_{0i} \sim N(0, \sigma^2_{b_0}) \tag{18.17}$$

If we average over (i.e., integrate out) the random effects, b_{0i}, we get the marginal distribution of Y. This distribution can be derived analytically in the case of linear mixed-effects models, and allows us to write down a likelihood for the data that does not involve the random effects. The marginal likelihood is what R uses when it estimates β_0, β_1, and the variance parameters, σ^2_{b0} and σ^2_ϵ. For the random-intercept model, the marginal distribution for the vector of observations from site i, Y_i, is given by:

$$Y_i \sim N(X_i\beta, V_i) \tag{18.18}$$
$$V_i = \begin{bmatrix} \sigma^2_\epsilon + \sigma^2_{b_0} & \sigma^2_{b_0} & \cdots & \sigma^2_{b_0} \\ \sigma^2_{b_0} & \sigma^2_\epsilon + \sigma^2_{b_0} & \sigma^2_{b_0} & \sigma^2_{b_0} \\ \vdots & \ddots & \ddots & \vdots \\ \sigma^2_{b_0} & \cdots & \sigma^2_{b_0} & \sigma^2_\epsilon + \sigma^2_{b_0} \end{bmatrix} \tag{18.19}$$

Here, V_i gives the variance covariance matrix for the observations from site i and has dimension $n_i \times n_i$. The diagonal elements hold the variances and the off-diagonal elements hold the covariances that we derived above. If we consider all of the data, not just the observations from site i, then we have:

$$Y \sim N(X\beta, \Sigma) \tag{18.20}$$
$$\Sigma = \begin{bmatrix} V_1 & 0 & 0 & 0 \\ 0 & V_2 & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & V_{40} \end{bmatrix} \tag{18.21}$$

Here, Σ is the variance/covariance matrix for all of the observations. This matrix is what is referred to as block diagonal. The 0s represent covariances for observations from different sites – these are 0, since we assume observations from different sites are independent. The V_i contain the variance/covariances for cluster i (eq. (18.19)).

Thus, the random intercept model is equivalent to assuming that all observations from the same site (or cluster) are equally correlated, and that observations from different sites (or clusters) are independent. This particular form of correlation is referred to as compound symmetry. Rather than fitting a random intercepts model, we could have posited this model as one that would be reasonable for our data *a priori*. Specifically, we could consider the vector of response data and directly model their multivariate mean and variance covariance matrix. The `gls` function in the `nlme` package will let us do just that.

```
gls.fit<-gls(dbh ~ agec, method="REML",
             correlation=corCompSymm(form= ~ 1 | site),
            data=pines)
summary(gls.fit)
```

```
Generalized least squares fit by REML
  Model: dbh ~ agec
  Data: pines
       AIC      BIC    logLik
  926.7807 939.0311 -459.3904

Correlation Structure: Compound symmetry
 Formula: ~1 | site
 Parameter estimate(s):
      Rho
0.2070437

Coefficients:
                Value Std.Error   t-value p-value
(Intercept) 16.474305 0.5608219 29.375289       0
agec         2.430468 0.4100210  5.927666       0

 Correlation:
     (Intr)
agec 0.012

Standardized residuals:
       Min         Q1        Med         Q3        Max
-2.1264342 -0.7593302 -0.2326614  0.4739741  3.0916412

Residual standard error: 4.512474
Degrees of freedom: 160 total; 158 residual
```

```
fixef(lmer.ri)
```

```
(Intercept)        agec
  16.474305    2.430471
```

Note that we get the exact same fixed effects coefficients as when fitting the random intercept model. In addition, the estimate of the intraclass correlation that is returned by `gls` = 0.207 is equivalent to $\frac{\hat{\sigma}^2_{b_0}}{\hat{\sigma}^2_{\epsilon}+\hat{\sigma}^2_{b_0}}$ from the random intercept model, and the residual standard error from the `gls` model is equal to $\sqrt{\hat{\sigma}^2_{\epsilon}+\hat{\sigma}^2_{b_0}}$. We show these equivalences, below.

```
# Extract variance parameters from the random intercept model
variancepars<-as.data.frame(VarCorr(lmer.ri))
```

```r
# rho calculated from random intercept model
variancepars[1,4]/(variancepars[1,4]+variancepars[2,4])
```

```
## [1] 0.207045
```

```r
# residual standard error in marginal model
sqrt(variancepars[1,4]+variancepars[2,4])
```

```
## [1] 4.512476
```

As was mentioned in Section 9.11.1, the Normal distribution is unique in that it has separate parameters for the mean and for the variance (and covariance terms). There is no multivariate analogue of other commonly used statistical models (e.g., the Binomial or Poisson distribution) that allow one to directly model the mean and variance/covariance. We will follow up on this point in Section 19.

18.15 Random effects specified using multiple grouping variables

You may find that you have multiple categorical variables that could be considered as grouping variables for structuring random effects. For example, if you follow multiple individuals over several years, you might consider adding a random intercept for each individual to model correlation among observations from the same individual or a random intercept for each year to allow for correlation among observations in the same year. In this section, we will also see that it is possible to include random intercepts (and, potentially, random slopes) using more than one grouping variable (e.g., individual *and* year).

We will consider data from Dickie et al. (2022), who explored associations between wolf home-range size and measures of primary productivity and the extent of linear features (e.g., roads) within individuals' home ranges. Data from this study are included in the `Data4Ecologists` package in the `HRData` data set and can be accessed using:

```r
library(Data4Ecologists)
data(HRData)
```

In addition to having repeated observations on individuals over time (home ranges were estimated seasonally and annually), more than one wolf was observed from some wolf packs (Figure 18.19). Thus, we might consider including `AnimalID`, `PackID`, and `Year` as grouping variables for random effects (i.e., random intercepts or random coefficients).

```r
HRsummary <- HRData %>% group_by(PackID, Year) %>%
  count()
ggplot(HRsummary, aes(Year, PackID, size = n)) + geom_point() +
  ylab("Pack ID")
```

Dickie et al. (2022) considered models with the following fixed effects:

- `Season` = snow (Nov-April) or snow-free (May-October)

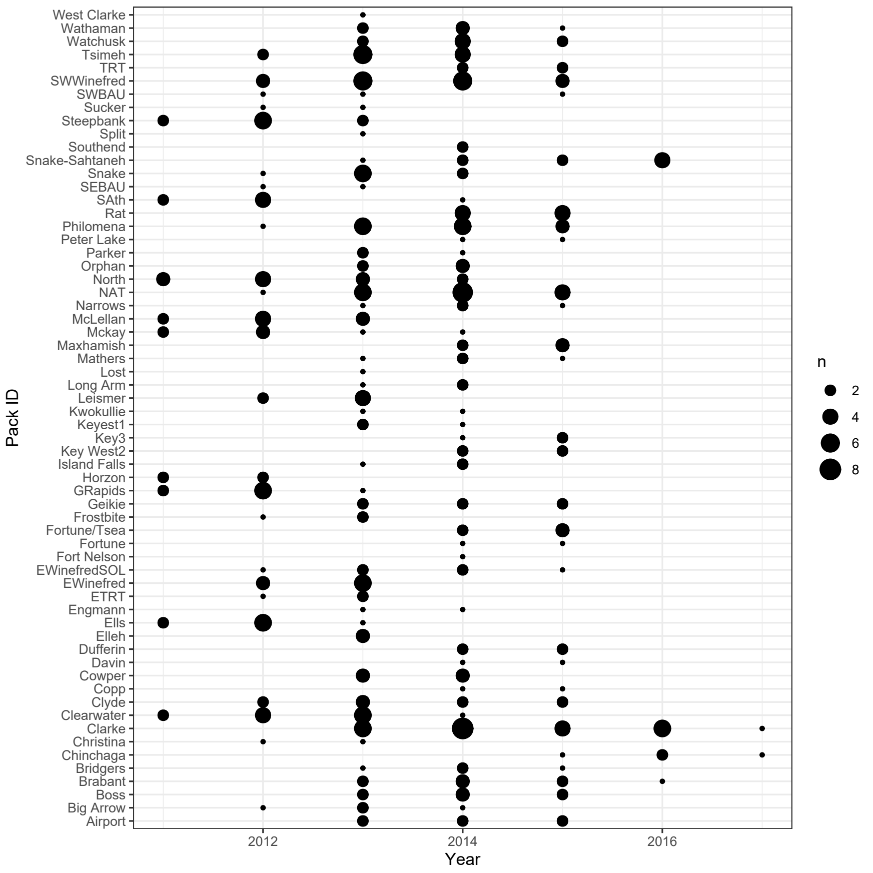


FIGURE 18.19 Number of observations of estimated home-range size for each of several wolf packs in the `HRData` data set from Dickie et al. (2022).

- `StudyArea` (`SA` in their original data set) = BC for northeastern British Columbia, WHEC for northern Alberta, RICC for northeastern Alberta, and SK for Saskatchewan
- `DiffDTScaled` = the number of days of monitoring data associated with the home-range estimate, scaled between 0 and 1
- `EVIScaled` = a scaled measure of primary productivity within the estimated home ranges (assumed to influence the abundance of wolf prey)
- `LFD` = density of linear features (`LFD`) within the estimated home ranges.

They also considered an interaction between `EVIScaled` and `LFD` to determine if the effect of linear features depended on resource density. Dickie et al. (2022) compared three different models, each with a different random-effects specification. Although there paper is quite interesting, I find their choice of models a bit strange (see Section 18.15.1). Instead, let's begin by considering the following model:

```
model1 <- lmer(log(HRsize) ~ Season + StudyArea + DiffDTScaled + LFD*EVIScaled +
                     (1 | AnimalId) + (1 | Year),  REML=TRUE, data = HRData)
summary(model1)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: log(HRsize) ~ Season + StudyArea + DiffDTScaled + LFD * EVIScaled +
##     (1 | AnimalId) + (1 | Year)
##    Data: HRData
##
## REML criterion at convergence: 718.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.1974 -0.5314 -0.0044  0.5087  2.6537
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  AnimalId (Intercept) 0.20867  0.4568
##  Year     (Intercept) 0.03643  0.1909
##  Residual             0.28067  0.5298
## Number of obs: 359, groups:  AnimalId, 142; Year, 7
##
## Fixed effects:
##                 Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)      9.50774    0.41954 232.87651  22.662  < 2e-16 ***
## SeasonSnowFree  -0.48662    0.07464 242.09915  -6.519 4.07e-10 ***
## StudyAreaRICC   -0.94537    0.15550 127.37452  -6.079 1.30e-08 ***
## StudyAreaSK     -1.27522    0.27456 165.51587  -4.645 6.91e-06 ***
## StudyAreaWHEC   -1.30407    0.18430 151.99775  -7.076 5.10e-11 ***
## DiffDTScaled     0.40872    0.10629 293.98415   3.845 0.000148 ***
## LFD             -4.87658    1.80615 328.18140  -2.700 0.007294 **
## EVIScaled       -2.47882    0.57135 269.65609  -4.339 2.03e-05 ***
## LFD:EVIScaled    5.09616    2.83954 334.21395   1.795 0.073603 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) SsnSnF SARICC StdASK SAWHEC DffDTS LFD    EVIScl
## SeasonSnwFr -0.015
## StudyArRICC -0.352  0.004
## StudyAreaSK -0.842  0.036  0.542
## StudyArWHEC -0.388 -0.050  0.647  0.542
## DiffDTScald -0.068 -0.507 -0.042 -0.006 -0.051
## LFD         -0.698  0.008  0.102  0.525  0.098 -0.017
## EVIScaled   -0.849 -0.005 -0.005  0.585  0.036 -0.002  0.713
## LFD:EVIScld  0.601  0.002  0.025 -0.370  0.040  0.005 -0.963 -0.729
```

Writing down a set of equations for the model is challenging due to the incorporation of multiple categorical variables, including our grouping variables for the random effects (`Year` and `AnimalId`) and other categorical predictors modeled using fixed effects (`Season`, `StudyArea`). We will use indicator variables as in Section 3 to represent fixed effects predictors associated with the categorical variables `Season` and `StudyArea`. Let Y_{ij} be the home-range size for j^{th} observation associated with individual i. Further, let b_i and $\gamma_{k(ij)}$ represent random intercepts associated with individual i and year k (where the $k(ij)$ subscript indicates the j^{th} observation of individual i is from year k). One option for describing the model is as follows:

$$Y_{ij}|b_i, \gamma_{k(ij)} \sim N(\mu_{ij}, \sigma^2_\epsilon) \tag{18.22}$$
$$\mu_{ij} = \beta_0 + b_i + \gamma_{k(ij)} + \beta_1 I(Season = SnowFree)_{ij} + \beta_2 I(StudyArea = RICC)_{ij} +$$
$$\beta_3 I(StudyArea = SK)_{ij} + \beta_4 I(StudyArea = WHEC)_{ij} + \beta_5 DiffDTScaled_{ijkl} +$$
$$\beta_6 LFD_{ijkl} + \beta_7 EVIScaled_{ij} + \beta_8 LFD_{ij} \cdot EVIScaled_{ij}$$
$$b_i \sim N(0, \sigma^2_b)$$
$$\gamma_{k(ij)} \sim N(0, \sigma^2_\gamma),$$

If we look at the `Random effects` section near the top of the output returned by the `summary` function, we see that we have 3 variance components, one for `AnimalId`, one for `Year`, and one for `Residual`. The first two variance components correspond to σ^2_b an σ^2_γ and quantify variation among animals and years that is not attributable to the other (fixed effect) variables included in the model. The last variance component is our standard residual variance, σ^2_ϵ.

One way to understand this model is to consider the induced correlation implied by the random effects. As discussed in Section 18.14, observations that share one or more random effects will be correlated whereas observations that do not share any random effects will be independent of one another. Thus, we can infer the following:

- Observations on the same animal will be correlated due to sharing a random intercept, b_i, associated with the `(1 | AnimalId)` term.
- Observations from the same year will be correlated due to sharing a random intercept, γ_k associated with the `(1 | Year)`.
- Observations on the same animal *and in the same year* will share random intercepts from both terms. Thus, these observations will be more correlated than observations on the same animal but in *different years* or observations in the same year but on *different animals*.
- Observations will only be independent if they are on different animals *and* taken in different years.

Alternatively, we could consider a model with random intercepts for `PackID` and `Year`.

```
model2 <- lmer(log(HRsize) ~ Season + StudyArea + DiffDTScaled + LFD*EVIScaled +
                     (1 | PackID) + (1 | Year),  REML=TRUE, data = HRData)
```

Think-Pair-Share: What are the induced correlations that arise from this model specification?

In principle, it would also be possible to consider a model that includes random intercepts for `AnimalId`, `PackID`, and `Year`, but `lmer` provides a warning that one or more parameters are on the boundary, causing a singular fit.

```
model3 <- lmer(log(HRsize) ~ Season + StudyArea + DiffDTScaled + LFD*EVIScaled +
                      (1 | AnimalId) + (1 | PackID ) + (1 | Year), REML=TRUE, data = HRData)
```

```
## boundary (singular) fit: see help('isSingular')
```

If we inspect the output of `model 3`, we see that it returns an estimate of the variance attributable to `AnimalId` that is essentially 0.

```
summary(model3)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: log(HRsize) ~ Season + StudyArea + DiffDTScaled + LFD * EVIScaled +
##     (1 | AnimalId) + (1 | PackID) + (1 | Year)
##    Data: HRData
##
## REML criterion at convergence: 672.5
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -3.12454 -0.49717 -0.01873  0.51040  2.81691
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  AnimalId (Intercept) 0.00000  0.0000
##  PackID   (Intercept) 0.19393  0.4404
##  Year     (Intercept) 0.03662  0.1914
##  Residual             0.28908  0.5377
## Number of obs: 359, groups:  AnimalId, 142; PackID, 62; Year, 7
##
## Fixed effects:
##                 Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)      9.42538    0.45339 185.58848  20.789  < 2e-16 ***
## SeasonSnowFree  -0.47648    0.07333 301.02719  -6.498 3.37e-10 ***
## StudyAreaRICC   -1.02732    0.20353  62.31087  -5.048 4.15e-06 ***
## StudyAreaSK     -1.27402    0.31023 119.41966  -4.107 7.38e-05 ***
## StudyAreaWHEC   -1.32111    0.23965  68.60989  -5.513 5.79e-07 ***
## DiffDTScaled     0.41984    0.09964 317.57043   4.213 3.28e-05 ***
## LFD             -4.43188    1.86342 302.34969  -2.378 0.018011 *
## EVIScaled       -2.27967    0.60234 229.18118  -3.785 0.000197 ***
## LFD:EVIScaled    4.13155    2.90450 315.60162   1.422 0.155879
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) SsnSnF SARICC StdASK SAWHEC DffDTS LFD    EVIScl
## SeasonSnwFr -0.022
## StudyArRICC -0.355  0.011
## StudyAreaSK -0.845  0.050  0.521
## StudyArWHEC -0.391 -0.023  0.576  0.527
## DiffDTScald -0.046 -0.485 -0.059 -0.042 -0.070
## LFD         -0.679 -0.005  0.111  0.515  0.122 -0.015
## EVIScaled   -0.832 -0.018 -0.007  0.556  0.039  0.009  0.675
## LFD:EVIScld  0.574  0.027 -0.005 -0.352  0.002 -0.013 -0.956 -0.695
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

This likely occurs because there were very few animals observed from each pack, so the `AnimalId` and `PackID` variables are largely accounting for similar dependencies in the data.

18.15.1 Aside: Digging deeper into models fit by Dickie et al. (2022)

Rather than use `AnimalId`, `WolfID`, or `Year` directly as grouping variables to structure random effects, Dickie et al. (2022) created new variables, `PackYear` and `AnimalYear`, by pasting together `PackID` and `Year` and `AnimalId` and `Year`, respectively.

```
HRData <- HRData %>% mutate(AnimalYear = paste(HRData$AnimalId, HRData$Year, sep=""),
                            PackYear = paste(HRData$PackID, HRData$Year, sep=""))
```

They then compared the following three models using AIC to select an appropriate random-effects structure for capturing the correlation in the data (see this github repository here and a zenendo repository here)[12]:

```
model4 <- lmer(log(HRsize) ~ Season + StudyArea + DiffDTScaled + LFD*EVIScaled +
                     (1 | AnimalYear), REML=TRUE, data = HRData)
model5 <- lmer(log(HRsize) ~ Season + StudyArea + DiffDTScaled + LFD*EVIScaled +
                     (1 | PackYear), REML=TRUE, data = HRData)
model6 <- lmer(log(HRsize) ~ Season + StudyArea + DiffDTScaled + LFD*EVIScaled +
                     (1 | AnimalYear/PackID), REML=TRUE, data = HRData)
```

The first model includes random intercepts for each unique combination of `AnimalId` and `Year` and the second model includes random intercepts for each unique combination of `PackID` and `Year`. The last model is equivalent to a specification that includes two sets of random intercepts, one for each unique animal and year combination (i.e., `AnimalYear`) and one for each unique combination of `AnimalYear` and `PackID`. We demonstrate this equivalence below by fitting an alternative specification of this model and then showing that the AIC, variance components, and all fixed effects parameters are equal.

```
model6b <- lmer(log(HRsize) ~ Season + StudyArea + DiffDTScaled + LFD*EVIScaled + (1|AnimalYear)
                   + (1 | AnimalYear:PackID), REML=TRUE, data = HRData)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unidenti
##  - Rescale variables?
```

```
AIC(model6, model6b)
```

```
##         df     AIC
## model6  12 799.056
## model6b 12 799.056
```

```
all.equal(fixef(model6), fixef(model6b))
```

```
## [1] TRUE
```

```
print(VarCorr(model6), comp = "Variance")
```

```
##  Groups            Name        Variance
##  PackID:AnimalYear (Intercept) 0.072756
##  AnimalYear        (Intercept) 0.152451
##  Residual                      0.294430
```

[12]Note that their `est` variable has been renamed as `HRest` and their `SA` variable has been renamed `StudyArea` in the `Data4Ecologists` package.

```
print(VarCorr(model6b), comp = "Variance")
```

```
## Groups             Name        Variance
## AnimalYear         (Intercept) 0.072756
## AnimalYear:PackID (Intercept) 0.152451
## Residual                       0.294430
```

Again, it is helpful to consider the implied correlation structure in these additional models (`model4`, `model5`, and `model6b`).

- In model 4, any two observations from the same animal and year will be correlated, whereas observations from different animals or from the same individual but in different years will be assumed to be independent.
- In model 5, any two observations from the same wolf pack and year will be correlated, whereas observations from different wolf packs or from the same pack but in different years will be assumed to be independent.
- In model 6, any two observations from the same animal and same year will be correlated due to sharing a random intercept from the `(1 | AnimalYear)` term. In addition, observations from the same animal, year, and pack will share a random intercept from the `(1 | AnimalYear:PackID)` term. Observations from the same individual, year, and pack will have the highest degree of correlation because these observations will share two random intercepts, followed by observations from the same individual and year (but different pack). Lastly, observations from different individuals or from the same individual but in different years will be assumed to be independent.

After considering this last model and the implied correlation structure, you may wonder whether individuals actually switch packs? If not, it seems likely that the second random intercept term in `model6`, i.e., `(1 | AnimalYear:PackID)`, would be unnecessary. Interestingly, it turns out that there are no observations where an individual wolf was associated with two different packs in the same year.

```
Npacks <- HRData %>% group_by(AnimalYear) %>% dplyr::summarize(npacks = length(unique(PackID)))
table(Npacks$npacks)
```

```
##
##   1
## 289
```

Thus, the two random intercept terms in the third model turn out to be redundant. We can show this by demonstrating the equivalence between models with `(1 | AnimalYear:PackID)` and `(1 | AnimalYear)`.

```
model4 <- lmer(log(HRsize) ~ Season + StudyArea + DiffDTScaled + LFD*EVIScaled +
                     (1 | AnimalYear), REML=TRUE, data = HRData)
model4b <- lmer(log(HRsize) ~ Season + StudyArea + DiffDTScaled + LFD*EVIScaled +
                     (1 | AnimalYear:PackID), REML=TRUE, data = HRData)
all.equal(fixef(model4), fixef(model4b))
```

```
## [1] TRUE
```

```
AIC(model4, model4b)
```

```
##         df     AIC
## model4  11 797.056
## model4b 11 797.056
```

```
print(VarCorr(model4), comp = "Variance")
```

```
##  Groups     Name        Variance
##  AnimalYear (Intercept) 0.22521
##  Residual               0.29443
```

```
print(VarCorr(model4b), comp = "Variance")
```

```
##  Groups            Name        Variance
##  AnimalYear:PackID (Intercept) 0.22521
##  Residual                      0.29443
```

Given this equivalence, the model convergence warning that is given when fitting `model6b` makes sense, yet it seems really odd that `lmer` ends up converging when specifying the model using `(1 | AnimalYear/PackID)` as in `model6`. It is also not clear how `lmer` determines how much of the variance to associate with `AnimalYear` and how much to associate with `AnimalYear:PackID` since these terms provide equivalent grouping structures. We do see that the estimated variance components, 0.07276 and 0.15245, add up to the variance component in our model that includes only one of the two terms.

```
print(VarCorr(model4), comp = "Variance")
```

```
##  Groups     Name        Variance
##  AnimalYear (Intercept) 0.22521
##  Residual               0.29443
```

```
print(VarCorr(model6), comp = "Variance")
```

```
##  Groups            Name        Variance
##  PackID:AnimalYear (Intercept) 0.072756
##  AnimalYear        (Intercept) 0.152451
##  Residual                      0.294430
```

18.16 Implementing mixed-effects models in JAGS

18.16.1 Random intercept model

Let's begin by loading packages for communicating with JAGS and summarizing JAGS output:

```
library(R2jags)
library(MCMCvis)
library(mcmcplots)
```

In this section, we will go back to considering the `pines` data set. We begin by implementing the random intercept model, which is fairly straightforward in JAGS. We have one additional variance parameter, σ_{b0}, for which we need to specify a prior; we use a uniform distribution similar to what we have used when specifying a prior for σ_ϵ. In the JAGS code, below, we write the model in terms of an overall mean intercept and deviations from this mean for each site, b_{0i}.

```
jags.lme<-function(){

  # Priors for the site-specific deviations from the mean intercept
  for (i in 1:n.groups){
    b0[i] ~ dnorm(0, tau.b0)   # Random intercepts
  }

  # Priors for parameters describing the intercepts and
  # their variability
  beta0 ~ dnorm(0, 0.001)      # Mean intercept
  sigma.b0 ~ dunif(0, 100)     # SD of the random intercepts
  tau.b0 <- 1 / (sigma.b0 * sigma.b0)  # precision of intercepts

  # Priors for fixed effect regression parameters
  beta1 ~ dnorm(0, 0.001)           # Common slope agec
  tau.eps <- 1 / ( sigma.eps * sigma.eps)       # Residual precision
  sigma.eps ~ dunif(0, 100)         # Residual standard deviation

  # Calculate site-specific intercepts as derived parameters
  for (i in 1:n.groups){
    beta0i[i] <- beta0 + b0[i]
  }

  # Likelihood
  for (i in 1:nobs) {
    dbh[i] ~ dnorm(mu[i], tau.eps)     # The random variable
    mu[i] <- beta0 + b0[site[i]] + beta1*agec[i]  # Expectation
  }
  }
```

Alternatively, we could have written down the model in terms of the site-specific intercepts, $\beta_{0i} = \beta_0 + b_{oi}$, noting that $\beta_{0i} \sim N(\beta_0, \sigma^2_{b0})$ as shown below. With this specification, we do not have to estimate the b_{0i}, but we could add a line of code to calculate them as derived parameters (i.e., $b_{0i} = \beta_{0i} - \beta_0$):

```
jags.lme.alt<-function(){

  # Priors for the intercepts
  for (i in 1:n.groups){
    alpha[i] ~ dnorm(beta0, tau.b0)   # Random intercepts
  }
  # Priors for parameters describing the intercepts and
  # their variability
  beta0 ~ dnorm(0, 0.001)      # Mean intercept
  sigma.b0 ~ dunif(0, 100)     # SD of the random intercepts
  tau.b0 <- 1 / (sigma.b0 * sigma.b0)

  # Priors for fixed effect regression parameters
  beta1 ~ dnorm(0, 0.001)           # Common slope agec
  tau.eps <- 1 / ( sigma.eps * sigma.eps)       # Residual precision
  sigma.eps ~ dunif(0, 100)         # Residual standard deviation

  # Likelihood
  for (i in 1:nobs) {
    dbh[i] ~ dnorm(mu[i], tau.eps)     # The random variable
    mu[i] <- alpha[site[i]] + beta1*agec[i]  # Expectation
  }
  }
```

We then gather our data for model fitting, creating the site indicator variables, MCMC settings, and call JAGS.

```
# Bundle data
jags.data <- list(dbh = pines$dbh, agec = as.numeric(pines$agec),
                  site = as.numeric(as.factor(pines$site)),
                  n.groups=length(unique(pines$site)),
                  nobs = nrow(pines))


# Parameters to estimate
parameters <- c("beta0i", "beta0", "beta1", "sigma.b0", "sigma.eps")


# Start Gibbs sampling
out.ri <- jags.parallel(data=jags.data,
                       parameters.to.save=parameters,
                       model=jags.lme,
                       n.thin=1, n.chains=3, n.burnin=1000, n.iter=10000)
```

Next, we inspect the output and compare it to the results we obtained from `lmer`

```
MCMCsummary(out.ri, params=c("beta0", "beta1", "sigma.b0", "sigma.eps"), round=3)
```

```
##             mean    sd   2.5%    50%  97.5% Rhat n.eff
## beta0     16.465 0.609 15.283 16.460 17.699    1 27000
## beta1      2.425 0.481  1.503  2.420  3.374    1  7097
## sigma.b0   2.158 0.698  0.912  2.108  3.671    1  2572
## sigma.eps  4.078 0.255  3.614  4.063  4.615    1  8635
```

```
summary(lmer.ri)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: dbh ~ agec + (1 | site)
##    Data: pines
##
## REML criterion at convergence: 918.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.9218 -0.7456 -0.0585  0.6020  2.8824
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  site     (Intercept)  4.216   2.053
##  Residual             16.147   4.018
## Number of obs: 160, groups:  site, 20
##
## Fixed effects:
##             Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)  16.4743     0.5608  13.8574  29.375 7.02e-14 ***
## agec          2.4305     0.4100 108.5317   5.928 3.70e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr)
## agec 0.012
```

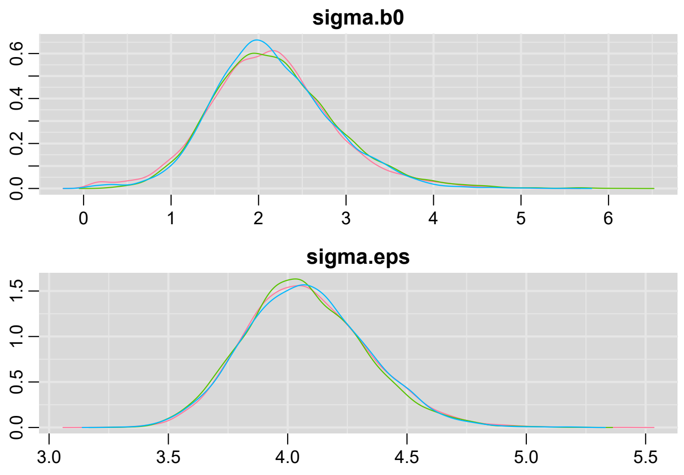
We see that the two sets of results are quite similar. An advantage of the Bayesian implementation, however, is that we can easily estimate uncertainty associated with both site-specific parameters ($\beta_{0i} = \beta_0 + b_{0i}$) as well as the population-level parameters (β_0).

```
MCMCsummary(out.ri, params="beta0i", round=3)
```

```
##               mean    sd   2.5%    50%  97.5% Rhat n.eff
## beta0i[1]  17.991 1.251 15.645 17.967 20.517    1  9402
## beta0i[2]  15.352 1.149 13.064 15.372 17.563    1 22767
## beta0i[3]  18.792 1.422 16.141 18.766 21.667    1  7244
## beta0i[4]  19.141 1.539 16.260 19.093 22.257    1  5787
## beta0i[5]  16.636 1.232 14.283 16.610 19.139    1 24331
## beta0i[6]  18.159 1.557 15.274 18.108 21.351    1 12102
## beta0i[7]  15.977 1.164 13.620 16.005 18.194    1 21720
## beta0i[8]  18.243 1.228 15.914 18.224 20.719    1  8940
## beta0i[9]  15.652 1.236 13.160 15.679 18.043    1 24875
## beta0i[10] 16.188 1.155 13.889 16.201 18.433    1 25820
## beta0i[11] 18.461 1.449 15.773 18.406 21.440    1  8017
## beta0i[12] 14.069 1.330 11.376 14.088 16.573    1  7029
## beta0i[13] 14.750 1.315 12.083 14.777 17.214    1 11749
## beta0i[14] 16.141 1.236 13.733 16.146 18.570    1 27569
## beta0i[15] 16.136 1.138 13.890 16.135 18.391    1 26585
## beta0i[16] 17.963 1.292 15.528 17.927 20.576    1 12329
## beta0i[17] 14.432 1.298 11.778 14.472 16.826    1  8487
## beta0i[18] 14.633 1.209 12.211 14.655 16.903    1  8913
## beta0i[19] 16.400 1.147 14.106 16.413 18.644    1 25307
## beta0i[20] 14.266 1.351 11.529 14.304 16.791    1  7486
```

We can also easily inspect and quantify uncertainty in our variance parameters using their estimated posterior distributions:

```
denplot(out.ri, parms=c("sigma.b0", "sigma.eps"))
```



18.16.2 Random intercept and slope model

When specifying mixed models that include multiple random effects (e.g., site-specific intercepts and slopes), we have to consider how these parameters vary and co-vary. In the simple random intercept and slope model from Section 18.7.2, we had 4 variance parameters: $\sigma_{b_0}, \sigma_{b_1}, Cov(b_0, b_1)$ (or $cor(b_0, b_1)$), and σ_ϵ. We will need priors for each of these parameters. One option is to specify a uniform distribution for $cor(b_0, b_1)$ and priors for each of σ_{b0}, σ_{b1} and σ_ϵ. Together, these priors specify a prior distribution for the 2×2 variance/covariance matrix of the random effects, Σ_b:

$$\Sigma_b = \begin{bmatrix} \sigma^2_{b_0} & Cov(b_0, b_1) \\ Cov(b_0, b_1) & \sigma^2_{b1} \end{bmatrix}$$

We can then specify the distribution of the random effects using `dmnorm` in JAGS, which refers to the multivariate normal distribution. We specify the mean of this distribution using:

$$\texttt{B} = \begin{bmatrix} \beta_{0i} = \beta_0 + b_{0i} \\ \beta_{1i} = \beta_1 + b_{1i} \end{bmatrix}$$

We also need to specify the precision matrix:

$\texttt{Tau.B} = \Sigma_b^{-1}$.

We demonstrate how to implement this model using JAGS, below:

```
jags.lme.rc<-function(){

# Site-specific parameters
 for (i in 1:n.groups){
  # To allow for correlation between alpha[i] and beta1[i], we need to model
  # their joint (multivariate) distribution
    B[i,1:2]~ dmnorm(B.hat[i,], Tau.B[,]) # distribution of the vector (beta0[i], beta1[i])
    B.hat[i,1]<-beta0 # mean for intercepts
    B.hat[i,2]<-beta1 # mean for slopes
    beta0i[i] <- B[i,1] # Random intercepts
    beta1i[i] <- B[i,2] # Random slopes for age
 }

 # Hyperpriors for mean intercept and slope
  beta0 ~ dnorm(0, 0.001)
  beta1 ~ dnorm(0, 0.001)

 # Prior for parameters in Sigma =  var/cov matrix
 # of the slope/intercept parameters
  sigma.b0 ~ dunif(0,100) # sd intercepts
  sigma.b1 ~ dunif(0,100) # sd of slopes
  cor.b0.b1 ~ dunif(-1,1) # correlation among intercepts and slopes
  Sigma.B[1,1]<-pow(sigma.b0,2)
  Sigma.B[2,2]<-pow(sigma.b1,2)
  Sigma.B[1,2]<-cor.b0.b1*sigma.b0*sigma.b1
  Sigma.B[2,1]<-Sigma.B[1,2]

 # Tau = inverse of Sigma (analogous to precision for univariate normal distribution)
  Tau.B[1:2,1:2]<-inverse(Sigma.B[,])

 # Prior for within-site errors
  tau.eps <- 1 / ( sigma.eps * sigma.eps)       # Residual precision
  sigma.eps ~ dunif(0, 100)         # Residual standard deviation

# Likelihood
 for (i in 1:nobs) {
    dbh[i] ~ dnorm(mu[i], tau.eps)
    mu[i] <- beta0i[site[i]] + beta1i[site[i]]*agec[i]
 }
}


# Parameters to estimate
  parameters <- c("beta0", "beta1", "betaoi", "beta1i", "sigma.b0",
                 "sigma.b1", "sigma.eps", "cor.b0.b1")


# Start Gibbs sampling
  out.rc <- jags.parallel(data = jags.data,
                          parameters.to.save=parameters,
                          model=jags.lme.rc,
                          n.thin=1, n.chains=3, n.burnin=100, n.iter=10000)
```

Below, we compare the output from JAGS and `lmer` for the random intercept and slope model:

```
MCMCsummary(out.rc, params=c("beta0", "beta1", "sigma.b0",
                             "sigma.b1", "cor.b0.b1", "sigma.eps"), round=3)
```

```
##              mean    sd   2.5%    50%  97.5% Rhat n.eff
## beta0      16.460 0.617 15.262 16.451 17.718 1.00  6968
## beta1       2.376 0.603  1.247  2.359  3.613 1.00  1301
## sigma.b0    2.009 0.746  0.581  1.976  3.604 1.00  1298
## sigma.b1    1.148 0.683  0.107  1.096  2.659 1.02   737
## cor.b0.b1   0.134 0.479 -0.821  0.169  0.927 1.01   860
## sigma.eps   4.030 0.258  3.561  4.017  4.568 1.00  7246
```

```
summary(lmer.rc)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: dbh ~ agec + (1 + agec | site)
##    Data: pines
##
## REML criterion at convergence: 917.4
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.90307 -0.72242 -0.08194  0.58100  2.93686
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  site     (Intercept)  3.740   1.934
##           agec         1.122   1.059    0.25
##  Residual             15.631   3.954
## Number of obs: 160, groups:  site, 20
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  16.5258     0.5601  29.506
## agec          2.4502     0.4934   4.966
##
## Correlation of Fixed Effects:
##      (Intr)
## agec 0.157
```

We see that posterior means for the variance parameters tend to be larger than the estimates from `lmer`. The posterior medians are closer to the frequentist estimates. The estimated correlation between β_{0i} and β_{1i} also differs considerably between the two methods, but the 95% credible interval for this parameter is quite wide (-0.821, 0.927). Therefore, the data are not very informative with respect to this parameter.

This brings up an important point – if we include additional random effects (e.g., random intercepts associated with additional grouping variables or random slopes associated with additional level-1 covariates), then our covariance matrix describing the random effects will quickly grow in size, requiring the estimation of additional variance and covariance parameters. If, for example, we had a second level-1 covariate and wanted to allow each site to have its own coefficient, we would need to estimate 3 variance parameters (1 for the intercept and 2 for the level-1 coefficients) and 3 covariance parameters describing how the intercept co-varies with the level-1 coefficients and how the 2 level-1 coefficients co-vary. Estimation of these variance parameters can be challenging, unless we have a large number of groups (in this case, sites). Bayesian applications also get more complicated as they require specification of a valid prior for the variance-covariance matrix (see e.g., Chung et al. 2015).

One way to simplify the model is to assume the random intercepts vary independently from the random slopes. Using `lmer`, we can fit this model using the following notation:

```
lmer.rc.ind <- lmer(dbh ~ agec + (1 | site) + (0 + agec | site), data=pines)
summary(lmer.rc.ind)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: dbh ~ agec + (1 | site) + (0 + agec | site)
##    Data: pines
##
## REML criterion at convergence: 917.6
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.92349 -0.73821 -0.08194  0.57261  2.87679
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  site     (Intercept)  3.539   1.881
##  site.1   agec         1.129   1.063
##  Residual             15.678   3.960
## Number of obs: 160, groups:  site, 20
##
## Fixed effects:
##             Estimate Std. Error      df t value Pr(>|t|)
## (Intercept)  16.4512     0.5506 14.0053  29.878 4.38e-14 ***
## agec          2.4094     0.4898 15.2072   4.919 0.000178 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr)
## agec 0.037
```

Here, the `(0 + agec | site)` tells R that we want each site to have its own coefficient for `agec`, but we want the coefficients to vary independently of the intercept. Fitting this model in JAGS is rather simple and only requires priors for the three variance parameters, σ_{b_0}, σ_{b_1}, and σ_ϵ.

19

Generalized linear mixed effects models (GLMMs)

Learning objectives

- Learn how to implement mixed models in R for count and binary data using generalized linear mixed effect models (GLMMs)
- Gain an appreciation for why generalized linear mixed effects can be difficult to fit
- Understand why parameters in mixed-effects models may have a different interpretation when modeling using a non-linear link function
- Learn how to estimate population-level response patterns for count and binary data using GLMMs by integrating over the distribution of the random effects
- Be able to describe models and their assumptions using equations and text and match parameters in these equations to estimates in computer output.

We will begin by briefly reviewing modeling approaches that are appropriate for repeated measures data where the response variable, Y, conditional on predictor variables, X is Normally distributed. These methods include linear mixed-effects models (Chapter 18) and generalized least squares (Section 18.14). We will then consider modeling approaches for repeated measures in cases where $Y|X$ has a distribution other than a Normal distribution, including generalized linear mixed effect models (GLMMs) in this Section, and Generalized Estimating Equations (GEE) in Chapter 20. Throughout, we will draw heavily upon the material in Fieberg et al. (2009), which explores data from a study of mallard nesting structures, including the clutch size data from Chapter 4.

19.1 R packages

We begin by loading a few packages upfront:

```
library(tidyverse) # for data wrangling
library(gridExtra) # for multi-panel plots
library(lme4)  # for fitting random-effects models
library(nlme) # for fitting random-effect and gls models
library(glmmTMB) # for fitting random-effects models
library(modelsummary) # for creating output tables
library(kableExtra) # for tables
options(kableExtra.html.bsTable = T)
library(ggplot2)# for plotting
library(performance) # for model diagnostics
library(ggeffects) # for effect plots
```

In addition, we will use data and functions from the following packages:

- `Data4Ecologists` for the `clutch` data set
- `GLMMadaptive` for fitting generalized linear mixed effect models and estimating marginal response patterns from them
- `car` for multiple degree-of-freedom hypothesis tests using the `Anova` function

19.2 Case study: A comparison of single and double-cylinder nest structures

In the late 90's, the Minnesota Department of Natural Resources (MN DNR) conducted a study to compare the cost-effectiveness of single- and double-cylinder mallard nesting structures and to identify the best places to locate them in the landscape (Zicus, Fieberg, and Rave 2003; Zicus, Rave, and Fieberg 2006; Zicus et al. 2006). A drawing of a single cylinder structure by a former MN DNR employee, Ross Hier, who happens to be a fantastic artist is depicted below (Figure 19.1); a double-cylinder structure would have two cylinders associated with a single pole. To accomplish the project objectives, 110 nest structures (53 single-cylinder and 57 double-cylinder) were placed in 104 wetlands (the largest eight wetlands included two structures each). The structure type (single or double) was randomly chosen for each location. The nest structures were inspected ≥ 4 times annually from 1997-1999 to determine whether or not each nest cylinder was occupied. In addition, clutch sizes were recorded for 139 nests during the course of the study. Thus, we have multiple observations for each nest structure, and we might expect our response variables (structure occupancy or clutch size) from the same structure to be more similar than observations from different structures even after accounting for known covariates.

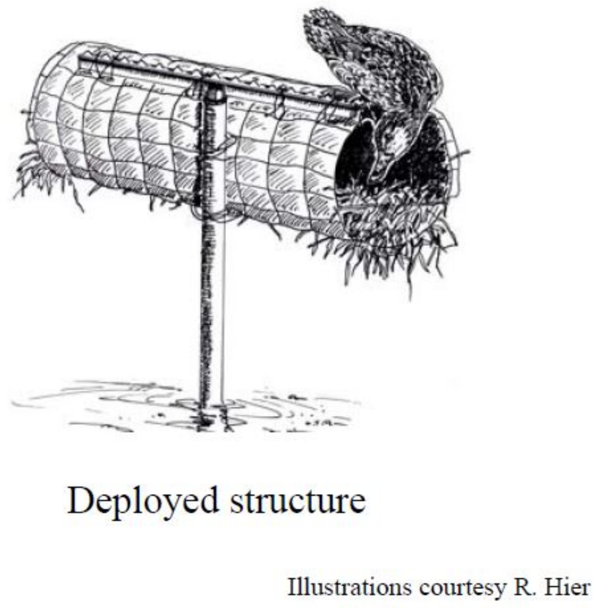


FIGURE 19.1 Single-cylinder mallard nesting structure.

19.3 Review of approaches for modeling correlated data for Normally distributed response variables.

Let's start by considering the clutch size data. Because clutch sizes (number of eggs in the nest) are count data, it is natural to consider a Poisson or Negative Binomial regression model. Nonetheless, the counts are fairly large, and for large counts, a Normal approximation may be reasonable. Note, at the time this study was conducted, software for fitting linear mixed effects models was becoming widely available, but software for fitting generalized linear mixed effects models was not yet well developed. Thus, the analyses in Zicus, Fieberg, and Rave (2003) were conducted using linear mixed-effects models. To avoid having to model the non-linear relationship between clutch size and nest initiation date (Figure 4.1), we will drop outliers associated with nests initiated late in the season. We will also exclude observations with more than 12 eggs as these are likely associated with nests that have been parasitized.

```
library(Data4Ecologists) # data package
data(clutch)

# Get rid of observations with nest initiation dates > 30 May
# and a few outliers representing nests that were likely parasitized
clutch <- clutch %>% filter(CLUTCH < 13 & date < 149) %>%
  mutate(year = as.factor(year)) %>%
  mutate(Ideply = ifelse(DEPLY==2, TRUE, FALSE))
ggplot(clutch, aes(date, CLUTCH, col=Ideply))+
  geom_point() +
  geom_smooth(method= lm, formula = y ~ x) +
  facet_wrap(~year)
```

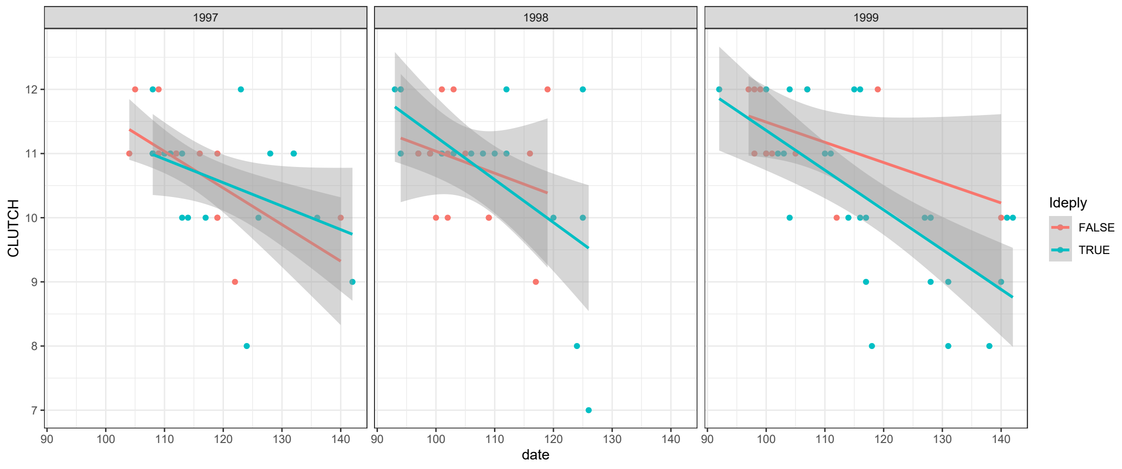


FIGURE 19.2 Clutch sizes in mallard nest boxes in Minneosta versus nest initiation date for 3 separate years and different deployment (single versus double-cylinder) types.

In this data set, we have 2 potential grouping variables, `year` and `strtno`, the latter is a unique identifier for each structure.

Think-Pair-Share: is it better to model the effect of these grouping variables (and any interactions with them) as fixed or random effects? What is the difference?

We only have data from 3 years, so it is best to model variability among years using a set of fixed effects. It would be difficult to estimate a variance parameter for `year` or to generalize to a larger population of years with so little data. On the other hand, we have data from 110 nest structures. It makes more sense to model variability among structures using random effects, which will also allow us to generalize our results to a larger population (of locations where we could place nest structures). Let's consider the following model for the clutch sizes:

$$\begin{aligned} Y_{ij}|b_{0i} &\sim N(\mu_i, \sigma^2_\epsilon) \\ \mu_i &= (\beta_0 + b_{0i}) + \beta_1 Init.Date_{ij} + \beta_2 I(deply = 2)_i \\ b_{0i} &\sim N(0, \sigma^2_{b0}), \end{aligned} \tag{19.1}$$

where Y_{ij} is the clutch size for the i^{th} structure during year j, $Init.Date_{ij}$ = the nest initiation date (Julian day) for the i^{th} structure during year j, and $I(deply = 2)_i = 0$ if i^{th} structure is a single cylinder, and 1 if double cylinder.

How does this model differ from a fixed effects model that might also include `strtno`? A fixed-effects model would use a series of dummy variables to model differences between the nest structures, allowing each structure to have its own intercept. The random effects model is similar, except that we:

1. Assume the intercepts for each structure come from a common Normal distribution.
2. We estimate the variance of the structure-specific intercepts, σ^2_{b0}, rather than separate parameters for each structure (we can, however, *predict* structure-specific intercepts once we have estimates of the other parameters).

```
clutch.ri <- lmer(CLUTCH ~ date + Ideply + (1 | strtno), data=clutch)
summary(clutch.ri)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: CLUTCH ~ date + Ideply + (1 | strtno)
##    Data: clutch
##
## REML criterion at convergence: 283.2
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.81168 -0.44368 -0.03218  0.55879  2.14321
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  strtno   (Intercept) 0.2532   0.5032
##  Residual             0.5755   0.7586
## Number of obs: 106, groups:  strtno, 52
##
## Fixed effects:
##              Estimate Std. Error t value
## (Intercept) 16.093267   0.785919  20.477
## date        -0.047134   0.007023  -6.712
## IdeplyTRUE  -0.219184   0.217833  -1.006
##
## Correlation of Fixed Effects:
##            (Intr) date
## date       -0.978
## IdeplyTRUE  0.053 -0.214
```

Think-Pair-Share: Find all of the model parameters in equation (19.1) in the above output.

19.3.1 Parameter interpretation

In Section 18.10, we discussed two types of response patterns that might be of interest:

Subject-specific mean response patterns (associated with each nest structure):

$$E[Y|Init.date, Ideply, b_{0i}] = (\beta_0 + b_{0i}) + \beta_1 Init.Date_{ij} + \beta_2 I(deply = 2)_i \tag{19.2}$$

Population-averaged mean response patterns (associated with the population of structures):

$$E[Y|Init.date, Ideply] = \beta_0 + \beta_1 Init.Date_{ij} + \beta_2 I(deply = 2)_i \tag{19.3}$$

From the subject-specific (or conditional) mean (eq. (19.2)), we can infer that β_1 describes the difference in the expected clutch size for structure i if the nest initiation date were to be increased by 1 day (and $Ideply$ was held constant). Similarly, β_2 describes the difference in expected clutch size for structure i if it were changed from a single to a double-cylinder structure and nest initiation date was held constant. In the expression for the conditional (subject-specific) mean, b_{0i} can be thought of as a missing covariate associated with structure i (Muff, Held, and Keller 2016). Thus, we need to hold it constant just like any other covariate when interpreting the regression parameters in the model.

From the population-averaged mean (eq. (19.3)), we can also infer that β_1 describes the difference in the expected clutch size across the population of structures as we increase the nest initiation date by 1 day and hold $Ideply$ constant. Similarly, β_2 describes the difference in expected clutch size across the population of double and single-cylinder structures initiated on the same date.

Lastly, we can consider the expected response for a "typical structure", i.e., one with random effects set to their mean values = 0 (eq. (19.4)).

Subject-specific mean response pattern for a typical nest structure:

$$E[Y|Init.date, deply, b_{0i} = 0] = \beta_0 + \beta_1 Init.Date_{ij} + \beta_2 I(deply = 2)_i \tag{19.4}$$

This equation is identical to the one describing the population-average response pattern (eq. (19.3)).

Thus, β_1 and β_2 can be interpreted in terms of the expected clutch size associated with a particular structure, a typical nesting structure, or the population of structures. These equivalencies between population-averaged and subject-specific response patterns fall apart in the case of generalized linear mixed effects models.

19.3.2 Marginal models

Recall that for Normally distributed response data, we can fit correlated data models without having to resort to random effects (see Section 18.14). For example, our random-intercept model is equivalent to a marginal model with compound symmetry in which all observations from the same cluster are assumed to be equally correlated and observations from different clusters are assumed to be independent:

$$Y_i \sim N(\mu_i, \Sigma_i)$$
$$\mu_i = \beta_0 + \beta_1 Init.Date_{ij} + \beta_2 I(deply = 2)_i \tag{19.5}$$
$$\Sigma_i = \begin{bmatrix} \sigma^2 & \rho\sigma^2 & \cdots & \rho\sigma^2 \\ \rho\sigma^2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \rho\sigma^2 \\ \rho\sigma^2 & \cdots & \rho\sigma^2 & \sigma^2 \end{bmatrix} \tag{19.6}$$

where Y_i contains the vector of responses associated with the i^{th} nest structure and Σ_i is the variance-covariance matrix of these observations, which are no longer assumed to be independent. Other options are available for modeling the correlation among observations from the same nest structure. For example, one could model temporal correlation by replacing the $(j, j\prime)$ entry in Σ_i (eq. (19.6)) with $\rho^{|t_{ij}-t_{ij'}|}\sigma^2$, where $t_{ij} - t_{ij'}$ represents the time between the j^{th} and j'^{th} observations.

Under the assumption that observations from different structures are independent, the variance/covariance matrix for the full data set, Σ, will be given by:

$$\Sigma = \begin{bmatrix} \Sigma_i & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \Sigma_i \end{bmatrix} \tag{19.7}$$

One could also consider the potential for spatial correlation, e.g., by replacing the 0's in eq. (19.7) with an assumed spatial covariance structure, say $\sigma^2_d\rho^{d_{ii'}}$, where $d_{ii'}$ represents the spatial distance between the i^{th} and i'^{th} nest structures.

Below, we fit the marginal model with compound symmetry, which is equivalent to the random intercept model:

```
clutch.gls <- gls(CLUTCH ~ date + Ideply,
                       corCompSymm(value = 0.3, form = ~ 1 | strtno), data=clutch)
modelsummary(list("Random intercept" = clutch.ri, "GLS" = clutch.gls),
             title = "Comparison of random intercept and compound symmetry models fit to the clutch size da
             gof_omit = ".*", )  %>%
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

TABLE 19.1 Comparison of random intercept and compound symmetry models fit to the clutch size data set.

	Random intercept	GLS
(Intercept)	16.093	16.093
	(0.786)	(0.786)
date	−0.047	−0.047
	(0.007)	(0.007)
IdeplyTRUE	−0.219	−0.219
	(0.218)	(0.218)
SD (Intercept strtno)	0.503	
SD (Observations)	0.759	

We again see that the parameter estimates and their standard errors are identical for the random intercept and gls models (p-values may differ slightly between the two models depending on how df are approximated; Table 19.1). And, again, we see that the estimate of the intraclass correlation that is returned by `gls` is equivalent to $\frac{\hat{\sigma}^2_{b_0}}{\hat{\sigma}^2_{\epsilon}+\hat{\sigma}^2_{b_0}}$ from the random intercept model, and the residual standard error from the `gls` model is equal to $\sqrt{\hat{\sigma}^2_{\epsilon}+\hat{\sigma}^2_{b_0}}$

```
# Extract variance parameters from the random intercept model
variancepars<-as.data.frame(VarCorr(clutch.ri))

# rho calculated from random intercept model
variancepars[1,4]/(variancepars[1,4]+variancepars[2,4])
```

```
## [1] 0.3055926
```

```
# residual standard error in marginal model
sqrt(variancepars[1,4]+variancepars[2,4])
```

```
## [1] 0.9103319
```

```
# gls estimates
summary(clutch.gls)
```

```
## Generalized least squares fit by REML
##   Model: CLUTCH ~ date + Ideply
##   Data: clutch
##        AIC      BIC    logLik
##   293.2069 306.3806 -141.6035
##
## Correlation Structure: Compound symmetry
##  Formula: ~1 | strtno
##  Parameter estimate(s):
##       Rho
## 0.3055927
##
## Coefficients:
##                  Value Std.Error   t-value p-value
## (Intercept) 16.093267 0.7859185 20.477017  0.0000
## date        -0.047134 0.0070227 -6.711706  0.0000
## IdeplyTRUE  -0.219184 0.2178328 -1.006201  0.3167
##
##  Correlation:
##            (Intr) date
## date       -0.978
## IdeplyTRUE  0.053 -0.214
##
## Standardized residuals:
##         Min          Q1         Med          Q3         Max
## -3.22426210 -0.43920936  0.09712766  0.55853176  2.21646271
##
## Residual standard error: 0.9103319
## Degrees of freedom: 106 total; 103 residual
```

19.4 Extentions to Count and Binary Data

Unlike the multivariate Normal distribution, the Poisson, Negative Binomial, Bernoulli, and Binomial distributions we have used to model count and binary data are *not* parameterized in terms of separate mean and variance parameters. Recall from Chapter 9 that the variance of most commonly used distributions are a function of the same parameters that describe the mean. For example, the mean *and* variance of the Poisson distribution are both equal to λ and the mean and variance of the Bernoulli distribution are a function of p (the mean $= p$ and the variance $= p(1-p)$). Thus, there are no multivariate analogues of these distributions that will allow us to construct marginal models with separate parameters describing the mean and the variance/covariance of binary or count data (as in the previous section).

We will explore 2 options for extending generalized linear models to data containing repeated measures:

1. Add random effects to the linear predictor, leading to generalized linear mixed effect models (Section 19.5).
2. Explicitly model the mean and variance/covariance of our data without assuming a particular distribution. We can then appeal to large sample (asymptotic) results to derive an appropriate sampling distribution of our regression parameter estimators under an assumption that our clusters are independent sample units. This is the Generalized Estimating Equations (GEE) approach (Chapter 20).

As we will see, the former approach leads to a model for the conditional (subject-specific) means whereas the latter approach is parameterized in terms of the population mean. These two means will not, in general, be equivalent when using a non-linear link function (e.g., logit or log).

19.5 Generalized linear mixed effects models

Recall from Chapter 14 that generalized linear models assume that some transformation of the the mean, $g(\mu_i)$, can be modeled as a linear function of predictors and regression parameters:

$g(\mu_i) = \eta_i = \beta_0 + \beta_1 x_1 + \dots \beta_p x_p$

Further, we assume that the response variable has a particular distribution (e.g., Normal, Poisson, Bernoulli, Gamma, etc), $f(y_i;\theta)$, that describes unmodeled variation about $\mu_i = E[Y_i|X_i]$ using one or more parameters, θ:

$Y_i \sim f(y_i;\theta), i = 1,\dots,n$

We can easily add random effects to the model for the transformed mean (e.g., random intercepts and random slopes). In this case, we will have two random components to the model:

1. The random effects, $b \sim N(0,\Sigma_b)$; and
2. The *conditional distribution* of Y_i given the random effects, $Y_i|b \sim f(y_i;\theta,b)$.

It is important to recognize that the *marginal distribution* of Y_i, which is determined by integrating over the random effects, will not generally be the same as the conditional distribution. In fact, we may not even be able to write down an analytical expression for the marginal distribution. This contrasts with the case for linear mixed effects models in which both the conditional and marginal distributions turned out to be Normal

(Section 18.14). These differences have important implications when it comes to model fitting (Section 19.7) and parameter interpretation (Section 19.8).

When describing generalized linear mixed effects models using equations, it is most appropriate to specify the model in terms of the conditional distribution of Y_i given the random effects and the random effects distribution. Examples are given below for a random intercept and slope model with a single covariate, X:

Poisson-normal model:

$$\begin{aligned} Y_{ij}|b_i \sim & \text{ Poisson}(\lambda_{ij}) \\ \log(\lambda_{ij}) = & (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})X_{ij} \\ (b_{0i}, b_{1i}) \sim & N(0, \Sigma_b) \end{aligned}$$

Logistic-normal model:

$$\begin{aligned} Y_{ij}|b_i \sim & \text{ Bernoulli}(p_{ij}) \\ \text{logit}(p_{ij}) = & (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})X_{ij} \\ (b_{0i}, b_{1i}) \sim & N(0, \Sigma_b) \end{aligned}$$

19.6 Modeling nest occupancy using generalized linear mixed effects models

Zicus et al. (2006) modeled the probability that a nesting structure would be occupied as a function of the type of structure (single versus double cylinder; `deply`), the size of the wetland holding the structure (`wetsize`), a measure of the extent of surrounding cover affecting a nest's visibility (`vom`), and the year (`year`) and observation period (`period`). They also included interactions between `year` and `period` and between `period` and `vom`. Lastly, they included a random intercept for nest structure.

Below, we access the data from the `Data4Ecologists` package, convert several variables to factors, and then fit this model using the `glmer` function in the `lme4` package, specifying that we want to use the `bobyqa` optimizer to avoid warnings that appear when using the default optimizer:

```r
# load data from Data4Ecologists package
data(nestocc)

# turn deply, period, year, strtno into factor variables
nestocc$deply <- as.factor(nestocc$deply)
nestocc$period <- as.factor(nestocc$period)
nestocc$year <- as.factor(nestocc$year)
nestocc$strtno <- as.factor(nestocc$strtno)
# remove observations with missing records and fit model
nestocc <- nestocc %>% filter(complete.cases(.))
nest.ri <- glmer(nest ~ year + period + deply + vom +
      year * period + vom * period + poly(wetsize, 2) + (1 | strtno),
      family = binomial(), data = nestocc, control = glmerControl(optimizer = "bobyqa"))
```

Given the large number of fixed effects parameters, it is easiest to describe the model using a combination of text (as in the introduction to this Section) and simplified equations that refer to the predictors that appear in the model's design matrix:

$$Y_{ij}|b_i \sim Bernoulli(p_{ij})$$
$$\log[p_{ij}/(1-p_{ij})] = X_{ij}\beta + b_{0i}$$
$$b_{0i} \sim N(0, \sigma^2_{b_0}),$$

where Y_{ij} is equal to 1 if the i^{th} nest structure was occupied during the j^{th} sampling period and 0 otherwise, p_{ij} represents the probability that structure i is occupied during the j^{th} sampling period, and X_{ij} contains the vector of predictors in the design matrix associated with structure i during the j^{th} sampling period; X_{ij} includes a set of 1's associated with the overall intercept, an indicator variable for whether the structure was a double-cylinder structure (`I(deply==2)`), basis vectors for linear and quadratic effects of wetland size, the level of visual obstruction (`vom`; i.e., how easy it is for predators to see the nest), indicator variables for year and sampling period, and interactions between year and sampling period and between sampling period and visual obstruction. If we wanted to look at the first few observations in the design matrix, we could do so using the `model.matrix` function:

```
nestocc[1:3,]
```

```
##   strtno year period       vom deply nest wetsize
## 1      1 1997      1 0.2255628     2    0      24
## 2      1 1997      2 0.2274146     2    0      24
## 3      1 1997      3 0.8913011     2    0      24
```

```
model.matrix(nest.ri)[1:3,]
```

```
##   (Intercept) year1998 year1999 period2 period3 period4 deply2       vom poly(wetsize, 2)1 poly(wetsize
## 1           1        0        0       0       0       0      1 0.2255628        0.08133736        0.0214
## 2           1        0        0       1       0       0      1 0.2274146        0.08133736        0.0214
## 3           1        0        0       0       1       0      1 0.8913011        0.08133736        0.0214
##   year1999:period3 year1998:period4 year1999:period4 period2:vom period3:vom period4:vom
## 1                0                0                0   0.0000000   0.0000000           0
## 2                0                0                0   0.2274146   0.0000000           0
## 3                0                0                0   0.0000000   0.8913011           0
```

Note that the model considers the effects of 3 different categorical predictors, `year`, `period`, and `strtno`. The first two are modeled using fixed effects with separate intercept parameters to distinguish among the different `year` and `period` combinations. The column of 1's in the design matrix is associated with the overall intercept, which reflects `year` = 1997 and `period` = 1. The columns in the design matrix associated with `year1998` and `year1999` capture differences between years 1998 and 1997 and 1999 and 1997, but only during the reference period, i.e., `period` = 1, due to the interaction between `year` and `period`. Similarly, the columns in the design matrix associated with `period2`, `period3`, and `period4` contrast these periods with period 1 during the reference year, 1997. The columns in the design matrix that code for the interactions capture differences in differences – e.g., `year1998:period2` quantifies how the difference between period 2 and period 1 differ between 1998 and 1997 (the reference year). Or, we can view this parameter as quantifying how the difference between years 1998 and 1997 differ between period 2 and period 1 (the reference period). If you find this discussion confusing, it might help to review design matrices from Chapter 3, but it is not particularly important to understanding the key points of this Section.

Fixed or random effects: We model the effects of `period` and `year` using fixed effects, because there are only 4 periods of interest, and we only observed data from 3 years. Therefore, estimating a variance for `period` makes little sense and would be extremely challenging for `year`. By contrast, we observed data from

110 different nesting structures, making it possible to estimate variability among structures. In addition, we likely want to make inference to the population of structures that could be located in the landscape. Thus, we will model the effect of `strtno` using random effects. We can think of the random intercepts for each structure as capturing unmeasured characteristics associated with the different structures that make them more or less attractive to mallards, and thus, allow the structures to have different *propensities* of being occupied.

Let's now look at the summary output:

```
summary(nest.ri)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace Approximation) ['glmerMod']
##  Family: binomial  ( logit )
## Formula: nest ~ year + period + deply + vom + year * period + vom * period +      poly(wetsize, 2) + (1
##    Data: nestocc
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##    778.0    878.8   -369.0    738.0     1120
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.8896 -0.3290 -0.1902 -0.1138  5.8768
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  strtno (Intercept) 1.91     1.382
## Number of obs: 1140, groups:  strtno, 109
##
## Fixed effects:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -4.2418     0.7103  -5.972 2.35e-09 ***
## year1998             1.2322     0.4409   2.795 0.005194 **
## year1999             1.1736     0.4552   2.578 0.009930 **
## period2              1.5982     0.8295   1.927 0.054011 .
## period3              3.2272     2.2404   1.440 0.149745
## period4              6.1325     2.7091   2.264 0.023593 *
## deply2               0.1077     0.3650   0.295 0.767842
## vom                  5.0353     1.9279   2.612 0.009006 **
## poly(wetsize, 2)1   19.5887     5.9304   3.303 0.000956 ***
## poly(wetsize, 2)2  -16.4711     5.6088  -2.937 0.003318 **
## year1998:period2    -2.2450     0.7007  -3.204 0.001355 **
## year1999:period2    -1.4567     0.6603  -2.206 0.027386 *
## year1998:period3    -1.7489     1.0573  -1.654 0.098104 .
## year1999:period3    -1.3053     1.0439  -1.250 0.211139
## year1998:period4    -1.8957     0.7726  -2.454 0.014137 *
## year1999:period4    -1.5073     0.7328  -2.057 0.039682 *
## period2:vom         -2.9871     2.6575  -1.124 0.260995
## period3:vom         -6.6756     3.1663  -2.108 0.035004 *
## period4:vom         -7.7030     2.3938  -3.218 0.001292 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Correlation matrix not shown by default, as p = 19 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

The summary output is similar to what we saw when using `lmer`. We see that the variance of b_{0i} is given by 1.924 and the standard deviation of the $b_{0i} = \sqrt{Var(b_{0i})} = 1.387$. We also see a list of fixed effects parameters, associated standard errors, and p-values for null hypothesis tests (that the parameters are 0) based on a Normality assumption, which may be reasonable for large samples due to using maximum likelihood (see Section 10.6). Other than the complications associated with interpreting parameters in models with interactions and the need to specify an optimizer other than the default choice, everything seems pretty straightforward on the surface.

19.7 Parameter estimation

In the last section, we specified a different optimizer than the default choice when using `glmer` in order to estimate parameters without a convergence warning. It turns out that estimating parameters in generalized linear mixed effects models is much more challenging than when fitting generalized linear models (Chapter 14) or linear mixed effects models with Normal responses (Chapter 18). If you don't believe me, try fitting the same model using `lmer` (after dropping the `family=binomial()` and `control=glmerControl(optimizer="bobyqa")`) arguments; you will have results almost immediately. Why is it so much more difficult to fit GLMMs than LMMs?

Well, before we can estimate model parameters, we have to determine the *marginal* likelihood of the data. We specify the GLMM in terms of a *conditional* model, $Y_{ij}|b_i$. To determine the marginal distribution of Y_{ij}, we need to integrate over the random effects distribution:

$$L(\beta, \Sigma_b; Y_{ij}) = \prod_{i=1}^{n} \int f(Y_{ij}|b_i) f(b_i) db_i \tag{19.8}$$

where β contains our fixed effects parameters, Σ_b are variance parameters associated with the random-effects distribution, $f(Y_{ij}|b_i)$ is the conditional distribution of Y given the random effects, and $f(b_i)$ is the distribution of the random effects; Typically, $f(Y_{ij}|b_i)$ will be a distribution from the exponential family (e.g., Bernoulli, Poisson), and $f(b_i) \sim N(0, \Sigma_b)$.

Think-Pair-Share: write down the different components of the likelihood for the nest occupancy model.

For our nest occupancy model, we have:

- $f(Y_{ij}|b_i) \sim$ Bernoulli$(p_{ij}) = p_{ij}^{Y_{ij}}(1-p_{ij})^{1-Y_{ij}}$, with logit$(p_{ij}) = X_{ij}\beta + b_{0i} \Rightarrow p_{ij} = \frac{\exp(X_{ij}\beta + b_{0i})}{1+exp(X_{ij}\beta + b_{0i})}$
- $f(b_i) \sim N(0, \sigma_b^2) = \frac{1}{\sqrt{2\pi}\sigma_b} \exp\left(\frac{b_{0i}^2}{2\sigma_b^2}\right)$

Importantly, if we plug these distributions into our equation for the likelihood (eq. (19.8)), we find that there is no closed form, analytical solution – i.e., we can't write down an expression for the likelihood that does not involve an intractable integral. So, how does R estimate the parameters of the model? We have 3 options:

1. Approximate the integrand, i.e., the expression for the likelihood, $L(Y_{ij}|\beta, D)$, by a multivariate Normal distribution. This leads to a closed form, known solution. This approach is referred to as the *Laplace approximation*.

2. Approximate the integral using simulation-based or numerical integration techniques. These approaches can be slow, particularly with multiple random effects since the integral needs to be solved numerically for each iteration of the numerical optimization routine.
3. Add priors, and use Bayesian techniques to approximate the posterior distribution of β and Σ_b.

Most users of `glmer`, and GLMMs more generally, are blissfully unaware of these challenges except when they try to fit models that return convergence warnings. If, however, you look at the help page for `glmer`, you will see that there is an argument, `nAGQ`, with the following description:

> integer scalar - the number of points per axis for evaluating the adaptive Gauss-Hermite approximation to the log-likelihood. Defaults to 1, corresponding to the Laplace approximation. Values greater than 1 produce greater accuracy in the evaluation of the log-likelihood at the expense of speed. A value of zero uses a faster but less exact form of parameter estimation for GLMMs by optimizing the random effects and the fixed-effects coefficients in the penalized iteratively reweighted least squares step. (See Details.)

Further down in the details section, you will also see this point addressed:

> The expression for the likelihood of a mixed-effects model is an integral over the random effects space. For a linear mixed-effects model (LMM), as fit by lmer, this integral can be evaluated exactly. For a GLMM the integral must be approximated. The most reliable approximation for GLMMs is adaptive Gauss-Hermite quadrature, at present implemented only for models with a single scalar random effect. The nAGQ argument controls the number of nodes in the quadrature formula. A model with a single, scalar random-effects term could reasonably use up to 25 quadrature points per scalar integral.

Again, the reason we did not encounter this issue with linear mixed effects models is that for a linear mixed-effects model (LMM), as fit by lmer, this integral can be evaluated exactly (the marginal distribution is, in this case, given by a Normal distribution - see 18.14).

19.8 Parameter interpretation

In Section 19.3.1, we discussed how fixed-effects parameters in linear mixed-effects models can be interpreted in terms of changes in subject-specific means, population-averaged means, or the mean associated with a "typical" subject/site/individual with all random effects set to 0 (Fieberg et al. 2009). In generalized linear mixed effects where the random effects enter on a transformed scale, this equivalence will no longer hold in general.

More specifically:

- We cannot determine population-averaged response patterns by setting all random effects to 0 and then solving for $\mu_{ij} = g^{-1}(X\beta)$. This approach will, however, give us the mean for a "typical" individual, i.e., an individual with all random effects equal to 0.
- When interpreting parameters and their impact on means, we need to hold the random effects, b_i, constant. With non-linear transformations of the mean, parameters will then only have a subject-specific interpretation.
- To determine population-averaged means, we need to integrate over the random-effects distribution. Similar to our discussion regarding the marginal likelihood (see Section 19.7), there will often be no analytical expression for the population mean.

Consider, for example, the simplified version of our nest occupancy model given below:

$$\begin{gathered} Y_i|b_i \sim Bernoulli(p_i) \\ \log[p_{ij}/(1-p_{ij})] = \beta_0 + b_{0i} + \beta_1 VOM_{ij} + \beta_2 I(deply = 2)_i \\ b_{0i} \sim N(0, \sigma^2_{b_0}) \end{gathered}$$

In this case, $\exp(\beta_1)$ represents the change in the odds of structure occupancy if we change VOM by 1 unit at a *particular structure* while keeping the structure's deployment type (single or double cylinder) constant. Similarly, $\exp(\beta_2)$ estimates the change in the odds of structure occupancy if we were to change the structure from a single to double cylinder model at a *particular location* with VOM held constant. The interpretation here focuses on a particular structure, since we have to hold both b_{0i} and all other fixed effects constant. As we will see shortly, these conditional (i.e., subject-specific) odds ratios will be larger than marginal (population-averaged) odds ratios.

It is important to note that there may be cases where a subject-specific interpretation is not of interest or may not even make sense. Consider, e.g., models that include the sex of individual animals in a study. Although many fish species can change their sex, we may be more interested in comparing differences between males and females at the population rather than individual level. In Section 19.8.1, we will discuss methods for estimating marginal means using parameters from fitted generalized linear mixed effects models, including a few specific cases where we can derive analytical expressions for marginal means from fitted GLMMs. We will also discuss ways to approximate marginal means using numerical integration techniques.

First, let's explore differences between conditional and marginal means using a simple simulation framework involving a binary regression model with a random intercept and a single covariate. Consider a group of first graders learning how to play hockey for the first time[1]. One of the key challenges they will face is learning how to skate backwards. The more times they skate, the more likely they will be able to skate backwards without falling, but individuals may have more or less experience with ice skating before they begin the season. Let $Y_{it} = 1$ if individual i can skate the width of the ice backwards without falling at time t and 0 otherwise. Assume these data were collected pre-covid and that all skaters attended 10 practices. Thus, let $X_{it} = t$ be the number of practices that individual i has attended by time t. We will assume the probability of successfully skating backwards depends on the number of practices and can be modeled as:

$$\begin{gathered} Y_{it}|b_i \sim Bernoulli(p_{it}) \\ \log[p_{it}/(1-p_{it})] = \beta_0 + b_{0i} + \beta_1 X_{it} \\ b_{0i} \sim N(0, \sigma^2_{b_0}) \end{gathered}$$

The random intercept, b_{0i}, accounts for among-individual differences in their initial abilities and experiences with skating. Let's simulate 10 observations for each of 100 individuals, setting:

[1]Why hockey? Well, I'm writing this section during my sabbatical from a rink where my daughter has hockey practice.

- $\beta_0 = -6$
- $\beta_1 = 1$
- $\sigma_{b_0} = 2$

```r
# ## Simulation set up
set.seed(1200)

# Sample size
nindividuals <- 100 # number of individuals
nperindividual <- 10 # number of observations per individual
ntotal <- nindividuals * nperindividual # total number of observations

# Data generating parameters
b0_i <- rnorm(nindividuals, 0, 2) # deviations from mean intercept
beta_0 <- -6 # mean intercept
beta_1 <- 1 # slope for number of practices

# Generate 10 different observations for each individual
# for practices 1 through 10
practice.num <- seq(1, 10, length = 10)
individual <- rep(1:nindividuals, each = nperindividual) # individual id

# Y|boi is Bernouli p[ij]
logitp <- beta_0 + rep(b0_i, each = nperindividual) + practice.num * beta_1
p <- plogis(logitp)
Y <- rbinom(ntotal, size = 1, prob = p)

# Create data set for model fitting
skatedat <- data.frame(Y = Y, practice.num = practice.num, individual = individual)
```

We can then fit the GLMM used to generate the data and show that we can recover the simulation parameters:

```r
glmermod <- glmer(Y ~ practice.num + (1 | individual), family = binomial(), data = skatedat)
summary(glmermod)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace Approximation) ['glmerMod']
##  Family: binomial  ( logit )
## Formula: Y ~ practice.num + (1 | individual)
##    Data: skatedat
##
##      AIC      BIC   logLik deviance df.resid
##    766.9    781.6   -380.5    760.9      997
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.0829 -0.2764 -0.0439  0.2765  6.4474
##
## Random effects:
##  Groups     Name        Variance Std.Dev.
##  individual (Intercept) 4.909    2.216
## Number of obs: 1000, groups:  individual, 100
```

```
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -5.94865    0.48505  -12.26   <2e-16 ***
## practice.num  0.99702    0.07088   14.07   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## practice.nm -0.862
```

For example, the true parameters are contained within Wald-based confidence intervals formed using a Normality assumption.

```
confint(glmermod, method="Wald")
```

```
##                   2.5 %    97.5 %
## .sig01               NA        NA
## (Intercept)  -6.8993197 -4.997975
## practice.num  0.8580956  1.135941
```

Now, let's plot the average value of Y at each time point – i.e., the proportion of the 100 skaters that can successfully skate backward at each practice (Figure 19.3). We will also plot the predicted mean response curve formed by setting the random effect, $b_{0i} = 0$. The latter response curve corresponds to the predicted probability that a "typical skater" (i.e., the median individual) will be able to skate backwards at each practice.

```
# Predictions for a "typical subject" with b0i=0
practice.num <- seq(1, 10, length = 10)
mod <- data.frame(practice.num = practice.num)
mod$glmer.logit <- fixef(glmermod)[1] + practice.num * fixef(glmermod)[2]
mod$glmer.p <- plogis(mod$glmer.logit)

# Summarize average Y (i.e., proportion of skaters that successfully
# skate backwards) at each practice
mdat <- skatedat %>%
  group_by(practice.num) %>%
  dplyr::summarize(PopProportion = mean(Y))

# Combine two curves and plot
mdat2 <- data.frame(
  practice.num = rep(practice.num, 2),
  p = c(mod$glmer.p, mdat$PopProportion),
  type = rep(c("Probability, typical individual", "Sample proportion"), each = 10)
)

# Plot
ggplot(mdat2, aes(practice.num, p, col=type)) +
  geom_line() + geom_point() +
  xlab("Number of practices") +
  ylab("Proportion of skaters that can skate backwards")
```

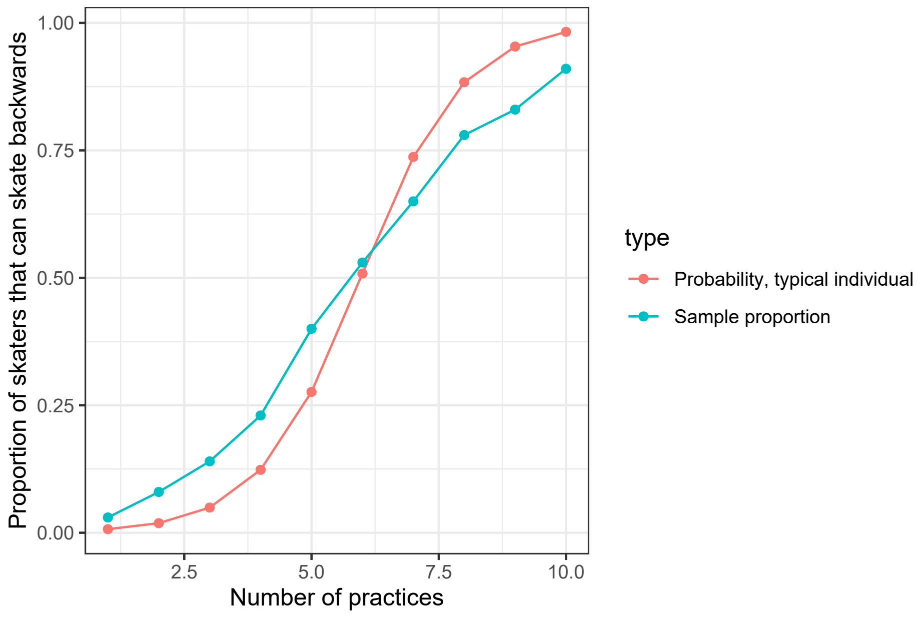


FIGURE 19.3 Sample proportion of skaters that can skate backwards as a function of the number of practices they have attended. Also plotted is the response curve for a typical individual with $b_{0i} = 0$.

We see that the two curves do not line up – the predicted probability of skating backwards for a "typical individual" (with $b_{0i} = 0$) differs from the proportion of the sample that can skate backwards. This occurs because of the non-linear logit transformation, $E[Y|X] \neq E[Y|X, b_{0i} = 0]$. To understand this mismatch, we next compare:

- the mean response curve for each individual on both the logit and probability scales
- the mean of these individual-specific curves on both the logit and probability scales
- the overall mean population response curve on both the logit and probability scales

We begin by computing the individual-specific curves, below:

```
pdat <- NULL
for (i in 1:nindividuals) {
  logitp.indiv <- beta_0 + b0_i[i] + practice.num * beta_1 # individual i's curve on logit scale
  p.indiv <- plogis(logitp.indiv) # individual i's curve on the probability scale
  tempdata <- data.frame(p = p.indiv,
                         logitp = logitp.indiv,
                         individual = i)
  pdat <- rbind(pdat, tempdata)
}
pdat$practice.num<-practice.num
```

We then calculate the average of the individual-specific curves on both the logit and probability scales:

```
pop.patterns<-pdat %>% group_by(practice.num) %>%
  dplyr::summarize(meanlogitp=mean(logitp), meanp=mean(p))
```

Lastly, we compare the individual-specific (i.e., subject-specific or conditional means; in black) to their average (in red) and to the population-average response curves (in blue) on both logit and probability response scales (Figure 19.4).

```
m1 <- ggplot(pdat, aes(practice.num, logitp)) +
  geom_line(aes(group = individual)) + ylab("logit p") +
  geom_line(data = pop.patterns, aes(practice.num, meanlogitp), col = "blue", linewidth = 1.2) +
  geom_line(data = mod, aes(practice.num, glmer.logit), col = "red", linewidth = 1.2, linetype = "dashed")
  theme(legend.position = 'none') + xlab("Practice Number") +
  ylab(expression(Logit(p))) + ggtitle("A)")

m2 <- ggplot(pdat, aes(practice.num, p)) +
  geom_line(aes(group = individual)) + ylab("p") +
  geom_line(data = pop.patterns, aes(practice.num, meanp), col = "blue", size = 1.2) +
  geom_line(data = mod, aes(practice.num, glmer.p), col = "red", linewidth = 1.2, linetype = "dashed") +
  theme(legend.position = 'none') + xlab("Practice Number") +
  ylab(expression(p[i])) + ggtitle("B)")
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was generated.
```

```
grid.arrange(m1, m2, ncol = 2)
```

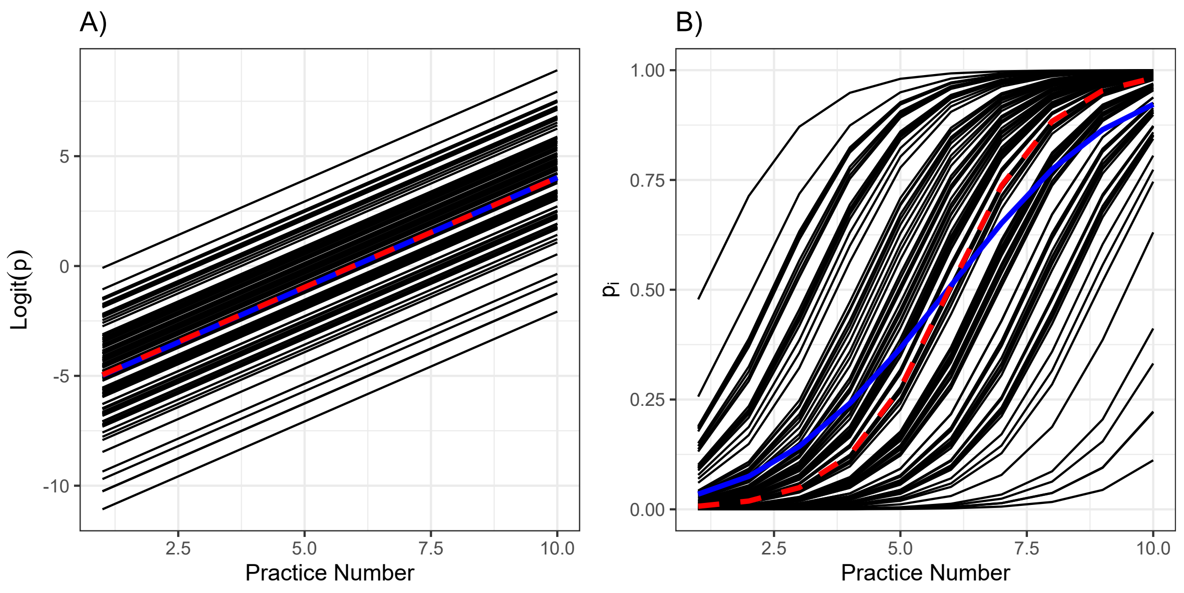


FIGURE 19.4 Individual response curves (black), the response curve for a typical individual with $b_{0i} = 0$ (red), and the population mean response curve (blue) and on the logit and probability scales.

Let's begin by looking at the predicted curves on the logit scale (Figure 19.4A). As expected, the logit probabilities of skating backwards increase linearly with the number of practices. Further, when we average the individual curves on the logit scale, we find that average of the individual-specific curves (in blue) and the curve for a typical individual with $b_{0i} = 0$ (in red) are identical (Figure 19.4A).

Now, let's look at the same curves but on the probability scale (Figure 19.4B). We see that all individuals improve at the same rate (governed by β_1), but that they have different initial abilities (determined by b_{0i}). Thus, some individuals are able to skate backwards successfully by the 5th practice while others still have a low probability of skating backwards after 10 practices (Figure 19.4B). The average skater (with $b_{0i} = 0$), plotted in red, is extremely unlikely to skate backwards successfully after the first practice, but is nearly guaranteed to be skating backwards by the end of the season. When we average these curves, we find that there is a small, but non-negligible proportion of skaters that can skate backwards after the first practice and similarly, a small and non-negligible proportion of skaters that cannot skate backwards after 10 practices. The population mean curve (in blue) is flatter than the curve for a typical individual (in red). This will always be the case when fitting a logistic regression model with a random intercept – the population mean response curve will be attenuated with slope that is smaller in absolute value than the curve formed by setting all random effects equal to 0 (Zeger, Liang, and Albert 1988; Fieberg et al. 2009).

If we are interested in directly modeling how the population mean (or proportion)[2] changes as a function of one or more variables, we can consider Generalized Estimating Equations (Chapter 20). Alternatively, we can use the methods in the next section to approximate population mean response curves from fitted GLMMs.

19.8.1 Approximating marginal (population-average) means from fitted GLMMs

As discussed in Section 19.7, to calculate the marginal distribution (and thus, likelihood) of the data, we have to integrate over the distribution of the random effects. Unfortunately, there will not typically be a closed form solution to the integral that allows us to derive an analytical expression for the marginal distribution. Similarly, we will need to integrate over the distribution of the random effects to determine the marginal mean, $E[Y|X] = \sum_y y f_{Y,b}(y) = \sum_y y \int f_{Y|b}(y) f_b(b)db$. Except for a few special cases (see Section 19.8.2), there will not generally be a closed form solution for this expression. Thus, we must resort to the same set of approaches discussed in Section 19.7 to derive an expression for the marginal mean. Specifically, we can:

1. Approximate the integrand, $\int f_{Y|b}(y) f_b(b)db$, in a way that gives us a closed form or known solution.
2. Use numerical integration techniques to solve the integral.
3. Use simulation techniques to approximate the solution to the integral.

19.8.2 Special cases

There are a few special cases where we can derive expressions for the marginal (population) means from the subject-specific parameters estimated when fitting generalized linear mixed effects models (Ritz and Spiegelman 2004). We list a few of the more notable cases here. We already highlighted that setting $b_{0i} = 0$ in **linear mixed effects models** will give us an expression for the marginal (population-average) mean (Section 19.3.1); this equivalence holds more generally for models with an identity link. Another important example is the case where a random intercept is added to a Poisson regression model:

$$\begin{gathered} Y_{ij}|b_{0i} \sim Poisson(\lambda_{ij}) \\ \log(\lambda_{ij}) = X_{ij}\beta + b_{0i} \\ b_{0i} \sim N(0, \sigma_{b_0}) \end{gathered}$$

In this case, the population-averaged response pattern is determined by adjusting the intercept: $\beta_0^M =$

[2]The mean of a binary variable **is** a proportion. To convince yourself, take the mean of (0, 1, 0).

$\beta_0^C + \sigma^2_{b_0}/2$, where β_0^M and β_0^C are the intercepts in the expression for the marginal and conditional means, respectively, and $\sigma^2_{b_0}$ is the variance of the random intercept. The slope parameters will not need adjusting and have both conditional (subject-specific) and marginal (population-averaged) interpretations. We can demonstrate this result with another simple simulation example, below.

Assume 100 students in an ornithology class from the University of Minnesota go birding 20 times during the course of the semester. Each time out, they look for birds for between 30 min and 2 hours, recording the time they spent surveying to the nearest 5 minutes. Let Y_{ij} = the number of birds seen by student i during the j^{th} observation period and X_{ij} be the time (in hours) they spent looking for birds during the j^{th} observation period. Individuals will have different skill levels, which we represent using a Normally distributed random intercept, b_{0i}. We will assume, conditional on b_{0i}, that the number of birds observed can be described by a Poisson distribution with mean depending on the log number of hours of observation. Thus, we simulate data from the following model:

$$\begin{gathered}
Y_{ij}|b_{0i} \sim Poisson(\lambda_{ij}) \\
\log(\lambda_{ij}) = \beta_0 + b_{0i} + \log(X_{ij})\beta_1 \\
b_{0i} \sim N(0, \sigma_{b_0}),
\end{gathered}$$

with parameters:

- $\beta_0 = -1.5$
- $\beta_1 = 0.2$
- $\sigma_{b_0} = 1.5$

```r
# Simulation set up
# Generate 100 random intercepts
set.seed(610)

# Sample size
nindividuals <- 100 # number of individuals
nperindivid <- 10 # number of observations per individual
ntotal <- nindividuals * nperindivid # total number of observations

# Data generating parameters
b0_i <- rnorm(nindividuals, 0, 1.5) # deviations from mean intercept
beta_0 <- -1.5 # mean intercept
beta_1 <- 0.2 # slope for time

# Generate 10 different observations for each individual Here, x will
# record the observation time to the nearest 5 minutes
lobs.time <- log(sample(seq(30, 120, by = 5), size = ntotal, replace = TRUE))

# Y|boi is Poisson lambda[ij]
loglam <- beta_0 + rep(b0_i, each = nperindivid) + lobs.time * beta_1
lam <- exp(loglam)
Y <- rpois(ntotal, lam)

# Create data set for model fitting
individual <- rep(1:nindividuals, each = nperindivid) # Cluster id
birdobs <- data.frame(Y = Y, lobs.time = lobs.time, individual = individual)
```

We then fit the GLMM corresponding to the data generating model and show that we are able to recover the parameter values used to simulate the data.

```r
# ## Fit glmer model
# Estimates are close to values used to simulate data
fitsim <- glmer(Y ~ lobs.time + (1 | individual), family = poisson(), data = birdobs)
summary(fitsim)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace Approximation) ['glmerMod']
##  Family: poisson  ( log )
## Formula: Y ~ lobs.time + (1 | individual)
##    Data: birdobs
##
##      AIC      BIC   logLik deviance df.resid
##   2382.9   2397.6  -1188.4   2376.9      997
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.4449 -0.5947 -0.3465  0.3592  4.5086
##
## Random effects:
##  Groups     Name        Variance Std.Dev.
##  individual (Intercept) 1.924    1.387
## Number of obs: 1000, groups:  individual, 100
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.95290    0.33671  -5.800 6.63e-09 ***
## lobs.time    0.31214    0.07018   4.448 8.68e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##           (Intr)
## lobs.time -0.895
```

```r
cbind(confint(fitsim), trueparam = c(1.5, -1.5, 0.2))
```

```
## Computing profile confidence intervals ...
```

```
##                  2.5 %     97.5 % trueparam
## .sig01       1.1759471  1.6614586       1.5
## (Intercept) -2.6309957 -1.2907895      -1.5
## lobs.time    0.1733807  0.4525831       0.2
```

Next, we compute and plot the following:

- the estimated mean response curve for a typical student, $E[Y_{ij}|X_{ij}, b_{0i} = 0] = \exp(\beta_0 + \beta_1 \log(X_{ij}))$
- the estimated mean response curve for the population of students, $E[Y_{ij}|X_{ij}] = \exp(\beta_0 + \sigma^2_{b_0}/2 + \beta_1 \log(X_{ij}))$
- the mean number of birds observed in the population of students as a function of log(observation time)

```
# Summarize the mean number of birds seen as a function of lobs.time at the population-level
mdatb <- birdobs %>% group_by(lobs.time) %>% dplyr::summarize(meanY = mean(Y))

# Now, add on appropriate conditional and population average mean
# response curves
mdatb$ss <- exp(fixef(fitsim)[1] + mdatb$lobs.time * fixef(fitsim)[2])
mdatb$pa <- exp(fixef(fitsim)[1] + as.data.frame(VarCorr(fitsim))[1,4] / 2 +
                mdatb$lobs.time * fixef(fitsim)[2])

# plot
ggplot(mdatb, aes(lobs.time, meanY)) + geom_point() +
  geom_line(aes(lobs.time, ss), color="red") +
  geom_line(aes(lobs.time, pa), color="blue") +
  xlab("log(Observation time)") + ylab("Number of birds seen")
```

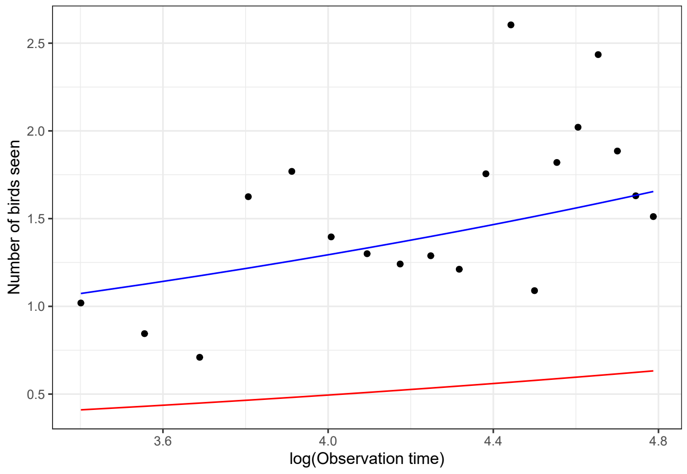


FIGURE 19.5 Comparison of the mean response curve for a typical student (red), formed by setting $b_{0i} = 0$, estimated from a Poisson generalized linear mixed effects model (GLMM) with the mean response curve for the population of students (blue) estimated by adjusting the intercept from the fitted GLMM. Points represent the empirical means at each log(observation) time.

Examining Figure 19.5, we see that the mean response curve for the "typical student", formed by setting $b_{0i} = 0$ (in red) underestimates the number of birds seen in the population of students. However, the mean response curve derived by adjusting the intercept (blue curve) fits the data well. This adjustment is only appropriate when fitting Poisson GLMMs that include only a random intercept but no random slope terms.

Another important case where we can derive expressions for the marginal mean from a fitted random-effects model is when using Probit regression (eq. (19.9); Ritz and Spiegelman 2004; Young et al. 2007):

$$\begin{aligned} Y_{ij}|b_{0i} &\sim Bernoulli(p_i) \\ p_i &= \Phi(X_{ij}\beta + b_{0i}), \end{aligned} \tag{19.9}$$

where Φ is the cumulative distribution function of the standard Normal distribution (and can be calculated using the `pnorm` function in R). In this case, marginal parameters, β_M can be derived from conditional parameters β_C and the random effects variance component, $\sigma^2_{b_0}$ using:

$$\beta^M = \frac{\beta^C}{\sqrt{\sigma^2_{b_0} + 1}}$$

Lastly, we note that Zeger, Liang, and Albert (1988) derived a useful approximation for describing marginal (i.e., population average) mean response patterns using the parameters estimated from a generalized linear mixed effects model in the case of logistic regression with a random intercept:

$$\beta^M = (\sigma^2_{b_0} 0.346 + 1)^{-1/2} \beta^C \qquad (19.10)$$

This approximation works well for our simulated hockey player data (Figure 19.6).

```
sigma2b0 <- as.data.frame(VarCorr(glmermod))[1, 4]
betaM_0 <- fixef(glmermod)[1]/sqrt(sigma2b0 * 0.346 +1)
betaM_1 <- fixef(glmermod)[2]/sqrt(sigma2b0 * 0.346 +1)
mdat3<-rbind(mdat2,
             data.frame(practice.num = practice.num,
                        p = plogis(betaM_0 + betaM_1 * practice.num),
                        type = rep("Zeger approximation", 10)))

# Plot
ggplot(mdat3, aes(practice.num, p, col=type)) +
  geom_line() + geom_point() +
  xlab("Number of practices") +
  ylab("Proportion of skaters that can skate backwards")
```

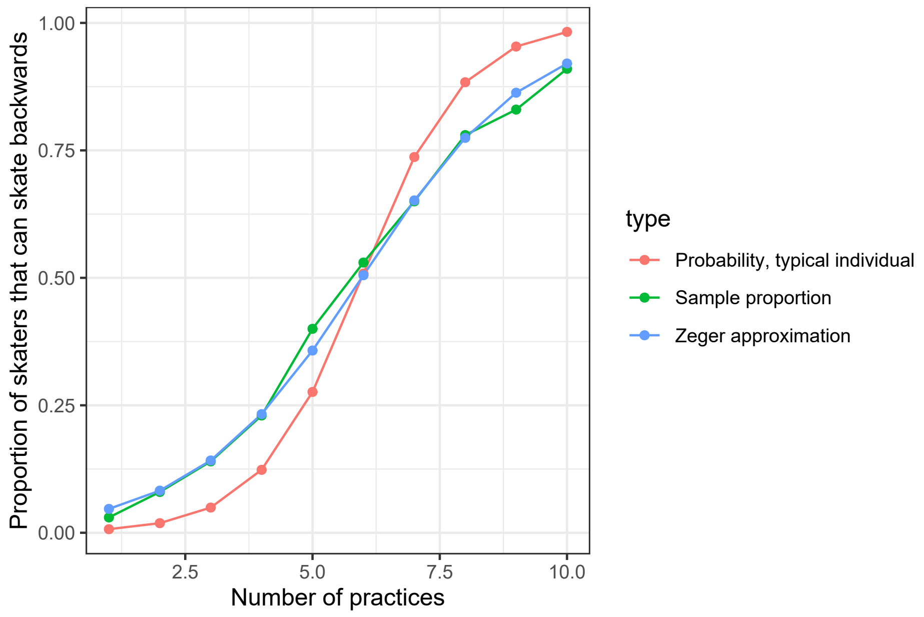


FIGURE 19.6 Population mean response curve using Zeger's approximation (eq. (19.10); blue) versus the response curve for a typical individual formed by setting $b_{0i} = 0$ (green). Population proportions are indicated by the red curve.

Again, this approximation only works for logistic regression models containing random intercepts but not random slopes. Alternatively, we can use numerical integration to average over the distribution of the random effects and thereby determine the marginal mean response curve.

19.8.3 Numerical integration

Recall, to determine the marginal mean response curve, we need to integrate over the distribution of the random effects:

$$E[Y|X] = \sum_y y f_{Y,b}(y) = \sum_Y y \int f_{Y|b}(y) f_b(b) db \tag{19.11}$$

For binary regression models, note that Y can take only 1 of 2 values (0 and 1), and only the value of 1 contributes a term to the expected value (eq. (19.11)). Furthermore, when $Y = 1$, $f_{Y|b}(y) = \frac{e^{\beta_0+b+\beta_1 X}}{1+e^{\beta_0+b+\beta_1 X}}$. Lastly, $f_b(b)$ is a Normal distribution with mean 0 and variance $\sigma^2_{b_0}$. Thus, we need to compute:

$$E[Y|X] = \sum_y y f_{Y,b}(y) = \int \frac{e^{\beta_0+b+\beta_1 X}}{1+e^{\beta_0+b+\beta_1 X}} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{b^2}{2\sigma^2_{b_0}}} db$$

We could make use of the `integrate` function in R to approximate this integral, as we demonstrate below for our model describing how quickly young hockey players learn how to skate backwards. We then plot the result (Figure 19.7).

```r
# Matrix to hold marginal mean response curve, E[Y/X]
pa.rate<-matrix(NA,10,1)
  for(i in 1:10){
     intfun<-function(x){
       plogis(fixef(glmermod)[1] + x + practice.num[i]*fixef(glmermod)[2])*dnorm(x,0,sqrt(sigma2b0))}
       pa.rate[i]<-integrate(intfun,-Inf, Inf)[1]
   }
mdat4<-rbind(mdat2,
             data.frame(practice.num = practice.num,
                        p = unlist(pa.rate),
                        type = rep("Marginal mean by Integration", 10)))

# Plot
ggplot(mdat4, aes(practice.num, p, col=type)) +
  geom_line() + geom_point() +
  xlab("Number of practices") +
  ylab("Proportion of skaters that can skate backwards")
```

Again, we see that we are able to approximate the population proportions well once we translate the fit of our GLMM to a prediction about the marginal mean response curve.

19.8.4 Simulation

Rather than use numerical integration, we could solve for marginal response patterns using a simulation approach, where we:

1) generate a large sample of random effects from the fitted random-effects distribution, $N(0, \hat{\sigma}^2_{b_0})$;

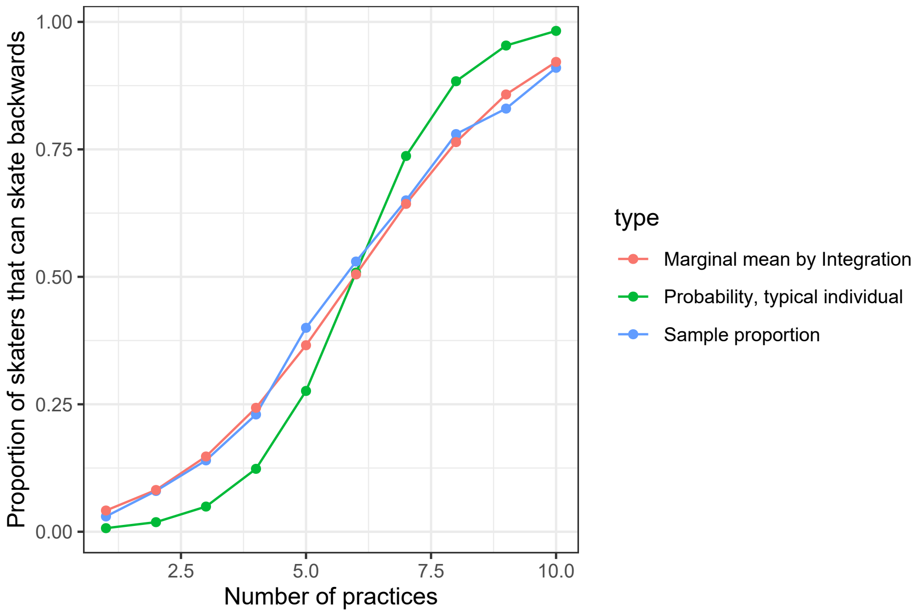


FIGURE 19.7 Population mean response curve determined by numerical integration (red) versus the response curve for a typical individual, formed by setting $b_{0i} = 0$ (green). Population proportions are indicated by the blue curve.

2) generate the conditional, subject-specific curves for each of these randomly chosen individuals;
3) average the subject-specific curves from step 2 to estimate the marginal mean response curve.

We leave this as a potential exercise.

19.8.5 Approximating marginal means using GLMMadpative

In Sections 19.8.3 and 19.8.4, we discussed approaches for piecing together marginal response curves that essentially estimate $E[Y_i|X_i]$ at a set of unique values of X_i chosen by the user. We can then plot $E[Y_i|X_i]$ versus X_i to examine these marginal (population-average) response curves. However, these approaches do not give us a set of marginal coefficients for summarizing the effects of X_i at the population level. In Section 19.8.2, we saw a few special cases where we could derive or approximate *marginal* or population-level regression parameters from the fitted GLMM. In this section, we will explore the `GLMMadaptive` package (Rizopoulos 2021), which provides a function, `marginal_coefs`, that estimates coefficients describing marginal response patterns for a wider variety of GLMMS.

We begin by fitting the same GLMM to our simulated skating data, but using the `mixed_model` function in the `GLMMadaptive` package. Its syntax is similar to that of `lme` in the `nlme` package:

```
library(GLMMadaptive)
fit.glmm <- mixed_model(fixed = Y ~ practice.num,
  random = ~ 1 | individual,
  family = binomial(link = "logit"),
  data = skatedat
)
summary(fit.glmm)
```

```
##
## Call:
## mixed_model(fixed = Y ~ practice.num, random = ~1 | individual,
##     data = skatedat, family = binomial(link = "logit"))
##
## Data Descriptives:
## Number of Observations: 1000
## Number of Groups: 100
##
## Model:
##  family: binomial
##  link: logit
##
## Fit statistics:
##    log.Lik      AIC      BIC
##  -379.886 765.772 773.5875
##
## Random effects covariance matrix:
##               StdDev
## (Intercept) 2.230948
##
## Fixed effects:
##              Estimate Std.Err  z-value p-value
## (Intercept)   -5.9444  0.4860 -12.2303 < 1e-04
## practice.num   0.9960  0.0709  14.0464 < 1e-04
##
## Integration:
## method: adaptive Gauss-Hermite quadrature rule
## quadrature points: 11
##
## Optimization:
## method: hybrid EM and quasi-Newton
## converged: TRUE
```

We see that we get very similar answers as to when we fit the model using the `glmer` function. The `mixed_model` function uses numerical integration to approximate the likelihood (see Section 19.7), but uses more quadrature points than `glmer` does by default, and thus, should result in a more accurate approximation of the likelihood. In addition, the `glmer` function only uses numerical integration when fitting models with a single random intercept, whereas the `mixed_model` function can perform numerical integration for models that include random intercepts and random slopes. Another advantage of `mixed_model` over `glmer` is that it can fit zero-inflated Poisson and Negative Binomial models that allow for random effects in the zero part of the model; `glmmTMB` also has this capability, but it only uses the Laplace approximation when fitting models (see Section 19.7). On the other hand, the `mixed_model` function only allows for a single grouping factor for the random effects whereas `glmer` and `glmmTMB` can fit models using multiple grouping factors (e.g., one could include `(1 | individual)` and `(1 | year)` for an analysis of data that included many individuals followed for many years).

The primary reason we will explore `mixed_model` in the `GLMMadaptive` package is because of its ability to provide coefficient estimates, $\hat{\beta}^M$, for describing marginal response patterns. The `marginal_coefs` function estimates these coefficients via the following method (Hedeker et al. 2018):

1. Marginal means at the observed X_i are estimated using Monte Carlo simulations (Section

19.8.4). Specifically, a large number of random effects are generated from a Normal distribution, $b^{sim} \sim N(0, \hat{\Sigma}_b)$. These simulated random effects are then used to estimate subject-specific means, $\hat{E}[Y_i|X_i, b^{sim}]$. The subject-specific means are then averaged to form population-averaged means, $\hat{E}[Y_i|X_i]$.

2. Marginal coefficients, β^M, are estimated by fitting a linear model with logit$(\hat{E}[Y_i|X_i])$ as the response variable and X_i as the predictor variable. This results in the following estimator:

$$\hat{\beta}^M = (X'X)^{-1}X'\text{logit}(\hat{E}[Y_i|X_i]), \quad (19.12)$$

Below, we calculate these marginal coefficients and compare them to our estimates using numerical integration, the estimated mean curve for a "typical skater" (formed by setting $b_{0i} = 0$), and to the population proportions at each practice number (Figure 19.8).

```
marginal_beta <- marginal_coefs(fit.glmm, std_errors = FALSE)
mdat5<-rbind(mdat4,
             data.frame(practice.num = practice.num,
                        p = plogis(marginal_beta$betas[1]+practice.num * marginal_beta$betas[2]),
                        type = rep("GLMMadaptive", 10)))

# Plot
ggplot(mdat5, aes(practice.num, p, col=type)) +
  geom_line() + geom_point() +
  xlab("Number of practices") +
  ylab("Proportion of skaters that can skate backwards")
```

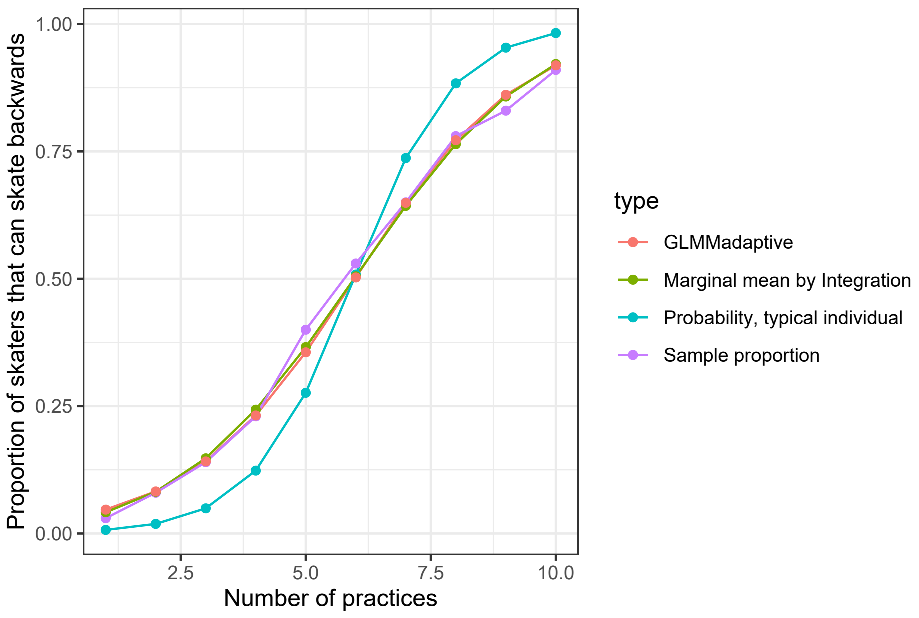


FIGURE 19.8 Population mean response curve determined by numerical integration (olive green) and using the `marginal_coef` function (red) from the `GLMMadaptive` package versus the response curve for a typical individual (formed by setting $b_{0i} = 0$; aqua blue). Population proportions are given in purple.

Again, we see that we are able to recover marginal (population-average) response patterns using this approach, but with the advantage of being able to report population-level odds ratios:

```
exp(marginal_beta[1]$betas[2])
```

```
## practice.num
##     1.830493
```

I.e., we can report that the odds of skating backwards increases by a factor of 1.8 at the population level for each additional practice. This odds ratio will always be smaller than the corresponding subject-specific odds ratio that holds b_{0i} constant:

```
exp(fixef(glmermod)[2])
```

```
## practice.num
##     2.710188
```

Again, the difference is expected since the subject-specific curves rise faster than the population average curve (Figure 19.4B).

19.9 Hypothesis testing and modeling strategies

Methods for selecting an appropriate model and testing hypotheses involving regression parameters in generalized linear mixed effects models typically rely on an asymptotic Normality assumption. Thus, we do not have to worry about various degrees-of-freedom approximations, and there is no REML option to consider. On the other hand, inference is more reliant on large samples. In addition, the challenges associated with fitting GLMMS (see Section 19.7) often limits the complexity of the models we can fit, particularly with binary data which tend to be less informative than continuous data (e.g., see Section 8.5).

Below, we reconsider our model for the mallard nest occupancy data and show how the `Anova` function can be used to test hypotheses regarding the fixed effects parameters:

```
library(car)
Anova(nest.ri)
```

```
## Analysis of Deviance Table (Type II Wald chisquare tests)
##
## Response: nest
##                    Chisq Df Pr(>Chisq)
## year              0.8804  2  0.6438999
## period            2.0199  3  0.5682928
## deply             0.0871  1  0.7678415
## vom               0.0028  1  0.9579025
## poly(wetsize, 2) 17.9392  2  0.0001272 ***
## year:period      13.3628  6  0.0376227 *
## period:vom       10.5617  3  0.0143483 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on the significant interactions, it seems reasonable to stick with the full model for inference. We would conclude that nest occupancy rates depend on `vom`, but that the relationship between `voc` and nest occupancy varies by `period`. We might also consider using functions in the `ggeffects` package to explore the non-linear relationship between wetland size and nest occupancy (Figure 19.9).

```
plot(ggpredict(nest.ri, "wetsize [all]"))
plot(ggeffect(nest.ri, "wetsize [all]"))
```

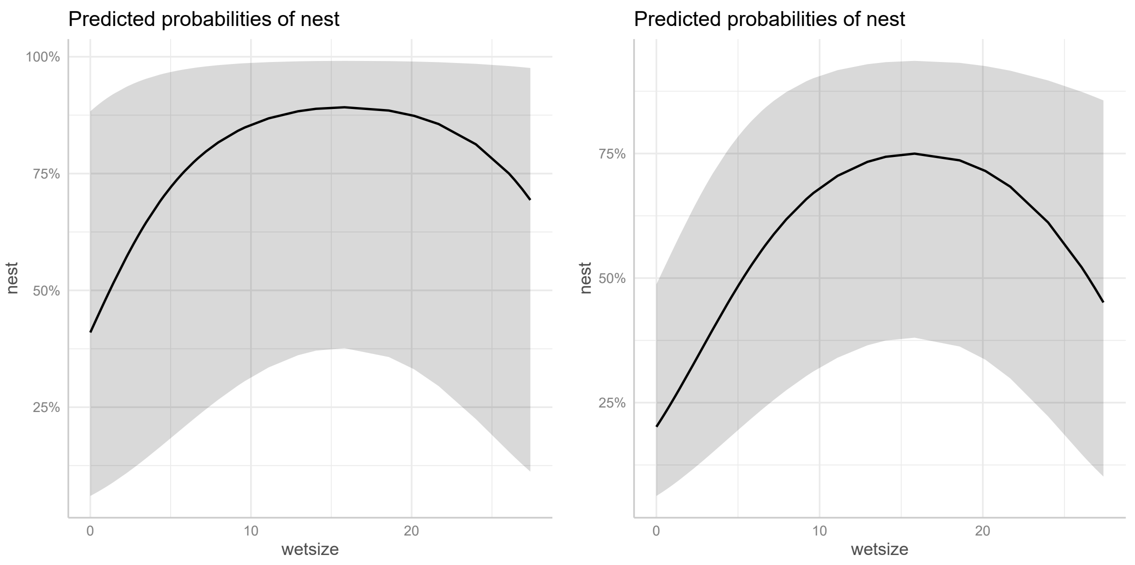


FIGURE 19.9 Effect plots using the `ggpredict` (left panel) and `ggeffect` (right panel) functions in the `ggeffects` package (Lüdecke 2018) depicting the relationship between wetland size (`wetsize`) on nest occupancy rates inferred from the generalized linear mixed effects model fit to the data from Zicus et al. (2006).

To understand what is plotted, we can look at the output from the `ggpredict` and `ggeffect` functions.

```
ggpredict(nest.ri, "wetsize [all]")
```

```
## # Predicted probabilities of nest
##
## wetsize | Predicted |       95% CI
## ----------------------------------
##    0.00 |      0.41 | [0.06, 0.88]
##    0.23 |      0.43 | [0.06, 0.89]
##    0.63 |      0.46 | [0.07, 0.90]
##    1.37 |      0.51 | [0.09, 0.92]
##    3.14 |      0.63 | [0.13, 0.95]
##    5.05 |      0.72 | [0.19, 0.97]
##    7.19 |      0.80 | [0.25, 0.98]
##   27.39 |      0.69 | [0.11, 0.98]
##
## Adjusted for:
## *   year = 1997
## * period =    1
## *  deply =    1
## *    vom = 0.89
## * strtno = 0 (population-level)

##
## Not all rows are shown in the output. Use ‘print(..., n = Inf)‘ to show all rows.
```

The output from `ggpredict` is relatively easy to understand – these are predictions formed by setting all covariates other than the focal covariate (`wetsize`) to specific values listed below the output. The random effect is also set to 0, implying that these are predictions for a "typical" subject when `year` = 1997, `period` = 1, `deply` = 1 (single cylinder), and `vom` = 0.89.

Now, let's look at the output from `ggeffect`.

```
ggeffect(nest.ri, "wetsize [all]")
```

```
## # Predicted probabilities of nest
##
## wetsize | Predicted |       95% CI
## ----------------------------------
##    0.00 |      0.20 | [0.06, 0.49]
##    0.23 |      0.21 | [0.07, 0.50]
##    0.63 |      0.23 | [0.08, 0.53]
##    1.37 |      0.28 | [0.09, 0.58]
##    3.14 |      0.38 | [0.14, 0.69]
##    5.05 |      0.49 | [0.20, 0.79]
##    7.19 |      0.59 | [0.26, 0.86]
##   27.39 |      0.45 | [0.10, 0.86]

##
## Not all rows are shown in the output. Use ‘print(..., n = Inf)‘ to show all rows.
```

The output from `ggeffect` is more difficult to decipher as it is averaging over the values of the other covariates in some way. I also suspect that it is setting the random effect to 0. We can see this if we compare our earlier plots for the ice skating example to the output from `ggeffects` (Figure 19.10). The response pattern from `ggeffects` most closely matches the response curve for a typical individual formed by setting the random effect equal to 0.

```
 ggplot(mdat2, aes(practice.num, p, col=type)) +
     geom_line() + geom_point() +
     xlab("Number of practices") +
     ylab("Proportion of skaters that can skate backwards") +
     theme(legend.position = "bottom", legend.box = "horizontal")
plot(ggeffect(glmermod))
```

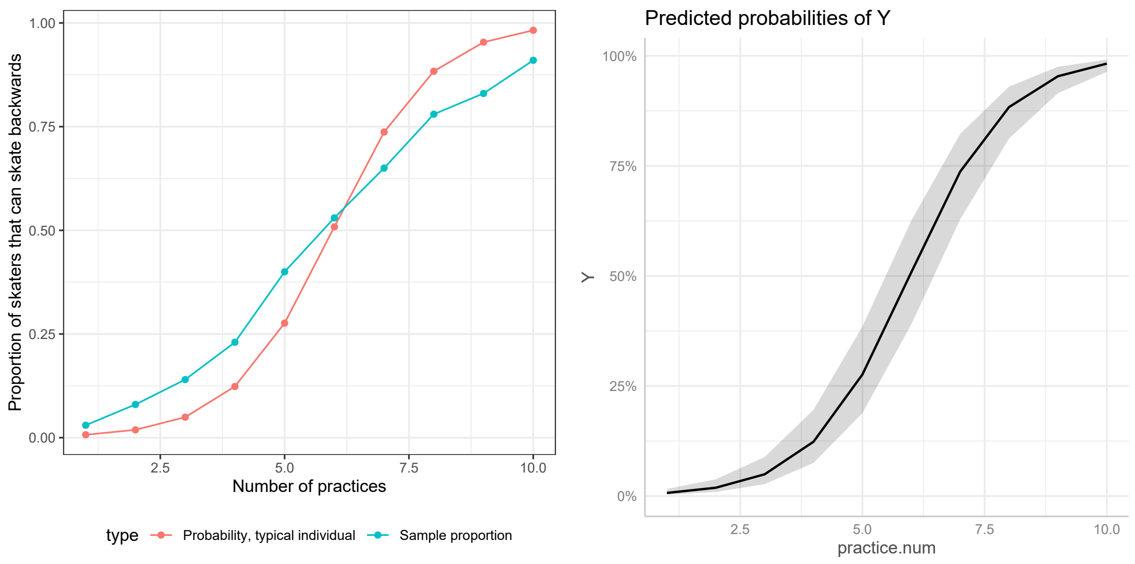


FIGURE 19.10 Comparison of effect plot created using the `ggeffects` function (right panel) to the response curve for a typical individual, formed by setting $b_{0i} = 0$ (left panel, red). Population proportions are indicated by the aqua blue curve in the left panel.

20

Generalized Estimating Equations (GEE)

Learning Objectives

- Learn how to model population-level response patterns for count and binary data using Generalized Estimating Equations (GEEs)

20.1 R packages

We begin by loading a few packages upfront:

```r
library(geepack) # for estimating parameters using GEEs
library(lme4) # for fitting glmms
library(ggplot2) # for plotting
```

20.2 Introduction to Generalized Estimating Equations

In the last section, we saw how the parameters in typical GLMMs have a subject-specific interpretation. We also saw how we could estimate mean-response patterns at the population level by integrating over the distribution of the random effects. Lastly, we saw how we could use the `marginal_coef` function in the `GLMMadaptive` package to estimate parameters describing marginal (i.e., population-average) mean-response patterns from fitted GLMMS.

There are times, however, when we may be interested in directly modeling how the population mean changes as a function of our predictor variables. For Normally distributed response variables, we can use the `gls` function in the `nlme` package, which allows us to specify various correlation structures for our residuals (Section 19.3.2). Yet, as we highlighted in Section 19.4, multivariate distributions for count and binary data do not have separate mean and variance parameters, making it challenging to fit marginal models to binary or count data in the case where we have repeated measures. Simply put, there is no multivariate Poisson or Binomial distribution that contains separate parameters for describing correlation among observations from the same cluster, so we cannot take a simple marginal modeling approach as we saw for Normally distributed data (see Sections 18.14 and 19.3.2).

Generalized Estimating Equations (GEEs) offer an alternative approach to modeling correlated binary or count data in which one specifies a model for the first two moments (the mean and variance) of the data but not the full marginal distribution (Zeger, Liang, and Albert 1988; Hilbe and Hardin 2008; Fieberg et al. 2009). Parameters are estimated by solving:

$$\sum_{i=1}^{n}\frac{\partial\mu_i}{\partial\beta}V_i^{-1}(Y_i-\mu_i)=0, \text{where} \tag{20.1}$$

$Y_i = (Y_{i1}, Y_{i2}, \ldots, Y_{im_i})$ denotes the vector of response data for individual i, $\mu_i = g^{-1}(X_i\beta)$ represents the mean response as a function of covariates and regression parameters, and $\frac{\partial\mu_i}{\partial\beta}$ is an $m_i \times p$ matrix of first derivatives of μ_i with respect to the regression parameters, β. The variance-covariance matrix for subject i, V_i, is typically modeled by adopting a common form for the variance based on an appropriate member of the exponential family, combined with a "working correlation model" to describe within-subject dependencies. We can write $V_i = A_i^{1/2}R_i(\alpha)A_i^{1/2}$ where A_i captures the assumed standard deviations, and R_i is our working correlation model that describes within subject correlation and may include additional parameters, α. It is common to assume $A_i = \sqrt{\mu_i(1-\mu_i)}$ for binary data and $A_i = \sqrt{\mu_i}$ for count data. Typical examples of working correlation assumptions include independence, compound symmetry (see Section 18.14), or a simple (ar1) auto-regressive model for temporal data.

Importantly, estimates of the regression parameters will be (asymptotically) unbiased even when the working correlation and variance model is mis-specified, provided that the model for the marginal mean is correct and the mechanisms responsible for any missing data or sample-size imbalance can be assumed to be completely random (Zeger, Liang, and Albert 1988); proper specification of V_i should increase precision. Separate analyses using multiple working correlation matrices could be used as a sort of sensitivity analysis, or one can use various tools (e.g., lorelograms; Heagerty and Zeger 1998; Iannarilli et al. 2019) to determine an appropriate correlation structure. The sampling distribution of the regression parameter estimators will approach that of a Normal distribution as the number of clusters (i) goes to infinity. Robust("sandwich") standard errors that assume clusters are independent are typically used when forming confidence intervals or conducting hypothesis tests (White 1982; Liang and Zeger 1986).

20.3 Motivating GEEs

You may be wondering where equation (20.1) comes from. In Section 10.11 we saw that the maximum likelihood estimator for the linear model is equivalent to the least-squares estimator, which finds the values of β that minimize:

$$\sum_{i=1}^{n}\frac{(Y_i-(\beta_0+X_{1i}\beta_1+\ldots))^2}{2\sigma^2}=\sum_{i=1}^{n}\frac{(Y_i-\mu_i)^2}{2\sigma^2} \tag{20.2}$$

Taking the derivative with respect to β and setting the result to 0, gives:

$$\sum_{i=1}^{n}\frac{(Y_i-\mu_i)}{\sigma^2}\frac{\partial\mu_i}{\partial\beta}=0, \text{ or equivalently,} \sum_{i=1}^{n}\frac{\partial\mu_i}{\partial\beta}V_i^{-1}(Y_i-\mu_i)=0, \tag{20.3}$$

where $V_i = Var[Y_i|X_i] = \sigma^2$. Similarly, for generalized linear models (e.g., logistic and Poisson regression), it turns out that maximum likelihood estimators are found by solving:

$$\sum_{i=1}^{n}\frac{\partial\mu_i}{\partial\beta}V_i^{-1}(Y_i-\mu_i)=0 \tag{20.4}$$

To demonstrate this fact, we will derive this result for the logistic regression model, which assumes:

$$Y_i \sim Bernoulli(\mu_i) \tag{20.5}$$

$$\mu_i = g^{-1}(X_i\beta) = \frac{e^{x_i\beta}}{1+e^{x_i\beta}} \tag{20.6}$$

First, note that for the logistic regression model:

$$\frac{\partial \mu_i}{\partial \beta} = \frac{X_i e^{X_i\beta}\left(1+e^{X_i\beta}\right) - X_i e^{X_i\beta} e^{X_i\beta}}{\left(1+e^{X_i\beta}\right)^2} \tag{20.7}$$

$$= \frac{X_i e^{X_i\beta}}{\left(1+e^{X_i\beta}\right)^2} \tag{20.8}$$

Furthermore:

$$V_i = \mu_i(1-\mu_i) = \left(\frac{e^{X_i\beta}}{1+e^{X_i\beta}}\right)\left(1-\frac{e^{X_i\beta}}{1+e^{X_i\beta}}\right) \tag{20.9}$$

$$= \left(\frac{e^{X_i\beta}}{1+e^{X_i\beta}}\right)\left(\frac{1}{1+e^{X_i\beta}}\right) \tag{20.10}$$

$$= \frac{e^{X_i\beta}}{\left(1+e^{X_i\beta}\right)^2} \tag{20.11}$$

Thus, $\frac{\partial \mu_i}{\partial \beta} V_i^{-1} = X_i$ and equation (20.4) can be simplified to:

$$\sum_{i=1}^{n} X_i \left(Y_i - \frac{e^{X_i\beta}}{1+e^{X_i\beta}}\right) = 0 \tag{20.12}$$

We can show that we end up in the same place if we take the derivative of the log-likelihood with respect to β and set the expression to 0. The likelihood of the data is thus given by:

$$L(\beta; Y) = \prod_{i=1}^{n} \mu_i^{Y_i}(1-\mu_i)^{1-Y_i} \tag{20.13}$$

$$= \prod_{i=1}^{n} \left(\frac{e^{X_i\beta}}{1+e^{x_i\beta}}\right)^{Y_i} \left(1-\frac{e^{X_i\beta}}{1+e^{X_i\beta}}\right)^{1-Y_i} \tag{20.14}$$

$$= \prod_{i=1}^{n} \left(\frac{e^{X_i\beta}}{1+e^{X_i\beta}}\right)^{Y_i} \left(\frac{1}{1+e^{X_i\beta}}\right)^{1-Y_i} \tag{20.15}$$

$$= \prod_{i=1}^{n} \left(e^{x_i\beta}\right)^{Y_i} \left(\frac{1}{1+e^{X_i\beta}}\right) \tag{20.16}$$

Thus, the log-likelihood is given by:

$$logL(\beta; Y) = \sum_{i=1}^{n} Y_i(X_i\beta) - \sum_{i=1}^{n} \log\left(1+e^{X_i\beta}\right) \tag{20.17}$$

Taking the derivative of the log-likelihood with respect to β and setting the expression equal to 0 gives us:

$$\frac{\partial logL(\beta;Y)}{\partial\beta} = \sum_{i=1}^{n} X_iY_i - \sum_{i=1}^{n} \frac{1}{1+e^{X_i\beta}} e^{X_i\beta} X_i = 0 \tag{20.18}$$

$$= \sum_{i=1}^{n} X_i \left(Y_i - \frac{e^{X_i\beta}}{1+e^{X_i\beta}} \right) = 0, \tag{20.19}$$

which is the same as equation (20.12).

If you found this exercise to be fun, you might follow up by proving that the maximum likelihood estimator for Poisson regression is also equivalent to equation (20.4).

What is the point of all this? The equations used to estimate parameters in linear models and generalized linear models take a similar form. Generalized estimating equations (eq. (20.1)) extend this basic idea to situations where we have multiple observations from the same cluster. We just replace V_i with a variance covariance matrix associated with the observations from the same cluster.

20.4 GEE applied to our previous simulation examples

There are a few different packages for fitting GEEs in R. We will explore the `geeglm` function in the `geepack` package (Yan 2002; Halekoh, Højsgaard, and Yan 2006). Let's revisit the data from our hockey skaters in Chapter (glmm). Similar to random intercept models, let's assume the observations from the same skater are all `exchangeable`, giving a compound symmetry correlation structure:

$$R(\alpha) = \begin{bmatrix} 1 & \alpha & \cdots & \alpha \\ \alpha & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \alpha \\ \alpha & \cdots & \alpha & 1 \end{bmatrix} \tag{20.20}$$

Further, as with logistic regression, we will assume:

- $E[Y_i|X_i] = \frac{e^{X_i\beta}}{1+e^{X_i\beta}}$
- $Var[Y_i|X_i] = E[Y_i|X_i](1 - E[Y_i|X_i])$

We specify this assumption using the `family = binomial()` argument. Note, however, that we are *not* assuming our data come from a binomial distribution. We only adopt the mean and variance assumptions from logistic regression. Furthermore, in contrast to GLMMs with a random intercept (from Section 19). GEEs can be used to model the *population mean* as a logit-linear function of predictors and regression parameters. To make sure we account for the repeated measures aspect of the data, we have to specify a clustering variable, `individual` in this case, using `id = individual`. "

```
fit.gee.e <- geeglm(Y ~ practice.num, family = binomial(),
                    id = individual, corstr = "exchangeable",
                    data = skatedat)
summary(fit.gee.e)
```

```
##
## Call:
## geeglm(formula = Y ~ practice.num, family = binomial(), data = skatedat,
##     id = individual, corstr = "exchangeable")
##
##  Coefficients:
##              Estimate Std.err  Wald Pr(>|W|)
## (Intercept)   -3.4833  0.2708 165.4   <2e-16 ***
## practice.num   0.5960  0.0421 200.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation structure = exchangeable
## Estimated Scale Parameters:
##
##             Estimate Std.err
## (Intercept)    0.979  0.1663
##   Link = identity
##
## Estimated Correlation Parameters:
##       Estimate Std.err
## alpha   0.2619 0.06906
## Number of clusters:   100  Maximum cluster size: 10
```

If we compare the coefficients from the GLMM with a random intercept to the coefficients from the Generalized Estimating Equations approach, we find that the latter estimates are closer to 0.

```r
fixef(glmermod)
```

```
##  (Intercept) practice.num
##       -5.949        0.997
```

This result was to be expected since the GLMM estimates subject-specific parameters whereas we are directly modeling the population-averaged response pattern using GEEs. We also see that unlike the GLMM, the GEE does a nice job of estimating the proportion of skaters in the population that can skate backwards at each practice (Figure 20.1).

```r
mdat6<-rbind(mdat5,
             data.frame(practice.num = practice.num,
                        p = plogis(marginal_beta$betas[1]+practice.num * marginal_beta$betas[2]),
                        type = rep("GEE", 10)))

# Plot
ggplot(mdat6, aes(practice.num, p, col=type)) +
  geom_line() + geom_point() +
  xlab("Number of practices") +
  ylab("Proportion of skaters that can skate backwards")
```

Similarly, a GEE with an exchangeable working correlation structure does a nice job of estimating the population-mean-response pattern for our simulated birders (Figure 20.2:

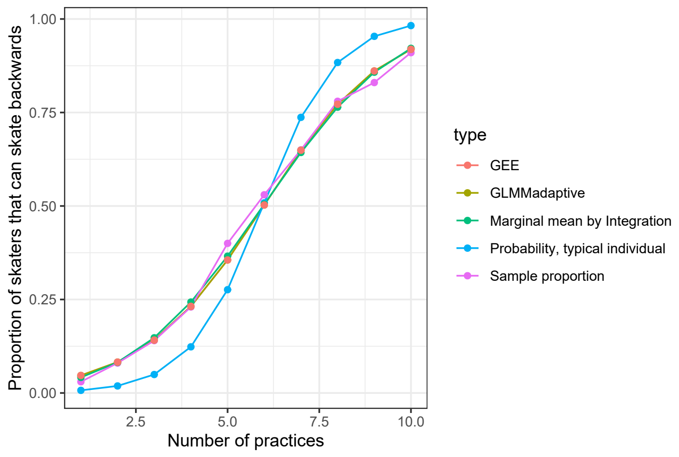


FIGURE 20.1 Population mean response curve determined by numerical integration (green), using the `marginal_coef` function (olive green) from the `GLMMadaptive` package, and using Generalized Estimating Equations (red) versus the GLMM response curve for a typical individual (formed by setting $b_{0i} = 0$; purple). Population proportions are given in blue.

```
geebird <- geeglm(Y ~ lobs.time, family = poisson(),
                  id = individual, corstr = "exchangeable", data = birdobs)

# Summarize the mean number of birds seen as a function of lobs.time at the population-level
mdatb$gee <- exp(coef(geebird)[1] + mdatb$lobs.time*coef(geebird)[2])

# plot
ggplot(mdatb, aes(lobs.time, meanY)) + geom_point() +
  geom_line(aes(lobs.time, gee), color="black") +
  xlab("log(Observation time)") + ylab("Number of birds seen")
```

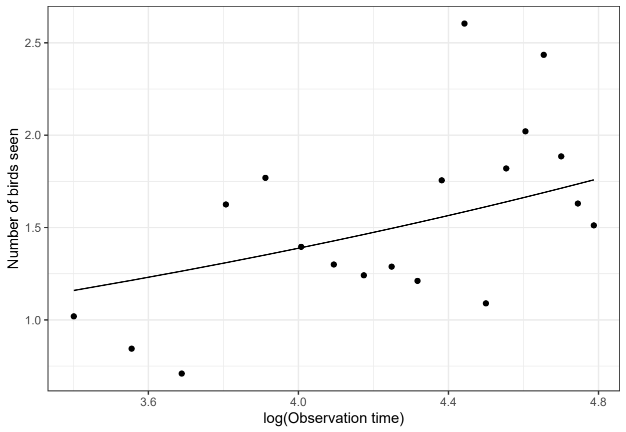


FIGURE 20.2 Population mean response curve describing the number of birds seen as a function of log observation time in the population of students. Mean response curve was estimated using a Generalized Estimating Equation with exchageable working correlation. Points are the empirical means at each unique value of log(observation time).

20.5 GEEs versus GLMMs

Given that GLMMs and GEEs estimate parameters that have different interpretations, much of the literature that aims to provide guidance on choosing between these methods focuses on the estimation target of interest – i.e., whether one is interested in describing subject-specific response patterns (GLMM) or population-average response patterns (GEE). Yet, we saw that it is possible to estimate population-average response patterns from GLMMs if we integrate over the distribution of the random effects, so the choice of estimation target may not be a deciding factor when choosing between GEEs and GLMMs. In this short section, we further contrast the two approaches in hopes of providing additional guidance to practitioners.

1. **Complexity**: GLMMS require approximating an integral in the likelihood before parameters can be estimated (Section 19.7). This makes GLMMs much more challenging to fit than GEEs. On the other hand, GLMMs can describe much more complex hierarchical data structures (e.g., clustering due to multiple grouping variables – such as individual and year), whereas GEEs typically only allow for a single clustering variable.
2. **Robustness**: GLMMs require distributional assumptions (e.g., Normally distributed random effects, a distribution for Y conditional on the random effects). GEEs, by contrast, only make assumptions about the mean and variance/covariance of the data, and estimates of regression parameters should be robust to incorrect assumptions about the variance/covariance structure. On the other hand, GEEs have more strict requirements when it comes to missing data. Whereas standard applications of GEE require data to be missing completely at random, likelihood approaches are still appropriate if missingness depends only on the individual's observed data (i.e. their covariates and their other response values) and not on data that were not observed (e.g. the missing response) (Carrière and Bouyer 2002; Roderick JA Little and Rubin 2019).
3. **Tools for model selection and inference**: since GLMMs are likelihood based, we have a number of methods that can be used to compare and select an appropriate model (e.g., AIC, likelihood ratio tests). We can also assess goodness-of-fit using simulation-based methods. Furthermore, we can simulate future responses and estimate prediction intervals when using GLMMs. Since GEEs only make assumptions about the mean and variance of the data, we cannot use likelihood-based methods for comparing models. In addition, GEEs typically require large numbers of clusters so that a Normal approximation is appropriate for the sampling distribution of regression parameter estimators.

A

Projects and Reproducible Reports in R

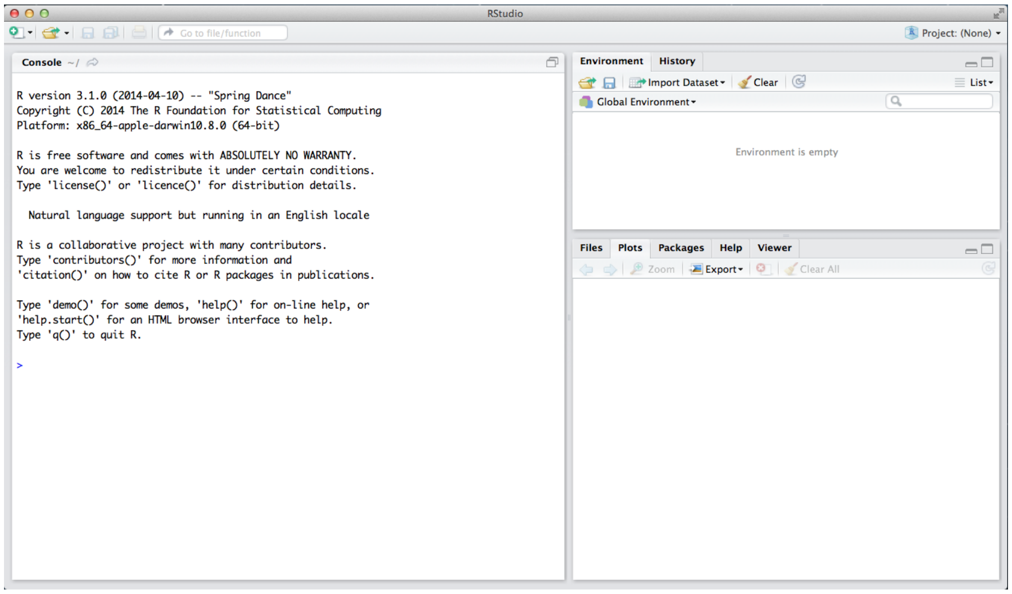
Much of this section has been reused and revised from Rafeal A. Irizarry's *Introduction to Data Science: Data Analysis and Prediction Algorithms with R*, which can be found here: https://rafalab.github.io/dsbook/

The material in his book (and thus, this chapter) is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International CC BY-NC-SA 4.0.

I have added a few initial sections related to R and Rstudio. In addition, I made a few minor changes in places to simplify the text, provide connections to the rest of the book, and to also fill in spots where I thought additional information would be useful.

A.1 Introduction to R and RStudio

We will use R and Rstudio throughout to learn the statistical concepts discussed in the textbook. Many new users are often confused about the difference between R and Rstudio (many students list Rstudio on their CV, when it is probably more important to list R or both R and Rstudio!). R is the name of the programming language itself and RStudio is a graphical user interface (or GUI) used to interact with R. Rstudio lets us run R in an enhanced working environment by providing us with additional functionality (e.g., menu options, multiple windows for plots, code, help files, etc.).



```
R version 3.1.0 (2014-04-10) -- "Spring Dance"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

The panel in the upper right contains your *workspace* as well as a history of the commands that you've previously entered. Any plots that you generate will show up in the panel in the lower right corner.

The panel on the left is where the action happens. It's called the *console*. Every time you launch RStudio, it will have the same text at the top of the console telling you the version of R that you're running. Below that information is the *prompt*. As its name suggests, this prompt is really a request, a request for a command. Initially, interacting with R is all about typing commands and interpreting the output.

You can name and store results in objects using either an equal sign (`=`) or an arrow (`<-`); we will use the latter throughout this book. For example, we can store a vector containing the numbers 1, 2, and 3 (generated by typing `1:3` in R) into an object called `a` using:

```
 a <- 1:3
```

To refer to the stored object, we can just Type its name (`a` in this case):

```
a
```

```
## [1] 1 2 3
```

This will display the object we created. We can also use `a` in future calculations. For example, we can add 2 to all of these numbers by typing:

```
a+2
```

```
## [1] 3 4 5
```

For more complex problems, you will want to write code in a file that we can save, share, and access at a later point in time.

A.1.1 Installing packages and accessing data

Many users contribute code to do all sorts of things in R. They do this by writing 'packages' (bundles of code, sometimes combined with data) and making them available for public download. Accessing this code requires 2 steps:

A. The package has to be downloaded ("installed") onto your computer. This step can be accomplished using the function `install.packages()` or via the menus in RStudio (*Tools -> Install Packages*). Packages only need to be installed once.

B. *Each time* we open R, we have to "tell R" if we want to use any of the add-on packages that we have downloaded. We do this by typing `library(packagename)` (replacing *packagename* with the name of the package we are interested in using).

Note that we can also access specific functions in packages by using `package-name::function-name`. For example, `dplyr::summarize` will use the the `summarize` function in the `dplyr` package (Wickham et al. 2021). This syntax will be useful when we want to highlight which package contains the function we are using. It can also be critical in cases where we have loaded more than 1 package that contains a function with the same name (e.g., there is also a `summarize` function in the `plyr` package).

A.2 Reproducible projects with RStudio and R markdown

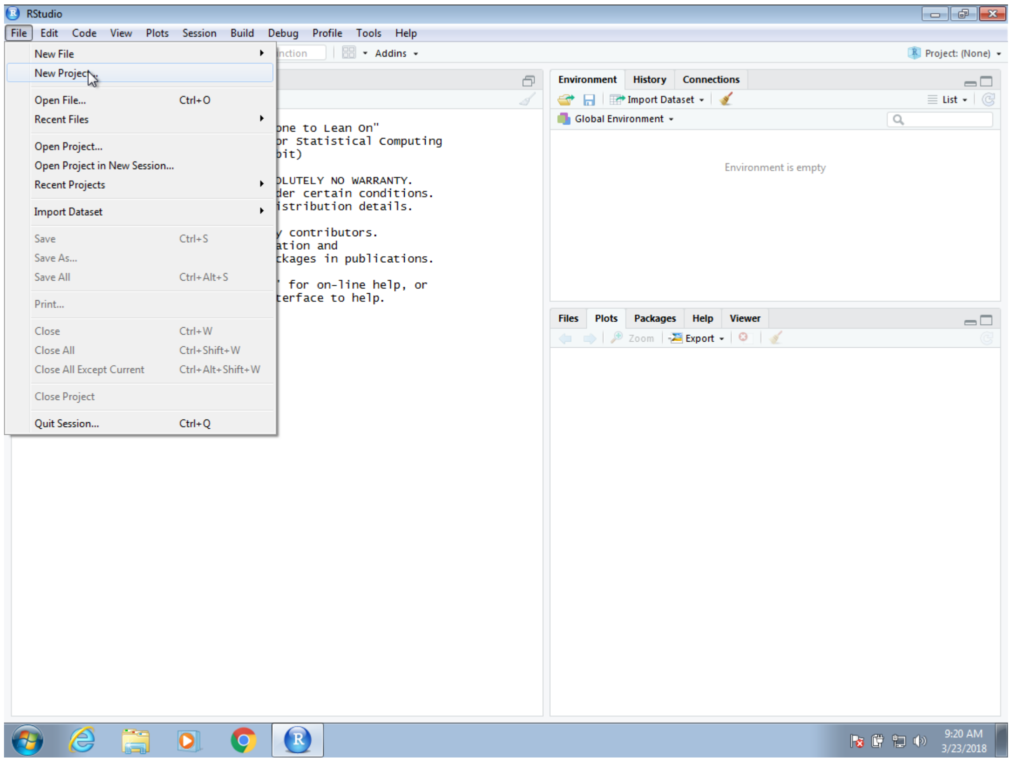
The final product of any data analysis project is often a report or scientific publication containing a description of the findings along with some figures and tables resulting from the analysis. Imagine that after you finish your analysis and the report, you are told that you were given the wrong data set. You thus need to run the same analysis with a new data set. Similarly, you may realize that a mistake was made and need to re-examine the code, fix the error, and re-run the analysis. Or, perhaps your advisor or a researcher from another group studying the same phenomenon would like to see your code and be able to reproduce the results to learn about your approach.

Situations like the ones just described are actually quite common. Here, we describe how you can keep your projects organized with RStudio so that re-running an analysis is straight-forward. We then demonstrate how to generate reproducible reports with R markdown and the **knitR** package in a way that will greatly help with recreating reports with minimal work. This is possible due to the fact that R markdown documents permit code and textual descriptions to be combined into the same document, and the figures and tables produced by the code are automatically added to the document.

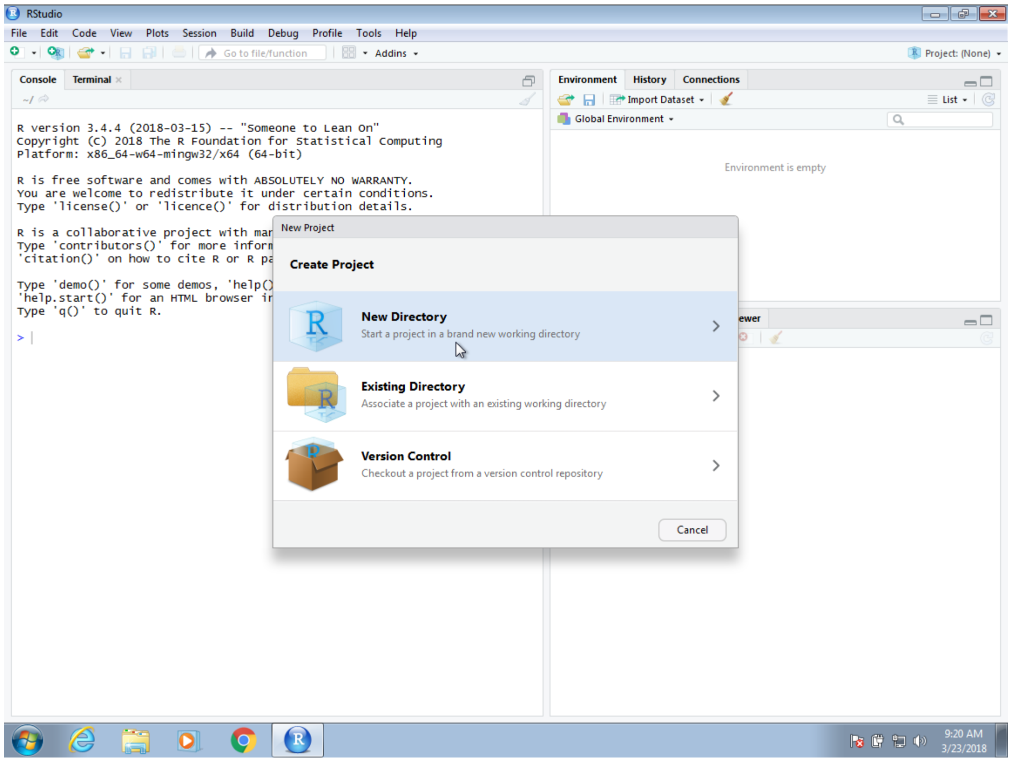
A.3 RStudio projects

RStudio provides a way to keep all the components of a data analysis project organized. In this section, we quickly demonstrate how to start a new a project and some recommendations on how to keep these organized. RStudio projects also permit you to have several RStudio sessions open and keep track of which is which.

To start a project, click on *File* and then *New Project*. Often, we have already created a folder to save the work. If so, we select *Existing Directory*. Here we show an example in which we have not yet created a folder and select the *New Directory* option.

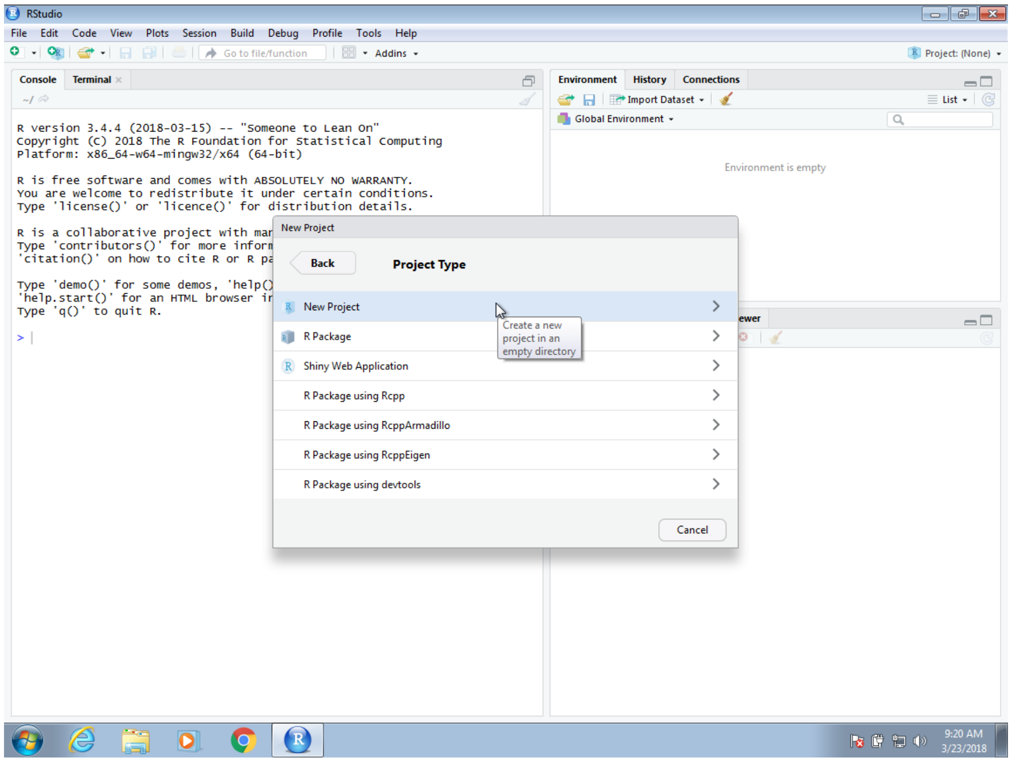


```
one to Lean On"
or Statistical Computing
bit)

OLUTELY NO WARRANTY.
der certain conditions.
istribution details.

y contributors.
ation and
ckages in publications.

' for on-line help, or
terface to help.
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console Terminal
~/
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
New Project
Create Project
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel
9:20 AM
3/23/2018

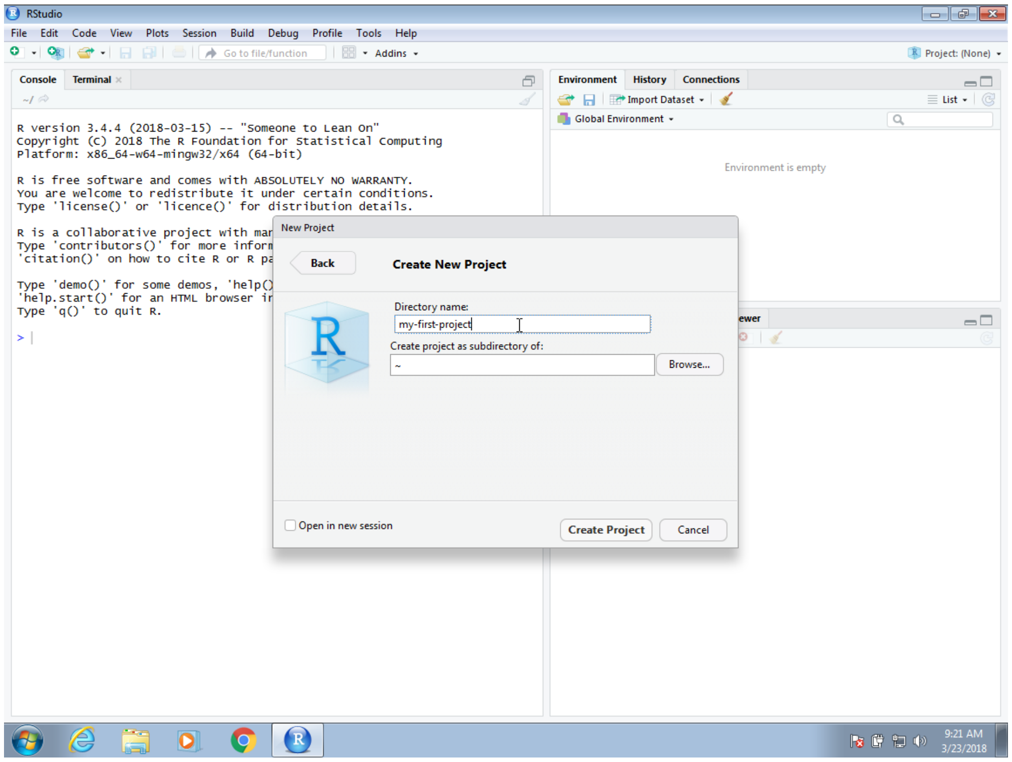
```
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with mar
Type 'contributors()' for more inform
'citation()' on how to cite R or R pa

Type 'demo()' for some demos, 'help()
'help.start()' for an HTML browser in
Type 'q()' to quit R.

> |
```

Then, for a data analysis project, you usually select the *New Project* option:



```
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with mar
Type 'contributors()' for more inform
'citation()' on how to cite R or R pa

Type 'demo()' for some demos, 'help()
'help.start()' for an HTML browser ir
Type 'q()' to quit R.

> |
```

Now you will have to decide on the location of the folder that will be associated with your project, as well as the name of the folder. When choosing a folder name, just like with file names, make sure it is a meaningful name that will help you remember what the project is about. As with files, we recommend using lower case letters, no spaces, and hyphens to separate words. For example, we could call the folder for this project *my-first-project*. This will then generate a *Rproj* file called *my-first-project.Rproj* in the folder associated with the project. We will see how this is useful a few lines below.



```
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with mar
Type 'contributors()' for more inform
'citation()' on how to cite R or R pa

Type 'demo()' for some demos, 'help()
'help.start()' for an HTML browser ir
Type 'q()' to quit R.

> |
```

Directory name: my-first-project

Create project as subdirectory of: ~

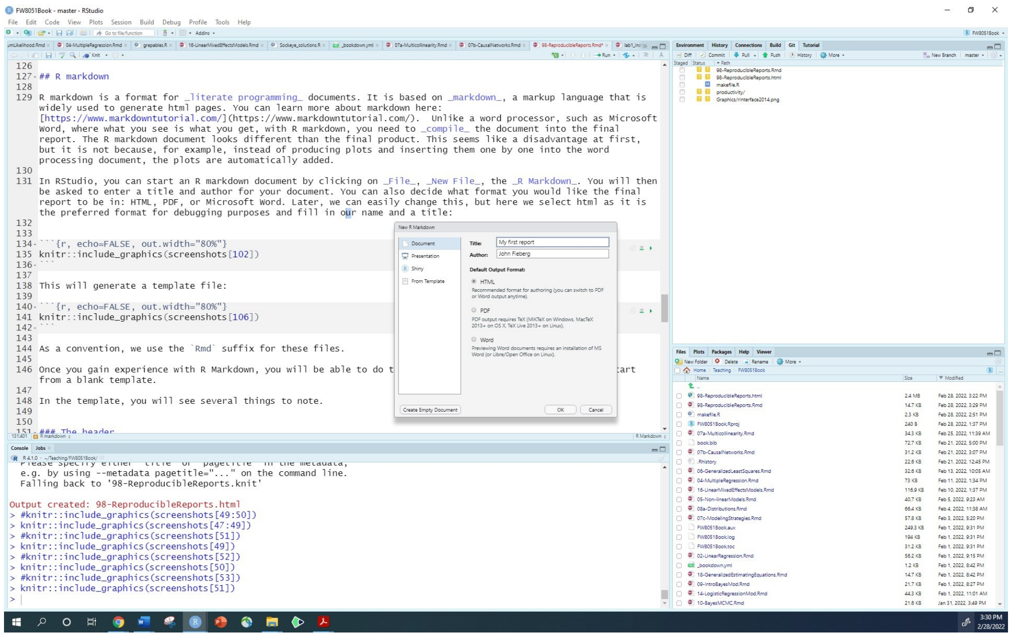
You will be given options on where this folder should be on your filesystem. In this example, we will place it in our home `My Documents` folder, but this is generally not good practice. You want to organize your filesystem following a hierarchical approach. For work associated with this book, you might create separate projects under a common *Statistics4Ecologists* folder.

One of the main advantages of using Projects is that after closing RStudio, if we wish to continue where we left off on the project, we simply double click or open the file saved when we first created the RStudio project. In this case, the file is called *my-first-project.Rproj*. If we open this file, RStudio will start up and open any scripts we previously had open from this project.

A.4 R markdown

R markdown is a format for *literate programming* documents. It is based on *markdown*, a markup language that is widely used to generate html pages. You can learn more about markdown here: https://www.markdowntutorial.com/. Unlike a word processor, such as Microsoft Word, where what you see is what you get, with R markdown, you need to *compile* the document into the final report. The R markdown document looks different than the final product. This seems like a disadvantage at first, but it is not because, for example, instead of producing plots and inserting them one by one into the word processing document, the plots are automatically added.

In RStudio, you can start an R markdown document by clicking on *File*, *New File*, the *R Markdown*. You will then be asked to enter a title and author for your document. You can also decide what format you would like the final report to be in: HTML, PDF, or Microsoft Word. Later, we can easily change this, but here we select html as it is the preferred format for debugging purposes and fill in our name and a title:


FW8051Book - master - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Knit
Run
Console
Jobs
R 4.1.0 · ~/Teaching/FW8051Book/

~~~
## R markdown

R markdown is a format for _literate programming_ documents. It is based on _markdown_, a markup language that is
widely used to generate html pages. You can learn more about markdown here:
[https://www.markdowntutorial.com/](https://www.markdowntutorial.com/).  Unlike a word processor, such as Microsoft
Word, where what you see is what you get, with R markdown, you need to _compile_ the document into the final
report. The R markdown document looks different than the final product. This seems like a disadvantage at first,
but it is not because, for example, instead of producing plots and inserting them one by one into the word
processing document, the plots are automatically added.

In RStudio, you can start an R markdown document by clicking on _File_, _New File_, the _R Markdown_. You will then
be asked to enter a title and author for your document. You can also decide what format you would like the final
report to be in: HTML, PDF, or Microsoft Word. Later, we can easily change this, but here we select html as it is
the preferred format for debugging purposes and fill in our name and a title:


```{r, echo=FALSE, out.width="80%"}
knitr::include_graphics(screenshots[102])
~~~

This will generate a template file:

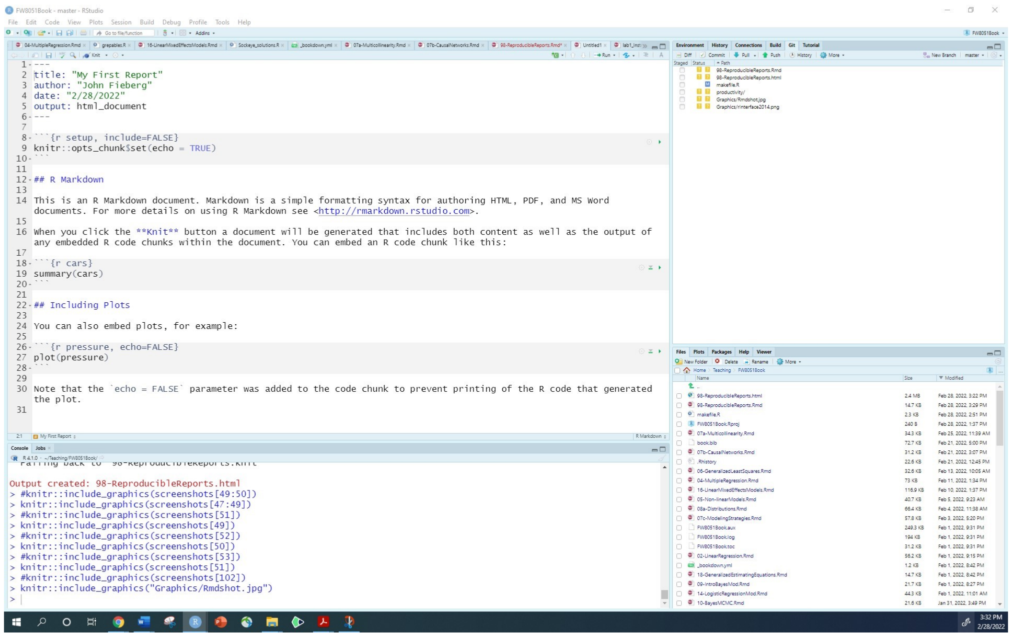
```{r, echo=FALSE, out.width="80%"}
knitr::include_graphics(screenshots[106])
```

As a convention, we use the `Rmd` suffix for these files.

Once you gain experience with R Markdown, you will be able to do t   tart
from a blank template.

In the template, you will see several things to note.

The header
```

New R Markdown

Document
Presentation
Shiny
From Template

Title: My first report
Author: John Fieberg

Default Output Format:

(● HTML) Recommended format for authoring (you can switch to PDF or Word output anytime).

PDF — PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

Word — Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

Create Empty Document | OK | Cancel

```
  e.g. by using --metadata pagetitle="..." on the command line.
  Falling back to '98-ReproducibleReports.knit'

Output created: 98-ReproducibleReports.html
> #knitr::include_graphics(screenshots[49:50])
> knitr::include_graphics(screenshots[47:49])
> #knitr::include_graphics(screenshots[51])
> knitr::include_graphics(screenshots[49])
> #knitr::include_graphics(screenshots[52])
> knitr::include_graphics(screenshots[50])
> #knitr::include_graphics(screenshots[53])
> knitr::include_graphics(screenshots[51])
> 
```

This will generate a template file:


FW8051Book - master - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Knit
Run
My First Report
R Markdown
Console
Jobs
R 4.1.0 · ~/Teaching/FW8051Book/

```
---
title: "My First Report"
author: "John Fieberg"
date: "2/28/2022"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word
documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of
any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r cars}
summary(cars)
```

Including Plots

You can also embed plots, for example:

```{r pressure, echo=FALSE}
plot(pressure)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated
the plot.
```

```
Output created: 98-ReproducibleReports.html
> #knitr::include_graphics(screenshots[49:50])
> knitr::include_graphics(screenshots[47:49])
> #knitr::include_graphics(screenshots[51])
> knitr::include_graphics(screenshots[49])
> #knitr::include_graphics(screenshots[52])
> knitr::include_graphics(screenshots[50])
> #knitr::include_graphics(screenshots[53])
> knitr::include_graphics(screenshots[51])
> #knitr::include_graphics(screenshots[102])
> knitr::include_graphics("Graphics/Rmdshot.jpg")
> 
```

As a convention, we use the `Rmd` suffix for these files. In the template, you will see several things to note.

A.4.1 The header

At the top you see:

```
---
```
```

```
title: "My First Report"
author: "John Fieberg"
date: "2/28/2022"
output: html_document
---
```

The `---` is used to indicate a YAML header. We do not actually need a header, but it is often useful. You can define many other things in the header than what is included in the template (see e.g., the Markdown resources in Section A.5). Also, note that we can change the `output` parameter to `pdf_document` (if we have LaTex installed) or `word_document` if we have MS Word installed. This will change the type of output that is produced when we compile our report.

A.4.2 R code chunks

In various places in the document, we see something like this:

```
‘‘‘{r}
summary(pressure)
‘‘‘
```

These are the code chunks. When you compile the document, the R code inside the chunk, in this case `summary(pressure)`, will be evaluated and the result included in that position in the final document.

To add your own R chunks, you can type the characters above quickly with the key binding command-option-I on the Mac and Ctrl-Alt-I on Windows.

This applies to plots as well; the plot will be placed in that position. We can write something like this:

```
‘‘‘{r}
plot(pressure)
‘‘‘
```

By default, the code will show up as well. To avoid having the code show up, you can use an argument. To avoid this, you can use the argument `echo=FALSE`. For example:

```
‘‘‘{r, echo=FALSE}
summary(pressure)
‘‘‘
```

We recommend getting into the habit of adding a label to the R code chunks. This will be very useful when debugging, among other situations. You do this by adding a descriptive word like this:

```
‘‘‘{r pressure-summary}
summary(pressure)
‘‘‘
```

Lastly, note that any errors in your code will prevent you from being able to generate a report to share. One way around this is to add the following code to the top of your .Rmd file:

```
‘‘‘{r}
knitr::opts_chunk$set(
  error = TRUE # do not interrupt in case of errors
)
‘‘‘
```

A.4.3 Global options

One of the R chunks contains a complex looking call:

~~~
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
~~~

We will not cover this here, but as you become more experienced with R Markdown, you will learn the advantages of setting global options for the compilation process.

A.4.4 knitR

We use the **knitR** package to compile R markdown documents. The specific function used to compile is the `knit` function, which takes a filename as input. RStudio provides a button that makes it easier to compile the document. The first time you click on the *Knit* button, a dialog box may appear asking you to install packages you need.

Once you have installed the packages, clicking the *Knit* will compile your R markdown file and the resulting document will pop up and an html document will be saved in your working directory. To view it, open a terminal and list the files. You can open the file in a browser and use this to present your analysis.

A.5 More on R markdown

There is a lot more you can do with R markdown. We highly recommend you continue learning as you gain more experience writing reports in R. There are many free resources on the internet including:

- RStudio's tutorial: https://rmarkdown.rstudio.com
- The cheat sheet: https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf
- The knitR book: https://yihui.name/knitr/
- The R Markdown cookbookhttps://bookdown.org/yihui/rmarkdown-cookbook/

References

Abdi, Hervé, and Lynne J Williams. 2010. "Tukey's Honestly Significant Difference (HSD) Test." *Encyclopedia of Research Design* 3 (1): 1–5.

Addicott, Ethan T, Eli P Fenichel, Mark A Bradford, Malin L Pinsky, and Stephen A Wood. 2022. "Toward an Improved Understanding of Causation in the Ecological Sciences." *Frontiers in Ecology and the Environment* 20 (8): 474–80.

Agresti, Alan. 2018. *An Introduction to Categorical Data Analysis*. John Wiley & Sons.

Akaike, Hirotugu. 1974. "A New Look at the Statistical Model Identification." In *Selected Papers of Hirotugu Akaike*, 215–22. Springer.

Allison, Truett, and Domenic V Cicchetti. 1976. "Sleep in Mammals: Ecological and Constitutional Correlates." *Science* 194 (4266): 732–34.

Anderson, D, and K Burnham. 2004. "Model Selection and Multi-Model Inference." *Second. NY: Springer-Verlag* 63.

ArchMiller, Althea A, Robert M Dorazio, Katherine St Clair, and John Fieberg. 2018. "Time Series Sightability Modeling of Animal Populations." *PloS One* 13 (1).

Arel-Bundock, Vincent. 2022. "modelsummary: Data and Model Summaries in R." *Journal of Statistical Software* 103 (1): 1–23. https://doi.org/10.18637/jss.v103.i01.

Arif, Suchinta, and M Aaron MacNeil. 2022. "Predictive Models Aren't for Causal Inference." *Ecology Letters* 25 (8): 1741–45.

Arnold, Todd W. 2010. "Uninformative Parameters and Model Selection Using Akaike's Information Criterion." *The Journal of Wildlife Management* 74 (6): 1175–78.

Arnqvist, Göran. 2020. "Mixed Models Offer No Freedom from Degrees of Freedom." *Trends in Ecology & Evolution* 35 (4): 329–35.

Babyak, Michael A. 2004. "What You See May Not Be What You Get: A Brief, Nontechnical Introduction to Overfitting in Regression-Type Models." *Psychosomatic Medicine* 66 (3): 411–21.

Barr, Dale J, Roger Levy, Christoph Scheepers, and Harry J Tily. 2013. "Random Effects Structure for Confirmatory Hypothesis Testing: Keep It Maximal." *Journal of Memory and Language* 68 (3): 255–78.

Barton, Kamil. 2020. *MuMIn: Multi-Model Inference*. https://CRAN.R-project.org/package=MuMIn.

Bates, Douglas, Martin Mächler, Ben Bolker, and Steve Walker. 2015. "Fitting Linear Mixed-Effects Models Using lme4." *Journal of Statistical Software* 67 (1): 1–48. https://doi.org/10.18637/jss.v067.i01.

Benjamini, Yoav, and Yosef Hochberg. 1995. "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing." *Journal of the Royal Statistical Society: Series B (Methodological)* 57 (1): 289–300.

Beverton, Raymond JH, and Sidney J Holt. 2012. *On the Dynamics of Exploited Fish Populations*. Vol. 11. Springer Science & Business Media.

Bigler, Christof. 2016. "Trade-Offs Between Growth Rate, Tree Size and Lifespan of Mountain Pine (Pinus Montana) in the Swiss National Park." *PloS One* 11 (3): e0150402.

Bliss, Chester Ittner. 1935. "The Calculation of the Dosage-Mortality Curve." *Annals of Applied Biology* 22 (1): 134–67.

Bolker, Ben. 2023. *Emdbook: Support Functions and Data for "Ecological Models and Data"*. https://CRAN.R-project.org/package=emdbook.

Bolker, Benjamin M. 2008. *Ecological Models and Data in R*. Princeton University Press.

Bolker, Ben, and R Development Core Team. 2020. *Bbmle: Tools for General Maximum Likelihood Estimation*. https://CRAN.R-project.org/package=bbmle.

Bolker, Ben, and David Robinson. 2021. *Broom.mixed: Tidying Methods for Mixed Models*. https://CRAN.R-project.org/package=broom.mixed.

Bramwell, Ros, Helen West, and Peter Salmon. 2006. "Health Professionals' and Service Users' Interpretation of Screening Test Results: Experimental Study." *Bmj* 333 (7562): 284.

Breheny, Patrick, and Woodrow Burchett. 2013. "Visualization of Regression Models Using Visreg." *R Package*, 1–15.

Brenning, Alexander. 2012. "Spatial Cross-Validation and Bootstrap for the Assessment of Prediction Rules in Remote Sensing: The r Package 'Sperrorest'." In *IEEE International Symposium on Geoscience and Remote Sensing IGARSS*. https://doi.org/10.1109/igarss.2012.6352393.

Brewer, Mark J, Adam Butler, and Susan L Cooksley. 2016. "The Relative Performance of AIC, AICC and BIC in the Presence of Unobserved Heterogeneity." *Methods in Ecology and Evolution* 7 (6): 679–92.

Brooks, Mollie E., Kasper Kristensen, Koen J. van Benthem, Arni Magnusson, Casper W. Berg, Anders Nielsen, Hans J. Skaug, Martin Maechler, and Benjamin M. Bolker. 2017. "glmmTMB Balances Speed and Flexibility Among Packages for Zero-Inflated Generalized Linear Mixed Modeling." *The R Journal* 9 (2): 378–400. https://journal.r-project.org/archive/2017/RJ-2017-066/index.html.

Buckland, Steven T, Kenneth P Burnham, and Nicole H Augustin. 1997. "Model Selection: An Integral Part of Inference." *Biometrics*, 603–18.

Cade, Brian S. 2015. "Model Averaging and Muddled Multimodel Inferences." *Ecology* 96 (9): 2370–82.

Carpenter, Bob, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. "Stan: A Probabilistic Programming Language." *Journal of Statistical Software* 76 (1).

Carrière, Isabelle, and Jean Bouyer. 2002. "Choosing Marginal or Random-Effects Models for Longitudinal Binary Responses: Application to Self-Reported Disability Among Older Persons." *BMC Medical Research Methodology* 2 (1): 1–10.

Casella, George, Malay Ghosh, Jeff Gill, and Minjung Kyung. 2010. "Penalized Regression, Standard Errors, and Bayesian Lassos." *Bayesian Analysis* 5 (2): 369–411.

Çetinkaya-Rundel, Mine, David Diez, Andrew Bray, Albert Y. Kim, Ben Baumer, Chester Ismay, Nick Paterno, and Christopher Barr. 2021. *Openintro: Data Sets and Supplemental Functions from 'OpenIntro' Textbooks and Labs*. https://CRAN.R-project.org/package=openintro.

Chung, Yeojin, Andrew Gelman, Sophia Rabe-Hesketh, Jingchen Liu, and Vincent Dorie. 2015. "Weakly Informative Prior for Point Estimation of Covariance Matrices in Hierarchical Models." *Journal of Educational and Behavioral Statistics* 40 (2): 136–57.

Cohen, Jacob. 1992. "Things i Have Learned (so Far)." In *Annual Convention of the American Psychological Association, 98th, Aug, 1990, Boston, MA, US; Presented at the Aforementioned Conference*. American Psychological Association.

Cole, Stephen R, Robert W Platt, Enrique F Schisterman, Haitao Chu, Daniel Westreich, David Richardson, and Charles Poole. 2010. "Illustrating Bias Due to Conditioning on a Collider." *International Journal of Epidemiology* 39 (2): 417–20.

Conn, Paul B, Devin S Johnson, Perry J Williams, Sharon R Melin, and Mevin B Hooten. 2018. "A Guide to Bayesian Model Checking for Ecologists." *Ecological Monographs* 88 (4): 526–42.

Conor, E. F., and E. D. McCoy. 2013. "Species–Area Relationships." In *Encyclopedia of Biodiversity (Second Edition)*, edited by Simon A Levin, Second Edition, 640–50. Waltham: Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-384719-5.00132-5.

Cooke, Steven J, Michael R Donaldson, Scott G Hinch, Glenn T Crossin, David A Patterson, Kyle C Hanson, Karl K English, J Mark Shrimpton, and Anthony P Farrell. 2009. "Is Fishing Selective for Physiological and Energetic Characteristics in Migratory Adult Sockeye Salmon?" *Evolutionary Applications* 2 (3): 299–311.

Copas, JB, and Tianyong Long. 1991. "Estimating the Residual Variance in Orthogonal Regression with Variable Selection." *Journal of the Royal Statistical Society: Series D (The Statistician)* 40 (1): 51–59.

Crainiceanu, Ciprian M, and David Ruppert. 2004. "Likelihood Ratio Tests in Linear Mixed Models with One Variance Component." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 66 (1): 165–85.

Crawley, Michael J. 2012. *The r Book*. John Wiley & Sons.

Cummings, Jonathan W, Merran J Hague, David A Patterson, and Randall M Peterman. 2011. "The Impact of Different Performance Measures on Model Selection for Fraser River Sockeye Salmon." *North American Journal of Fisheries Management* 31 (2): 323–34.

Curtis, S. McKay. 2018. *Mcmcplots: Create Plots from MCMC Output*. https://CRAN.R-project.org/package=mcmcplots.

Cutler, D Richard, Thomas C Edwards Jr, Karen H Beard, Adele Cutler, Kyle T Hess, Jacob Gibson, and Joshua J Lawler. 2007. "Random Forests for Classification in Ecology." *Ecology* 88 (11): 2783–92.

Dablander, Fabian. 2020. "An Introduction to Causal Inference. https://doi.org/10.31234/osf.io/b3fkw."

Dahlgren, Johan P. 2010. "Alternative Regression Methods Are Not Considered in Murtaugh (2009) or by Ecologists in General." *Ecology Letters* 13 (5): E7–9.

Davison, Anthony Christopher, and David Victor Hinkley. 1997. *Bootstrap Methods and Their Application*. 1. Cambridge university press.

De Valpine, Perry. 2014. "The Common Sense of p Values." *Ecology* 95 (3): 617–21.

DelGiudice, Glenn D, Michael R Riggs, Pierre Joly, and Wei Pan. 2002. "Winter Severity, Survival, and Cause-Specific Mortality of Female White-Tailed Deer in North-Central Minnesota." *The Journal of Wildlife Management*, 698–717.

Dennis, Brian. 1996. "Discussion: Should Ecologists Become Bayesians?" *Ecological Applications* 6 (4): 1095–1103.

Dickie, M, R Serrouya, T Avgar, P McLoughlin, RS McNay, C DeMars, S Boutin, and AT Ford. 2022. "Resource Exploitation Efficiency Collapses the Home Range of an Apex Predator." *Ecology*, e3642.

Diggle, Peter, Kung-Yee Liang, and Scott L Zeger. 1994. "Longitudinal Data Analysis." *New York: Oxford University Press* 5: 13.

Ditmer, Mark A, DL Garshelis, KV Noyce, TG Laske, Paul A Iaizzo, TE Burk, James D Forester, and John Fieberg. 2015. "Behavioral and Physiological Responses of American Black Bears to Landscape Features Within an Agricultural Region." *Ecosphere* 6 (3): 1–21.

Dorazio, Robert M. 2016. "Bayesian Data Analysis in Population Ecology: Motivations, Methods, and Benefits." *Population Ecology* 58 (1): 31–44.

Dormann, Carsten F, Jane Elith, Sven Bacher, Carsten Buchmann, Gudrun Carl, Gabriel Carré, Jaime R García Marquéz, et al. 2013. "Collinearity: A Review of Methods to Deal with It and a Simulation Study Evaluating Their Performance." *Ecography* 36 (1): 27–46.

Efron, Bradley, Elizabeth Halloran, and Susan Holmes. 1996. "Bootstrap Confidence Levels for Phylogenetic Trees." *Proceedings of the National Academy of Sciences* 93 (14): 7085–90.

Elith, Jane, John R Leathwick, and Trevor Hastie. 2008. "A Working Guide to Boosted Regression Trees." *Journal of Animal Ecology* 77 (4): 802–13.

Faraway, Julian J. 2016. *Extending the Linear Model with r: Generalized Linear, Mixed Effects and Nonparametric Regression Models*. CRC press.

Felsenstein, Joseph. 1985. "Confidence Limits on Phylogenies: An Approach Using the Bootstrap." *Evolution* 39 (4): 783–91.

Fieberg, John. 2012. "Estimating Population Abundance Using Sightability Models: R SightabilityModel Package." *Journal of Statistical Software* 51 (9): 1–20.

———. 2021. *Data4Ecologists: Data Associated with Statistics for Ecologists*.

Fieberg, John, Michael Alexander, Scarlett Tse, and Katie St. Clair. 2013. "Abundance Estimation with Sightability Data: A Bayesian Data Augmentation Approach." *Methods in Ecology and Evolution* 4 (9): 854–64.

Fieberg, John, and Douglas H Johnson. 2015. "MMI: Multimodel Inference or Models with Management Implications?" *The Journal of Wildlife Management* 79 (5): 708–18.

Fieberg, John, Randall H Rieger, Michael C Zicus, and Jonathan S Schildcrout. 2009. "Regression Modelling of Correlated Data in Ecology: Subject-Specific and Population Averaged Response Patterns." *Journal of Applied Ecology* 46 (5): 1018–25.

Fieberg, John, Johannes Signer, Brian Smith, and Tal Avgar. 2021. "A 'How to' Guide for Interpreting Parameters in Habitat-Selection Analyses." *Journal of Animal Ecology* 90 (5): 1027–43.

Fieberg, John, Kelsey Vitense, and Douglas H Johnson. 2020. "Resampling-Based Methods for Biologists." *PeerJ* 8: e9089.

Fijorek, Kamil, and Andrzej Sokolowski. 2012. "Separation-Resistant and Bias-Reduced Logistic Regression: Statistica Macro." *Journal of Statistical Software* 47: 1–12.

Firth, David. 1993. "Bias Reduction of Maximum Likelihood Estimates." *Biometrika* 80 (1): 27–38.

Forstmeier, Wolfgang, Eric-Jan Wagenmakers, and Timothy H Parker. 2017. "Detecting and Avoiding Likely False-Positive Findings–a Practical Guide." *Biological Reviews* 92 (4): 1941–68.

Fox, John. 2003. "Effect Displays in R for Generalised Linear Models." *Journal of Statistical Software* 8 (15): 1–27. https://www.jstatsoft.org/article/view/v008i15.

Fox, John, and Sanford Weisberg. 2018. "Visualizing Fit and Lack of Fit in Complex Regression Models with Predictor Effect Plots and Partial Residuals." *Journal of Statistical Software* 87 (9): 1–27. https://doi.org/10.18637/jss.v087.i09.

———. 2019. *An R Companion to Applied Regression*. Third. Thousand Oaks CA: Sage. https://socialsciences.mcmaster.ca/jfox/Books/Companion/.

Friederichs, Samuel J, Kyle D Zimmer, Brian R Herwig, Mark A Hanson, and John Fieberg. 2011. "Total Phosphorus and Piscivore Mass as Drivers of Food Web Characteristics in Shallow Lakes." *Oikos* 120 (5): 756–65.

Friedman, Jerome, Trevor Hastie, and Rob Tibshirani. 2010. "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software* 33 (1): 1.

García, Luis V. 2004. "Escaping the Bonferroni Iron Claw in Ecological Studies." *Oikos* 105 (3): 657–63.

Gelman, Andrew, and Jennifer Hill. 2006. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge university press.

Gelman, Andrew, Aleks Jakulin, Maria Grazia Pittau, Yu-Sung Su, et al. 2008. "A Weakly Informative Default Prior Distribution for Logistic and Other Regression Models." *The Annals of Applied Statistics* 2 (4): 1360–83.

Gelman, Andrew, Donald B Rubin, et al. 1992. "Inference from Iterative Simulation Using Multiple Sequences." *Statistical Science* 7 (4): 457–72.

Giesbrecht, Francis G, and Joseph C Burns. 1985. "Two-Stage Analysis Based on a Mixed Model: Large-Sample Asymptotic Theory and Small-Sample Simulation Results." *Biometrics*, 477–86.

Giudice, John H, John Fieberg, and Mark S Lenarz. 2012. "Spending Degrees of Freedom in a Poor Economy: A Case Study of Building a Sightability Model for Moose in Northeastern Minnesota." *The Journal of Wildlife Management* 76 (1): 75–87.

Glymour, Madelyn, Judea Pearl, and Nicholas P Jewell. 2016. *Causal Inference in Statistics: A Primer*. John Wiley & Sons.

Goode, Katherine, and Kathleen Rey. 2019. *ggResidpanel: Panels and Interactive Versions of Diagnostic Plots Using 'Ggplot2'*. https://CRAN.R-project.org/package=ggResidpanel.

Grace, James B. 2008. "Structural Equation Modeling for Observational Studies." *The Journal of Wildlife Management* 72 (1): 14–22.

Graham, Michael H. 2003. "Confronting Multicollinearity in Ecological Multiple Regression." *Ecology* 84 (11): 2809–15.

Gray, Brian R. 2005. "Selecting a Distributional Assumption for Modelling Relative Densities of Benthic Macroinvertebrates." *Ecological Modelling* 185 (1): 1–12.

Greven, Sonja, and Thomas Kneib. 2010. "On the Behaviour of Marginal and Conditional AIC in Linear Mixed Models." *Biometrika* 97 (4): 773–89.

Halekoh, Ulrich, and Søren Højsgaard. 2014. "A Kenward-Roger Approximation and Parametric Bootstrap Methods for Tests in Linear Mixed Models – the R Package pbkrtest." *Journal of Statistical Software* 59 (9): 1–30. http://www.jstatsoft.org/v59/i09/.

Halekoh, Ulrich, Søren Højsgaard, and Jun Yan. 2006. "The r Package Geepack for Generalized Estimating Equations." *Journal of Statistical Software* 15/2: 1–11.

Harrell Jr, Frank E. 2015. *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*. Springer.

———. 2021. *Rms: Regression Modeling Strategies*. https://CRAN.R-project.org/package=rms.

Harrison, Xavier A. 2014. "Using Observation-Level Random Effects to Model Overdispersion in Count Data in Ecology and Evolution." *PeerJ* 2: e616.

Hartig, Florian. 2021. *DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed) Regression Models*. https://CRAN.R-project.org/package=DHARMa.

Heagerty, Patrick J, and Scott L Zeger. 1998. "Lorelogram: A Regression Approach to Exploring Dependence in Longitudinal Categorical Responses." *Journal of the American Statistical Association* 93 (441): 150–62.

Hedeker, Donald, Stephen HC du Toit, Hakan Demirtas, and Robert D Gibbons. 2018. "A Note on Marginalization of Regression Parameters from Mixed Models of Binary Outcomes." *Biometrics* 74 (1): 354–61.

Hefley, Trevor J, Kristin M Broms, Brian M Brost, Frances E Buderman, Shannon L Kay, Henry R Scharf, John R Tipton, Perry J Williams, and Mevin B Hooten. 2017. "The Basis Function Approach for Modeling Autocorrelation in Ecological Data." *Ecology* 98 (3): 632–46.

Hegyi, Gergely, and László Zsolt Garamszegi. 2011. "Using Information Theory as a Substitute for Stepwise Regression in Ecology and Behavior." *Behavioral Ecology and Sociobiology* 65 (1): 69–76.

Heinze, Georg, Christine Wallisch, and Daniela Dunkler. 2018. "Variable Selection–a Review and Recommendations for the Practicing Statistician." *Biometrical Journal* 60 (3): 431–49.

Henry, Lionel, and Hadley Wickham. 2020. *Purrr: Functional Programming Tools*. https://CRAN.R-project.org/package=purrr.

Herbranson, Walter T, and Julia Schroeder. 2010. "Are Birds Smarter Than Mathematicians? Pigeons (Columba Livia) Perform Optimally on a Version of the Monty Hall Dilemma." *Journal of Comparative Psychology* 124 (1): 1.

Hesterberg, Tim C. 2015. "What Teachers Should Know about the Bootstrap: Resampling in the Undergraduate Statistics Curriculum." *The American Statistician* 69 (4): 371–86.

Hilbe, Joseph M, and James W Hardin. 2008. "Generalized Estimating Equations for Longitudinal Panel Analysis." *Handbook of Longitudinal Research: Design, Measurement, and Analysis* 1: 467–74.

Hoerl, Arthur E, and Robert W Kennard. 1970. "Ridge Regression: Biased Estimation for Nonorthogonal Problems." *Technometrics* 12 (1): 55–67.

Hoffman, Matthew D, and Andrew Gelman. 2014. "The No-u-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo." *Journal of Machine Learning Research* 15 (1): 1593–623.

Hooten, Mevin B, and N Thompson Hobbs. 2015. "A Guide to Bayesian Model Selection for Ecologists." *Ecological Monographs* 85 (1): 3–28.

Hosmer Jr, David W, Stanley Lemeshow, and Rodney X Sturdivant. 2013. *Applied Logistic Regression*. Vol. 398. John Wiley & Sons.

Hrong-Tai Fai, Alex, and Paul L Cornelius. 1996. "Approximate f-Tests of Multiple Degree of Freedom Hypotheses in Generalized Least Squares Analyses of Unbalanced Split-Plot Experiments." *Journal of Statistical Computation and Simulation* 54 (4): 363–78.

Iannarilli, Fabiola, Todd W Arnold, John Erb, and John Fieberg. 2019. "Using Lorelograms to Measure and Model Correlation in Binary Data: Applications to Ecological Studies." *Methods in Ecology and Evolution* 10 (12): 2153–62.

Iannarilli, Fabiola, John Erb, Todd W Arnold, and John Fieberg. 2021. "Evaluating Species-Specific Responses to Camera-Trap Survey Designs." *Wildlife Biology* 2021 (1).

Ioannidis, John PA. 2005. "Why Most Published Research Findings Are False." *PLoS Medicine* 2 (8): e124.

Isaac, Nick JB, Marta A Jarzyna, Petr Keil, Lea I Dambly, Philipp H Boersch-Supan, Ella Browning, Stephen N Freeman, et al. 2020. "Data Integration for Large-Scale Models of Species Distributions." *Trends in Ecology & Evolution* 35 (1): 56–67.

Jackman, Simon. 2020. *pscl: Classes and Methods for R Developed in the Political Science Computational Laboratory*. Sydney, New South Wales, Australia: United States Studies Centre, University of Sydney. https://github.com/atahk/pscl/.

Janssen, Gerard, and Saskia Mulder. 2005. "Zonation of Macrofauna Across Sandy Beaches and Surf Zones Along the Dutch Coast." *Oceanologia* 47 (2).

Janssen, GM, and S Mulder. 2004. "De Ecologie van de Zandige Kust van Nederland: Inventarisatie van Het Marcobenthos van Zand En Brandingszone." *Rapportnr.: 2004.033*.

Johnson, Douglas H. 1999. "The Insignificance of Statistical Significance Testing." *The Journal of Wildlife Management*, 763–72.

Johnson, Michael P, and Peter H Raven. 1973. "Species Number and Endemism: The Galápagos Archipelago Revisited." *Science* 179 (4076): 893–95.

Jolliffe, Ian T. 1982. "A Note on the Use of Principal Components in Regression." *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 31 (3): 300–303.

Kaplan, D. 2009. *Statistical Modeling: A Fresh Approach*. CreateSpace Independent Publishing Platform. https://books.google.com/books?id=xphtQgAACAAJ.

Kaplan, Daniel, and Randall Pruim. 2021. *Ggformula: Formula Interface to the Grammar of Graphics*. https://CRAN.R-project.org/package=ggformula.

Kass, J. M., R. Muscarella, P. J. Galante, C. L. Bohl, G. E. Pinilla-Buitrago, R. A. Boria, M. Soley-Guardia, and R. P. Anderson. 2021. "ENMeval 2.0: Redesigned for Customizable and Reproducible Modeling of Species' Niches and Distributions." *Methods in Ecology and Evolution*. https://doi.org/10.1111/2041-210X.13628.

Keele, Luke, and Luke Keele. 2008. *Semiparametric Regression for the Social Sciences*. Sirsi) i9780470319918. Wiley Online Library.

Kenward, Michael G, and James H Roger. 1997. "Small Sample Inference for Fixed Effects from Restricted Maximum Likelihood." *Biometrics*, 983–97.

Kéry, Marc. 2010. *Introduction to WinBUGS for Ecologists: Bayesian Approach to Regression, ANOVA, Mixed Models and Related Analyses*. Academic Press.

Knief, Ulrich, and Wolfgang Forstmeier. 2021. "Violating the Normality Assumption May Be the Lesser of Two Evils." *Behavior Research Methods*, 1–15.

Knowles, Jared E., and Carl Frederick. 2023. *merTools: Tools for Analyzing Mixed Effect Regression Models*. https://CRAN.R-project.org/package=merTools.

Kuhn, Max. 2021. *Caret: Classification and Regression Training*. https://CRAN.R-project.org/package=caret.

Kuhn, Max, and Hadley Wickham. 2020. *Tidymodels: A Collection of Packages for Modeling and Machine Learning Using Tidyverse Principles.* https://www.tidymodels.org.

Kuiper, Shonda, and Jeff Sklar. 2012. *Practicing Statistics: Guided Investigations for the Second Course*. Pearson Higher Ed.

Kutner, Michael H., Christopher J. Nachtsheim, and John Neter. 2004. *Applied Linear Regression Models*. McGraw-Hill.

Kutner, Michael H, Christopher J Nachtsheim, John Neter, William Li, et al. 2005. *Applied Linear Statistical Models*. Vol. 5. McGraw-Hill Irwin Boston.

Kuznetsova, Alexandra, Per B. Brockhoff, and Rune H. B. Christensen. 2017. "lmerTest Package: Tests in Linear Mixed Effects Models." *Journal of Statistical Software* 82 (13): 1–26. https://doi.org/10.18637/jss.v082.i13.

Laubach, Zachary M, Eleanor J Murray, Kim L Hoke, Rebecca J Safran, and Wei Perng. 2021. "A Biologist's Guide to Model Selection and Causal Inference." *Proceedings of the Royal Society B* 288 (1943): 20202815.

Lele, Subhash R. 2020. "Consequences of Lack of Parameterization Invariance of Non-Informative Bayesian Analysis for Wildlife Management: Survival of San Joaquin Kit Fox and Declines in Amphibian Populations." *Frontiers in Ecology and Evolution* 7: 501.

Lele, Subhash R, and Brian Dennis. 2009. "Bayesian Methods for Hierarchical Models: Are Ecologists Making a Faustian Bargain?" *Ecological Applications* 19 (3): 581–84.

Lele, Subhash R, Brian Dennis, and Frithjof Lutscher. 2007. "Data Cloning: Easy Maximum Likelihood Estimation for Complex Ecological Models Using Bayesian Markov Chain Monte Carlo Methods." *Ecology Letters* 10 (7): 551–63.

Lele, Subhash R., Jonah L. Keim, and Peter Solymos. 2019. *ResourceSelection: Resource Selection (Probability) Functions for Use-Availability Data*. https://CRAN.R-project.org/package=ResourceSelection.

Lenth, Russell V. 2021. *Emmeans: Estimated Marginal Means, Aka Least-Squares Means*. https://CRAN.R-project.org/package=emmeans.

Lever, Jake, Martin Krzywinski, and Naomi Altman. 2016. “Points of Significance: Regularization.” *Nature Methods* 13 (10): 803–5.

Liang, Kung-Yee, and Scott L Zeger. 1986. “Longitudinal Data Analysis Using Generalized Linear Models.” *Biometrika* 73 (1): 13–22.

Lindeløv, Jonas Kristoffer. 2020. “Mcp: An r Package for Regression with Multiple Change Points.” *OSF Preprints*. https://doi.org/10.31219/osf.io/fzqxv.

Little, Roderick J. 2006. “Calibrated Bayes: A Bayes/Frequentist Roadmap.” *The American Statistician* 60 (3): 213–23.

Little, Roderick JA, and Donald B Rubin. 2019. *Statistical Analysis with Missing Data*. Vol. 793. John Wiley & Sons.

Lock, Robin. 2017. *Lock5Data: Datasets for "Statistics: UnLocking the Power of Data"*. https://CRAN.R-project.org/package=Lock5Data.

Lock, Robin H, Patti Frazer Lock, Kari Lock Morgan, Eric F Lock, and Dennis F Lock. 2020. *Statistics: Unlocking the Power of Data*. John Wiley & Sons.

Lucas, Tim CD. 2020. “A Translucent Box: Interpretable Machine Learning in Ecology.” *Ecological Monographs* 90 (4): e01422.

Lüdecke, Daniel. 2018. “Ggeffects: Tidy Data Frames of Marginal Effects from Regression Models.” *Journal of Open Source Software* 3 (26): 772. https://doi.org/10.21105/joss.00772.

———. 2021. *sjPlot: Data Visualization for Statistics in Social Science*. https://CRAN.R-project.org/package=sjPlot.

Lüdecke, Daniel, Mattan S. Ben-Shachar, Indrajeet Patil, Philip Waggoner, and Dominique Makowski. 2021. “performance: An R Package for Assessment, Comparison and Testing of Statistical Models.” *Journal of Open Source Software* 6 (60): 3139. https://doi.org/10.21105/joss.03139.

Lukacs, Paul M, Kenneth P Burnham, and David R Anderson. 2010. “Model Selection Bias and Freedman’s Paradox.” *Annals of the Institute of Statistical Mathematics* 62 (1): 117.

Luke, Steven G. 2017. “Evaluating Significance in Linear Mixed-Effects Models in r.” *Behavior Research Methods* 49 (4): 1494–1502.

Luque-Fernandez, Miguel Angel, Michael Schomaker, Daniel Redondo-Sanchez, Maria Jose Sanchez Perez, Anand Vaidya, and Mireille E Schnitzer. 2019. “Educational Note: Paradoxical Collider Effect in the Analysis of Non-Communicable Disease Epidemiological Data: A Reproducible Illustration and Web Application.” *International Journal of Epidemiology* 48 (2): 640–53.

Lyon, Aidan. 2010. “Philosophy of Probability.” *Philosophies of the Sciences: A Guide*, 92–125.

MacKenzie, Darryl I, James D Nichols, J Andrew Royle, Kenneth H Pollock, Larissa Bailey, and James E Hines. 2017. *Occupancy Estimation and Modeling: Inferring Patterns and Dynamics of Species Occurrence*. Elsevier.

Manly, BFJ. 1991. “Randomization and Monte Carlo Methods in Biology London.” *UK: Chapman and Hall.*

Marchetti, Giovanni M., Mathias Drton, and Kayvan Sadeghi. 2020. *Ggm: Graphical Markov Models with Mixed Graphs*. https://CRAN.R-project.org/package=ggm.

McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models, Second Edition*. Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series. Chapman & Hall. http://books.google.com/books?id=h9kFH2/_FfBkC.

McCulloch, Charles E, and John M Neuhaus. 2005. “Generalized Linear Mixed Models.” *Encyclopedia of Biostatistics* 4.

McElreath, Richard. 2020. *Statistical Rethinking: A Bayesian Course with Examples in r and Stan*. CRC press.

McFadden, D. 1977. “Quantitative Methods for Analyzing Travel Behavior of Individuals: Some Recent 857 Developments: Institute of Transportation Studies.” *University of California* 858.

Mech, L David, John Fieberg, and Shannon Barber-Meyer. 2018. “An Historical Overview and Update of Wolf–Moose Interactions in Northeastern Minnesota.” *Wildlife Society Bulletin* 42 (1): 40–47.

Meier, Ulrich. 2006. “A Note on the Power of Fisher’s Least Significant Difference Procedure.” *Pharmaceutical Statistics: The Journal of Applied Statistics in the Pharmaceutical Industry* 5 (4): 253–63.

Middleton, Kevin M., and Randall Pruim. 2015. *Abd: The Analysis of Biological Data*. https://CRAN.R-project.org/package=abd.

Moran, Matthew D. 2003. "Arguments for Rejecting the Sequential Bonferroni in Ecological Studies." *Oikos* 100 (2): 403–5.

Morris, William, Daniel Doak, Martha Groom, Peter Kareiva, John Fieberg, Leah Gerber, Peter Murphy, and Diane Thomson. 1999. "A Practical Handbook for Population Viability Analysis." *The Nature Conservancy*.

Morrissey, Michael B, and Graeme D Ruxton. 2018. "Multiple Regression Is Not Multiple Regressions: The Meaning of Multiple Regression and the Non-Problem of Collinearity." *Philosophy, Theory, and Practice in Biology* 10 (3).

Mosteller, Frederick, and John W Tukey. 1968. "Data Analysis, Including Statistics." *Handbook of Social Psychology* 2: 80–203.

Mount, John, and Nina Zumel. 2021. *WVPlots: Common Plots for Analysis*. https://CRAN.R-project.org/package=WVPlots.

Moya-Laraño, Jordi, and Guadalupe Corcobado. 2008. "Plotting Partial Correlation and Regression in Ecological Studies." *Web Ecology* 8 (1): 35–46.

Muff, Stefanie, Leonhard Held, and Lukas F Keller. 2016. "Marginal or Conditional Regression Models for Correlated Non-Normal Data?" *Methods in Ecology and Evolution* 7 (12): 1514–24.

Muff, Stefanie, Johannes Signer, and John Fieberg. 2020. "Accounting for Individual-Specific Variation in Habitat-Selection Studies: Efficient Estimation of Mixed-Effects Models Using Bayesian or Frequentist Computation." *Journal of Animal Ecology* 89 (1): 80–92.

Muggeo, Vito M. R., David C. Atkins, Robert J Gallop, and Sona Dimidjian. 2014. "Segmented Mixed Models with Random Changepoints: A Maximum Likelihood Approach with Application to Treatment for Depression Study." *Statistical Modelling* 14: 293–313.

Murdoch, Duncan, and Daniel Adler. 2021. *Rgl: 3D Visualization Using OpenGL*. https://CRAN.R-project.org/package=rgl.

Murtaugh, Paul A. 1989. "Size and Species Composition of Zooplankton in Experimental Ponds with and Without Fishes." *Journal of Freshwater Ecology* 5 (1): 27–38.

———. 2007. "Simplicity and Complexity in Ecological Data Analysis." *Ecology* 88 (1): 56–62.

———. 2014. "In Defense of p Values." *Ecology* 95 (3): 611–17.

Nakagawa, Shinichi. 2004. "A Farewell to Bonferroni: The Problems of Low Statistical Power and Publication Bias." *Behavioral Ecology* 15 (6): 1044–45.

Nakagawa, Shinichi, and Holger Schielzeth. 2013. "A General and Simple Method for Obtaining R2 from Generalized Linear Mixed-Effects Models." *Methods in Ecology and Evolution* 4 (2): 133–42.

O'Hara, Robert, and Johan Kotze. 2010. "Do Not Log-Transform Count Data." *Nature Precedings*, 1–1.

Oberpriller, Johannes, Melina de Souza Leite, and Maximilian Pichler. 2021. "Fixed or Random? On the Reliability of Mixed-Effect Models for a Small Number of Levels in Grouping Variables." *bioRxiv*.

Ogle, Derek H., Jason C. Doll, Powell Wheeler, and Alexis Dinno. 2021. *FSA: Fisheries Stock Analysis*. https://github.com/droglenc/FSA.

Otis, David L, Kenneth P Burnham, Gary C White, and David R Anderson. 1978. "Statistical Inference from Capture Data on Closed Animal Populations." *Wildlife Monographs*, no. 62: 3–135.

Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

Pearl, Judea. 1995. "Causal Diagrams for Empirical Research." *Biometrika* 82 (4): 669–88. https://doi.org/10.1093/biomet/82.4.669.

Pearl, Judea, Madelyn Glymour, and Nicholas P Jewell. 2016. *Causal Inference in Statistics: A Primer*. John Wiley & Sons.

Pearl, Judea, and Dana Mackenzie. 2018. *The Book of Why: The New Science of Cause and Effect*. Basic books.

Perneger, Thomas V. 1998. "What's Wrong with Bonferroni Adjustments." *Bmj* 316 (7139): 1236–38.

Perperoglou, Aris, Willi Sauerbrei, Michal Abrahamowicz, and Matthias Schmid. 2019. "A Review of Spline Function Procedures in r." *BMC Medical Research Methodology* 19 (1): 1–16.

Pewsey, Arthur, Markus Neuhäuser, and Graeme D Ruxton. 2013. *Circular Statistics in r*. Oxford University Press.

Piegorsch, Walter W, and A John Bailer. 2005. *Analyzing Environmental Data*. John Wiley & Sons.

Pike, Nathan. 2011. "Using False Discovery Rates for Multiple Comparisons in Ecology and Evolution." *Methods in Ecology and Evolution* 2 (3): 278–82.

Pinheiro, Jose, Douglas Bates, Saikat DebRoy, Deepayan Sarkar, and R Core Team. 2021. *nlme: Linear and Nonlinear Mixed Effects Models*. https://CRAN.R-project.org/package=nlme.

Pinheiro, José, and Douglas Bates. 2006. *Mixed-Effects Models in s and s-PLUS*. Springer Science & Business Media.

Plummer, Martyn et al. 2003. "JAGS: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling." In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, 124:1–10. 125.10. Vienna, Austria.

Pollock, Kenneth H, James D Nichols, Cavell Brownie, and James E Hines. 1990. "Statistical Inference for Capture-Recapture Experiments." *Wildlife Monographs*, 3–97.

Ponisio, Lauren C, Perry de Valpine, Nicholas Michaud, and Daniel Turek. 2020. "One Size Does Not Fit All: Customizing MCMC Methods for Hierarchical Models Using NIMBLE." *Ecology and Evolution* 10 (5): 2385–2416.

Puth, Marie-Therese, Markus Neuhäuser, and Graeme D Ruxton. 2015. "On the Variety of Methods for Calculating Confidence Intervals by Bootstrapping." *Journal of Animal Ecology* 84 (4): 892–97.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Reitan, Trond, and Anders Nielsen. 2016. "Do Not Divide Count Data with Count Data; a Story from Pollination Ecology with Implications Beyond." *PloS One* 11 (2).

Ritz, John, and Donna Spiegelman. 2004. "Equivalence of Conditional and Marginal Regression Models for Clustered and Longitudinal Data." *Statistical Methods in Medical Research* 13 (4): 309–23.

Rizopoulos, Dimitris. 2021. *GLMMadaptive: Generalized Linear Mixed Models Using Adaptive Gaussian Quadrature*. https://CRAN.R-project.org/package=GLMMadaptive.

Roback, Paul, and Julie Legler. 2021. *Beyond Multiple Linear Regression: Applied Generalized Linear Models and Multilevel Models in r*. Chapman; Hall/CRC.

Roberts, David R, Volker Bahn, Simone Ciuti, Mark S Boyce, Jane Elith, Gurutzeta Guillera-Arroita, Severin Hauenstein, et al. 2017. "Cross-Validation Strategies for Data with Temporal, Spatial, Hierarchical, or Phylogenetic Structure." *Ecography* 40 (8): 913–29.

Robinson, David, Alex Hayes, and Simon Couch. 2021. *Broom: Convert Statistical Objects into Tidy Tibbles*. https://CRAN.R-project.org/package=broom.

Rok Blagus. 2017. *Abe: Augmented Backward Elimination*. https://CRAN.R-project.org/package=abe.

Rossman, Allan J. 1994. "Televisions, Physicians, and Life Expectancy." *Journal of Statistics Education* 2 (2).

Ruxton, Graeme D, and Markus Neuhäuser. 2010. "When Should We Use One-Tailed Hypothesis Testing?" *Methods in Ecology and Evolution* 1 (2): 114–17.

Sackett, David L. 1979. "Bias in Analytic Research." In *The Case-Control Study Consensus and Controversy*, 51–63. Elsevier.

Säfken, Benjamin, David Rügamer, Thomas Kneib, and Sonja Greven. 2021. "Conditional Model Selection in Mixed-Effects Models with cAIC4." *Journal of Statistical Software* 99 (8): 1–30. https://doi.org/10.18637/jss.v099.i08.

Savage, Van M, and Geoffrey B West. 2007. "A Quantitative, Theoretical Framework for Understanding Mammalian Sleep." *Proceedings of the National Academy of Sciences* 104 (3): 1051–56.

Schad, Daniel J, Shravan Vasishth, Sven Hohenstein, and Reinhold Kliegl. 2020. "How to Capitalize on a Priori Contrasts in Linear (Mixed) Models: A Tutorial." *Journal of Memory and Language* 110: 104038.

Schaub, Michael, and Marc Kéry. 2021. *Integrated Population Models: Theory and Ecological Applications with r and JAGS*. Academic Press.

Scheipl, Fabian, Sonja Greven, and Helmut Kuechenhoff. 2008. "Size and Power of Tests for a Zero Random

Effect Variance or Polynomial Regression in Additive and Linear Mixed Models." *Computational Statistics & Data Analysis* 52 (7): 3283–99.

Schielzeth, Holger. 2010. "Simple Means to Improve the Interpretability of Regression Coefficients." *Methods in Ecology and Evolution* 1 (2): 103–13.

Schielzeth, Holger, and Wolfgang Forstmeier. 2009. "Conclusions Beyond Support: Overconfident Estimates in Mixed Models." *Behavioral Ecology* 20 (2): 416–20.

Schloerke, Barret, Di Cook, Joseph Larmarange, Francois Briatte, Moritz Marbach, Edwin Thoen, Amos Elberg, and Jason Crowley. 2023. *GGally: Extension to 'Ggplot2'*. https://CRAN.R-project.org/package=GGally.

Schwarz, C. J. 2014. *Ch 16: Regression with Pseudo-Replication. In Course Notes for Beginning and Intermediate Statistics.* Retrieved 2015-03-1 and http://people.stat.sfu.ca/~cschwarz/Stat-650/Notes/MyPrograms/Reg-pseudo/Se-Lake/Se-lake.html.

Scotson, Lorraine, Gabriella Fredriksson, Dusit Ngoprasert, Wai-Ming Wong, and John Fieberg. 2017. "Projecting Range-Wide Sun Bear Population Trends Using Tree Cover and Camera-Trap Bycatch Data." *PloS One* 12 (9).

Scott, Eric R, and Elizabeth E Crone. 2021. "Using the Right Tool for the Job: The Difference Between Unsupervised and Supervised Analyses of Multivariate Ecological Data." *Oecologia* 196 (1): 13–25.

Self, Steven G, and Kung-Yee Liang. 1987. "Asymptotic Properties of Maximum Likelihood Estimators and Likelihood Ratio Tests Under Nonstandard Conditions." *Journal of the American Statistical Association* 82 (398): 605–10.

Shipley, B. 2002. *Cause and Correlation in Biology: A User's Guide to Path Analysis, Structural Equations and Causal Inference*. Cambridge University Press.

Shmueli, Galit. 2010. "To Explain or to Predict?" *Statistical Science* 25 (3): 289–310.

Signer, Johannes, John Fieberg, and Tal Avgar. 2019. "Animal Movement Tools (Amt): R Package for Managing Tracking Data and Conducting Habitat Selection Analyses." *Ecology and Evolution* 9 (2): 880–90.

Sileshi, Gudeta. 2008. "The Excess-Zero Problem in Soil Animal Count Data and Choice of Appropriate Models for Statistical Inference." *Pedobiologia* 52 (1): 1–17.

Silk, Matthew J, Xavier A Harrison, and David J Hodgson. 2020. "Perils and Pitfalls of Mixed-Effects Regression Models in Biology." *PeerJ* 8: e9522.

Singer, Judith D, John B Willett, John B Willett, et al. 2003. *Applied Longitudinal Data Analysis: Modeling Change and Event Occurrence*. Oxford university press.

Stan Development Team. 2019. "RStan: The R Interface to Stan." http://mc-stan.org/.

Statisticat, and LLC. 2021. *LaplacesDemon: Complete Environment for Bayesian Inference*. Bayesian-Inference.com. https://web.archive.org/web/20150206004624/http://www.bayesian-inference.com/software.

Stewart, Peter S, Philip A Stephens, Russell A Hill, Mark J Whittingham, and Wayne Dawson. 2023. "Model Selection in Occupancy Models: Inference Versus Prediction." *Ecology* 104 (3): e3942.

Su, Yu-Sung, and Masanao Yajima. 2021. *R2jags: Using r to Run 'JAGS'*. https://CRAN.R-project.org/package=R2jags.

Su, Zhenming, and Ji X He. 2013. "Analysis of Lake Huron Recreational Fisheries Data Using Models Dealing with Excessive Zeros." *Fisheries Research* 148: 81–89.

Thompson, Sarah J, Douglas H Johnson, Neal D Niemuth, and Christine A Ribic. 2015. "Avoidance of Unconventional Oil Wells and Roads Exacerbates Habitat Loss for Grassland Birds in the North American Great Plains." *Biological Conservation* 192: 82–90.

Tibshirani, Robert. 1996. "Regression Shrinkage and Selection via the LASSO." *Journal of the Royal Statistical Society: Series B (Methodological)* 58 (1): 267–88.

———. 2011. "Regression Shrinkage and Selection via the Lasso: A Retrospective." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73 (3): 273–82.

Toms, Judith D, and Mary L Lesperance. 2003. "Piecewise Regression: A Tool for Identifying Ecological Thresholds." *Ecology* 84 (8): 2034–41.

Tredennick, Andrew T, Giles Hooker, Stephen P Ellner, and Peter B Adler. 2021. "A Practical Guide to Selecting Models for Exploration, Inference, and Prediction in Ecology." *Ecology* 102 (6): e03336.

Vaida, Florin, and Suzette Blanchard. 2005. "Conditional Akaike Information for Mixed-Effects Models." *Biometrika* 92 (2): 351–70.

Valavi, Roozbeh, Jane Elith, José J. Lahoz-Monfort, and Gurutzeta Guillera-Arroita. 2019. "blockCV: An r Package for Generating Spatially or Environmentally Separated Folds for k-Fold Cross-Validation of Species Distribution Models." *Methods in Ecology and Evolution* 10 (2): 225–32.

Vaughan, Davis. 2021. *Workflows: Modeling Workflows*. https://CRAN.R-project.org/package=workflows.

Vehtari, Aki, Jonah Gabry, Mans Magnusson, Yuling Yao, Paul-Christian Bürkner, Topi Paananen, and Andrew Gelman. 2020. "Loo: Efficient Leave-One-Out Cross-Validation and WAIC for Bayesian Models." https://mc-stan.org/loo/.

Vehtari, Aki, Andrew Gelman, and Jonah Gabry. 2017. "Practical Bayesian Model Evaluation Using Leave-One-Out Cross-Validation and WAIC." *Statistics and Computing* 27: 1413–32. https://doi.org/10.1007/s11222-016-9696-4.

Vehtari, Aki, and Jouko Lampinen. 2002. "Bayesian Model Assessment and Comparison Using Cross-Validation Predictive Densities." *Neural Computation* 14 (10): 2439–68.

Venables, W. N., and B. D. Ripley. 2002. *Modern Applied Statistics with s*. Fourth. New York: Springer. https://www.stats.ox.ac.uk/pub/MASS4/.

Verhoeven, Koen JF, Katy L Simonsen, and Lauren M McIntyre. 2005. "Implementing False Discovery Rate Control: Increasing Your Power." *Oikos* 108 (3): 643–47.

Warton, David I. 2005. "Many Zeros Does Not Mean Zero Inflation: Comparing the Goodness-of-Fit of Parametric Models to Multivariate Abundance Data." *Environmetrics: The Official Journal of the International Environmetrics Society* 16 (3): 275–89.

Warton, David I, Mitchell Lyons, Jakub Stoklosa, and Anthony R Ives. 2016. "Three Points to Consider When Choosing a LM or GLM Test for Count Data." *Methods in Ecology and Evolution* 7 (8): 882–90.

Watanabe, Sumio, and Manfred Opper. 2010. "Asymptotic Equivalence of Bayes Cross Validation and Widely Applicable Information Criterion in Singular Learning Theory." *Journal of Machine Learning Research* 11 (12).

Weber, Patrick, Karin Binder, and Stefan Krauss. 2018. "Why Can Only 24% Solve Bayesian Reasoning Problems in Natural Frequencies: Frequency Phobia in Spite of Probability Blindness." *Frontiers in Psychology* 9: 1833.

White, Halbert. 1982. "Maximum Likelihood Estimation of Misspecified Models." *Econometrica: Journal of the Econometric Society*, 1–25.

Whitlock, Michael, and Dolph Schluter. 2015. *The Analysis of Biological Data*. Roberts Publishers.

Whitman, Karyl, Anthony M Starfield, Henley S Quadling, and Craig Packer. 2004. "Sustainable Trophy Hunting of African Lions." *Nature* 428 (6979): 175.

Whittingham, Mark J, Philip A Stephens, Richard B Bradbury, and Robert P Freckleton. 2006. "Why Do We Still Use Stepwise Modelling in Ecology and Behaviour?" *Journal of Animal Ecology* 75 (5): 1182–89.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

———. 2021. *Tidyr: Tidy Messy Data*. https://CRAN.R-project.org/package=tidyr.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2021. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Williamson, John M. 2014. "Informative Cluster Size." *Wiley StatsRef: Statistics Reference Online*, 1–2.

Wolfson, David W, David E Andersen, and John Fieberg. 2022. "Using Piecewise Regression to Identify Biological Phenomena in Biotelemetry Datasets." *Journal of Animal Ecology* 91 (9): 1755–69.

Wood, S. N. 2017. *Generalized Additive Models: An Introduction with r*. 2nd ed. Chapman; Hall/CRC.

Wood, S. N. 2004. "Stable and Efficient Multiple Smoothing Parameter Estimation for Generalized Additive Models." *Journal of the American Statistical Association* 99 (467): 673–86.

Wright, Sewall. 1921. "Correlation and Causation." *Journal of Agricultural Research* 20: 557–85.

Xie, Yihui. 2016. *Bookdown: Authoring Books and Technical Documents with R Markdown*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/bookdown.

———. 2022. *Bookdown: Authoring Books and Technical Documents with R Markdown*. https://github.com/rstudio/bookdown.

Yackulic, Charles B, Michael Dodrill, Maria Dzul, Jamie S Sanderlin, and Janice A Reid. 2020. "A Need for Speed in Bayesian Population Models: A Practical Guide to Marginalizing and Recovering Discrete Latent States." *Ecological Applications* 30 (5): e02112.

Yan, Jun. 2002. "Geepack: Yet Another Package for Generalized Estimating Equations." *R-News* 2/3: 12–14.

Young, Mary L, John S Preisser, Bahjat F Qaqish, and Mark Wolfson. 2007. "Comparison of Subject-Specific and Population Averaged Models for Count Data from Cluster-Unit Intervention Trials." *Statistical Methods in Medical Research* 16 (2): 167–84.

Youngflesh, Casey. 2018. "MCMCvis: Tools to Visualize, Manipulate, and Summarize MCMC Output." *Journal of Open Source Software* 3 (24): 640. https://doi.org/10.21105/joss.00640.

Zeger, Scott L, Kung-Yee Liang, and Paul S Albert. 1988. "Models for Longitudinal Data: A Generalized Estimating Equation Approach." *Biometrics*, 1049–60.

Zicus, Michael C, John Fieberg, and David P Rave. 2003. "Does Mallard Clutch Size Vary with Landscape Composition: A Different View." *The Wilson Journal of Ornithology* 115 (4): 409–14.

Zicus, Michael C, David P Rave, Abhik Das, Michael R Riggs, and Michelle L Buitenwerf. 2006. "Influence of Land Use on Mallard Nest-Structure Occupancy." *The Journal of Wildlife Management* 70 (5): 1325–33.

Zicus, Michael C, David P Rave, and John Fieberg. 2006. "Cost-Effectiveness of Single-Versus Double-Cylinder over-Water Nest Structures." *Wildlife Society Bulletin* 34 (3): 647–55.

Zuur, Alain, Elena N Ieno, and Chris S Elphick. 2010. "A Protocol for Data Exploration to Avoid Common Statistical Problems." *Methods in Ecology and Evolution* 1 (1): 3–14.

Zuur, Alain, Elena N Ieno, Neil Walker, Anatoly A Saveliev, and Graham M Smith. 2009. *Mixed Effects Models and Extensions in Ecology with r*. Springer Science & Business Media.