Supporting Data Analysis for a talk to be given at Evolution 2008 University of Minnesota, June 20–24

By
Charles J. Geyer and Ruth G. Shaw

Technical Report No. 669

School of Statistics
University of Minnesota

May 14, 2008

Abstract

A solution to the problem of estimating fitness landscapes was proposed by Lande and Arnold (1983). Another solution, which avoids problematic aspects of the Lande-Arnold methodology, was proposed by Shaw, Geyer, Wagenius, Hangelbroek, and Etterson (2008), who also provided an illustrative example. Here we provide another example using simulated data that are more suitable to aster analysis.

All analyses are done in R (R Development Core Team, 2008) using the `aster` contributed package described by Geyer et al. (2007) except for analyses in the style of Lande and Arnold (1983), which use ordinary least squares regression. Furthermore, all analyses are done using the `Sweave` function in R, so this entire technical report and all of the analyses reported in it are completely reproducible by anyone who has R with the `aster` package installed and the R noweb file specifying the document.

1 R Package Aster

We use R statistical computing environment (R Development Core Team, 2008) in our analysis. It is free software and can be obtained from `http://cran.r-project.org`. Precompiled binaries are available for Windows, Macintosh, and popular Linux distributions. We use the contributed package `aster`. If R has been installed, but this package has not yet been installed, do

```
install.packages("aster")
```

from the R command line (or do the equivalent using the GUI menus if on Apple Macintosh or Microsoft Windows). This may require root or administrator privileges.

Assuming the `aster` package has been installed, we load it

```
> library(aster)
```

The version of the package used to make this document is 0.7-4 (which is available on CRAN). The version of R used to make this document is 2.7.0.

This entire document and all of the calculations shown were made using the R command `Sweave` and hence are exactly reproducible by anyone who has R and the R noweb (RNW) file from which it was created. Both the RNW file and and the PDF document produced from it are available at `http://www.stat.umn.edu/geyer/aster`. For further details on the use of Sweave and R see Chapter 1 of the technical report by Shaw, et al. (2007a) available at the same web site.

Not only can one exactly reproduce the results in the printable document, one can also modify the parameters of the simulation and get different results. Obvious modifications to try are noted on pages 1, 4, 6, and 10 below. But, of course, anything at all can be changed once one has the RNW file.

Finally, we set the seeds of the random number generator so that we obtain the same results every time. To get different results, obtain the RNW file, change this statement, and reprocess using `Sweave` and LaTeX.

```
> set.seed(42)
```

2 Data Structure

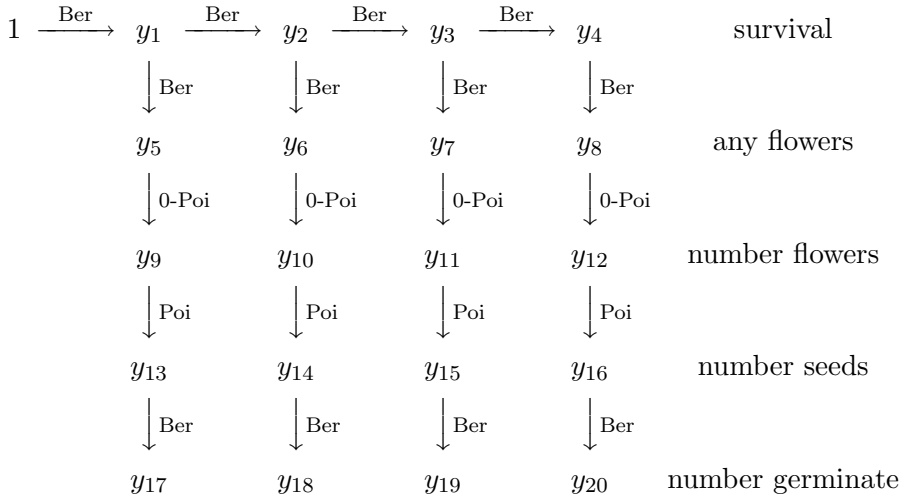
We simulate data because there does not, to our knowledge, exist a data set that can show the full potential of aster analysis. Our simulated data have three important characteristics

1. (simulated) phenotypic trait measurements,
2. graphical model in which not all predecessor variables are Bernoulli, and
3. fitness is the sum of reproduction variables for many time periods.

See Shaw, et al. (2008) for an example showing the same kind of analysis we do here with real data having feature 1 above. See Geyer et al. (2007) for an example with feature 3 above. There are, to our knowledge, no published examples with feature 2 above. Since any or all of these features may arise in practical examples, we use this example as a good illustration of what is possible with aster.

2.1 Graph

We use the following aster model graphical structure. This is the subgraph for a single individual; the full graph consists of n isomorphic copies of this subgraph, one for each of n individuals.



Letters y_j represent random variables. Arrows represent conditional distributions (more on this below). Subgraphs for different individuals differ only in the subscripts for the y_j, each individual having a different set of variables.

2.2 Variables and Their Conditional Distributions

The variables for one individual are the y_j in the graph. Those in each column represent the data for one time period. Those in each row represent the data for one kind of variable, one "component of fitness." The layers are

- y_1, ..., y_4 are survival indicators (zero if dead, one if alive).

- y_5, ..., y_8 are flowering indicators (zero if no flowers, one if one or more flowers).
- y_9, ..., y_{12} are flowering counts (number of flowers).
- y_{13}, ..., y_{16} are seed counts (number of seeds).
- y_{17}, ..., y_{20} are germination counts (number of seeds that germinate).

It is important to understand, so we emphasize this point, that everything is counted in each time period. In the first time period, $y_1 = 1$ if and only if the individual is alive, $y_5 = 1$ if and only if the individual has at least one flower, y_9 counts all of the flowers, y_{13} counts all of the seeds produced by all of those flowers, and y_{17} counts all of those seeds that germinate. It is possible to collect data on only a sample of flowers or only a sample of seeds — this is discussed in Section 8 of a technical report by Shaw, et al. (2007b) — but we are not doing that in this example.

The conditional distributions of one variable given another are indicated by the text over the arrows. An arrow $y_j \longrightarrow y_k$ indicates that y_k is the sum of y_j independent and identically distributed (IID) random variables with the distribution named by the text over the arrow. The sum of zero things is zero by convention, so $y_j = 0$ implies $y_k = 0$.

- Ber is for Bernoulli. A random variable is Bernoulli if and only if its only possible values are zero and one. The sum of IID Bernoullis is binomial. Thus, e. g., y_{17} is binomial with sample size y_{13}.
- 0-Poi is for zero-truncated Poisson, meaning a Poisson random variable conditioned on being nonzero. In a graph

  $$y_j \xrightarrow{\text{Ber}} y_k \xrightarrow{\text{0-Poi}} y_l$$

  the conditional distribution of y_l given y_j is zero-inflated Poisson, and this is the only way zero-inflated Poisson can appear in an aster model.
- Poi is for Poisson. The sum of IID Poisson is again Poisson, thus, e. g., the conditional distribution of y_{13} given y_9 is Poisson with mean that is y_9 times a constant (which is a function of the parameters of the model).

2.3 Fitness

In this model fitness (more pedantically, the best surrogate of fitness) is the sum of the variables in the bottom layer, $y_{17} + y_{18} + y_{19} + y_{20}$, the total lifetime (more pedantically, the total over the four time periods) number of seeds produced that germinate. Expected fitness is the sum of the corresponding mean value parameters $\mu_{17} + \mu_{18} + \mu_{19} + \mu_{20}$, where $\mu_j = E(y_j)$.

Readers may ask, why only those variables, don't the other components of fitness count too? They do count. A seed can't germinate if it doesn't exist, there can't be seeds if there weren't flowers, and there can't be flowers if the individual is dead. The total number of seeds that germinate incorporates all earlier components of fitness. Moreover, statistical theory says the other components of fitness do "count" even though they aren't counted explicitly. Maximum likelihood estimation uses all the data to calculate the most efficient possible estimates.

2.4 Setup

The following R statements set up this graphical structure

```
> pred <- seq(1, 20) - 4
> pred[1:4] <- 0:3
> fam <- rep(c(1, 1, 3, 2, 1), each = 4)
> matrix(pred, 5, 4, byrow = TRUE)
```

```
     [,1] [,2] [,3] [,4]
[1,]    0    1    2    3
[2,]    1    2    3    4
[3,]    5    6    7    8
[4,]    9   10   11   12
[5,]   13   14   15   16
```

```
> matrix(fam, 5, 4, byrow = TRUE)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    1    1    1    1
[3,]    3    3    3    3
[4,]    2    2    2    2
[5,]    1    1    1    1
```

`pred` and `fam` are displayed as matrices so they have the same layout as the graph.

3 Data Simulation

3.1 Flat Fitness Landscape

We first simulate data with the following parameters

```
> psurv <- 0.9
> pflow <- 0.8
> mflow <- 5
> mseed <- 10
> pgerm <- 0.1
```

Anyone wishing to see the results of changing these parameters can obtain the R noweb (RNW) file from which this document was created (`http://www.stat.umn.edu/geyer/aster`), change these parameters and rerun.

The meaning of these parameters is

- `psurv` is the conditional mean value parameter for Bernoulli distributions in the first (survival) layer (of y_1, ..., y_4 given their predecessors).
- `pflow` is the conditional mean value parameter for Bernoulli distributions in the second (any flowers) layer (of y_5, ..., y_8 given their predecessors).

- `mflow` is the conditional mean value parameter before truncation for zero-truncated Poisson distributions in the third (number flowers) layer (of y_9, ..., y_{12} given their predecessors). See below for meaning of "before truncation."
- `mseed` is the conditional mean value parameter for Poisson distributions in the fourth (number seeds) layer (of y_{13}, ..., y_{16} given their predecessors).
- `pgerm` is the conditional mean value parameter for Bernoulli distributions in the bottom (number germinate) layer (of y_{17}, ..., y_{20} given their predecessors).

We now explain the meaning of "before truncation" in the definition of `mflow`. Referring to the discussion of truncated Poisson distributions on the help page `?families` we see that if a Poisson distribution having (untruncated) mean `mflow` is zero-truncated, the corresponding conditional mean is

```
> beta.flow <- ppois(1, mflow, lower.tail = FALSE)/dpois(1,
+     mflow)
> mflow + 1/(1 + beta.flow)

[1] 5.033918
```

Hence the conditional mean after truncation 5.0339 is slightly more than the conditional mean before truncation `mflow` $= 5$.

Some facts that are perhaps not completely obvious

- The conditional distribution of y_9 given y_1 is zero-inflated Poisson (and similarly for y_{10} given y_2, etc.)
- The conditional distribution of y_{13} given y_9 is Poisson with mean $y_9 \times$ `mflow` (and similarly for y_{14} given y_{10}, etc.)
- The conditional distribution of y_{17} given y_{13} is binomial with sample size y_{13} and success probability `pgerm` (and similarly for y_{18} given y_{14}, etc.)

Unconditional mean value parameters are found by multiplying conditional ones (in an aster model, not in general).

```
> xi <- matrix(c(psurv, pflow, mflow, mseed, pgerm),
+     5, 4)
> xi

     [,1] [,2] [,3] [,4]
[1,]  0.9  0.9  0.9  0.9
[2,]  0.8  0.8  0.8  0.8
[3,]  5.0  5.0  5.0  5.0
[4,] 10.0 10.0 10.0 10.0
[5,]  0.1  0.1  0.1  0.1
```

```
> mu <- xi
> mu[1, ] <- cumprod(mu[1, ])
> mu <- apply(mu, 2, cumprod)
> mu
```

```
      [,1]   [,2]    [,3]     [,4]
[1,]  0.90  0.810  0.7290  0.65610
[2,]  0.72  0.648  0.5832  0.52488
[3,]  3.60  3.240  2.9160  2.62440
[4,] 36.00 32.400 29.1600 26.24400
[5,]  3.60  3.240  2.9160  2.62440
```

Thus expected fitness — lifetime expected number of germinating seeds — in this model (with flat fitness landscape) is

```
> sum(mu[5, ])
```

```
[1] 12.3804
```

The following are the conditional canonical parameters corresponding to the conditional mean value parameters defined above.

```
> theta.surv <- log(psurv) - log(1 - psurv)
> theta.flow <- log(pflow) - log(1 - pflow)
> theta.nflow <- log(mflow)
> theta.nseed <- log(mseed)
> theta.germ <- log(pgerm) - log(1 - pgerm)
> theta <- matrix(c(theta.surv, theta.flow, theta.nflow,
+     theta.nseed, theta.germ), 5, 4)
> theta
```

```
          [,1]      [,2]      [,3]      [,4]
[1,]  2.197225  2.197225  2.197225  2.197225
[2,]  1.386294  1.386294  1.386294  1.386294
[3,]  1.609438  1.609438  1.609438  1.609438
[4,]  2.302585  2.302585  2.302585  2.302585
[5,] -2.197225 -2.197225 -2.197225 -2.197225
```

Now we are ready to simulate some data. First we set the number of individuals.

```
> nind <- 500
```

(This too can be changed by anyone who has obtained the RNW source for this document.)

Referring to the help page `?raster` we see that to simulate data on n individuals, each having the same graph with k nodes (variables), we hand the `raster` function an $n \times k$ matrix `theta` whose rows are the conditional canonical parameter vectors for each individual. In this case, since the fitness surface is flat, each individual has the same parameter vector.

```
> theta.mat <- matrix(as.vector(t(theta)), nind, length(theta),
+     byrow = TRUE)
> y <- raster(theta.mat, pred, fam, root = theta.mat^0)
> dim(y)

[1] 500  20
```

We also simulate a bivariate normal trait vector

```
> library(MASS)
> z <- mvrnorm(nind, mu = c(0, 0), Sigma = matrix(c(1,
+     0.5, 0.5, 1), 2, 2))
> dim(z)

[1] 500   2
```

We then combine this in the usual way (see help page `?aster`) to make a data frame ready for aster analysis

```
> data.flat <- cbind(y, z)
> vars <- outer(c("isurv", "iflow", "nflow", "nseed",
+     "ngerm"), 1:4, paste, sep = "")
> vars

     [,1]     [,2]     [,3]     [,4]
[1,] "isurv1" "isurv2" "isurv3" "isurv4"
[2,] "iflow1" "iflow2" "iflow3" "iflow4"
[3,] "nflow1" "nflow2" "nflow3" "nflow4"
[4,] "nseed1" "nseed2" "nseed3" "nseed4"
[5,] "ngerm1" "ngerm2" "ngerm3" "ngerm4"

> vars <- as.vector(t(vars))
> colnames(data.flat) <- c(vars, "z1", "z2")
> data.flat <- as.data.frame(data.flat)
> redata.flat <- reshape(data.flat, varying = list(vars),
+     direction = "long", timevar = "varb", times = as.factor(vars),
+     v.names = "resp")
> redata.flat <- data.frame(redata.flat, root = 1)
> names(redata.flat)

[1] "z1"   "z2"   "varb" "resp" "id"   "root"
```

We are now ready to fit our first aster model

```
> out1 <- aster(resp ~ varb + 0, pred, fam, varb, id,
+     root, data = redata.flat, type = "conditional")
> summary(out1)
```

```
Call:
aster.formula(formula = resp ~ varb + 0, pred = pred, fam = fam,
    varvar = varb, idvar = id, root = root, data = redata.flat,
    type = "conditional")

            Estimate Std. Error z value Pr(>|z|)
varbiflow1  1.219690   0.113247   10.77   <2e-16 ***
varbiflow2  1.233090   0.121140   10.18   <2e-16 ***
varbiflow3  1.389809   0.132640   10.48   <2e-16 ***
varbiflow4  1.445954   0.142318   10.16   <2e-16 ***
varbisurv1  2.050519   0.140717   14.57   <2e-16 ***
varbisurv2  1.995855   0.146397   13.63   <2e-16 ***
varbisurv3  2.348570   0.179501   13.08   <2e-16 ***
varbisurv4  2.184802   0.175792   12.43   <2e-16 ***
varbnflow1  1.671257   0.023701   70.51   <2e-16 ***
varbnflow2  1.630806   0.025788   63.24   <2e-16 ***
varbnflow3  1.604615   0.026933   59.58   <2e-16 ***
varbnflow4  1.622221   0.027978   57.98   <2e-16 ***
varbngerm1 -2.226307   0.024942  -89.26   <2e-16 ***
varbngerm2 -2.237824   0.027171  -82.36   <2e-16 ***
varbngerm3 -2.253404   0.028674  -78.59   <2e-16 ***
varbngerm4 -2.219203   0.029260  -75.84   <2e-16 ***
varbnseed1  2.302749   0.007396  311.37   <2e-16 ***
varbnseed2  2.304452   0.008020  287.36   <2e-16 ***
varbnseed3  2.292662   0.008410  272.62   <2e-16 ***
varbnseed4  2.303267   0.008701  264.71   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

All of the estimates are about what they are supposed to be (statistics works, no surprise).

We now need to switch to an unconditional aster model because that's what we need to model fitness (Shaw, et al., 2008).

```
> out2 <- aster(resp ~ varb + 0, pred, fam, varb, id,
+     root, data = redata.flat)
```

3.2 A Digression on Aster Model Theory

We take a break from computing to re-explain a theoretical issue. Our previous attempt was Section 3.10 of Shaw, et al. (2007a). This time we use a slightly different argument based on the inequality

$$(\boldsymbol{\mu} - \boldsymbol{\mu}')^T (\boldsymbol{\varphi} - \boldsymbol{\varphi}') > 0 \tag{1}$$

which holds whenever $\boldsymbol{\varphi}$ and $\boldsymbol{\varphi}'$ are two distinct values of the linear predictor vector of an unconditional aster model and $\boldsymbol{\mu}$ and $\boldsymbol{\mu}'$ are the corresponding mean value parameter vectors (Barndorff-Nielsen, 1978, Equation 28, p. 121). A function that maps vector to

vectors and has property (1) is called *strictly monotone* (Rockafellar and Wets, 2004, Definition 12.1). This is the multivariate analog of strictly increasing functions that map real numbers to real numbers. Using this terminology, the mapping from the linear predictor parameter of an unconditional aster model to the unconditional mean value parameter is always strictly monotone. There is an analogous monotonicity relation for conditional aster models (Appendix B), but it is not useful in modeling fitness landscapes.

We consider the situation, which is the most common one, where fitness (or to be more pedantic the best surrogate of fitness) is the sum of data on a subset G of nodes of the graph. That is the situation in our example where G is the bottom layer, the germination nodes. Observed fitness is $\sum_{j \in G} y_j$, and expected fitness is $\sum_{j \in G} \mu_j$, where $\mu_j = E(y_j)$ denotes components of the mean value parameter vector $\boldsymbol{\mu}$.

These models are unconditional aster models in which the linear predictor for each germination node has the form

$$\varphi_j(\mathbf{x}, \mathbf{z}) = a_j(\mathbf{x}) + q(\mathbf{z}), \tag{2a}$$

that is, the linear predictor φ_j for germination node j is the same function $q(\mathbf{z})$ of trait values $\mathbf{z}$ plus a possibly different function $a_j(\mathbf{x})$ of other covariates $\mathbf{x}$. At other (non-germination, not bottom layer of graph) nodes the linear predictor is

$$\varphi_j(\mathbf{x}, \mathbf{z}) = a_j(\mathbf{x}) \tag{2b}$$

(does not actually depend on trait values $\mathbf{z}$ despite the notation). In applications the functions $a_j(\mathbf{x})$ and $q(\mathbf{z})$ also depend on the regression coefficients, hence are estimated rather than known, but this does not matter for the issue under discussion, so is not indicated in the notation.

Now if we consider the difference of linear predictor values for two individuals having different trait values $\mathbf{z}$ and $\mathbf{z}'$ but the same values $\mathbf{x}$ of other covariates, we get

$$\varphi_j(\mathbf{x}, \mathbf{z}) - \varphi_j(\mathbf{x}, \mathbf{z}') = \begin{cases} q(\mathbf{z}) - q(\mathbf{z}'), & j \in G \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Let $\boldsymbol{\mu}(\mathbf{x}, \mathbf{z})$ denote the corresponding mean value parameter vectors and $\mu_j(\mathbf{x}, \mathbf{z})$ their components. In this particular case (1) becomes

$$\big(\boldsymbol{\mu}(\mathbf{x}, \mathbf{z}) - \boldsymbol{\mu}(\mathbf{x}, \mathbf{z}')\big)^T \big(\boldsymbol{\varphi}(\mathbf{x}, \mathbf{z}) - \boldsymbol{\varphi}(\mathbf{x}, \mathbf{z}')\big) > 0 \tag{4}$$

and written out in coordinates this is

$$\sum_{j \in J} \big(\mu_j(\mathbf{x}, \mathbf{z}) - \mu_j(\mathbf{x}, \mathbf{z}')\big) \big(\varphi_j(\mathbf{x}, \mathbf{z}) - \varphi_j(\mathbf{x}, \mathbf{z}')\big) > 0$$

where J is the set of all nodes. Using (3), this becomes

$$\big(q(\mathbf{z}) - q(\mathbf{z}')\big) \sum_{j \in G} \big(\mu_j(\mathbf{x}, \mathbf{z}) - \mu_j(\mathbf{x}, \mathbf{z}')\big) > 0$$

or

$$q(\mathbf{z}) > q(\mathbf{z}') \quad \text{implies} \quad \sum_{j \in G} \mu_j(\mathbf{x}, \mathbf{z}) > \sum_{j \in G} \mu_j(\mathbf{x}, \mathbf{z}') \tag{5}$$

The sums on the right-hand side are expected fitness (unconditional expected number of germinated seeds summed over the four time periods). So this says, in short, that expected fitness is a strictly monotone function of $q(\mathbf{z})$. There exists a strictly increasing function $F_{\mathbf{x}}$ such that expected fitness is $F_{\mathbf{x}}[q(\mathbf{z})]$.

We reiterate that it is important that we compare individuals having different trait values $\mathbf{z}$ but the same values $\mathbf{x}$ of other covariates. Thus we treat $\mathbf{z}$ as a vector variable here and $\mathbf{x}$ as a fixed vector constant. The strictly increasing function $F_{\mathbf{x}}$ depends on which $\mathbf{x}$ value we fix, but this does not really matter since we do not have an explicit representation of this function anyway. The important fact is that it is monotone so facts about fitness transformed to the linear predictor scale, i. e., facts about $q(\mathbf{z})$, imply facts about fitness itself, i. e., $\boldsymbol{\mu}(\mathbf{x}, \mathbf{z})$.

Fitness itself, being bounded below by zero, is hard to model sensibly, and we know that in generalized linear models, much less in aster models, which generalize them, it makes no sense to model means directly. We have to work on the linear predictor scale to do any modeling at all. Thus it is crucial that whatever function $q(\mathbf{z})$ we use on the linear predictor scale maps monotonely to the mean value scale. Without this monotonicity property, we couldn't interpret $q(\mathbf{z})$: we wouldn't know that when $q(\mathbf{z})$ is high then fitness is high and vice versa.

We emphasize that the argument here applies to any aster model in which fitness is deemed $\sum_{j \in G} \mu_j(\mathbf{z})$. G can be any set of nodes; they don't have to be thought of as the "bottom layer" of the graph; they don't have to be somehow similar. This argument is further generalized in Appendix A.

3.3 Fitness Landscape Quadratic on Linear Predictor Scale

Following Lande and Arnold (1983) we use a quadratic function $q(\mathbf{z})$ to model fitness. Unlike them, we make fitness quadratic on the linear predictor scale rather than on the mean value scale. Figure 1 shows the scatter plot of the two (simulated) phenotypic traits z_1 and z_2.

We center our quadratic function $q(\mathbf{z})$ somewhat off-center in the scatter plot point cloud

```
> c1 <- 2
> c2 <- 0.5
> a11 <- -1
> a22 <- -0.5
> a12 <- 0.5
> b0 <- 0.1
> b1 <- 0.045
```

Anyone wishing to see the results of changing these parameters can obtain the R noweb (RNW) file from which this document was created change these parameters and rerun. Then we define

$$\begin{aligned} q(\mathbf{z}) &= b_0 + b_1\bigl(a_{11}(z_1 - c_1)^2 + a_{12}(z_1 - c_1)(z_2 - c_2) + a_{22}(z_2 - c_2)^2\bigr) \\ &= b_0 + b_1\bigl[(a_{11}c_1^2 + a_{12}c_1c_2 + a_{22}c_2^2) - (2a_{11}c_1 + a_{12}c_2)z_1 - (2a_{22}c_2 + a_{12}c_1)z_2 \\ &\qquad + a_{11}z_1^2 + a_{22}z_2^2 + 2a_{12}z_1z_2\bigr] \end{aligned}$$

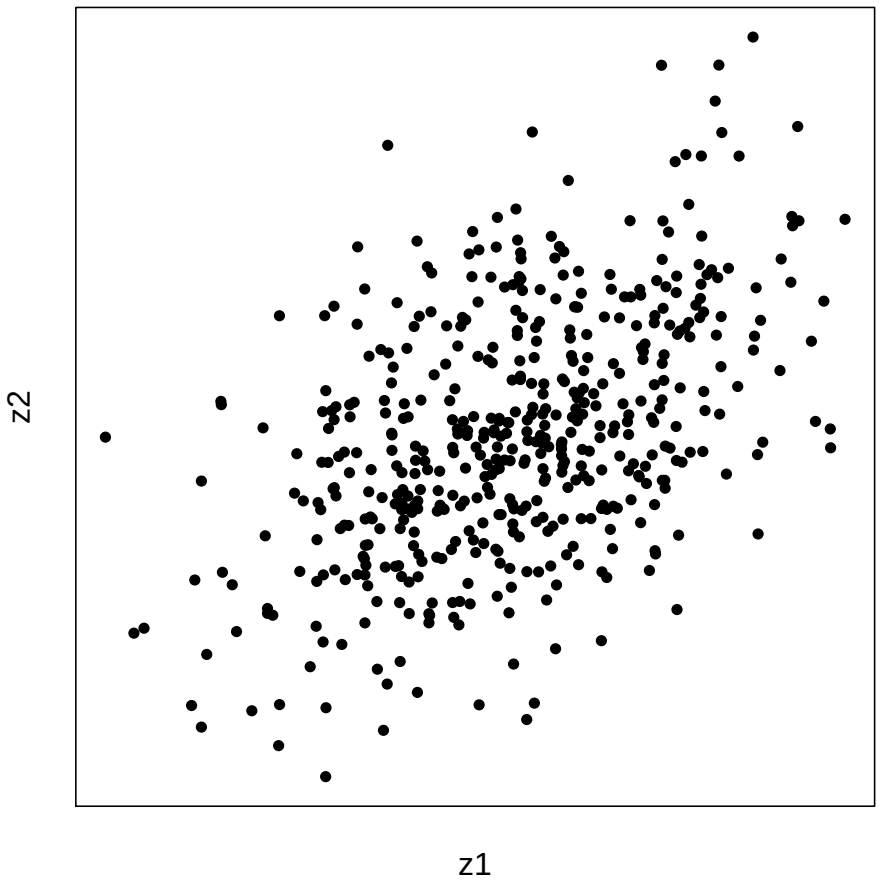


Figure 1: Scatterplot of simulated phenotypic traits z_1 and z_2.

Note that because a_{11} and a_{22} are both negative, this is a case of stabilizing selection. Since this is a simulation, we know the "simulation truth" parameter values, so there is no question about what the estimates should be estimating.

First we need to set up the model structure for the quadratic model. To do that we fit the desired model to the data we have now. The parameter values are not interesting, but the structure of the parameter vector (the meaning of each regression coefficient) is what we need.

To do this we need to play a trick on the R formula mini-language. We want the trait values **z** to not count except for germination nodes. Thus we set them to zero for other nodes.

```
> layer <- substr(as.character(redata.flat$varb), 1,
+     5)
> unique(layer)

[1] "isurv" "iflow" "nflow" "nseed" "ngerm"

> redata.curve <- redata.flat
> redata.curve$z1[layer != "ngerm"] <- 0
> redata.curve$z2[layer != "ngerm"] <- 0
```

We then fit the model of interest

```
> out3 <- aster(resp ~ varb + 0 + z1 + z2 + I(z1^2) +
+     I(z1 * z2) + I(z2^2), pred, fam, varb, id, root,
+     data = redata.curve)
```

Now we adjust the coefficients to follow our quadratic model

```
> coef <- out3$coef
> const <- b0 + b1 * (a11 * c1^2 + a12 * c1 * c2 +
+     a22 * c2^2)
> coef["varbngerm1"] <- coef["varbngerm1"] + const
> coef["varbngerm2"] <- coef["varbngerm2"] + const
> coef["varbngerm3"] <- coef["varbngerm3"] + const
> coef["varbngerm4"] <- coef["varbngerm4"] + const
> coef["z1"] <- b1 * (-a11 * 2 * c1 - a12 * c2)
> coef["z2"] <- b1 * (-a22 * 2 * c2 - a12 * c1)
> coef["I(z1^2)"] <- b1 * a11
> coef["I(z1 * z2)"] <- b1 * a12
> coef["I(z2^2)"] <- b1 * a22
> beta.true <- coef
```

Then we plug this back into the `out3` structure because we need it as an argument to the `predict` function.

```
> out3$coefficients <- beta.true
> mu.true <- predict(out3)
```

```
> phi.true <- predict(out3, parm.type = "canonical")
> theta.true <- predict(out3, parm.type = "canonical",
+     model.type = "conditional")
> sum(mu.true[layer == "ngerm"])/nind

[1] 8.181325
```

Now we are ready to simulate the data of interest.

```
> theta.mat <- matrix(theta.true, nrow = nind)
> y <- raster(theta.mat, pred, fam, root = theta.mat^0)
> dim(y)

[1] 500  20

> dim(redata.curve)

[1] 10000     6

> redata.curve$resp <- as.vector(y)
```

Now we have simulated the data we want and put it in its proper location in the proper order.

So now we are ready to fit some models to these data.

```
> out4 <- aster(resp ~ varb + 0, pred, fam, varb, id,
+     root, data = redata.curve)
> out5 <- aster(resp ~ varb + 0 + z1 + z2, pred, fam,
+     varb, id, root, data = redata.curve)
> out6 <- aster(resp ~ varb + 0 + z1 + z2 + I(z1^2) +
+     I(z1 * z2) + I(z2^2), pred, fam, varb, id, root,
+     data = redata.curve)
> anova(out4, out5, out6)

Analysis of Deviance Table

Model 1: resp ~ varb + 0
Model 2: resp ~ varb + 0 + z1 + z2
Model 3: resp ~ varb + 0 + z1 + z2 + I(z1^2) + I(z1 * z2) + I(z2^2)
  Model Df Model Dev Df Deviance P(>|Chi|)
1       20    -84703
2       22    -84959  2      256 2.704e-56
3       25    -84975  3       16 9.570e-04
```

The P-value for the comparison of "Model 2" in which $q(\mathbf{z})$ is linear in $\mathbf{z}$ and "Model 3" in which $q(\mathbf{z})$ is quadratic in $\mathbf{z}$ shows that the fit of Model 3 is highly statistically significantly better than that of Model 2 ($P = 9.6 \times 10^{-4}$) but not so significant that statistical analysis seems unnecessary. We take this data set as our simulated data.

Here are the regression coefficients

```
> summary(out6)

Call:
aster.formula(formula = resp ~ varb + 0 + z1 + z2 + I(z1^2) +
    I(z1 * z2) + I(z2^2), pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = redata.curve)

             Estimate Std. Error z value Pr(>|z|)
varbiflow1 -3.444251   0.180123 -19.122  < 2e-16 ***
varbiflow2 -3.064152   0.203311 -15.071  < 2e-16 ***
varbiflow3 -3.207467   0.218952 -14.649  < 2e-16 ***
varbiflow4 -3.284180   0.236597 -13.881  < 2e-16 ***
varbisurv1 -0.065167   0.160348  -0.406  0.68444
varbisurv2 -0.700847   0.225747  -3.105  0.00191 **
varbisurv3 -0.094013   0.275111  -0.342  0.73256
varbisurv4  1.217672   0.234288   5.197 2.02e-07 ***
varbnflow1 -7.264353   0.090581 -80.198  < 2e-16 ***
varbnflow2 -7.452760   0.102617 -72.627  < 2e-16 ***
varbnflow3 -7.227782   0.105711 -68.373  < 2e-16 ***
varbnflow4 -7.044131   0.107792 -65.349  < 2e-16 ***
varbngerm1 -2.264595   0.030308 -74.720  < 2e-16 ***
varbngerm2 -2.270312   0.033766 -67.237  < 2e-16 ***
varbngerm3 -2.325980   0.036102 -64.429  < 2e-16 ***
varbngerm4 -2.304824   0.036977 -62.332  < 2e-16 ***
varbnseed1  2.881224   0.009182 313.789  < 2e-16 ***
varbnseed2  2.895118   0.010258 282.241  < 2e-16 ***
varbnseed3  2.880964   0.010737 268.332  < 2e-16 ***
varbnseed4  2.864026   0.011117 257.619  < 2e-16 ***
z1          0.146950   0.013695  10.730  < 2e-16 ***
z2         -0.020598   0.009842  -2.093  0.03637 *
I(z1^2)    -0.027807   0.009508  -2.925  0.00345 **
I(z1 * z2)  0.023713   0.012352   1.920  0.05489 .
I(z2^2)    -0.017986   0.006536  -2.752  0.00593 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

3.4 Data for Lande-Arnold Analysis

These data can also be used for Lande-Arnold analysis, the comparison of the two methodologies being the main point of this technical report, but they must be reshaped for that. We need a data frame with three variables of length `nind`. The response is observed fitness

```
> yfit <- redata.curve$resp[redata.curve$varb == "ngerm1"] +
+     redata.curve$resp[redata.curve$varb == "ngerm2"] +
+     redata.curve$resp[redata.curve$varb == "ngerm3"] +
+     redata.curve$resp[redata.curve$varb == "ngerm4"]
```

and the predictors `z1` and `z2` can be taken from the data frame `data.flat`

```
> ladata <- data.frame(y = yfit, z1 = data.flat$z1,
+     z2 = data.flat$z2)
```

Now we fit a quadratic model to these data using ordinary least squares (OLS).

```
> lout <- lm(y ~ z1 + z2 + I(z1^2) + I(z1 * z2) + I(z2^2),
+     data = ladata)
> summary(lout)

Call:
lm(formula = y ~ z1 + z2 + I(z1^2) + I(z1 * z2) + I(z2^2), data = ladata)

Residuals:
     Min       1Q   Median       3Q      Max
-17.6321  -4.5203  -0.9696   3.7392  25.8501

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   7.6740     0.4449  17.250  < 2e-16 ***
z1            5.6369     0.3589  15.704  < 2e-16 ***
z2           -0.7077     0.3472  -2.038  0.04207 *
I(z1^2)       0.8166     0.3046   2.681  0.00758 **
I(z1 * z2)    0.3549     0.4389   0.809  0.41909
I(z2^2)      -0.7045     0.2739  -2.572  0.01040 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.912 on 494 degrees of freedom
Multiple R-squared: 0.3855,        Adjusted R-squared: 0.3793
F-statistic: 61.98 on 5 and 494 DF,  p-value: < 2.2e-16
```

3.5 Write Data

For use as a teaching tool, we write these data out for inclusion in a future version of the `aster` package. We also write out the simulation truth parameter values, and the graph and family structure. We also remove all other R objects so we know what we do henceforth depends only on the saved data.

```
> redata <- redata.curve
> save(redata, ladata, beta.true, mu.true, phi.true,
+     theta.true, pred, fam, vars, file = "sim.rda")
> rm(list = ls())
> ls(all.names = TRUE)

[1] ".Random.seed"

> load("sim.rda")
```

4 Estimation of Fitness Landscape

4.1 Plot Fitness Landscape

First we make a grid of points on which to evaluate fitness.

```
> par(mar = c(2, 2, 1, 1) + 0.1)
> ufoo <- par("usr")
> nx <- 101
> ny <- 101
> xfoo <- seq(ufoo[1], ufoo[2], length = nx)
> yfoo <- seq(ufoo[3], ufoo[4], length = ny)
```

Then we make a data frame like redata with these predictor values

```
> xx <- outer(xfoo, yfoo^0)
> yy <- outer(xfoo^0, yfoo)
> xx <- as.vector(xx)
> yy <- as.vector(yy)
> nn <- length(xx)
> foo <- rep(1, nn)
> bar <- list(z1 = xx, z2 = yy, root = foo)
> for (lab in levels(redata$varb)) {
+     bar[[lab]] <- foo
+ }
> bar <- as.data.frame(bar)
> rebar <- reshape(bar, varying = list(vars), direction = "long",
+     timevar = "varb", times = as.factor(vars), v.names = "resp")
```

We also have to zero out some non-bottom-layer elements of `z1` and `z2`, just like with the "real" (simulated) data

```
> barlayer <- substr(as.character(rebar$varb), 1, 5)
> rebar$z1[barlayer != "ngerm"] <- 0
> rebar$z2[barlayer != "ngerm"] <- 0
```

and we are finally ready to fit data and make a prediction. We must redo `out6` because we threw it away. (The following can be the example for these data when added to the `aster` package.)

```
> out6 <- aster(resp ~ varb + 0 + z1 + z2 + I(z1^2) +
+     I(z1 * z2) + I(z2^2), pred, fam, varb, id, root,
+     data = redata)
```

Then we predict at the new points.

```
> pbar <- predict(out6, newdata = rebar, varvar = varb,
+     idvar = id, root = root)
> pbar <- matrix(pbar, nrow = nrow(bar))
> pbar <- pbar[, grep("germ", vars)]
> zz <- apply(pbar, 1, sum)
> zz <- matrix(zz, nx, ny)
```

While we are at it, figure out the point where fitness is maximized

```
> Afoo <- matrix(NA, 2, 2)
> Afoo[1, 1] <- out6$coef["I(z1^2)"]
> Afoo[2, 2] <- out6$coef["I(z2^2)"]
> Afoo[1, 2] <- out6$coef["I(z1 * z2)"]/2
> Afoo[2, 1] <- out6$coef["I(z1 * z2)"]/2
> bfoo <- rep(NA, 2)
> bfoo[1] <- out6$coef["z1"]
> bfoo[2] <- out6$coef["z2"]
> cfoo <- solve(-2 * Afoo, bfoo)
> cfoo

[1] 3.335738 1.626314
```

The R statements make Figure 2 (page 18)

```
> par(mar = c(2, 2, 1, 1) + 0.1)
> plot(ladata$z1, ladata$z2, xlab = "", ylab = "",
+     pch = 20, axes = FALSE)
> title(xlab = "z1", line = 1)
> title(ylab = "z2", line = 1)
> box()
> contour(xfoo, yfoo, zz, add = TRUE, col = "blue",
+     labcex = 1, lwd = 2)
> points(cfoo[1], cfoo[2], col = "blue", pch = 19)
```

4.2 Compare with Simulation Truth Fitness Landscape

We also want to compare with the simulation truth.

```
> out6.true <- out6
> out6.true$coefficients <- beta.true
> pbar.true <- predict(out6.true, newdata = rebar,
+     varvar = varb, idvar = id, root = root)
> pbar.true <- matrix(pbar.true, nrow = nrow(bar))
> pbar.true <- pbar.true[, grep("germ", vars)]
> zz.true <- apply(pbar.true, 1, sum)
> zz.true <- matrix(zz.true, nx, ny)
```

While we are at it, figure out the point where fitness is maximized

```
> Abar <- matrix(NA, 2, 2)
> Abar[1, 1] <- beta.true["I(z1^2)"]
> Abar[2, 2] <- beta.true["I(z2^2)"]
> Abar[1, 2] <- beta.true["I(z1 * z2)"]/2
> Abar[2, 1] <- beta.true["I(z1 * z2)"]/2
```

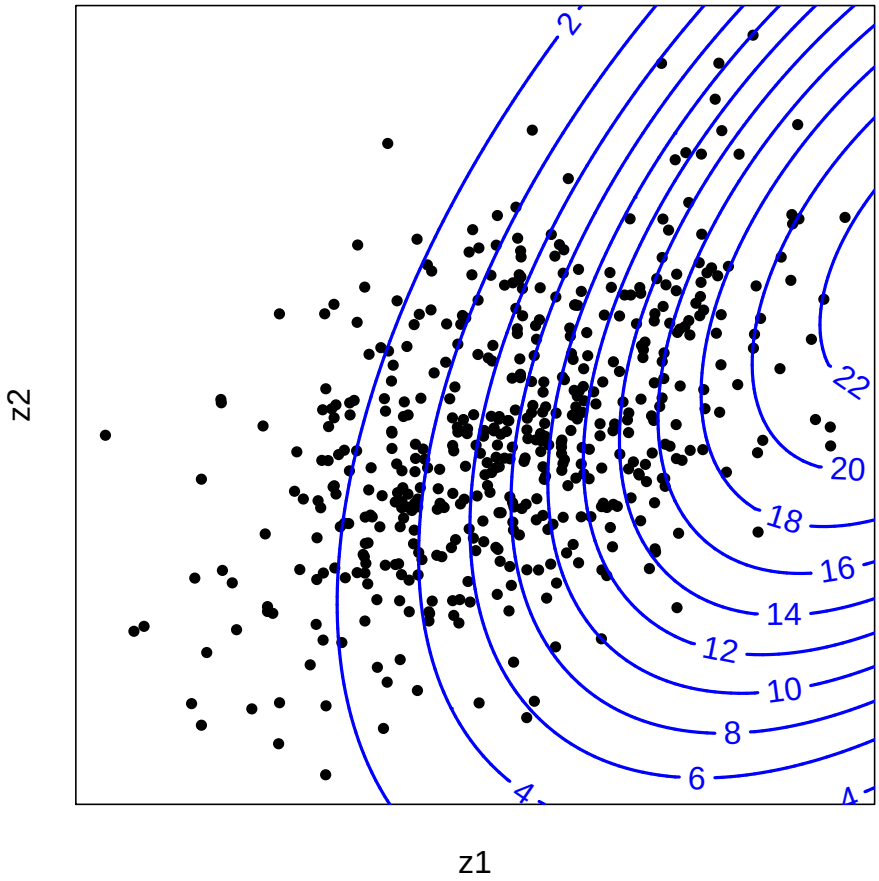


Figure 2: Scatterplot of `z1` versus `z2` with contours of the fitness landscape as estimated by the aster model (blue).

```
> bbar <- rep(NA, 2)
> bbar[1] <- beta.true["z1"]
> bbar[2] <- beta.true["z2"]
> cbar <- solve(-2 * Abar, bbar)
> cbar

[1] 2.0 0.5
```

The R statements make Figure 3 (page 20)

```
> par(mar = c(2, 2, 1, 1) + 0.1)
> plot(ladata$z1, ladata$z2, xlab = "", ylab = "",
+     pch = 20, axes = FALSE)
> title(xlab = "z1", line = 1)
> title(ylab = "z2", line = 1)
> box()
> contour(xfoo, yfoo, zz.true, add = TRUE, col = "green3",
+     labcex = 1, lwd = 2)
> points(cbar[1], cbar[2], col = "green3", pch = 19)
```

4.3 Compare with Lande-Arnold Estimate of Fitness Landscape

We also want to compare with the Lande-Arnold estimate, which is the best linear unbiased estimator of the best quadratic approximation to the fitness landscape. (We need to refit the Lande-Arnold estimate because we threw it away.)

```
> lout <- lm(y ~ z1 + z2 + I(z1^2) + I(z1 * z2) + I(z2^2),
+     data = ladata)
> zz.la <- predict(lout, newdata = bar)
> zz.la <- matrix(zz.la, nx, ny)
```

While we are at it, figure out the point where fitness is maximized

```
> beta.la <- lout$coefficients
> Alob <- matrix(NA, 2, 2)
> Alob[1, 1] <- beta.la["I(z1^2)"]
> Alob[2, 2] <- beta.la["I(z2^2)"]
> Alob[1, 2] <- beta.la["I(z1 * z2)"]/2
> Alob[2, 1] <- beta.la["I(z1 * z2)"]/2
> blob <- rep(NA, 2)
> blob[1] <- beta.la["z1"]
> blob[2] <- beta.la["z2"]
> clob <- solve(-2 * Alob, blob)
> clob

[1] -3.169010 -1.300450
```

The R statements make Figure 4 (page 21)

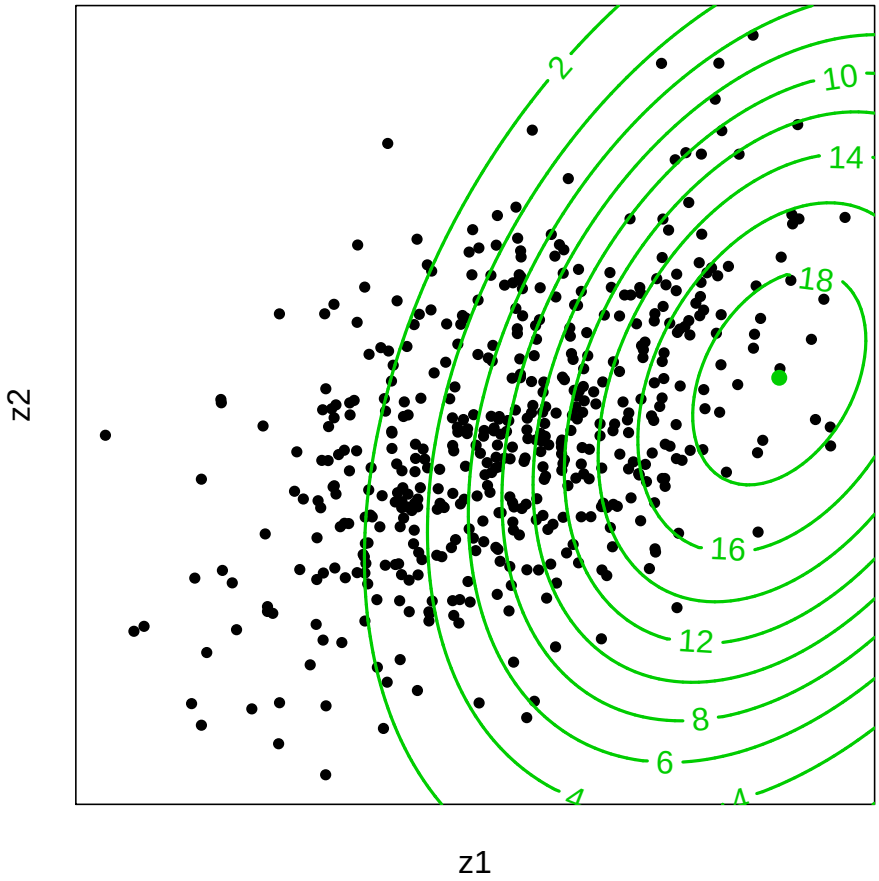


Figure 3: Scatterplot of `z1` versus `z2` with contours of the simulation truth fitness landscape (green).

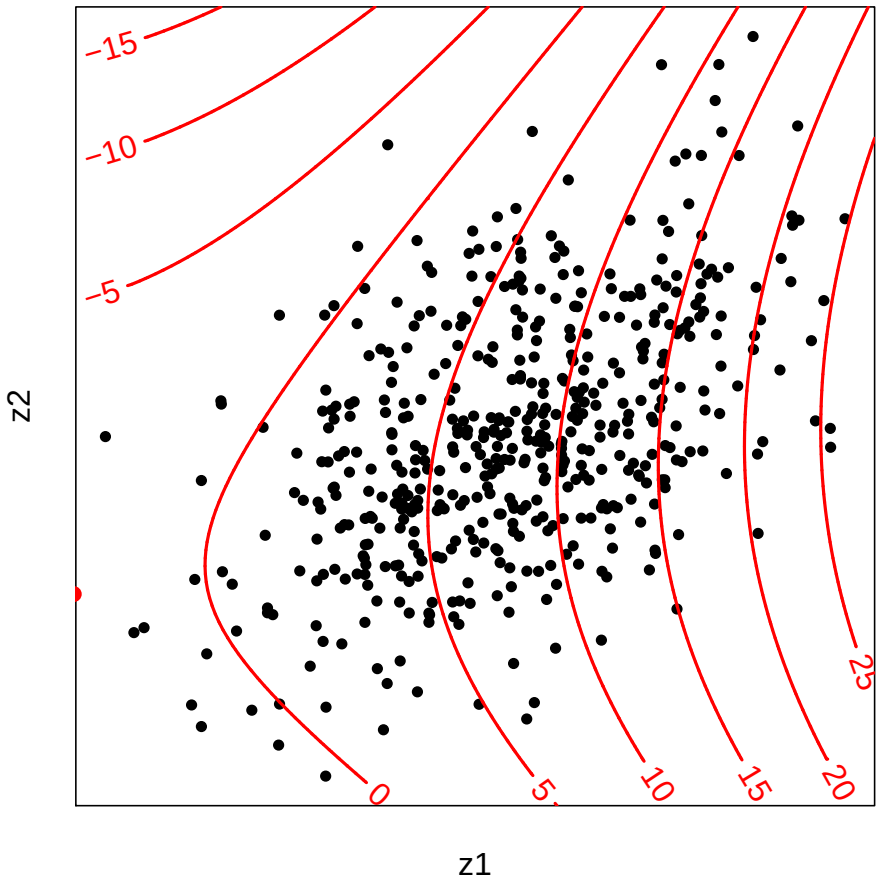


Figure 4: Scatterplot of `z1` versus `z2` with contours of the best quadratic approximation of the fitness landscape as estimated by the Lande-Arnold method (red).

```
> par(mar = c(2, 2, 1, 1) + 0.1)
> plot(ladata$z1, ladata$z2, xlab = "", ylab = "",
+     pch = 20, axes = FALSE)
> title(xlab = "z1", line = 1)
> title(ylab = "z2", line = 1)
> box()
> contour(xfoo, yfoo, zz.la, add = TRUE, col = "red",
+     labcex = 1, lwd = 2)
> points(clob[1], clob[2], col = "red", pch = 19)
```

A A Generalization of the Argument in Section 3.2

The argument in Section 3.2 can be further generalized. Rewrite (2a) and (2b) as

$$\varphi_j = \begin{cases} a_j(\mathbf{x}) + b_j(\mathbf{x}) q(\mathbf{z}), & j \in G \\ a_j(\mathbf{x}), & j \notin G \end{cases}$$

where $b_j(\mathbf{x})$ are arbitrary functions of the "other" covariates. Follow the same argument and the conclusion (5) becomes

$$q(\mathbf{z}) \geq q(\mathbf{z}') \quad \text{implies} \quad \sum_{j \in G} b_j(\mathbf{x}) \mu_j(\mathbf{x}, \mathbf{z}) \geq \sum_{j \in G} b_j(\mathbf{x}) \mu_j(\mathbf{x}, \mathbf{z}').$$

It is important that the comparison is between individuals with different values $\mathbf{z}$ and $\mathbf{z}'$ of phenotypic covariates but the same value $\mathbf{x}$ of other covariates.

In short, fitness, now defined as an arbitrary linear combination of unconditional expectations of nodes, is still a monotone function of $q(\mathbf{z})$. This would be useful, for example, if one wanted to take account of population growth rate λ when fitness would be defined as $\sum_{j \in G} \lambda^{-t_j} \mu_j(\mathbf{x}, \mathbf{z})$, where t_j is the time at which the j-th node is observed.

B Monotonicity in Conditional Aster Models

So-called conditional aster models may be of some use in some situations, but they are not useful in this context. They too have a monotonicity property, but between the linear predictor and the conditional mean vector $\boldsymbol{\xi}$ having components defined by

$$E(y_j \mid y_{p(j)}) = y_{p(j)} \xi_j$$

where $p(j)$ denotes the predecessor of j, all arrows in the graph going

$$y_{p(j)} \longrightarrow y_j$$

for various values of j.

Conditional aster models actually satisfy a much stronger monotonicity property than unconditional aster models, because ξ_j is a function of θ_j only (does not depend on the other components of $\boldsymbol{\theta}$) and the map $\theta_j \mapsto \xi_j$ is strictly increasing. This does imply

$$(\boldsymbol{\xi} - \boldsymbol{\xi}')^T (\boldsymbol{\theta} - \boldsymbol{\theta}') > 0, \tag{6}$$

when $\boldsymbol{\theta} \neq \boldsymbol{\theta}'$, which is just like (4) except that here conditional means replace unconditional means and the linear predictor vectors are those for the conditional model.

However, this property is of no use in modeling fitness. The conditional means do determine the unconditional means and hence determine expected fitness, but the relationship is nonlinear and hence (6) cannot be used to argue analogously to the argument proceeding from (1). Hence when a conditional model is used there is no monotone relationship between expected fitness and the function $q(\mathbf{z})$ used on the linear predictor scale. Thus, in the context of estimating fitness landscapes, conditional aster models are of no use whatsoever.

References

Barndorff-Nielsen, O. E. (1978). *Information and Exponential Families*. Chichester: John Wiley.

Brown, L. D. (1986). *Fundamentals of Statistical Exponential Families: with Applications in Statistical Decision Theory*. Hayward, CA: Institute of Mathematical Statistics.

Geyer, C. J., Wagenius, S. and Shaw, R. G. (2007). Aster models for life history analysis. *Biometrika*, **94**, 415–426.

Lande, R. and Arnold, S. J. (1983). The measurement of selection on correlated characters. *Evolution*, **37**, 1210–1226.

R Development Core Team (2008). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. `http://www.R-project.org`.

Rockafellar, R. T. and Wets, R. J.-B. (2004). *Variational Analysis*, corr. 2nd printing. Berlin: Springer-Verlag.

Shaw, R. G., Geyer, C. J., Wagenius, S., Hangelbroek, H. H., and Etterson, J. R. (2008). Unifying life history analysis for inference of fitness and population growth. *American Naturalist*, in press. `http://www.stat.umn.edu/geyer/aster/`

Shaw, R. G., Geyer, C. J., Wagenius, S., Hangelbroek, H. H., and Etterson, J. R. (2007a). Supporting data analysis for "Unifying life history analysis for inference of fitness and population growth". University of Minnesota School of Statistics Technical Report No. 658 `http://www.stat.umn.edu/geyer/aster/`

Shaw, R. G., Geyer, C. J., Wagenius, S., Hangelbroek, H. H., and Etterson, J. R. (2007b). More Supporting data analysis for "Unifying life history analysis for inference of fitness and population growth". University of Minnesota School of Statistics Technical Report No. 661 `http://www.stat.umn.edu/geyer/aster/`