Aster Models for Life History Analysis

By
Charles J. Geyer, Stuart Wagenius, and Ruth G. Shaw

Technical Report No. 644

School of Statistics
University of Minnesota

September 5, 2005

Abstract

We present a new class of statistical models designed for life history analysis of plants and animals. They allow joint analysis of data on survival and reproduction over multiple years, allow for variables having different statistical distributions, and correctly account for the dependence of variables on earlier variables (for example, that a dead individual stays dead and cannot reproduce). We illustrate their utility with an analysis of data taken from an experimental study of *Echinacea angustifolia* sampled from remnant prarie populations in western Minnesota. Statistically, they are graphical models with some resemblance to generalized linear models and survival analysis. They have directed acyclic graphs with nodes having no more than one parent. The conditional distribution of each node given the parent is a one-parameter exponential family with the parent variable the sample size. The model may be heterogeneous, each node having a different exponential family. We show that the joint distribution is a flat exponential family and derive its canonical parameters, Fisher information, and other properties. These models are implemented in an R package 'aster' available from CRAN.

Keywords: Conditional Exponential Family; Curved Exponential Family; Flat Exponential Family; Generalized Linear Model; Graphical Model; Maximum Likelihood; Nuisance Variable.

This technical report consists of a draft paper about aster models supplemented by 5 appendicies on technical subjects.

- Appendix A (p. 19) gives details about "prediction" (what the `predict.aster` function does), although the technical details are about change-of-parameter formulas and their derivatives.
- Appendix B (p. 24) gives details about the one-parameter exponential families currently available as conditional distribution of variables given their parent variable in the `aster` package.
- Appendix C (p. 29) gives details about simulating a Poisson distribution conditional on being nonzero (the only non-trivial issue being getting efficient simulation for unconditional means close to zero).
- Appendix D (p. 31) gives details of the data analysis that is briefly described in the draft paper.
- Appendix E (p. 58) shows that steepness of conditional exponential families implies stepness of corresponding unconditional families, and the full unconditional family gets no new parameter points (that do not correspond to conditional parameter points).

Chapter 1

Draft Paper

1.1 Introduction

This article introduces a class of statistical models that we call *aster models*. They were invented for life history analysis (LHA) of plants and animals, and are best introduced by example. Archetypal data for these models are about perennial plants censused at various times. For each individual planted, we record whether it is still alive, whether it has flowered, and how many flowers it has. These data are complicated, especially when recorded for several years, but when considered conditionally, simple models may suffice. We consider mortality status (dead or alive) to be Bernoulli given the preceding mortality status. Similarly for flowering status given mortality status. Given flowering, the number of flowers may have a Poisson distribution conditioned on being nonzero. Figure 1.1 gives a graphical representation of this kind of data.

The most important feature of these models is the simplest one. A simultaneous analysis that models the joint distribution of all the variables in a life history analysis can answer questions that cannot be addressed when one does a separate analysis of each variable (conditional on the values of the others).

Although we called the Figure 1.1 data 'archetypal', there is nothing special about the three particular measurements in that example. We could add a fourth variable, seed number, modelled conditional on flower number. And so forth. Nor is there anything special about plants or even living organisms. This methodology applies to any similar conditional modelling.

Our models have some resemblance to discrete time Cox regression (Cox, 1972; Breslow, 1972, 1974) when the graph is linear and all the responses are Bernoulli, but does not have exactly the same likelihood (so those aster models are competitors rather than generalizations of Cox models). Of course, if the graph is a general forest or the responses are not all Bernoulli, then the

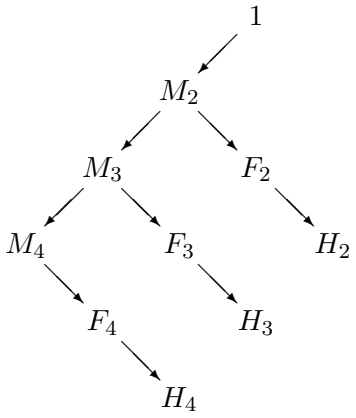

Figure 1.1: Graph for Archetypal Aster Data. Arrows go from nodes to their successors. Nodes are labelled by their associated variables. The only founder node is associated with the constant variable 1. M_j is the mortality status in year $2000+j$. F_j is the flowering status in year $2000+j$. H_j is the flower head count in year $2000 + j$. The M_j and F_j are Bernoulli conditional on their predecessor variables being one (and zero otherwise). The H_j are Poisson conditioned on being nonzero conditional on their predecessor variables being one (and zero otherwise).

resemblance to survival analysis is faint.

Our models also have some resemblance to generalized linear models (GLM; McCullagh & Nelder, 1989) when the graph has only one node, but we do not allow arbitrary link functions or quasi-likelihood, using only canonical exponential family parameters as linear predictors (so those aster models are specializations of GLM). Of course, if the graph is a general forest and especially if the responses do not all have the same exponential family, then the resemblance to GLM is faint.

Our models are graphical but of the simplest kind, associated with directed acyclic graphs (Lauritzen, 1996, Section 3.2.2), in which the joint density is a product of conditionals as in equation (1.1) below. Readers need no knowledge of graphical model theory to understand aster models.

The main innovative aspect of aster models is the interplay between two canonical parameterizations described in Sections 1.1.2 and 1.1.3 below. One, the *conditional canonical parameterization*, arises when each distribution in the product of conditionals (1.1) belongs to an exponential family and we use the canonical parameterization for each. These are the conditional exponential family (CEF) aster models. The other, the *unconditional canonical parameterization*, arises from observing that the joint model is a full flat exponential family (Barndorff-Nielsen, 1978, Chapter 8) and using the canonical

parameters for that family, defined by equation (1.5) below. These are the flat exponential family (FEF) aster models. And we see that CEF could also stand for 'curved exponential family' since, considered unconditionally, that is what they are.

We named our models after flowers because the name is short and much nicer than *forest graph exponential family conditional or unconditional canonical statistic models* or any other descriptive name we could think of. The particular name was chosen for the organism in our example, the purple coneflower *Echinacea angustifolia*, which is in the family *Asteraceae* of which *Aster* is the type genus. They also come with a neat motto: *per aspera cum astris*, a take-off on the motto of the sunflower state (sunflowers are also *Asteraceae*).

1.1.1 Forest Graph Models

We describe conditional dependence structure graphically, as in Figure 1.1. Each node in the graph is associated with a variable (in Figure 1.1 we labelled the nodes by their variable names). Aster models have *forest graphs:* directed acyclic graphs in which each node has at most one predecessor and there are no isolated nodes. Each edge (arrow) in the graph represents a conditional distribution of the variable at the arrow head given the variable at the arrow tail (its predecessor). Thus the joint distribution of all the variables is the product of conditionals, equation (1.1) below, one conditional for each arrow in the graph.

When m is the *predecessor* of j, we also say j is a *successor* of m. Nodes that have no predecessors are called *root* nodes. Nodes that have no successors are called *leaf* nodes. We also use an alternative terminology saying *parent* instead of predecessor, *child* instead of successor, *founder* instead of root, and *childless* instead of leaf.

We divide the nodes of the graph into disjoint sets F and J, the founder and non-founder nodes, respectively, and introduce a function $p : J \to J \cup F$ that maps nodes j to their predecessors $p(j)$. The diagram of p is just like the graph of the graphical model (like Figure 1.1) except that all the arrows are reversed. With this notation, we can write the joint distribution of all the variables as the product of conditionals

$$\prod_{j \in J} \Pr\{X_j | X_{p(j)}\}. \tag{1.1}$$

The fact that the variables X_j associated with root nodes do not appear 'in front of the bar' in (1.1) means these variables are nonrandom (or at least are treated as nonrandom in that we are conditioning on them and not modelling their distributions).

It will simplify notation in what follows if we define the partial order relations determined by the graph. The predecessor function p determines

a relation $\{ \big(j, p(j)\big) : j \in J \}$. We denote its transient closure by $\prec$, so $j \prec m$ means $m = p(j)$, or $m = p(p(j))$, or .... And we denote its transitive reflexive closure by $\preceq$, so $j \preceq m$ means $j \prec m$ or $j = m$. We call $\prec$ the *ancestor relation*, and read $j \prec m$ as m is an *ancestor* of j. We call $\preceq$ the *ancestor-or-self relation*.

It will also simplify notation if we define the function $f : J \to F$ that maps a node j to its (unique) founder ancestor $f(j)$.

1.1.2 Conditional Exponential Families

We take each of the conditional distributions in (1.1) to be a one-parameter exponential family (perhaps a different such family for each j) with X_j the canonical statistic and the dependence on $X_{p(j)}$ being that X_j is the sum of $X_{p(j)}$ i. i. d. (independent and identically distributed) random variables. In order that this make sense the variables X_j for $j \in p(J)$, where $p(J)$ denotes the range of the predecessor function (the set of non-leaf nodes), must be nonnegative-integer-valued. No such restriction is placed on X_j for $j \notin p(J)$ (for leaf nodes).

Then the log likelihood for the whole family has the form

$$\sum_{j \in J} X_j \theta_j - X_{p(j)} \psi_j(\theta_j) \tag{1.2}$$

where θ_j is the canonical parameter for the j-th conditional family and ψ_j is the so-called *cumulant function* for that family (Barndorff-Nielsen, 1978, pp. 105, 139, and 150) that satisfies

$$E_{\theta_j}\{X_j | X_{p(j)}\} = \psi'(\theta_j) \tag{1.3a}$$

$$\operatorname{var}_{\theta_j}\{X_j | X_{p(j)}\} = \psi''(\theta_j) \tag{1.3b}$$

(for examples of cumulant functions, see Appendix B).

When we have *independent and identically modelled* (i. i. m.) observations (independent and come from the same model but may have different parameter values) of data from such families, the log likelihood becomes

$$\sum_{i \in I} \sum_{j \in J} X_{ij} \theta_{ij} - X_{ip(j)} \psi_j(\theta_{ij}) \tag{1.4}$$

where I is a finite set (indexing individuals). Note that ψ_j is not ψ_{ij} so each node of the graph has only one exponential family associated with it, an idea required by the rows of the data matrix being i. i. m.

1.1.3 Unconditional Exponential Families

Introducing the notation

$$S(j) = \{\, m \in J : j = p(m) \,\}$$

for the set of successors of j and collecting terms with the same X_j in (1.2), we get

$$\sum_{j \in J} X_j \left[\theta_j - \sum_{m \in S(j)} \psi_m(\theta_m) \right] - \sum_{j \in S(F)} X_{p(j)} \psi_j(\theta_j)$$

where $S(F)$ is the set of children of founders.

Now we see that by introducing new parameters

$$\varphi_j = \theta_j - \sum_{m \in S(j)} \psi_m(\theta_m), \qquad j \in J \tag{1.5}$$

we have an unconditional exponential family with canonical statistics X_j and canonical parameters φ_j for $j \in J$.

Collecting the canonical statistics and parameters into vectors $\mathbf{X}$ and $\boldsymbol{\varphi}$ we can write the log likelihood of this unconditional family as

$$l(\boldsymbol{\varphi}) = \langle \mathbf{X}, \boldsymbol{\varphi} \rangle - \psi(\boldsymbol{\varphi}) \tag{1.6a}$$

where $\langle \mathbf{X}, \boldsymbol{\varphi} \rangle$ denotes the inner product $\sum_i X_j \varphi_j$ and where the cumulant function of this family is

$$\psi(\boldsymbol{\varphi}) = \sum_{j \in S(F)} X_{p(j)} \psi_j(\theta_j) \tag{1.6b}$$

Note that all of the $X_{p(j)}$ in (1.6b) are at founder nodes and are constant so ψ is a deterministic (not random) function, and also note that, although it is not obvious that the right hand side of (1.6b) is (as the left hand side says) a function of $\boldsymbol{\varphi}$, this must be true by the logic of exponential families (Barndorff-Nielsen, 1978, pp. 105 ff.).

A little more thought shows that the system of equations (1.5) can be solved for the θ_j in terms of the φ_j in one pass through the equations in any order that finds θ_j for children before parents. Thus (1.5) determines an invertible change of parameter.

The analog of (1.6a) and (1.6b) when we have i. i. m. replication requires we interpret matrices as vectors, elements of the finite-dimensional vector space $\mathbb{R}^{I \times J}$. Equation (1.6a) remains, although we reinterpret its inner product as $\sum_{ij} X_{ij} \varphi_{ij}$, and (1.6b) gets additional indices

$$\psi(\boldsymbol{\varphi}) = \sum_{i \in I} \sum_{j \in S(F)} X_{ip(j)} \psi_j(\theta_{ij}) \tag{1.6c}$$

as does (1.5)

$$\varphi_{ij} = \theta_{ij} - \sum_{m \in S(j)} \psi_m(\theta_{im}), \qquad i \in I,\ j \in J. \tag{1.6d}$$

1.1.4 Sufficient Statistics

As is well known (Barndorff-Nielsen, 1978, p. 111) the canonical statistic of an exponential family is minimal sufficient. Since we have both conditional and unconditional families in play, we stress that this well-known result is about *unconditional* families.

One of the desirable aspects of exponential family GLM defined by reparameterization of the form

$$\boldsymbol{\varphi} = \mathbf{M}\boldsymbol{\beta}, \tag{1.7}$$

where $\mathbf{M}$ is a known matrix (the *model matrix*), is that from the identity $\langle \mathbf{X}, \mathbf{M}\boldsymbol{\beta} \rangle = \langle \mathbf{M}^T\mathbf{X}, \boldsymbol{\beta} \rangle$ we see that the result is a new exponential family with canonical statistic $\mathbf{M}^T\mathbf{X}$ and canonical parameter $\boldsymbol{\beta}$. The dimension of this new family will be the dimension of $\boldsymbol{\beta}$, if $\mathbf{M}$ has full rank.

If $\mathbf{X}$ is a matrix interpreted as a vector, then $\mathbf{M}$ is an array representing a linear operator $\mathbb{R}^K \to \mathbb{R}^{I \times J}$. So (1.7) means

$$\varphi_{ij} = \sum_{k \in K} m_{ijk}\beta_k, \tag{1.8}$$

and $\mathbf{Y} = \mathbf{M}^T\mathbf{X}$ means

$$Y_k = \sum_{i \in I} \sum_{j \in J} X_{ij} m_{ijk}. \tag{1.9}$$

This 'dimension reduction' to sufficient statistics Y_k does not occur when the conditional parameters $\boldsymbol{\theta}$ are 'generalized linear modelled' in a similar way, and this suggests that generalized linear models for the unconditional parameterization may be scientifically more interesting despite their more complicated structure.

1.1.5 Mean Value Parameters

Canonical parameters, although interesting both theoretically and practically because they are the GLM 'linear predictors', have no real-world interpretation. The parameters that do are the *mean value parameters*.

The conditional mean value parameters are the conditional means

$$\xi_{ij} = E_{\theta_{ij}}\{X_{ij} | X_{ip(j)}\} = X_{ip(j)}\psi_j'(\theta_{ij}). \tag{1.10}$$

Strictly speaking, the ξ_{ij} are not *parameters* because they contain random data $X_{ip(j)}$. Nevertheless, they do play the role of mean value parameters

when one is thinking conditionally, treating $X_{ip(j)}$ as constant. Standard exponential family theory (Barndorff-Nielsen, 1978, p. 121) says that ψ_j' is an invertible change of parameter.

Similarly, the unconditional mean value parameters are the unconditional means

$$\boldsymbol{\tau} = E_{\boldsymbol{\varphi}}\{\mathbf{X}\} = \nabla\psi(\boldsymbol{\varphi}) \tag{1.11}$$

(∇ denotes the vector of partial derivatives). Again, standard theory says that $\nabla\psi : \boldsymbol{\varphi} \mapsto \boldsymbol{\tau}$ is an invertible change of parameter. The unconditional expectation in (1.11) can be calculated using the iterated expectation theorem (from probability theory) or the chain rule (from calculus)

$$E_{\boldsymbol{\varphi}}\{X_{ij}\} = X_{if(j)} \prod_{\substack{m \in J \\ j \preceq m \prec f(j)}} \psi_m'(\theta_{im}), \tag{1.12}$$

where the θ_{ij} are determined from $\boldsymbol{\varphi}$ by solving (1.5).

1.2 Theory

1.2.1 Conditional Models

The score for conditional canonical parameters is particularly simple

$$\frac{\partial l(\boldsymbol{\theta})}{\partial \theta_{ij}} = X_{ij} - X_{ip(j)}\psi_j'(\theta_{ij})$$

and, if these parameters are modelled linearly in terms of other parameters, equation (1.8) with φ replaced by θ, then we have

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \beta_k} = \sum_{i \in I} \sum_{j \in J} \Big[X_{ij} - X_{ip(j)}\psi_j'(\theta_{ij}) \Big] m_{ijk} \tag{1.13}$$

The observed Fisher information for $\boldsymbol{\theta}$ is diagonal with

$$-\frac{\partial^2 l(\boldsymbol{\theta})}{\partial \theta_{ij}^2} = X_{ip(j)}\psi_j''(\theta_{ij})$$

and zero for mixed second derivatives. And from this we see the observed Fisher information for $\boldsymbol{\beta}$ is

$$-\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \beta_k \partial \beta_{k'}} = \sum_{i \in I} \sum_{j \in J} X_{ip(j)}\psi_j''(\theta_{ij}) m_{ijk} m_{ijk'}. \tag{1.14a}$$

The expected Fisher information is just the (unconditional) expectation of the observed Fisher information. So it is

$$-E_{\boldsymbol{\beta}} \left\{ \frac{\partial^2 l(\boldsymbol{\beta})}{\partial \beta_k \partial \beta_{k'}} \right\} = \sum_{i \in I} \sum_{j \in J} E_{\boldsymbol{\beta}}\{X_{ip(j)}\}\psi_j''(\theta_{ij}) m_{ijk} m_{ijk'} \tag{1.14b}$$

the unconditional expectation on the right hand side being evaluated by using (1.12).

1.2.2 Unconditional Models

The score for unconditional canonical parameters is, as in every (unconditional) exponential family, 'observed minus expected'

$$\frac{\partial l(\boldsymbol{\varphi})}{\partial \varphi_{ij}} = X_{ij} - E_{\boldsymbol{\varphi}}\{X_{ij}\}$$

the unconditional expectation on the right hand side being evaluated by using (1.12), and, if these parameters are modelled linearly in terms of other parameters (1.8), then we have

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \beta_k} = \sum_{i \in I} \sum_{j \in J} \Big[X_{ij} - E_{\boldsymbol{\beta}}\{X_{ij}\} \Big] m_{ijk}. \tag{1.15}$$

Second derivatives with respect to the (unconditional) canonical parameters of an exponential family are nonrandom. Hence there is no difference between observed and expected Fisher information. The information for $\boldsymbol{\varphi}$ is given by either of the expressions $\nabla^2 \psi(\boldsymbol{\varphi})$ or $\operatorname{var}_{\boldsymbol{\varphi}}(\mathbf{X})$, the former being the matrix of second partial derivatives of the cumulant function (1.6c) and the latter being the variance-covariance matrix of the random vector with components X_{ij} (Barndorff-Nielsen, 1978, p. 150). We choose to work on the latter. The iterated variance formula gives

$$\begin{aligned} \operatorname{var}_{\boldsymbol{\varphi}}\{X_{ij}\} &= E_{\boldsymbol{\varphi}}\big[\operatorname{var}_{\boldsymbol{\varphi}}\{X_{ij} | X_{ip(j)}\}\big] + \operatorname{var}_{\boldsymbol{\varphi}}\big[E_{\boldsymbol{\varphi}}\{X_{ij} | X_{ip(j)}\}\big] \\ &= \psi_j''(\theta_{ij}) E_{\boldsymbol{\varphi}}\{X_{ip(j)}\} + \psi_j'(\theta_{ij})^2 \operatorname{var}_{\boldsymbol{\varphi}}\{X_{ip(j)}\} \end{aligned} \tag{1.16a}$$

Since we already have (1.12) to calculate expectations, (1.16a) allows calculation of all variances (diagonal elements of the Fisher information matrix) recursively. Now we work on the covariance of X_{ij} and $X_{ij'}$ with $j \neq j'$, in which case we may assume without loss of generality that $j' \not\preceq j$ so

$$\operatorname{cov}_{\boldsymbol{\varphi}}\{X_{ij}, X_{ij'} | X_{ip(j)}\} = 0 = \operatorname{cov}_{\boldsymbol{\varphi}}\{X_{ij}, X_{ij'} | X_{ip(j)}, X_{ij'}\}$$

because X_{ij} is conditionally independent given $X_{ip(j)}$ of all variables except its descendants, which do not include $X_{ij'}$. Then the iterated covariance formula gives

$$\begin{aligned} \operatorname{cov}_{\boldsymbol{\varphi}}\{X_{ij}, X_{ij'}\} &= \operatorname{cov}_{\boldsymbol{\varphi}}\big\{E_{\boldsymbol{\varphi}}(X_{ij} | X_{ip(j)}), X_{ij'}\big\} \\ &= \psi_j'(\theta_{ij}) \operatorname{cov}_{\boldsymbol{\varphi}}\big\{X_{ip(j)}, X_{ij'}\big\} \end{aligned} \tag{1.16b}$$

and this allows us to determine all of the covariances recursively. The calculation of the information for $\boldsymbol{\beta}$ from that for $\boldsymbol{\varphi}$ is analogous to that in the preceding section and is omitted.

1.2.3 Prediction

By 'prediction', we mean no more than evaluation of a function of estimated parameters, one job the `predict` function in R does for linear or generalized linear model fits. In aster models we have five different parameterizations of interest ($\boldsymbol{\beta}$, $\boldsymbol{\theta}$, $\boldsymbol{\varphi}$, $\boldsymbol{\xi}$, and $\boldsymbol{\tau}$). The Fisher information for $\boldsymbol{\beta}$, already described, tells us what we need to know about predicting $\boldsymbol{\beta}$. So this section is about 'predicting' the remaining four.

One often predicts for new individuals having different covariate values from the observed individuals. Then the model matrix $\mathbf{M}$ used for the prediction is different from that used for fitting. So in this section $\mathbf{M}$ denotes this possibly different model matrix. However when we use Fisher information $I(\boldsymbol{\beta})$, either observed or expected (Sections 1.2.1 and 1.2.2), this is based on the original model used to obtain parameter estimates $\hat{\boldsymbol{\beta}}$ and hence on the original model matrix.

Let $\boldsymbol{\eta}$ be the linear predictor ($\boldsymbol{\eta} = \boldsymbol{\theta}$ for conditional models and $\boldsymbol{\eta} = \boldsymbol{\varphi}$ for unconditional models). Let $\boldsymbol{\zeta}$ be any one of $\boldsymbol{\theta}$, $\boldsymbol{\varphi}$, $\boldsymbol{\xi}$, or $\boldsymbol{\tau}$. Let $f_{\boldsymbol{\eta},\boldsymbol{\zeta}}$ denote the map $\boldsymbol{\eta} \to \boldsymbol{\zeta}$, and suppose we wish to predict the parameter

$$g(\boldsymbol{\beta}) = h(\boldsymbol{\zeta}) = h\big(f_{\boldsymbol{\eta},\boldsymbol{\zeta}}(\mathbf{M}\boldsymbol{\beta})\big) \tag{1.17}$$

Then by the chain rule (1.17) has derivative

$$\nabla g(\boldsymbol{\beta}) = \nabla h(\boldsymbol{\zeta}) \circ \nabla f_{\boldsymbol{\eta},\boldsymbol{\zeta}}(\boldsymbol{\eta}) \circ \mathbf{M} \tag{1.18}$$

(where here $\mathbf{M}$ denotes the linear operator associated with the model matrix) and by the 'usual' asymptotics of maximum likelihood and the delta method, the asymptotic distribution of the prediction $h(\hat{\boldsymbol{\zeta}}) = g(\hat{\boldsymbol{\beta}})$ is

$$\text{Normal}\big(g(\boldsymbol{\beta}), [\nabla g(\hat{\boldsymbol{\beta}})] I(\hat{\boldsymbol{\beta}})^{-1} [\nabla g(\hat{\boldsymbol{\beta}})]^T\big)$$

where $\nabla g(\hat{\boldsymbol{\beta}})$ is given by (1.18) with $\hat{\boldsymbol{\eta}} = \mathbf{M}\hat{\boldsymbol{\beta}}$ plugged in for $\boldsymbol{\eta}$ and $\hat{\boldsymbol{\zeta}} = f_{\boldsymbol{\eta},\boldsymbol{\zeta}}(\hat{\boldsymbol{\eta}})$ plugged in for $\boldsymbol{\zeta}$

The point of writing 'predictions' in this complicated form is to separate the parts of the specification, the functions h and ∇h and the model matrix $\mathbf{M}$, that are easy but change from application to application from the hard part $\nabla f_{\boldsymbol{\eta},\boldsymbol{\zeta}}$ that does not change and can be done by computer (see Appendix A for details).

For mean value parameters the user must also specify new 'response' data X_{ij} as well as new 'covariate' data in $\mathbf{M}$. This is one way that aster models differ from GLM. Unconditional mean value parameters $\boldsymbol{\tau}$ will depend only on the root elements X_{ij}, $j \in F$. Conditional mean value parameters $\boldsymbol{\xi}$ will depend on all elements X_{ij}, $j \in J \cup F$.

This makes conditional mean value parameters as described almost useless. Thus we envisage that users will usually specify $X_{ij} = 1$ for all hypothetical individuals i and all nodes j so that then $\xi_{ij} = \psi_j'(\theta_{ij})$ and hence are really parameters.

1.3 Software

We have developed an R (R Development Core Team, 2004) package `aster` that fits, tests, and predicts aster models. It uses the R formula mini-language, originally developed for GENSTAT and S (Wilkinson & Rogers, 1973; Chambers & Hastie, 1992) so that model fitting is much like that for linear or generalized linear models. Similarly, R functions `summary.aster`, `anova.aster`, and `predict.aster` provide regression coefficients with standard errors, z statistics, and p-values, likelihood ratio tests for model comparison, and the predictions with standard errors described in Section 1.2.3. It is available from CRAN (`http://www.cran.r-project.org`).

The current version of the `aster` package has two limitations of the general model described in this article.

- In predictions, the only h allowed in (1.17) are linear, that is, $h(\boldsymbol{\zeta}) = \mathbf{A}^T\boldsymbol{\zeta}$, for some matrix $\mathbf{A}$.
- In modelling, the only families allowed are
  - Bernoulli,
  - Poisson, and
  - Poisson conditioned on being nonzero.

The package contains 1500 lines of C source code, only 200 lines of which are R-specific, and 925 lines of R. It is provided under a permissive X11-like open source license so the 1300 lines of C that are the computational core could be reused in another implementation, even a proprietary one. Adding another one-parameter exponential family requires only implementation of the ψ, ψ', and ψ'' functions for the family.

1.4 Example

Data having the aster model structure shown in Figure 1.1 were collected on 570 individuals of *Echinacea angustifolia*. These plants were sampled as seeds from seven remnant populations that are surviving fragments of the tall-grass prairie that a century ago covered western Minnesota and other parts of the Great Plains of North America. The plants were experimentally randomized at the time of planting into a field within 6.5 km of all populations

Table 1.1: Tests for Model Comparison. The model formulae are given above and the analysis of deviance below (deviance is twice log likelihood).

```
1: resp ~ varb + level:(nsloc + ewloc)
2: resp ~ varb + level:(nsloc + ewloc) + hdct * pop - pop
3: resp ~ varb + level:(nsloc + ewloc) + hdct * pop
4: resp ~ varb + level:(nsloc + ewloc) + level * pop
```

Model Number	Model d. f.	Model Deviance	Test d. f.	Test Deviance	Test p-value
1	15	2728.72			
2	21	2712.54	6	16.18	0.013
3	27	2684.86	6	27.67	0.00011
4	33	2674.70	6	10.17	0.12

of origin. The data set contains three other predictor variables: `ewloc` and `nsloc` are spatial coordinates, east-west and north-south positions, of the individuals within the field and `pop` is the predictor variable of scientific interest. It is categorical giving the remnant population of origin.

The greatest advantage of aster models over previously available models for analyzing these data is that the aster model is a joint model for all the variables so a single analysis can account for all effects of predictors and relationships among responses. A secondary advantage is that unconditional aster models directly model marginal distributions by controlling their unconditional mean value parameters.

In order to use the R formula mini-language it is necessary to create some artificial variables. The variable `resp` is a vector comprising the nine response variables (the M_j, F_j, and H_j). The variable `varb` is categorical naming these response variables (so they can still be distinguished within `resp`). The variable `level` is categorical naming the type of response variable (M, F, or H). The variable `hdct` is an indicator variable indicating the H_j responses (a convenient shorthand for `level` $= H$).

We fit many models. See Appendix D for details. Scientific interest focuses on the model comparison shown in Table 1.1. All models contain the quantitative spatial effect `level:(nsloc + ewloc)`, which was chosen after extensive model comparison (details Appendix D). We explain here only the differences among the models.

The terms that differ are all categorical. Such categorical terms include dummy variables in the model (one for each category) and (in an unconditional aster model, which these are) require the maximum likelihood mean

value parameters for each category (summed over all individuals in the category) to match the observed values ('observed equals expected').

The models are nested, numbered in increasing order, so Model 1 has no terms not in the others. Model 2 adds `hdct * pop - pop`, which makes observed equal expected for *total flower head count* for each of the 7 populations. Model 3 adds `pop`, which makes observed equal expected for *total non-head-count* ($\sum_j M_j + F_j$) for each of the 7 populations. Model 4 adds `level * pop`, which splits total non-head-count into *total survival* ($\sum_j M_j$) and *total flowering* ($\sum_j F_j$) and makes observed equal expected for them for each of the 7 populations.

From purely statistical considerations, Model 3 is the best of these four nested models. Model 4 does not fit significantly better. Model 2 fits significantly worse. According to Model 1, the life history depends only on position within the field in which the plants are growing. Model 2 fits differences among populations in total headcount. From scientific considerations Model 3 is rather difficult to interpret, because it fits differences among populations in "non-head-count" ($M_j + F_j$), scoring each individual 0 for dead, 1 for alive without flowers, or 2 for alive with flowers. Model 4 fits differences among populations in each of survival, flowering, and headcount.

Model 2 is the model of primary interest for reasons we now explain. Evolutionary biologists are very interested in fitness. For our purposes here the fitness of an individual may be defined as its contribution over its lifespan in descendants to the next generation (see Beatty, 1992; Keller, 1992; Paul, 1992, for further discussion). Fitness is notoriously difficult to measure. One reason for this is that it is expressed over the lifespan, rather than instantaneously. For these data the most direct surrogate measure for fitness is *total flower head count* ($\sum_j H_j$). The currently available data represent a small fraction of this plant's lifespan. To obtain more complete measures of fitness, we are continuing these experiments and collecting these data for successive years.

Biologists call all our measured variables (the M_j, F_j, and H_j) *components of fitness*. Since M_j and F_j contribute to fitness (descendants) only through H_j, in an aster model the unconditional expectation of H_j (its mean value parameter) completely accounts for the contributions of M_j and F_j. Strictly speaking, this is not quite true, since we do not have H_j measured over the whole life span, so the last M_j contains the information that future reproduction is possible, but it becomes truer as more data are collected in future years. Moreover, we have no data about life span and do not wish to inject subjective opinion about future flower head count into the analysis.

The statistical point of this is that the M_j and F_j are only in the model to produce the correct stochastic structure. If we could directly model the marginal distribution of the H_j (but we can't), we would not need the other

variables. They are 'nuisance variables' that must be in the model but are of no interest in this particular analysis. (This is similar to division of parameters into 'interest' and 'nuisance'. In fact the mean value parameters for those variables are nuisance parameters.)

Statisticians seem not to have studied this kind of nuisance variable (there is an established usage meaning 'confounded with treatment effect' that is not what we mean). Model 3 is the best according to the likelihood ratio test, but does it fit the variables of interest better than Model 2? We do not know of an established methodology addressing this issue, so we propose looking at confidence intervals for the mean value parameters for total flower head count shown in Figure 1.2 (page 14). Although we have no formal test to propose, we claim it is obvious that Model 3 is no better than Model 2 at 'predicting' the best surrogate of expected fitness. We take this as justification for using Model 2 in scientific discussion and infer from it significant differences among the populations in flower head count and, thus, fitness.

As we said above, Model 3 is difficult to interpret scientifically. Model 4 is the next larger readily interpretable model. The fact that Model 4 fits significantly better than Model 2 ($P = 0.00016$) implies that there are differences among populations in mortality and flowering (the M_j and F_j) that may be of scientific interest even though they make no direct contribution to fitness (since Model 2 already fully accounts for their contributions through H_j).

Note that we would have gotten very different results had we used a conditional model (not shown, see Section D.3.2). The parameters of interest are *unconditional* expectations of total flower head count. This alone suggests an unconditional model. Furthermore, we see in (1.5) that unconditional aster models 'mix levels' passing information up from children to parents. This is why Model 2 in our example was successful in predicting total head count while only modelling `pop` effects at head count nodes. By not mixing levels in this way, a *conditional* aster model *must* model all levels and so usually needs many more parameters than an unconditional model.

1.5 Discussion

The key idea of aster models (as we see it) is the usefulness of what we have called unconditional aster models (FEF), which have low-dimensional sufficient statistics (1.9). Following Geyer and Thompson (1992), who argued in favour of exponential family models with the 'right' sufficient statistics (chosen to be scientifically interpretable), an idea they attributed to Jaynes (1978), we argue that scientists are likely to find among these models the ones of scientific interest.

We don't insist, though. The `aster` R package is even-handed with respect to conditional and unconditional models and conditional and uncondi-

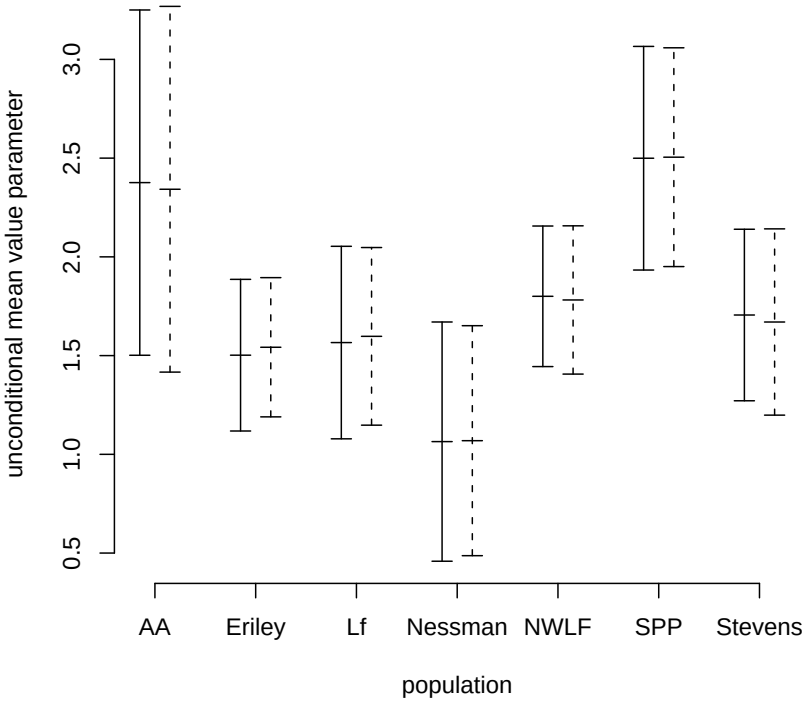

Figure 1.2: Confidence Intervals for Total Head Count. 95% (non-simultaneous) confidence intervals for the unconditional expectation of total flower head count (all three years) for individuals from different populations and central spatial location. Solid bars based on Model 2 in Table 1.1. Dashed bars based on Model 3 in Table 1.1.

tional parameters. Users may use whatever seems best to them. Any joint analysis is better than any separate analyses of different variables.

But we have one warning for naive users. Conditional models, though algebraically simple, are deceptively complicated statistically. As with all exponential family models, the map from canonical parameters to mean value parameters is monotone. So with sufficient statistics Y_k given by (1.9) we have

$$-\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \beta_k^2} = \frac{\partial E_{\boldsymbol{\beta}}\{Y_k\}}{\partial \beta_k} > 0 \tag{1.19a}$$

in an unconditional model. The middle term gives the regression coefficients their simple interpretation: an increase in β_k causes an increase in $E_\beta\{Y_k\}$, other betas being held constant. The analog for a conditional model is

$$-\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \beta_k^2} = \sum_{i \in I} \sum_{j \in J} \frac{\partial}{\partial \theta_{ij}} E_{\theta_{ij}}\{X_{ij} | X_{ip(j)}\} m_{ijk}^2 > 0. \tag{1.19b}$$

There is no way to give the middle term in (1.19b) an interpretation as a single conditional or unconditional expectation and hence no corresponding simple interpretation of regression coefficients.

Unconditional aster models seem algebraically complicated. If one attempts to understand them by understanding (1.5) and (1.16a) and (1.16b) and similarly complicated equations relegated to the appendices, one will find them very confusing. One can only understand them by understanding the big picture. They are the flat exponential families with the desired sufficient statistics. They are the aster models that behave according to the intuitions one has developed from linear and generalized linear models. They are statistically simple. In contrast, conditional aster models are algebraically simple but statistically complicated.

Aster models are, once one accepts the restriction to one-parameter exponential families for the single-node models, simply the right thing. There is no reason for the restriction to one-parameter exponential families other than our desire to keep the complexity manageable. Aster models are already very complicated and take some time to digest. We were afraid that allowing, for example, leaf nodes to be two-parameter normal would overwhelm both implementers (us) and users.

We saw in our example that aster models allowed us to successfully model the surrogate of fitness. In humans, Darwinian fitness is not the *summum bonum* that it is for organisms lacking language, memes, and minds. But neither is mere survival! Survival analysis, by its very name, rules out any consideration of quality of life. Aster models do permit such consideration. We understand that application of aster models to clinical trials involves medical and ethical considerations we are incompetent to address. We merely raise the point.

Acknowledgements

Our key equation (1.5), at least in the case where the graph is linear so each node has at most one successor, was figured out about 1980 when the third author was a graduate student and the first author not yet a graduate student. The third author's adviser Janis Antonovics provided some funding for development (from a now forgotten source, but we thank it), but we did not publish then, and other work intervened. High quality life history data generated recently by the second author with funding from NSF (DMS-0083468) and also by Julie Etterson, David Heiser, and Stacey Halpern finally motivated implementation of these ideas. Also Helen Hangelbroek did what in hindsight we would call a conditional aster model analysis of mortality data, and questions about her analysis led us back to the full aster theory. Were it not for Janis's help and encouragement back then, we might not have been able to do this now. We should also acknowledge the R team. Without R we would never have made software as usable and powerful as the `aster` package. We also thank Robert Gentleman for pointing out the medical implications.

Bibliography

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

Beatty, J. (1992). Fitness: Theoretical contexts. In *Keywords in Evolutionary Biology*, Ed. E. F. Keller and E. A. Lloyd, pp. 115–119. Cambridge, Mass: Harvard University Press.

Breslow, N. (1972). Discussion of the paper by Cox (1972). *J. R. Statist. Soc.* B **34**, 216–17.

Breslow, N. (1974). Covariance analysis of censored survival data. *Biometrics*, **30**, 89-99.

Chambers, J. M. & Hastie, T. J. (1992). Statistical models. In *Statistical Models in S*, J. M. Chambers and T. J. Hastie, eds. Pacific Grove, Calif.: Wadsworth & Brooks/Cole.

Cox, D. R. (1972). Regression models and life-tables (with discussion). *J. R. Statist. Soc.* B **34**, 187–220.

Geyer, C. J. & Thompson, E. A. (1992). Constrained Monte Carlo maximum likelihood for dependent data, (with discussion). *J. R. Statist. Soc.* B **54** 657–99.

Jaynes, E. T. (1978). Where do we stand on maximum entropy? In *The Maximum Entropy Formalism*, Ed. R. D. Levine and M. Tribus, pp. 15—118. Cambridge, Mass: MIT Press.

Keller, E. F. (1992). Fitness: Reproductive ambiguities. In *Keywords in Evolutionary Biology*, Ed. E. F. Keller and E. A. Lloyd, pp. 120–121. Cambridge, Mass: Harvard University Press.

Lauritzen, S. L. (1996). *Graphical Models.* New York: Oxford University Press.

McCullagh, P. & Nelder, J. A. (1989). *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

Paul, D. (1992). Fitness: Historical perspective. In *Keywords in Evolutionary Biology*, Ed. E. F. Keller and E. A. Lloyd, pp. 112–114. Cambridge, Mass: Harvard University Press.

R Development Core Team (2004). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. `http://www.R-project.org`.

Wilkinson, G. N. & Rogers, C. E. (1973). Symbolic description of factorial models for analysis of variance. *Applied Statistics* **22**, 392–99.

Appendix A

More Details on Prediction

A.1 Introduction

A.1.1 Functions of Regression Coefficients

Suppose we have an aster model, a maximum likelihood estimate $\hat{\boldsymbol{\beta}}$, and Fisher information $I(\hat{\boldsymbol{\beta}})$, whether observed or expected makes no difference. The asymptotic distribution of $\hat{\boldsymbol{\beta}}$ is

$$\text{Normal}\big(\boldsymbol{\beta}, I(\hat{\boldsymbol{\beta}})^{-1}\big)$$

where $\boldsymbol{\beta}$ is the true unknown parameter value.

But we usually don't want to make predictions about $\boldsymbol{\beta}$ (regression coefficients are meaningless, only probabilities and expectations are directly interpretable) but about some function $g(\boldsymbol{\beta})$, where g is a scalar-valued or vector-valued function. The delta method then says that the asymptotic distribution of $g(\hat{\boldsymbol{\beta}})$ is

$$\text{Normal}\big(g(\boldsymbol{\beta}), [\nabla g(\hat{\boldsymbol{\beta}})] I(\hat{\boldsymbol{\beta}})^{-1} [\nabla g(\hat{\boldsymbol{\beta}})]^T\big)$$

Since we already have $\hat{\boldsymbol{\beta}}$ and $I(\hat{\boldsymbol{\beta}})$, we only need $\nabla g(\hat{\boldsymbol{\beta}})$ to finish our problem.

A.1.2 Functions of Other Parameters

Since regression coefficients are meaningless, users will typically want to specify a function of some other parameter, for example the unconditional mean value parameter $\boldsymbol{\tau}$, which, of course, is itself a function of $\boldsymbol{\beta}$ if the model is correct. That is, we have a composition

$$g = h \circ f_{\boldsymbol{\beta},\boldsymbol{\tau}}$$

where h is an arbitrary function specified by the user and $f_{\boldsymbol{\beta},\boldsymbol{\tau}}$ is the map $\boldsymbol{\beta} \mapsto \boldsymbol{\tau}$. The point is that $f_{\boldsymbol{\beta},\boldsymbol{\tau}}$ is quite complicated but is understood (or should be understood) by the computer, whereas h may be quite simple.

Of course, from the chain rule we have

$$\nabla g(\boldsymbol{\beta}) = \nabla h(\boldsymbol{\tau}) \circ \nabla f_{\boldsymbol{\beta},\boldsymbol{\tau}}(\boldsymbol{\beta}) \tag{A.1a}$$

Since $\nabla h(\boldsymbol{\tau})$ and $\nabla f_{\boldsymbol{\beta},\boldsymbol{\tau}}(\boldsymbol{\beta})$ are linear operators represented by matrices (of partial derivatives), the composition here is represented by matrix multiplication. Of course, we want to use this with estimates plugged in.

$$\nabla g(\hat{\boldsymbol{\beta}}) = \nabla h(\hat{\boldsymbol{\tau}}) \circ \nabla f_{\boldsymbol{\beta},\boldsymbol{\tau}}(\hat{\boldsymbol{\beta}}) \tag{A.1b}$$

So the general idea is that the user supplies the easy part, the matrix representing $\nabla h(\hat{\boldsymbol{\tau}})$, and the computer does the hard part, the matrix representing $\nabla f_{\boldsymbol{\beta},\boldsymbol{\tau}}(\hat{\boldsymbol{\beta}})$.

There is one slight complication. The user often provides part of the specification of $\nabla f_{\boldsymbol{\beta},\boldsymbol{\tau}}(\hat{\boldsymbol{\beta}})$, for which see Section A.1.4 below.

A.1.3 What Other Parameters?

We suppose that in place of $\boldsymbol{\tau}$ in the preceding section, the user may chose to specify h in terms of any of the parameters we use in discussing aster models. There are four (the one already mentioned and three others)

- $\boldsymbol{\theta}$, the conditional canonical parameters.
- $\boldsymbol{\varphi}$, the unconditional canonical parameters.
- $\boldsymbol{\xi}$, the conditional mean value parameters.
- $\boldsymbol{\tau}$, the unconditional mean value parameters.

Letting $\boldsymbol{\zeta}$ stand for any one of these parameters, we replace $\boldsymbol{\tau}$ by $\boldsymbol{\zeta}$ in (A.1b) obtaining

$$\nabla g(\hat{\boldsymbol{\beta}}) = \nabla h(\hat{\boldsymbol{\zeta}}) \circ \nabla f_{\boldsymbol{\beta},\boldsymbol{\zeta}}(\hat{\boldsymbol{\beta}}) \tag{A.1c}$$

we need the user to be able to specify a $\nabla h(\hat{\boldsymbol{\zeta}})$ and have the computer produce the required $\nabla f_{\boldsymbol{\beta},\boldsymbol{\zeta}}(\hat{\boldsymbol{\beta}})$.

A.1.4 What Covariates?

An important thing the `predict.lm` function in R does is allow prediction at covariate values other than those in the observed data. In fact, this is its main "feature." The multiplicity of parameter values found in aster models is absent in simple least squares regression.

Let $\boldsymbol{\eta}$ be the canonical parameter that is linearly modeled in terms of $\boldsymbol{\beta}$ (either $\boldsymbol{\theta}$ for a conditional model or $\boldsymbol{\varphi}$ for an unconditional model). Then the "generalized linear" part of the aster model is

$$\boldsymbol{\eta} = \mathbf{M}\boldsymbol{\beta}$$

where $\mathbf{M}$ is a linear operator, represented by a matrix (the model matrix), but, *and this is important*, the $\mathbf{M}$ used in the prediction problem need not be the $\mathbf{M}$ used in fitting the model to observed data. The two model matrices must have the same column dimension and the columns must have the same meaning (covariate variables), but the rows need bear no analogy. Each row of the model matrix for the real data corresponds to a real individual, but rows of the model matrix for a prediction problem may correspond to entirely hypothetical individuals, or newly observed individuals not in the original data, or whatever. In this note we use the notation $\mathbf{M}$ to stand for the model matrix involved in our prediction problem. The model matrix for the original data, when needed, will be denoted $\mathbf{M}_{\text{orig}}$.

Now we can write our function to predict as the composition

$$g = h \circ f_{\boldsymbol{\eta},\boldsymbol{\zeta}} \circ \mathbf{M}$$

and the chain rule with plug-in becomes

$$\nabla g(\hat{\boldsymbol{\beta}}) = \nabla h(\hat{\boldsymbol{\zeta}}) \circ \nabla f_{\boldsymbol{\eta},\boldsymbol{\zeta}}(\hat{\boldsymbol{\eta}}) \circ \mathbf{M} \tag{A.1d}$$

(the derivative of a linear operator being the operator itself).

So the division of labor we envisage is that the user will specify the two matrices $\nabla h(\hat{\boldsymbol{\zeta}})$ and $\mathbf{M}$ (the latter either explicitly or, more usually, implicitly by specifying a new data frame to be used with the formula for the regression) and the computer must figure out $\nabla f_{\boldsymbol{\eta},\boldsymbol{\zeta}}(\hat{\boldsymbol{\eta}})$ by itself.

A.2 Changes of Parameter

A.2.1 Identity

For two "changes of parameter" of interest to us, $f_{\boldsymbol{\eta},\boldsymbol{\zeta}}$ is the identity mapping and hence so is $\nabla f_{\boldsymbol{\eta},\boldsymbol{\zeta}}$. These are the cases $\boldsymbol{\eta} = \boldsymbol{\zeta} = \boldsymbol{\theta}$ and $\boldsymbol{\eta} = \boldsymbol{\zeta} = \boldsymbol{\varphi}$. In these cases the computer's part of (A.1d) is trivial.

A.2.2 Conditional Canonical to Unconditional Canonical

This section deals with $f_{\boldsymbol{\theta},\boldsymbol{\varphi}}$, which is described by (1.6d). Let $\Delta\theta_{ij}$ denote an increment in θ_{ij} and similarly for $\Delta\varphi_{ij}$. Then

$$\Delta\varphi_{ij} = \Delta\theta_{ij} - \sum_{k \in S(j)} \psi_k'(\hat{\theta}_{ik}) \Delta\theta_{ik} \tag{A.2}$$

provides a description of $\nabla f_{\boldsymbol{\theta},\boldsymbol{\varphi}}(\hat{\boldsymbol{\theta}})$. It is a linear operator that maps a vector with components $\Delta\theta_{ij}$ to a vector with components $\Delta\varphi_{ij}$.

It is perhaps easier to understand this (and more useful to the actual computer programming of predictions) if we compute the composition $\nabla f_{\boldsymbol{\eta},\boldsymbol{\zeta}}(\hat{\boldsymbol{\eta}}) \circ \mathbf{M}$, the last two bits of (A.1d). We get, in this case where $\boldsymbol{\eta} = \boldsymbol{\theta}$,

$$\frac{\partial \varphi_{ij}}{\partial \beta_k} = m_{ijk} - \sum_{l \in S(j)} \psi_l'(\hat{\theta}_{il}) m_{ilk}$$

where m_{ijk} are the components of $\mathbf{M}$.

A.2.3 Unconditional Canonical to Conditional Canonical

This section deals with $f_{\boldsymbol{\varphi},\boldsymbol{\theta}}$, which is implicitly described by (1.6d). It is clear that inverting (A.2) gives

$$\Delta\theta_{ij} = \Delta\varphi_{ij} + \sum_{k \in S(j)} \psi_k'(\hat{\theta}_{ik}) \Delta\theta_{ik} \tag{A.3}$$

Since (A.3) has $\Delta\theta_{im}$ terms on both sides, it must be used recursively, as with many other aster model equations, including (1.6d) itself. Clearly, if (A.3) is used when $S(j)$ is empty, it is trivial. This gives us (A.3) for all "leaf" notes of the graph. We can then use (A.3) for j such that $S(j)$ is contained in the leaf nodes. This gives us more nodes done, and we can repeat, at each stage being able to use (A.3) for j such that $S(j)$ is contained in the set of nodes already done. When the graphical model is drawn, like Figure 1.1, with parents above children then the recursion moves up the graph from children to parents to grandparents and so forth.

If $\mathbf{G} = \nabla f_{\boldsymbol{\eta},\boldsymbol{\zeta}}(\hat{\boldsymbol{\eta}}) \circ \mathbf{M}$ had components g_{ijk} and $\mathbf{M}$ has components m_{ijk}, then we get in this case where $\boldsymbol{\eta} = \boldsymbol{\varphi}$,

$$g_{ijk} = m_{ijk} + \sum_{l \in S(j)} \psi_l'(\hat{\theta}_{il}) g_{ilk} \tag{A.4}$$

and (as discussed above), since $g_{i\cdot k}$ is on both sides of the equation (A.4) must be used recursively going from the leaves of the graph toward the roots.

A.2.4 Conditional Canonical to Conditional Mean Value

This section deals with $f_{\boldsymbol{\theta},\boldsymbol{\xi}}$, which is trivial given exponential family theory. This map is given by equation (1.10) which we repeat here

$$\xi_{ij} = X_{ip(j)} \psi_j'(\theta_{ij})$$

and so the derivative is trivially

$$\frac{\partial \xi_{ij}}{\partial \theta_{ij}} = X_{ip(j)} \psi_j''(\theta_{ij}) \tag{A.5}$$

(other partial derivatives being zero).

A.2.5 Unconditional Canonical to Unconditional Mean Value

This section deals with $f_{\boldsymbol{\varphi},\boldsymbol{\tau}}$, which is also trivial given exponential family theory. This map is given by equation (1.12) which we repeat here

$$\boldsymbol{\tau} = f_{\boldsymbol{\varphi},\boldsymbol{\tau}}(\boldsymbol{\varphi}) = \nabla\psi(\boldsymbol{\varphi})$$

and so the derivative is trivially

$$\nabla f_{\boldsymbol{\varphi},\boldsymbol{\tau}}(\boldsymbol{\varphi}) = \nabla^2\psi(\boldsymbol{\varphi}). \tag{A.6}$$

Although algebraically complicated, this map is already known to the computer, since it needs it to calculate Fisher information for unconditional models. It is completely described by equations (1.16a) and (1.16b) in Section 1.2.2.

A.2.6 Conditional Canonical to Unconditional Mean Value

This section deals with $f_{\boldsymbol{\theta},\boldsymbol{\tau}}$, which could be considered already done because of $f_{\boldsymbol{\theta},\boldsymbol{\tau}} = f_{\boldsymbol{\varphi},\boldsymbol{\tau}} \circ f_{\boldsymbol{\theta},\boldsymbol{\varphi}}$. We could compute derivatives using the chain rule.

But let us try something different. We have a simple expression of $\boldsymbol{\tau}$ in terms of $\boldsymbol{\theta}$ given by (1.12) in Chapter 1

$$\tau_{ij}(\boldsymbol{\theta}) = X_{if(j)} \prod_{\substack{m \in J \\ j \preceq m \prec f(j)}} \psi'_m(\theta_{im}).$$

We can easily differentiate this directly

$$\frac{\partial \tau_{ij}(\boldsymbol{\theta})}{\partial \theta_{im}} = \tau_{ij}(\boldsymbol{\theta}) \frac{\psi''_m(\theta_{im})}{\psi'_m(\theta_{im})}, \qquad j \preceq m \prec f(j)$$

(and other partial derivatives are zero).

A.2.7 Unconditional Canonical to Conditional Mean Value

This section deals with $f_{\boldsymbol{\varphi},\boldsymbol{\xi}}$, and this one we probably should consider already done using $f_{\boldsymbol{\varphi},\boldsymbol{\xi}} = f_{\boldsymbol{\theta},\boldsymbol{\xi}} \circ f_{\boldsymbol{\varphi},\boldsymbol{\theta}}$. We compute derivatives using the chain rule

$$\nabla f_{\boldsymbol{\varphi},\boldsymbol{\xi}}(\hat{\boldsymbol{\varphi}}) = \nabla f_{\boldsymbol{\theta},\boldsymbol{\xi}}(\hat{\boldsymbol{\theta}}) \circ \nabla f_{\boldsymbol{\varphi},\boldsymbol{\theta}}(\hat{\boldsymbol{\varphi}})$$

since one of these $\nabla f_{\boldsymbol{\theta},\boldsymbol{\xi}}(\hat{\boldsymbol{\theta}})$ is diagonal, given by (A.5), this should be easy. The other bit $\nabla f_{\boldsymbol{\varphi},\boldsymbol{\theta}}(\hat{\boldsymbol{\varphi}})$ is given by (A.4) and the following discussion.

A.3 Discussion

All of this is a bit brief, but it is the design document that was used to implement and test the code in the `aster` package for R.

Appendix B

Some One-Parameter Exponential Families for Aster Models

B.1 Bernoulli

B.1.1 Density

$$f_p(x) = p^x (1-p)^{1-x}$$

B.1.2 Canonical Parameter

The log density is

$$\begin{aligned} \log f_p(x) &= x \log p + (1-x) \log(1-p) \\ &= x \log\left(\frac{p}{1-p}\right) + \log(1-p) \end{aligned}$$

from which it is seen that the canonical parameter is

$$\theta = \operatorname{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

Note that the inverse map is

$$p = \operatorname{logit}^{-1}(\theta) = \frac{1}{1+e^{-\theta}}$$

B.1.3 Cumulant Function

We must have

$$\log f_p(x) = x\theta - \psi(\theta) + h(x)$$

from which we get

$$\begin{aligned} \psi(\theta) &= -\log(1-p) \\ &= -\log\left(\frac{1}{1+e^{\theta}}\right) \\ &= \log\left(1+e^{\theta}\right) \end{aligned}$$

B.1.4 Mean Function

By *mean function* we mean the map between the canonical parameter and the mean value parameter

$$\tau(\theta) = E_\theta X = \psi'(\theta).$$

Here

$$\tau(\theta) = \frac{1}{1+e^{-\theta}}$$

B.1.5 Variance Function

By *variance function* we mean the derivative of τ

$$\nu(\theta) = \tau'(\theta) = \operatorname{var}_\theta X = \psi''(\theta).$$

Here

$$\nu(\theta) = \frac{e^{-\theta}}{(1+e^{-\theta})^2}$$

B.1.6 Check

We do have

$$\begin{aligned} \tau(\theta) &= p \\ \nu(\theta) &= p(1-p) \end{aligned}$$

the familiar mean and variance of the Bernoulli distribution.

B.2 Poisson

B.2.1 Density

$$f_\mu(x) = \frac{\mu^x}{x!} e^{-\mu}$$

B.2.2 Canonical Parameter

The log density is

$$\log f_\mu(x) = x \log \mu - \mu - \log(x!)$$

from which it is seen that the canonical parameter is

$$\theta = \log(\mu) \tag{B.1a}$$

Note that the inverse map is

$$\mu = e^\theta \tag{B.1b}$$

B.2.3 Cumulant Function

And

$$\begin{aligned} \psi(\theta) &= \mu \\ &= e^\theta \end{aligned}$$

B.2.4 Mean Function

And

$$\tau(\theta) = e^\theta$$

B.2.5 Variance Function

And

$$\nu(\theta) = e^\theta$$

B.2.6 Check

We do have

$$\begin{aligned} \tau(\theta) &= \mu \\ \nu(\theta) &= \mu \end{aligned}$$

the familiar mean and variance of the Poisson distribution.

B.3 Poisson Conditioned on Non-Zero

B.3.1 Density

$$\begin{aligned} f_\mu(x) &= \frac{\mu^x}{x!} \cdot \frac{e^{-\mu}}{1 - e^{-\mu}} \\ &= \frac{\mu^x}{x!} \cdot \frac{1}{e^\mu - 1} \end{aligned}$$

(μ is the mean of the Poisson distribution not this distribution).

B.3.2 Canonical Parameter

The log density is

$$\log f_\mu(x) = x \log \mu - \log\left(e^\mu - 1\right) - \log(x!)$$

from which it is seen that the canonical parameter and its inverse are still given by (B.1a) and (B.1b).

B.3.3 Cumulant Function

And

$$\begin{aligned} \psi(\theta) &= \log\left(e^\mu - 1\right) \\ &= \log\left(e^{e^\theta} - 1\right) \end{aligned}$$

B.3.4 Mean Function

Here

$$\begin{aligned} \tau(\theta) &= \frac{e^{e^\theta} e^\theta}{e^{e^\theta} - 1} \\ &= \frac{e^\theta}{1 - e^{-e^\theta}} \end{aligned}$$

B.3.5 Variance Function

And

$$\begin{aligned} \nu(\theta) &= \frac{e^\theta}{1 - e^{-e^\theta}} - \frac{e^{-e^\theta} e^{2\theta}}{(1 - e^{-e^\theta})^2} \\ &= \tau(\theta)\left(1 - \tau(\theta) e^{-e^\theta}\right) \end{aligned}$$

B.3.6 Check

We do have

$$\begin{aligned} \tau(\theta) &= \frac{\mu}{1 - e^{-\mu}} \\ \nu(\theta) &= \frac{\mu[1 - (1 + \mu)e^{-\mu}]}{(1 - e^{-\mu})^2} \\ &= \tau(\theta)[1 - \tau(\theta) e^{-\mu}] \end{aligned}$$

the somewhat unfamiliar mean and variance of the Poisson distribution conditioned on not being zero. Despite it not being obvious from the formula, $\nu(\theta)$ is indeed nonnegative, as a variance must be, and goes to zero as θ goes to minus infinity.

Appendix C

Simulation of Poisson Conditioned on Being Nonzero

How to simulate Poisson conditional on nonzero? We have two cases.

- For large μ, we can do rejection sampling. Just simulate Poisson(μ) random variates until we get one greater than zero, and return it.
- For small μ, naive rejection sampling can be arbitrarily slow. So we need another strategy.

Let us be slightly more sophisticated about our rejection sampling for small μ. The density of X is

$$f_\mu(x) = \frac{\mu^x}{x!} \frac{e^{-\mu}}{1 - e^{-\mu}}, \qquad x = 1, 2, \ldots.$$

Consider rejection sampling from Y which is one plus a Poisson(ν). The density of Y is

$$g_\nu(y) = \frac{\nu^{y-1}}{(y-1)!} e^{-\nu}, \qquad y = 1, 2, \ldots.$$

The ratio of the two densities is

$$\begin{aligned} \frac{f_\mu(x)}{g_\nu(x)} &= \frac{\mu^x}{\nu^{x-1}} \cdot \frac{(x-1)!}{x!} \cdot \frac{e^{-\mu}}{1 - e^{-\mu}} \cdot \frac{1}{e^{-\nu}} \\ &= \left(\frac{\mu}{\nu}\right)^{x-1} \cdot \frac{1}{x} \cdot \frac{\mu e^{-\mu}}{1 - e^{-\mu}} \cdot e^{\nu} \end{aligned}$$

This is bounded above considered as a function of x (so rejection sampling is possible at all) if and only if $\mu \le \nu$.

Introduce the notation

$$\tau = \frac{\mu}{1 - e^{-\mu}}$$

for the mean of X. And the upper bound is

$$\max_{x \ge 1} \frac{f_\mu(x)}{g_\nu(x)} = \frac{f_\mu(1)}{g_\nu(1)} = \tau e^{-\mu} \cdot e^{\nu}$$

If we're not worried about being maximally clever, then we can take $\mu = \nu$, so the upper bound becomes just τ. And what happens if we take the break point between the two schemes to be $\mu = 1$? The first scheme then has worst case rejection fraction

```
> dpois(0, 1)

[1] 0.3678794
```

And the second scheme then has worst case rejection fraction

```
> mu <- 1
> tau <- mu/(1 - exp(-mu))
> print(tau)

[1] 1.581977

> fmu <- function(x) dpois(x, mu)/(1 - dpois(0, mu))
> gnu <- function(y) dpois(y - 1, mu)
> xxx <- seq(1, 100)
> yyy <- fmu(xxx)/(tau * gnu(xxx))
> all.equal(yyy, 1/xxx)

[1] TRUE

> max(yyy)

[1] 1
```

Now rejection sampling simulates $X \sim 1 + \text{Poisson}(\mu)$ and $U \sim \text{Uniform}(0, 1)$ and accepts this X if $U < 1/X$.

The probability it fails to do so is

```
> 1 - sum(1/xxx * gnu(xxx))

[1] 0.3678794
```

Perfect balance!

Appendix D

Example Analysis of Purple Coneflower (*Echinacea angustifolia*) Data

D.1 Preliminaries

Load the `aster` package and the data.

```
> library(aster)
> data(echinacea)
> names(echinacea)
```

```
 [1] "hdct02" "hdct03" "hdct04" "pop"    "ewloc"  "nsloc"
 [7] "ld02"   "fl02"   "ld03"   "fl03"   "ld04"   "fl04"
```

the variables with numbers in the names are the columns of the response matrix of the aster model. The variables `ld0`x (where x is a digit) are the survival indicator variables for year $200x$ (one for alive, zero for dead). The variables `fl0`x are the flowering indicator variables (one for any flowers, zero for none). The variables `hdct0`x are the inflorescence (flower head) count variables (number of flower heads). The variables without numbers are other predictors. The variables `ewloc` and `nsloc` are spatial positions (east-west and north-south location, respectively). The variable `pop` is the remnant population of origin of the plant, so plants with different values of `pop` may be more genetically diverse than those with the same values of `pop`.

Make the graph (of the graphical model, specified by a function p given by an R vector `pred`).

```
> pred <- c(0, 1, 2, 1, 2, 3, 4, 5, 6)
> fam <- c(1, 1, 1, 1, 1, 1, 3, 3, 3)
```

Reshape the data.

```
> vars <- c("ld02", "ld03", "ld04", "fl02", "fl03",
+     "fl04", "hdct02", "hdct03", "hdct04")
> redata <- reshape(echinacea, varying = list(vars),
+     direction = "long", timevar = "varb", times = as.factor(vars),
+     v.names = "resp")
> redata <- data.frame(redata, root = 1)
> names(redata)

[1] "pop"   "ewloc" "nsloc" "varb"  "resp"  "id"    "root"
```

D.2 Modeling

D.2.1 First Model

For our first model we try something simple (moderately simple). We have no population effects. We put in a mean and effect of north-south and east-west position for each of the nine variables. That gives us $3 \times 9 = 27$ parameters.

```
> out1 <- aster(resp ~ varb + varb:nsloc + varb:ewloc,
+     pred, fam, varb, id, root, data = redata)
> summary(out1, show.graph = TRUE)

Call:
aster.formula(formula = resp ~ varb + varb:nsloc + varb:ewloc,
    pred = pred, fam = fam, varvar = varb, idvar = id, root = root,
    data = redata)


Graphical Model:
 variable predecessor family
 ld02     root        bernoulli
 ld03     ld02        bernoulli
 ld04     ld03        bernoulli
 fl02     ld02        bernoulli
 fl03     ld03        bernoulli
 fl04     ld04        bernoulli
 hdct02   fl02        non.zero.poisson
 hdct03   fl03        non.zero.poisson
 hdct04   fl04        non.zero.poisson

                  Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept)      -1.6418322  0.1928540  -8.513  < 2e-16 ***
varbfl03         -0.2585034  0.2961442  -0.873  0.38272
varbfl04         -0.3368137  0.2601966  -1.294  0.19551
varbhdct02        1.9661735  0.2675730   7.348 2.01e-13 ***
varbhdct03        1.9185775  0.2203482   8.707  < 2e-16 ***
varbhdct04        2.4696479  0.2001443  12.339  < 2e-16 ***
varbld02         -0.9772646  0.3502261  -2.790  0.00526 **
varbld03          1.0373523  0.4281242   2.423  0.01539 *
varbld04          4.1243999  0.3469256  11.888  < 2e-16 ***
varbfl02:nsloc    0.0838694  0.0265213   3.162  0.00157 **
varbfl03:nsloc    0.0655407  0.0302581   2.166  0.03031 *
varbfl04:nsloc    0.0698900  0.0244358   2.860  0.00423 **
varbhdct02:nsloc -0.0179987  0.0116776  -1.541  0.12324
varbhdct03:nsloc  0.0001156  0.0138761   0.008  0.99335
varbhdct04:nsloc -0.0034831  0.0074230  -0.469  0.63891
varbld02:nsloc   -0.0463586  0.0376371  -1.232  0.21805
varbld03:nsloc    0.0291040  0.0527152   0.552  0.58088
varbld04:nsloc    0.0339060  0.0405000   0.837  0.40249
varbfl02:ewloc    0.0460555  0.0267844   1.719  0.08553 .
varbfl03:ewloc   -0.0060486  0.0290874  -0.208  0.83527
varbfl04:ewloc   -0.0094399  0.0239818  -0.394  0.69385
varbhdct02:ewloc  0.0024187  0.0120993   0.200  0.84156
varbhdct03:ewloc  0.0230630  0.0135295   1.705  0.08826 .
varbhdct04:ewloc  0.0084179  0.0072661   1.159  0.24665
varbld02:ewloc    0.0149197  0.0357336   0.418  0.67629
varbld03:ewloc   -0.0285742  0.0517154  -0.553  0.58059
varbld04:ewloc    0.0048502  0.0409964   0.118  0.90582
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

D.2.2 Second Model

So now we put in population.

```
> levels(echinacea$pop)

[1] "AA"      "Eriley"  "Lf"      "Nessman" "NWLF"    "SPP"
[7] "Stevens"
```

Let us put `pop` in only at the top level in this model (just to see what happens). In order to do that we have to add a predictor that "predicts" the top level.

```
> hdct <- grep("hdct", as.character(redata$varb))
> hdct <- is.element(seq(along = redata$varb), hdct)
```

```
> redata <- data.frame(redata, hdct = as.integer(hdct))
> names(redata)

[1] "pop"   "ewloc" "nsloc" "varb"  "resp"  "id"    "root"
[8] "hdct"

> out2 <- aster(resp ~ varb + varb:nsloc + varb:ewloc +
+     hdct * pop - pop, pred, fam, varb, id, root,
+     data = redata)
> summary(out2)

Call:
aster.formula(formula = resp ~ varb + varb:nsloc + varb:ewloc +
    hdct * pop - pop, pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = redata)

                  Estimate Std. Error z value Pr(>|z|)
(Intercept)     -1.6259835  0.1925029  -8.447  < 2e-16 ***
varbfl03        -0.2591673  0.2953707  -0.877  0.38025
varbfl04        -0.3272159  0.2594037  -1.261  0.20716
varbhdct02       2.0516689  0.2751442   7.457 8.87e-14 ***
varbhdct03       2.0032366  0.2298569   8.715  < 2e-16 ***
varbhdct04       2.5544660  0.2106706  12.125  < 2e-16 ***
varbld02        -0.9847880  0.3499238  -2.814  0.00489 **
varbld03         1.0268923  0.4279181   2.400  0.01641 *
varbld04         4.1086115  0.3467427  11.849  < 2e-16 ***
varbfl02:nsloc   0.0835490  0.0264359   3.160  0.00158 **
varbfl03:nsloc   0.0655031  0.0301469   2.173  0.02980 *
varbfl04:nsloc   0.0698326  0.0243307   2.870  0.00410 **
varbhdct02:nsloc -0.0184434  0.0116321  -1.586  0.11284
varbhdct03:nsloc -0.0004162  0.0138183  -0.030  0.97597
varbhdct04:nsloc -0.0038684  0.0073788  -0.524  0.60010
varbld02:nsloc   -0.0459235  0.0376414  -1.220  0.22245
varbld03:nsloc    0.0295182  0.0527428   0.560  0.57571
varbld04:nsloc    0.0339706  0.0405350   0.838  0.40200
varbfl02:ewloc    0.0462769  0.0267337   1.731  0.08345 .
varbfl03:ewloc   -0.0054404  0.0289865  -0.188  0.85112
varbfl04:ewloc   -0.0089080  0.0239051  -0.373  0.70942
varbhdct02:ewloc  0.0028066  0.0120847   0.232  0.81635
varbhdct03:ewloc  0.0231490  0.0134693   1.719  0.08568 .
varbhdct04:ewloc  0.0086106  0.0072716   1.184  0.23636
varbld02:ewloc    0.0153283  0.0357260   0.429  0.66789
varbld03:ewloc   -0.0283036  0.0517141  -0.547  0.58417
```

```
varbld04:ewloc    0.0048550  0.0409914   0.118  0.90572
hdct:popEriley   -0.1782285  0.0893919  -1.994  0.04618 *
hdct:popLf       -0.1617692  0.0960825  -1.684  0.09225 .
hdct:popNessman  -0.3152202  0.1387855  -2.271  0.02313 *
hdct:popNWLF     -0.1076672  0.0832346  -1.294  0.19582
hdct:popSPP       0.0198694  0.0862319   0.230  0.81777
hdct:popStevens  -0.1288083  0.0891936  -1.444  0.14870
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Original predictor variables dropped (aliased)
     hdct
```

```
> anova(out1, out2)

Analysis of Deviance Table

Model 1: resp ~ varb + varb:nsloc + varb:ewloc
Model 2: resp ~ varb + varb:nsloc + varb:ewloc + hdct * pop - pop
  Model Df Model Dev Df Deviance P(>|Chi|)
1       27   2717.39
2       33   2701.26  6    16.13      0.01
```

Comment The last bit of the summary that the "original predictor" `hdct` was "dropped (aliased)" means just what it (tersely) says. The R formula mini-language as implemented in the R functions `model.frame` and `model.matrix` produces a model matrix that is not full rank. In order to estimate anything we must drop some dummy variable that it constructed. In this particular case the (dummy variable that is the indicator of) `hdct` is equal to the sum of the (dummy variables that are the indicators of) `varbhdct02`, `varbhdct03`, and `varbhdct04`. Thus we must drop one of these variables for the model to be identifiable. So `aster` does.

Comment The reason for the `- pop` in the formula is not obvious. In fact, we originally did not write the formula this way and got the wrong model (see "Tenth Model" in Section D.2.10 below). It took some grovelling in various bits of R documentation to come up with this `- pop` trick, but once you see it, the effect is clear.

We want population effects only at the "hdct" level. But the `hdct * pop` crosses the `hdct` indicator variable, which has two values (zero and one) with the `pop` variable, which has seven values (the seven populations), giving 14 parameters, one of which R drops (because it is aliased with the intercept). But that's not what we want. We don't want `pop` effects at the "non-hdct"

levels. The way the R formula mini-language allows us to specify that is `- pop` which means to leave out the population main effects (7 fewer parameters, leaving 6) and we see that we do indeed have 6 degrees of freedom difference between models one and two.

D.2.3 Third Model

Let us now put `pop` in at all levels in this model.

```
> level <- gsub("[0-9]", "", as.character(redata$varb))
> redata <- data.frame(redata, level = as.factor(level))
> out3 <- aster(resp ~ varb + varb:nsloc + varb:ewloc +
+     level * pop, pred, fam, varb, id, root, data = redata)
> summary(out3)

Call:
aster.formula(formula = resp ~ varb + varb:nsloc + varb:ewloc +
    level * pop, pred = pred, fam = fam, varvar = varb, idvar = id,
    root = root, data = redata)

                    Estimate Std. Error z value Pr(>|z|)
(Intercept)         -1.937453   0.423538  -4.574 4.77e-06 ***
varbfl03            -0.268508   0.293765  -0.914  0.36071
varbfl04            -0.303001   0.257928  -1.175  0.24009
varbhdct02           2.477882   0.546517   4.534 5.79e-06 ***
varbhdct03           2.433586   0.525941   4.627 3.71e-06 ***
varbhdct04           2.972531   0.516328   5.757 8.56e-09 ***
varbld02            -0.739801   0.582421  -1.270  0.20401
varbld03             1.256976   0.632563   1.987  0.04691 *
varbld04             4.322854   0.581362   7.436 1.04e-13 ***
popEriley            0.807390   0.451694   1.787  0.07386 .
popLf                0.877245   0.481248   1.823  0.06833 .
popNessman          -0.602180   0.681430  -0.884  0.37686
popNWLF             -0.111507   0.434671  -0.257  0.79754
popSPP               0.541625   0.450808   1.201  0.22958
popStevens           0.112023   0.465345   0.241  0.80976
varbfl02:nsloc       0.083795   0.026329   3.183  0.00146 **
varbfl03:nsloc       0.066937   0.029982   2.233  0.02558 *
varbfl04:nsloc       0.070146   0.024162   2.903  0.00370 **
varbhdct02:nsloc    -0.018769   0.011491  -1.633  0.10240
varbhdct03:nsloc    -0.001417   0.013619  -0.104  0.91715
varbhdct04:nsloc    -0.004397   0.007261  -0.606  0.54477
varbld02:nsloc      -0.044854   0.037640  -1.192  0.23340
```

```
varbld03:nsloc        0.030181   0.052719   0.572  0.56699
varbld04:nsloc        0.034831   0.040522   0.860  0.39003
varbfl02:ewloc        0.035972   0.026717   1.346  0.17817
varbfl03:ewloc       -0.015152   0.028934  -0.524  0.60050
varbfl04:ewloc       -0.018685   0.023867  -0.783  0.43371
varbhdct02:ewloc      0.006926   0.012016   0.576  0.56435
varbhdct03:ewloc      0.026914   0.013389   2.010  0.04441 *
varbhdct04:ewloc      0.012554   0.007241   1.734  0.08298 .
varbld02:ewloc        0.013009   0.035812   0.363  0.71641
varbld03:ewloc       -0.030830   0.051730  -0.596  0.55119
varbld04:ewloc        0.002422   0.040976   0.059  0.95287
levelhdct:popEriley  -1.341288   0.587473  -2.283  0.02242 *
levelld:popEriley    -0.560351   0.553094  -1.013  0.31100
levelhdct:popLf      -1.408221   0.631418  -2.230  0.02573 *
levelld:popLf        -0.674658   0.588417  -1.147  0.25156
levelhdct:popNessman  0.418459   0.893928   0.468  0.63970
levelld:popNessman    0.922668   0.778683   1.185  0.23605
levelhdct:popNWLF     0.046822   0.553566   0.085  0.93259
levelld:popNWLF       0.140229   0.531548   0.264  0.79192
levelhdct:popSPP     -0.702929   0.573865  -1.225  0.22061
levelld:popSPP       -0.479659   0.561548  -0.854  0.39301
levelhdct:popStevens -0.215472   0.593594  -0.363  0.71661
levelld:popStevens   -0.240029   0.569320  -0.422  0.67331
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Original predictor variables dropped (aliased)
     levelhdct
     levelld


> anova(out1, out2, out3)


Analysis of Deviance Table

Model 1: resp ~ varb + varb:nsloc + varb:ewloc
Model 2: resp ~ varb + varb:nsloc + varb:ewloc + hdct * pop - pop
Model 3: resp ~ varb + varb:nsloc + varb:ewloc + level * pop
  Model Df Model Dev Df Deviance P(>|Chi|)
1       27   2717.39
2       33   2701.26  6    16.13      0.01
3       45   2663.55 12    37.71 0.0001715
```

Comment This would finish a sensible analysis, but we're really not sure we have dealt with the "geometry" (the variables `ewloc` and `nsloc`) correctly.

D.2.4 Fourth Model, Less Geometry

Thus we experiment with different ways to put in the spatial effects. First we reduce the geometry to a product, either year or level.

```
> year <- gsub("[a-z]", "", as.character(redata$varb))
> year <- paste("yr", year, sep = "")
> redata <- data.frame(redata, year = as.factor(year))
> out4 <- aster(resp ~ varb + (level + year):(nsloc +
+     ewloc) + level * pop, pred, fam, varb, id, root,
+     data = redata)
> summary(out4)

Call:
aster.formula(formula = resp ~ varb + (level + year):(nsloc +
    ewloc) + level * pop, pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = redata)

                     Estimate Std. Error z value Pr(>|z|)
(Intercept)         -1.875208   0.419723  -4.468 7.91e-06 ***
varbfl03            -0.384991   0.268534  -1.434  0.15166
varbfl04            -0.373643   0.244457  -1.528  0.12640
varbhdct02           2.387001   0.541799   4.406 1.05e-05 ***
varbhdct03           2.399402   0.522036   4.596 4.30e-06 ***
varbhdct04           2.913906   0.513754   5.672 1.41e-08 ***
varbld02            -0.679149   0.562506  -1.207  0.22729
varbld03             1.114165   0.611123   1.823  0.06828 .
varbld04             4.225862   0.573484   7.369 1.72e-13 ***
popEriley            0.809271   0.451391   1.793  0.07300 .
popLf                0.878454   0.480895   1.827  0.06774 .
popNessman          -0.600230   0.681076  -0.881  0.37816
popNWLF             -0.111683   0.434221  -0.257  0.79702
popSPP               0.540528   0.450468   1.200  0.23017
popStevens           0.110291   0.464961   0.237  0.81250
levelfl:nsloc        0.068603   0.015126   4.535 5.75e-06 ***
levelhdct:nsloc     -0.014500   0.007501  -1.933  0.05324 .
levelld:nsloc        0.001078   0.007299   0.148  0.88258
levelfl:ewloc        0.004590   0.015092   0.304  0.76102
levelhdct:ewloc      0.020550   0.007740   2.655  0.00793 **
levelld:ewloc       -0.001435   0.007502  -0.191  0.84825
```

```
yearyr03:nsloc        0.009125   0.007155   1.275  0.20224
yearyr04:nsloc        0.009184   0.006137   1.496  0.13453
yearyr03:ewloc       -0.001812   0.007242  -0.250  0.80241
yearyr04:ewloc       -0.011085   0.006324  -1.753  0.07965 .
levelhdct:popEriley  -1.344673   0.587258  -2.290  0.02204 *
levelld:popEriley    -0.561644   0.552345  -1.017  0.30923
levelhdct:popLf      -1.410639   0.631143  -2.235  0.02541 *
levelld:popLf        -0.674982   0.587591  -1.149  0.25067
levelhdct:popNessman  0.416160   0.893648   0.466  0.64144
levelld:popNessman    0.919968   0.777808   1.183  0.23690
levelhdct:popNWLF     0.046781   0.553089   0.085  0.93259
levelld:popNWLF       0.140718   0.530656   0.265  0.79087
levelhdct:popSPP     -0.701296   0.573572  -1.223  0.22145
levelld:popSPP       -0.477678   0.560739  -0.852  0.39429
levelhdct:popStevens -0.213370   0.593238  -0.360  0.71909
levelld:popStevens   -0.237795   0.568455  -0.418  0.67571
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Original predictor variables dropped (aliased)
     levelhdct
     levelld

> anova(out4, out3)

Analysis of Deviance Table

Model 1: resp ~ varb + (level + year):(nsloc + ewloc) + level * pop
Model 2: resp ~ varb + varb:nsloc + varb:ewloc + level * pop
  Model Df Model Dev Df Deviance P(>|Chi|)
1       37   2668.39
2       45   2663.55  8     4.83      0.78
```

So we have goodness of fit, and this can be our "big model".

D.2.5 Fifth Model, Much Less Geometry

Now we reduce the geometry to just two predictors.

```
> out5 <- aster(resp ~ varb + nsloc + ewloc + level *
+     pop, pred, fam, varb, id, root, data = redata)
> summary(out5)
```

```
Call:
aster.formula(formula = resp ~ varb + nsloc + ewloc + level *
    pop, pred = pred, fam = fam, varvar = varb, idvar = id, root = root,
    data = redata)

                     Estimate Std. Error z value Pr(>|z|)
(Intercept)          -1.740778   0.415402  -4.191 2.78e-05 ***
varbfl03             -0.334899   0.264510  -1.266 0.205473
varbfl04             -0.328208   0.240321  -1.366 0.172032
varbhdct02            2.190026   0.536935   4.079 4.53e-05 ***
varbhdct03            2.209167   0.516225   4.279 1.87e-05 ***
varbhdct04            2.712799   0.509537   5.324 1.01e-07 ***
varbld02             -0.769927   0.562247  -1.369 0.170882
varbld03              0.982700   0.611052   1.608 0.107789
varbld04              4.110627   0.573689   7.165 7.76e-13 ***
nsloc                 0.013506   0.001725   7.828 4.97e-15 ***
ewloc                 0.005965   0.001722   3.465 0.000531 ***
popEriley             0.755567   0.448815   1.683 0.092284 .
popLf                 0.809411   0.478696   1.691 0.090862 .
popNessman           -0.721910   0.677652  -1.065 0.286735
popNWLF              -0.138061   0.433020  -0.319 0.749854
popSPP                0.516341   0.448626   1.151 0.249757
popStevens            0.080025   0.464014   0.172 0.863074
levelhdct:popEriley  -1.259136   0.585222  -2.152 0.031433 *
levelld:popEriley    -0.548422   0.554771  -0.989 0.322881
levelhdct:popLf      -1.308189   0.629591  -2.078 0.037724 *
levelld:popLf        -0.635814   0.590884  -1.076 0.281910
levelhdct:popNessman  0.581044   0.890106   0.653 0.513898
levelld:popNessman    1.048141   0.779258   1.345 0.178609
levelhdct:popNWLF     0.072362   0.552482   0.131 0.895795
levelld:popNWLF       0.183780   0.534492   0.344 0.730967
levelhdct:popSPP     -0.662518   0.572464  -1.157 0.247147
levelld:popSPP       -0.474797   0.564778  -0.841 0.400528
levelhdct:popStevens -0.171288   0.593123  -0.289 0.772742
levelld:popStevens   -0.216441   0.572985  -0.378 0.705622
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Original predictor variables dropped (aliased)
     levelhdct
     levelld
```

```
> anova(out5, out4, out3)
```

```
Analysis of Deviance Table

Model 1: resp ~ varb + nsloc + ewloc + level * pop
Model 2: resp ~ varb + (level + year):(nsloc + ewloc) + level * pop
Model 3: resp ~ varb + varb:nsloc + varb:ewloc + level * pop
  Model Df Model Dev Df Deviance P(>|Chi|)
1       29   2696.56
2       37   2668.39  8    28.18 0.0004416
3       45   2663.55  8     4.83      0.78
```

So we do not have goodness of fit, and this cannot be our "big model".

D.2.6 Sixth Model, Intermediate Geometry, Levels

So we try again with the geometry.

```
> out6 <- aster(resp ~ varb + level:(nsloc + ewloc) +
+     level * pop, pred, fam, varb, id, root, data = redata)
> summary(out6)

Call:
aster.formula(formula = resp ~ varb + level:(nsloc + ewloc) +
    level * pop, pred = pred, fam = fam, varvar = varb, idvar = id,
    root = root, data = redata)

                    Estimate Std. Error z value Pr(>|z|)
(Intercept)         -1.907186   0.420138  -4.539 5.64e-06 ***
varbfl03            -0.354186   0.266796  -1.328   0.1843
varbfl04            -0.319952   0.242895  -1.317   0.1878
varbhdct02           2.424075   0.542402   4.469 7.85e-06 ***
varbhdct03           2.447742   0.521950   4.690 2.74e-06 ***
varbhdct04           2.945476   0.514751   5.722 1.05e-08 ***
varbld02            -0.626114   0.562017  -1.114   0.2653
varbld03             1.128493   0.611062   1.847   0.0648 .
varbld04             4.254223   0.573494   7.418 1.19e-13 ***
popEriley            0.812524   0.451595   1.799   0.0720 .
popLf                0.882088   0.481103   1.833   0.0667 .
popNessman          -0.593161   0.680863  -0.871   0.3837
popNWLF             -0.110570   0.434822  -0.254   0.7993
popSPP               0.543515   0.450805   1.206   0.2280
popStevens           0.114072   0.465336   0.245   0.8063
levelfl:nsloc        0.070763   0.014568   4.857 1.19e-06 ***
levelhdct:nsloc     -0.006425   0.005465  -1.176   0.2398
levelld:nsloc        0.008036   0.005924   1.356   0.1749
```

```
levelfl:ewloc          0.007768   0.014546   0.534   0.5933
levelhdct:ewloc        0.011689   0.005561   2.102   0.0356 *
levelld:ewloc         -0.007240   0.006114  -1.184   0.2363
levelhdct:popEriley   -1.350424   0.587897  -2.297   0.0216 *
levelld:popEriley     -0.563945   0.552117  -1.021   0.3071
levelhdct:popLf       -1.416802   0.631793  -2.243   0.0249 *
levelld:popLf         -0.678297   0.587394  -1.155   0.2482
levelhdct:popNessman   0.405845   0.893667   0.454   0.6497
levelld:popNessman     0.912760   0.777262   1.174   0.2403
levelhdct:popNWLF      0.044673   0.554325   0.081   0.9358
levelld:popNWLF        0.139453   0.530726   0.263   0.7927
levelhdct:popSPP      -0.705876   0.574445  -1.229   0.2191
levelld:popSPP        -0.481472   0.560611  -0.859   0.3904
levelhdct:popStevens  -0.218838   0.594180  -0.368   0.7126
levelld:popStevens    -0.242044   0.568322  -0.426   0.6702
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Original predictor variables dropped (aliased)
     levelhdct
     levelld

> anova(out5, out6, out4, out3)

Analysis of Deviance Table

Model 1: resp ~ varb + nsloc + ewloc + level * pop
Model 2: resp ~ varb + level:(nsloc + ewloc) + level * pop
Model 3: resp ~ varb + (level + year):(nsloc + ewloc) + level * pop
Model 4: resp ~ varb + varb:nsloc + varb:ewloc + level * pop
  Model Df Model Dev Df Deviance P(>|Chi|)
1       29   2696.56
2       33   2674.70  4    21.87  0.000213
3       37   2668.39  4     6.31      0.18
4       45   2663.55  8     4.83      0.78
```

So we have goodness of fit, and this can be our “big model”. But why drop year rather than level?

D.2.7 Seventh Model, Intermediate Geometry, Years

So we try again with the geometry.

```
> out7 <- aster(resp ~ varb + year:(nsloc + ewloc) +
+     level * pop, pred, fam, varb, id, root, data = redata)
> summary(out7)

Call:
aster.formula(formula = resp ~ varb + year:(nsloc + ewloc) +
    level * pop, pred = pred, fam = fam, varvar = varb, idvar = id,
    root = root, data = redata)

                     Estimate Std. Error z value Pr(>|z|)
(Intercept)          -1.737420   0.416050  -4.176 2.97e-05 ***
varbfl03             -0.316825   0.265066  -1.195 0.231981
varbfl04             -0.337380   0.241251  -1.398 0.161973
varbhdct02            2.193871   0.538051   4.077 4.55e-05 ***
varbhdct03            2.168558   0.517159   4.193 2.75e-05 ***
varbhdct04            2.713968   0.510169   5.320 1.04e-07 ***
varbld02             -0.778587   0.562724  -1.384 0.166480
varbld03              0.992898   0.611831   1.623 0.104625
varbld04              4.099985   0.574053   7.142 9.19e-13 ***
popEriley             0.741522   0.449483   1.650 0.099000 .
popLf                 0.799687   0.479291   1.668 0.095221 .
popNessman           -0.727352   0.678773  -1.072 0.283914
popNWLF              -0.128513   0.433273  -0.297 0.766765
popSPP                0.507401   0.449178   1.130 0.258636
popStevens            0.075879   0.464452   0.163 0.870225
yearyr02:nsloc        0.009909   0.004372   2.266 0.023441 *
yearyr03:nsloc        0.019664   0.004995   3.937 8.27e-05 ***
yearyr04:nsloc        0.012311   0.003276   3.758 0.000172 ***
yearyr02:ewloc        0.010774   0.004435   2.430 0.015116 *
yearyr03:ewloc        0.008521   0.004798   1.776 0.075742 .
yearyr04:ewloc        0.001578   0.003246   0.486 0.626836
levelhdct:popEriley  -1.239189   0.585903  -2.115 0.034429 *
levelld:popEriley    -0.532712   0.555127  -0.960 0.337246
levelhdct:popLf      -1.294188   0.630147  -2.054 0.039996 *
levelld:popLf        -0.624375   0.591157  -1.056 0.290881
levelhdct:popNessman  0.587678   0.891330   0.659 0.509686
levelld:popNessman    1.055395   0.779991   1.353 0.176028
levelhdct:popNWLF     0.059302   0.552518   0.107 0.914526
levelld:popNWLF       0.174231   0.534494   0.326 0.744444
levelhdct:popSPP     -0.650145   0.572908  -1.135 0.256452
levelld:popSPP       -0.463775   0.564970  -0.821 0.411712
levelhdct:popStevens -0.165745   0.593407  -0.279 0.780007
levelld:popStevens   -0.211556   0.573114  -0.369 0.712028
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Original predictor variables dropped (aliased)
     levelhdct
     levelld

> anova(out5, out7, out4, out3)

Analysis of Deviance Table

Model 1: resp ~ varb + nsloc + ewloc + level * pop
Model 2: resp ~ varb + year:(nsloc + ewloc) + level * pop
Model 3: resp ~ varb + (level + year):(nsloc + ewloc) + level * pop
Model 4: resp ~ varb + varb:nsloc + varb:ewloc + level * pop
  Model Df Model Dev Df Deviance P(>|Chi|)
1       29   2696.56
2       33   2691.95  4     4.61      0.33
3       37   2668.39  4    23.57 9.761e-05
4       45   2663.55  8     4.83      0.78
```

And we do not have goodness of fit! So Model Six is our “big model” and we have been logical in our model selection. Note that it is not valid to compare Models Six and Seven because they are not nested, but both fit between Models Five and Four, so Six and Seven can each be compared to both Five and Four (and this tells us what we want to know).

D.2.8 Eighth Model, Like Models Two and Six

We need to make a model with the structure of Model Two with respect to variables and like Model Six with respect to geometry.

```
> out8 <- aster(resp ~ varb + level:(nsloc + ewloc) +
+     hdct * pop - pop, pred, fam, varb, id, root,
+     data = redata)
> summary(out8)

Call:
aster.formula(formula = resp ~ varb + level:(nsloc + ewloc) +
    hdct * pop - pop, pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = redata)

                Estimate Std. Error z value Pr(>|z|)
(Intercept)    -1.591968   0.184332  -8.636  < 2e-16 ***
```

```
varbfl03        -0.349096   0.267919  -1.303  0.19258
varbfl04        -0.344222   0.243899  -1.411  0.15815
varbhdct02       1.991976   0.265192   7.511 5.85e-14 ***
varbhdct03       2.013936   0.219375   9.180  < 2e-16 ***
varbhdct04       2.521890   0.205048  12.299  < 2e-16 ***
varbld02        -0.874272   0.315703  -2.769  0.00562 **
varbld03         0.895081   0.396189   2.259  0.02387 *
varbld04         4.036755   0.334266  12.076  < 2e-16 ***
levelfl:nsloc    0.070102   0.014652   4.785 1.71e-06 ***
levelhdct:nsloc -0.005804   0.005550  -1.046  0.29564
levelld:nsloc    0.007165   0.005867   1.221  0.22196
levelfl:ewloc    0.017977   0.014413   1.247  0.21229
levelhdct:ewloc  0.007606   0.005561   1.368  0.17138
levelld:ewloc   -0.004787   0.005919  -0.809  0.41863
hdct:popEriley  -0.178799   0.089411  -2.000  0.04553 *
hdct:popLf      -0.162516   0.096116  -1.691  0.09087 .
hdct:popNessman -0.315507   0.138823  -2.273  0.02304 *
hdct:popNWLF    -0.108209   0.083110  -1.302  0.19292
hdct:popSPP      0.019942   0.086198   0.231  0.81704
hdct:popStevens -0.129238   0.089129  -1.450  0.14706
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Original predictor variables dropped (aliased)
     hdct

> anova(out8, out6)

Analysis of Deviance Table

Model 1: resp ~ varb + level:(nsloc + ewloc) + hdct * pop - pop
Model 2: resp ~ varb + level:(nsloc + ewloc) + level * pop
  Model Df Model Dev Df Deviance P(>|Chi|)
1       21   2712.54
2       33   2674.70 12    37.84 0.0001632
```

D.2.9 Ninth Model, Like Models One and Eight

We need to make a model with the structure of Model Eight except no populations.

```
> out9 <- aster(resp ~ varb + level:(nsloc + ewloc),
+     pred, fam, varb, id, root, data = redata)
> summary(out9)
```

```
Call:
aster.formula(formula = resp ~ varb + level:(nsloc + ewloc),
    pred = pred, fam = fam, varvar = varb, idvar = id, root = root,
    data = redata)

                 Estimate Std. Error z value Pr(>|z|)
(Intercept)     -1.607690   0.184593  -8.709  < 2e-16 ***
varbfl03        -0.349426   0.268532  -1.301   0.1932
varbfl04        -0.353847   0.244529  -1.447   0.1479
varbhdct02       1.906022   0.257183   7.411 1.25e-13 ***
varbhdct03       1.929506   0.209331   9.217  < 2e-16 ***
varbhdct04       2.436417   0.194176  12.547  < 2e-16 ***
varbld02        -0.866520   0.315991  -2.742   0.0061 **
varbld03         0.905229   0.396379   2.284   0.0224 *
varbld04         4.052121   0.334412  12.117  < 2e-16 ***
levelfl:nsloc    0.070281   0.014705   4.779 1.76e-06 ***
levelhdct:nsloc -0.005378   0.005578  -0.964   0.3350
levelld:nsloc    0.006855   0.005868   1.168   0.2427
levelfl:ewloc    0.017686   0.014441   1.225   0.2207
levelhdct:ewloc  0.007312   0.005542   1.320   0.1870
levelld:ewloc   -0.005027   0.005932  -0.847   0.3968
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> anova(out9, out8, out6)

Analysis of Deviance Table

Model 1: resp ~ varb + level:(nsloc + ewloc)
Model 2: resp ~ varb + level:(nsloc + ewloc) + hdct * pop - pop
Model 3: resp ~ varb + level:(nsloc + ewloc) + level * pop
  Model Df Model Dev Df Deviance P(>|Chi|)
1       15   2728.72
2       21   2712.54  6    16.18      0.01
3       33   2674.70 12    37.84 0.0001632
```

D.2.10 Tenth Model, Between Models Six and Eight

We accidentally created a new tenth model by not understanding the "minus populations" stuff in the formulae for Models Two and Eight.

```
> out10 <- aster(resp ~ varb + level:(nsloc + ewloc) +
+     hdct * pop, pred, fam, varb, id, root, data = redata)
> summary(out10)
```

```
Call:
aster.formula(formula = resp ~ varb + level:(nsloc + ewloc) +
    hdct * pop, pred = pred, fam = fam, varvar = varb, idvar = id,
    root = root, data = redata)

                Estimate Std. Error z value Pr(>|z|)
(Intercept)     -1.726122   0.223462  -7.724 1.12e-14 ***
varbfl03        -0.349106   0.266693  -1.309  0.19053
varbfl04        -0.330903   0.242658  -1.364  0.17268
varbhdct02       2.203913   0.331242   6.653 2.86e-11 ***
varbhdct03       2.226234   0.296256   7.515 5.71e-14 ***
varbhdct04       2.732111   0.285814   9.559  < 2e-16 ***
varbld02        -0.861178   0.316060  -2.725  0.00644 **
varbld03         0.888715   0.396388   2.242  0.02496 *
varbld04         4.008350   0.334436  11.985  < 2e-16 ***
popEriley        0.375414   0.154884   2.424  0.01536 *
popLf            0.355237   0.164870   2.155  0.03119 *
popNessman       0.231430   0.189693   1.220  0.22245
popNWLF          0.012016   0.145094   0.083  0.93400
popSPP           0.185498   0.158948   1.167  0.24319
popStevens      -0.069680   0.153063  -0.455  0.64894
levelfl:nsloc    0.070918   0.014584   4.863 1.16e-06 ***
levelhdct:nsloc -0.006532   0.005504  -1.187  0.23528
levelld:nsloc    0.007901   0.005959   1.326  0.18488
levelfl:ewloc    0.014308   0.014359   0.996  0.31904
levelhdct:ewloc  0.010083   0.005557   1.814  0.06961 .
levelld:ewloc   -0.009182   0.006100  -1.505  0.13231
hdct:popEriley  -0.794844   0.264102  -3.010  0.00262 **
hdct:popLf      -0.744006   0.282716  -2.632  0.00850 **
hdct:popNessman -0.705433   0.355627  -1.984  0.04730 *
hdct:popNWLF    -0.114923   0.245244  -0.469  0.63935
hdct:popSPP     -0.270775   0.262750  -1.031  0.30276
hdct:popStevens  0.003604   0.260305   0.014  0.98895
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Original predictor variables dropped (aliased)
     hdct

> anova(out9, out8, out10, out6)

Analysis of Deviance Table
```

```
Model 1: resp ~ varb + level:(nsloc + ewloc)
Model 2: resp ~ varb + level:(nsloc + ewloc) + hdct * pop - pop
Model 3: resp ~ varb + level:(nsloc + ewloc) + hdct * pop
Model 4: resp ~ varb + level:(nsloc + ewloc) + level * pop
  Model Df Model Dev Df Deviance P(>|Chi|)
1       15   2728.72
2       21   2712.54  6    16.18      0.01
3       27   2684.86  6    27.67 0.0001083
4       33   2674.70  6    10.17      0.12
```

So this says that Model Ten (which only has 12 d. f. for "pop") can be the big model.

D.3 Mean Value Parameters

D.3.1 Unconditional

As argued in Chapter 1, canonical parameters are "meaningless." Only mean value parameters have real world, scientific interpretability.

So in this section we compare predicted values for a typical individual (say zero-zero geometry) in each population under both Models Six and Eight. The functional of mean value parameters we want is *total head count*, since this has the biological interpretation of the best surrogate measure of fitness in this data set. A biologist (at least an evolutionary biologist) is interested in the "ancestor variables" of head count only insofar as they contribute to head count. Two sets of parameter values that "predict" the same expected total head count (over the three years the data were collected) have the same contribution to fitness. So that is the "prediction" (really functional of mean value parameters) we "predict."

To do this we must construct "newdata" for these hypothetical individuals.

```
> newdata <- data.frame(pop = levels(echinacea$pop))
> for (v in vars) newdata[[v]] <- 1
> newdata$root <- 1
> newdata$ewloc <- 0
> newdata$nsloc <- 0
> renewdata <- reshape(newdata, varying = list(vars),
+     direction = "long", timevar = "varb", times = as.factor(vars),
+     v.names = "resp")
> names(redata)

 [1] "pop"   "ewloc" "nsloc" "varb"  "resp"  "id"    "root"
 [8] "hdct"  "level" "year"
```

```
> names(renewdata)

[1] "pop"   "root"  "ewloc" "nsloc" "varb"  "resp"  "id"

> hdct <- grep("hdct", as.character(renewdata$varb))
> hdct <- is.element(seq(along = renewdata$varb), hdct)
> renewdata$hdct <- as.integer(hdct)
> level <- gsub("[0-9]", "", as.character(renewdata$varb))
> renewdata$level <- as.factor(level)
> year <- gsub("[a-z]", "", as.character(renewdata$varb))
> year <- paste("yr", year, sep = "")
> renewdata$year <- as.factor(year)
> setequal(names(redata), names(renewdata))

[1] TRUE
```

We are using bogus data $x_{ij} = 1$ for all i and j because unconditional mean value parameters do not depend on x. We have to have an `x` argument because that's the way the aster package functions work (ultimately due to limitations of the R formula mini-language). So it doesn't matter what we make it. In the following section, the predictions will depend on `x`, but then (as we shall argue), this is the `x` we want.

```
> nind <- nrow(newdata)
> nnode <- length(vars)
> amat <- array(0, c(nind, nnode, nind))
> for (i in 1:nind) amat[i, grep("hdct", vars), i] <- 1
> pout6 <- predict(out6, varvar = varb, idvar = id,
+     root = root, newdata = renewdata, se.fit = TRUE,
+     amat = amat)
> pout8 <- predict(out8, varvar = varb, idvar = id,
+     root = root, newdata = renewdata, se.fit = TRUE,
+     amat = amat)
```

Figure D.1 is produced by the following code

```
> conf.level <- 0.95
> crit <- qnorm((1 + conf.level)/2)

> popnames <- as.character(newdata$pop)
> fit8 <- pout8$fit
> i <- seq(along = popnames)
> ytop <- fit8 + crit * pout8$se.fit
> ybot <- fit8 - crit * pout8$se.fit
```

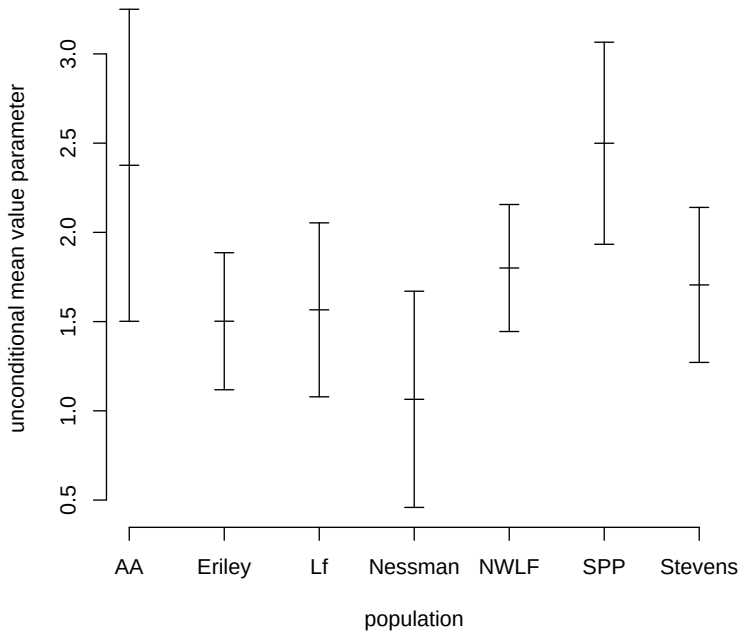

Figure D.1: 95% confidence intervals for unconditional mean value parameter for fitness (sum of head count for all years) at each population for a "typical" individual having position zero-zero and having the parameterization of Model Eight. Tick marks in the middle of the bars are the center (the MLE).

```
> plot(c(i, i), c(ytop, ybot), type = "n", axes = FALSE,
+     xlab = "", ylab = "")
> segments(i, ybot, i, ytop)
> foo <- 0.1
> segments(i - foo, ybot, i + foo, ybot)
> segments(i - foo, ytop, i + foo, ytop)
> segments(i - foo, fit8, i + foo, fit8)
> axis(side = 2)
> title(ylab = "unconditional mean value parameter")
> axis(side = 1, at = i, labels = popnames)
> title(xlab = "population")
```

and appears on p. 50.

Figure D.2 is produced by the following code

```
> fit6 <- pout6$fit
> i <- seq(along = popnames)
> foo <- 0.1
> y8top <- fit8 + crit * pout8$se.fit
> y8bot <- fit8 - crit * pout8$se.fit
> y6top <- fit6 + crit * pout6$se.fit
> y6bot <- fit6 - crit * pout6$se.fit
> plot(c(i - 1.5 * foo, i - 1.5 * foo, i + 1.5 * foo,
+     i + 1.5 * foo), c(y8top, y8bot, y6top, y6bot),
+     type = "n", axes = FALSE, xlab = "", ylab = "")
> segments(i - 1.5 * foo, y8bot, i - 1.5 * foo, y8top)
> segments(i - 2.5 * foo, y8bot, i - 0.5 * foo, y8bot)
> segments(i - 2.5 * foo, y8top, i - 0.5 * foo, y8top)
> segments(i - 2.5 * foo, fit8, i - 0.5 * foo, fit8)
> segments(i + 1.5 * foo, y6bot, i + 1.5 * foo, y6top,
+     lty = 2)
> segments(i + 2.5 * foo, y6bot, i + 0.5 * foo, y6bot)
> segments(i + 2.5 * foo, y6top, i + 0.5 * foo, y6top)
> segments(i + 2.5 * foo, fit6, i + 0.5 * foo, fit6)
> axis(side = 2)
> title(ylab = "unconditional mean value parameter")
> axis(side = 1, at = i, labels = popnames)
> title(xlab = "population")
```

and appears on p. 52.

D.3.2 Conditional

This section is very incomplete. We don't redo everything using conditional models. That's not the point. We only want to show that conditional models and conditional mean value parameters just don't do the same thing as unconditional models (which is obvious, but some people like examples, and in any case, this gives us an opportunity to show some options of aster model fitting).

Conditional Models

Let us redo Figure D.2 based on conditional models with the same model matrices (a dumb idea, since the meaning of the models is entirely different despite the similarity in algebra, but we want to hammer the point home).

```
> cout6 <- aster(resp ~ varb + level:(nsloc + ewloc) +
+     level * pop, pred, fam, varb, id, root, data = redata,
+     type = "conditional")
```

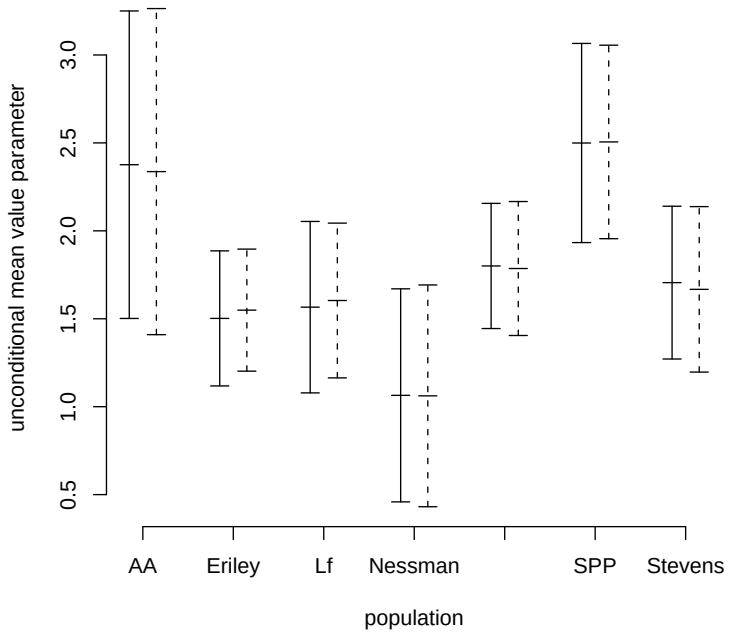

Figure D.2: 95% confidence intervals for unconditional mean value parameter for fitness (sum of head count for all years) at each population for a "typical" individual having position zero-zero and having the parameterization of Model Eight (solid bar) or Model Six (dashed bar). Tick marks in the middle of the bars are the center (the MLE).

```
> cout8 <- aster(resp ~ varb + level:(nsloc + ewloc) +
+     hdct * pop - pop, pred, fam, varb, id, root,
+     data = redata, type = "conditional")
> pcout6 <- predict(cout6, varvar = varb, idvar = id,
+     root = root, newdata = renewdata, se.fit = TRUE,
+     amat = amat)
> pcout8 <- predict(cout8, varvar = varb, idvar = id,
+     root = root, newdata = renewdata, se.fit = TRUE,
+     amat = amat)
```

Note that these are exactly like the analogous statements making the analogous objects without the "c" in their names except for the `type = "conditional"` arguments in the two `aster` function calls. Then we make Figure D.3 just like Figure D.2 except for using `pcout6` and `pcout8` instead of `pout6` and `pout8`. It appears on p. 54.

Note the huge difference between Figure D.2 and Figure D.3. The same model matrices are used in both cases. The linear predictor satisfies $\boldsymbol{\eta} = \mathbf{M}\boldsymbol{\beta}$, but in one case (Figure D.2) the linear predictor is the unconditional canonical parameter ($\boldsymbol{\eta} = \boldsymbol{\varphi}$) and in the other case (Figure D.3) the linear predictor is the conditional canonical parameter ($\boldsymbol{\eta} = \boldsymbol{\theta}$). In one case (Figure D.2) the predictions of a linear functional of the unconditional mean value parameter ($\boldsymbol{\tau}$) are nearly the same for the two models and in the other case (Figure D.3) the predictions of the same linear functional of $\boldsymbol{\tau}$ are wildly different.

Conclusion: conditional models and unconditional models are different. That's the whole point. That's why unconditional models were invented, because conditional models can't be made to do the same thing.

More on Conditional Models

Let us redo the analysis of deviance table in Section D.2.10 based on conditional models with the same model matrices (again we reiterate, this is a very dumb idea, since the meaning of the models is entirely different despite the similarity in algebra, but we want to hammer the point home).

```
> cout9 <- aster(resp ~ varb + level:(nsloc + ewloc),
+     pred, fam, varb, id, root, data = redata, type = "cond")
> cout10 <- aster(resp ~ varb + level:(nsloc + ewloc) +
+     hdct * pop, pred, fam, varb, id, root, data = redata,
+     type = "cond")
> anova(cout9, cout8, cout10, cout6)

Analysis of Deviance Table

Model 1: resp ~ varb + level:(nsloc + ewloc)
```

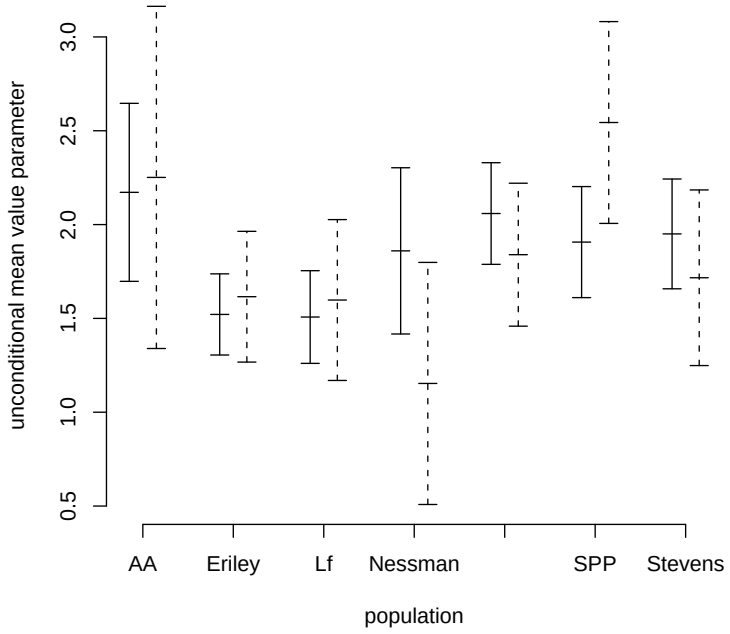

Figure D.3: 95% confidence intervals for unconditional mean value parameter for fitness (sum of head count for all years) at each population for a "typical" individual having position zero-zero and having the parameterization of Model Eight (solid bar) or Model Six (dashed bar). Tick marks in the middle of the bars are the center (the MLE). The difference between this figure and Figure D.2 is that the models fitted are *conditional* rather than *unconditional*.

```
Model 2: resp ~ varb + level:(nsloc + ewloc) + hdct * pop - pop
Model 3: resp ~ varb + level:(nsloc + ewloc) + hdct * pop
Model 4: resp ~ varb + level:(nsloc + ewloc) + level * pop
  Model Df Model Dev Df Deviance P(>|Chi|)
1       15   2720.97
2       21   2693.09  6    27.88 9.916e-05
3       27   2678.37  6    14.72      0.02
4       33   2660.97  6    17.40      0.01
```

It is hard to know what lesson to draw from this. Presumably since all three "large" models fit about equally well, none of them fit as well as the corresponding unconditional models (we see from the analysis in the preceding section). But since conditional and unconditional models are not nested, we cannot use standard likelihood ratio test methodology to test this. (So we have no clear lesson here, but leave it in to not hide anything).

Conditional Parameter

Let us redo Figure D.2 now not changing the model (we still use the fits `out6` and `out8`) but changing the thingummy we "predict". In Figure D.2 we "predict" a linear functional $\mathbf{A}'\boldsymbol{\tau}$ of the unconditional mean value parameter (the sum of three components of $\boldsymbol{\tau}$, those for flower head count). In Figure D.4 we predict the same linear functional $\mathbf{A}'\boldsymbol{\xi}$ of the *conditional* mean value parameter $\boldsymbol{\xi}$.

```
> pxout6 <- predict(out6, varvar = varb, idvar = id,
+     root = root, newdata = renewdata, se.fit = TRUE,
+     amat = amat, model.type = "conditional")
> pxout8 <- predict(out8, varvar = varb, idvar = id,
+     root = root, newdata = renewdata, se.fit = TRUE,
+     amat = amat, model.type = "conditional")
```

Note that these are exactly like the analogous statements making the analogous objects without the "x" in their names except for the `model.type = "conditional"` arguments in the two `predict` function calls. Then we make Figure D.4 just like Figure D.2 except for using `pxout6` and `pxout8` instead of `pout6` and `pout8`. It appears on p. 56.

Note the huge difference between Figure D.2 and Figure D.4. The same models are used in both cases but in one case (Figure D.2) we "predict" a linear functional $\mathbf{A}'\boldsymbol{\tau}$ of the unconditional mean value parameter ($\boldsymbol{\tau}$) and in the other case (Figure D.4) we "predict" the *same* linear functional $\mathbf{A}'\boldsymbol{\xi}$ of the conditional mean value parameter ($\boldsymbol{\xi}$).

Conclusion: conditional expectations and unconditional expectations are different. (Duh!) The two sorts of predictions can't be made to do the same thing.

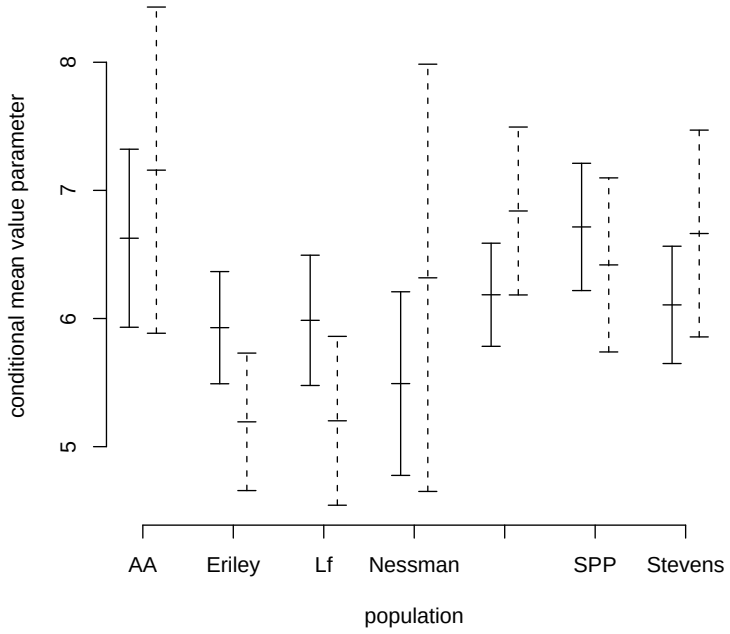

Figure D.4: 95% confidence intervals for conditional mean value parameter for fitness (sum of head count for all years) at each population for a "typical" individual having position zero-zero and having the parameterization of Model Eight (solid bar) or Model Six (dashed bar). Tick marks in the middle of the bars are the center (the MLE). The difference between this figure and Figure D.2 is that the parameters "predicted" are *conditional* rather than *unconditional*.

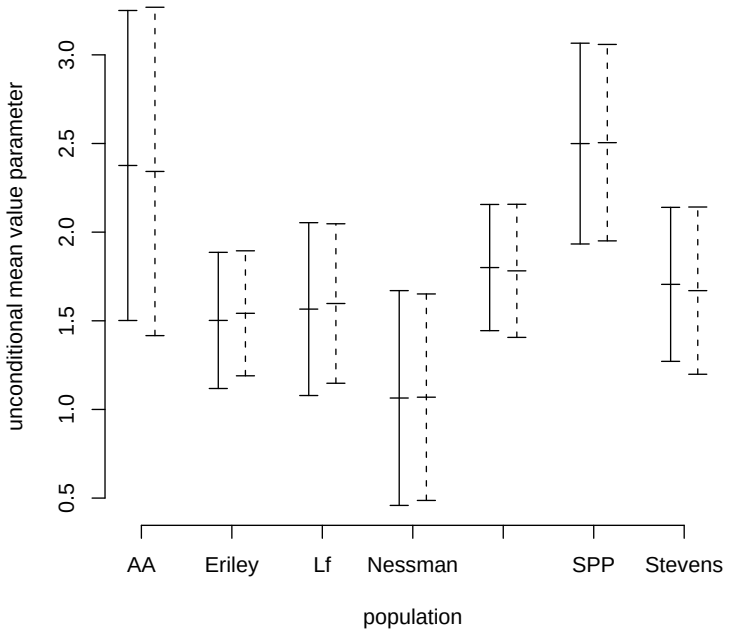

Figure D.5: 95% confidence intervals for unconditional mean value parameter for fitness (sum of head count for all years) at each population for a "typical" individual having position zero-zero and having the parameterization of Model Eight (solid bar) or Model Ten (dashed bar). Tick marks in the middle of the bars are the center (the MLE).

D.4 Plot for the Paper

We redo Figure D.2 changing the models compared to Model 8 and Model 10 (fits in `out8` and `out10`).

```
> pout10 <- predict(out10, varvar = varb, idvar = id,
+     root = root, newdata = renewdata, se.fit = TRUE,
+     amat = amat)
```

It appears on p. 57.

Appendix E

Canonical Parameter Spaces of Aster Models

E.1 Basic Exponential Family Theory

The *Laplace transform* of a positive measure λ on $\mathbb{R}^J$ is (Barndorff-Nielsen, 1978, Chapter 7) the function $c : \mathbb{R}^J \to (0, \infty]$ defined by

$$c(\boldsymbol{\varphi}) = \int e^{\langle \mathbf{x}, \boldsymbol{\varphi} \rangle} \lambda(d\mathbf{x}).$$

(note that the value $+\infty$ is allowed so the function is defined for all $\boldsymbol{\varphi}$). A log Laplace transform is both convex and lower semicontinuous (Barndorff-Nielsen, 1978, Theorem 7.1). This implies that

$$\Phi = \{ \boldsymbol{\varphi} \in \mathbb{R}^J : c(\boldsymbol{\varphi}) < \infty \}$$

is a convex set.

The full standard exponential family generated by λ is the family

$$\mathcal{P} = \{ P_{\boldsymbol{\varphi}} : \boldsymbol{\varphi} \in \Phi \}$$

where $P_{\boldsymbol{\varphi}}$ is the distribution having density with respect to λ defined by

$$f_{\boldsymbol{\varphi}}(x) = \frac{1}{c(\boldsymbol{\varphi})} e^{\langle \mathbf{x}, \boldsymbol{\varphi} \rangle}, \qquad \boldsymbol{\varphi} \in \Phi$$

(Barndorff-Nielsen, 1978, Chapter 8). The moment generating function of $P_{\boldsymbol{\varphi}}$ is defined by

$$M_{\boldsymbol{\varphi}}(t) = \int e^{\langle \mathbf{x}, t \rangle} P_{\boldsymbol{\varphi}}(d\mathbf{x}) = \frac{c(\boldsymbol{\varphi} + t)}{c(\boldsymbol{\varphi})}$$

From this we see that a Laplace transform is just like a moment generating function, except for a general positive measure λ instead of for a probability measure $P_{\boldsymbol{\varphi}}$.

The *cumulant generating function* is the log of the moment generating function. Its derivatives evaluated at $t = 0$ are the *cumulants* of the distribution (the first two are mean and variance). Since the derivatives of $\log M_\varphi$ evaluated at $t = 0$ are the same as the derivatives of $\psi = \log c$ evaluated at $\boldsymbol{\varphi}$, we call ψ the *cumulant function* of the family.

It is sometimes convenient to choose for the dominating measure of the family (λ in our original notation) one of the distributions in the family, a $P_{\boldsymbol{\varphi}^*}$ for some $\boldsymbol{\varphi}^* \in \Phi$. We need to see what that does to our original formulation with λ as the dominating measure.

The density of $P_{\boldsymbol{\varphi}}$ with respect to $P_{\boldsymbol{\varphi}^*}$ is just the ratio of their densities with respect to λ

$$g_{\boldsymbol{\varphi}}(x) = \frac{f_{\boldsymbol{\varphi}}(x)}{f_{\boldsymbol{\varphi}^*}(x)} = \frac{c(\boldsymbol{\varphi}^*)}{c(\boldsymbol{\varphi})} e^{\langle x, \boldsymbol{\varphi} - \boldsymbol{\varphi}^* \rangle} \tag{E.1}$$

and we see that the Laplace transform for this "new" family is $c(\boldsymbol{\varphi})/c(\boldsymbol{\varphi}^*)$ and the cumulant function is $\psi(\boldsymbol{\varphi}) - \psi(\boldsymbol{\varphi}^*)$.

E.2 Exponential Family Theory Applied to Aster Models

E.2.1 Cumulant Function and Full Canonical Parameter Space

This appendix investigates what happens when the canonical parameter spaces of an aster model are not all of $\mathbb{R}^d$. The full *conditional* canonical parameter space is a Cartesian product

$$\Theta = \prod_{j \in J} \Theta_j$$

where Θ_j is the full canonical parameter space of the conditional one-parameter exponential family model at the j-th node (which has cumulant function ψ_j), the set

$$\Theta_j = \{\, \theta \in \mathbb{R} : \psi_j(\theta) < \infty \,\}.$$

Let $P_{\theta,j}$ be the conditional probability measure of the j-th family. So the joint distribution is

$$P_{\boldsymbol{\theta}}(d\mathbf{x}) = \prod_{j \in J} P_{\theta_j, j}(dx_j \mid x_{p(j)}).$$

Fix $\boldsymbol{\theta}^* \in \Theta$ and let $\boldsymbol{\varphi}^*$ be the corresponding unconditional canonical parameter found by applying the map (1.5) to $\boldsymbol{\theta}^*$. Write $\boldsymbol{\varphi} = \boldsymbol{\varphi}^* + \boldsymbol{\delta}$ for a

general unconditional canonical parameter. The Laplace transform (which in this case is also a moment generating function) of $P_{\boldsymbol{\varphi}^*}$ is

$$\int e^{\langle \mathbf{x}, \boldsymbol{\delta} \rangle} P_{\boldsymbol{\varphi}^*}(d\mathbf{x}) = \exp\bigl(\psi(\boldsymbol{\varphi}^* + \boldsymbol{\delta}) - \psi(\boldsymbol{\varphi}^*)\bigr), \tag{E.2}$$

which agrees with the analysis in (E.1). Now we define the full canonical parameter space

$$\Phi = \{\, \boldsymbol{\varphi} \in \mathbb{R}^J : \psi(\boldsymbol{\varphi}) < \infty \,\}$$

(we are using the original parameter $\boldsymbol{\varphi}$ instead of the "new" parameter $\boldsymbol{\delta}$). It stands to reason that Φ is just the set of points obtained by mapping Θ through the change of parameter defined by (1.5). The whole point of this appendix is to show that what seems obvious actually is obvious.

E.2.2 Leaf Nodes

For j a leaf node, the only part of the integral in (E.2) involving x_j is

$$\int e^{x_j \delta_j} P_{\theta_j^*, j}(dx_j \mid x_{p(j)}).$$

Now from the uniparameter case of the exponential family theory embodied in (E.2) we see that

$$\int e^{x_j \delta_j} P_{\theta_j^*, j}(dx_j \mid x_{p(j)} = 1) = \exp\bigl(\psi_j(\theta_j^* + \delta_j) - \psi_j(\theta_j^*)\bigr),$$

from which it follows from the multiplication rule for moment generating functions and the structure of aster models that

$$\int e^{x_j \delta_j} P_{\theta_j^*, j}(dx_j \mid x_{p(j)}) = \exp\bigl(x_{p(j)}[\psi_j(\theta_j^* + \delta_j) - \psi_j(\theta_j^*)]\bigr). \tag{E.3}$$

This is finite if and only if

$$\varphi_j = \theta_j = \theta_j^* + \delta_j = \varphi_j^* + \delta_j \in \Theta_j$$

in short if

$$\varphi_j = \theta_j \in \Theta_j.$$

So that is the finiteness condition for leaf nodes. The part of the integral (E.2) that pertains to θ_j and φ_j has the "obvious" condition for being finite.

E.2.3 Non-Leaf Nodes

Now consider a non-leaf node j whose children are all leaf nodes. After integrating out the x_m for all child nodes m, as above, the only part of the integral in (E.2) that contains x_j is

$$\int \exp\left(x_j \left[\delta_j + \sum_{m \in S(j)} \big(\psi_m(\theta_m) - \psi_m(\theta_m^*) \big) \right] \right) P_{\theta_j^*, j}(dx_j \mid x_{p(j)}) \tag{E.4a}$$

where we have written $\theta_m = \varphi_m = \theta_m^* + \delta_m$ for the m, all of which are leaf nodes. We also write $\varphi_j = \varphi_j^* + \delta_j$ and note that from (1.5) we have

$$\varphi_j^* + \delta_j + \sum_{m \in S(j)} \psi_m(\theta_m) = \varphi_j + \sum_{m \in S(j)} \psi_m(\theta_m) = \theta_j$$

and similarly

$$\varphi_j^* + \sum_{m \in S(j)} \psi_m(\theta_m^*) = \theta_j^*$$

so the term in large square brackets in (E.4a) is just $\theta_j^* - \theta_j$. Hence (E.4a) is

$$\int \exp\big(x_j \left[\theta_j^* - \theta_j \right] \big) P_{\theta_j^*, j}(dx_j \mid x_{p(j)}) = \exp\big(x_{p(j)} [\psi_j(\theta_j) - \psi_j(\theta_j^*)] \big). \tag{E.4b}$$

and we see that (E.4a) and (E.4b) say exactly the same thing as (E.3).

However, the implication about finiteness of bits of (E.2) is a bit different than the case for leaf nodes. Now we have that

$$\theta_j \in \Theta_j$$

is the finiteness condition for the term on the right hand side of (E.4b) (no surprise there) and that translates to the following about φ_j. Since

$$\theta_j = \varphi_j + \sum_{m \in S(j)} \psi_m(\theta_m)$$

the finiteness condition for this section is

$$\varphi_j \in \Theta_j - \sum_{m \in S(j)} \psi_m(\theta_m). \tag{E.5}$$

E.2.4 All Nodes by Mathematical Induction

Now assume (the induction hypothesis) that

$$\int \exp\left(x_j \left[\delta_j + \sum_{m \in S(j)} \big(\psi_m(\theta_m) - \psi_m(\theta_m^*) \big) \right] \right) P_{\theta_j^*, j}(dx_j \mid x_{p(j)}) = \exp\big(x_{p(j)} [\psi_j(\theta_j) - \psi_j(\theta_j^*)] \big). \tag{E.6}$$

As we have seen, this holds for all of the nodes we have examined so far. But then we see that by the argument in the preceding section that if (E.6) holds for all children (successors) of a node, then it holds for the node itself.

Thus (E.5) is the finiteness condition for all nodes, including leaf nodes for which $S(j)$ is the empty set.

Hence Φ is indeed the image of Θ under the map (1.5). And the argument did turn out to be "obvious".

E.2.5 Convexity

The only non-obvious fact in this whole appendix is that hence Φ must be a convex set (like all full canonical parameter spaces). Moreover, since the map (1.5) is a diffeomorphism (both it and its inverse are differentiable) between the interiors of Θ and Φ, it follows that Θ and Φ are both open sets or both not open.

E.2.6 Regularity

The important special case where the full canonical parameter space is an open set is called *regular* (the family is a *regular* exponential family) (Barndorff-Nielsen, 1978, p. 116). They are the most well-behaved with respect to maximum likelihood (no boundaries of the parameter space to worry about).

Since Θ is open if and only if each Θ_j is, we see that the full flat exponential family (with the unconditional canonical parameterization) is regular if and only if each one-parameter conditional family is regular.

E.2.7 Steepness

A full exponential family is *steep* (Barndorff-Nielsen, 1978, p. 117) if given $\boldsymbol{\varphi}_i$ in the interior of Φ and $\boldsymbol{\varphi}_b$ on the boundary of Φ, then

$$\langle \nabla\psi\big(t\boldsymbol{\varphi}_i + (1-t)\boldsymbol{\varphi}_b\big), \boldsymbol{\varphi}_b - \boldsymbol{\varphi}_i \rangle \to \infty, \qquad \text{as } t \downarrow 0.$$

Clearly this carries over to any flat subfamily (formed by intersecting the full canonical parameter space Φ with an affine subspace). The subfamily is steep if the full family is. Thus we concentrate on the full (FEF) family only.

An equivalent (if and only if) condition for steepness is that the MLE map $\mathbf{x} \mapsto \hat{\boldsymbol{\varphi}}(\mathbf{x})$ is one-to-one and is found by solving the "likelihood equations" which have "observed equals expected" form, as in equation (1.15) and the preceding unnumbered equation (Barndorff-Nielsen, 1978, Theorem 9.14, Corollary 9.6 and their surrounding discussion).

Yet another equivalent (if and only if) condition for steepness is that the mean value parameterization map, $\nabla\psi : \boldsymbol{\varphi} \mapsto \boldsymbol{\tau}$ in aster model notation,

maps the interior of the full canonical parameter space Φ onto the interior of the closed convex support of the canonical statistic (Barndorff-Nielsen, 1978, pp. 117, 142, and 152).

Let a_j and b_j be the endpoints (possibly infinite) of the full mean-value parameter space of the one-parameter exponential family for the j-th node. The endpoints themselves may or may not be in the mean-value parameter space

$$\{ \psi_j'(\theta) : \theta \in \Theta_j \}$$

but the closed convex support of the j-th one-parameter exponential family is the closed interval $\mathbb{R} \cap [a_j, b_j]$ if this j-th family is steep (the point of the $\mathbb{R} \cap$ is to omit $-\infty$ and $+\infty$ if either $a_j = -\infty$ or $b_j = +\infty$).

Clearly from equation (1.12) the closed convex support of X_j in the FEF is the closed interval

$$\mathbb{R} \cap \left[X_{if(j)} \prod_{\substack{m \in J \\ j \preceq m \prec f(j)}} a_m, X_{if(j)} \prod_{\substack{m \in J \\ j \preceq m \prec f(j)}} b_m \right]. \tag{E.7}$$

It is clear also that as $\boldsymbol{\theta}$ runs over the interior of Θ the mean value parameter $\boldsymbol{\tau}(\boldsymbol{\theta})$ runs over the interior of (E.7). Hence if each of the one-parameter exponential families is steep, then the FEF of an aster model is steep.

E.3 Conclusions

This whole appendix is (in hindsight) "obvious." What we have learned is that what we thought was obvious is indeed obvious and there is no problem with maximum likelihood in an aster model if each of the one-parameter exponential families is regular or steep. Moreover, we have learned that the canonical parameter space of the FEF is convex (like every full canonical parameter space of every full exponential family) something that is "obvious" only from exponential family theory and not from looking at the definition of the map (1.5) between conditional and unconditional canonical parameterizations.