

**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



## Summary

We present a new class of statistical models, designed for life history analysis of plants and animals, that allow joint analysis of data on survival and reproduction over multiple years, allow for variables having different probability distributions, and correctly account for the dependence of variables on earlier variables. We illustrate their utility with an analysis of data taken from an experimental study of *Echinacea angustifolia* sampled from remnant prairie populations in western Minnesota. These models generalise both generalised linear models and survival analysis. The joint distribution is factorised as a product of conditional distributions, each an exponential family with the conditioning variable being the sample size of the conditional distribution. The model may be heterogeneous, each conditional distribution being from a different exponential family. We show that the joint distribution is from 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 the Comprehensive R Archive Network, 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 appendices on technical subjects.

- Appendix A (p. 16) 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. 21) 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. 26) 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. 28) gives details of the data analysis that is briefly described in the draft paper.
- Appendix E (p. 56) 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).

Note added for this git repository. The text file of the draft paper that was used to make this technical report seems to have been lost. (We were not using version control when the paper was written.) So this document does not reproduce the technical report exactly (the text here is the text of the published version of the paper). The PDF of the original technical is found at <https://www.stat.umn.edu/geyer/aster/tr644.pdf>.

# Chapter 1

## Draft Paper

### 1.1 Introduction

This article introduces a class of statistical models we call ‘aster models’. They were invented for life history analysis of plants and animals and are best introduced by an example about perennial plants observed over several years. For each individual planted, at each census, we record whether or not it is alive, whether or not it flowers, and its number of flower heads. These data are complicated, especially when recorded for several years, but simple conditional models may suffice. We consider mortality status, dead or alive, to be Bernoulli given the preceding mortality status. Similarly, flowering status given mortality status is also Bernoulli. Given flowering, the number of flower heads may have a zero-truncated Poisson distribution (Martin, et al., 2005). Figure 1.1 shows the graphical model for a single individual. This aster model generalises both discrete time Cox regression (Cox, 1972; Breslow, 1972, 1974) and generalised linear models (McCullagh & Nelder, 1989). Aster models apply to any similar conditional modelling. We could, for example, add other variables, such as seed count modelled conditional on flower head count.

A simultaneous analysis that models the joint distribution of all the variables in a life history analysis can answer questions that cannot be addressed through separate analyses of each variable conditional on the others. Joint analysis also deals with structural zeros in the data; for example, a dead individual remains dead and cannot flower, so in Fig. 1.1 any arrow that leads from a variable that is zero to another variable implies that the other variable must also be zero. Such zeros present intractable missing data problems in separate analyses of individual variables. Aster models have no problem with structural zeros; likelihood inference automatically handles them correctly.

Aster models are simple graphical models (Lauritzen, 1996, § 3.2.3) in which the joint density is a product of conditionals as in equation (1.1) be-

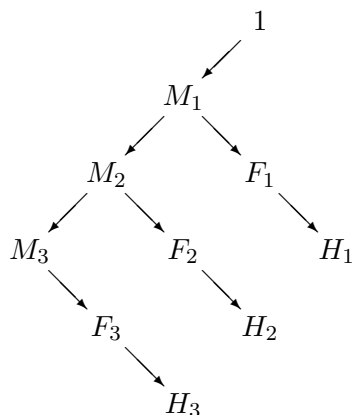


Figure 1.1: Graph for *Echinacea* aster data. Arrows go from parent nodes to child nodes. Nodes are labelled by their associated variables. The only root node is associated with the constant variable 1.  $M_j$  is the mortality status in year  $2001 + j$ .  $F_j$  is the flowering status in year  $2001 + j$ .  $H_j$  is the flower head count in year  $2001 + j$ . The  $M_j$  and  $F_j$  are Bernoulli conditional on their parent variables being one, and zero otherwise. The  $H_j$  are zero-truncated Poisson conditional on their parent variables being one, and zero otherwise.

low. No knowledge of graphical model theory is needed to understand aster models. One innovative aspect of aster models is the interplay between two parameterisations described in §§ 1.2.2 and 1.2.3 below. The ‘conditional canonical parameterisation’ arises when each conditional distribution in the product is an exponential family and we use the canonical parameterisation for each. The ‘unconditional canonical parameterisation’ arises from observing that the joint model is a full flat exponential family (Barndorff-Nielsen, 1978, Ch. 8) and using the canonical parameters for that family, defined by equation (1.5) below.

## 1.2 Aster Models

### 1.2.1 Factorisation and graphical model

Variables in an aster model are denoted by  $X_j$ , where  $j$  runs over the nodes of a graph. A general aster model is a chain graph model (Lauritzen, 1996, pp. 7, 53) having both arrows, corresponding to directed edges, and lines, corresponding to undirected edges. Figure 1.1 is special, having only arrows. Arrows go from parent to child, lines between neighbours. Nodes that are not children are called root nodes. Those that are not parents are called terminal nodes.

Let  $F$  and  $J$  denote root and non-root nodes. Aster models have very

special chain graph structure determined by a partition  $\mathcal{G}$  of  $J$  and a function  $p : \mathcal{G} \rightarrow J \cup F$ . For each  $G \in \mathcal{G}$  there is an arrow from  $p(G)$  to each element of  $G$  and a line between each pair of elements of  $G$ . For any set  $S$ , let  $\mathbf{X}_S$  denote the vector whose components are  $X_j$ ,  $j \in S$ . The graph determines a factorisation

$$\Pr(\mathbf{X}_J | \mathbf{X}_F) = \prod_{G \in \mathcal{G}} \Pr\{\mathbf{X}_G | X_{p(G)}\}; \quad (1.1)$$

compare equation 3.23 in Lauritzen (1996).

Elements of  $\mathcal{G}$  are called chain components because they are connectivity components of the chain graph (Lauritzen, 1996, pp. 6–7). Since Fig. 1.1 has no undirected edge, each node is a chain component by itself. Allowing nontrivial chain components allows the elements of  $\mathbf{X}_G$  to be conditionally dependent given  $X_{p(G)}$  with merely notational changes to the theory. In our example in § 1.5 the graph consists of many copies of Fig. 1.1, one for each individual plant. Individuals have no explicit representation. For any set  $S$ , let  $p^{-1}(S)$  denote the set of  $G$  such that  $p(G) \in S$ . Then each subgraph consisting of one  $G \in p^{-1}(F)$ , its descendants, children, children of children, etc., and arrows and lines connecting them, corresponds to one individual. If we make each such  $G$  have a distinct root element  $p(G)$ , then the set of descendants of each root node corresponds to one individual. Although all individuals in our example have the same subgraph, this is not required.

## 1.2.2 Conditional exponential families

We take each of the conditional distributions in (1.1) to be an exponential family having canonical statistic  $\mathbf{X}_G$  that is the sum of  $X_{p(G)}$  independent and identically distributed random vectors, possibly a different such family for each  $G$ . Conditionally,  $X_{p(G)} = 0$  implies that  $\mathbf{X}_G = \mathbf{0}_G$  almost surely. If  $j \neq p(G)$  for any  $G$ , then the values of  $X_j$  are unrestricted. If the distribution of  $\mathbf{X}_G$  given  $X_{p(G)}$  is infinitely divisible, such as Poisson or normal, for each  $G \in p^{-1}(\{j\})$ , then  $X_j$  must be nonnegative and real-valued. Otherwise,  $X_j$  must be nonnegative and integer-valued.

The loglikelihood for the whole family has the form

$$\sum_{G \in \mathcal{G}} \left[ \sum_{j \in G} X_j \theta_j - X_{p(G)} \psi_G(\boldsymbol{\theta}_G) \right] = \sum_{j \in J} X_j \theta_j - \sum_{G \in \mathcal{G}} X_{p(G)} \psi_G(\boldsymbol{\theta}_G), \quad (1.2)$$

where  $\boldsymbol{\theta}_G$  is the canonical parameter vector for the  $G$ th conditional family, having components  $\theta_j$ ,  $j \in G$ , and  $\psi_G$  is the cumulant function for that family (Barndorff-Nielsen, 1978, pp. 105, 139, 150) that satisfies

$$E_{\boldsymbol{\theta}_G} \{\mathbf{X}_G | X_{p(G)}\} = X_{p(G)} \nabla \psi_G(\boldsymbol{\theta}_G) \quad (1.3)$$

$$\text{var}_{\boldsymbol{\theta}_G} \{\mathbf{X}_G | X_{p(G)}\} = X_{p(G)} \nabla^2 \psi_G(\boldsymbol{\theta}_G), \quad (1.4)$$

where  $\text{var}_{\boldsymbol{\theta}}(\mathbf{X})$  is the variance-covariance matrix of  $\mathbf{X}$  and  $\nabla^2\psi(\boldsymbol{\theta})$  is the matrix of second partial derivatives of  $\psi$  (Barndorff-Nielsen, 1978, p. 150).

### 1.2.3 Unconditional exponential families

Collecting terms with the same  $X_j$  in (1.2), we obtain

$$\sum_{j \in J} X_j \left[ \theta_j - \sum_{G \in p^{-1}(\{j\})} \psi_G(\boldsymbol{\theta}_G) \right] - \sum_{G \in p^{-1}(F)} X_{p(G)} \psi_G(\boldsymbol{\theta}_G)$$

and see that

$$\varphi_j = \theta_j - \sum_{G \in p^{-1}(\{j\})} \psi_G(\boldsymbol{\theta}_G), \quad j \in J, \quad (1.5)$$

are the canonical parameters of an unconditional exponential family with canonical statistics  $X_j$ . We now write  $\mathbf{X}$  instead of  $\mathbf{X}_J$ ,  $\boldsymbol{\varphi}$  instead of  $\boldsymbol{\varphi}_J$ , and so forth, and let  $\langle \mathbf{X}, \boldsymbol{\varphi} \rangle$  denote the inner product  $\sum_j X_j \varphi_j$ . Then we can write the loglikelihood of this unconditional family as

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

where the cumulant function of this family is

$$\psi(\boldsymbol{\varphi}) = \sum_{G \in p^{-1}(F)} X_{p(G)} \psi_G(\boldsymbol{\theta}_G). \quad (1.7)$$

All of the  $X_{p(G)}$  in (1.7) are at root nodes, and hence are nonrandom, so that  $\psi$  is a deterministic function. Also the right-hand side of (1.7) is a function of  $\boldsymbol{\varphi}$  by the logic of exponential families (Barndorff-Nielsen, 1978, pp. 105 ff.).

The system of equations (1.5) can be solved for the  $\theta_j$  in terms of the  $\varphi_j$  in one pass through the equations in any order that finds  $\theta_j$  for children before parents. Thus (1.5) determines an invertible change of parameter.

### 1.2.4 Canonical affine models

One of the desirable aspects of exponential family canonical parameter affine models defined by reparameterisation of the form

$$\boldsymbol{\varphi} = \mathbf{a} + \mathbf{M}\boldsymbol{\beta}, \quad (1.8)$$

where  $\mathbf{a}$  is a known vector, called the origin, and  $\mathbf{M}$  is a known matrix, called the model matrix, is that, because  $\langle \mathbf{X}, \mathbf{M}\boldsymbol{\beta} \rangle = \langle \mathbf{M}^T \mathbf{X}, \boldsymbol{\beta} \rangle$ , the result is a new exponential family with canonical statistic

$$\mathbf{Y} = \mathbf{M}^T \mathbf{X} \quad (1.9)$$



and canonical parameter  $\boldsymbol{\beta}$ . The dimension of this new family will be the dimension of  $\boldsymbol{\beta}$ , if  $\mathbf{M}$  has full rank.

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. A dimension reduction to a low-dimensional sufficient statistic like (1.9) does not occur when the conditional canonical parameters  $\boldsymbol{\theta}$  are modelled affinely, and this suggests that affine models for the unconditional parameterisation may be scientifically more interesting despite their more complicated structure.

### 1.2.5 Mean-value parameters

Conditional mean values

$$\xi_j = E_{\boldsymbol{\theta}_G}\{X_j|X_{p(G)}\} = X_{p(G)} \frac{\partial \psi_G(\boldsymbol{\theta}_G)}{\partial \theta_j}, \quad j \in G, \quad (1.10)$$

are not parameters because they contain random data  $X_{p(G)}$ , although they play the role of mean-value parameters when we condition on  $X_{p(G)}$ , treating it as constant. By standard exponential family theory (Barndorff-Nielsen, 1978, p. 121),  $\nabla \psi_G$  is an invertible change of parameter.

Unconditional mean-value parameters are the unconditional expectations

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

By standard theory,  $\nabla \psi : \boldsymbol{\varphi} \mapsto \boldsymbol{\tau}$  is an invertible change of parameter. The unconditional expectation in (1.11) can be calculated using the iterated expectation theorem

$$E_{\boldsymbol{\varphi}}\{X_j\} = E_{\boldsymbol{\varphi}}\{X_{p(G)}\} \frac{\partial \psi_G(\boldsymbol{\theta}_G)}{\partial \theta_j}, \quad j \in G, \quad (1.12)$$

where  $\boldsymbol{\theta}$  is determined from  $\boldsymbol{\varphi}$  by solving (1.5). The system of equations (1.12) can produce the  $\tau_j$  in one pass through the equations in any order that finds  $\tau_j$  for parents before children.

## 1.3 Likelihood Inference

### 1.3.1 Conditional models

The score  $\nabla l(\boldsymbol{\theta})$  for conditional canonical parameters is particularly simple, having components

$$\frac{\partial l(\boldsymbol{\theta})}{\partial \theta_j} = X_j - X_{p(G)} \frac{\partial \psi_G(\boldsymbol{\theta}_G)}{\partial \theta_j}, \quad j \in G, \quad (1.13)$$

and, if these parameters are modelled affinely as in (1.8) but with  $\varphi$  replaced by  $\boldsymbol{\theta}$ , then

$$\nabla l(\boldsymbol{\beta}) = \nabla l(\boldsymbol{\theta})\mathbf{M}. \quad (1.14)$$

The observed Fisher information matrix for  $\boldsymbol{\theta}$ , which is the matrix  $-\nabla^2 l(\boldsymbol{\theta})$ , is block diagonal with

$$-\frac{\partial^2 l(\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} = X_{p(G)} \frac{\partial^2 \psi_G(\boldsymbol{\theta}_G)}{\partial \theta_i \partial \theta_j}, \quad i, j \in G, \quad (1.15)$$

the only nonzero entries. The expected Fisher information matrix for  $\boldsymbol{\theta}$  is the unconditional expectation of the observed Fisher information matrix, calculated using (1.15) and (1.12). If  $I(\boldsymbol{\theta})$  denotes either the observed or the expected Fisher information matrix for  $\boldsymbol{\theta}$  and similarly for other parameters, then

$$I(\boldsymbol{\beta}) = \mathbf{M}^T I(\boldsymbol{\theta})\mathbf{M}. \quad (1.16)$$

### 1.3.2 Unconditional models

The score  $\nabla l(\varphi)$  for unconditional canonical parameters is, as in every unconditional exponential family, ‘observed minus expected’:

$$\frac{\partial l(\varphi)}{\partial \varphi_j} = X_j - E_\varphi\{X_j\},$$

the unconditional expectation on the right-hand side being evaluated by using (1.12). If these parameters are modelled affinely as in (1.8), then we have (1.14) with  $\boldsymbol{\theta}$  replaced by  $\varphi$ . Note that (1.13) is not ‘observed minus conditionally expected’ if considered as a vector equation because the conditioning would differ amongst components.

Second derivatives with respect to unconditional canonical parameters of an exponential family are nonrandom, and hence observed and expected Fisher information matrices  $I(\varphi)$  are equal, given by either of the expressions  $\nabla^2 \psi(\varphi)$  and  $\text{var}_\varphi(\mathbf{X})$ . Fix  $\boldsymbol{\theta}$  and  $\varphi$  related by (1.5). For  $i, j \in G$ , the iterated covariance formula gives

$$\text{cov}_\varphi\{X_i, X_j\} = \frac{\partial^2 \psi_G(\boldsymbol{\theta}_G)}{\partial \theta_i \partial \theta_j} E_\varphi\{X_{p(G)}\} + \frac{\partial \psi_G(\boldsymbol{\theta}_G)}{\partial \theta_i} \frac{\partial \psi_G(\boldsymbol{\theta}_G)}{\partial \theta_j} \text{var}_\varphi\{X_{p(G)}\}. \quad (1.17)$$

Otherwise we may assume that  $j \in G$  and  $i$  is not a descendant of  $j$  so that  $\text{cov}_\varphi\{X_i, X_j | X_{p(G)}\} = 0$  because  $X_j$  is conditionally independent given  $X_{p(G)}$  of all variables except itself and its descendants. Then the iterated covariance formula gives

$$\text{cov}_\varphi\{X_i, X_j\} = \frac{\partial \psi_G(\boldsymbol{\theta}_G)}{\partial \theta_j} \text{cov}_\varphi\{X_i, X_{p(G)}\}. \quad (1.18)$$

Expectations having been calculated using (1.12), variances, the case  $i = j$  in (1.17), are calculated in one pass through (1.17) in any order that deals with parents before children. Then another pass using (1.17) and (1.18) gives covariances. The information matrix for  $\beta$  is given by (1.16) with  $\theta$  replaced by  $\varphi$ .

### 1.3.3 Prediction

By ‘prediction’, we only mean evaluation of a function of estimated parameters, what the `predict` function in R does for generalised linear models. In aster models we have five different parameterisations of interest,  $\beta$ ,  $\theta$ ,  $\varphi$ ,  $\xi$  and  $\tau$ . The Fisher information matrix for  $\beta$ , already described, handles predictions of  $\beta$ , 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  $\widetilde{\mathbf{M}}$  used for the prediction is different from that used for calculating  $\hat{\beta}$  and the Fisher information matrix  $I(\hat{\beta})$ , either observed or expected.

Let  $\eta$  be the affine predictor, i. e.,  $\eta = \theta$  for conditional models and  $\eta = \varphi$  for unconditional models, let  $\zeta$  be any one of  $\theta$ ,  $\varphi$ ,  $\xi$  and  $\tau$ , and let  $f : \eta \mapsto \zeta$ . Suppose we wish to predict

$$g(\beta) = h(\zeta) = h\{f(\widetilde{\mathbf{M}}\beta)\}. \quad (1.19)$$

Then, by the chain rule, (1.19) has derivative

$$\nabla g(\beta) = \nabla h(\zeta) \nabla f(\eta) \widetilde{\mathbf{M}}, \quad (1.20)$$

and, by the ‘usual’ asymptotics of maximum likelihood and the delta method, the asymptotic distribution of the prediction  $h(\hat{\zeta}) = g(\hat{\beta})$  is

$$N[g(\beta), \{\nabla g(\hat{\beta})\} I(\hat{\beta})^{-1} \{\nabla g(\hat{\beta})\}^T],$$

where  $\nabla g(\hat{\beta})$  is given by (1.20) with  $\hat{\eta} = \widetilde{\mathbf{M}}\hat{\beta}$  plugged in for  $\eta$  and  $\hat{\zeta} = f(\hat{\eta})$  plugged in for  $\zeta$ . We write ‘predictions’ in this complicated form to separate the parts of the specification, the functions  $h$  and  $\nabla h$  and the model matrix  $\widetilde{\mathbf{M}}$ , that change from application to application from the part  $\nabla f$  that does not change and can be dealt with by computer; see Appendix A for details.

To predict mean-value parameters one must specify new ‘response’ data  $X_j$  as well as new ‘covariate’ data in  $\widetilde{\mathbf{M}}$ . Unconditional mean-value parameters  $\tau$  depend on  $X_j$ ,  $j \in F$ , whereas conditional mean-value parameters  $\xi$  depend on  $X_j$ ,  $j \in J \cup F$ . It is often interesting, however, to predict  $\xi$  for hypothetical individuals with  $X_j = 1$  for all  $j$ , thus obtaining conditional mean values per unit sample size.

## 1.4 Software

We have released an R (R Development Core Team, 2004) package `aster` that fits, tests, predicts and simulates 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 generalised linear models. The R function `summary.aster` provides regression coefficients with standard errors,  $z$  statistics and  $p$ -values; `anova.aster` provides likelihood ratio tests for model comparison; `predict.aster` provides the predictions with standard errors described in § 1.3.3; `raster` generates random aster model data for simulation studies or parametric bootstrap calculations. The package is available from CRAN (<http://www.cran.r-project.org>) and is open source.

The current version of the package limits the general model described in this article in several ways. In predictions, only linear  $h$  are allowed in (1.19), but this can be worked around. For general  $h$ , observe that  $h(\hat{\zeta})$  and  $\mathbf{A}^T \hat{\zeta}$ , where  $\mathbf{A} = \nabla h(\zeta)$ , have the same standard errors. Thus, obtain  $h(\hat{\zeta})$  by one call to `predict.aster` and the standard errors for  $\mathbf{A}^T \hat{\zeta}$ , where  $\mathbf{A} = \nabla h(\hat{\zeta})$  by a second call. In models, the only conditional families currently implemented are Bernoulli, Poisson,  $k$ -truncated Poisson, negative binomial and  $k$ -truncated negative binomial, but adding another one-parameter exponential family would require only implementation of its  $\psi$ ,  $\psi'$  and  $\psi''$  functions and its random variate generator. Multiparameter conditional families, chain components, are not yet implemented. Allowing terminal nodes that are two-parameter normal or allowing child nodes that are multinomial given a common parent would require more extensive changes to the package.

## 1.5 Example

Data were collected on 570 individuals of *Echinacea angustifolia*, each having the data structure shown in Fig. 1.1. 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 randomised at the time of planting into a field within 6.5 km of all populations of origin. The dataset contains three predictor variables: `ewloc` and `nsloc` give east-west and north-south positions of individuals and `pop` gives their remnant population of origin. To use the R formula mini-language we need to create more variables: `resp` is a vector comprising the response variables, the  $M_j$ ,  $F_j$  and  $H_j$ ; `level` is categorical naming the type of response variable, with values  $M$ ,  $F$  and  $H$ ; `year` is categorical giving the year; `varb` is the interaction of `level` and `year`; and `hdct` is short for `level = H`.

Table 1.1: Tests for model comparison. The model formulae are given above and the analysis of deviance below; deviance is double the loglikelihood.

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 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

d. f., degrees of freedom.

We fitted many models; see Appendix D for details. Scientific interest focuses on the model comparison shown in Table 1.1. The models are nested. The affine predictor for Model 4 can be written

$$\varphi_j = \mu_{L_j, Y_j} + \alpha_{L_j} U_j + \beta_{L_j} V_j + \gamma_{L_j, P_j}, \quad (1.21)$$

where  $L_j$ ,  $Y_j$ ,  $U_j$ ,  $V_j$  and  $P_j$  are `level`, `year`, `ewloc`, `nsloc` and `pop`, respectively, and the alphas, betas and gammas are regression coefficients. Model 3 is the submodel of Model 4 that imposes the constraint  $\gamma_{M,P} = \gamma_{F,P}$ , for all populations  $P$ . Model 2 is the submodel of Model 3 that imposes the constraint  $\gamma_{M,P} = \gamma_{F,P} = 0$ , for all  $P$ . Model 1 is the submodel of Model 2 that imposes the constraint  $\gamma_{L,P} = 0$ , for all  $L$  and  $P$ .

All models contain the graph node effect, `varb` or  $\mu_{L_j, Y_j}$ , and the quantitative spatial effect, `level:(nsloc + ewloc)` or  $\alpha_{L_j} U_j + \beta_{L_j} V_j$ , which was chosen by comparing many models; for details see Appendix D. We explain here only differences amongst models, which involve only categorical predictors. In an unconditional aster model, which these are, such terms require the maximum likelihood estimates of mean-value parameters for each category, summed over all individuals in the category, to match their observed values: ‘observed equals expected’. Model 4 makes observed equal expected for total head count  $\sum_j H_j$ , for total flowering  $\sum_j F_j$ , and for total survival  $\sum_j M_j$  within each population. Model 3 makes observed equal expected for total head count  $\sum_j H_j$  and for total non-head count  $\sum_j (M_j + F_j)$  within each population. Model 2 makes observed equal expected for total head count  $\sum_j H_j$  within each population.

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. It is difficult to interpret Model 3 scientifically, because ‘non-head count’  $\sum_j(M_j + F_j)$  is unnatural, in effect scoring 0 for dead, 1 for alive without flowers or 2 for alive with flowers.

Model 2 is the model of primary scientific interest. Evolutionary biologists are fundamentally interested in fitness, but it is notoriously difficult to measure; see Beatty (1992), Keller (1992) and Paul (1992). The fitness of an individual may be defined as its lifetime contribution of progeny to the next generation. For these data the most direct surrogate measure for fitness is total 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 only through  $H_j$ , in an aster model the unconditional expectation of  $H_j$ , its unconditional 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 about future reproduction, but it becomes more true 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 in the model only to produce the correct stochastic structure. If we could directly model the marginal distribution of the  $H_j$ , but we cannot, 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; the mean-value parameters for those variables are nuisance parameters. 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 for addressing this issue, so we propose looking at confidence intervals for the mean-value parameters for total flower head count shown in Fig. 1.2. Although we have no formal test to propose, we claim that 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.

It being difficult to interpret Model 3 scientifically, Model 4 is the next larger readily interpretable model. Model 4 does fit significantly better than Model 2,  $P = 0.00016$ , which implies 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

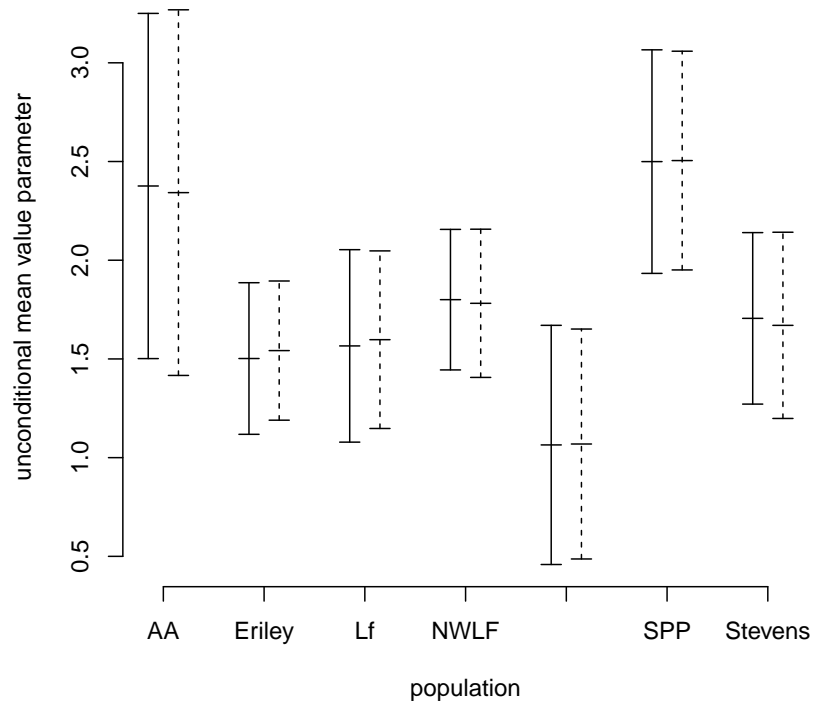


Figure 1.2: Confidence intervals for total head count. Shown are 95% non-simultaneous confidence intervals for the unconditional expectation of total flower head count, all three years, for individuals from different populations and central spatial location. Solid bars are based on Model 2 in Table 1.1, and dashed bars are based on Model 3 in Table 1.1.

fully accounts for their contributions through  $H_j$ .

Note that we would have obtained 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. Since a conditional aster model does not mix levels in this way, it must model all levels and so usually needs many more parameters than an unconditional model.

## 1.6 Discussion

The key idea of aster models, as we see it, is the usefulness of what we have called unconditional aster models, which have low-dimensional sufficient statistics (1.9). Following Geyer & Thompson (1992), who argued in favour of exponential family models with sufficient statistics chosen to be scientifically interpretable, an idea they attributed to Jaynes (1978), we emphasise the value of these models in analyses of life histories and overall fitness.

We do not insist, however, the R package `aster` is even-handed with respect to conditional and unconditional models and conditional and unconditional parameters. Users may use whatever seems best to them. Any joint analysis is better than any separate analyses of different variables. We have, however, one warning. In an unconditional aster model, as in all exponential family models, the map from canonical parameters to mean-value parameters is monotone; with sufficient statistic vector  $\mathbf{Y}$  given by (1.9) having components  $Y_k$  we have

$$-\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \beta_k^2} = \frac{\partial E_{\boldsymbol{\beta}}\{Y_k\}}{\partial \beta_k} > 0. \quad (1.22)$$

This gives regression coefficients their simple interpretation: an increase in  $\beta_k$  causes an increase in  $E_{\boldsymbol{\beta}}\{Y_k\}$ , other betas being held constant. The analogue of (1.22) for a conditional model is

$$-\frac{\partial^2 l(\boldsymbol{\beta})}{\partial \beta_k^2} = \frac{\partial}{\partial \beta_k} \sum_{G \in \mathcal{G}} \sum_{j \in G} E_{\boldsymbol{\beta}}\{X_j | X_{p(G)}\} m_{jk} > 0, \quad (1.23)$$

where  $m_{jk}$  are components of  $\mathbf{M}$ . Because we cannot move the sums in (1.23) inside the conditional expectation, there is no corresponding simple interpretation of regression coefficients. Conditional aster models are therefore algebraically simple but statistically complicated. Unconditional aster models are algebraically complicated but statistically simple. They can be simply explained as flat exponential families having the desired sufficient statistics.



They are the aster models that behave according to the intuitions derived from linear and generalised linear models.

We saw in our example that aster models allowed us to model fitness successfully. In medical applications, Darwinian fitness is rarely interesting, but aster models may allow data on mortality or survival to be analyzed in combination with other data, such as quality of life measures.

## Acknowledgements

Our key equation (1.5) was derived about 1980 when R. G. S. was a graduate student and C. J. G. not yet a graduate student. R. G. S.'s Ph. D. adviser Janis Antonovics provided some funding for development, from a now forgotten source, but which we thank nevertheless. High quality life history data, generated recently by S. W., Julie Etterson, David Heiser and Stacey Halpern, and work of Helen Hangelbroek, supported by the U. S. National Science Foundation (DMS-0083468), about what in hindsight we would call a conditional aster model analysis of mortality data, motivated implementation of these ideas. Were it not for Janis's enthusiastic 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. 115–9. Cambridge, MA: Harvard University Press.
- BRESLOW, N. (1972). Discussion of the paper by Cox (1972). *J. R. Statist. Soc. B* **34**, 216–7.
- 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*, Ed. J. M. Chambers and T. J. Hastie, pp. 13–44. Pacific Grove, CA: 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, MA: MIT Press.
- KELLER, E. F. (1992). Fitness: Reproductive ambiguities. In *Keywords in Evolutionary Biology*, Ed. E. F. Keller and E. A. Lloyd, pp. 120–1. Cambridge, MA: Harvard University Press.
- LAURITZEN, S. L. (1996). *Graphical Models*. New York: Oxford University Press.

- MARTIN, T. G., WINTLE, B. A., RHODES, J. R., KUHNERT, P. M., FIELD, S. A., LOW-CHOY, S. J., TYRE, A. J. & POSSINGHAM, H. P. (2005). Zero tolerance ecology: improving ecological inference by modelling the source of zero observations. *Ecol. Lett.* **8** 1235–46.
- MCCULLAGH, P. & NELDER, J. A. (1989). *Generalized Linear Models*, 2nd ed. London: Chapman and Hall.
- PAUL, D. (1992). Fitness: Historical perspective. In *Keywords in Evolutionary Biology*, Ed. E. F. Keller and E. A. Lloyd, pp. 112–4. Cambridge, MA: 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. *Appl. Statist.* **22**, 392–9.

# 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  $\hat{\beta}$ , and Fisher information  $I(\hat{\beta})$ , whether observed or expected makes no difference. The asymptotic distribution of  $\hat{\beta}$  is

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

where  $\beta$  is the true unknown parameter value.

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

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

Since we already have  $\hat{\beta}$  and  $I(\hat{\beta})$ , we only need  $\nabla g(\hat{\beta})$  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  $\tau$ , which, of course, is itself a function of  $\beta$  if the model is correct. That is, we have a composition

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

where  $h$  is an arbitrary function specified by the user and  $f_{\beta, \tau}$  is the map  $\beta \mapsto \tau$ . The point is that  $f_{\beta, \tau}$  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

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

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.

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

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

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

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

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

(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 (??). Let  $\Delta\theta_{ij}$  denote an increment in  $\theta_{ij}$  and similarly for  $\Delta\varphi_{ij}$ . Then

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

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  $\nabla f_{\eta, \zeta}(\hat{\eta}) \circ \mathbf{M}$ , the last two bits of (A.1d). We get, in this case where  $\eta = \theta$ ,

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

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

### A.2.3 Unconditional Canonical to Conditional Canonical

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

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

Since (A.3) has  $\Delta \theta_{im}$  terms on both sides, it must be used recursively, as with many other aster model equations, including (??) itself. Clearly, if (A.3) is used when  $S(j)$  is empty, it is trivial. This gives us (A.3) for all “leaf” nodes 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  $\mathbf{G} = \nabla f_{\eta, \zeta}(\hat{\eta}) \circ \mathbf{M}$  had components  $g_{ijk}$  and  $\mathbf{M}$  has components  $m_{ijk}$ , then we get in this case where  $\eta = \varphi$ ,

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

and (as discussed above), since  $g_{i.k}$  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  $f_{\theta, \xi}$ , 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

$$\frac{\partial \xi_{ij}}{\partial \theta_{ij}} = X_{ip(j)} \psi''_j(\theta_{ij}) \quad (\text{A.5})$$

(other partial derivatives being zero).

### A.2.5 Unconditional Canonical to Unconditional Mean Value

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

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

and so the derivative is trivially

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

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 (1.17) and (1.18) in Section 1.3.2.

### A.2.6 Conditional Canonical to Unconditional Mean Value

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

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

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

We can easily differentiate this directly

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

(and other partial derivatives are zero).

### A.2.7 Unconditional Canonical to Conditional Mean Value

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

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

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

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

Note that the inverse map is

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

### B.1.3 Cumulant Function

We must have

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

from which we get

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

### 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  $\tau$

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

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

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

( $\mu$  is the mean of the Poisson distribution not this distribution).

### B.3.2 Canonical Parameter

The log density is

$$\log f_{\mu}(x) = x \log \mu - \log(e^{\mu} - 1) - \log(x!)$$

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

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

### 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  $\theta$  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  $\mu$ , we can do rejection sampling. Just simulate  $\text{Poisson}(\mu)$  random variates until we get one greater than zero, and return it.
- For small  $\mu$ , naive rejection sampling can be arbitrarily slow. So we need another strategy.

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

$$f_{\mu}(x) = \frac{\mu^x}{x!} \frac{e^{-\mu}}{1 - e^{-\mu}}, \quad x = 1, 2, \dots$$

Consider rejection sampling from  $Y$  which is one plus a  $\text{Poisson}(\nu)$ . The density of  $Y$  is

$$g_{\nu}(y) = \frac{\nu^{y-1}}{(y-1)!} e^{-\nu}, \quad y = 1, 2, \dots$$

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  $\mu \leq \nu$ .

Introduce the notation

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

for the mean of  $X$ . And the upper bound is

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

If we're not worried about being maximally clever, then we can take  $\mu = \nu$ , so the upper bound becomes just  $\tau$ . 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! Woof!

## 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"   "f102"   "ld03"   "f103"   "ld04"   "f104"
```

the variables with numbers in the names are the columns of the response matrix of the `aster` model. The variables `ld0x` (where  $x$  is a digit) are the survival indicator variables for year 200 $x$  (one for alive, zero for dead). The variables `f10x` are the flowering indicator variables (one for any flowers, zero for none). The variables `hdct0x` 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", "f102", "f103", "f104",
+          "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
f102	ld02	bernoulli
f103	ld03	bernoulli
f104	ld04	bernoulli
hdct02	f102	truncated.poisson(truncation = 0)
hdct03	f103	truncated.poisson(truncation = 0)
hdct04	f104	truncated.poisson(truncation = 0)

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

```

(Intercept)      -1.1005073  0.1928540  -5.706  1.15e-08 ***
varbfl03         -0.2585034  0.2961442  -0.873  0.38272
varbfl04         -0.3368137  0.2601966  -1.294  0.19551
varbhdct02       1.4248487  0.2675730   5.325  1.01e-07 ***
varbhdct03       1.3772526  0.2203482   6.250  4.10e-10 ***
varbhdct04       1.9283230  0.2001443   9.635  < 2e-16 ***
varbld02        -0.1322951  0.3502261  -0.378  0.70562
varbld03         1.8823218  0.4281242   4.397  1.10e-05 ***
varbld04         4.2762222  0.3469256  12.326  < 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"      "NWLf"    "Nessman" "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)	-1.0846586	0.1925029	-5.635	1.76e-08	***
varbfl03	-0.2591673	0.2953707	-0.877	0.38025	
varbfl04	-0.3272159	0.2594037	-1.261	0.20716	
varbhdct02	1.5103441	0.2751442	5.489	4.04e-08	***
varbhdct03	1.4619118	0.2298569	6.360	2.02e-10	***
varbhdct04	2.0131411	0.2106706	9.556	< 2e-16	***
varbld02	-0.1398184	0.3499238	-0.400	0.68947	
varbld03	1.8718618	0.4279181	4.374	1.22e-05	***
varbld04	4.2604338	0.3467427	12.287	< 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:popNWLf      -0.1076672  0.0832346  -1.294  0.19582
hdct:popNessman   -0.3152202  0.1387855  -2.271  0.02313 *
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  -2717.4
2       33  -2701.3  6   16.135  0.01305 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

**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)	-1.396128	0.423538	-3.296	0.000980	***
varbfl03	-0.268508	0.293765	-0.914	0.360706	
varbfl04	-0.303001	0.257928	-1.175	0.240094	
varbhdct02	1.936558	0.546517	3.543	0.000395	***
varbhdct03	1.892261	0.525941	3.598	0.000321	***
varbhdct04	2.431206	0.516328	4.709	2.49e-06	***
varbld02	0.105169	0.582421	0.181	0.856704	
varbld03	2.101945	0.632563	3.323	0.000891	***
varbld04	4.474676	0.581362	7.697	1.39e-14	***
popEriley	0.807390	0.451694	1.787	0.073862	.
popLf	0.877245	0.481248	1.823	0.068325	.
popNWLf	-0.111507	0.434671	-0.257	0.797541	
popNessman	-0.602180	0.681430	-0.884	0.376858	
popSPP	0.541625	0.450808	1.201	0.229575	
popStevens	0.112023	0.465345	0.241	0.809764	
varbfl02:nsloc	0.083795	0.026329	3.183	0.001459	**
varbfl03:nsloc	0.066937	0.029982	2.233	0.025576	*
varbfl04:nsloc	0.070146	0.024162	2.903	0.003695	**
varbhdct02:nsloc	-0.018769	0.011491	-1.633	0.102397	
varbhdct03:nsloc	-0.001417	0.013619	-0.104	0.917146	
varbhdct04:nsloc	-0.004397	0.007261	-0.606	0.544766	

```

varbld02:nsloc      -0.044854    0.037640   -1.192  0.233398
varbld03:nsloc      0.030181    0.052719    0.572  0.566993
varbld04:nsloc      0.034831    0.040522    0.860  0.390030
varbfl02:ewloc      0.035972    0.026717    1.346  0.178168
varbfl03:ewloc     -0.015152    0.028934   -0.524  0.600505
varbfl04:ewloc     -0.018685    0.023867   -0.783  0.433709
varbhdct02:ewloc    0.006926    0.012016    0.576  0.564347
varbhdct03:ewloc    0.026914    0.013389    2.010  0.044412 *
varbhdct04:ewloc    0.012554    0.007241    1.734  0.082977 .
varbld02:ewloc      0.013009    0.035812    0.363  0.716410
varbld03:ewloc     -0.030830    0.051730   -0.596  0.551187
varbld04:ewloc      0.002422    0.040976    0.059  0.952866
levelhdct:popEriley -1.341288    0.587473   -2.283  0.022422 *
levelld:popEriley  -0.560351    0.553094   -1.013  0.311003
levelhdct:popLf     -1.408221    0.631418   -2.230  0.025731 *
levelld:popLf       -0.674658    0.588417   -1.147  0.251562
levelhdct:popNWLF   0.046822    0.553566    0.085  0.932593
levelld:popNWLF     0.140229    0.531548    0.264  0.791925
levelhdct:popNessman 0.418459    0.893928    0.468  0.639704
levelld:popNessman  0.922668    0.778683    1.185  0.236054
levelhdct:popSPP    -0.702929    0.573865   -1.225  0.220611
levelld:popSPP      -0.479659    0.561548   -0.854  0.393009
levelhdct:popStevens -0.215472    0.593594   -0.363  0.716608
levelld:popStevens -0.240029    0.569320   -0.422  0.673312
---
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   -2717.4
2         33   -2701.3  6    16.135 0.0130497 *
3         45   -2663.6 12    37.705 0.0001715 ***
---

```

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

**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)	-1.333883	0.419723	-3.178	0.001483	**
varbfl03	-0.384991	0.268534	-1.434	0.151665	
varbfl04	-0.373643	0.244457	-1.528	0.126398	
varbhdct02	1.845677	0.541799	3.407	0.000658	***
varbhdct03	1.858078	0.522036	3.559	0.000372	***
varbhdct04	2.372582	0.513754	4.618	3.87e-06	***
varbld02	0.165820	0.562506	0.295	0.768155	
varbld03	1.959135	0.611123	3.206	0.001347	**
varbld04	4.377685	0.573484	7.633	2.28e-14	***
popEriley	0.809271	0.451391	1.793	0.072999	.
popLf	0.878454	0.480895	1.827	0.067744	.
popNWLf	-0.111683	0.434221	-0.257	0.797023	
popNessman	-0.600230	0.681076	-0.881	0.378157	
popSPP	0.540528	0.450468	1.200	0.230168	
popStevens	0.110291	0.464961	0.237	0.812499	
levelfl:nsloc	0.068603	0.015126	4.535	5.75e-06	***
levelhdct:nsloc	-0.014500	0.007501	-1.933	0.053245	.
levelld:nsloc	0.001078	0.007299	0.148	0.882584	
levelfl:ewloc	0.004590	0.015092	0.304	0.761021	
levelhdct:ewloc	0.020550	0.007740	2.655	0.007927	**

```

levelld:ewloc      -0.001435    0.007502   -0.191  0.848252
yearyr03:nsloc     0.009125    0.007155    1.275  0.202241
yearyr04:nsloc     0.009184    0.006137    1.496  0.134535
yearyr03:ewloc     -0.001812    0.007242   -0.250  0.802409
yearyr04:ewloc     -0.011085    0.006324   -1.753  0.079649 .
levelhdct:popEriley -1.344673    0.587258   -2.290  0.022036 *
levelld:popEriley  -0.561644    0.552345   -1.017  0.309232
levelhdct:popLf    -1.410639    0.631143   -2.235  0.025414 *
levelld:popLf      -0.674982    0.587591   -1.149  0.250668
levelhdct:popNWLF  0.046781    0.553089    0.085  0.932594
levelld:popNWLF    0.140718    0.530656    0.265  0.790872
levelhdct:popNessman 0.416160    0.893648    0.466  0.641440
levelld:popNessman 0.919968    0.777808    1.183  0.236900
levelhdct:popSPP   -0.701296    0.573572   -1.223  0.221449
levelld:popSPP     -0.477678    0.560739   -0.852  0.394286
levelhdct:popStevens -0.213370    0.593238   -0.360  0.719093
levelld:popStevens -0.237795    0.568455   -0.418  0.675714
---
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	-2668.4			
2	45	-2663.6	8	4.8342	0.7751

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)	-1.199454	0.415402	-2.887	0.003884	**
varbfl03	-0.334899	0.264510	-1.266	0.205473	
varbfl04	-0.328208	0.240321	-1.366	0.172032	
varbhdct02	1.648701	0.536935	3.071	0.002136	**
varbhdct03	1.667842	0.516225	3.231	0.001234	**
varbhdct04	2.171474	0.509537	4.262	2.03e-05	***
varbld02	0.075042	0.562247	0.133	0.893823	
varbld03	1.827669	0.611052	2.991	0.002780	**
varbld04	4.262450	0.573689	7.430	1.09e-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	.
popNWLf	-0.138061	0.433020	-0.319	0.749854	
popNessman	-0.721910	0.677652	-1.065	0.286735	
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:popNWLf	0.072362	0.552482	0.131	0.895795	
levelld:popNWLf	0.183780	0.534492	0.344	0.730967	
levelhdct:popNessman	0.581044	0.890106	0.653	0.513898	
levelld:popNessman	1.048141	0.779258	1.345	0.178609	
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  -2696.6
2      37  -2668.4  8   28.1778 0.0004416 ***
3      45  -2663.6  8    4.8342 0.7751464
```

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

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)	-1.365862	0.420138	-3.251	0.001150	**
varbfl03	-0.354186	0.266796	-1.328	0.184326	
varbfl04	-0.319952	0.242895	-1.317	0.187756	
varbhdct02	1.882750	0.542402	3.471	0.000518	***
varbhdct03	1.906417	0.521950	3.652	0.000260	***
varbhdct04	2.404151	0.514751	4.671	3.00e-06	***
varbld02	0.218855	0.562017	0.389	0.696973	
varbld03	1.973463	0.611062	3.230	0.001240	**
varbld04	4.406045	0.573494	7.683	1.56e-14	***
popEriley	0.812524	0.451595	1.799	0.071982	.
popLf	0.882088	0.481103	1.833	0.066733	.
popNWLf	-0.110570	0.434822	-0.254	0.799272	
popNessman	-0.593161	0.680863	-0.871	0.383651	
popSPP	0.543515	0.450805	1.206	0.227951	
popStevens	0.114072	0.465336	0.245	0.806349	
levelfl:nsloc	0.070763	0.014568	4.857	1.19e-06	***

```

levelhdct:nsloc      -0.006425   0.005465  -1.176  0.239751
levelld:nsloc        0.008036   0.005924   1.356  0.174949
levelfl:ewloc        0.007768   0.014546   0.534  0.593340
levelhdct:ewloc      0.011689   0.005561   2.102  0.035575 *
levelld:ewloc       -0.007240   0.006114  -1.184  0.236296
levelhdct:popEriley -1.350424   0.587897  -2.297  0.021616 *
levelld:popEriley   -0.563945   0.552117  -1.021  0.307054
levelhdct:popLf     -1.416802   0.631793  -2.243  0.024928 *
levelld:popLf       -0.678297   0.587394  -1.155  0.248190
levelhdct:popNWLF   0.044673   0.554325   0.081  0.935768
levelld:popNWLF     0.139453   0.530726   0.263  0.792736
levelhdct:popNessman 0.405845   0.893667   0.454  0.649732
levelld:popNessman  0.912760   0.777262   1.174  0.240264
levelhdct:popSPP    -0.705876   0.574445  -1.229  0.219148
levelld:popSPP      -0.481472   0.560611  -0.859  0.390432
levelhdct:popStevens -0.218838   0.594180  -0.368  0.712648
levelld:popStevens  -0.242044   0.568322  -0.426  0.670187
---
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	-2696.6			
2	33	-2674.7	4	21.8671	0.000213 ***
3	37	-2668.4	4	6.3107	0.177116
4	45	-2663.6	8	4.8342	0.775146

```

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

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)	-1.196096	0.416050	-2.875	0.004042	**
varbfl03	-0.316825	0.265066	-1.195	0.231981	
varbfl04	-0.337380	0.241251	-1.398	0.161973	
varbhdct02	1.652546	0.538051	3.071	0.002131	**
varbhdct03	1.627233	0.517159	3.146	0.001652	**
varbhdct04	2.172643	0.510169	4.259	2.06e-05	***
varbld02	0.066383	0.562724	0.118	0.906094	
varbld03	1.837868	0.611831	3.004	0.002666	**
varbld04	4.251807	0.574053	7.407	1.30e-13	***
popEriley	0.741522	0.449483	1.650	0.099000	.
popLf	0.799687	0.479291	1.668	0.095221	.
popNWLf	-0.128513	0.433273	-0.297	0.766765	
popNessman	-0.727352	0.678773	-1.072	0.283914	
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:popNWLf	0.059302	0.552518	0.107	0.914526	
levelld:popNWLf	0.174231	0.534494	0.326	0.744444	
levelhdct:popNessman	0.587678	0.891330	0.659	0.509686	
levelld:popNessman	1.055395	0.779991	1.353	0.176028	
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  -2696.6
2      33  -2691.9  4   4.6127   0.3294
3      37  -2668.4  4  23.5651 9.761e-05 ***
4      45  -2663.6  8   4.8342   0.7751
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

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)	-1.050644	0.184332	-5.700	1.20e-08	***
varbfl03	-0.349096	0.267919	-1.303	0.1926	
varbfl04	-0.344222	0.243899	-1.411	0.1581	
varbhdct02	1.450651	0.265192	5.470	4.50e-08	***
varbhdct03	1.472612	0.219375	6.713	1.91e-11	***
varbhdct04	1.980565	0.205048	9.659	< 2e-16	***
varbld02	-0.029302	0.315703	-0.093	0.9260	
varbld03	1.740051	0.396189	4.392	1.12e-05	***
varbld04	4.188577	0.334266	12.531	< 2e-16	***
levelfl:nsloc	0.070102	0.014652	4.785	1.71e-06	***
levelhdct:nsloc	-0.005804	0.005550	-1.046	0.2956	
levelld:nsloc	0.007165	0.005867	1.221	0.2220	
levelfl:ewloc	0.017977	0.014413	1.247	0.2123	
levelhdct:ewloc	0.007606	0.005561	1.368	0.1714	
levelld:ewloc	-0.004787	0.005919	-0.809	0.4186	
hdct:popEriley	-0.178799	0.089411	-2.000	0.0455	*
hdct:popLf	-0.162516	0.096116	-1.691	0.0909	.
hdct:popNWLf	-0.108209	0.083110	-1.302	0.1929	
hdct:popNessman	-0.315507	0.138823	-2.273	0.0230	*
hdct:popSPP	0.019942	0.086198	0.231	0.8170	
hdct:popStevens	-0.129238	0.089129	-1.450	0.1471	

---

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	-2712.5			
2	33	-2674.7	12	37.838	0.0001632 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## 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)	-1.066365	0.184593	-5.777	7.61e-09	***
varbfl03	-0.349426	0.268532	-1.301	0.193	
varbfl04	-0.353847	0.244529	-1.447	0.148	
varbhdct02	1.364697	0.257183	5.306	1.12e-07	***
varbhdct03	1.388181	0.209331	6.632	3.32e-11	***
varbhdct04	1.895092	0.194176	9.760	< 2e-16	***
varbld02	-0.021551	0.315991	-0.068	0.946	
varbld03	1.750199	0.396379	4.415	1.01e-05	***
varbld04	4.203943	0.334412	12.571	< 2e-16	***
levelfl:nsloc	0.070281	0.014705	4.779	1.76e-06	***
levelhdct:nsloc	-0.005378	0.005578	-0.964	0.335	
levelld:nsloc	0.006855	0.005868	1.168	0.243	
levelfl:ewloc	0.017686	0.014441	1.225	0.221	
levelhdct:ewloc	0.007312	0.005542	1.320	0.187	
levelld:ewloc	-0.005027	0.005932	-0.847	0.397	

---

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	-2728.7			
2	21	-2712.5	6	16.181	0.0128141 *
3	33	-2674.7	12	37.838	0.0001632 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

### 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)	-1.184797	0.223462	-5.302	1.15e-07	***
varbfl03	-0.349106	0.266693	-1.309	0.19053	
varbfl04	-0.330903	0.242658	-1.364	0.17268	
varbhdct02	1.662588	0.331242	5.019	5.19e-07	***
varbhdct03	1.684909	0.296256	5.687	1.29e-08	***
varbhdct04	2.190786	0.285814	7.665	1.79e-14	***
varbld02	-0.016209	0.316060	-0.051	0.95910	
varbld03	1.733685	0.396388	4.374	1.22e-05	***
varbld04	4.160172	0.334436	12.439	< 2e-16	***
popEriley	0.375414	0.154884	2.424	0.01536	*
popLf	0.355237	0.164870	2.155	0.03119	*
popNWLf	0.012016	0.145094	0.083	0.93400	
popNessman	0.231430	0.189693	1.220	0.22245	
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:popNWLf	-0.114923	0.245244	-0.469	0.63935	
hdct:popNessman	-0.705433	0.355627	-1.984	0.04730	*



```

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  -2728.7
2      21  -2712.5  6   16.181 0.0128141 *
3      27  -2684.9  6   27.672 0.0001083 ***
4      33  -2674.7  6   10.166 0.1178435
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

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
> 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. 48.

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")
```

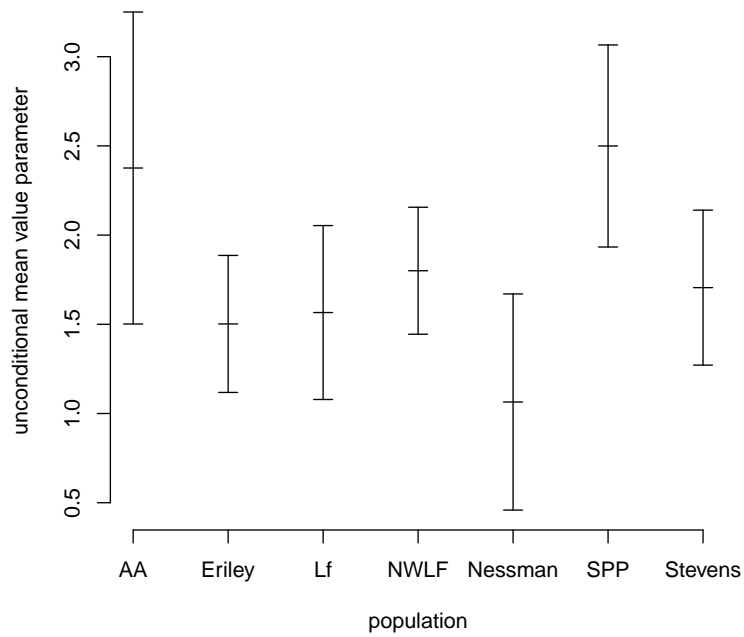


Figure D.1: 95% confidence intervals for unconditional mean value parameter for fitness (sum of head count for all years) at each population for a “typical” individual having position zero-zero and having the parameterization of Model Eight. Tick marks in the middle of the bars are the center (the MLE).

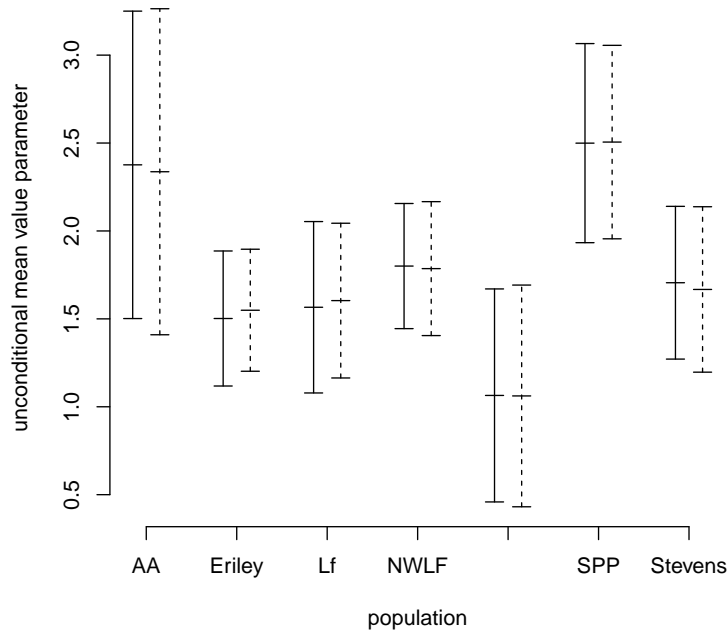


Figure D.2: 95% confidence intervals for unconditional mean value parameter for fitness (sum of head count for all years) at each population for a “typical” individual having position zero-zero and having the parameterization of Model Eight (solid bar) or Model Six (dashed bar). Tick marks in the middle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

```
> axis(side = 1, at = i, labels = popnames)
> title(xlab = "population")
```

and appears on p. 49.

### 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")
> 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. 51.

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)
```

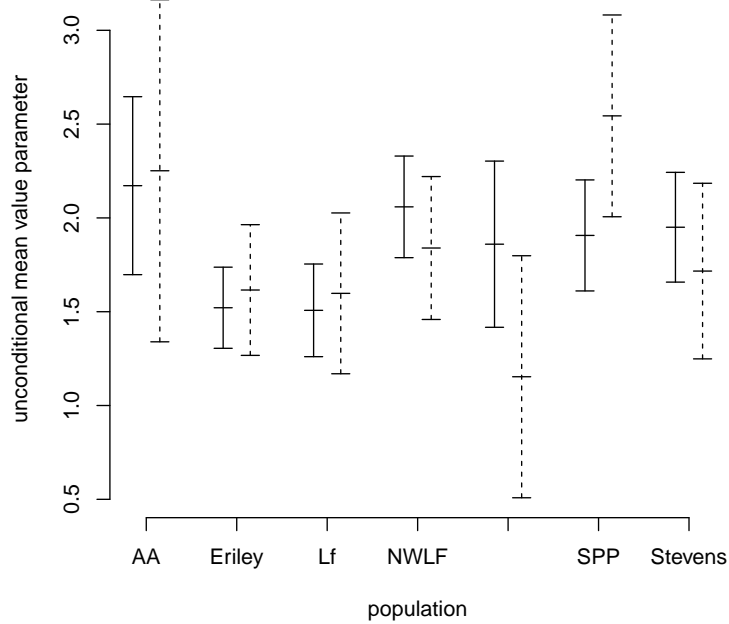


Figure D.3: 95% confidence intervals for unconditional mean value parameter for fitness (sum of head count for all years) at each population for a “typical” individual having position zero-zero and having the parameterization of Model Eight (solid bar) or Model Six (dashed bar). Tick marks in the middle 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*.

## 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	-2721.0				
	2	21	-2693.1	6	27.876	9.916e-05	***
	3	27	-2678.4	6	14.722	0.022533	*
	4	33	-2661.0	6	17.398	0.007925	**

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

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. 53.

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



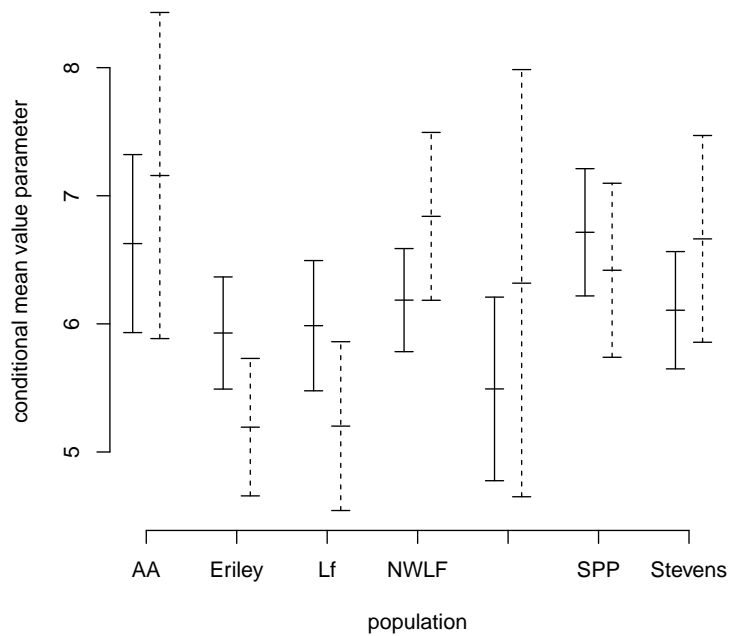


Figure D.4: 95% confidence intervals for conditional mean value parameter for fitness (sum of head count for all years) at each population for a “typical” individual having position zero-zero and having the parameterization of Model Eight (solid bar) or Model Six (dashed bar). Tick marks in the middle 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*.

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.

## 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. 55.

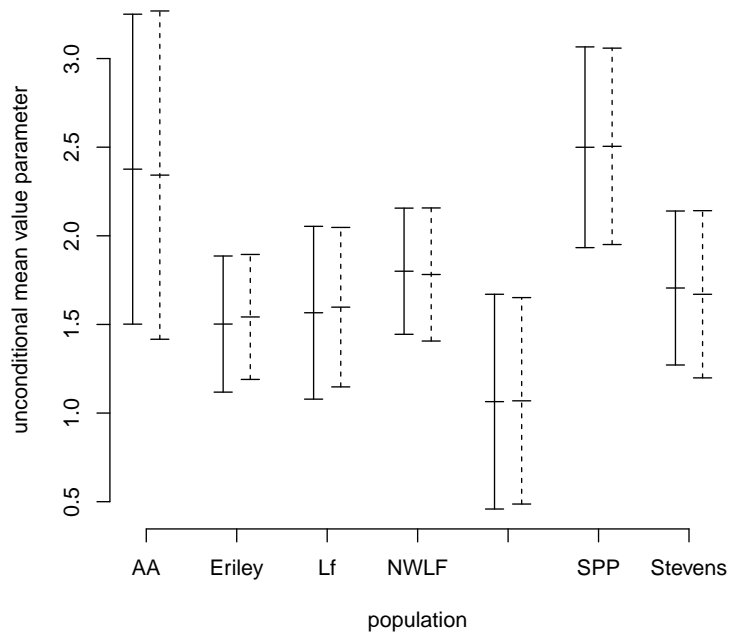


Figure D.5: 95% confidence intervals for unconditional mean value parameter for fitness (sum of head count for all years) at each population for a “typical” individual having position zero-zero and having the parameterization of Model Eight (solid bar) or Model Ten (dashed bar). Tick marks in the middle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

# Appendix E

## Canonical Parameter Spaces of Aster Models

### E.1 Basic Exponential Family Theory

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

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

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

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

is a convex set.

The full standard exponential family generated by  $\lambda$  is the family

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

where  $P_\varphi$  is the distribution having density with respect to  $\lambda$  defined by

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

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

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

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

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  $\varphi$ , we call  $\psi$  the *cumulant function* of the family.

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

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

$$g_\varphi(x) = \frac{f_\varphi(x)}{f_{\varphi^*}(x)} = \frac{c(\varphi^*)}{c(\varphi)} e^{(x, \varphi - \varphi^*)} \quad (\text{E.1})$$

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

## 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  $\Theta_j$  is the full canonical parameter space of the conditional one-parameter exponential family model at the  $j$ -th node (which has cumulant function  $\psi_j$ ), 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

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

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

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

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

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

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

(we are using the original parameter  $\varphi$  instead of the “new” parameter  $\boldsymbol{\delta}$ ). It stands to reason that  $\Phi$  is just the set of points obtained by mapping  $\Theta$  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

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

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

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

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  $\theta_j$  and  $\varphi_j$  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

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

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

$$\int \exp(x_j [\theta_j^* - \theta_j]) P_{\theta_j^*, j}(dx_j | x_{p(j)}) = \exp(x_{p(j)}[\psi_j(\theta_j) - \psi_j(\theta_j^*)]). \quad (\text{E.4b})$$

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  $\varphi_j$ . Since

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

the finiteness condition for this section is

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

### E.2.4 All Nodes by Mathematical Induction

Now assume (the induction hypothesis) that

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

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  $\Phi$  is indeed the image of  $\Theta$  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  $\Phi$  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  $\Theta$  and  $\Phi$ , it follows that  $\Theta$  and  $\Phi$  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  $\Theta$  is open if and only if each  $\Theta_j$  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  $\varphi_i$  in the interior of  $\Phi$  and  $\varphi_b$  on the boundary of  $\Phi$ , then

$$\langle \nabla \psi(t\varphi_i + (1-t)\varphi_b), \varphi_b - \varphi_i \rangle \rightarrow \infty, \quad \text{as } t \downarrow 0.$$

Clearly this carries over to any flat subfamily (formed by intersecting the full canonical parameter space  $\Phi$  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  $\mathbf{x} \mapsto \hat{\varphi}(\mathbf{x})$  is one-to-one and is found by solving the “likelihood equations” which have “observed equals expected” form, as in equation (??) 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,  $\nabla \psi : \varphi \mapsto \tau$  in aster model notation,



maps the interior of the full canonical parameter space  $\Phi$  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

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

It is clear also that as  $\theta$  runs over the interior of  $\Theta$  the mean value parameter  $\tau(\theta)$  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.