More Supporting Data Analysis for "Unifying Life History Analysis for Inference of Fitness and Population Growth"

By

Ruth G. Shaw, Charles J. Geyer, Stuart Wagenius,
Helen H. Hangelbroek, and Julie R. Etterson

Technical Report No. 661
School of Statistics
University of Minnesota
December 26, 2007

Abstract

This technical report (TR) gives details of a data reanalysis backing up a paper having the same authors as this TR and having the title that is quoted in the title of this TR. This reanalysis was not in the first submission of the paper, which instead had analyses given in Chapters 3 and 4 of TR 658. This analysis is for the second submission (to the same journal, *American Naturalist*) of that paper. Unlike the first analyses, these reanalyses directly estimate the fitness landscape rather than quantities related to it. The two analyses are also much more alike than the two analyses for the first submission. Both estimate exactly the same quantities, although one has to work harder to do so.

In an unrelated issue, we also give an example of subsampling a component of fitness and its affect on parameter estimates. This issue was mentioned in the first draft of the paper, but this is the first worked example illustrating this method.

1 Creating this Document

This document is created from its source file `tr661.Rnw` using the R `Sweave` command and the $\LaTeX$ document preparation system. First do

```
Sweave("tr661.Rnw")
```

if you have downloaded the file, or do

```
Sweave(url("http://www.stat.umn.edu/geyer/aster/tr661/tr661.Rnw"))
```

otherwise. This step takes an hour and a half on a fairly fast computer because of the Monte Carlo calculation in Section 3.4, and this step needs to be redone until the statements `print(ok)` on pages 13 and 33 print `TRUE`.

Then process the output, `tr661.tex` and several files with suffixes `pdf` and `eps` in the usual fashion (which depends on your system and installation).

2 Introduction

The analysis presented in this technical report is one more attempt to do full justice to the *Chamaecrista* data described below. As the experiment was designed there were multiple components of fitness. For each plant that survived to that stage, fruits were counted (`fruit`) and then a random sample of fruits of size 3 was taken and the seeds in those fruits counted (`seed`). This experimental design does not fit aster models perfectly (not the fault of the experimenters because the experiment was done before aster models were described). It would have been better if seeds were counted for all fruits or for a fraction p of fruits.

Nevertheless, we do what we can. Using a Monte Carlo calculation we can still estimate the fitness surface that corresponds to any aster model we decide fits the data. We can use the parametric bootstrap to carry out statistical tests or confidence intervals, although these no longer have a simple relationship to the parameters of the fitted aster model (as they would if the experimental design had been more favorable to aster analysis).

In Section 3 we perform an aster analysis in which both components of fitness, `fruit` and `seed` are used, and fitness is deemed to be `fruit * seed / 3`. The multiplication in this definition complicates estimation of expected fitness. The aster software can calculate the expectation of any linear combination of components of fitness, but it cannot calculate expectations of nonlinear functions of components of fitness. Fortunately, expectations that cannot be calculated exactly can be approximated by Monte Carlo. This takes time but is not otherwise problematic.

In Section 4 we perform an aster analysis in which `fruit` is deemed fitness. This illustrates the typical situation in which a linear combination of fitness components is deemed fitness and no Monte Carlo calculation is needed.

3 Analysis involving Both Components of Fitness

3.1 Data

We reanalyze a subset of the data analyzed by Etterson and Shaw (2001). These data are in the `chamae` dataset in the `aster` contributed package to the R statistical computing environment (R Development Core Team, 2006). Individuals of *Chamaecrista fasciculata* (common name, partridge pea) were obtained from three locations in the country and planted in three field sites. Of the complete data we only reanalyze here individuals planted in one field site (Minnesota).

These data are already in "long" format, no need to use the `reshape` function on them to do aster analysis. We will, however, need the "wide" format for Lande-Arnold analysis (Lande and Arnold, 1983). So we do that, before making any changes (we will add newly defined variables) to `chamae`.

```
> library(aster)
> data(chamae)
> chamaew <- reshape(chamae, direction = "wide", timevar = "varb",
+     v.names = "resp", varying = list(levels(chamae$varb)))
> names(chamaew)

[1] "id"     "root"   "STG1N"  "LOGLVS" "LOGSLA" "BLK"    "fecund"
[8] "fruit"  "seed"
```

For each individual, many characteristics were measured, three of which we consider phenotypic characters (so our z is three-dimensional), and others which combine to make up an estimate of fitness. The three phenotypic characters are reproductive stage (`STG1N`), log leaf number (`LOGLVS`), and log leaf thickness (`LOGSLA`). "At the natural end of the growing season, [they] recorded total pod number and seed counts from three representative pods; from these measures, [they] estimated [fitness]" (Etterson and Shaw, 2001, further explained in their note 12).

Although aster model theory in the published version of Geyer, et al. (2007) does allow conditionally multinomial response variables, versions of the `aster` package up through 0.7-2, the current version

$$\begin{tikzcd}
 & & 1 \arrow[dl] \\
 & \texttt{fecund} \arrow[dl] \arrow[dr] & \\
\texttt{seed} & & \texttt{fruit}
\end{tikzcd}$$

Figure 1: Graph for *Chamaecrista* Aster Data. Arrows go from parent nodes to child nodes. Nodes are labeled by their associated variables. The only root node is associated with the constant variable 1. `fecund` is Bernoulli (zero indicates no seeds, one indicates nonzero seeds). If `fecund` is zero, then so are the other variables. If `fecund` is nonzero, then `fruit` (fruit count) and `seed` (seed count) are conditionally independent, `fruit` has a two-truncated negative binomial distribution, and `seed` has a zero-truncated negative binomial distribution.

at the time this was written, do not. Multinomial response, if we could use it, would allow us to deal individuals having seeds counted from 0, 1, 2, or 3 fruits. To avoid multinomial response, we remove individuals with seeds counted for only one or two fruits (there were only four such).

Figure 1 shows the graph of the aster model we use for these data. Fruit count (`fruit`) and seed count (`seed`) are dependent only in that if one is zero, then so is the other (we only model fruit count for individuals who have seeds, because fruit count for other individuals is irrelevant). Given that neither is zero (when `fecund == 1`), they are conditionally independent. Given that fruit count is nonzero, it is at least three (by our data modifications). The conditional distribution of `seed` given that it is nonzero is what is called zero-truncated negative binomial, which is negative binomial conditioned on being greater than zero. By analogy we call the conditional distribution of `fruit` given that it is nonzero, two-truncated negative binomial, which is negative binomial conditioned on being greater than two.

3.2 Aster Analysis

We need to choose the non-exponential-family parameters (sizes) for the negative binomial distributions, since the `aster` package only does maximum likelihood for exponential family parameters. We start with the following values, which were chosen with knowledge of the maximum likelihood estimates for these parameters, which we find in Section 3.3. The values that are found then are written out to a file and loaded here if the file exists, so after several runs (of `Sweave`) we are reading in here the maximum likelihood values of these non-exponential-family parameters.

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("chamae-alpha.rda"))
> options(show.error.messages = TRUE, warn = 0)
> ok <- exists("alpha.fruit") && exists("alpha.seed")
> if (! ok) {
+     alpha.fruit <- 3.0
+     alpha.seed <- 15.0
+ }
> print(alpha.fruit)

[1] 3

> print(alpha.seed)

[1] 15
```

Then we set up the aster model framework.

```
> vars <- c("fecund", "fruit", "seed")
> pred <- c(0,1,1)
> famlist <- list(fam.bernoulli(), fam.poisson(),
+     fam.truncated.negative.binomial(size = alpha.seed, truncation = 0),
+     fam.truncated.negative.binomial(size = alpha.fruit, truncation = 2))
> fam <- c(1,4,3)
```

We can now fit our first aster model.

```
> out1 <- aster(resp ~ varb + BLK, pred, fam, varb, id, root,
+     data = chamae, famlist = famlist)
> summary(out1, show.graph = TRUE)

Call:
aster.formula(formula = resp ~ varb + BLK, pred = pred, fam = fam,
    varvar = varb, idvar = id, root = root, data = chamae, famlist = famlist)


Graphical Model:
 variable predecessor
 fecund   root
 fruit    fecund
 seed     fecund
 family
 bernoulli
 truncated.negative.binomial(size = 3, truncation = 2)
 truncated.negative.binomial(size = 15, truncation = 0)

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

```
(Intercept) -2.086e+01  1.211e-01 -172.276   <2e-16 ***
varbfruit    2.184e+01  1.213e-01  180.095   <2e-16 ***
varbseed     2.142e+01  1.241e-01  172.652   <2e-16 ***
BLK2         4.070e-04  5.242e-04    0.776    0.438
BLK4         4.808e-03  4.771e-04   10.077   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The “response” `resp` is a numeric vector containing all the response variables (`fecund`, `fruit`, and `seed`). The “predictor” `varb` is a factor with three levels distinguishing with `resp` which original response variable an element is. The predictor `BLK` has not been mentioned so far. It is block within the field where the plants were grown.

Now we add phenotypic variables.

```
> out2 <- aster(resp ~ varb + BLK + LOGLVS + LOGSLA + STG1N,
+     pred, fam, varb, id, root, data = chamae, famlist = famlist)
> summary(out2)

Call:
aster.formula(formula = resp ~ varb + BLK + LOGLVS + LOGSLA +
    STG1N, pred = pred, fam = fam, varvar = varb, idvar = id,
    root = root, data = chamae, famlist = famlist)

              Estimate Std. Error  z value Pr(>|z|)
(Intercept) -2.041e+01  1.234e-01 -165.320   <2e-16 ***
varbfruit    2.135e+01  1.241e-01  172.039   <2e-16 ***
varbseed     2.093e+01  1.268e-01  165.065   <2e-16 ***
BLK2         2.556e-04  5.760e-04    0.444   0.6573
BLK4         2.241e-03  5.456e-04    4.108    4e-05 ***
LOGLVS       1.091e-02  9.971e-04   10.938   <2e-16 ***
LOGSLA       7.123e-03  2.998e-03    2.376   0.0175 *
STG1N        5.751e-03  2.925e-04   19.661   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

One might think we should use `varb * (LOGLVS + LOGSLA + STG1N)` but it turns out this is too many parameters and the Fisher information is ill conditioned, as shown by the need to use the `info.tol` argument.

```
> out2foo <- aster(resp ~ BLK + varb * (LOGLVS + LOGSLA + STG1N),
+     pred, fam, varb, id, root, data = chamae, famlist = famlist)
> summary(out2foo, info.tol = 1e-11)

Call:
aster.formula(formula = resp ~ BLK + varb * (LOGLVS + LOGSLA +
    STG1N), pred = pred, fam = fam, varvar = varb, idvar = id,
    root = root, data = chamae, famlist = famlist)

                   Estimate Std. Error z value Pr(>|z|)    
(Intercept)      -1.058e+01  1.578e+00  -6.702 2.06e-11 ***
BLK2              8.992e-04  5.977e-04   1.504    0.132    
BLK4              2.258e-03  5.773e-04   3.912 9.17e-05 ***
varbfruit         1.151e+01  1.580e+00   7.286 3.18e-13 ***
varbseed          1.095e+01  1.618e+00   6.771 1.28e-11 ***
LOGLVS           -4.550e+00  4.392e-01 -10.361  < 2e-16 ***
LOGSLA            1.627e+00  1.669e+00   0.975    0.330    
STG1N             1.254e+00  1.584e-01   7.916 2.46e-15 ***
varbfruit:LOGLVS  4.571e+00  4.393e-01  10.404  < 2e-16 ***
varbseed:LOGLVS   4.591e+00  4.510e-01  10.181  < 2e-16 ***
varbfruit:LOGSLA -1.623e+00  1.671e+00  -0.971    0.331    
varbseed:LOGSLA  -1.666e+00  1.707e+00  -0.976    0.329    
varbfruit:STG1N  -1.255e+00  1.585e-01  -7.914 2.48e-15 ***
varbseed:STG1N   -1.232e+00  1.627e-01  -7.575 3.60e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> anova(out2, out2foo)
```

```
Analysis of Deviance Table

Model 1: resp ~ varb + BLK + LOGLVS + LOGSLA + STG1N
Model 2: resp ~ BLK + varb * (LOGLVS + LOGSLA + STG1N)
  Model Df Model Dev Df Deviance P(>|Chi|)    
1        8    -87067                          
2       14    -86235  6   831.88 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Despite the statistically significant improvement (based on the chi-square approximation to the log likelihood ratio, which may not be valid with such an ill-conditioned Fisher information), we do not adopt this model (`out2foo`) either.

Although we cannot afford 9 parameters (3 levels of `varb` times 3 predictor variables) for the interaction, we can afford 6, only putting the phenotype variables in at level `fruit` and `seed`. Because we are fitting an unconditional aster model, the effects of these terms are passed down to `fecund`. See the example in Geyer, et al. (2007) for discussion of this phenomenon.

```
> foo <- as.numeric(as.character(chamae$varb) == "fruit")
> chamae$LOGLVSfr <- chamae$LOGLVS * foo
> chamae$LOGSLAfr <- chamae$LOGSLA * foo
> chamae$STG1Nfr <- chamae$STG1N * foo
> foo <- as.numeric(as.character(chamae$varb) == "seed")
> chamae$LOGLVSsd <- chamae$LOGLVS * foo
> chamae$LOGSLAsd <- chamae$LOGSLA * foo
> chamae$STG1Nsd <- chamae$STG1N * foo
> out6 <- aster(resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + STG1Nfr +
+     LOGLVSsd + LOGSLAsd + STG1Nsd, pred, fam, varb, id, root, data = chamae,
+     famlist = famlist)
> summary(out6)

Call:
aster.formula(formula = resp ~ varb + BLK + LOGLVSfr + LOGSLAfr +
    STG1Nfr + LOGLVSsd + LOGSLAsd + STG1Nsd, pred = pred, fam = fam,
    varvar = varb, idvar = id, root = root, data = chamae, famlist = famlist)

              Estimate Std. Error  z value Pr(>|z|)
(Intercept) -1.990e+01  1.275e-01 -156.165  < 2e-16 ***
varbfruit    2.084e+01  1.281e-01  162.656  < 2e-16 ***
varbseed     2.051e+01  1.341e-01  152.900  < 2e-16 ***
BLK2         5.026e-04  5.902e-04    0.852   0.3944
BLK4         2.281e-03  5.642e-04    4.042  5.3e-05 ***
LOGLVSfr     1.930e-02  1.150e-03   16.782  < 2e-16 ***
LOGSLAfr     8.938e-03  4.060e-03    2.201   0.0277 *
STG1Nfr      2.732e-04  3.415e-04    0.800   0.4237
LOGLVSsd    -8.217e-02  8.101e-03  -10.143  < 2e-16 ***
LOGSLAsd    -1.029e-02  2.926e-02   -0.352   0.7251
STG1Nsd      5.800e-02  3.056e-03   18.982  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

When we analyzed the Minnesota-Minnesota subset alone (the subset of these data consisting of only the Minnesota population) the there was no statistically significant effect of the phenotypic predictors on seed count. In these data that effect is significant.

```
> out5 <- aster(resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + STG1Nfr,
+     pred, fam, varb, id, root, data = chamae, famlist = famlist)
> summary(out5)

Call:
aster.formula(formula = resp ~ varb + BLK + LOGLVSfr + LOGSLAfr +
    STG1Nfr, pred = pred, fam = fam, varvar = varb, idvar = id,
    root = root, data = chamae, famlist = famlist)

              Estimate Std. Error  z value Pr(>|z|)
(Intercept) -2.045e+01  1.230e-01 -166.325  < 2e-16 ***
varbfruit    2.139e+01  1.238e-01  172.861  < 2e-16 ***
```

```
varbseed     2.102e+01  1.259e-01  166.924  < 2e-16 ***
BLK2         1.688e-04  5.673e-04    0.298   0.7661
BLK4         2.145e-03  5.375e-04    3.992 6.56e-05 ***
LOGLVSfr     1.286e-02  1.081e-03   11.892  < 2e-16 ***
LOGSLAfr     7.266e-03  3.254e-03    2.233   0.0255 *
STG1Nfr      5.744e-03  3.129e-04   18.357  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> anova(out5, out6)

Analysis of Deviance Table

Model 1: resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + STG1Nfr
Model 2: c("resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + STG1Nfr + LOGLVSsd + ", "    LOGSLAsd +
  Model Df Model Dev Df Deviance P(>|Chi|)
1        8    -87129
2       11    -86451  3   677.77 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

A similar test

```
> out4 <- aster(resp ~ varb + BLK + LOGLVSsd + LOGSLAsd + STG1Nsd,
+     pred, fam, varb, id, root, data = chamae, famlist = famlist)
> summary(out4)

Call:
aster.formula(formula = resp ~ varb + BLK + LOGLVSsd + LOGSLAsd +
    STG1Nsd, pred = pred, fam = fam, varvar = varb, idvar = id,
    root = root, data = chamae, famlist = famlist)

              Estimate Std. Error  z value Pr(>|z|)
(Intercept) -2.001e+01  1.279e-01 -156.419  < 2e-16 ***
varbfruit    2.099e+01  1.281e-01  163.843  < 2e-16 ***
varbseed     2.044e+01  1.350e-01  151.416  < 2e-16 ***
BLK2         1.130e-03  5.948e-04    1.900   0.0575 .
BLK4         4.333e-03  5.544e-04    7.815 5.48e-15 ***
LOGLVSsd    -4.689e-03  7.523e-03   -0.623   0.5331
LOGSLAsd     1.974e-02  2.494e-02    0.792   0.4285
STG1Nsd      5.686e-02  2.379e-03   23.896  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> anova(out4, out6)

Analysis of Deviance Table
```

```
Model 1: resp ~ varb + BLK + LOGLVSsd + LOGSLAsd + STG1Nsd
Model 2: c("resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + STG1Nfr + LOGLVSsd + ", "    LOGSLAsd +
  Model Df Model Dev Df Deviance P(>|Chi|)
1        8    -86736
2       11    -86451  3   284.05 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

shows that the effect of these variables on fruit is significant.

Now we consider quadratic terms. Since the variable `STG1N` has only a few values

```
> sort(unique(chamae$STG1N))

[1] 1 2 3

> tabulate(chamae$STG1N)

[1] 3276  684 2745
```

there is little sense adding terms quadratic in this variable.

The test

```
> out7 <- aster(resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + I(LOGLVSfr^2) +
+     I(LOGSLAfr^2) + I(LOGLVSfr * LOGSLAfr) + STG1Nfr + LOGLVSsd +
+     LOGSLAsd + STG1Nsd,
+     pred, fam, varb, id, root, data = chamae, famlist = famlist)
> summary(out7, info.tol = 1e-9)

Call:
aster.formula(formula = resp ~ varb + BLK + LOGLVSfr + LOGSLAfr +
    I(LOGLVSfr^2) + I(LOGSLAfr^2) + I(LOGLVSfr * LOGSLAfr) +
    STG1Nfr + LOGLVSsd + LOGSLAsd + STG1Nsd, pred = pred, fam = fam,
    varvar = varb, idvar = id, root = root, data = chamae, famlist = famlist)

                         Estimate Std. Error  z value Pr(>|z|)
(Intercept)            -1.983e+01  1.276e-01 -155.356  < 2e-16 ***
varbfruit               2.035e+01  1.423e-01  143.038  < 2e-16 ***
varbseed                2.055e+01  1.343e-01  153.042  < 2e-16 ***
BLK2                    4.353e-04  5.780e-04    0.753 0.451411
BLK4                    2.571e-03  5.525e-04    4.653 3.27e-06 ***
LOGLVSfr                2.669e-01  2.684e-02    9.945  < 2e-16 ***
LOGSLAfr               -2.656e-01  7.612e-02   -3.490 0.000484 ***
I(LOGLVSfr^2)          -3.995e-02  4.432e-03   -9.015  < 2e-16 ***
I(LOGSLAfr^2)          -9.385e-02  3.928e-02   -2.389 0.016892 *
I(LOGLVSfr * LOGSLAfr)  5.397e-02  1.847e-02    2.922 0.003482 **
STG1Nfr                -1.930e-05  3.256e-04   -0.059 0.952739
```

```
LOGLVSsd               -1.213e-01  8.734e-03  -13.887  < 2e-16 ***
LOGSLAsd                1.873e-02  2.989e-02    0.627 0.530974
STG1Nsd                 5.857e-02  3.063e-03   19.122  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> anova(out6, out7)

Analysis of Deviance Table

Model 1: c("resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + STG1Nfr + LOGLVSsd + ", "    LOGSLAsd +
Model 2: c("resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + I(LOGLVSfr^2) + I(LOGSLAfr^2) + ", "
  Model Df Model Dev Df Deviance P(>|Chi|)
1       11    -86451
2       14    -86309  3    142.2 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

shows that there appears to be a quadratic effect on fruit. The similar
test

```
> out8 <- aster(resp ~ varb + BLK + LOGLVSsd + LOGSLAsd + I(LOGLVSsd^2) +
+     I(LOGSLAsd^2) + I(LOGLVSsd * LOGSLAsd) + STG1Nsd + LOGLVSfr +
+     LOGSLAfr + STG1Nfr,
+     pred, fam, varb, id, root, data = chamae, famlist = famlist)
> summary(out8, info.tol = 1e-9)

Call:
aster.formula(formula = resp ~ varb + BLK + LOGLVSsd + LOGSLAsd +
    I(LOGLVSsd^2) + I(LOGSLAsd^2) + I(LOGLVSsd * LOGSLAsd) +
    STG1Nsd + LOGLVSfr + LOGSLAfr + STG1Nfr, pred = pred, fam = fam,
    varvar = varb, idvar = id, root = root, data = chamae, famlist = famlist)

                         Estimate Std. Error  z value Pr(>|z|)
(Intercept)            -1.986e+01  1.277e-01 -155.519  < 2e-16 ***
varbfruit               2.079e+01  1.284e-01  161.912  < 2e-16 ***
varbseed                1.915e+01  2.737e-01   69.972  < 2e-16 ***
BLK2                    5.654e-04  5.918e-04    0.956   0.3393
BLK4                    2.399e-03  5.682e-04    4.223 2.41e-05 ***
LOGLVSsd                8.373e-01  1.283e-01    6.523 6.87e-11 ***
LOGSLAsd               -7.466e-01  4.056e-01   -1.841   0.0656 .
I(LOGLVSsd^2)          -1.608e-01  2.320e-02   -6.934 4.10e-12 ***
I(LOGSLAsd^2)          -1.553e-01  2.513e-01   -0.618   0.5365
I(LOGLVSsd * LOGSLAsd)  2.319e-01  1.081e-01    2.146   0.0319 *
STG1Nsd                 5.481e-02  3.021e-03   18.146  < 2e-16 ***
LOGLVSfr                2.130e-02  1.072e-03   19.875  < 2e-16 ***
LOGSLAfr                4.579e-03  4.197e-03    1.091   0.2753
```

```
STG1Nfr                 3.448e-04  3.286e-04    1.049   0.2941
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> anova(out6, out8)

Analysis of Deviance Table

Model 1: c("resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + STG1Nfr + LOGLVSsd + ", "    LOGSLAsd +
Model 2: c("resp ~ varb + BLK + LOGLVSsd + LOGSLAsd + I(LOGLVSsd^2) + I(LOGSLAsd^2) + ", "
  Model Df Model Dev Df Deviance P(>|Chi|)
1       11    -86451
2       14    -86376  3   75.526 2.795e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

shows that there appears to also be a quadratic effect on seed. And

```
> out9 <- aster(resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + I(LOGLVSfr^2) +
+     I(LOGSLAfr^2) + I(LOGLVSfr * LOGSLAfr) + STG1Nfr + LOGLVSsd + LOGSLAsd +
+     I(LOGLVSsd^2) + I(LOGSLAsd^2) + I(LOGLVSsd * LOGSLAsd) + STG1Nsd,
+     pred, fam, varb, id, root, data = chamae, famlist = famlist)
> summary(out9, info.tol = 1e-9)

Call:
aster.formula(formula = resp ~ varb + BLK + LOGLVSfr + LOGSLAfr +
    I(LOGLVSfr^2) + I(LOGSLAfr^2) + I(LOGLVSfr * LOGSLAfr) +
    STG1Nfr + LOGLVSsd + LOGSLAsd + I(LOGLVSsd^2) + I(LOGSLAsd^2) +
    I(LOGLVSsd * LOGSLAsd) + STG1Nsd, pred = pred, fam = fam,
    varvar = varb, idvar = id, root = root, data = chamae, famlist = famlist)

                         Estimate Std. Error  z value Pr(>|z|)
(Intercept)            -1.982e+01  1.278e-01 -155.119  < 2e-16 ***
varbfruit               2.040e+01  1.432e-01  142.447  < 2e-16 ***
varbseed                2.001e+01  2.822e-01   70.909  < 2e-16 ***
BLK2                    4.391e-04  5.791e-04    0.758 0.448356
BLK4                    2.550e-03  5.535e-04    4.607 4.09e-06 ***
LOGLVSfr                2.245e-01  2.895e-02    7.755 8.84e-15 ***
LOGSLAfr               -2.437e-01  8.852e-02   -2.753 0.005905 **
I(LOGLVSfr^2)          -3.347e-02  4.740e-03   -7.061 1.65e-12 ***
I(LOGSLAfr^2)          -1.045e-01  4.704e-02   -2.221 0.026357 *
I(LOGLVSfr * LOGSLAfr)  3.845e-02  2.046e-02    1.880 0.060156 .
STG1Nfr                 6.362e-05  3.215e-04    0.198 0.843148
LOGLVSsd                3.487e-01  1.332e-01    2.619 0.008831 **
LOGSLAsd                3.109e-02  4.538e-01    0.069 0.945379
I(LOGLVSsd^2)          -8.210e-02  2.442e-02   -3.362 0.000774 ***
I(LOGSLAsd^2)           1.981e-01  2.861e-01    0.692 0.488793
```

```
I(LOGLVSsd * LOGSLAsd)  1.217e-01  1.156e-01    1.052 0.292609
STG1Nsd                 5.695e-02  3.072e-03   18.540  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> anova(out6, out7, out9)

Analysis of Deviance Table

Model 1: c("resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + STG1Nfr + LOGLVSsd + ", "    LOGSLAsd +
Model 2: c("resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + I(LOGLVSfr^2) + I(LOGSLAfr^2) + ", "
Model 3: c("resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + I(LOGLVSfr^2) + I(LOGSLAfr^2) + ", "
  Model Df Model Dev Df Deviance P(>|Chi|)
1       11    -86451
2       14    -86309  3  142.203 < 2.2e-16 ***
3       17    -86293  3   16.745 0.0007974 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> anova(out6, out8, out9)

Analysis of Deviance Table

Model 1: c("resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + STG1Nfr + LOGLVSsd + ", "    LOGSLAsd +
Model 2: c("resp ~ varb + BLK + LOGLVSsd + LOGSLAsd + I(LOGLVSsd^2) + I(LOGSLAsd^2) + ", "
Model 3: c("resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + I(LOGLVSfr^2) + I(LOGSLAfr^2) + ", "
  Model Df Model Dev Df Deviance P(>|Chi|)
1       11    -86451
2       14    -86376  3   75.526 2.795e-16 ***
3       17    -86293  3   83.422 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Shows that the model that is quadratic in the effects on both fruit and seed is supported by the data. There is some question about these because the Fisher information is close to singular (as evidenced by our need to supply the `info.tol` argument to the `summary` command), but we will go with `out9` as out “best fitting” model.

3.3 Maximum Likelihood Estimation of Size

The `aster` function does not calculate the correct likelihood when the size parameters are considered unknown, because it drops terms that do not involve the exponential family parameters. However, the full log likelihood is easily calculated in R.

```
> x <- out9$x
> logl <- function(alpha.fruit, alpha.seed, theta, x) {
+     x.fecund <- x[ , 1]
+     theta.fecund <- theta[ , 1]
+     p.fecund <- 1 / (1 + exp(- theta.fecund))
+     logl.fecund <- sum(dbinom(x.fecund, 1, p.fecund, log = TRUE))
+     x.fruit <- x[x.fecund == 1, 2]
+     theta.fruit <- theta[x.fecund == 1, 2]
+     p.fruit <- (- expm1(theta.fruit))
+     logl.fruit <- sum(dnbinom(x.fruit, size = alpha.fruit,
+         prob = p.fruit, log = TRUE) - pnbinom(2, size = alpha.fruit,
+         prob = p.fruit, lower.tail = FALSE, log = TRUE))
+     x.seed <- x[x.fecund == 1, 3]
+     theta.seed <- theta[x.fecund == 1, 3]
+     p.seed <- (- expm1(theta.seed))
+     logl.seed <- sum(dnbinom(x.seed, size = alpha.seed,
+         prob = p.seed, log = TRUE) - pnbinom(0, size = alpha.seed,
+         prob = p.seed, lower.tail = FALSE, log = TRUE))
+     logl.fecund + logl.fruit + logl.seed
+ }
```

We then calculate the profile likelihood for the two size parameters (`alpha.fruit` and `alpha.seed`), maximizing over the other parameters. Evaluating the profile log likelihood on a grid of points. We do not do this if the results would be the same as we got last time and have stored in the variable `logl.seq`.

```
> ok <- exists("alpha.fruit.save") && (alpha.fruit.save == alpha.fruit) &&
+     exists("alpha.seed.save") && (alpha.seed.save == alpha.seed) &&
+     exists("coef.save") && isTRUE(all.equal(coef.save, coefficients(out9)))
> print(ok)

[1] FALSE

> alpha.fruit.seq <- seq(1.5, 3.5, 0.25)
> alpha.seed.seq <- seq(10, 30, 0.5)
> if (! ok) {
+     logl.seq <- matrix(NA, nrow = length(alpha.fruit.seq),
+         ncol = length(alpha.seed.seq))
+     for (i in 1:length(alpha.fruit.seq)) {
+         for (j in 1:length(alpha.seed.seq)) {
+             famlist.seq <- famlist
+             famlist.seq[[3]] <- fam.truncated.negative.binomial(size =
+                 alpha.seed.seq[j], truncation = 0)
+             famlist.seq[[4]] <- fam.truncated.negative.binomial(size =
+                 alpha.fruit.seq[i], truncation = 2)
+             out9.seq <- aster(out9$formula, pred, fam, varb, id, root,
+                 data = chamae, famlist = famlist.seq, parm = out9$coefficients)
+             theta.seq <- predict(out9.seq, model.type = "cond",
+                 parm.type = "canon")
+             dim(theta.seq) <- dim(x)
+             logl.seq[i, j] <- logl(alpha.fruit.seq[i], alpha.seed.seq[j],
+                 theta.seq, x)
+         }
+     }
+ }
> ##### interpolate #####
> alpha.fruit.interp <- seq(min(alpha.fruit.seq), max(alpha.fruit.seq), 0.01)
> alpha.seed.interp <- seq(min(alpha.seed.seq), max(alpha.seed.seq), 0.01)
> logl.foo <- matrix(NA, nrow = length(alpha.fruit.interp),
+     ncol = length(alpha.seed.seq))
> for (i in 1:length(alpha.seed.seq))
+     logl.foo[ , i] <- spline(alpha.fruit.seq, logl.seq[ , i],
+         n = length(alpha.fruit.interp))$y
> logl.bar <- matrix(NA, nrow = length(alpha.fruit.interp),
+     ncol = length(alpha.seed.interp))
> for (i in 1:length(alpha.fruit.interp))
+     logl.bar[i, ] <- spline(alpha.seed.seq, logl.foo[i, ],
+         n = length(alpha.seed.interp))$y
> imax.fruit <- row(logl.bar)[logl.bar == max(logl.bar)]
> imax.seed <- col(logl.bar)[logl.bar == max(logl.bar)]
> alpha.fruit.save <- alpha.fruit
> alpha.seed.save <- alpha.seed
> alpha.fruit <- alpha.fruit.interp[imax.fruit]
> alpha.seed <- alpha.seed.interp[imax.seed]
> coef.save <- coefficients(out9)
> ##### save #####
> if (! ok) {
+     save(alpha.fruit, alpha.seed, alpha.fruit.save, alpha.seed.save,
+         coef.save, logl.seq, file = "chamae-alpha.rda", ascii = TRUE)
+ }
```

At the end of this chunk we save the maximum likelihood estimates in a file which is read in at the beginning of this document. We also save some extra information so there is no need to do this step every time if there is no change in the alphas.

Figure 2 (page 15) shows the profile log likelihood for the size parameters.

3.4 The Fitness Landscape

If we had "aster-friendly" data in which expected fitness was a mean value parameter of the aster model, we could immediately cal-

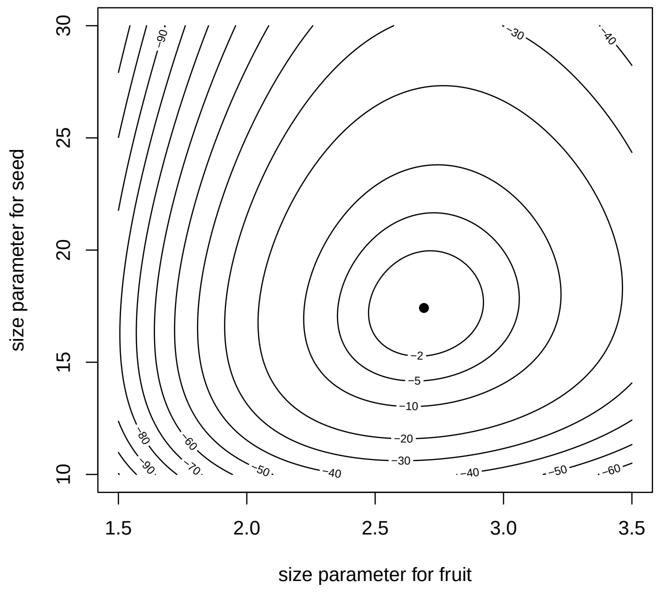

Figure 2: Profile log likelihood for size parameters for the negative binomial distributions of fruit and seed. Solid dot is maximum likelihood estimate.

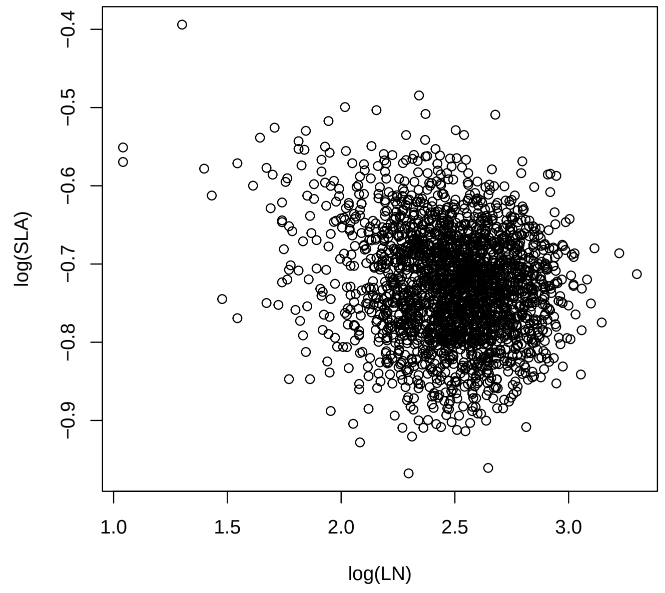

Figure 3: Scatterplot of phenotypic variables.

culate the fitness landscape using the predict function (as in Chapter 3 of TR 658). Unfortunately, fitness, which in this example we take to be the product of `fruit` and `seed` divided by 3 (because seeds were counted for three fruits), has expectation that is not a mean value parameter (because the expectation of a product is not the product of the expectations). Nevertheless, we can calculate its expectation by simulation (Monte Carlo).

We calculate for just one value of `BLK` and `STG1N`.

```
> theblk <- "1"
> thestg <- 1
```

Figure 3 (page 16) shows the scatter plots of the two phenotypic variables (`LOGLVS` and `LOGSLA`, labeled `LN` and `SLA` because that is what they are called in the paper). It is made by the following code.

```
> plot(chamaew$LOGLVS, chamaew$LOGSLA, xlab = "log(LN)", ylab = "log(SLA)")
```

The point of making the plot Figure 3 is that we want to add contour lines showing the estimated fitness landscape. To do that we first start with a grid of points across the figure.

```
> ufoo <- par("usr")
> nx <- 101
> ny <- 101
> z <- matrix(NA, nx, ny)
> x <- seq(ufoo[1], ufoo[2], length = nx)
> y <- seq(ufoo[3], ufoo[4], length = ny)
> xx <- outer(x, y^0)
> yy <- outer(x^0, y)
> xx <- as.vector(xx)
> yy <- as.vector(yy)
> n <- length(xx)
```

Then we create an appropriate `newdata` argument for the `predict.aster` function to "predict" at these points

```
> newdata <- data.frame(BLK = factor(rep(theblk, n), levels = levels(chamae$BLK)),
+     STG1N = rep(thestg, n), LOGLVS = xx, LOGSLA = yy, fecund = rep(1, n),
+     fruit = rep(3, n), seed = rep(5, n))
> renewdata <- reshape(newdata, varying = list(vars), direction = "long",
+     timevar = "varb", times = as.factor(vars), v.names = "resp")
> renewdata <- data.frame(renewdata, root = 1)
> foo <- as.numeric(as.character(renewdata$varb) == "fruit")
> renewdata$LOGLVSfr <- renewdata$LOGLVS * foo
> renewdata$LOGSLAfr <- renewdata$LOGSLA * foo
> renewdata$STG1Nfr <- renewdata$STG1N * foo
> foo <- as.numeric(as.character(renewdata$varb) == "seed")
> renewdata$LOGLVSsd <- renewdata$LOGLVS * foo
> renewdata$LOGSLAsd <- renewdata$LOGSLA * foo
> renewdata$STG1Nsd <- renewdata$STG1N * foo
```

Then we predict the conditional canonical parameter θ which is needed for simulation using the `raster` function.

```
> theta <- predict(out9, newdata = renewdata, varvar = varb, idvar = id,
+     root = root, model.type = "conditional", parm.type = "canonical")
> theta <- matrix(theta, nrow = nrow(newdata), ncol = ncol(out9$x))
```

Then we carry out a Monte Carlo approximation of the fitness landscape. Because this function may take a lot of time to run, we store the results in the current working directory, and simply load them if they exist.

```
> root <- matrix(1, nrow(theta), ncol(theta))
> nsim <- 5e5
> options(show.error.messages = FALSE, warn = -1)
> try(load("zzz.rda"))
> options(show.error.messages = TRUE, warn = 0)
> ok <- exists("zfit") && exists("stime") && exists("nsim.save") &&
+     (nsim == nsim.save) && exists("theta.save") &&
+     isTRUE(all.equal(theta.save, theta))
```

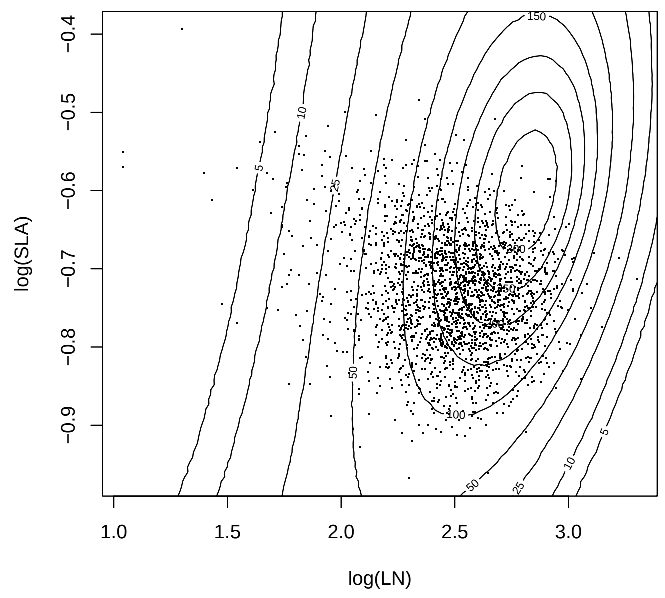

Figure 4: Scatterplot of phenotypic variables with contours of fitness landscape estimated by Monte Carlo.

```
> if (! ok) {
+     zfit <- double(n)
+     stime <- system.time(
+     for (isim in 1:nsim) {
+         xnew <- raster(theta, pred, fam, root = root, famlist = famlist)
+         zfit <- zfit + xnew[ , 2] * xnew[ , 3] / 3
+     }
+     )
+     zfit <- zfit / nsim
+     nsim.save <- nsim
+     theta.save <- theta
+     save(zfit, nsim.save, theta.save, stime, file = "zzz.rda")
+ }
```

The vector `zfit` is the Monte Carlo estimate; Figure 4 (page 18), which is made by the following code, shows it.

```
> plot(chamaew$LOGLVS, chamaew$LOGSLA, xlab = "log(LN)", ylab = "log(SLA)", pch = ".")
> zfit <- matrix(zfit, nrow = length(x))
> contour(x, y, zfit, add = TRUE)
> contour(x, y, zfit, levels = c(5, 10, 25), add = TRUE)
```

The time spent doing the Monte Carlo calculation of the likelihood surface was

```
> secs <- floor(stime[1])
> mins <- floor(secs / 60)
> secs <- secs - mins * 60
> hrs <- floor(mins / 60)
> mins <- mins - hrs * 60
```

0 hours, 14 minutes, and 24 seconds. We could easily use an even larger Monte Carlo sample size to get smoother curves in this figure.

3.5 Lande-Arnold Analysis

In contrast to the aster analysis, the Lande-Arnold analysis is very simple.

```
> chamaew$fit <- chamaew$fruit * chamaew$seed / 3
> lout <- lm(fit ~ LOGLVS + LOGSLA + STG1N + I(LOGLVS^2) +
+     I(LOGLVS * LOGSLA) + I(LOGSLA^2), data = chamaew)
> summary(lout)

Call:
lm(formula = fit ~ LOGLVS + LOGSLA + STG1N + I(LOGLVS^2) + I(LOGLVS *
    LOGSLA) + I(LOGSLA^2), data = chamaew)

Residuals:
    Min      1Q  Median      3Q     Max
-2143.9  -548.5  -227.9   188.3  7900.6

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         -7134.23    2113.14  -3.376 0.000748 ***
LOGLVS                906.60    1077.93   0.841 0.400407
LOGSLA             -13003.71    4797.30  -2.711 0.006767 **
STG1N                 524.26      24.53  21.375  < 2e-16 ***
I(LOGLVS^2)           437.02     221.02   1.977 0.048125 *
I(LOGLVS * LOGSLA)   2620.24    1235.00   2.122 0.033977 *
I(LOGSLA^2)         -5397.98    3278.27  -1.647 0.099782 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1057 on 2228 degrees of freedom
Multiple R-squared:  0.2012,       Adjusted R-squared:  0.1991
F-statistic: 93.54 on 6 and 2228 DF,  p-value: < 2.2e-16
```

The information contained in the printout of `summary(lout)` with the exception of the `Estimate` column is invalid because the OLS model assumptions are not satisfied, as acknowledged by Etterson and Shaw (2001) and Etterson (2004). All we know about the statistical properties of these estimators is that they are best linear unbiased by the Gauss-Markov theorem (Lindgren, 1993, p. 510). We know nothing about their sampling distribution except what we could learn by simulating the aster model. Therefore measures of statistical significance including standard errors (`Std. Error` column), t-statistics (`t value` column), and P-values (`Pr(>|t|)` column) are erroneous.

Figure 5 (page 21), which is made by the following code, shows the best quadratic approximation to the fitness landscape fit above by multiple regression together with the estimate from the aster model from Figure 4. It is made by the following code, first the prediction

```
> zzols <- predict(lout, newdata = data.frame(LOGLVS = xx, LOGSLA = yy,
+     STG1N = rep(thestg, length(xx))))
```

```
> plot(chamaew$LOGLVS, chamaew$LOGSLA, xlab = "log(LN)", ylab = "log(SLA)", pch = ".")
> contour(x, y, zfit, add = TRUE)
> contour(x, y, zfit, levels = c(5, 10, 25), add = TRUE)
> zzols <- matrix(zzols, nrow = length(x))
> contour(x, y, zzols, add = TRUE, lty = "dotted")
```

Note that fitness is a positive quantity. Hence the negative contours in the best quadratic approximation are nonsense, although they are the inevitable result of approximating a surface that is not close to quadratic with a quadratic function. Note also that the best quadratic approximation has a saddle point and no maximum, whereas it appears that the actual fitness landscape does have a maximum, albeit near the edge of the distribution of phenotypes. Apparently, the saddle point is the result of the quadratic function trying to be nearly flat on the left hand side of the figure (a quadratic function cannot have an asymptote; the saddle point is the next best thing). A quadratic function cannot have both a saddle point and a maximum; it has to choose one or the other. Unfortunately, least squares makes the wrong choice from the biological point of view. It is more important to get the maximum right than the flat spot (where fitness is close to zero).

3.6 Goodness of Fit

In this section we examine three issues. Is the assumed conditional independence of `fruit` and `seed` given `fecund == 1` correct? Are the assumed conditional distributions for `fruit` and `seed` given `fecund == 1` correct?

3.6.1 Conditional Independence of Fruit and Seed

We tackle the easiest first. Easy in a sense because impossible. We cannot test for independence. The best we can do is a nonparametric

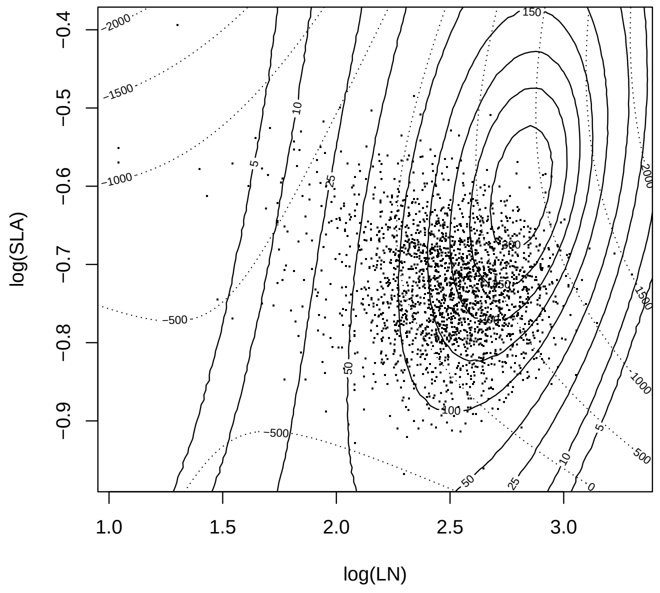

Figure 5: Scatterplot of phenotypic variables with contours of fitness landscape estimated by Monte Carlo (solid) and the best quadratic approximation (dotted).

test for lack of correlation.

```
> woof <- chamaew$fruit[chamaew$fecund == 1]
> meow <- chamaew$seed[chamaew$fecund == 1]
> cout <- cor.test(woof, meow, method = "kendall")
> print(cout)

        Kendall's rank correlation tau

data:  woof and meow
z = 4.0141, p-value = 5.967e-05
alternative hypothesis: true tau is not equal to 0
sample estimates:
       tau
0.08651135
```

The correlation (Kendall's tau) is statistically significantly different from zero, but perhaps, at 0.087 not practically significant. In any case, having no way put dependence in our aster model (other than the dependence induced by the predecessor-successor relationships indicated by the graphical model), we proceed as if not practically significant. Figure 6 (page 23) shows the scatter plot of the fitted mean value parameter (for each individual) versus the observed value for fruit count.

3.6.2 Conditional of Fruit given Nonzero Fitness

Residual analysis of generalized linear models (GLM) is tricky. (Our aster model becomes a GLM when we consider only the conditional distribution associated with one arrow.) Many different residuals have been proposed (Davison and Snell, 1991). We start with the simplest, so called Pearson residuals.

```
> xi.hat <- predict(out9, model.type = "cond", parm.type = "mean")
> xi.hat <- matrix(xi.hat, nrow = nrow(out9$x), ncol = ncol(out9$x))

> range(woof)

[1]   3 781

> nwoof <- length(woof)
> woof.theta <- theta[chamaew$fecund == 1, 2]
> woof.xi <- xi.hat[chamaew$fecund == 1, 2]
> wgrad <- double(nwoof)
> winfo <- double(nwoof)
> for (i in 1:nwoof) {
+     wgrad[i] <- famfun(famlist[[4]], deriv = 1, woof.theta[i])
```

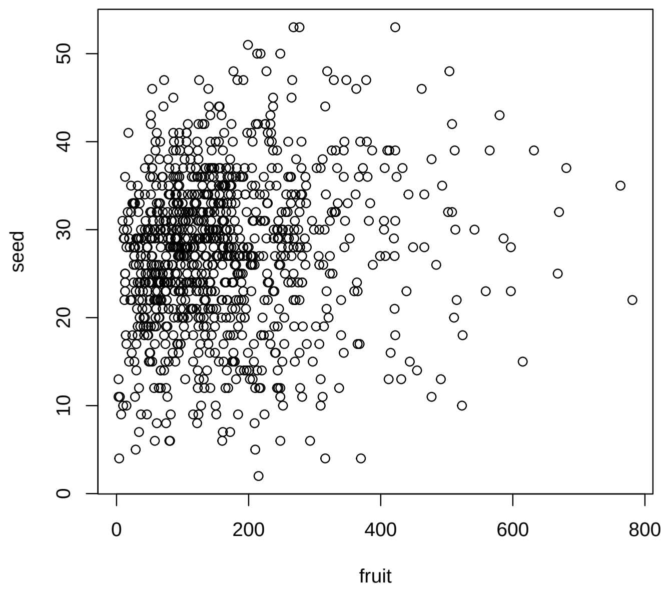

Figure 6: Scatter plot fruit count versus seed count conditioned on nonzero fitness.

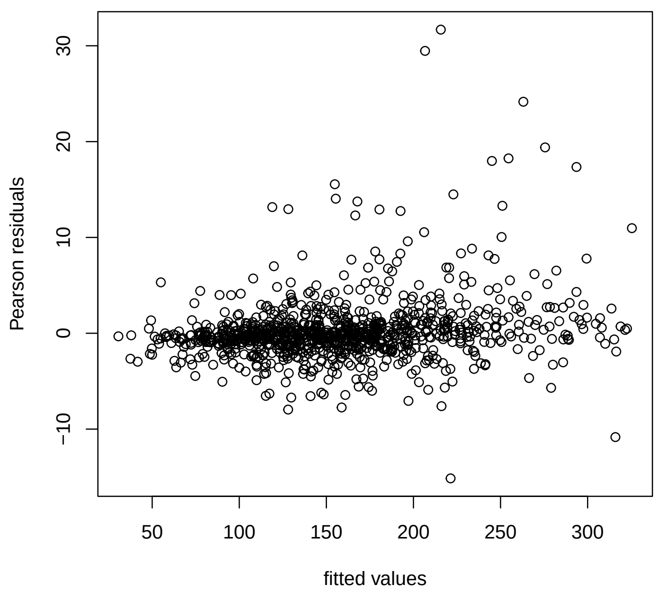

Figure 7: Pearson residuals for fruit count given nonzero fitness plotted against fitted values.

```
+     winfo[i] <- famfun(famlist[[4]], deriv = 2, woof.theta[i])
+ }
> all.equal(woof.xi, wgrad)

[1] "Mean relative difference: 0.5246045"

> pearson <- (woof - woof.xi) / sqrt(winfo)
```

Figure 7 (page 24) shows the scatter plot of the Pearson residuals for fruit count plotted against the expected fruit count given that fruit count is nonzero (for each individual) for individuals with nonzero fitness only.

Figure 7 is not perfect. There are 22 individuals with Pearson residual greater than 10 in absolute value and an additional 56 individuals with Pearson residual between 5 and 10 in absolute value (out of 994 total residuals). One does not expect Pearson residuals for a generalized linear model, much less an aster model, to behave as well for normal-theory linear models, but the lack of fit here is a bit worrying. The large positive "outliers" (which are not outliers in the sense of being bad data) indicate that our negative binomial model does not perfectly model these data (the negative binomial model is, however, an enormous improvement over the Poisson model, which is not shown).

3.6.3 Conditional of Seed given Nonzero Fitness

Now we do the analogous plot of the conditional distribution of `seed` given nonzero fitness.

```
> range(meow)
```

```
[1]  2 53
```

```
> nmeow <- length(meow)
> meow.theta <- theta[chamaew$fecund == 1, 3]
> meow.xi <- xi.hat[chamaew$fecund == 1, 3]
> wgrad <- double(nmeow)
> winfo <- double(nmeow)
> for (i in 1:nmeow) {
+     wgrad[i] <- famfun(famlist[[3]], deriv = 1, meow.theta[i])
+     winfo[i] <- famfun(famlist[[3]], deriv = 2, meow.theta[i])
+ }
> all.equal(meow.xi, wgrad)
```

```
[1] "Mean relative difference: 0.246243"
```

```
> pearson <- (meow - meow.xi) / sqrt(winfo)
```

Figure 8 (page 26) shows the scatter plot of the Pearson residuals for seed count plotted against the expected seed count given that fruit count is nonzero (for each individual) for individuals with nonzero fitness only. There are no obvious problem with Figure 8. Certainly, it is much less troubling than Figure 7.

3.7 OLS Diagnostic Plots

Although unnecessary because we know the assumptions justifying OLS are badly violated, here are some diagnostic plots for the OLS regression.

Figure 9 (page 27) shows the plot of residuals versus fitted values made by the R statement

```
> plot(lout, which = 1, add.smooth = FALSE, id.n = 0,
+     sub.caption = "", caption = "")
```

Figure 10 (page 27) shows the Normal Q-Q (quantile-quantile) plot made by the R statement

```
> plot(lout, which = 2, id.n = 0, sub.caption = "")
```

Clearly the errors are highly non-normal.

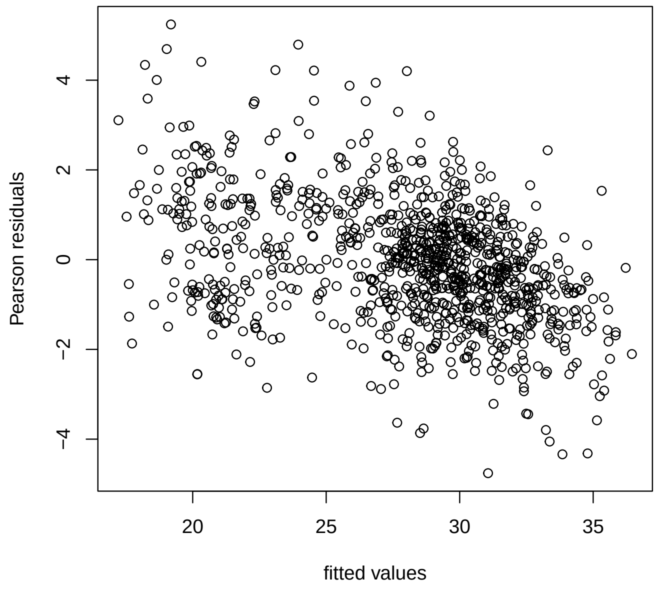

Figure 8: Pearson residuals for seed count given nonzero fitness plotted against fitted values.

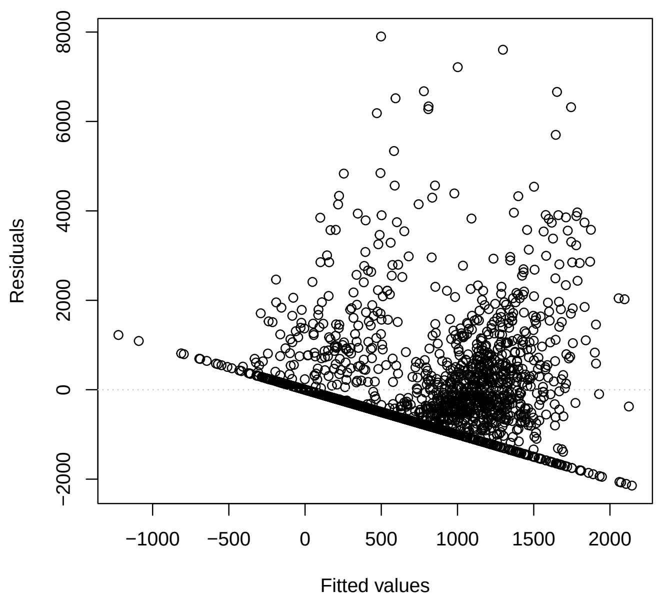

Figure 9: Residuals versus Fitted plot for OLS fit with blocks.

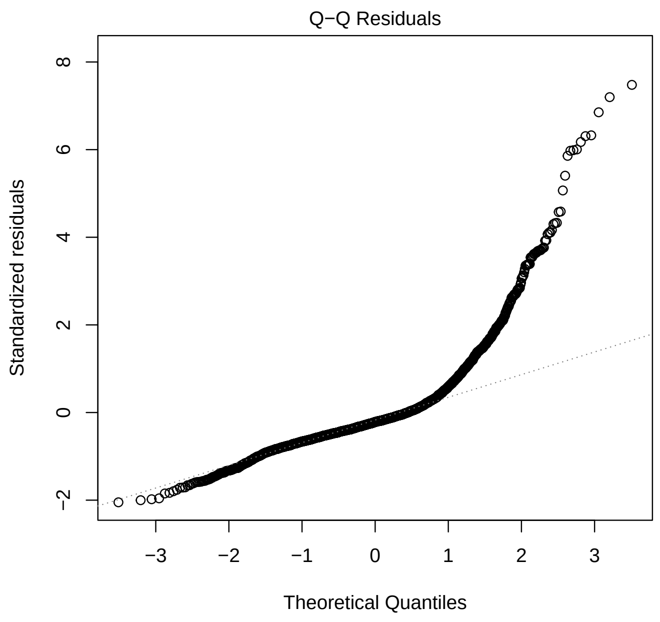

Figure 10: Normal Q-Q plot for OLS fit with blocks.

4 Analysis involving a Single Component of Fitness

Before doing anything, we remove all the variables generated in the preceding analyses.

```
> rm(list = ls())
> ls(all.names = TRUE)

[1] ".Random.seed"
```

4.1 Data

We reanalyze a subset of the data analyzed by Etterson and Shaw (2001). These data are in the `chamae2` dataset in the `aster` contributed package to the R statistical computing environment. This dataset is restricted to the Minnesota site of the original (larger) data.

These data are already in "long" format, no need to use the `reshape` function on them to do aster analysis. We will, however, need the "wide" format for Lande-Arnold analysis. So we do that, before making any changes (we will add newly defined variables) to `chamae2`.

```
> library(aster)
> data(chamae2)
> chamae2w <- reshape(chamae2, direction = "wide", timevar = "varb",
+     v.names = "resp", varying = list(levels(chamae2$varb)))
> names(chamae2w)

[1] "id"     "root"   "STG1N"  "LOGLVS" "LOGSLA" "BLK"    "fecund"
[8] "fruit"
```

We model fruit count as having a zero-inflated negative binomial distribution. The zero inflation allows for excess (or deficit) of individuals having zero fruit (over and above the small number of zeros that would occur if the distribution were pure negative binomial). In an aster model this is done by having a Bernoulli node followed by a zero-truncated negative binomial node (each individual having a simple graph with two nodes). This means the event that an individual has one or more fruits is modeled as Bernoulli, and the distribution of the number of fruit given that the number is at least one is modeled as zero-truncated negative binomial.

5 Aster Analysis

We need to choose the non-exponential-family parameter (size) for the negative binomial distribution, since the `aster` package only does maximum likelihood for exponential family parameters. We start with the following value, which was chosen with knowledge of the maximum likelihood estimate for this parameter, which we find in Section 5.1. The value that is found then is written out to a file and loaded here if the file exists, so after several runs (of `Sweave`) we are reading in here the maximum likelihood value of this non-exponential-family parameter.

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("chamae2-alpha.rda"))
> options(show.error.messages = TRUE, warn = 0)
> ok <- exists("alpha.fruit")
> if (! ok) {
+     alpha.fruit <- 3.0
+ }
> print(alpha.fruit)
```

```
[1] 2.51
```

Then we set up the aster model framework.

```
> vars <- c("fecund", "fruit")
> pred <- c(0, 1)
> famlist <- list(fam.bernoulli(),
+     fam.truncated.negative.binomial(size = alpha.fruit, truncation = 0))
> fam <- c(1,2)
```

We can now fit our first aster model.

```
> out1 <- aster(resp ~ varb + BLK, pred, fam, varb, id, root,
+     data = chamae2, famlist = famlist)
> summary(out1, show.graph = TRUE)
```

```
Call:
aster.formula(formula = resp ~ varb + BLK, pred = pred, fam = fam,
    varvar = varb, idvar = id, root = root, data = chamae2, famlist = famlist)


Graphical Model:
 variable predecessor
 fecund   root
 fruit    fecund
 family
 bernoulli
 truncated.negative.binomial(size = 2.51, truncation = 0)

             Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.6045304  0.1355057 -48.740  < 2e-16 ***
varbfruit    7.5915205  0.1355518  56.005  < 2e-16 ***
```

```
BLK2        -0.0010722  0.0004343  -2.469   0.0136 *
BLK4         0.0021980  0.0003878   5.668 1.44e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The “response” `resp` is a numeric vector containing all the response variables (`fecund` and `fruit`). The “predictor” `varb` is a factor with two levels distinguishing with `resp` which original response variable an element is. The predictor `BLK` is block within the field where the plants were grown.

Now we add phenotypic variables.

```
> out2 <- aster(resp ~ varb + BLK + LOGLVS + LOGSLA + STG1N,
+     pred, fam, varb, id, root, data = chamae2, famlist = famlist)
> summary(out2, info.tol = 1e-9)

Call:
aster.formula(formula = resp ~ varb + BLK + LOGLVS + LOGSLA +
    STG1N, pred = pred, fam = fam, varvar = varb, idvar = id,
    root = root, data = chamae2, famlist = famlist)

              Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.1663282  0.1362857 -45.246  < 2e-16 ***
varbfruit    7.1098537  0.1365495  52.068  < 2e-16 ***
BLK2        -0.0023311  0.0004215  -5.530 3.20e-08 ***
BLK4        -0.0006305  0.0004080  -1.545  0.12230
LOGLVS       0.0161360  0.0004771  33.821  < 2e-16 ***
LOGSLA      -0.0067775  0.0024641  -2.750  0.00595 **
STG1N       -0.0011344  0.0001966  -5.771 7.89e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

An alternative model with the same number of parameters as `out2` puts in the regression coefficients only at the “fitness” level (here `fruit`). This is similar to the example in Geyer, et al. (2007). Because we are fitting an unconditional aster model, the effects of these terms are passed down to `fecund`.

```
> foo <- as.numeric(as.character(chamae2$varb) == "fruit")
> chamae2$LOGLVSfr <- chamae2$LOGLVS * foo
> chamae2$LOGSLAfr <- chamae2$LOGSLA * foo
> chamae2$STG1Nfr <- chamae2$STG1N * foo
> out6 <- aster(resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + STG1Nfr,
+     pred, fam, varb, id, root, data = chamae2, famlist = famlist)
> summary(out6, info.tol = 1e-9)

Call:
aster.formula(formula = resp ~ varb + BLK + LOGLVSfr + LOGSLAfr +
    STG1Nfr, pred = pred, fam = fam, varvar = varb, idvar = id,
    root = root, data = chamae2, famlist = famlist)

              Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.1263979  0.1364738 -44.891  < 2e-16 ***
varbfruit    7.0699154  0.1367763  51.690  < 2e-16 ***
BLK2        -0.0023306  0.0004215  -5.530 3.21e-08 ***
BLK4        -0.0006308  0.0004080  -1.546  0.12212
LOGLVSfr     0.0161366  0.0004771  33.819  < 2e-16 ***
LOGSLAfr    -0.0067875  0.0024647  -2.754  0.00589 **
STG1Nfr     -0.0011349  0.0001966  -5.772 7.85e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

It is not possible to compare `out2` and `out6` by standard methods (likelihood ratio test) because the models are not nested. They seem to fit equally well, and `out6` more directly models the relation of fitness (here defined as `fruit`) to phenotypic variables.

Now we consider quadratic terms. Since the variable `STG1N` has only a few values

```
> sort(unique(chamae2$STG1N))

[1] 1 2 3

> tabulate(chamae2$STG1N)

[1] 2188  456 1834
```

there is little sense adding terms quadratic in this variable.

The test

```
> out7 <- aster(resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + I(LOGLVSfr^2) +
+     I(LOGSLAfr^2) + I(LOGLVSfr * LOGSLAfr) + STG1Nfr,
+     pred, fam, varb, id, root, data = chamae2, famlist = famlist)
> summary(out7, info.tol = 1e-9)

Call:
aster.formula(formula = resp ~ varb + BLK + LOGLVSfr + LOGSLAfr +
    I(LOGLVSfr^2) + I(LOGSLAfr^2) + I(LOGLVSfr * LOGSLAfr) +
    STG1Nfr, pred = pred, fam = fam, varvar = varb, idvar = id,
    root = root, data = chamae2, famlist = famlist)

                         Estimate Std. Error z value Pr(>|z|)
(Intercept)            -5.637e+00  1.462e-01 -38.555  < 2e-16 ***
varbfruit               6.296e+00  1.558e-01  40.410  < 2e-16 ***
BLK2                   -2.596e-03  4.114e-04  -6.309 2.80e-10 ***
```

```
BLK4                   -7.296e-06  3.805e-04  -0.019 0.984704
LOGLVSfr                1.669e-01  1.537e-02  10.856  < 2e-16 ***
LOGSLAfr               -2.382e-01  5.371e-02  -4.435 9.19e-06 ***
I(LOGLVSfr^2)          -2.398e-02  2.458e-03  -9.758  < 2e-16 ***
I(LOGSLAfr^2)          -1.099e-01  3.059e-02  -3.593 0.000326 ***
I(LOGLVSfr * LOGSLAfr)  2.848e-02  1.214e-02   2.347 0.018921 *
STG1Nfr                -1.250e-03  1.890e-04  -6.617 3.67e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> anova(out6, out7)

Analysis of Deviance Table

Model 1: resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + STG1Nfr
Model 2: c("resp ~ varb + BLK + LOGLVSfr + LOGSLAfr + I(LOGLVSfr^2) + I(LOGSLAfr^2) + ", "
  Model Df Model Dev Df Deviance P(>|Chi|)
1        7    -58400
2       10    -58187  3   213.22 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

shows that there appears to be a quadratic effect on fruit.

5.1 Maximum Likelihood Estimation of Size

The `aster` function does not calculate the correct likelihood when the size parameters are considered unknown, because it drops terms that do not involve the exponential family parameters. However, the full log likelihood is easily calculated in R.

```
> x <- out7$x
> logl <- function(alpha.fruit, theta, x) {
+     x.fecund <- x[ , 1]
+     theta.fecund <- theta[ , 1]
+     p.fecund <- 1 / (1 + exp(- theta.fecund))
+     logl.fecund <- sum(dbinom(x.fecund, 1, p.fecund, log = TRUE))
+     x.fruit <- x[x.fecund == 1, 2]
+     theta.fruit <- theta[x.fecund == 1, 2]
+     p.fruit <- (- expm1(theta.fruit))
+     logl.fruit <- sum(dnbinom(x.fruit, size = alpha.fruit,
+         prob = p.fruit, log = TRUE) - pnbinom(0, size = alpha.fruit,
+         prob = p.fruit, lower.tail = FALSE, log = TRUE))
+     logl.fecund + logl.fruit
+ }
```

We then calculate the profile likelihood for the size parameter `alpha.fruit` maximizing over the other parameters, evaluating the profile log likelihood on a grid of points. We do not do this if the results would be the same as we got last time and have stored in the variable `logl.seq`.

```
> ok <- exists("alpha.fruit.save") && (alpha.fruit.save == alpha.fruit) &&
+     exists("coef.save") && isTRUE(all.equal(coef.save, coefficients(out7)))
> print(ok)

[1] FALSE

> alpha.fruit.seq <- seq(1.5, 4.5, 0.25)
> if (! ok) {
+ logl.seq <- double(length(alpha.fruit.seq))
+     for (i in 1:length(alpha.fruit.seq)) {
+         famlist.seq <- famlist
+         famlist.seq[[2]] <- fam.truncated.negative.binomial(size =
+             alpha.fruit.seq[i], truncation = 0)
+         out7.seq <- aster(out7$formula, pred, fam, varb, id, root,
+             data = chamae2, famlist = famlist.seq, parm = out7$coefficients)
+         theta.seq <- predict(out7.seq, model.type = "cond",
+             parm.type = "canon")
+         dim(theta.seq) <- dim(x)
+         logl.seq[i] <- logl(alpha.fruit.seq[i], theta.seq, x)
+     }
+ }
> ##### interpolate #####
> alpha.foo <- seq(min(alpha.fruit.seq), max(alpha.fruit.seq), 0.01)
> logl.foo <- spline(alpha.fruit.seq, logl.seq, n = length(alpha.foo))$y
> imax <- seq(along = alpha.foo)[logl.foo == max(logl.foo)]
> alpha.fruit.save <- alpha.fruit
> alpha.fruit <- alpha.foo[imax]
> coef.save <- coefficients(out7)
> ##### save #####
> if (! ok) {
+     save(alpha.fruit, alpha.fruit.save, coef.save, logl.seq,
+         file = "chamae2-alpha.rda", ascii = TRUE)
+ }
```

At the end of this chunk we save the maximum likelihood estimate in a file which is read in at the beginning of this document. We also save some extra information so there is no need to do this step every time if there is no change in the alpha.

Figure 11 (page 34) shows the profile log likelihood for the size parameter.

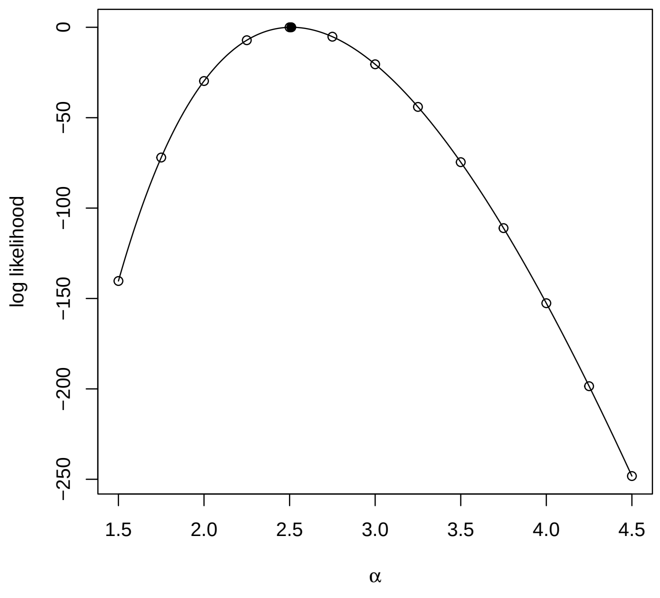

Figure 11: Profile log likelihood for size parameter for the (zero-truncated) negative binomial distribution of fruit. Hollow dots are points at which the log likelihood was evaluated exactly. Curve is the interpolating spline. Solid dot is maximum likelihood estimate.

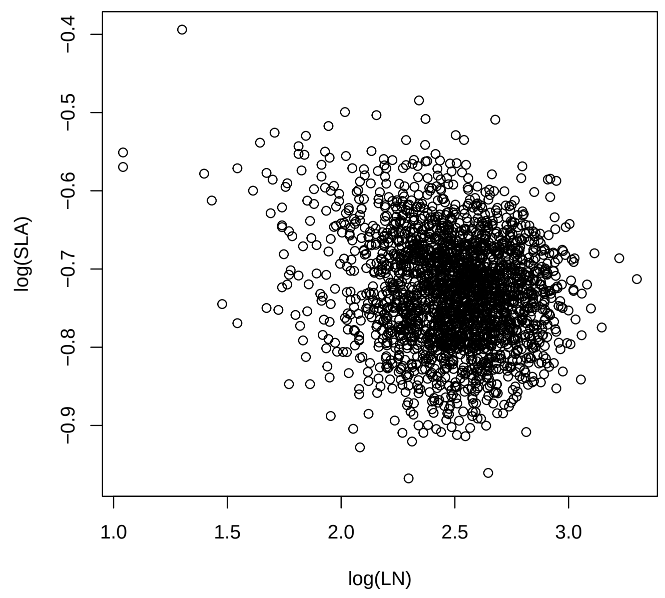

Figure 12: Scatterplot of phenotypic variables.

5.2 The Fitness Landscape

We calculate for just one value of `BLK` and `STG1N`.

```
> theblk <- "1"
> thestg <- 1
```

Figure 12 (page 35) shows the scatter plots of the two phenotypic variables (`LOGLVS` and `LOGSLA`, labeled `LN` and `SLA` because that is what they are called in the paper). It is made by the following code.

```
> plot(chamae2w$LOGLVS, chamae2w$LOGSLA, xlab = "log(LN)", ylab = "log(SLA)")
```

The point of making the plot Figure 12 is that we want to add contour lines showing the estimated fitness landscape. To do that we first start with a grid of points across the figure.

```
> ufoo <- par("usr")
> nx <- 101
> ny <- 101
> z <- matrix(NA, nx, ny)
> x <- seq(ufoo[1], ufoo[2], length = nx)
> y <- seq(ufoo[3], ufoo[4], length = ny)
> xx <- outer(x, y^0)
> yy <- outer(x^0, y)
> xx <- as.vector(xx)
> yy <- as.vector(yy)
> n <- length(xx)
```

Then we create an appropriate `newdata` argument for the `predict.aster` function to "predict" at these points

```
> newdata <- data.frame(
+     BLK = factor(rep(theblk, n), levels = levels(chamae2$BLK)),
+     STG1N = rep(thestg, n), LOGLVS = xx, LOGSLA = yy, fecund = rep(1, n),
+     fruit = rep(3, n))
> renewdata <- reshape(newdata, varying = list(vars), direction = "long",
+     timevar = "varb", times = as.factor(vars), v.names = "resp")
> renewdata <- data.frame(renewdata, root = 1)
> foo <- as.numeric(as.character(renewdata$varb) == "fruit")
> renewdata$LOGLVSfr <- renewdata$LOGLVS * foo
> renewdata$LOGSLAfr <- renewdata$LOGSLA * foo
> renewdata$STG1Nfr <- renewdata$STG1N * foo
```

Then we predict the unconditional mean value parameter τ, for which the "fruit" component is expected fitness.

```
> tau <- predict(out7, newdata = renewdata, varvar = varb, idvar = id,
+     root = root)
> tau <- matrix(tau, nrow = nrow(newdata), ncol = ncol(out7$x))
> dimnames(tau) <- list(NULL, vars)
> zfit <- tau[ , "fruit"]
```

Figure 13 (page 37), which is made by the following code, shows it.

```
> plot(chamae2w$LOGLVS, chamae2w$LOGSLA, xlab = "log(LN)", ylab = "log(SLA)", pch = ".")
> zfit <- matrix(zfit, nrow = length(x))
> contour(x, y, zfit, add = TRUE)
> contour(x, y, zfit, levels = c(5, 10, 25), add = TRUE)
```

5.3 Lande-Arnold Analysis

In contrast to the aster analysis, the Lande-Arnold analysis is very simple.

```
> lout <- lm(fruit ~ LOGLVS + LOGSLA + STG1N + I(LOGLVS^2) +
+     I(LOGLVS * LOGSLA) + I(LOGSLA^2), data = chamae2w)
> summary(lout)

Call:
lm(formula = fruit ~ LOGLVS + LOGSLA + STG1N + I(LOGLVS^2) +
```

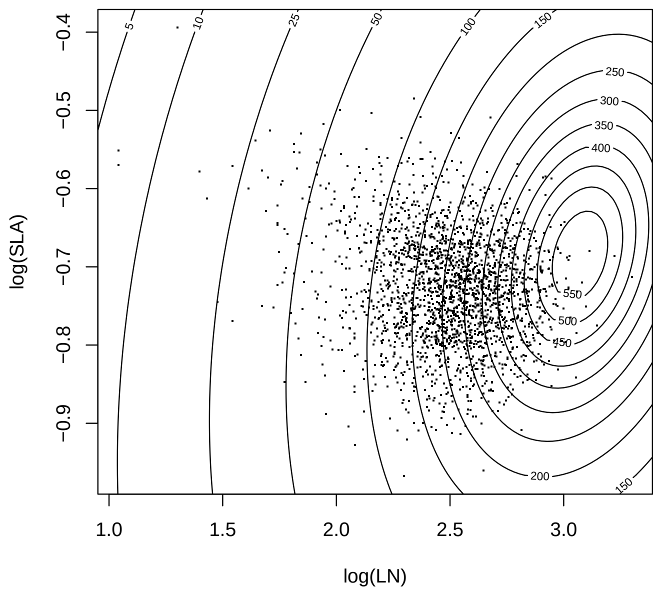

Figure 13: Scatterplot of phenotypic variables with contours of fitness landscape estimated by the aster model.

```
    I(LOGLVS * LOGSLA) + I(LOGSLA^2), data = chamae2w)

Residuals:
    Min      1Q  Median      3Q     Max 
-460.02  -67.90  -10.54   55.77  738.74 

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)          -19.220    244.336  -0.079 0.937309    
LOGLVS              -770.292    124.674  -6.178 7.67e-10 ***
LOGSLA             -2066.538    554.420  -3.727 0.000198 ***
STG1N                -17.364      2.835  -6.125 1.07e-09 ***
I(LOGLVS^2)          248.174     25.512   9.728  < 2e-16 ***
I(LOGLVS * LOGSLA)   153.662    142.506   1.078 0.281025    
I(LOGSLA^2)        -1150.074    379.128  -3.033 0.002445 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1 

Residual standard error: 122.3 on 2232 degrees of freedom
Multiple R-squared:  0.3557,	Adjusted R-squared:  0.3539 
F-statistic: 205.4 on 6 and 2232 DF,  p-value: < 2.2e-16 
```

The information contained in the printout of `summary(lout)` with the exception of the `Estimate` column is invalid because the OLS model assumptions are not satisfied, as acknowledged by Etterson and Shaw (2001) and Etterson (2004). All we know about the statistical properties of these estimators is that they are best linear unbiased by the Gauss-Markov theorem (Lindgren, 1993, p. 510). We know nothing about their sampling distribution except what we could learn by simulating the aster model. Therefore measures of statistical significance including standard errors (`Std. Error` column), t-statistics (`t value` column), and P-values (`Pr(>|t|)` column) are erroneous.

Figure 14 (page 39), which is made by the following code, shows the best quadratic approximation to the fitness landscape fit above by multiple regression together with the estimate from the aster model from Figure 13. It is made by the following code, first the prediction

```
> zzols <- predict(lout, newdata = data.frame(LOGLVS = xx, LOGSLA = yy,
+     STG1N = rep(thestg, length(xx))))
```

```
> plot(chamae2w$LOGLVS, chamae2w$LOGSLA, xlab = "log(LN)", ylab = "log(SLA)", pch = ".")
> contour(x, y, zfit, add = TRUE)
> contour(x, y, zfit, levels = c(5, 10, 25), add = TRUE)
> zzols <- matrix(zzols, nrow = length(x))
> contour(x, y, zzols, add = TRUE, lty = "dotted")
```

Note that fitness is a positive quantity. Hence the negative contours in the best quadratic approximation are nonsense, although they

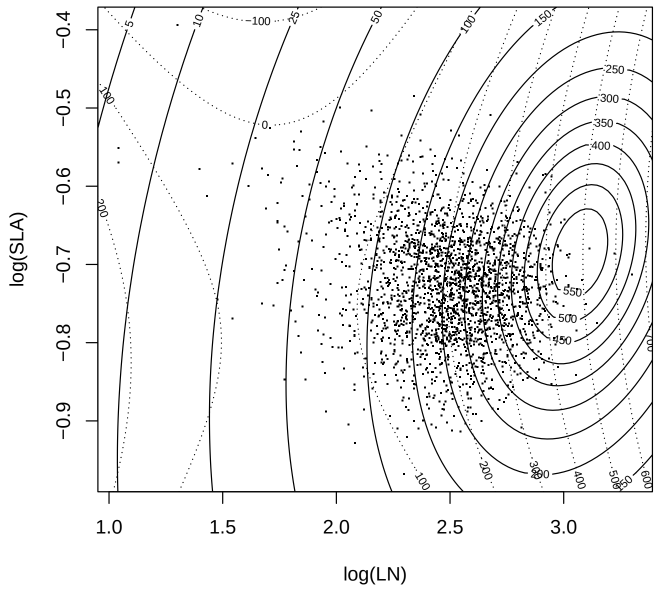

Figure 14: Scatterplot of phenotypic variables with contours of fitness landscape estimated by the aster model (solid) and the best quadratic approximation (dotted).

are the inevitable result of approximating a surface that is not close to quadratic with a quadratic function. Note also that the best quadratic approximation has a saddle point and no maximum, whereas it appears that the actual fitness landscape does have a maximum, albeit near the edge of the distribution of phenotypes. Apparently, the saddle point is the result of the quadratic function trying to be nearly flat on the left hand side of the figure (a quadratic function cannot have an asymptote; the saddle point is the next best thing). A quadratic function cannot have both a saddle point and a maximum; it has to choose one or the other. Unfortunately, least squares makes the wrong choice from the biological point of view. It is more important to get the maximum right than the flat spot (where fitness is close to zero).

5.4 Goodness of Fit

In this section we examine goodness of fit to the assumed conditional distributions for `fruit` given `fecund == 1` by looking at a residual plot.

Residual analysis of generalized linear models (GLM) is tricky. (Our aster model becomes a GLM when we consider only the conditional distribution associated with one arrow.) Many different residuals have been proposed (Davison and Snell, 1991). We start with the simplest, so called Pearson residuals.

```
> xi.hat <- predict(out7, model.type = "cond", parm.type = "mean")
> xi.hat <- matrix(xi.hat, nrow = nrow(out7$x), ncol = ncol(out7$x))
> theta.hat <- predict(out7, model.type = "cond", parm.type = "canon")
> theta.hat <- matrix(theta.hat, nrow = nrow(out7$x), ncol = ncol(out7$x))

> woof <- chamae2w$fruit[chamae2w$fecund == 1]
> range(woof)

[1]    1 1390

> nwoof <- length(woof)
> woof.theta <- theta.hat[chamae2w$fecund == 1, 2]
> woof.xi <- xi.hat[chamae2w$fecund == 1, 2]
> wgrad <- double(nwoof)
> winfo <- double(nwoof)
> for (i in 1:nwoof) {
+     wgrad[i] <- famfun(famlist[[2]], deriv = 1, woof.theta[i])
+     winfo[i] <- famfun(famlist[[2]], deriv = 2, woof.theta[i])
+ }
> all.equal(woof.xi, wgrad)

[1] TRUE
```

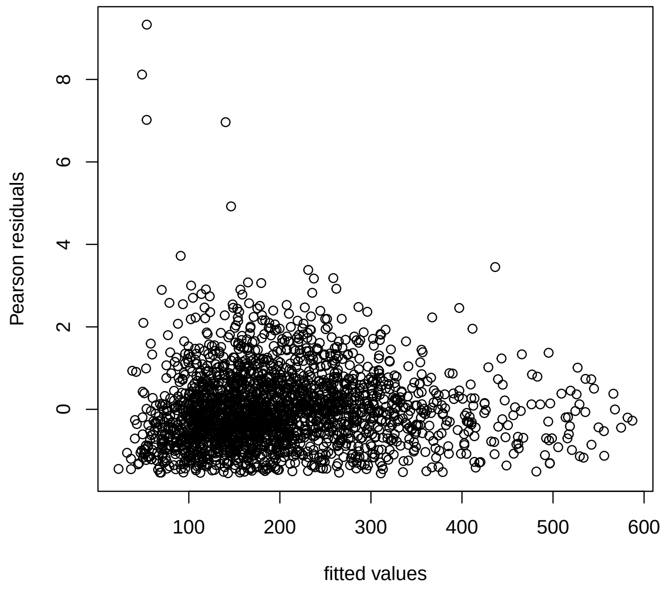

Figure 15: Pearson residuals for fruit count given nonzero fitness plotted against fitted values.

```
> pearson <- (woof - woof.xi) / sqrt(winfo)
```

Figure 15 (page 41) shows the scatter plot of the Pearson residuals for fruit count plotted against the expected fruit count given that fruit count is nonzero (for each individual) for individuals with nonzero fitness only.

Figure 15 is not perfect. There are 4 individuals with Pearson residual greater than 5 and an additional 9 individuals with Pearson residual between 3 and 5 (out of 2179 total residuals). There are 0 individuals with Pearson residual less than -3. One does not expect Pearson residuals for a generalized linear model, much less an aster model, to behave as well for normal-theory linear models, but the lack of fit here is a bit worrying. The large positive "outliers" (which are not outliers in the sense of being bad data) indicate that our negative binomial model does not perfectly model these data (the negative binomial model is, however, an enormous improvement over the Poisson model, which is not shown).

5.5 OLS Diagnostic Plots

Although unnecessary because we know the assumptions justifying OLS are badly violated, here are some diagnostic plots for the OLS

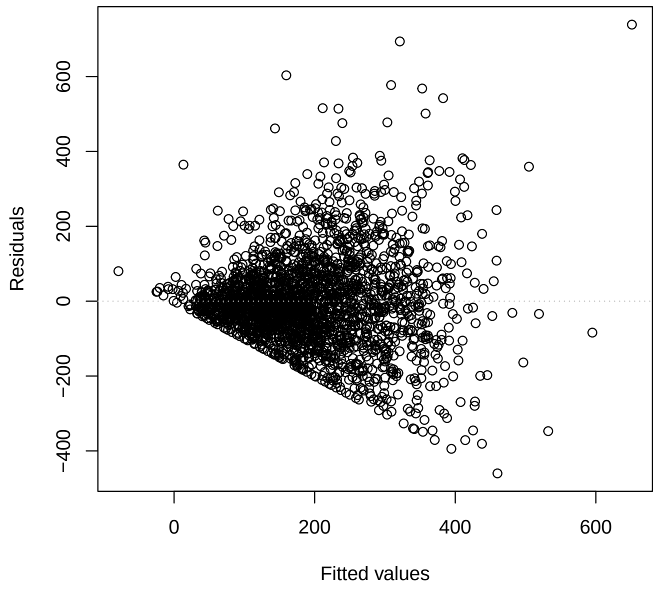

Figure 16: Residuals versus Fitted plot for OLS fit with blocks.

regression.

Figure 16 (page 42) shows the plot of residuals versus fitted values made by the R statement

```
> plot(lout, which = 1, add.smooth = FALSE, id.n = 0,
+     sub.caption = "", caption = "")
```

Figure 17 (page 43) shows the Normal Q-Q (quantile-quantile) plot made by the R statement

```
> plot(lout, which = 2, id.n = 0, sub.caption = "")
```

Clearly the errors are highly non-normal.

6 Discussion

Our two analyses, Section 3 and Section 4 are quite similar. The main results are similar: Figure 4 resembles Figure 13 and Figure 5 resembles Figure 14. The details are different, but the "big picture" is the same.

The main difference and the reason for doing the second analysis is to illustrate the analysis of an "aster-friendly" model where some linear combination of fitness components is deemed fitness, which leads to two important simplifications of the analysis

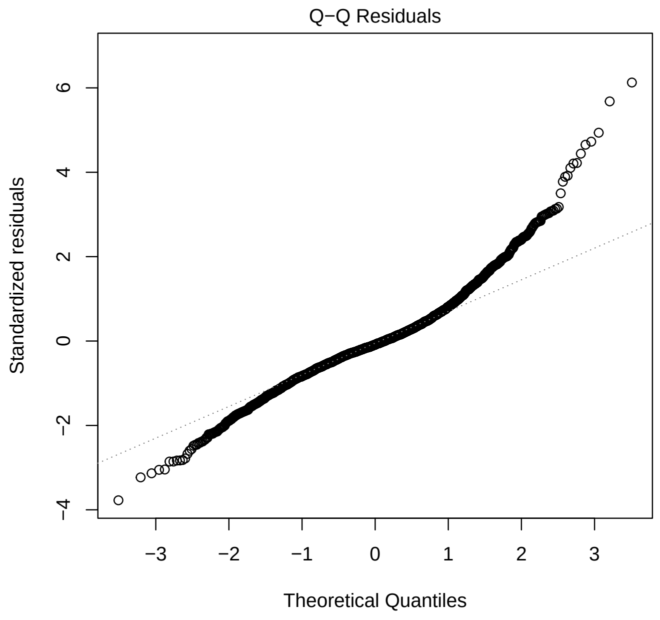

Figure 17: Normal Q-Q plot for OLS fit with blocks.

- no Monte Carlo calculation is necessary to obtain expected fitness, and

- there is a canonical statistic that is a monotone function of fitness so it is only necessary to have one quadratic function of phenotypes in the model.

In contrast, the analysis in Section 3 had both complications. We needed Monte Carlo approximation of the fitness landscape, and we needed two quadratic functions, one for the canonical parameter corresponding to `fruit` and the other for the canonical parameter corresponding to `seed`.

In conclusion, the analysis is simpler when the data are "aster-friendly" but it can be done even when not.

7 Diagnostic Plots for Paper

Here we just put Figure 15 and Figure 16 in one plot.

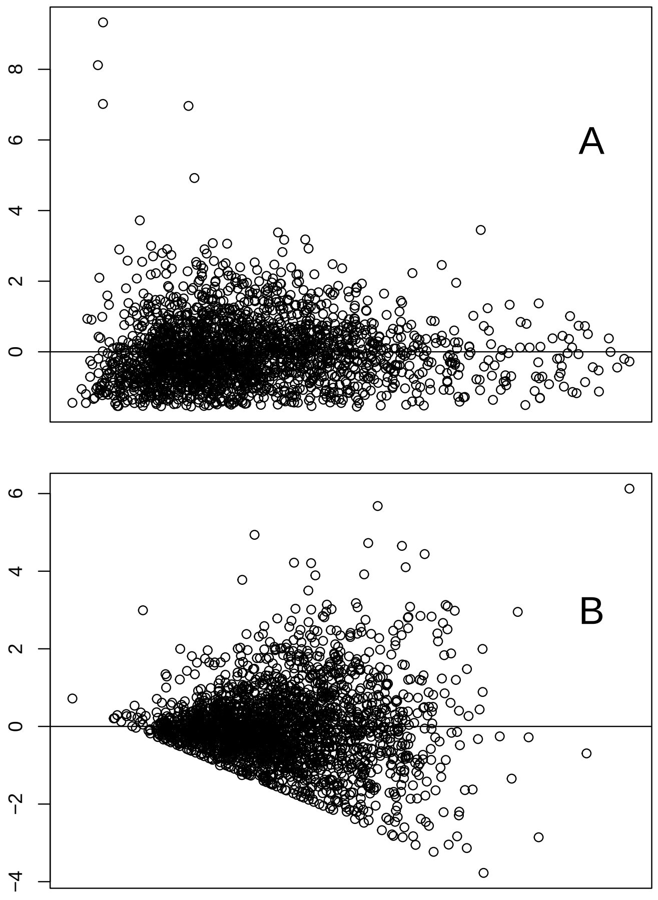

Figure 18: Diagnostic Plots. A: Pearson residuals for fruit count given nonzero fitness plotted against fitted values. B: standardized OLS residuals for fruit count plotted against fitted values.

8 Subsampling a Component of Fitness

Before doing anything, we remove all the variables generated in the preceding analyses.

```
> rm(list = ls())
> ls(all.names = TRUE)

[1] ".Random.seed"
```

8.1 Introduction

We investigate an aster model with graph $1 \to$ `reprod` $\to$ `fruit` $\to$ `samp` $\to$ `seed`, where

- `reprod` is Bernoulli,
- `fruit` is zero-truncated Poisson conditional on `reprod == 1`,
- `samp` is binomial with sample size `fruit` and known success probability p, and
- `fruit` is Poisson with mean `samp` $\times \mu$, where μ is an unknown parameter (mean value parameter).

Each of these specifies a one-parameter exponential family whether the parameter was specifically mentioned or not. Each of these is in aster model form in which the predecessor plays the role of sample size, whether it was described as sample size or not.

The somewhat odd thing about this proposal is that the parameter p is *known* and is a *conditional* mean value parameter, but we intend to use an *unconditional* aster model and treat the unconditional canonical parameter as *unknown*. Nevertheless, we try an example to see how it works. (With modification to the aster code, we could treat p as known, but the current code cannot handle this.)

Because this model is a bit odd, we start with the simpler model with graph $1 \to$ `reprod` $\to$ `fruit` $\to$ `seed` which has no sampling so seeds are counted for all fruits rather than just for a sample. This model is acknowledged to be the Right Thing (with a capital R and a capital T) but may not be feasible because counting seeds for all fruits may be too much work.

8.2 The Models

First we set the "simulation truth" parameter values. Since unconditional parameterizations are difficult to imagine, we set conditional mean value parameters.

```
> nind <- 1000
> preprod <- 0.75
> mfruit <- 100
> psamp <- 1 / 10
> mseed <- 10
```

Then we set up the aster model structures.

```
> fam <- c(1, 3, 1, 2)
> pred <- c(0, 1, 2, 3)
> vars <- c("reprod", "fruit", "samp", "seed")
> Fam <- fam[-3]
> Pred <- pred[-4]
> Vars <- vars[-3]
```

8.2.1 Simulate Data without Dependence on Covariates

```
> set.seed(42)
> Reprod <- sample(c(0, 1), nind, replace = TRUE,
+     prob = c(1 - preprod, preprod))
> Fruit <- rpois(nind, lambda = mfruit)
> Fruit <- Fruit * Reprod
> Seed <- rpois(nind, lambda = mseed * Fruit)
> zbase <- rnorm(nind)
> z1 <- zbase + rnorm(nind)
> z2 <- zbase + rnorm(nind)
> Dat <- data.frame(reprod = Reprod, fruit = Fruit, seed = Seed,
+     z1, z2, root = rep(1, nind))
> names(Dat)

[1] "reprod" "fruit"  "seed"   "z1"     "z2"     "root"

> Redata <- reshape(Dat, varying = list(Vars), direction = "long",
+     timevar = "varb", times = as.factor(Vars), v.names = "resp")
> names(Redata)

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

There is one further step. We need to zero out the phenotype values except those associated with `seed` since that is the variable that *directly* contributes to fitness.

```
> wind <- grep("seed", as.character(Redata$varb))
> for (labz in grep("z", names(Redata), value = TRUE)) {
+     Redata[[labz]][- wind] <- 0
+ }
```

Now fit a model.

```
> library(aster)
> out1 <- aster(resp ~ varb, Pred, Fam, varb, id, root, data = Redata,
+     type = "conditional")
> summary(out1)

Call:
aster.formula(formula = resp ~ varb, pred = Pred, fam = Fam,
    varvar = varb, idvar = id, root = root, data = Redata, type = "conditional")

             Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.605687   0.003641 1265.01   <2e-16 ***
varbreprod  -3.485626   0.073516  -47.41   <2e-16 ***
varbseed    -2.301077   0.003818 -602.66   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Check conditional mean value parameters.

```
> Renewdata <- Redata[Redata$id == 1, ]
> Renewdata$resp <- 1
> pout1 <- predict(out1, varvar = varb, idvar = id, root = root,
+     newdata = Renewdata, model.type = "conditional")
> pout1

[1]   0.75400 100.05172  10.02027
```

We recover the "simulation truth" to high accuracy.

8.2.2 Simulate Data with Dependence on Covariates

First we fit the model we want to use to the data we have. The fitted parameters will make no sense, because the fitness landscape is flat for the data we have, but we can use the model structure.

```
> out2 <- aster(resp ~ varb + z1 + z2 + I(z1^2) + I(z2^2) + I(z1 * z2),
+     Pred, Fam, varb, id, root, data = Redata)
> summary(out2, info.tol = 1e-12)

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

              Estimate Std. Error  z value Pr(>|z|)
(Intercept) -4.415e+00  1.209e-02 -365.251   <2e-16 ***
varbreprod  -9.397e+01  3.754e-01 -250.323   <2e-16 ***
varbseed     6.719e+00  1.319e-02  509.493   <2e-16 ***
```

```
z1           3.639e-05  5.928e-05    0.614    0.539
z2           2.008e-05  6.230e-05    0.322    0.747
I(z1^2)      7.588e-06  3.396e-05    0.223    0.823
I(z2^2)      3.332e-05  3.664e-05    0.909    0.363
I(z1 * z2)  -4.517e-05  5.673e-05   -0.796    0.426
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

We now want to make up a quadratic function of z. We just take the one from the third paper (about aster vs. Lande-Arnold) currently being written.

```
> # z1 <- Dat$z1
> # z2 <- Dat$z2
> ascal <- 0.001
> quad <- ascal * ((z1 + z2) - (z1^2 + z2^2) + z1 * z2)
> con <- mean(quad)
> mean(quad - con)
```

```
[1] 1.277715e-19
```

Now we change the coefficients in `out2` to be the ones for this quadratic model. Then convert to canonical parameters and use the `raster` function to simulate new data.

```
> fake <- out2
> fake$coefficients[3] <- fake$coefficients[3] - con
> fake$coefficients[4:5] <- ascal
> fake$coefficients[6:7] <- (- ascal)
> fake$coefficients[8] <- ascal
> fake$coefficients <- round(fake$coefficients, 3)
> fake$coefficients
```

```
(Intercept)  varbreprod    varbseed          z1          z2
     -4.415     -93.971       6.722       0.001       0.001
    I(z1^2)     I(z2^2)  I(z1 * z2)
     -0.001      -0.001       0.001
```

```
> theta <- predict(fake, model.type = "conditional", parm.type = "canonical")
> theta <- matrix(theta, nrow = nrow(fake$x), ncol = ncol(fake$x))
> root <- matrix(1, nrow = nind, ncol = length(Vars))
> xnew <- raster(theta, Pred, Fam, root)
```

Now we need to reshape these new data just like we did the old.

```
> dimnames(xnew) <- list(NULL, Vars)
> dnew <- as.data.frame(xnew)
> renew <- reshape(dnew, varying = list(Vars), direction = "long",
+     timevar = "varb", times = as.factor(Vars), v.names = "resp1")
> Redata$resp1 <- renew$resp1
```

Now we fit the model we want to use to this new data simulated from this model.

```
> out3 <- aster(resp1 ~ varb + z1 + z2 + I(z1^2) + I(z2^2) + I(z1 * z2),
+     Pred, Fam, varb, id, root, data = Redata)
> sout3 <- summary(out3, info.tol = 1e-11)
> print(sout3)

Call:
aster.formula(formula = resp1 ~ varb + z1 + z2 + I(z1^2) + I(z2^2) +
    I(z1 * z2), pred = Pred, fam = Fam, varvar = varb, idvar = id,
    root = root, data = Redata)

               Estimate Std. Error  z value Pr(>|z|)
(Intercept) -4.406e+00  1.296e-02 -340.030   <2e-16 ***
varbreprod  -9.454e+01  4.318e-01 -218.969   <2e-16 ***
varbseed     6.713e+00  1.414e-02  474.788   <2e-16 ***
z1           8.742e-04  9.461e-05    9.240   <2e-16 ***
z2           1.104e-03  1.087e-04   10.162   <2e-16 ***
I(z1^2)     -8.313e-04  7.219e-05  -11.514   <2e-16 ***
I(z2^2)     -9.147e-04  7.947e-05  -11.511   <2e-16 ***
I(z1 * z2)   8.657e-04  1.039e-04    8.331   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Pretty close agreement.

8.2.3 Simulate Data with Sampling

We don't simulate using `raster` because we know our model is a bit odd and doesn't fit the data. Instead we just subsample directly. Without subsampling `seed` is `Poisson`($\texttt{fruit} \cdot \mu$) where μ is the mean number of seeds per fruit (μ varies from individual to individual, but that is irrelevant to subsampling, which works on one individual at a time). With subsampling `samp` is `binomial`(`fruit`, p) where p is the subsampling fraction (p does not vary among individuals), and `seed` is `Poisson`($\texttt{samp} \cdot \mu$). It can be shown that if we define $q = \texttt{samp}/\texttt{fruit}$, (and $q = 0$ if `samp` = `fruit` = 0, so q varies from individual to individual), then we can set `seed` to be `binomial`(`fruit`, q) and this will have the required Poisson distribution.

```
> reprod <- Redata$resp1[as.character(Redata$varb) == "reprod"]
> fruit <- Redata$resp1[as.character(Redata$varb) == "fruit"]
> samp <- rbinom(nind, size = fruit, prob = psamp)
> oldseed <- Redata$resp1[as.character(Redata$varb) == "seed"]
> pseed <- samp / fruit
> pseed[samp == 0] <- 0
> seed <- rbinom(nind, size = oldseed, prob = pseed)
> dat2 <- data.frame(reprod, fruit, samp, seed, z1, z2, root = rep(1, nind))
> redata <- reshape(dat2, varying = list(vars), direction = "long",
+     timevar = "varb", times = as.factor(vars), v.names = "resp")
> names(redata)

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

> wind <- grep("seed", as.character(redata$varb))
> for (labz in grep("z", names(redata), value = TRUE)) {
+     redata[[labz]][- wind] <- 0
+ }
```

Now fit this model.

```
> out4 <- aster(resp ~ varb + z1 + z2 + I(z1^2) + I(z2^2) + I(z1 * z2),
+     pred, fam, varb, id, root, data = redata)
> sout4 <- summary(out4, info.tol = 1e-11)
> print(sout4)

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

              Estimate Std. Error  z value Pr(>|z|)
(Intercept)  5.216e+00  4.107e-03 1269.956  < 2e-16 ***
varbreprod  -1.050e+02  4.142e-01 -253.395  < 2e-16 ***
varbsamp    -1.642e+01  4.220e-02 -388.989  < 2e-16 ***
varbseed    -2.909e+00  5.730e-03 -507.784  < 2e-16 ***
z1           6.121e-03  7.850e-04    7.797 6.32e-15 ***
z2           8.656e-03  8.911e-04    9.714  < 2e-16 ***
I(z1^2)     -5.794e-03  5.486e-04  -10.561  < 2e-16 ***
I(z2^2)     -6.593e-03  6.150e-04  -10.720  < 2e-16 ***
I(z1 * z2)   5.633e-03  8.446e-04    6.669 2.57e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> names(sout4)

 [1] "coefficients" "iter"         "converged"    "deviance"
 [5] "gradient"     "hessian"      "newton"       "rank"
 [9] "x"            "root"         "pred"         "fam"
[13] "modmat"       "type"         "famlist"      "fisher"
[17] "origin"       "call"         "formula"      "terms"
[21] "data"         "xlevels"
```

Compare estimates with and without sampling.

```
> foo <- sout3$coefficients[ , "Estimate"]
> foo <- foo[grep("z", names(foo))]
> bar <- sout4$coefficients[ , "Estimate"]
> bar <- bar[grep("z", names(bar))]
> baz <- cbind(foo, bar)
> dimnames(baz)[[2]] <- c("without samp.", "with samp.")
> baz <- round(baz, 6)
> print(baz)
```

```
           without samp. with samp.
z1              0.000874   0.006121
z2              0.001104   0.008656
I(z1^2)        -0.000831  -0.005794
I(z2^2)        -0.000915  -0.006593
I(z1 * z2)      0.000866   0.005633
```

And compare standard errors with and without sampling.

```
> foo <- sout3$coefficients[ , "Std. Error"]
> foo <- foo[grep("z", names(foo))]
> bar <- sout4$coefficients[ , "Std. Error"]
> bar <- bar[grep("z", names(bar))]
> baz <- cbind(foo, bar)
> dimnames(baz)[[2]] <- c("without samp.", "with samp.")
> baz <- round(baz, 7)
> print(baz)
```

```
           without samp. with samp.
z1             0.0000946  0.0007850
z2             0.0001087  0.0008911
I(z1^2)        0.0000722  0.0005486
I(z2^2)        0.0000795  0.0006150
I(z1 * z2)     0.0001039  0.0008446
```

Clearly, standard errors are several times larger with sampling. The estimates also seem larger in absolute value but seem to have increased proportionally. So there may be some bias due to subsampling. This needs more investigation, but that will have to wait until we have a real experiment with this subsampling design.

Actually, this "bias" may be an illusion. The models being compared are different, and there is no reason their canonical parameters should be comparable (canonical parameters are meaningless). Let us do the same comparison with mean value parameters, or, better yet, with expected fitness, which is a certain particular mean value parameter (expected seed count).

```
> pout3 <- predict(out3, se.fit = TRUE, info.tol = 1e-9)
> fit3 <- pout3$fit[as.character(out3$data$varb) == "seed"]
```

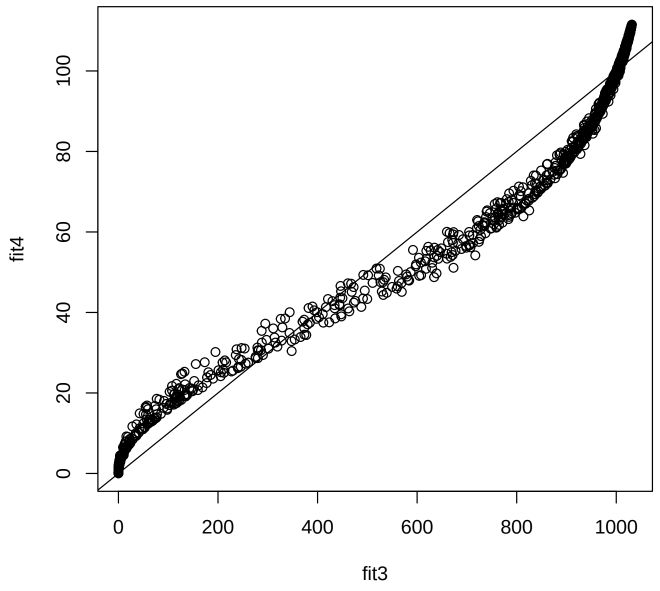

Figure 19: Scatterplot of expected seed count with and without subsampling. Line has intercept zero and slope the sampling fraction.

```
> se3 <- pout3$se.fit[as.character(out3$data$varb) == "seed"]
> pout4 <- predict(out4, se.fit = TRUE)
> fit4 <- pout4$fit[as.character(out4$data$varb) == "seed"]
> se4 <- pout4$se.fit[as.character(out4$data$varb) == "seed"]
```

Figure 19 (page 52) shows the scatter plot of expected seed count (for all individuals) without subsampling (horizontal axis) and with (vertical axis). The line is what should happen if the only effect of subsampling was to reduce the expected value proportional to the sampling fraction. It is made by the following code.

```
> plot(fit3, fit4)
> abline(0, psamp)
```

We can see from Figure 19 that the subsampling does have some effect, and does produce some bias, although nowhere near as large as it appears to be from our (incorrect) comparison of canonical parameter values. It is clear that, on average, there is no bias, but that some parts of the fitness surface are distorted somewhat by the subsampling.

References

Davison, A. C., and Snell, E. J. (1991). Residuals and diagnostics. In *Statistical Theory and Modelling: In honour of Sir David Cox, FRS.* D. V. Hinkley, N. Reid, E. J. Snell (eds.) Chapman & Hall.

Etterson, J. R. (2004) Evolutionary potential of *Chamaecrista fasciculata* in relation to climate change. I. Clinal patterns of selection along an environmental gradient in the great plains. *Evolution*, **58**, 1446–1458.

Etterson, J. R., and Shaw, R. G. (2001). Constraint to adaptive evolution in response to global warming. *Science*, **294**, 151–154.

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

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

Lindgren, B. W. (1993). *Statistical Theory*, 4th ed. New York: Chapman & Hall.

R Development Core Team (2006). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. `http://www.R-project.org`.