Aster Models with Random Effects and Additive Genetic Variance for Fitness

By
Charles J. Geyer and Ruth G. Shaw

Technical Report No. 696

School of Statistics
University of Minnesota

July 10, 2013

Abstract

This technical report is a minor supplement to the paper Geyer et al. (in press) and its accompanying technical report Geyer et al. (2012). It shows how to move variance components from the canonical parameter scale to the mean value parameter scale. This is useful in estimating additive genetic variance for fitness, and that appears in Fisher's fundamental theorem of natural selection, which predicts the rate of increase in fitness via natural selection.

1 R

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

```
> library(aster)
```

The version of the package used to make this document is 1.1-3. The version of R used to make this document is 4.3.2.

2 Data and Aster Model Fits

We use data on the partridge pea (*Chamaecrista fasciculata*) described in Section 8 of Geyer et al. (2012) and contained in the dataset `chamae3` in the R contributed package `aster`. For each individual, two response variables are observed, connected by the following graphical model

$$1 \xrightarrow{\text{Ber}} y_1 \xrightarrow{\text{0-Poi}} y_2$$

y_1 being an indicator of whether any fruits were produced, y_2 being the count of the number of fruits produced, the unconditional distribution of y_1 being Bernoulli, and the conditional distribution of y_2 given y_1 being zero-truncated Poisson.

We load the data

```
> data(chamae3)
> names(chamae3)
```

```
 [1] "SIRE" "DAM"  "POP"  "SITE" "ROW"  "BLK"  "varb" "resp" "id"   "root"
[11] "fit"
```

```
> levels(chamae3$varb)
```

```
[1] "fecund" "fruit"
```

Then set up the graphical model

```
> pred <- c(0, 1)
> fam <- c(1, 3)
> sapply(fam.default(), as.character)[fam]
```

```
[1] "bernoulli"                          "truncated.poisson(truncation = 0)"
```

First we subset the data, looking at each site-population pair separately. Make a list whose components are nine data frames (the data for the separate analyses).

```
> names(chamae3)

 [1] "SIRE" "DAM"  "POP"  "SITE" "ROW"  "BLK"  "varb" "resp" "id"   "root"
[11] "fit"

> site <- as.character(chamae3$SITE)
> pop <- as.character(chamae3$POP)
> usite <- sort(unique(site))
> upop <- sort(unique(pop))
> usite

[1] "K" "M" "O"

> upop

[1] "1" "2" "3"

> rsite <- rep(usite, times = length(upop))
> rpop <- rep(upop, each = length(usite))
> cbind(rsite, rpop)

      rsite rpop
 [1,] "K"   "1"
 [2,] "M"   "1"
 [3,] "O"   "1"
 [4,] "K"   "2"
 [5,] "M"   "2"
 [6,] "O"   "2"
 [7,] "K"   "3"
 [8,] "M"   "3"
 [9,] "O"   "3"

> nsitepop <- paste(rsite, rpop, sep = "")
> nsitepop

[1] "K1" "M1" "O1" "K2" "M2" "O2" "K3" "M3" "O3"

> subdata <- list()
> for (i in seq(along = rsite))
+     subdata[[nsitepop[i]]] <- droplevels(subset(chamae3,
+         site == rsite[i] & pop == rpop[i]))
> length(subdata)

[1] 9

> sapply(subdata, nrow)

  K1   M1   O1   K2   M2   O2   K3   M3   O3
2108 2054 2034 2342 2256 2292 2020 1894 2062
```

```
> sapply(subdata, function(x) unique(x$SITE))
```

```
K1 M1 O1 K2 M2 O2 K3 M3 O3
 K  M  O  K  M  O  K  M  O
Levels: K M O
```

```
> sapply(subdata, function(x) unique(x$POP))
```

```
K1 M1 O1 K2 M2 O2 K3 M3 O3
 1  1  1  2  2  2  3  3  3
Levels: 1 2 3
```

We see we have successfully done the subsetting.

Following Section 8.6 in Geyer et al. (2012) we look at only two subsets (merely to illustrate the method): the Kansas population in the Kansas site and in the Oklahoma site. These are the `"K2"` and `"O2"` elements of the `sublist` made above.

```
> subsubdata <- subdata[c("K2", "O2")]
> names(subsubdata)
```

```
[1] "K2" "O2"
```

```
> sapply(subsubdata, class)
```

```
          K2           O2
"data.frame" "data.frame"
```

Then we do the analysis. Since this analysis takes quite a bit of time, we save the results and load them from a file if they are already done.

```
> suppressWarnings(foo <- try(load("subsubout.rda"), silent = TRUE))
> done <- (! inherits(foo, "try-error"))
> done
```

```
[1] TRUE
```

```
> if (! done) {
+     subsubout <- lapply(subsubdata, function(x) reaster(resp ~ varb + fit:BLK,
+         list(sire = ~ 0 + fit:SIRE, dam = ~ 0 + fit:DAM),
+         pred, fam, varb, id, root, data = x))
+     save(subsubout, file = "subsubout.rda")
+ }
> names(subsubout)
```

```
[1] "K2" "O2"
```

```
> sapply(subsubout, class)
```

```
     K2                O2
[1,] "reaster.formula" "reaster.formula"
[2,] "reaster"         "reaster"
[3,] "asterOrReaster"  "asterOrReaster"
```

The summaries for these analyses are shown in Appendix B of Geyer et al. (2012) and so need not be shown here.

3 Mapping Variance Components

3.1 Theory

So now we need to figure out how to map canonical parameters to mean value parameters. The only tool for this in the `aster` package being the function `predict.aster`. Start with the formula, equation (3) in Geyer et al. (2012),

$$\varphi = a + M\alpha + Zb$$

where φ is the saturated model canonical parameter vector, where a is a known vector, M and Z are known matrices, b is a normal random vector with mean vector zero and variance matrix D. The vector a is called the offset vector and the matrices M and Z are called the model matrices for fixed and random effects, respectively. The transformation from the canonical to mean value parameter vector, equation (1) in Geyer et al. (2012), is

$$\mu(\varphi) = c'(\varphi), \tag{1a}$$

where c is the cumulant function of the saturated aster model exponential family. And this transformation has derivative

$$W(\varphi) = \mu'(\varphi) = c''(\varphi), \tag{1b}$$

equation (2) in Geyer et al. (2012). The R function `predict.aster` calculates the transformation (1a) and, if asked for, the derivative (1b). More precisely, if given an origin a, a new model matrix $M_{\texttt{new}}$, another matrix A, and a regression coefficient vector α, it will calculate

$$A^T \mu(a + M_{\texttt{new}}\alpha) \tag{2a}$$

and its derivative with respect to α

$$A^T W(a + M_{\texttt{new}}\alpha) M_{\texttt{new}} \tag{2b}$$

(Geyer, et al., 2007, Equations (19) and (20)). None of this description of what `predict.aster` does makes any mention of random effects, and as far as `predict.aster` knows, there are no random effects. It was designed to do fixed-effect aster models. If we are going to get it to say anything useful about variance components, we are going to have to trick it. We are going to have to find an A and $M_{\texttt{new}}$ so (2a) and (2b) tell us what we want to know.

One last comment about the function `predict.aster`: when the optional argument `se.fit = TRUE` is given, this function returns a list, the `fit` component of which is (2a) and the `gradient` component of which is (2b). The latter is undocumented. The `gradient` component was initially designed for testing and debugging, but sometimes is useful in scientific inference, as in the current situation.

The way the delta method works is to treat a nonlinear function as a linear one using the Taylor series up through first derivatives. So if we linearize $A^T\mu(a + M\alpha + Zb)$, thought of as a function of b, and expanding around $b = 0$, we get

$$A^T\mu(a + M\alpha + Zb) \approx A^T\mu(a + M\alpha) + A^T W(a + M\alpha)Zb$$

and the variance of this is what we want (variance of b transferred to the mean value parameter scale), that is,

$$A^T W(a + M\alpha)ZDZ^T W(a + M\alpha)A. \tag{3}$$

The first thing we observe is that on the canonical parameter scale the variance matrix D of the random effect vector b is diagonal (this is a limitation of the R function `reaster` and the paper Geyer et al. (in press) it is based on), but (3) is a general variance matrix (not necessarily diagonal and not even usually diagonal).

When computing "additive genetic variance for fitness" (which is a scalar quantity) the latter issue does not arise because A is a column vector so (3) is a scalar (or a one-by-one matrix).

More precisely, (3) is a scalar when we compute variance for fitness for one individual, which we make a (made-up) typical individual.

3.2 Practice

3.2.1 Try 1

In aid of this we first fit an entirely fixed effects model, ignoring dam effects, which is the same as setting them to zero (evaluating for a "typical dam effect").

```
> mydata <- subsubdata[[1]]
> aout <- aster(resp ~ varb + fit : (BLK + SIRE),
+     pred, fam, varb, id, root, data = mydata)
> summary(aout)

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

               Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.033e+02  1.589e+00 -65.020  < 2e-16 ***
varbfruit     1.090e+02  1.589e+00  68.619  < 2e-16 ***
fit:BLK1     -5.562e-01  5.821e-03 -95.552  < 2e-16 ***
fit:BLK2     -2.758e-01  5.313e-03 -51.915  < 2e-16 ***
fit:BLK3     -7.060e-02  5.049e-03 -13.983  < 2e-16 ***
fit:SIRE2003 -3.883e-02  1.873e-02  -2.073 0.038134 *
fit:SIRE2010 -2.123e-01  1.960e-02 -10.832  < 2e-16 ***
fit:SIRE2012 -1.947e-02  1.864e-02  -1.044 0.296310
fit:SIRE2016 -2.000e-01  1.997e-02 -10.013  < 2e-16 ***
fit:SIRE2020 -2.134e-01  2.007e-02 -10.633  < 2e-16 ***
fit:SIRE2024 -3.312e-01  2.026e-02 -16.346  < 2e-16 ***
fit:SIRE2031 -3.663e-01  2.047e-02 -17.894  < 2e-16 ***
fit:SIRE2038  4.300e-02  1.836e-02   2.342 0.019159 *
fit:SIRE2045 -1.251e-01  1.942e-02  -6.443 1.17e-10 ***
fit:SIRE2049  1.111e-01  1.806e-02   6.150 7.76e-10 ***
fit:SIRE2051  6.709e-02  1.848e-02   3.630 0.000284 ***
fit:SIRE2056 -5.204e-01  1.949e-02 -26.693  < 2e-16 ***
```

```
fit:SIRE2072 -3.090e-01  2.014e-02 -15.349  < 2e-16 ***
fit:SIRE2074 -1.197e-01  1.912e-02  -6.261 3.83e-10 ***
fit:SIRE2079 -2.868e-01  2.018e-02 -14.211  < 2e-16 ***
fit:SIRE2082  2.630e-01  1.746e-02  15.064  < 2e-16 ***
fit:SIRE2084 -1.108e-01  1.908e-02  -5.807 6.37e-09 ***
fit:SIRE2085 -5.036e-01  1.948e-02 -25.853  < 2e-16 ***
fit:SIRE2087 -1.972e-02  1.912e-02  -1.031 0.302532
fit:SIRE2089 -2.279e-01  1.968e-02 -11.580  < 2e-16 ***
fit:SIRE2093 -1.817e-01  2.003e-02  -9.071  < 2e-16 ***
fit:SIRE2094  2.051e-02  1.846e-02   1.111 0.266547
fit:SIRE2095 -2.134e-01  1.987e-02 -10.739  < 2e-16 ***
fit:SIRE2098  7.369e-02  1.822e-02   4.044 5.25e-05 ***
fit:SIRE2102 -2.410e-01  1.976e-02 -12.200  < 2e-16 ***
fit:SIRE2108 -1.378e-01  1.921e-02  -7.174 7.27e-13 ***
fit:SIRE2116 -4.661e-02  1.877e-02  -2.484 0.012990 *
fit:SIRE2117 -2.692e-02  1.867e-02  -1.442 0.149346
fit:SIRE2124 -9.518e-02  1.942e-02  -4.900 9.57e-07 ***
fit:SIRE2133  2.214e-01  1.783e-02  12.419  < 2e-16 ***
fit:SIRE2134 -5.822e-02  1.882e-02  -3.094 0.001978 **
fit:SIRE2141 -9.246e-02  1.952e-02  -4.738 2.16e-06 ***
fit:SIRE2150 -3.158e-01  2.041e-02 -15.477  < 2e-16 ***
fit:SIRE2151  1.484e-01  1.811e-02   8.193 2.55e-16 ***
fit:SIRE2166  3.272e-02  1.840e-02   1.778 0.075410 .
fit:SIRE2172 -1.113e-01  1.908e-02  -5.836 5.35e-09 ***
fit:SIRE2173  5.040e-02  1.874e-02   2.690 0.007148 **
fit:SIRE2174  1.267e-02  1.849e-02   0.685 0.493174
fit:SIRE2178  6.268e-02  1.844e-02   3.398 0.000678 ***
fit:SIRE2184 -1.036e-01  1.929e-02  -5.371 7.85e-08 ***
fit:SIRE2191  1.321e-01  1.798e-02   7.349 2.00e-13 ***
fit:SIRE2192 -2.602e-01  1.986e-02 -13.100  < 2e-16 ***
fit:SIRE2195  1.448e-01  1.792e-02   8.078 6.58e-16 ***
fit:SIRE2196  1.673e-01  1.795e-02   9.318  < 2e-16 ***
fit:SIRE2200  2.894e-02  1.842e-02   1.571 0.116152
fit:SIRE2204 -4.064e-01  2.071e-02 -19.623  < 2e-16 ***
fit:SIRE2214 -8.665e-02  1.896e-02  -4.570 4.87e-06 ***
fit:SIRE2215 -2.780e-02  1.868e-02  -1.488 0.136709
fit:SIRE2224 -4.431e-02  1.876e-02  -2.363 0.018149 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Original predictor variables dropped (aliased)
     fit:BLK4
```

Now we want to use as "newdata" the data for just one individual

```
> id <- mydata$id
> inies <- id == min(id)
> mynewdata <- mydata[inies, ]
> dim(mynewdata)
```

```
[1]  2 11
```

Now we do the prediction, which we want to do at the parameter values for the random effects fit.

```
> rout <- subsubout[[1]]
> alpha.hat <- rout$alpha
> b.hat <- rout$b
> fred <- c(alpha.hat, b.hat)
> idx <- match(names(aout$coefficients), names(fred))
> idx
```

```
 [1]  1  2  3  4  5  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
[26] 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
[51] 52 53 54 55
```

```
> head(fred[- idx])
```

```
fit:SIRE2001  fit:DAM2002  fit:DAM2004  fit:DAM2007  fit:DAM2008  fit:DAM2009
  0.03439038   0.07186533  -0.08213122   0.10718912  -0.07539800   0.18907438
```

We see the omitted regression cofficients in our fixed effects fit `aout` are not important. We do not care that sire 2001 was dropped, because we are only going to predict for one "generic" sire and we do not care which. Similarly we deliberately dropped all the dams.

```
> pout <- predict(aout, varvar = varb, idvar = id, root = root,
+     newdata = mynewdata, se.fit = TRUE, newcoef = fred[idx])
> foo <- pout$gradient
> rownames(foo) <- levels(chamae3$varb)
> colnames(foo) <- names(aout$coefficients)
> t(head(t(foo), n = 11))
```

```
        (Intercept)    varbfruit     fit:BLK1 fit:BLK2 fit:BLK3 fit:SIRE2003
fecund 8.983133e-21 8.919344e-21 8.919344e-21        0        0            0
fruit  1.398260e+02 1.398260e+02 1.398260e+02        0        0            0
       fit:SIRE2010 fit:SIRE2012 fit:SIRE2016 fit:SIRE2020 fit:SIRE2024
fecund            0            0            0            0 8.919344e-21
fruit             0            0            0            0 1.398260e+02
```

```
> thegradient <- foo["fruit", "fit:SIRE2024"]
> thegradient
```

```
[1] 139.826
```

We see that there are only two different nonzero numbers in the gradient, one in the first row corresponding to the first component in the graph and one in the second row corresponding to the second component in the graph, which is our measure of fitness. Thus we want the latter.

Finally we can apply the delta method. The additive genetic variance for fitness (or its best surrogate in these data, the sire variance transferred to the mean value parameter scale) is

```
> thevariance1 <- thegradient^2 * rout$nu["sire"]
> thevariance1

    sire
285.5482
```

3.2.2 Try 2

In aid of repeating the preceding analysis, we make a function to do it.

```
> doit <- function(mydata, rout)
+ {
+     aout <- aster(resp ~ varb + fit : (BLK + SIRE),
+         pred, fam, varb, id, root, data = mydata)
+     id <- mydata$id
+     inies <- id == min(id)
+     mynewdata <- mydata[inies, ]
+
+     alpha.hat <- rout$alpha
+     b.hat <- rout$b
+     fred <- c(alpha.hat, b.hat)
+     idx <- match(names(aout$coefficients), names(fred))
+
+     pout <- predict(aout, varvar = varb, idvar = id, root = root,
+         newdata = mynewdata, se.fit = TRUE, newcoef = fred[idx])
+     foo <- pout$gradient
+     rownames(foo) <- levels(chamae3$varb)
+     bar <- foo["fruit", ]
+     bar <- bar[bar != 0]
+     baz <- unique(bar)
+     stopifnot(all.equal(max(baz), min(baz)))
+     baz[1]
+ }
```

and then we try it out, seeing if it repeats the analysis of the preceding section.

```
> thegradient.redo <- doit(subsubdata[[1]], subsubout[[1]])
> identical(thegradient, thegradient.redo)

[1] TRUE
```

3.2.3 Try 3

And we apply this function to do the analysis for the other data set.

```
> thegradient.too <- doit(subsubdata[[2]], subsubout[[2]])
> thegradient

[1] 139.826

> thegradient.too
```

```
[1] 52.79209
```

These are the gradients of the mappings from the canonical parameter scale to the mean value parameter scale.

```
> thevariance2 <- thegradient.too^2 * subsubout[[1]]$nu["sire"]
> thevariance1

    sire
285.5482

> thevariance2

    sire
40.70436
```

These are the sire variance component for two different population-site combinations, both mapped to the mean value parameter scale.

4 Mean Fitness

To apply the fundamental theorem of natural selection we also need mean fitness.

```
> meanfit1 <- with(subsubdata[[1]], mean(resp[as.character(varb) == "fruit"]))
> meanfit2 <- with(subsubdata[[2]], mean(resp[as.character(varb) == "fruit"]))
> meanfit1

[1] 227.7575

> meanfit2

[1] 57.774
```

5 Fundamental Theorem of Natural Selection

We can now apply Fisher's fundamental theorem of natural selection to predict the rate of increase in fitness as the ratio of the additive genetic variance for fitness to the mean fitness. This evolutionary principle has been highly influential conceptually but, as noted by Shaw and Shaw (2013), has not been implemented empirically. For the mating design used in this experiment, dams nested within sires (NC I), quantitative genetic theory shows that the component of variance due to sires estimates 1/4 of the additive genetic variance (Falconer and Mackay, 1996, Chapter 9).

```
> 4 * thevariance1 / meanfit1

    sire
5.014952

> 4 * thevariance2 / meanfit2
```

```
    sire
2.818179
```

Thus, we predict that this Kansas population would increase in absolute fitness by about 5 fruits per plant, over a generation of selection in the Kansas site. In the Oklahoma site, this population is predicted to increase in fitness somewhat less, about 3 fruits per plant over one generation. These predictions are made on the assumption that the environment within each site has the same effect on fitness each generation. Nevertheless, these estimates are important as quantitative predictors of the rate of change in fitness to be expected through genetic change due to natural selection under current environmental conditions.

References

Falconer, D. S., and Mackay, T. F. C. (1996). *Introduction to Quantitative Genetics*, 4th ed. Pearson Education Ltd., Harlow, U. K.

Geyer, C. J., Ridley, C. E., Latta, R. G., Etterson, J. R., and Shaw, R. G. (2012). Aster Models with Random Effects via Penalized Likelihood. Technical Report 692, University of Minnesota School of Statistics. `http://purl.umn.edu/135870`.

Geyer, C. J., Ridley, C. E., Latta, R. G., Etterson, J. R., and Shaw, R. G. (in press). Local Adaptation and Genetic Effects on Fitness: Calculations for Exponential Family Models with Random Effects. To appear in *Annals of Applied Statistics*.

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

Shaw, R. G., and Shaw, F. H. (2013). Quantitative genetic study of the adaptive process. *Heredity*, doi:10.1038/hdy.2013.42.