Fisher Example for Le Cam Made Simple

Charles J. Geyer

May 20, 2005

1 Introduction

1.1 Model

We do, rather inefficiently, an example from quantitative genetics, using a model originally proposed by Fisher (1918) that directly began modern quantitative genetics and indirectly led to much of modern regression and analysis of variance. The data y are multivariate normal, decomposed as

$$y = \mu + b + e$$

where μ is an unknown parameter vector (the mean of y) and b and e are independent multivariate normal

$$b \sim \mathcal{N}(0, \sigma^2 A)$$
$$e \sim \mathcal{N}(0, \tau^2 I)$$

and σ^2 and τ^2 are unknown parameters called the *additive genetic* and *environmental* variance, respectively, A is a known matrix called the *numerator relationship matrix* in the animal breeding literature (Henderson, 1976; Quaas, 1976), and I is the identity matrix. We will say more about A in Section 1.3.

1.2 Pedigree

First we simulate a pedigree.

```
> set.seed(42)
> ngen <- 20
> popsize <- 200
> obsgen <- seq(11, 20)
```

```
> trips <- matrix(0, 0, 5)
> dimnames(trips) <- list(NULL, c("ind", "pa", "ma", "sex", "gen"))
> for (i in 1:ngen) {
+     if (i == 1) {
+         pas <- numeric(0)
+         mas <- numeric(0)
+     }
+     else {
+         pas <- trips[, "sex"] == 1 & trips[, "gen"] == (i - 1)
+         mas <- trips[, "sex"] == 2 & trips[, "gen"] == (i - 1)
+         pas <- seq(along = pas)[pas]
+         mas <- seq(along = mas)[mas]
+         if (length(pas) == 0 & length(mas) == 0)
+             stop("one or the other sex missing")
+     }
+     for (j in 1:popsize) {
+         sex <- sample(2:1, 1)
+         if (i == 1) {
+             pa <- 0
+             ma <- 0
+         }
+         else {
+             if (length(pas) == 1)
+                 pa <- pas
+             else pa <- sample(pas, 1)
+             if (length(mas) == 1)
+                 ma <- mas
+             else ma <- sample(mas, 1)
+         }
+         ind <- nrow(trips) + 1
+         trips <- rbind(trips, c(ind, pa, ma, sex, i))
+     }
+ }
> trips <- as.data.frame(trips)
> write.table(trips, file = "pedigree.txt", row.names = FALSE)
> rm(i, ind, j, ma, pa, mas, pas, sex)
```

1.3 Numerator Relationship Matrix

Then we calculate the numerator relationship matrix. From Henderson (1976, Section 3.1) we find that the elements a_{ij} can be calculated recursively if, as here, parents come before children in the pedigree.

(i) If the i-th individual is a founder (parents labeled zero), then

$$a_{ii} = 1$$
$$a_{ij} = a_{ji} = 0, \qquad j < i$$

(ii) If the i-th individual is not a founder (parents nonzero, say p and m, but less than i), then

$$a_{ii} = 1 + \tfrac{1}{2} a_{pm}$$
$$a_{ij} = a_{ji} = \tfrac{1}{2}(a_{jp} + a_{jm}), \qquad j < i$$

```
> imat <- diag(nrow(trips))
> rmat <- imat
> founder <- trips$pa == 0
> for (ind in 1:nrow(trips)) if (!founder[ind]) {
+     pa <- trips[ind, "pa"]
+     ma <- trips[ind, "ma"]
+     for (j in 1:(ind - 1)) {
+         rmat[ind, j] <- rmat[j, ind] <- (rmat[j, pa] + rmat[j,
+             ma])/2
+     }
+     rmat[ind, ind] <- 1 + rmat[ma, pa]/2
+ }
> rm(founder, ind, j, ma, pa)
```

1.4 Data

And then we simulate some data. Let us suppose we only have data on the bottom half of the pedigree.

```
> inies <- is.element(trips$gen, obsgen)
> imat <- imat[inies, inies]
> rmat <- rmat[inies, inies]
> range(rmat[lower.tri(rmat)])

[1] 0.02201080 0.80165141
```

Note that everybody on which we have data is related to everybody else (relatedness greater than zero).

```
> sig <- tau <- 1
> mu <- 5
> library(MASS)
> zerovec <- rep(0, sum(inies))
> g <- mvrnorm(mu = zerovec, Sigma = sig^2 * rmat)
> e <- mvrnorm(mu = zerovec, Sigma = tau^2 * imat)
> y <- mu + g + e
> yout <- rep(NA, length(inies))
> yout[inies] <- y
> write.table(data.frame(y = yout), file = "data.txt", row.names = FALSE)
> rm(e, g, inies, yout, zerovec)
```

2 Variogram Estimation

The original plan for this note was that it would make no contribution to the quantitative genetics literature, just use existing methodology, but our need for a "good" starting point for Newton's method prompts the following new idea, which is a "technology transfer" from spatial statistics (not the first such, MCMC for linkage being another). The estimator proposed here is much better than, for example, the estimator of equations (2) and (3) of Thomas et al. (2000). One may consider our proposal the obvious fix of theirs from a statistician knowledgeable about kriging.

In spatial statistics, the term *variogram* refers to the variance of the difference of correlated quantities as a function of position. The *semivariogram* is just half the variogram. Here the semivariogram is the function from (i, j) pairs to

$$\tfrac{1}{2} E\{(y_i - y_j)^2\} = \tfrac{1}{2}(a_{ii} + a_{jj} - 2a_{ij})\sigma^2 + \tau^2 \tag{1a}$$

This suggests a regression

$$\text{regress} \quad \tfrac{1}{2}(y_i - y_j)^2 \quad \text{on} \quad \tfrac{1}{2}(a_{ii} + a_{jj} - 2a_{ij}) \tag{1b}$$

call the "response" here the *empirical semivariogram* and the "predictor" here the *theoretical additive genetic semivariogram*. The "slope" in the regression estimates σ^2 and the "intercept" estimates τ^2.

But the "response" is not suitable material for ordinary least squares (OLS) regression. For one thing, its components are not independent, but, following the usual variogram estimation idea, we ignore that. But worse,

its components are positive and highly skewed. They more resemble scaled chi^2(1) random variables, than normal random variables. Thus we use a gamma generalized linear model (GLM) with identity link.

```
> foo <- outer(y, y, "-")^2/2
> foo <- foo[lower.tri(foo)]
> bar <- outer(diag(rmat), diag(rmat), "+")/2 - rmat
> bar <- bar[lower.tri(bar)]
> gout <- glm(foo ~ bar, family = Gamma("identity"), mustart = rep(1,
+     length(foo)))
> summary(gout)

Call:
glm(formula = foo ~ bar, family = Gamma("identity"), mustart = rep(1,
    length(foo)))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-7.7396  -1.6578  -0.6855   0.3028   6.0265

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.46191    0.04759   30.72   <2e-16 ***
bar          0.61748    0.04932   12.52   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Gamma family taken to be 1.939356)

    Null deviance: 5033847  on 1998999  degrees of freedom
Residual deviance: 5033618  on 1998998  degrees of freedom
AIC: 6118894

Number of Fisher Scoring iterations: 4

> foof <- round(foo/max(foo) * 499)
> barf <- round(bar/max(bar) * 499)
> quxf <- 1000 * foof + barf
> i <- match(sort(unique(quxf)), quxf)
```

Figure 1 (page 6) shows the scatter plot of “empirical semivariogram” versus

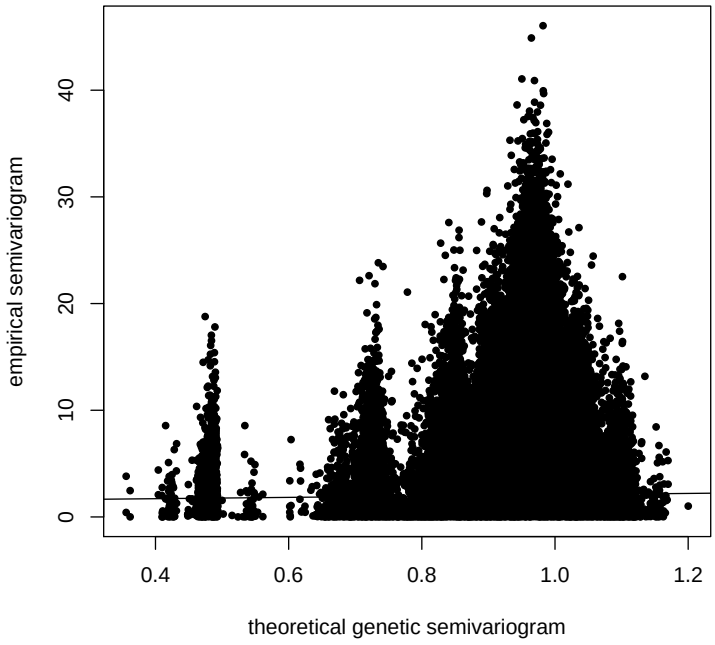

Figure 1: Semivariogram Estimation. Line is the gamma GLM with identity link regression line.

"theoretical additive genetic semivariogram" along with its fitted (gamma GLM with identity link) regression line.

3 Maximum Likelihood

3.1 Log Likelihood and Score

Now the log likelihood for these data is

$$l = -\tfrac{1}{2}(y-\mu)V^{-1}(y-\mu) - \tfrac{1}{2}\log\det V \tag{2}$$

where

$$V = \sigma^2 A + \tau^2 I$$

and the score is

$$\frac{\partial l}{\partial \mu} = u' V^{-1}(y-\mu) \tag{3a}$$

$$\frac{\partial l}{\partial \theta} = +\tfrac{1}{2}(y-\mu)V^{-1}\frac{\partial V}{\partial \theta}V^{-1}(y-\mu) - \tfrac{1}{2}\operatorname{tr}\left(\frac{\partial V}{\partial \theta}V^{-1}\right) \tag{3b}$$

where u is the vector with all components 1, where θ is either σ^2 or τ^2, the partial derivatives in (3b) being either A or I, respectively, and where tr denotes the trace of a matrix. Direct naive use of (2), (3a), and (3b) to find MLE is highly inefficient. This problem has been much studied (see Johnson and Thompson, 1995, and references therein) and efficient free implementations are available (e. g., Shaw and Shaw, 1994). We should also note that generally REML (Patterson and Thompson, 1971) rather than plain maximum likelihood is generally used to estimate these variance components. Nevertheless, we make just such naive use in this example (no high-quality implementation being available for R).

```
> logl <- function(theta, y) {
+     if (length(theta) != 3)
+         stop("length(theta) != 3")
+     mu <- theta[1]
+     sigsq <- theta[2]
+     tausq <- theta[3]
+     vmat <- sigsq * rmat + tausq * imat
+     r <- y - mu
+     logl <- -(1/2) * sum(r * solve(vmat, r)) - (1/2) * determinant(vmat,
+         logarithm = TRUE)$modulus
```

```
+     return(logl)
+ }
> score <- function(theta, y) {
+     if (length(theta) != 3)
+         stop("length(theta) != 3")
+     mu <- theta[1]
+     sigsq <- theta[2]
+     tausq <- theta[3]
+     vmat <- sigsq * rmat + tausq * imat
+     r <- y - mu
+     result <- double(3)
+     foo <- solve(vmat, r)
+     result[1] <- sum(foo)
+     vinv <- solve(vmat)
+     result[2] <- (1/2) * t(foo) %*% rmat %*% foo - (1/2) * sum(vinv *
+         rmat)
+     result[3] <- (1/2) * t(foo) %*% imat %*% foo - (1/2) * sum(vinv *
+         imat)
+     return(result)
+ }
```

3.2 Maximum Likelihood

A try at maximum likelihood

```
> theta.start <- as.numeric(c(mean(y), coef(gout)))
> lower <- c(-Inf, 0, 0)
> out <- optim(theta.start, logl, score, lower = lower, method = "L-BFGS-B",
+     control = list(fnscale = -length(y)), y = y)
> print(out)

$par
[1] 5.136033 1.049012 1.012568

$value
[1] -1554.177

$counts
function gradient
      12       12
```

```
$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

```
> theta.hat <- out$par
> mu.hat <- theta.hat[1]
> sig.hat <- sqrt(theta.hat[2])
> tau.hat <- sqrt(theta.hat[3])
> rm(foo, bar, foof, barf, quxf, i, gout)
```

3.3 Observed Fisher Information

For further likelihood analysis we also need the Hessian

$$\frac{\partial^2 l}{\partial \mu^2} = -u' V^{-1} u \tag{4a}$$

$$\frac{\partial^2 l}{\partial \theta \partial \theta'} = -(y - \mu) V^{-1} \frac{\partial V}{\partial \theta} V^{-1} \frac{\partial V}{\partial \theta'} V^{-1} (y - \mu) \tag{4b}$$

$$+ \tfrac{1}{2} \operatorname{tr} \left(\frac{\partial V}{\partial \theta} V^{-1} \frac{\partial V}{\partial \theta'} V^{-1} \right) \tag{4c}$$

$$\frac{\partial^2 l}{\partial \mu \partial \theta} = -u' V^{-1} \frac{\partial V}{\partial \theta} V^{-1} (y - \mu) \tag{4d}$$

```
> info <- function(theta, y) {
+     if (length(theta) != 3)
+         stop("length(theta) != 3")
+     mu <- theta[1]
+     sigsq <- theta[2]
+     tausq <- theta[3]
+     vmat <- sigsq * rmat + tausq * imat
+     r <- y - mu
+     result <- matrix(NA, 3, 3)
+     foo <- solve(vmat, r)
+     vinv <- solve(vmat)
+     result[1, 1] <- sum(vinv)
+     result[1, 2] <- sum(vinv %*% rmat %*% foo)
+     result[1, 3] <- sum(vinv %*% imat %*% foo)
+     result[2, 1] <- result[1, 2]
+     result[3, 1] <- result[1, 3]
```

```
+     bar <- rmat %*% vinv %*% rmat
+     result[2, 2] <- t(foo) %*% bar %*% foo - (1/2) * sum(vinv *
+         bar)
+     bar <- imat %*% vinv %*% imat
+     result[3, 3] <- t(foo) %*% bar %*% foo - (1/2) * sum(vinv *
+         bar)
+     bar <- rmat %*% vinv %*% imat
+     result[2, 3] <- t(foo) %*% bar %*% foo - (1/2) * sum(vinv *
+         bar)
+     result[3, 2] <- result[2, 3]
+     return(result)
+ }
```

3.4 One-Step Newton

```
> theta.one <- theta.start + solve(info(theta.start, y = y), score(theta.start,
+     y = y))
> rbind(theta.start, theta.one, theta.hat)
```

```
                [,1]     [,2]      [,3]
theta.start 5.077667 1.461912 0.6174799
theta.one   5.149669 1.105019 0.8854382
theta.hat   5.136033 1.049012 1.0125677
```

4 Parametric Bootstrap

4.1 Data

Great! So now we try a simple parametric bootstrap

```
> nboot <- 100
> yboot <- matrix(NA, nboot, length(y))
> for (iboot in 1:nboot) {
+     zerovec <- rep(0, length(y))
+     gstar <- mvrnorm(mu = zerovec, Sigma = sig.hat^2 * rmat)
+     estar <- mvrnorm(mu = zerovec, Sigma = tau.hat^2 * imat)
+     yboot[iboot, ] <- mu.hat + gstar + estar
+ }
```

4.2 Semivariogram Estimation

First we bundle up our semivariogram estimation as a procedure

```
> semivariogram <- function(y) {
+     foo <- outer(y, y, "-")^2/2
+     foo <- foo[lower.tri(foo)]
+     bar <- outer(diag(rmat), diag(rmat), "+")/2 - rmat
+     bar <- bar[lower.tri(bar)]
+     gout <- glm(foo ~ bar, family = Gamma("identity"), mustart = rep(1,
+         length(foo)))
+     qux <- coef(gout)
+     if (qux[1] < 0) {
+         cat("genetic variance component estimate negative -- fixing up\n")
+         gout <- glm(foo ~ 1, family = Gamma("identity"), mustart = rep(1,
+             length(foo)))
+         qux <- c(0, coef(gout))
+     }
+     else if (qux[2] < 0) {
+         cat("environmental variance component estimate negative -- fixing up\n")
+         gout <- glm(foo ~ bar + 0, family = Gamma("identity"),
+             mustart = rep(1, length(foo)))
+         qux <- c(coef(gout), 0)
+     }
+     return(as.numeric(c(mean(y), qux)))
+ }

> theta.start.boot <- matrix(NA, nboot, 3)
> for (iboot in 1:nboot) {
+     theta.start.boot[iboot, ] <- semivariogram(yboot[iboot, ])
+ }

environmental variance component estimate negative -- fixing up
environmental variance component estimate negative -- fixing up
environmental variance component estimate negative -- fixing up
environmental variance component estimate negative -- fixing up
```

4.3 Maximum Likelihood

```
> theta.star <- matrix(NA, nboot, 3)
> conv.star <- rep(NA, nboot)
> for (iboot in 1:nboot) {
+     out <- optim(theta.start.boot[iboot, ], logl, score, lower = lower,
+         method = "L-BFGS-B", control = list(fnscale = -length(y)),
+         y = yboot[iboot, ])
```

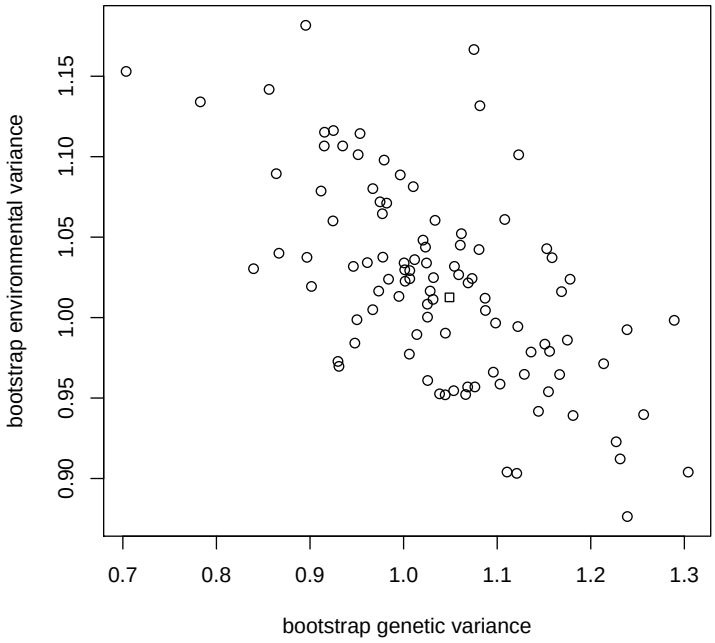

Figure 2: Parametric Bootstrap Distribution of MLE for Variance Components. Square is the MLE for the original data. Dots are MLE for bootstrap data.

```
+     theta.star[iboot, ] <- out$par
+     conv.star[iboot] <- out$convergence
+ }
> all(conv.star == 0)

[1] TRUE
```

The mean parameter μ is just a nuisance parameter. Let's look at the joint distribution of the other two. The distribution (Figure 2) seems fairly normal. The number of individuals $n = 2000$ does seem "large" but all of data y_i are highly correlated, there is no independence or stationarity that we can use to justify an appeal to the central limit theorem.

4.4 One-Step Newton

```
> theta.one.boot <- matrix(NA, nboot, 3)
> for (iboot in 1:nboot) {
+     foo <- theta.start.boot[iboot, ]
+     bar <- yboot[iboot, ]
+     H <- info(foo, y = bar)
+     g <- score(foo, y = bar)
+     q1 <- foo + solve(H, g)
+     if (any(q1[2:3] < 0)) {
+         cat("variance component estimate negative -- fixing up\n")
+         fred <- function(delta) {
+             baz <- delta - foo
+             as.numeric(t(baz) %*% g - (1/2) * t(baz) %*% H %*%
+                 baz)
+         }
+         sally <- function(delta) {
+             baz <- delta - foo
+             as.numeric(g - H %*% baz)
+         }
+         lower <- c(-Inf, 0, 0)
+         qout <- optim(foo, fred, sally, lower = lower, method = "L-BFGS-B",
+             control = list(fnscale = -1))
+         q1 <- qout$par
+     }
+     theta.one.boot[iboot, ] <- q1
+ }
```

Let us just consider how much progress Newton makes in one step. Figure 3 (page 14) should show the one-step Newton estimator nearly converging in one step. We can see from these plots that Newton does nearly converge in one step for a majority of the bootstrap random data sets, but not for all (as would be the case if we were really in "asymptopia").

4.5 Confidence Interval for Heritability

We make a confidence interval for heritability first by putting it on the log scale, getting first a confidence interval for $\phi = \log \sigma^2 - \log \tau^2$. Using the delta method or the formula for change-of-parameter with Fisher

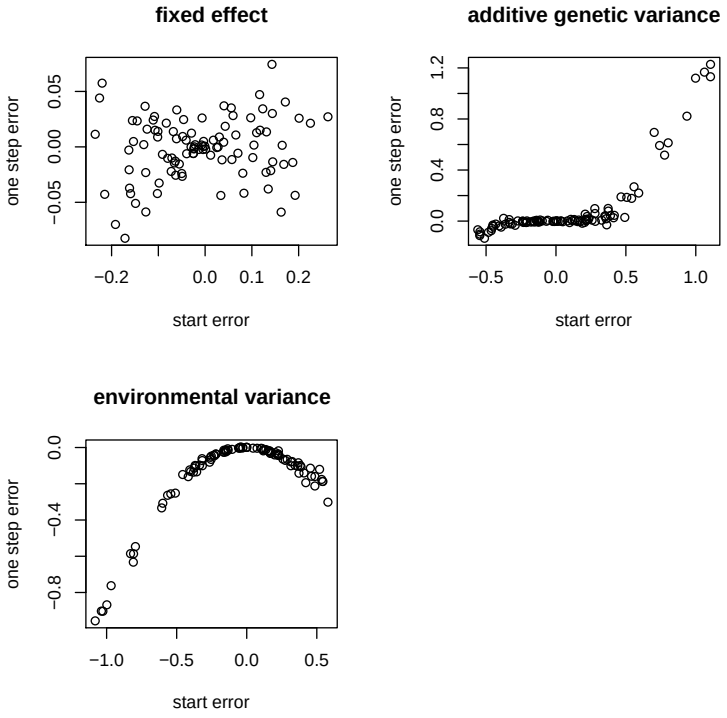

Figure 3: Error of starting point for Newton's method (semivariogram estimator) versus error of one-step estimator, where "error" is difference from MLE (infinite-step Newton estimator).

information, we obtain

$$\frac{I(\theta)^{22}}{\sigma^4} + \frac{I(\theta)^{33}}{\tau^4} - 2\frac{I(\theta)^{23}}{\sigma^2\tau^2}$$

for the asymptotic variance of ϕ, where $I(\theta)^{ij}$ is the (i, j) component of inverse (observed) Fisher information.

```
> phi.hat <- log(theta.hat[2]) - log(theta.hat[3])
> phi.star <- log(theta.star[, 2]) - log(theta.star[, 3])
> se.phi.star <- double(nboot)
> for (iboot in 1:nboot) {
+     foo <- theta.star[iboot, ]
+     bar <- yboot[iboot, ]
+     qux <- solve(info(foo, y = bar))
+     fred <- qux[2, 2]/foo[2]^2 + qux[3, 3]/foo[3]^2 - 2 * qux[2,
+         3]/(foo[2] * foo[3])
+     se.phi.star[iboot] <- sqrt(fred)
+ }
> z.phi.star <- (phi.star - phi.hat)/se.phi.star
> mean(phi.star)

[1] 0.01381908

> median(phi.star)

[1] 0.01316959
```

Figure 4 (page 16) shows the the distribution of the "standardized" bootstrap estimator of logit heritability. If we were in asymptopia, the points would lie on the line. That they lie below the line shows we have some bias. Figure 5 (page 17) shows a histogram of the bootstrap distribution of logit heritability ϕ^*. The histogram is clearly skewed. It is both mean biased and median biased

```
> mean(phi.star < phi.hat)

[1] 0.59

> mean(phi.star - phi.hat)

[1] -0.02154033
```

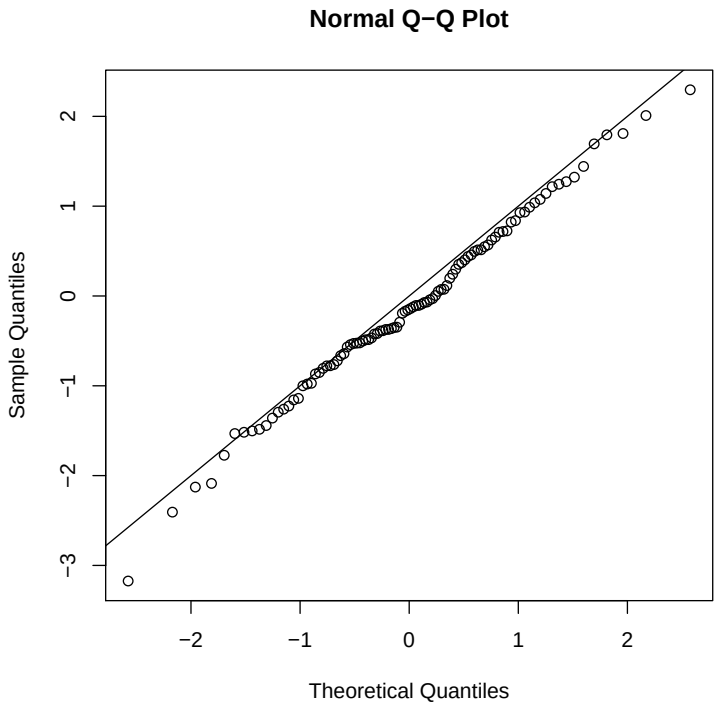

Figure 4: Q-Q Plot of Bootstrap Distribution of Pivot for Logit Heritability.

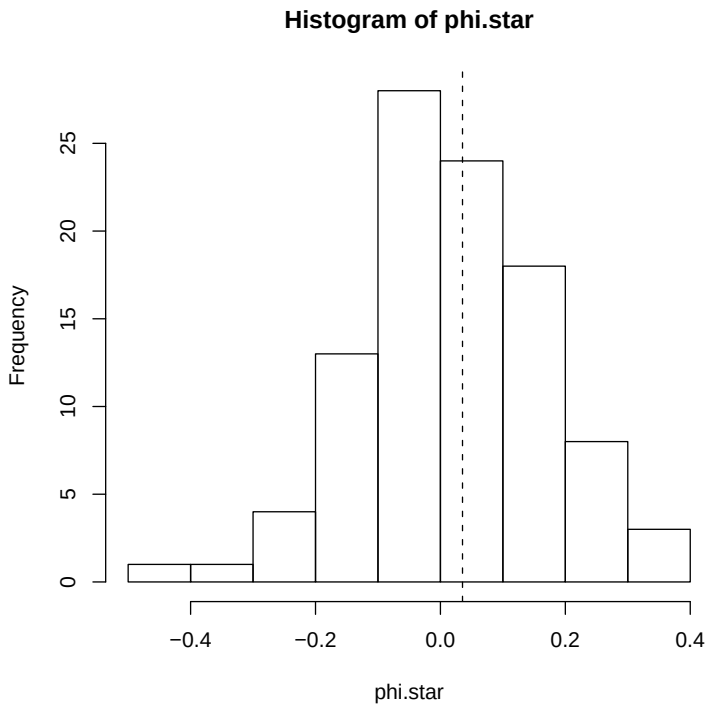

Figure 5: Histogram of Bootstrap Distribution of Logit Heritability. Dashed line shows original MLE.

So, again, we see that we are not in asymptopia. Figure 5 looks fairly normal, but not quite normal enough. A simple asymptotic 95% confidence interval for ϕ is

```
> qux <- solve(info(theta.hat, y = y))
> fred <- qux[2, 2]/theta.hat[2]^2 + qux[3, 3]/theta.hat[3]^2 -
+     2 * qux[2, 3]/(theta.hat[2] * theta.hat[3])
> se.phi.hat <- sqrt(fred)
> phi.hat + c(-1, 1) * qnorm(0.975) * se.phi.hat

[1] -0.2565746  0.3272934
```

whereas a 95% bootstrap-t confidence interval is

```
> crit <- as.numeric(quantile(z.phi.star, probs = c(0.025, 0.975)))
> phi.hat - rev(crit) * se.phi.hat

[1] -0.2330632  0.3493724
```

that they are not the same says a (single) parametric bootstrap is not pointless. It does provide nontrivial correction of the asymptotic interval.

5 Discussion

Admittedly, in this toy example, we have used a simulated pedigree structure that does have some stationarity (discrete generations and random mating), but that sort of stationarity assumption would not be appropriate for real data, even supposing we could base a traditional "n goes to infinity" story on it. Miller (1977) shows how difficult it is to make up such stories and prove theorems about them when the variance components are for highly structured designed experiments. It is not clear that there can be any analogous theory for quantitative genetics on general pedigrees, much less any theory relevant to actual applications.

The theory of "Le Cam Made Simple" applies without strain. We see that our problem is on the borderline of asymptopia. Asymptotics sort of works, but not perfectly. If we were to increase the pedigree size, we would, of course, eventually arrive in asymptopia. We do not provide such an example, because we think this "borderline" example, which illustrates what can go wrong and some (by no means all) ways to detect failure of asymptotics, is more informative than an example where asymptotics works perfectly.

Since all of the code is in the file `fisher.Rnw`. Any reader can do their own example. If you want to see what happens with larger or smaller pedigree sizes, just change `popsize` or `ngen` or `obsgen` in the first code chunk.

References

Fisher, R. A. (1918). The correlation between relatives on the supposition of Mendelian inheritance. *Trans. R. Soc. Edinburgh* **52** 399–433.

Henderson, C. R. (1976). A simple method for computing the inverse of a numerator relationship matrix used in prediction of breeding values. *Biometrics* **32** 69–83.

Johnson, D. L. and Thompson, R. (1995). Restricted maximum likelihood estimation of variance components for univariate animal models using sparse matrix techniques and average information. *J. Dairy Sci.* **78** 449–456.

Miller, J. J. (1977). Asymptotic properties of maximum likelihood estimates in the mixed model of the analysis of variance. *Ann. Statist.* **5** 746–762.

Patterson, H. D. and Thompson, R. (1971). Recovery of inter-block information when block sizes are unequal. *Biometrika* **58** 545–554.

Quaas, R. L. (1976). Computing the diagonal elements and inverse of a large numerator relationship matrix. *Biometrics* **32** 949–953.

Shaw, R. G. and Shaw, F. H. (1994). Quercus: programs for quantitative genetic analysis using maximum likelihood. `http://biosci.cbs.umn.edu/eeb/quercus.html`.

Thomas, S. C., Pemberton, J. M. and Hill, W. G. (2000). Estimating variance components in natural populations using inferred relationships. *Heredity* **84** 427–436.