An Example of Logit-Normal Modeling

Yun Ju Sung Charles J. Geyer

December 20, 2005

1 Introduction

Coull and Agresti (2000) propose models which can be fit using the methods of Sung and Geyer (submitted). Here we fit one of the models proposed in Coull and Agresti (2000) to the data about influenza whose provenance is described by Coull and Agresti (2000) using the R package `bernor`, which is available at `http://www.stat.umn.edu/geyer/bernor`.

1.1 Data

First we load the data, copied from Table 1 in Coull and Agresti (2000), and the package.

```
> library(bernor)
> data(flu)
> library(trust)
```

there are

```
> sum(flu$nobs)
```

```
[1] 263
```

independent and identically distributed (IID) individuals, each with four Bernoulli observations (Y_1, ..., Y_4). The meaning is that the observations are IID random vectors $y = (y_1, y_2, y_3, y_4)$ and the i-th pattern of y values (the i-th row of `flu`) is observed `flu$nobs[i]` times.

To put these data in the form the `bernor` package needs, we need to create a data matrix in which the columns are IID. Thus we do this by

```
> y <- rbind(flu$y1, flu$y2, flu$y3, flu$y4)
> dimnames(y) <- NULL
> w <- flu$nobs
> y <- y[, w > 0]
> w <- w[w > 0]
```

1.2 Model

We fit the model the random effects having variance matrix

$$\Sigma = \sigma^2 \begin{bmatrix} 1 & \rho_1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 & \rho_2 \\ \rho_1 & \rho_1 & 1 & \rho_2 \\ \rho_2 & \rho_2 & \rho_2 & 1 \end{bmatrix} \tag{1}$$

which is given by equation (7) in Coull and Agresti (2000). In order to use the `bernor` package, we need to write a normal random vector having variance matrix (1) in the form $Z\Delta b$ where Z is a fixed known matrix that does not contain parameters, Δ is a diagonal positive semi-definite matrix, and b is a standard normal random vector (having IID standard normal components). We try

$$Z = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

Then

$$\begin{aligned} \operatorname{var}(Z\Delta b) &= Z\Delta^2 Z^T \\ &= \begin{pmatrix} \delta_1^2 + \delta_2^2 + \delta_3^2 & \delta_1^2 + \delta_2^2 & \delta_1^2 + \delta_2^2 & \delta_1^2 - \delta_2^2 \\ \delta_1^2 + \delta_2^2 & \delta_1^2 + \delta_2^2 + \delta_4^2 & \delta_1^2 + \delta_2^2 & \delta_1^2 - \delta_2^2 \\ \delta_1^2 + \delta_2^2 & \delta_1^2 + \delta_2^2 & \delta_1^2 + \delta_2^2 + \delta_5^2 & \delta_1^2 - \delta_2^2 \\ \delta_1^2 - \delta_2^2 & \delta_1^2 - \delta_2^2 & \delta_1^2 - \delta_2^2 & \delta_1^2 + \delta_2^2 + \delta_6^2 \end{pmatrix} \end{aligned} \tag{3}$$

and we see that (1) and (3) do indeed have the same form if we impose the constraints

$$\delta_3 = \delta_4 = \delta_5 = \delta_6 \tag{4}$$

(which the `bernor` package is designed to allow) with one caveat. The representation (1) is positive semi-definite if

$$-\frac{1}{2} \le \rho_1 \le 1 \tag{5a}$$

$$\rho_2^2 \le \frac{1 + 2\rho_1}{3} \tag{5b}$$

whereas the representation (3) is necessarily positive semi-definite, but as the δ_i range over all non-negative numbers the correlations

$$\rho_1 = \frac{\delta_1^2 + \delta_2^2}{\delta_1^2 + \delta_2^2 + \delta_3^2}$$

$$\rho_2 = \frac{\delta_1^2 - \delta_2^2}{\delta_1^2 + \delta_2^2 + \delta_3^2}$$

only fill out the set

$$0 \le \rho_1 \le 1 \tag{6a}$$

$$-\rho_1 \le \rho_2 \le \rho_1 \tag{6b}$$

Hence our $Z\Delta b$ model is a restriction of the full model specified by (1).

Nevertheless, if the MLE is in the interior of the parameter space for our $Z\Delta b$ model, then it is the same as the MLE for the full model. To completely specify a `bernor` model we must specify the design matrices

```
> x <- diag(4)
> z <- rbind(c(1, 1, 1, 0, 0, 0), c(1, 1, 0, 1, 0,
+     0), c(1, 1, 0, 0, 1, 0), c(1, -1, 0, 0, 0, 1))
> idx <- c(1, 2, 3, 3, 3, 3)
```

the last vector being used to impose the constraint (4).

1.3 Putative MLE

Coull and Agresti (2000) give the following MLE for this model.

```
> beta.putative <- c(-4, -4.4, -4.7, -4.5)
> sigmasq.putative <- 4.05^2
> rho1.putative <- 0.43
> rho2.putative <- (-0.25)
> delta3.putative <- sqrt(sigmasq.putative * (1 - rho1.putative))
> delta2.putative <- sqrt(sigmasq.putative * (rho1.putative -
```

```
+     rho2.putative)/2)
> delta1.putative <- sqrt(sigmasq.putative * (rho1.putative +
+     rho2.putative)/2)
> delta.putative <- c(delta1.putative, delta2.putative,
+     delta3.putative)
> theta.putative <- c(beta.putative, delta.putative)
> print(theta.putative)

[1] -4.000000 -4.400000 -4.700000 -4.500000  1.215000  2.361536
[7]  3.057683
```

1.4 Non-Mixed Model MLE

```
> nfix <- ncol(x)
> nran <- max(idx)
> nparm <- nfix + nran
> beta.start <- as.numeric(y %*% w)/sum(w)
> beta.start <- log(beta.start) - log(1 - beta.start)
> delta.start <- rep(0, nparm - nfix)
> theta.start <- c(beta.start, delta.start)
> print(theta.start)

[1] -1.499437 -1.660493 -1.778514 -1.718292  0.000000  0.000000
[7]  0.000000
```

2 Optimization Constrained to Cylinder

As in the file `spath.tex` we denote the constraint by $\theta' D \theta = \Delta^2$. We record the diagonal of D as "little d"

```
> d <- c(rep(0, nfix), rep(1, nparm - nfix))
```

Now we encode a Lagrange-Newton update. The current position is `theta`, `lambda` and the constraint length is `DeltaSq`.

```
> lnupdate <- function(theta, lambda) {
+     beta <- theta[seq(1, nfix)]
+     delta <- theta[-seq(1, nfix)]
+     .Random.seed <<- .save.Random.seed
+     foo <- bnlogl(y, beta, delta, nmiss, x, z, idx,
+         moo, deriv = 2, weigh = w)
```

```
+     a <- 2 * d * theta
+     b <- c(foo$gradient + lambda * a, sum(d * theta^2) -
+         DeltaSq)
+     A11 <- foo$hessian + 2 * lambda * diag(d)
+     A <- rbind(cbind(A11, a), c(a, 0))
+     z <- solve(A, b)
+     nz <- length(z)
+     theta <- theta - z[-nz]
+     lambda <- lambda - z[nz]
+     return(list(theta = theta, lambda = lambda, error = b,
+         step = z, value = foo$value, gradient = foo$gradient,
+         hessian = foo$hessian))
+ }

> set.seed(42)
> .save.Random.seed <- .Random.seed
> moo <- model("gaussian", length(idx), 1)
> nmiss <- 1e+06
```

3 Homotopy Path

Now we want to find the homotopy path, a continuous path of equality constrained local maxima, the equality constraint being $\theta' D \theta = \Delta^2$. Let θ_Δ be such a local solution. We wish to find a smooth path $\Delta \mapsto \theta_\Delta$.

3.1 Shrinking

In this section we try to obtain the path on $[0, \Delta_p]$, where $\Delta_p^2 = \theta_p' D \theta_p$ and where θ_p is what we call `theta.putative` in R.

We evaluate the path on a grid.

```
> DeltaStep <- 0.2
> DeltaP <- sqrt(sum(d * theta.putative^2))
> DeltaSeq <- seq(0, DeltaP, DeltaStep)
> DeltaSeq <- DeltaSeq[-1]
```

Setup memory for the path.

```
> nstep <- length(DeltaSeq)
> logl.path <- rep(NA, nstep)
> theta.path <- matrix(NA, nstep, nparm)
```

```
> grad.path <- matrix(NA, nstep, nparm)
> eigen.path <- matrix(NA, nstep, nparm)
```

And do it.

```
> theta <- theta.putative
> lambda <- 0
> for (ipath in rev(seq(along = DeltaSeq))) {
+     DeltaSq <- DeltaSeq[ipath]^2
+     repeat {
+         lout <- lnupdate(theta, lambda)
+         if (max(abs(lout$error)) < 1e-06)
+             break
+         theta <- lout$theta
+         lambda <- lout$lambda
+     }
+     logl.path[ipath] <- lout$value
+     theta.path[ipath, ] <- theta
+     grad.path[ipath, ] <- lout$gradient
+     eout <- eigen(lout$hessian, symmetric = TRUE)
+     eigen.path[ipath, ] <- eout$values
+ }
```

3.2 Zero

Special case $\Delta = 0$

```
> DeltaSeq <- c(0, DeltaSeq)
> theta.path <- rbind(theta.start, theta.path)
> .Random.seed <<- .save.Random.seed
> foo <- bnlogl(y, beta.start, delta.start, nmiss,
+     x, z, idx, moo, deriv = 2, weigh = w)
> logl.path <- c(foo$value, logl.path)
> grad.path <- rbind(foo$gradient, grad.path)
> eout <- eigen(foo$hessian, symmetric = TRUE)
> eigen.path <- rbind(eout$values, eigen.path)
```

3.3 Expanding

```
> mstep <- 10
```

In this section we try to obtain the path on (Δ_p, ∞). Of course, we don't go all the way to infinity. In fact we only take another 10 steps on the same grid.

```
> nstep <- length(DeltaSeq)
> theta <- theta.path[nstep, ]
> Delta <- DeltaSeq[nstep]
> lambda <- 0
> for (ipath in 1:mstep) {
+     Delta <- Delta + DeltaStep
+     DeltaSq <- Delta^2
+     repeat {
+         lout <- lnupdate(theta, lambda)
+         if (max(abs(lout$error)) < 1e-06)
+             break
+         theta <- lout$theta
+         lambda <- lout$lambda
+     }
+     DeltaSeq <- c(DeltaSeq, Delta)
+     logl.path <- c(logl.path, lout$value)
+     theta.path <- rbind(theta.path, theta)
+     grad.path <- rbind(grad.path, lout$gradient)
+     eout <- eigen(lout$hessian, symmetric = TRUE)
+     eigen.path <- rbind(eigen.path, eout$values)
+ }
```

3.4 Compare Putative

Evaluate at `theta.putative`.

```
> .Random.seed <- .save.Random.seed
> pout <- bnlogl(y, beta.putative, delta.putative,
+     nmiss, x, z, idx, moo, deriv = 2, weigh = w)
> lapply(pout, round, digits = 3)

$value
[1] -448.66

$gradient
[1] -0.167 -0.113  0.096 -0.313 -0.080 -0.314 -0.312
```

```
$hessian
       [,1]   [,2]   [,3]   [,4]   [,5]    [,6]    [,7]
[1,] -6.731  1.014  0.959 -0.400 -1.181  -2.143  -3.230
[2,]  1.014 -6.322  0.914 -0.362 -1.281  -2.224  -3.483
[3,]  0.959  0.914 -6.022 -0.341 -1.341  -2.280  -3.687
[4,] -0.400 -0.362 -0.341 -5.880 -2.066  -3.394  -5.022
[5,] -1.181 -1.281 -1.341 -2.066 -5.515  -1.723  -3.697
[6,] -2.143 -2.224 -2.280 -3.394 -1.723 -11.644  -3.586
[7,] -3.230 -3.483 -3.687 -5.022 -3.697  -3.586 -13.506
```

4 Blather

```
> dimnames(theta.path) <- NULL
> dimnames(grad.path) <- NULL
> dimnames(eigen.path) <- NULL

> round(logl.path, 3)

 [1] -461.220 -460.155 -457.446 -454.200 -451.461 -449.728
 [7] -448.871 -448.740 -448.716 -448.699 -448.685 -448.676
[13] -448.668 -448.663 -448.658 -448.655 -448.653 -448.651
[19] -448.649 -448.648 -448.647 -448.647 -448.646 -448.646
[25] -448.646 -448.646 -448.646 -448.646 -448.646 -448.646
[31] -448.646


> round(theta.path, 3)

        [,1]   [,2]   [,3]   [,4]  [,5]  [,6]   [,7]
 [1,] -1.499 -1.660 -1.779 -1.718 0.000 0.000  0.000
 [2,] -1.512 -1.674 -1.793 -1.732 0.000 0.200  0.000
 [3,] -1.551 -1.715 -1.836 -1.774 0.001 0.400  0.000
 [4,] -1.612 -1.782 -1.906 -1.841 0.002 0.600  0.000
 [5,] -1.693 -1.869 -1.998 -1.930 0.008 0.800  0.000
 [6,] -1.787 -1.972 -2.106 -2.032 0.285 0.958  0.000
 [7,] -1.892 -2.086 -2.227 -2.148 0.486 1.097 -0.001
 [8,] -2.007 -2.211 -2.358 -2.279 0.591 1.202  0.408
 [9,] -2.134 -2.349 -2.503 -2.421 0.633 1.271  0.737
[10,] -2.267 -2.495 -2.656 -2.571 0.677 1.346  0.985
[11,] -2.407 -2.647 -2.816 -2.727 0.723 1.424  1.204
```

```
[12,] -2.551 -2.805 -2.983 -2.890 0.769 1.505  1.408
[13,] -2.700 -2.967 -3.154 -3.057 0.816 1.590  1.602
[14,] -2.852 -3.134 -3.329 -3.229 0.864 1.676  1.790
[15,] -3.007 -3.303 -3.509 -3.404 0.913 1.765  1.973
[16,] -3.164 -3.476 -3.691 -3.582 0.962 1.856  2.152
[17,] -3.324 -3.651 -3.877 -3.762 1.012 1.948  2.329
[18,] -3.486 -3.829 -4.064 -3.945 1.062 2.041  2.503
[19,] -3.649 -4.008 -4.254 -4.130 1.113 2.136  2.676
[20,] -3.815 -4.189 -4.446 -4.317 1.164 2.232  2.847
[21,] -3.981 -4.372 -4.639 -4.505 1.215 2.328  3.017
[22,] -4.149 -4.555 -4.834 -4.695 1.267 2.425  3.186
[23,] -4.318 -4.740 -5.030 -4.886 1.319 2.523  3.355
[24,] -4.487 -4.927 -5.228 -5.078 1.371 2.622  3.522
[25,] -4.658 -5.114 -5.426 -5.271 1.423 2.721  3.689
[26,] -4.829 -5.301 -5.625 -5.464 1.476 2.821  3.855
[27,] -5.001 -5.490 -5.826 -5.659 1.529 2.921  4.021
[28,] -5.174 -5.680 -6.027 -5.855 1.582 3.022  4.187
[29,] -5.348 -5.870 -6.228 -6.051 1.635 3.122  4.352
[30,] -5.521 -6.060 -6.430 -6.247 1.688 3.224  4.516
[31,] -5.696 -6.251 -6.633 -6.444 1.742 3.325  4.681
```

```
> round(grad.path, 3)

      [,1] [,2] [,3] [,4]  [,5]   [,6]   [,7]
 [1,]    0    0    0    0 0.000  0.000  0.000
 [2,]    0    0    0    0 0.025 10.207  0.001
 [3,]    0    0    0    0 0.047 15.893  0.002
 [4,]    0    0    0    0 0.062 15.665  0.001
 [5,]    0    0    0    0 0.112 11.249 -0.001
 [6,]    0    0    0    0 1.835  6.163 -0.001
 [7,]    0    0    0    0 0.880  1.987 -0.002
 [8,]    0    0    0    0 0.058  0.118  0.040
 [9,]    0    0    0    0 0.041  0.082  0.047
[10,]    0    0    0    0 0.029  0.057  0.042
[11,]    0    0    0    0 0.020  0.040  0.034
[12,]    0    0    0    0 0.015  0.029  0.027
[13,]    0    0    0    0 0.011  0.021  0.021
[14,]    0    0    0    0 0.008  0.016  0.017
[15,]    0    0    0    0 0.006  0.012  0.013
```

```
[16,]    0    0    0    0 0.005  0.009  0.010
[17,]    0    0    0    0 0.003  0.007  0.008
[18,]    0    0    0    0 0.003  0.005  0.006
[19,]    0    0    0    0 0.002  0.004  0.005
[20,]    0    0    0    0 0.001  0.003  0.004
[21,]    0    0    0    0 0.001  0.002  0.003
[22,]    0    0    0    0 0.001  0.002  0.002
[23,]    0    0    0    0 0.001  0.001  0.001
[24,]    0    0    0    0 0.000  0.001  0.001
[25,]    0    0    0    0 0.000  0.001  0.001
[26,]    0    0    0    0 0.000  0.000  0.000
[27,]    0    0    0    0 0.000  0.000  0.000
[28,]    0    0    0    0 0.000  0.000  0.000
[29,]    0    0    0    0 0.000  0.000  0.000
[30,]    0    0    0    0 0.000  0.000  0.000
[31,]    0    0    0    0 0.000  0.000  0.000
```

```
> round(eigen.path, 3)
```

```
        [,1]   [,2]    [,3]    [,4]    [,5]    [,6]    [,7]
 [1,] 55.471 21.723   0.026 -32.510 -33.916 -35.293 -39.240
 [2,] 41.113 20.778  -0.555 -32.345 -33.791 -35.373 -39.139
 [3,] 18.083 10.782  -1.835 -31.069 -33.192 -36.419 -41.914
 [4,] 14.414 -2.833  -7.649 -28.537 -32.227 -35.750 -57.224
 [5,] 11.106 -2.811 -11.285 -25.537 -31.032 -34.384 -72.742
 [6,]  0.821 -2.012 -11.083 -22.802 -29.707 -32.835 -75.597
 [7,] -0.699 -5.665 -14.527 -20.309 -28.363 -31.274 -71.490
 [8,]  0.029 -7.251 -16.989 -18.994 -26.318 -28.906 -65.592
 [9,] -0.021 -8.901 -15.660 -18.816 -23.652 -25.848 -60.313
[10,] -0.024 -9.265 -14.123 -18.713 -21.209 -23.084 -55.547
[11,] -0.020 -9.017 -12.683 -18.268 -19.011 -20.624 -51.222
[12,] -0.015 -8.513 -11.388 -17.053 -17.508 -18.456 -47.273
[13,] -0.011 -7.919 -10.241 -15.326 -16.511 -16.585 -43.656
[14,] -0.008 -7.312  -9.230 -13.807 -14.882 -15.491 -40.338
[15,] -0.006 -6.727  -8.341 -12.471 -13.425 -14.414 -37.295
[16,] -0.005 -6.181  -7.560 -11.297 -12.149 -13.366 -34.507
[17,] -0.003 -5.679  -6.872 -10.265 -11.028 -12.373 -31.957
[18,] -0.003 -5.223  -6.266  -9.355 -10.043 -11.447 -29.627
[19,] -0.002 -4.809  -5.730  -8.551  -9.174 -10.594 -27.500
```

```
[20,] -0.002 -4.436  -5.255  -7.839  -8.406  -9.812 -25.560
[21,] -0.001 -4.100  -4.833  -7.206  -7.724  -9.098 -23.790
[22,] -0.001 -3.797  -4.457  -6.643  -7.118  -8.448 -22.175
[23,] -0.001 -3.523  -4.121  -6.139  -6.576  -7.856 -20.700
[24,] -0.001 -3.275  -3.820  -5.688  -6.092  -7.318 -19.353
[25,]  0.000 -3.051  -3.549  -5.282  -5.656  -6.828 -18.120
[26,]  0.000 -2.848  -3.304  -4.917  -5.264  -6.382 -16.992
[27,]  0.000 -2.664  -3.083  -4.586  -4.909  -5.974 -15.957
[28,]  0.000 -2.496  -2.883  -4.287  -4.588  -5.602 -15.007
[29,]  0.000 -2.343  -2.701  -4.015  -4.297  -5.262 -14.134
[30,]  0.000 -2.203  -2.535  -3.767  -4.032  -4.950 -13.330
[31,]  0.000 -2.074  -2.384  -3.541  -3.789  -4.664 -12.588
```

5 Picture

Figure 1 is produced by the following code

```
> plot(DeltaSeq, logl.path, xlab = expression(Delta),
+     ylab = "Monte Carlo log likelihood")
> points(DeltaP, pout$value, pch = 23)
```

and appears on p. 12.

6 Wind Up

```
> foo <- proc.time()[1]
> fooh <- floor(foo/60^2)
> foo <- foo - fooh * 60^2
> foom <- floor(foo/60)
> foo <- foo - foom * 60
> cat("total elapsed time:", fooh, "hours,", foom,
+     "minutes, and", foo, "seconds\n")

total elapsed time: 7 hours, 46 minutes, and 50.21 seconds

> foo <- try(system("hostname -f", intern = TRUE))
> if (!inherits(foo, "try-error")) cat("machine:",
+     foo, "\n")

machine: oak.stat.umn.edu
```

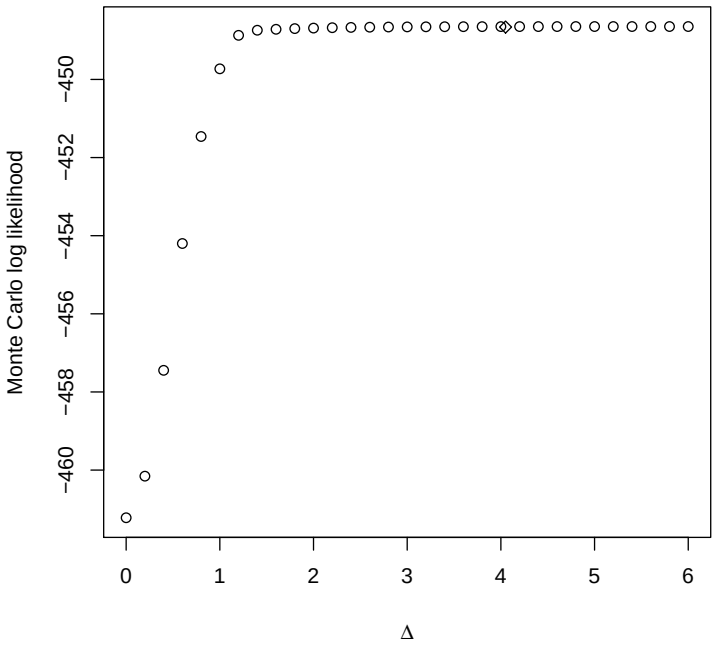

Figure 1: Monte Carlo approximation to log likelihood along the homotopy path examined in this document. Leftmost point is the MLE $\hat{\theta}_{\text{NRE}}$ in the model with no random effects (all variance components zero). Other points are MLE subject to the equality constraint $\Delta^2 = \delta_1^2 + \delta_2^2 + \delta_3^2$, where Δ is the horizontal coordinate in the plot.

```
> save(y, x, z, idx, w, nfix, nran, nparm, .save.Random.seed,
+     moo, theta.start, theta.putative, pout, DeltaSeq,
+     logl.path, theta.path, grad.path, eigen.path,
+     file = "flu6p.RData")
```

References

Sung, Y. J. and Geyer, C. J. (submitted). Monte Carlo likelihood inference for missing data models. `http://www.stat.umn.edu/geyer/bernor/ms.pdf`.

Coull, B. A. and Agresti, A. (2000). Random effects modeling of multiple binomial responses using the multivariate binomial logit-normal distribution. *Biometrics*, **56**, 73–80.