The Salamanders Logit-Normal GLMM Example

Yun Ju Sung   Charles J. Geyer

January 24, 2006

1   Logit-Normal GLMM

In a Logit-Normal generalized linear mixed model (GLMM), the observed data is a vector y whose components are conditionally independent Bernoulli random variables given the missing data vector b, which is unconditionally jointly mean-zero multivariate normal. The model specification is completed by the specification of the *linear predictor*

$$\eta = X\beta + Zb \tag{1}$$

and the link function. In (1) X and Z are known matrices (the "design" or "model" matrices for fixed and random effects, respectively), β is a vector of unknown parameters ("fixed effects"), b is the vector of missing data ("random effects"), and the conditional expectation of y given η is $\text{logit}^{-1}(\eta)$.

The unknown parameters to be estimated are β and any unknown parameters determining the variance matrix of b. Usually this variance matrix has simple structure and involves only a few unknown parameters. For this paper we have written an R package `bernor` that implements the methods of this paper for a class of Logit-Normal GLMM. The class of models our package handles is more easily described in R than in mathematical notation. The linear predictor has the form

$$\texttt{eta = X \%*\% beta + Z \%*\% (sigma[i] * b)} \tag{2}$$

where `X` and `Z` are the matrices X and Z in (1) and `X %*% beta` is the matrix multiplication $X\beta$ so the only way in which (1) differs from (2) other than notationally is that b in (1) is replaced by (`sigma[i] * b`) in (2), which, for readers not familiar with R, has the following interpretation: `sigma` is a vector of unknown parameters, `i` is a vector of the same length as `b` and having values that are possible indices for `sigma`, so `sigma[i]` is the vector $(\sigma_{i_1}, \ldots, \sigma_{i_m})$ in ordinary mathematical notation and `*` in (2) denotes coordinatewise multiplication, so if z_{jk} are the components of the matrix Z the second term on the right hand side of (2) has j-th component

$$\sum_{k=1}^{m} z_{jk}\sigma_{i_k}b_k$$

1

written in conventional mathematical notation.

We also change assumptions; in (1) b is general multivariate normal, but in (2) b is standard multivariate normal (mean vector zero, variance matrix the identity). Thus the only unknown parameters in our model are the vectors beta and sigma. Thus our package only deals with the simple situation in which the random effects are (unconditionally) independent.

We also allow for independent and identically distributed (IID) data, in which case the data y is a matrix with IID columns, each column of y modeled as described above.

2   Monte Carlo Maximum Likelihood

Our package provides no optimization capabilities, only evaluation of the log likelihood, its derivatives, and related quantities. For optimization we use trust.

```
> library(trust)
```

3   The Salamander Data

McCullagh and Nelder (1989, Section 14.5) discuss a "salamander mating experiment" whose data has been used by several groups of statisticians as an example of data appropriately analyzed by Logistic-Normal GLMM (see Booth and Hobert, 1999, for one analysis and citations of others), although McCullagh and Nelder (1989) did not use a GLMM and it is not clear that GLMM analyses address any of the questions of scientific interest for which the data were collected. Thus in the GLMM context these data are also "toy data" albeit not especially constructed to be such.

These data are the dataset salam in our bernor package. We are using what Booth and Hobert (1999) call "Model A" of Karim and Zeger (1992).

```
> library(bernor)
> data(salam)
> attach(salam)
> nfix <- ncol(x)
> nran <- max(i)
> nparm <- nfix + nran
> moo <- model("gaussian", length(i), 1)
> set.seed(42)
> .save.Random.seed <- .Random.seed
> nobs <- ncol(y)
> nmiss <- 1e+07
```

3.1 No Random Effects Model

```
> ysucc <- apply(y, 1, sum)
> yfail <- apply(1 - y, 1, sum)
> yresp <- cbind(ysucc, yfail)
> sout <- glm(yresp ~ x + 0, family = binomial)
> theta.start <- rep(0, nparm)
> theta.start[1:nfix] <- coefficients(sout)
> names(theta.start) <- c(dimnames(x)[[2]], paste("sigma", c("f",
+     "m"), sep = "_"))
> round(theta.start, 4)

    R/R     R/W     W/R     W/W sigma_f sigma_m
 0.6931  0.2231 -1.3182  0.6931  0.0000  0.0000
```

3.2 General Model, With Random Effects

3.2.1 Putative MLE

Booth and Hobert (1999) give the following MLE

```
> mu <- c(1.03, 0.32, -1.95, 0.99)
> sigmasq <- c(1.4, 1.25)
> theta.hat.booth <- c(mu, sqrt(sigmasq))
> names(theta.hat.booth) <- names(theta.start)
> print(theta.hat.booth, digits = 4)

    R/R     R/W     W/R     W/W sigma_f sigma_m
  1.030   0.320  -1.950   0.990   1.183   1.118
```

We have independently verified using MCMC that the latter appear to be correct to three significant figures. At least the MCMC approximation to the score is zero to three significant figures at this point.

3.2.2 Our MCMLE

The following evaluates the objective function (Monte Carlo approximation to the log likelihood) and its first two derivatives.

```
> objfun <- function(theta) {
+     beta <- theta[seq(1, nfix)]
+     delta <- theta[-seq(1, nfix)]
+     .Random.seed <<- .save.Random.seed
+     bnlogl(y, beta, delta, nmiss, x, z, i, moo, deriv = 2)
+ }
```

We then hand this to trust for maximization

```
> tout <- trust(objfun, theta.start, 0.1, 1, minimize = FALSE)
> print(tout$converged)
```

3

```
[1] TRUE

> print(tout$iterations)

[1] 11

> print(tout$value)

[1] -209.3673

> theta.hat <- tout$argument
> print(theta.hat, digits = 4)

    R/R     R/W     W/R     W/W sigma_f sigma_m
 0.9389  0.2742 -1.7618  0.9078 -0.9700 -0.9211
```

The current version of `bernor` does not enforce positivity constraints on the variance component parameters. This gives us an unfair advantage, since we have four choices of signs and hence four parameter values that are non-identifiable according to the exact likelihood but not according to the Monte Carlo likelihood. To avoid this unfair advantage, we force positivity and redo (if necessary)

```
> if (any(theta.hat[seq(nfix + 1, nparm)] < 0)) {
+     theta.hat[seq(nfix + 1, nparm)] <- abs(theta.hat[seq(nfix +
+         1, nparm)])
+     tout <- trust(objfun, theta.hat, 0.1, 1, minimize = FALSE)
+     print(tout$converged)
+     print(tout$iterations)
+     print(tout$value)
+     theta.hat <- tout$argument
+     print(theta.hat, digits = 4)
+ }

[1] TRUE
[1] 6
[1] -207.5881
    R/R     R/W     W/R     W/W sigma_f sigma_m
 1.0043  0.5337 -1.7829  1.2675  1.0988  1.1668
```

4   Monte Carlo Standard Errors

Standard errors for our method involve the matrices J, V, and W that are estimated as follows.

```
> beta.hat <- theta.hat[seq(1, nfix)]
> delta.hat <- theta.hat[-seq(1, nfix)]
> .Random.seed <- .save.Random.seed
```

4

```
> jvout <- bnlogl(y, beta.hat, delta.hat, nmiss, x, z, i, moo,
+     deriv = 3)
> .Random.seed <- .save.Random.seed
> wout <- bnbigw(y, beta.hat, delta.hat, nmiss, x, z, i, moo)
> nobs <- ncol(y)
> bigJ <- (-jvout$hessian)/nobs
> eigen(bigJ, symmetric = TRUE, only.values = TRUE)$values

[1] 10.9677435  8.9907365  4.4604794  2.8737272  2.3545394  0.8960508

> bigV <- jvout$bigv
> bigW <- wout
> bigSV <- solve(bigJ) %*% (bigV/nobs) %*% solve(bigJ)
> bigSW <- solve(bigJ) %*% (bigW/nmiss) %*% solve(bigJ)
> bigS <- bigSV + bigSW
> round(sqrt(diag(bigS)), 4)

[1] 0.1606 0.2710 0.1005 0.6055 0.1486 0.2370

> round(sqrt(diag(bigSV)), 4)

[1] 0.0713 0.2252 0.0912 0.4347 0.1452 0.1273

> round(sqrt(diag(bigSW)), 4)

[1] 0.1439 0.1508 0.0424 0.4216 0.0316 0.1999
```

5   Mahalanobis Distance

```
> delta <- theta.hat - theta.hat.booth
> as.numeric(delta %*% solve(bigS) %*% delta)

[1] 78.95967

> qchisq(0.975, nparm)

[1] 14.44938
```

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: 273 hours, 10 minutes, and 31.99 seconds

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

machine: moon.stat.umn.edu

> save(nmiss, nfix, nran, nparm, .save.Random.seed, moo, theta.start,
+      theta.hat.booth, theta.hat, jvout, wout, tout, file = "example7.RData")
```

References

Booth, J. G. and Hobert, J. P. (1999) Maximizing generalized linear mixed model likelihoods with an automated Monte Carlo EM algorithm. *Journal of the Royal Statistical Society Series B (Statistical Methodology)* 61, 265–285.

Karim, M. R. and Zeger, S. L. (1992) Generalized Linear Models with Random Effects: Salamander Mating Revisited. *Biometrics*, 48, 631–644.

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.