# More Supporting Data Analysis for "Likelihood Inference in Exponential Families and Directions of Recession"

By

Charles J. Geyer

Technical Report No. 673
School of Statistics
University of Minnesota
March 24, 2009

# 1 R Package Rcdd

We use the R statistical computing environment (R Development Core Team, 2008) in our analysis. It is free software and can be obtained from `http://cran.r-project.org`. Precompiled binaries are available for Windows, Macintosh, and popular Linux distributions. We use the contributed package `rcdd`. If R has been installed, but this package has not yet been installed, do

```
install.packages("rcdd")
```

from the R command line (or do the equivalent using the GUI menus if on Macintosh or Windows). This may require root or administrator privileges.

If the `rcdd` package has been installed, we load it

```
> library(rcdd)
```

The version of the package used to make this document is 1.1-1. The version of R used to make this document is 2.8.1.

This entire document and all of the calculations shown were made using the R command `Sweave` and hence are exactly reproducible by anyone who has R and the R noweb (RNW) file from which it was created. Both the RNW file and and the PDF document produced from it are available at `http://www.stat.umn.edu/geyer/gdor`.

# 2 Introduction

This technical report provides one more example, suggested by the referee, of an exponential family model in which solutions are "at infinity" so the maximum likelihood estimate (MLE) does not exist in the conventional sense. This is a Bradley-Terry model without ties. When team $i$ plays team $j$ the probability that team $i$ wins is $\operatorname{logit}^{-1}(\theta_i - \theta_j)$, where

$$\operatorname{logit}^{-1}(\theta) = \frac{1}{1 + \exp(-\theta)}.$$

# 3 Data

## 3.1 Matrix Form

We make up such data below for a sports league with 8 teams and a season in which each team plays each other team twice, so each team plays 14 games.

```
> team.names <- c("ants", "beetles", "cows", "dogs",
+     "egrets", "foxes", "gerbils", "hogs")
> data <- matrix(c(NA, 2, 2, 2, 2, 2, 2, 2, 0, NA,
+     1, 2, 2, 2, 2, 2, 0, 1, NA, 2, 1, 2, 2, 2, 0,
+     0, 0, NA, 1, 1, 2, 2, 0, 0, 1, 1, NA, 1, 2, 2,
+     0, 0, 0, 1, 1, NA, 2, 2, 0, 0, 0, 0, 0, 0, NA,
+     1, 0, 0, 0, 0, 0, 0, 1, NA), byrow = TRUE, nrow = 8)
> dimnames(data) <- list(team.names, team.names)
> print(data)
```

Table 1: Data for Bradley-Terry model. Entries give number of games (out of 2) that the team which is the row label beat the team which is the column label.

| | ants | beetles | cows | dogs | egrets | foxes | gerbils | hogs |
|---|---|---|---|---|---|---|---|---|
| ants | | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| beetles | 0 | | 1 | 2 | 2 | 2 | 2 | 2 |
| cows | 0 | 1 | | 2 | 1 | 2 | 2 | 2 |
| dogs | 0 | 0 | 0 | | 1 | 1 | 2 | 2 |
| egrets | 0 | 0 | 1 | 1 | | 1 | 2 | 2 |
| foxes | 0 | 0 | 0 | 1 | 1 | | 2 | 2 |
| gerbils | 0 | 0 | 0 | 0 | 0 | 0 | | 1 |
| hogs | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |

```
        ants beetles cows dogs egrets foxes gerbils hogs
ants      NA       2    2    2      2     2       2    2
beetles    0      NA    1    2      2     2       2    2
cows       0       1   NA    2      1     2       2    2
dogs       0       0    0   NA      1     1       2    2
egrets     0       0    1    1     NA     1       2    2
foxes      0       0    0    1      1    NA       2    2
gerbils    0       0    0    0      0     0      NA    1
hogs       0       0    0    0      0     0       1   NA
```

And we make it a nice LaTeX table (Table 1, page 2).

```
> library(xtable)
> print(xtable(data, align = "l|cccccccc", caption = paste("Data for Bradley-Terry model.",
+     "Entries give number", "of games (out of 2) that the team which is the row label beat
+     "team which is the column label."), digits = 0,
+     label = "tab:two"), caption.placement = "top",
+     table.placement = "tbp", include.rownames = TRUE)
```

## 3.2 Vector Form

Now we need to put these data in a form preferred by the R function glm, where the data are strung out in one long vector.

```
> wins <- data[upper.tri(data)]
> team.plus <- row(data)[upper.tri(data)]
> team.minus <- col(data)[upper.tri(data)]
> modmat <- matrix(0, length(wins), nrow(data))
> for (i in 1:ncol(modmat)) {
+     modmat[team.plus == i, i] <- 1
+     modmat[team.minus == i, i] <- (-1)
+ }
```

```
> losses <- 2 - wins
> resp <- cbind(wins, losses)
```

# 4 Try to Fit using GLM

Now we are ready to attempt to fit the model with `glm`.

```
> out <- glm(resp ~ modmat + 0, family = binomial)
> summary(out)

Call:
glm(formula = resp ~ modmat + 0, family = binomial)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-1.169e+00   6.222e-06   2.243e-05   4.374e-05   6.153e-01

Coefficients: (1 not defined because of singularities)
         Estimate Std. Error  z value Pr(>|z|)
modmat1 4.574e+01  1.529e+04    0.003    0.998
modmat2 2.479e+01  9.789e+03    0.003    0.998
modmat3 2.407e+01  9.789e+03    0.002    0.998
modmat4 2.176e+01  9.789e+03    0.002    0.998
modmat5 2.233e+01  9.789e+03    0.002    0.998
modmat6 2.176e+01  9.789e+03    0.002    0.998
modmat7 1.279e-16  1.414e+00 9.04e-17    1.000
modmat8        NA         NA       NA       NA

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 60.997  on 28  degrees of freedom
Residual deviance:  3.391  on 21  degrees of freedom
AIC: 25.709

Number of Fisher Scoring iterations: 21
```

The function `glm` does emit a warning (not shown because `Sweave` does not capture it) suggesting the MLE does not exist in the conventional sense. If we ignore the warning and apply the summary function, we get incomprehensible nonsense.

# 5 Linearity

## 5.1 Vector Form

Now we determine the linearity. See Geyer (2009, Section 3.12) for details.

```
> tanv <- modmat
> tanv[losses == 0, ] <- (-tanv[losses == 0])
> vrep <- cbind(0, 0, tanv)
> linear <- wins > 0 & losses > 0
> vrep[linear, 1] <- 1
> lout <- linearity(vrep, rep = "V")
> linear[lout] <- TRUE
> sum(linear == TRUE)

[1] 11

> sum(linear == FALSE)

[1] 17
```

The limiting conditional model (LCM), for which see Geyer (2009, Section 3.14), conditions on 17 components of the response, leaving 11 components free.

### 5.2 Matrix Form

It is hard to see what we are conditioning on when the data are strung out in a vector. Back to matrix form.

```
> foo <- matrix(FALSE, nrow(data), ncol(data))
> dimnames(foo) <- dimnames(data)
> foo[upper.tri(foo)] <- linear
> foo <- foo | t(foo)
> diag(foo) <- NA
> print(foo)

         ants beetles  cows  dogs egrets foxes gerbils  hogs
ants       NA   FALSE FALSE FALSE  FALSE FALSE   FALSE FALSE
beetles FALSE      NA  TRUE  TRUE   TRUE  TRUE   FALSE FALSE
cows    FALSE    TRUE    NA  TRUE   TRUE  TRUE   FALSE FALSE
dogs    FALSE    TRUE  TRUE    NA   TRUE  TRUE   FALSE FALSE
egrets  FALSE    TRUE  TRUE  TRUE     NA  TRUE   FALSE FALSE
foxes   FALSE    TRUE  TRUE  TRUE   TRUE    NA   FALSE FALSE
gerbils FALSE   FALSE FALSE FALSE  FALSE FALSE      NA  TRUE
hogs    FALSE   FALSE FALSE FALSE  FALSE FALSE    TRUE    NA
```

The TRUE entries are the ones that are free (we do not condition on the observed data) in the LCM. The FALSE entries are the ones that are fixed (we do condition on the observed data) in the LCM. The MLE says the ants always win, because they won all their games and we condition on this observed result. The MLE says the gerbils and the hogs always lose when playing anyone but each other, because they lost all their games, except to each other, and we condition on this result.

## 6 Generic Direction of Recession

Find the generic direction of recession (GDOR), for which see Geyer (2009, Section 3.6).

```
> p <- ncol(tanv)
> hrep <- cbind(0, 0, -tanv, 0)
> hrep[!linear, ncol(hrep)] <- (-1)
> hrep[linear, 1] <- 1
> hrep <- rbind(hrep, c(0, 1, rep(0, p), -1))
> objv <- c(rep(0, p), 1)
> pout <- lpcdd(hrep, objv, minimize = FALSE)
> gdor <- pout$primal.solution[1:p]
> names(gdor) <- team.names
> print(gdor)
```

```
   ants beetles    cows    dogs  egrets   foxes gerbils    hogs 
      2       1       1       1       1       1       0       0 
```

This agrees with our previous analysis. The natural parameter for a game between team $i$ and team $j$ has the form $\theta_i - \theta_j$. If $\delta$ is the GDOR, then we obtain the LCM by taking limits as $s \to \infty$ for linear predictor (matrix form) with components

$$\eta_{ij} = \theta_i - \theta_j + s(\delta_i - \delta_j) \tag{1}$$

when $\delta_i = \delta_j$, the limit changes nothing. When $\delta_i \neq \delta_j$, the limit makes the mean value parameter equal to the observed value for the corresponding natural statistic.

We see there are three groups having equal components of the GDOR. The first group is the singleton set $\{\text{ants}\}$. The second group is $\{\text{beetles}, \text{cows}, \text{dogs}, \text{foxes}, \text{egrets}\}$. The third group is $\{\text{gerbils}, \text{hogs}\}$.

We can see that the ants always win, according to the LCM, for either of two reasons. The first reason is that we are conditioning on the observed data for inter-group games and they always won in the observed data. The second reason is that their component of the GDOR is higher than anyone else's so (1) goes to $+\infty$ as $s \to \infty$ when $i = 1$ (ants being the team with index 1).

We can see that teams in group two (beetles, cows, dogs, foxes, egrets) always win when playing teams in group three (gerbils, hogs), according to the LCM, for either of two reasons. The first reason is that we are conditioning on the observed data for inter-group games and they always won in the observed data. The second reason is that their component of the GDOR for teams in group two is higher than that for teams in group three, so (1) goes to $+\infty$ as $s \to \infty$ when $2 \leq i \leq 6$ and $7 \leq j \leq 8$ (those being the ranges that have team $i$ in group two and team $j$ in group three).

## 7 Limiting Conditional Model

Fit the LCM

```
> resp.cond <- resp[linear, ]
> modmat.cond <- modmat[linear, ]
> out.cond <- glm(resp.cond ~ modmat.cond + 0, family = binomial)
> summary(out.cond)

Call:
glm(formula = resp.cond ~ modmat.cond + 0, family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1692  -0.1970   0.3941   0.5038   0.6153

Coefficients: (3 not defined because of singularities)
               Estimate Std. Error   z value Pr(>|z|)
modmat.cond1         NA         NA        NA       NA
modmat.cond2  3.024e+00  1.487e+00     2.034   0.0419 *
modmat.cond3  2.310e+00  1.328e+00     1.740   0.0819 .
modmat.cond4 -5.189e-17  1.080e+00 -4.81e-17   1.0000
modmat.cond5  5.609e-01  1.078e+00     0.520   0.6029
modmat.cond6         NA         NA        NA       NA
modmat.cond7  0.000e+00  1.414e+00     0.000   1.0000
modmat.cond8         NA         NA        NA       NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 13.863  on 11  degrees of freedom
Residual deviance:  3.391  on  6  degrees of freedom
AIC: 21.709

Number of Fisher Scoring iterations: 5
```

And extract the MLE

```
> beta.hat <- coefficients(out.cond)
> beta.hat[is.na(beta.hat)] <- 0
> beta.hat <- round(beta.hat, 15)
> names(beta.hat) <- team.names
> cbind(beta.hat, gdor)

         beta.hat gdor
ants    0.0000000    2
beetles 3.0237486    1
cows    2.3099766    1
dogs    0.0000000    1
egrets  0.5609163    1
```

Table 2: Maximum Likelihood Estimate. $\hat{\beta}$ the MLE for the LCM. $\delta$, the GDOR.

| | $\hat{\beta}$ | $\delta$ |
|---|---|---|
| ants | 0.000 | 2 |
| beetles | 3.024 | 1 |
| cows | 2.310 | 1 |
| dogs | 0.000 | 1 |
| egrets | 0.561 | 1 |
| foxes | 0.000 | 1 |
| gerbils | 0.000 | 0 |
| hogs | 0.000 | 0 |

```
foxes   0.0000000     1
gerbils 0.0000000     0
hogs    0.0000000     0
```

The MLE in the original model can be thought of as `beta.hat` sent to infinity in the direction `gdor`.

We make it a nice LaTeX table (Table 2, page 7).

```
> foo <- cbind(round(beta.hat, 3), gdor)
> colnames(foo) <- c("$\\hat{\\beta}$", "$\\delta$")
> library(xtable)
> print(xtable(foo, align = "l|cc", caption = paste("Maximum Likelihood",
+     "Estimate.", "$\\hat{\\beta}$ the MLE for the LCM.  $\\delta$, the GDOR."),
+     digits = c(3, 3, 0), label = "tab:mle"), caption.placement = "top",
+     table.placement = "tbp", include.rownames = TRUE,
+     sanitize.colnames.function = function(x) return(x))
```

# 8 Constancy Spaces

## 8.1 Original Model

```
> hrep <- cbind(1, 0, tanv)
> rout <- redundant(hrep, rep = "H")
> sout <- scdd(rout$output)
> const.orig <- sout$output[, -c(1, 2)]
> print(const.orig)
```

```
[1] 1 1 1 1 1 1 1 1
```

The vector having all components equal to one is a direction of constancy for the original model.

### 8.2 Limiting Conditional Model

```
> rout <- redundant(hrep[linear, ], rep = "H")
> sout <- scdd(rout$output)
> const.lcm <- sout$output[, -c(1, 2)]
> print(const.lcm)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    0    0    0    0    0    0    0
[2,]    0    1    1    1    1    1    0    0
[3,]    0    0    0    0    0    0    1    1
```

Each of the indicator vectors for the three groups is a direction of constancy for the LCM.

## 9 One-Sided Confidence Intervals

We try out the scheme discussed in Geyer (2009, Section 3.16.2 and 3.16.4). In calculating confidence intervals we need to take a union over a subspace (Geyer, 2009, Equation 23). This subspace may be taken to be the subspace of the constancy space of the LCM that is orthogonal to both the constancy space of the original model and to the GDOR.

```
> hrep.foo <- rout$output
> hrep.foo <- rbind(hrep.foo, c(1, 0, gdor))
> hrep.foo <- rbind(hrep.foo, c(1, 0, const.orig))
> sout.foo <- scdd(hrep.foo)
> qux <- sout.foo$output[, -c(1, 2)]
> qux
```

```
[1]  2.0 -0.8 -0.8 -0.8 -0.8 -0.8  1.0  1.0
```

To find $100(1-\alpha)\%$ confidence intervals, for each real number $r$ we are supposed to find the number $s$ such that

$$\Pr_{\hat{\beta}+r\gamma+s\delta}(H) = \alpha,$$

where $H$ is the support of the LCM, $\hat{\beta}$ is the vector denoted `beta.hat` above, $\gamma$ is the vector denoted `qux` above, and $\delta$ is the vector denoted `gdor` above.

```
> alpha <- 0.05
> rr <- seq(-3, 3, 0.5)
> sr <- double(length(rr))
> prob.lcm <- function(s, r) {
+     beta <- beta.hat + r * qux + s * gdor
+     eta <- modmat %*% beta
+     pp <- 1/(1 + exp(-eta))
+     qq <- 1/(1 + exp(eta))
+     probs <- pp^wins * qq^losses
+     return(prod(probs[!linear]))
+ }
> for (i in 1:length(rr)) {
+     foo <- function(s) prob.lcm(s, rr[i]) - alpha
+     uout <- uniroot(foo, lower = -100, upper = 100)
+     sr[i] <- uout$root
+ }
> print(sr)
```

```
 [1] 11.209266  9.809250  8.409277  7.009544  5.612184  4.238009
 [7]  3.051209  2.591890  3.046372  3.878236  4.770918  5.670211
[13]  6.570146
```

Now we need to say what this means in terms of mean value parameters.

```
> mu.win.bnd <- rep(Inf, length(wins))
> mu.loss.bnd <- rep(-Inf, length(losses))
> for (i in 1:length(rr)) {
+     beta <- beta.hat + rr[i] * qux + sr[i] * gdor
+     eta <- modmat %*% beta
+     pp <- 1/(1 + exp(-eta))
+     qq <- 1/(1 + exp(eta))
+     mu.win <- (wins + losses) * pp
+     mu.loss <- (wins + losses) * qq
+     mu.win.bnd <- pmin(mu.win.bnd, mu.win)
+     mu.loss.bnd <- pmax(mu.loss.bnd, mu.loss)
+ }
> mu.win.bnd[linear] <- NA
> mu.loss.bnd[linear] <- NA
> mu.win.bnd
```

```
 [1] 0.8931602 1.2445773        NA 1.8863473        NA        NA
 [7] 1.8090135        NA        NA        NA 1.8863473        NA
[13]        NA        NA        NA 1.9932217 1.9702737 1.9402337
[19] 1.5263428 1.6990965 1.5263428 1.9932217 1.9702737 1.9402337
[25] 1.5263428 1.6990965 1.5263428        NA
```

```
> mu.loss.bnd
```

```
 [1] 1.106839771 0.755422680          NA 0.113652690          NA
 [6]          NA 0.190986516          NA          NA          NA
[11] 0.113652690          NA          NA          NA          NA
[16] 0.006778302 0.029726330 0.059766254 0.473657195 0.300903512
[21] 0.473657195 0.006778302 0.029726330 0.059766254 0.473657195
[26] 0.300903512 0.473657195          NA
```

## 9.1 Matrix Form

We put these back in matrix form.

```
> foo <- matrix(0, nrow(data), ncol(data))
> dimnames(foo) <- dimnames(data)
> bar <- foo
> foo[upper.tri(foo)] <- mu.win.bnd
> bar[upper.tri(foo)] <- mu.loss.bnd
> foo <- foo + t(bar)
> diag(foo) <- NA
> foo <- round(foo, 3)
> print(foo)
```

```
         ants beetles  cows  dogs egrets foxes gerbils  hogs
ants       NA   0.893 1.245 1.886  1.809 1.886   1.993 1.993
beetles 1.107      NA    NA    NA     NA    NA   1.970 1.970
cows    0.755      NA    NA    NA     NA    NA   1.940 1.940
dogs    0.114      NA    NA    NA     NA    NA   1.526 1.526
egrets  0.191      NA    NA    NA     NA    NA   1.699 1.699
foxes   0.114      NA    NA    NA     NA    NA   1.526 1.526
gerbils 0.007   0.030 0.060 0.474  0.301 0.474      NA    NA
hogs    0.007   0.030 0.060 0.474  0.301 0.474      NA    NA
```

And we make it a nice LaTeX table (Table 3, page 11).

```
> library(xtable)
> print(xtable(foo, align = "l|cccccccc", caption = paste("One-sided 95\\%",
+     "Confidence intervals for Mean Values.", "Entries give expected number",
+     "of games (out of 2) that the team which is the row label beats the",
+     "team which is the column label.", "Numbers in the upper triangle",
+     "are lower bounds.", "Numbers in the lower triangle are upper bounds."),
+     digits = 3, label = "tab:one"), caption.placement = "top",
+     table.placement = "tbp", include.rownames = TRUE)
```

A little bit of experimentation (not shown) proves that the numbers in Table 3 are unchanged (to the three decimal precision used for this table) if the definition of rr is changed to

```
rr <- seq(-10, 10, 0.1)
```

and the numbers are only changed by at most one in the third decimal place if the definition of rr is changed to

```
rr <- seq(-2, 2, 0.5)
```

So it appears that we are getting sensible confidence intervals, even though we have no proof that the discretization used is o. k.

Table 3: One-sided 95% Confidence intervals for Mean Values. Entries give expected number of games (out of 2) that the team which is the row label beats the team which is the column label. Numbers in the upper triangle are lower bounds. Numbers in the lower triangle are upper bounds.

| | ants | beetles | cows | dogs | egrets | foxes | gerbils | hogs |
|---|---|---|---|---|---|---|---|---|
| ants | | 0.893 | 1.245 | 1.886 | 1.809 | 1.886 | 1.993 | 1.993 |
| beetles | 1.107 | | | | | | 1.970 | 1.970 |
| cows | 0.755 | | | | | | 1.940 | 1.940 |
| dogs | 0.114 | | | | | | 1.526 | 1.526 |
| egrets | 0.191 | | | | | | 1.699 | 1.699 |
| foxes | 0.114 | | | | | | 1.526 | 1.526 |
| gerbils | 0.007 | 0.030 | 0.060 | 0.474 | 0.301 | 0.474 | | |
| hogs | 0.007 | 0.030 | 0.060 | 0.474 | 0.301 | 0.474 | | |

# References

Fukuda, K. (2008) cddlib package, version 094f. `http://www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html`. Documentation is the file `cddlibman.pdf`, which is included in the `cddlib` source and also in every `rcdd` installation.

Geyer, C. J. (2009). Likelihood inference in exponential families and directions of recession Submitted to *Electronic Journal of Statistics*. `http://www.stat.umn.edu/geyer/gdor/`.

Geyer, C. J. and Meeden, G. D. (2008). R package `rcdd` (C Double Description for R), version 1.1. `http://http://www.stat.umn.edu/geyer/rcdd/`. Incorporates code from Fukuda (2008).

R Development Core Team (2008). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. `http://www.R-project.org`.