

S-PLUS/R Code to Perform the Analyses in
the Book *Nonprofit Trusteeship in Different
Contexts*

© Rikki Abzug and Jeffrey S. Simonoff

July 26, 2004

Introduction

This document is designed to demonstrate the fitting of the statistical models discussed in the book *Nonprofit Trusteeship in Different Contexts* by Rikki Abzug and Jeffrey S. Simonoff. All of the models involved are regression models falling into the class of *generalized linear models*, involving categorical response variables.

Many statistical packages include the ability to fit generalized linear models (and, in particular, the ones used in the book). Examples of such packages include LIMDEP, SAS, SPSS, and Stata. The analyses in the book were performed using the freeware package R (www.r-project.org), which is very similar to the commercial package S-PLUS (www.insightful.com). All of the code given here can be used in either R or S-PLUS.

Obviously, it is not possible to cover all of the details of fitting statistical models in general, or doing so with any specific packages in particular, in a document of this type. The book *Analyzing Categorical Data* by Jeffrey S. Simonoff (2003, Springer-Verlag) discusses all of the models used in *Nonprofit Trusteeship in Different Contexts* in detail. The web site for that book (www.stern.nyu.edu/~jsimonof/AnalCatData) gives more detailed information on packages that can be used to fit categorical data models. In particular, more details on the functions and libraries in S-PLUS and R that are used to fit the models described here can be found there.

The data analyzed in the book have been deposited at the Cultural Policy and the Arts National Data Archive web site (www.cpanda.org). Since different individuals will organize their data in different ways, and since the format of the available data at the Archive might change over time, we will not discuss specifics here of how to download, input, and organize data from the Six Cities study.

Data analysis using S-PLUS/R

Datasets in S-PLUS/R take the form of data frames. A typical analysis, such as fitting a regression model, involves applying a function (for many generalized linear models the `glm` function) to (a subset of) the data frame. Typically the results of the function are saved to a new object, which then can be used as input to further analysis (including scatter plots, diagnostics, summary statistics, hypothesis tests, confidence intervals, and so on).

Analyses in *Nonprofit Trusteeship in Different Contexts* were organized by type of response variable. So, for example, the analysis of the probability that a trustee is black on pages 83–86 is based on a data frame constructed at the organization level (each row representing a particular nonprofit organization’s board of trustees), with each variable being a characteristic of the organization (for example, the year, city, industry group, religion-based status, board size, number of white trustees, number of black trustees, number of Latin trustees, number of Asian trustees, and total number of trustees). Analyses related to gender, for example, were based on a similar data frame, only now the number of male, female, and total trustees were included. Analyses on variables such as the number of boards on which a trustee sits are done at the individual trustee level, so the related data frame takes as each row a different trustee, rather than a different board.

Logistic regression

The analysis relating to the race of the trustee is typical of the logistic regression analyses in the book. The data are in the data frame `Race.data`. The function that fits a logistic regression is the `glm` function. The function defines the variables in the model using a formula object

$$y \sim x1 + x2$$

where the response variable y is on the left side and the predicting variables $x1$ and $x2$ are on the right side. In the case of logistic regression there are several different ways of defining a binomial response variable; the one used here gives as the response a matrix with two columns, the number of “successes” (here the number of black trustees on the board) and the number of “failures” (here the number of nonblack trustees). The line above specifies a model with two additive numerical predictors. A categorical predictor is specified as `factor(x1)`. An interaction between two predictors is specified as `factor(x1)*factor(x2)`. The `glm` function is used to fit many different types of generalized linear models, so the underlying distribution family (in this case binomial) must be specified. So, the call to fit the first listed model in Table 8.1 (page 84) is as follows.

```
#
# This is how a comment is added
#
glm(cbind(black, totalrace - black) ~ factor(year), family=binomial,
    data=Race.data)
```

The code `cbind(black, totalrace - black)` forms the response matrix from the underlying variables `black` (the number of black trustees on the board) and `totalrace` (the total number of trustees on the board). The fragment `factor(year)` specifies that the year variable should be treated as a categorical predictor, rather than a numerical one.

The information-theoretic approach to modeling used in the book requires calculation of AIC_C for each model fit. The AIC value can be extracted from a `glm` object using the `extractAIC` function that is available from the `MASS` library, which is included in the latest version of `S-PLUS` and in the binary distribution of `R` (it can be added and updated to any version of `R`). To use this function, the command

```
library(MASS)
```

is first entered (this only needs to be entered once in each session, before the first call to `extractAIC`). Then, the first line in Table 8.1 can be obtained as follows:

```
gg <- glm(cbind(black, totalrace - black) ~ factor(year),
    family=binomial, data=Race.data)
cbind(deviance(gg), gg$df.residual,
    extractAIC(gg, k=2*sum(gg$prior.weights)/
    (sum(gg$prior.weights) - length(coef(gg)) - 1))[2])
```

The first command line fits the logistic regression, and assigns it to the object `gg`. The code `deviance(gg)` in the second command line extracts the deviance G^2 for the model, and the code `gg$df.residual` produces the degrees of freedom of the model. The `extractAIC` function extracts AIC for the object; the additional code converts this to AIC_C . A particular characteristic or variable of an object or data frame (such as the `df.residual` characteristic of the `glm` object `gg`) is generally accessed using the `$` form given in the code. The AIC_C value that this code gives is not the same as that in Table 8.1, since (as is discussed on page 84) in that table the values

are restandardized so that the best-fitting model has $AIC_C = 0$ (this is done by subtracting the minimal AIC_C value from all of the model values, as seen below). In fact, different packages can give different values of AIC (and hence AIC_C) for a given model (since it depends on constants that might not be the same over the different packages), but differences between AIC (and AIC_C) values for different models will be the same (other than small differences because of different glm convergence criteria used in the different packages).

Putting this all together, the following code will produce Table 8.1.

```
#
# The results matrix stores the deviance, degrees of freedom, and AICc values
#
results <- matrix(NA,10,3)
#
# Fitting the models
#
gg <- glm(cbind(black, totalrace - black) ~ factor(year),
  family=binomial, data=Race.data)
results[1,] <- cbind(deviance(gg), gg$df.residual,
  extractAIC(gg, k=2*sum(gg$prior.weights)/
  (sum(gg$prior.weights) - length(coef(gg)) - 1))[2])
gg <- glm(cbind(black, totalrace - black) ~ factor(year) + factor(industry),
  family=binomial, data=Race.data)
results[2,] <- cbind(deviance(gg), gg$df.residual,
  extractAIC(gg, k=2*sum(gg$prior.weights)/
  (sum(gg$prior.weights) - length(coef(gg)) - 1))[2])
gg <- glm(cbind(black, totalrace - black) ~ factor(year) + factor(industry)
  + factor(religious), family=binomial, data=Race.data)
results[3,] <- cbind(deviance(gg), gg$df.residual,
  extractAIC(gg, k=2*sum(gg$prior.weights)/
  (sum(gg$prior.weights) - length(coef(gg)) - 1))[2])
gg <- glm(cbind(black, totalrace - black) ~ factor(city) + factor(year)
  + factor(industry) + factor(religious), family=binomial, data=Race.data)
results[4,] <- cbind(deviance(gg), gg$df.residual,
  extractAIC(gg, k=2*sum(gg$prior.weights)/
  (sum(gg$prior.weights) - length(coef(gg)) - 1))[2])
gg <- glm(cbind(black, totalrace - black) ~ factor(city) + factor(year)
  + factor(industry) + factor(religious) + boardsize, family=binomial,
  data=Race.data)
```

```

results[5,] <- cbind(deviance(gg), gg$df.residual,
  extractAIC(gg, k=2*sum(gg$prior.weights)/
    (sum(gg$prior.weights) - length(coef(gg)) - 1))[2])
gg <- glm(cbind(black, totalrace - black) ~ factor(city)*factor(industry)
  + factor(year) + factor(religious) + boardsize, family=binomial,
  data=Race.data)
results[6,] <- cbind(deviance(gg), gg$df.residual,
  extractAIC(gg, k=2*sum(gg$prior.weights)/
    (sum(gg$prior.weights) - length(coef(gg)) - 1))[2])
gg <- glm(cbind(black, totalrace - black) ~ factor(city)*factor(industry)
  + factor(year) + factor(religious), family=binomial, data=Race.data)
results[7,] <- cbind(deviance(gg), gg$df.residual,
  extractAIC(gg, k=2*sum(gg$prior.weights)/
    (sum(gg$prior.weights) - length(coef(gg)) - 1))[2])
gg <- glm(cbind(black, totalrace - black) ~ factor(city)*factor(religious)
  + factor(year) + factor(industry) + boardsize, family=binomial,
  data=Race.data)
results[8,] <- cbind(deviance(gg), gg$df.residual,
  extractAIC(gg, k=2*sum(gg$prior.weights)/
    (sum(gg$prior.weights) - length(coef(gg)) - 1))[2])
gg <- glm(cbind(black, totalrace - black) ~ factor(city)*factor(religious)
  + factor(year) + factor(industry), family=binomial, data=Race.data)
results[9,] <- cbind(deviance(gg), gg$df.residual,
  extractAIC(gg, k=2*sum(gg$prior.weights)/
    (sum(gg$prior.weights) - length(coef(gg)) - 1))[2])
gg <- glm(cbind(black, totalrace - black) ~ factor(city)*factor(religious)
  + factor(city)*factor(industry) + factor(year), family=binomial,
  data=Race.data)
results[10,] <- cbind(deviance(gg), gg$df.residual,
  extractAIC(gg, k=2*sum(gg$prior.weights)/
    (sum(gg$prior.weights) - length(coef(gg)) - 1))[2])
#
# Restandardizing AICc to make minimal value 0
#
results[,3] <- results[,3] - min(results[,3])
results

```

The final model chosen in this case was based on the city \times religion-based status interaction, and the year and industry main effects. In order to examine this model further, we refit it, and assign it to an object. The

`options` statement is used to make sure that effect codings (rather than indicator variables) are being used to represent the categorical effects (such as the year effect). This only needs to be entered once in a session.

```
options(contrasts=c("contr.sum","contr.poly"))
gg <- glm(cbind(black, totalrace - black) ~ factor(city)*factor(religious)
  + factor(year) + factor(industry), family=binomial, data=Race.data)
```

The coefficients of the model are obtained using the `coef` function. Marginal proportions (to summarize the year and industry effects) are obtained by “applying” the `sum` function across the appropriate margins.

```
coef(gg)
#
# sapply function "applies" a function to margins of a list, matrix,
# or vector
#
sapply(split(Race.data$black,Race.data$year),sum)/
  sapply(split(Race.data$totalrace,Race.data$year),sum)
sapply(split(Race.data$black,Race.data$industry),sum)/
  sapply(split(Race.data$totalrace,Race.data$industry),sum)
```

Producing the table on page 86 (and the many similar tables in Chapter 9) requires estimating probabilities of a trustee being black for all 12 pairs of the (city, religion-based) combination, taking the other two predictors (year and industry) as “typical” values. In this case, we have set year to 1991 and industry to family services (which corresponds to `industry = 7`). First, a new data frame is created that contains all of the variables that are used in the model, and 12 rows, each corresponding to a (city, religion-based) combination and the specified values of year and industry. The `rep` function allows the combinations to be formed automatically. Since year, city, industry, and religious-based status are categorical (factor) effects, they must be defined that way in the new data frame, with levels being the same as those in the corresponding variables in the original data frame. The `predict` command then produces estimated probabilities for each of the possibilities, which are then printed out.

```

pred.df <- data.frame(year=factor(rep("91",12), levels=c("31","61","91")),
  city=factor(rep(c("1","2","3","4","5","6"),each=2),
  levels=c("1","2","3","4","5","6")), industry=factor(rep("7",12),
  levels=c("1","2","3","4","5","6","7","8")),
  religious=factor(rep(c("0","1"),6), levels=c("0","1")))
#
# The "response" type makes the predictions in the probability,
# rather than logit, scale
#
pred.prob <- predict(gg,pred.df, type="response")
cbind(pred.df, pred.prob)

```

The logistic regression estimating the probability that a trustee is black did not have overdispersion problems, but many of the models described in Chapter 9 did. This does not affect the estimated coefficients or estimated probabilities, but does affect model selection, since now it is based on $QAIC_C$ (page 80), rather than AIC_C . The `extractAIC` function cannot be used to get $QAIC_C$, so it is calculated directly. First, the best fitting (according to AIC_C) model is determined. The inflation factor $\hat{\phi}$ is calculated based on that model, which is then used to deflate the deviance in determining $QAIC_C$ for each of the models. The following code illustrates this for the gender analysis.

```

#
# The "best" model based on AICc was (CY, CI, CR, CB, YI, YR, IB, RB)
#
gg <- glm(cbind(female, totalgender - female) ~ factor(city)*factor(year)
  + factor(city)*factor(industry) + factor(city)*factor(religious)
  + factor(city)*boardsize + factor(year)*factor(industry)
  + factor(year)*factor(religious) + factor(industry)*boardsize
  + factor(religious)*boardsize, family=binomial, data=Gender.data)
phihat <- sum(residuals(gg,type="pearson")^2)/gg$df.residual
#
# Now, try models again, evaluating using QAICc
#
gg <- glm(cbind(female, totalgender - female) ~ factor(city), family=binomial,
  data=Gender.data)
results[1,] <- cbind(deviance(gg), gg$df.residual, deviance(gg)/phihat +
  2*length(coef(gg))*sum(gg$prior.weights)/(sum(gg$prior.weights)-
  length(coef(gg))-1))

```

.
.
.

Multinomial regression

Logistic regression (where there are two categories in the response variable) generalizes to multinomial regression using the `multinom` function, which is part of the `nnet` library. This method was used in the analysis of higher education achievement, since there were three education response levels. The response matrix generalizes the form used previously for logistic regression, in that it consists of k columns, representing the counts for each of the k levels of the response (here $k = 3$).

```
library(nnet)
#
# Columns 8 through 10 contain the counts of the number of trustees with some
# college, masters degree, and professional degree or PhD, respectively
#
edlevel <- as.matrix(Hied.data[,8:10])
gg <- multinom(edlevel ~ factor(city), data=Hied.data)
results[1,] <- cbind(deviance(gg), gg$df.residual,
  extractAIC(gg,k=2*sum(gg$prior.weights)/
    (sum(gg$prior.weights)-length(coef(gg))-1))[2])
```

Poisson regression

The analyses of board interlocks (pages 100–107) and board size (pages 108–110) are based on Poisson regression. Since this is also a generalized linear model, the structure is very similar to that of logistic regression, with only the distribution family changing. As noted earlier, for these analyses the underlying data frame has a different row for each trustee.

```
gg <- glm(OtherBoards ~ factor(city), family=poisson, data=Board.data)
results[1,] <- cbind(deviance(gg), gg$df.residual,
  extractAIC(gg,k=2*sum(gg$y)/(sum(gg$y)-length(coef(gg))-1))[2])
.
.
.
```