Splus Manual to Accompany Agresti’s
*Categorical Data Analysis* (1990)

Laura A. Thompson, 1999©
# Table of Contents

Introduction .................................................................................................................. 1
  A. Libraries in Splus ................................................................................................. 1
  B. Setting contrast types using Options() ............................................................... 2
  C. Credit for functions ........................................................................................... 2
  D. A note about using Splus Menus ......................................................................... 2
  E. Notice of errors .................................................................................................... 2
  F. References ........................................................................................................... 2

Chapter 2 - Agresti ...................................................................................................... 4
  A. Comparing proportion parameters for two samples, and associated sample statistics .................................................................................................................. 4
  B. Creating Cross-Classified Tables ........................................................................ 5
  C. Measures of Association - Gamma ........................................................................ 6

Chapter 3 - Agresti ...................................................................................................... 7
  A. Chi-square Tests .................................................................................................. 7
  B. Fisher Exact Test .................................................................................................. 8

Chapter 4 - Agresti ...................................................................................................... 10
  A. Generalized Linear Models .................................................................................. 10
  B. Logit models for categorical data .......................................................................... 13
    1. Linear logit model ............................................................................................... 13
    2. Model Diagnostics - Residuals .......................................................................... 16
  C. Overdispersed Poisson Model (Negative Binomial) .......................................... 17
  D. Extended Example (problem 4.7) ........................................................................ 18

Chapter 5 - Agresti ...................................................................................................... 23
  A. Demonstration of Simpson’s Paradox ................................................................. 23

Chapter 6 - Agresti ...................................................................................................... 25
  A. Fitting Loglinear Models ...................................................................................... 25
  B. Partial and Marginal Odds Ratios ........................................................................ 26
  C. Comparing nested models using LR tests ........................................................... 27
  D. Loglinear Models as Logit Models ....................................................................... 27
  E. Asymptotic Standard Errors and Wald tests ....................................................... 28
  F. Probability Estimates ........................................................................................... 29
  G. Analyzing survival rates using loglinear models ............................................... 29
  H. Modeling survival times using loglinear models ............................................... 31
    1. Illustration of using function loglin .................................................................... 32

Chapter 7 - Agresti ...................................................................................................... 35
  A. Stepwise Model Selection .................................................................................... 35
  B. Adjusting a model using indicator functions ....................................................... 36
  C. Cochran-Mantel-Haenszel Test ........................................................................... 38
## D. Sample size and power calculations

Chapter 8 - Agresti

- **A. Uniform Association model**
- **B. Row Effects model - (same effect for an entire row)**
- **C. Modeling three-factor interaction**
- **D. Uniform interaction model**
- **E. Mantel Score Test of Conditional Independence**
- **F. Correspondence Analysis**

Chapter 9 - Agresti

- **A. Generalized logit model for multiple nominal response categories**
  - **B. Multinomial Logits for Ordinal Responses (Continuation-Ratio logit model)**
    - **1. Continuation-Ratio logit model**
      - a. Odds Ratios
      - b. Using glm
    - **C. Cumulative Logit Models**
      - **1. Proportional Odds Model**
      - **2. Cumulative logit uniform association model**
      - **3. Cumulative logit row effects model**
    - **D. Cumulative Link Models**
      - **1. Proportional Hazards Models**
    - **E. Regression Model for Ordered Responses**
    - **F. (Supplement) Nonlinear Ordinal Models (Using libraries nolr and lrm)**
      - **1. Using nolr**
      - **2. Using lrm**

Chapter 10 - Agresti

- **A. Comparing dependent proportions**
- **B. Conditional Logistic Regression (CLR)**
- **C. Symmetry Models**
  - **1. Ordinary Symmetry Model**
  - **2. Quasi-Symmetry Model**
  - **3. Quasi-independence Model**
- **D. Marginal Homogeneity Models**
- **E. Square Tables with Ordered Categories**
  - **1. Symmetry model**
  - **2. Conditional Symmetry Model**
- **F. Measuring Agreement**
  - **1. Cohen’s Kappa**
  - **2. Uniform Association Plus Extra Agreement Model**
- **G. Bradley-Terry Model**
  - **1. Quasi-symmetry fit**
  - **2. Logit model fit**
- **H. Bradley-Terry Model-with Order Effect**
Chapter 11 - Agresti ................................................................. .................................. 78
A. Symmetry Models for Repeated Categorical Responses ............................................. 78
  1. Complete Symmetry ........................................................................................................ 78
  2. Quasi-symmetry Model ................................................................................................... 78
  3. No-three-factor interaction model ................................................................................ 79
B. Marginal Homogeneity ..................................................................................................... 79
C. Modeling a Repeated Categorical Response ................................................................. 81
  1. RMTOOLS ...................................................................................................................... 81
  2. YAGS ............................................................................................................................ 83
D. Modeling a Repeated Ordinal Response .......................................................................... 87
E. Markov Chain Models ...................................................................................................... 92
Introduction

This Splus manual accompanies Agresti’s (1990) *Categorical Data Analysis*. I have included examples of almost all of the major (and some minor) analyses introduced by Agresti. The manual chapters parallel those from Agresti so that, for example, in Chapter 2 I discuss using Splus to conduct analyses from Agresti’s Chapter 2. In most cases I use the data provided in the text. There are only one or two occasions where I use data from the problems sections in Agresti. With the exception of an extended example from Chapter 4, discussion of results of analyses is limited since the discussion appears in the text. As such, this manual is really not self-contained. It will be helpful (and many times essential) to use the text too to understand completely the analyses in this manual.

In the manual I frequently employ functions that come from user-contributed libraries of Splus. In the text, I note when a particular library is used. These libraries are not included with the Splus software, but can be easily obtained from the internet. I mention in the next section how to get these libraries and how to install them for use with Splus. Each library has its own help manual. I will demonstrate how to access these help files from inside Splus.

The version of Splus I used to do all analyses is 4.5 for Windows. However, Windows versions as far back as 3.0 will do many of the analyses. I do not rely on menus at all. I am also fairly certain that Splus 2000 for Windows will handle most, if not all, of the code in this manual.

To separate Splus input and output from other text in this manual, I have put normal text in Arial font and Splus commands and output in *courier* font. The input is in *bold* font, whereas the output is not.

Finally, this manual assumes some familiarity in using the basic commands of Splus. To keep the manual from being too long I do not discuss functions that I use which are not directly related to categorical data analysis. In most cases, typing out the command as it appears in the text can tell you exactly what it does.

A. Libraries in Splus

The libraries used in this manual that do not come with the software are

- MASS (B. Ripley) - (used throughout)
- Multiv (F. Murtagh) - (used for correspondence analysis)
- Design (F. Harrell) - (used for analyses in chapter 9)
- Hmisc (F. Harrell) - (support for Design library)
- nnet (B. Ripley) - for the function multinom (multinomial logit models)
- nolr (M. Mathieson) - (nonlinear ordinal models - supplement to chapter 9)
- rmtools (A. Azzalini & M. Chiogna) - (used for chapter 11)
- yags (V. Carey) - (used for chapter 11)

All libraries used can be obtained in .zip form from URL [http://lib.stat.cmu.edu/DOS/S/Swin](http://lib.stat.cmu.edu/DOS/S/Swin) or [http://lib.stat.cmu.edu/DOS/S](http://lib.stat.cmu.edu/DOS/S). To install a library, first download it to any folder on your computer. Next, “unzip” the file using an “unzipping” program. This will extract all the files into a new folder in the directory into which you downloaded the zip file. (I used unzip95, which is available from the URL above. Winzip should work also. I don’t know about pkunzip). Move the entire folder to the library directory under your Splus directory (e.g., c:\Program files\Splus45\library).

To load a library, you can either pull down the File menu in Splus and select Load Library or type one of the following in a script or command window

```splus
library("libraryname");first=T
library(libraryname)
```

To use the library’s help manual from inside Splus type in a script or command window
help(library="libraryname")

B. Setting contrast types using Options()

The options function can be used to set the type of contrasts used in estimating models. The default is Helmert contrasts for factors and polynomial contrasts for ordered factors. I use the default for most estimates, which makes them sometimes different from Agresti's. However, whenever estimates are emphasized, I note how to get his estimates from mine. Whenever I have not used the default setting, I have indicated this prior to obtaining the estimates.

One can find out exactly what the contrasts are in a glm type fit by using the functions model.matrix and contrasts. Issuing the command contrasts(model.matrix(fit)) gives the contrasts.

C. Credit for functions

The author of a function is named if the function was not written by me. Whenever I use functions that do not come with Splus, I will mention the Splus library that includes them. The authors for all libraries are mentioned above as well as at the URL address.

D. A note about using Splus Menus

Many of the more common methods I will illustrate can be accomplished via the Splus menus. There are at least two reasons not to use menus for most work (with the exception of graphing). First, using "code" helps one to learn the language. At times, one desires to modify an existing method. Modifications can't be done using the existing menus (albeit, one can write their own menus which do contain the modified method). Also, Splus (as well as any interpreted language) is one of the best vehicles to try out new methods without getting bogged down in compilation errors.

In addition, knowing the code helps in understanding what to put in the slots in the menus. If you want to know what code corresponds to a particular menu command, issue the menu command and call up the History window (using the Window menu). All commands you have issued from menus will be there in (MENU) code form which can be used in a command window or script.

E. Notice of errors

All code has been tested once, but I did not have time to check anything twice. So, please notify me of any errors in the manual or of easier ways to perform tasks. My email address is thompsonL@baylor.edu.

F. References


A. Comparing proportion parameters for two samples, and associated sample statistics

To set up Table 2.3 (heart attack v. no heart attack) on p. 17, in Agresti, type:

\[
\begin{align*}
\text{x} & \leftarrow \text{c}(104, 189) \quad \# \text{ aspirin, placebo} \\
\text{n} & \leftarrow \text{c}(11037, 11034)
\end{align*}
\]

Then, to test \(H_0: p_1 = p_2\) (equal probabilities of heart attack), one can use the \text{prop.test} function.

\[
\text{prop.test(x, n, p=(.5,.5))}
\]

2-sample test for equality of proportions with continuity correction

data:  x out of n
X-square = 24.4291, df = 1, p-value = 0
alternative hypothesis: two.sided
95 percent confidence interval:
-0.010814914 -0.004597134

sample estimates:
prop'n in Group 1 prop'n in Group 2
0.00942285 0.01712887

We can obtain the p-value by extracting it from \text{prop.test}:

\[
\text{prop.test(x, n)}$p.value
\]

[1] 7.709708e-007

A one-sided test of the hypotheses, \(H_0: p_1 = p_2\) v. \(H_1: p_1 < p_2\), can be obtained using the \text{alt} option:

\[
\text{prop.test(x, n, alt="less")}$p.value
\]

[1] 3.854854e-007

The sample difference of proportions is extracted as \text{estimate}:

\[
\text{temp<-prop.test(x, n)}
\]

\[
\text{names(temp$estimate)<-NULL} \quad \# \text{ optional}
\]

\[
\text{temp$estimate[[1]]-temp$estimate[[2]]}
\]

[1] -0.007706024

Other useful quantities are easily computed. Here, I calculate the relative risk and odds ratio:

Relative risk:

\[
\text{temp$estimate[[2]]/temp$estimate[[1]]}
\]

[1] 1.817802
### 5

#### Odds ratio:

\[
\hat{\text{OR}} = \frac{x[2] \times (n[1] - x[1])}{x[1] \times (n[2] - x[2])}
\]

[1] 1.832054

#### B. Creating Cross-Classified Tables

One can create a cross-classified table out of Table 2.4 (job satisfaction) p. 21, using the following commands:

```r
income <- c("<6000", "upto15000", "upto25000", ">25000")
jobsat <- c("very dissat", "little", "moderate", "very sat")
datalabel <- list(income, jobsat)
table.2.4 <- fac.design(c(4,4), factor.names=datalabel)  # sets up the combinations of the levels as a factorial design, using labels datalabel
data <- c(20, 22, 13, 7, 24, 38, 28, 18, 80, 104, 81, 54, 82, 125, 113, 92)
table.2.4 <- cbind(table.2.4, data)  # add the data
dimnames(table.2.4)[[2]] <- c("income", "jobsat", "count")  # change names of table.2.4
crosstabs(count ~ income + jobsat, table.2.4)
```

Call:
```
crosstabs(count ~ income + jobsat, table.2.4)
901 cases in table
```

#### 901 cases in table

<table>
<thead>
<tr>
<th>income</th>
<th>jobsat</th>
<th>N/RowTotal</th>
<th>N/ColTotal</th>
<th>N/Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;6000</td>
<td>vrydsst</td>
<td>0.0971</td>
<td>0.0222</td>
<td>0.0222</td>
</tr>
<tr>
<td></td>
<td>little</td>
<td>0.1165</td>
<td>0.2222</td>
<td>0.0266</td>
</tr>
<tr>
<td></td>
<td>moderat</td>
<td>0.3883</td>
<td>0.2508</td>
<td>0.0888</td>
</tr>
<tr>
<td></td>
<td>verysat</td>
<td>0.3981</td>
<td>0.1990</td>
<td>0.0910</td>
</tr>
<tr>
<td></td>
<td>RowTotal</td>
<td>0.23</td>
<td>0.26</td>
<td>0.19</td>
</tr>
</tbody>
</table>

| up15000  | vrydsst  | 0.0761     | 0.0222     | 0.0222  |
|          | little   | 0.1315     | 0.2222     | 0.0266  |
|          | moderat  | 0.3599     | 0.2508     | 0.0888  |
|          | verysat  | 0.4325     | 0.1990     | 0.0910  |
|          | RowTotal | 0.32       | 0.26       | 0.19    |

| up25000  | vrydsst  | 0.0553     | 0.0222     | 0.0222  |
|          | little   | 0.1191     | 0.2222     | 0.0266  |
|          | moderat  | 0.3447     | 0.2508     | 0.0888  |
|          | verysat  | 0.4809     | 0.1990     | 0.0910  |
|          | RowTotal | 0.26       | 0.26       | 0.19    |

| >25000   | vrydsst  | 0.0409     | 0.0222     | 0.0222  |
|          | little   | 0.1053     | 0.2222     | 0.0266  |
|          | moderat  | 0.3158     | 0.2508     | 0.0888  |
|          | verysat  | 0.5380     | 0.1990     | 0.0910  |
|          | RowTotal | 0.19       | 0.26       | 0.19    |

| ColTotl  | vrydsst  | 0.069      | 0.120      | 0.354   |
|          | little   | 0.108      | 0.319      | 0.457   |
|          | moderat  | 0.319      | 0.412      | 0.457   |
|          | verysat  | 0.412      | 0.412      | 0.457   |
|          | RowTotal | 0.457      | 0.457      | 0.457   |

Test for independence of all factors
```
\[ \chi^2 = 11.98857 \text{ d.f.} = 9 \quad (p=0.2139542) \]
```
```
Yates' correction not used
```

See the Splus manual for more uses of crosstabs.
C. Measures of Association - Gamma

Here is a function for computing Goodman and Kruskal’s gamma:

```r
Gamma.f<-function(x,verbose=F) {
  # x is a matrix of counts
  n<-nrow(x)
  m<-ncol(x)
  res<-numeric((n-1)*(m-1))
  for(i in 1:(n-1)) { for(j in 1:(m-1)) res[j+(m-1)*(i-1)]<-
    x[i,j]*sum(x[(i+1):n,(j+1):m]) }
  C<-sum(res)
  res<-numeric((n-1)*(m-1))
  iter<-0
  for(i in 1:(n-1)) for(j in 2:m) { iter<-iter+1; res[iter]<-
    x[i,j]*sum(x[(i+1):n,1:(j-1)]) }
  D<-sum(res)
  gamma<-(C-D)/(C+D)
  if(verbose) return( gamma=gamma, C=C, D=D)
  else return(gamma)
}
```

Using it on table 2.4, we get

```r
Gamma.f(matrix(table.2.4[,3],byrow=F,ncol=4) ,verbose=T)
```

$gamma:
[1] 0.1265461

$C:
[1] 109520

$D:
[1] 84915

Selvin (1998) computes the number of concordant and discordant pairs using the outer function along with ifelse statements. However, the procedure is very memory intensive and will not work with the default memory allocation (this is changed using options). The function above takes .05078125 CPU seconds. Selvin’s function takes over 30 seconds (on a system with 133 MH, 32 MB RAM and half the hard disk empty).

Other measures of association can be computed immediately from the chisquare value output from chisq.test (e.g., phi, Cramer’s V, Cramer’s C). See Selvin for more details.
Chapter 3 - Agresti

A. Chi-square Tests

This section uses the data in Table 3.2 (p.46-47).

First, I set up the data and the expected counts:

\[
\begin{align*}
\text{pneumonia.counts} & \leftarrow \text{c(30, 63, 63)} \\
\text{expected} & \leftarrow \text{c(38.1, 39, 78.9)}
\end{align*}
\]

We can compute Pearson's chi-square statistic and obtain the p-value using the function `.pearson.x2`:

\[
\begin{align*}
\text{x2value} & \leftarrow \text{unlist(.pearson.x2(pneumonia.counts, expected), use.names=F)} \\
\text{1-pchisq(x2value, df=1)} & \leftarrow \text{[1] 9.081684e-006}
\end{align*}
\]

The function `.pearson.x2` is not well-known. But, you can find it used in `chisq.test`. `chisq.test` does not allow one to set the expected frequencies. With the commands above, I "unlisted" the result of `.pearson.x2` since the result is a list and I wanted a numeric vector.

For calculating GOF tests to named parametric distributions, use `chisq.gof`. For example, taking the data from problem 3.3, we get:

\[
\begin{align*}
m & \leftarrow (200-109)/200 \quad \# \text{lambda} \\
data & \leftarrow \text{rep(c(0, 1, 2, 3, 4), c(109, 65, 22, 3, 1))} \\
\text{chisq.gof(data, cut.points=c(-1, 0, 1, 2, 3, 4), dist="pois", lambda=m)}
\end{align*}
\]

Warning messages:

```
Warning in chisq.gof(data, cut.points = c(-1, 0, 1, 2, : Expected counts < 5. Chi-squared approximation may not be appropriate.
```

Chi-square Goodness of Fit Test

```
data:  data
Chi-square = 12.5698, df = 4, p-value = 0.0136
alternative hypothesis:
  True cdf does not equal the poisson Distn. for at least one sample point.
```

To test independence of two factors, such as those in Table 3.5 (p. 52), I compute Pearson's chi-square statistic. A warning is apt here, as we have fewer than five expected counts for some cells, and the chi-squared approximation requires at least five. An exact test would be more appropriate (see the next section).

\[
\begin{align*}
schizo.mat & \leftarrow \text{matrix(c(90, 12, 78, 13, 1, 6, 19, 13, 50), byrow=T, ncol=3)} \\
\text{chisq.test(schizo.mat, correct=F)}
\end{align*}
\]

Warning messages:

```
Warning in chisq.test(schizo.mat, correct = F): Expected counts < 5. Chi-square approximation may not be appropriate.
```

Pearson's chi-square test without Yates' continuity correction

```
data:  schizo.mat
```

```
X-square = 22.3777, df = 4, p-value = 0.0002

To compute a likelihood ratio test (LRT) statistic, I modified `.pearson.x2` to obtain the expected frequencies.

\[
\text{expected} <- \text{my.pearson.x2(schizo.mat)} \times 2 \times \text{sum(schizo.mat}\times \log\left(\frac{\text{schizo.mat}}{\text{expected}}\right)\}
\]

\[\begin{align*}
\text{[1]} & \ 23.03619
\end{align*}\]

Specifically, I modified the third-to-last line in `.pearson.x2` to get the expected frequencies:

\[
\text{ret.val <- list(X2 = sum((abs(expected - observed) - (if(yates) 0.5 else 0))^2/ expected), expected = expected)}
\]

**B. Fisher Exact Test**

To demonstrate Fisher's Exact Test, Agresti uses the data from Table 3.7 (p.61).

The function `fisher.test` gives a two-sided test:

\[
\text{test<-fisher.test(matrix(c(3,1,1,3),byrow=T,ncol=2))}
\]

Fisher's exact test
data:  matrix(c(3, 1, 1, 3), byrow = T, ncol = 2) p-value = 0.4857 alternative hypothesis: two.sided

To get the one-sided p.value, type:

\[
\text{test$p.value/2}
\]

\[\begin{align*}
\text{[1]} & \ 0.2428572
\end{align*}\]

Note the one-sided p-value can be obtained using `dhyper`

\[
\text{sum(dhyper(q=c(3,4),m=4,n=4,k=4))}
\]

\[\begin{align*}
\text{[1]} & \ 0.2428571
\end{align*}\]

The next demonstration illustrates how to get the exact conditional test mentioned on p. 64.

**Table 3.8**

\[
\text{temp<-matrix(c(25,25,12,0,1,3),byrow=T,ncol=3)}
\]

\[
\begin{array}{ccc}
[1,] & 25 & 25 & 12 \\
[2,] & 0 & 1 & 3 \\
\end{array}
\]

\[
\text{Gamma.f(temp,T)}
\]

\[
\text{$gamma:}
\]

\[\begin{align*}
\text{[1]} & \ 0.8716578
\end{align*}\]

\[
\text{$C:}
\]

\[\begin{align*}
\text{[1]} & \ 175
\end{align*}\]

\[
\text{$D:}
\]

\[\begin{align*}
\text{[1]} & \ 12
\end{align*}\]
To get the p-value, I use the functions `factorial` and `prod`:

```r
num <- (prod(factorial(rep(1, 2) %*% temp)) * prod(factorial(rep(1, 3) %*% t(temp))))
den <- (factorial(sum(temp)) * prod(factorial(temp)))
term1 <- (num/den)

temp <- matrix(c(25, 26, 11, 0, 0, 4), byrow = T, ncol = 3)  # only other table with C-D as least as large

num <- (prod(factorial(rep(1, 2) %*% temp)) * prod(factorial(rep(1, 3) %*% t(temp))))
den <- (factorial(sum(temp)) * prod(factorial(temp)))
term2 <- (num/den)

term1 + term2  # sum the two probabilities
[1] 0.01830808
```
Chapter 4 - Agresti

A. Generalized Linear Models

To illustrate a logit, probit, and linear probability model, Agresti uses Table 4.2 (Cancer remissions, p. 88). To set up the data, type,

```r
n<-rep(c(2,3,1,3,2,1,3),c(2,3,1,1,1,5,1))
remissions<-rep(c(0,1,2,1,0,1,2),c(5,1,1,1,1,2,1,1,1))
LI<-rep(c(8,10,12,14,16,18,20,22,24,26,28,32,34,38),n)
LI.labels<-c("8-12","14-18","20-24","26-32","34-38")
LI.fac<-factor(cut(unique(LI),breaks=c(7,13,19,25,33,39)),labels=LI.labels) # make a factor out of LI
```

```r
temp<-table(LI.fac,remissions) # cross-classify LI.fac and remissions
temp2<-aggregate(cbind(LI.fac,n),by=LI.fac,FUN="sum")["n"] # get table 4.3 from table 4.2 (first three columns)
cancer.fr<-cbind.data.frame(LI.labels,temp2,temp[,2]+2*temp[,3],row.names=NULL)
names(cancer.fr)<-c("LI","no.cases","no.remissions")
cancer.fr
```

<table>
<thead>
<tr>
<th>LI</th>
<th>no.cases</th>
<th>no.remissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8-12</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>14-18</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>20-24</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>26-32</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>34-38</td>
<td>4</td>
</tr>
</tbody>
</table>

To calculate and plot empirical logits, type:

```r
attach(cancer.fr)
plot(1:5,log((no.remissions+.5)/(no.cases-no.remissions+.5)),ylab="logit",xlab="")
```

(plot not shown)

Now, I fit a glm. Since we have zeroes as responses, I use the function `glim`.

For a logit link:

```r
res1<-glim(x=unique(LI), y=remissions,n=n, error="binomial", link="logit", resid="deviance")
```

```r
glim.print(res1)

<table>
<thead>
<tr>
<th>coef</th>
<th>se(coef)</th>
<th>z</th>
<th>p</th>
<th>Deviance</th>
<th>df</th>
<th>change</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>-3.7771</td>
<td>1.37681</td>
<td>-2.74</td>
<td>0.0061</td>
<td>23.96</td>
<td>13</td>
<td>0.004</td>
</tr>
<tr>
<td>X1</td>
<td>0.1449</td>
<td>0.05925</td>
<td>2.44</td>
<td>0.0145</td>
<td>15.66</td>
<td>12</td>
<td>8.3</td>
</tr>
</tbody>
</table>
```

To obtain the odds ratio, type:

```r
as.numeric( exp(res1$coef["X1"])) # as.numeric strips the labeling here
[1] 1.155881
```

To obtain the Wald statistic (p.89), extract the ASE. The Wald statistic is $z^2$ from the `glim` output above.

```r
ASE<-sqrt(res1$var[2,2])
(res1$coef["X1"]/ASE)^2
```

```r
[1] 8.3
```
One can obtain the LR statistic under the column "change" in the glim output above. It can also be obtained using:

```
```

Now, I demonstrate the same analysis using a probit link.

```
res2 <- glm(x = unique(LI), y = remissions, n = n, error = "binomial", link = "probit", resid = "deviance")
summary(res2)
```

And, an identity link (least squares fit):

```
remissions <- c(rep(0,13),1,1,1,0,1,0,0,1,1,0,1,1,1,0) # reset remissions to 0/1
res3 <- glm(remissions ~ LI, family = binomial(link = identity), control = glm.control(maxit = 30))
summary(res3)$coefficients
```

We can obtain fitted probabilities for all three link functions.

Here are the fitted logistic probabilities:

```
logits <- unique(res1$coef["Intercept"] + LI*res1$coef["X1"])
pi.logis <- exp(logits)/(1 + exp(logits))
matrix(pi.logis*rep(c(2,3,1,3,2,1,3),c(2,3,1,1,1,5,1)))
```

probit probabilities:

```
probits <- unique(res2$coef["Intercept"] + LI*res2$coef["X1"])
pi.probits <- pnorm(probits)
matrix(pi.probits, ncol = 1)
```
To reproduce Figure 4.2, use the following commands:

```r
LI.plot <- unique(LI)
plot(LI, predict(res3), type="n", xlim=c(0, 40), ylim=c(-.005, 1.0), xlab="Labeling Index (LI)", ylab="pi-hat") # set-up plot using the largest spread of values
lines(LI.plot, pi.logis, type="b", pch=17)
lines(LI.plot, pi.probits, type="l", lty=5)
lines(LI.plot, predict(res3), type="l", lty=1)
key(x=5, y=.9, text=list(c("Logistic", "Probit", "Linear")), lines=list(type=c("b", "l", "l")), lty=c(1, 5, 1), pch=c(17, 1, 1), divide=3, border=T)
```

linear probabilities:

```r
matrix(predict(res3))
```

```
[1,] -0.002668923
[2,] -0.002668923
[3,]  0.052987892
[4,]  0.052987892
[5,]  0.108644708
[6,]  0.108644708
[7,]  0.108644708
[8,]  0.164301523
[9,]  0.164301523
[10,]  0.164301523
[11,]  0.219958339
[12,]  0.219958339
[13,]  0.219958339
[14,]  0.275615154
[15,]  0.331271970
[16,]  0.331271970
[17,]  0.331271970
[18,]  0.386928785
[19,]  0.386928785
[20,]  0.442585601
[21,]  0.498242416
[22,]  0.553899232
[23,]  0.665212863
[24,]  0.720869678
[25,]  0.832183309
[26,]  0.832183309
[27,]  0.832183309
```
B. Logit models for categorical data

For this section, I use the data in Table 4.4 (p.93).

```r
BP<-factor(c("<117","117-126","127-136","137-146","147-156","157-166","167-186",">186"))
CHD<-c(3,17,12,16,12,8,16,8)
n<-c(156,252,284,271,139,85,99,43)

To fit a GLM to Table 4.4, type:

Saturated Model (with null deviance):
resCHD<-glm(cbind(CHD,n-CHD)~BP,family=binomial(link=logit))
summary(resCHD)$null.deviance
[1] 30.02257

Calculate sample logits:

matrix(predict(resCHD),ncol=1)
[1,] -3.931826
[2,] -2.626372
[3,] -3.120895
[4,] -2.768675
[5,] -2.359280
[6,] -2.264364
[7,] -1.646252
[8,] -1.475907

Linear logit model

To illustrate fitting a linear logit model, Agresti uses the blood pressure data in Table 4.5 (p. 94).

scores<-c(seq(from=111.5,to=161.5,by=10),176.5,191.5)
resLL<-glm(CHD/n~scores,family=binomial,link=logit,weights=n)
summary(resLL)$coefficients
```
<table>
<thead>
<tr>
<th>Value</th>
<th>Std. Error</th>
<th>t value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>-6.08202789</td>
<td>0.723980447</td>
</tr>
<tr>
<td>scores</td>
<td>0.02433821</td>
<td>0.004841677</td>
</tr>
</tbody>
</table>

To compute the LR statistic, type:

```r
res$null.deviance-res$deviance
```

[1] 24.11341

To calculate estimated probabilities and expected frequencies, use `fitted.values`:

```r
probs<-fitted.values(resLL)
matrix(n*probs,ncol=1)
```

[1,] 5.194869
[2,] 10.606767
[3,] 15.072743
[4,] 18.081622
[5,] 11.616362
[6,] 8.856988
[7,] 14.208763
[8,] 8.361957

Here, I reestimate the linear logit model using an ordered factor:

```r
BPo<-as.ordered(BP) # make BP an ordered factor
res<-glm(cbind(CHD,n-CHD)~BPo,family=binomial(link=logit))
```

```
summary(res)$coefficients
```

<table>
<thead>
<tr>
<th>Value</th>
<th>Std. Error</th>
<th>t value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>-2.52419635</td>
<td>0.1258519</td>
</tr>
<tr>
<td>BPo.L</td>
<td>0.57561274</td>
<td>0.3682757</td>
</tr>
<tr>
<td>BPo.Q</td>
<td>0.04585354</td>
<td>0.3277022</td>
</tr>
<tr>
<td>BPo.C</td>
<td>0.24410276</td>
<td>0.3385445</td>
</tr>
<tr>
<td>BPo ^ 4</td>
<td>1.39413030</td>
<td>0.4080938</td>
</tr>
<tr>
<td>BPo ^ 5</td>
<td>1.01019203</td>
<td>0.3879685</td>
</tr>
<tr>
<td>BPo ^ 6</td>
<td>0.89519565</td>
<td>0.3258823</td>
</tr>
<tr>
<td>BPo ^ 7</td>
<td>0.46217593</td>
<td>0.3256979</td>
</tr>
</tbody>
</table>

The default setting for contrasts with ordered factors is polynomial contrasts. With eight levels of BP, we can fit up to septic effects. However, it is likely that any significant effects above cubic are spurious.

To reproduce Figure 4.3, use the following commands:

```r
plot(scores,CHD/n,pch="X",yaxt="n",xaxt="n",ylim=c(0,.2),xlab="Blood Pressure Level",ylab="Proportion")
axis(side=1,ticks=T,at=seq(from=110,to=200,by=10))
axis(side=2,ticks=T,at=seq(from=0,to=.2,by=.05))
lines(scores,probs,type="l")
```
I will use the data from Table 4.7 (p.106) to mention some properties of glm objects.

To set up the data, type:

```r
x<-c(1.691,1.724,1.755,1.784,1.811,1.837,1.861,1.884)
n<-c(59,60,62,56,63,59,62,60)
y<-c(6,13,18,28,52,53,61,60)
```

The following two calls give the same estimates for this problem:

```r
res1<-glm(y=y,x=x,n=n,error="binomial",link="loglog")
res1<-glm(y/n-x,weights=n,family=binomial(link=cloglog)) # this is the res1 used in the following commands
```

The summary for a glm.object has the following attributes that can be extracted:

```r
attributes(summary(res1))
```

$names:

1. "call"
2. "terms"
3. "coefficients"
4. "dispersion"
5. "df"
6. "deviance.resid"
7. "cov.unscaled"
8. "correlation"
9. "deviance"
10. "null.deviance"
11. "iter"
12. "nas"

$class:

1. "summary.glm"

For example, to extract the estimated coefficients, type:

```r
summary(res1)$coefficients
```

```
Value Std. Error   t value
(Intercept) -39.52232   3.234805 -12.21784
 x  22.01478   1.796516  12.25415
```

The glm.object itself has the following components. It inherits all the attributes of lm.objects

```r
attributes(res1)
```

$names:

1. "coefficients"
2. "residuals"
3. "fitted.values"
4. "effects"
5. "R"
6. "rank"
7. "assign"
8. "df.residual"
9. "weights"
For example, the fitted values in column 4 in Table 4.7 are

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
5.653471 & 11.28156 & 20.94184 & 30.33947 & 47.68091 & 54.18769 & 61.11662 & 59.94814 \\
\end{array}
\]

The same answer can be obtained by the call \( n*fitted(res1) \). The functions, \( fitted \), \( resid \), \( coef \) are shortened versions to extract fitted values, residuals, and coefficients from glm objects.

The data in Table 4.7 can be fit using the methods discussed above. A complementary log-log link is specified using the link="cloglog" option for function \( glim \) and the family=binomial(link=cloglog) option for function \( glm \).

### Model Diagnostics - Residuals

To demonstrate how to obtain the residuals in Agresti's section 4.6.1, I use again the Blood Pressure / Heart Disease data in Table 4.4 to reproduce Table 4.8.

First, fit the independence model:

\[
res<-glm(cbind(CHD,n-CHD)~1,family=binomial(link=logit))
\]

\[
n*fitted.values(res)
\]

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]

Get Pearson residuals for the independence model:

\[
matrix(resid(res,type="pearson"))
\]

\[
\begin{array}{cccccccc}
1,1 & -2.4599611 & -0.1103592 & -1.7906464 & -0.6604895 & 0.7945128 & 0.9041221 & 3.6215487 & 3.0178895 \\
\end{array}
\]

Now look at the linear logit model:

Get fitted values for the linear logit model:

\[
matrix(n*fitted.values(resLL))
\]

\[
\begin{array}{cccccccc}
\end{array}
\]

Get Pearson residuals for the linear logit model:


Deviance residuals are obtained using type="deviance". Residuals can be obtained from glim fits by extracting them via the dollar sign, $residuals

C. Overdispersed Poisson Model (Negative Binomial)

To estimate a negative binomial model, one must create a family. The function make.family can be used to make a new glm family. You must specify the name of the family, the link functions allowed for it (e.g., logit, log, cloglog) and their derivatives, inverses, and initialization expressions (used to initialize the linear predictor for Fisher scoring algorithm), and the variance and deviance functions.

The function negative.binomial is available from Venables and Ripley’s MASS library available from the website, www.statlib.cmu.edu. In the function listing, you can see where the links and variances lists are defined and where the function make.family is used. The reason why the link functions are not actually typed out in function form is because all the links already appear in other glm families in Splus. Their information is stored in the matrix glm.links. negative.binomial accesses the appropriate link function using glm.links[, link].

A simpler version of negative.binomial (called neg.bin, with only the log link) is also available from MASS, as well as a function called glm.nb. The function glm.nb assumes Y~negative binomial with pdf

\[
f_Y(y; \theta, \mu) = \frac{\Gamma(\theta+y)}{\Gamma(\theta)y!} \left(\frac{\mu^\theta}{\mu+\theta}\right)^y, \quad \text{mean } \mu, \text{ and variance } \mu + \mu^2/\theta, \text{ and allows one to estimate } \theta \text{ using ML.}
\]

void NegBin

```r
negative.binomial <- function(theta = stop("theta must be specified"), link = "log")
{
  link <- as.character(substitute(link))
  misnames <- c("log", "identity", "sqrt", "Log", "Identity", "Sqrt")
  corresp <- c(1, 2, 3, 1, 2, 3)
  lmatch <- pmatch(link, misnames, F)
  if(!lmatch)
    stop(paste("Negative binomial links are currently only",
      "log", "identity", "sqrt", "Log", "Identity", "Sqrt")
      corresp <- c(1, 2, 3, 1, 2, 3)
  lmatch <- pmatch(link, misnames, F)
  if(!lmatch)
    stop(paste("Negative binomial links are currently only",
      "log", "identity", "sqrt", "Log", "Identity", "Sqrt")
      corresp <- c(1, 2, 3, 1, 2, 3)
  lmatch <- pmatch(link, misnames, F)
  if(!lmatch)
    stop(paste("Negative binomial links are currently only",
      "log", "identity", "sqrt", "Log", "Identity", "Sqrt")
      corresp <- c(1, 2, 3, 1, 2, 3)
```

```r
  link <- misnames[corresp[lmatch]]
  var <- list(name = paste("mu + mu^2", format(round(theta, 4)), sep = "/"),
    variance = substitute(function(mu, th = .theta)
      mu * (1 + mu/th), list(.theta = theta)),
    deviance = substitute(
      function(mu, y, w, residuals = F, th = .theta)
      {
        devi <- 2 * w * (y * log(pmax(1, y)/mu) - (y + th) * log((y +
            th)/(mu + th)))
        if(residuals)
          sign(y - mu) * sqrt(abs(devi))
        else sum(devi)
      }, list(.theta = theta)))
```
make.family("Negative Binomial", glm.links[, link], var)

nnn function neg.bin (MASS library, V&R) nnn
neg.bin<-
function(theta = stop("theta must be given"))
{
  nb.lnk <- list(names = "Log: log(mu)", link = function(mu)
    log(mu), inverse = function(eta)
    exp(eta), deriv = function(mu)
    1/mu, initialize = expression(mu <- y + (y == 0)/6))
  nb.var <- list(names = "mu + mu^2/theta", variance = substitute(
    function(mu, th = .Theta)
    mu * (1 + mu/th), list(.Theta = theta)), deviance = substitute(
    function(mu, y, A, residuals = F, th = .Theta)
    {
      devi <- 2 * A * (y * log(pmax(1, y)/mu) - (y + th) * log((y +
        th)/(mu + th)))
      if(residuals)
        sign(y - mu) * sqrt(abs(devi))
      else sum(devi)
    }, list(.Theta = theta))
  make.family("Negative Binomial", link = nb.lnk, variance = nb.var)
}

To briefly illustrate its use, I take an excerpt from an example in V&R (1994), using the data set quine included with MASS (this part probably needs permission before I include it):

##### Quine data set (in Splus): number days absent from school

<table>
<thead>
<tr>
<th>Eth</th>
<th>Sex</th>
<th>Age</th>
<th>Lrn</th>
<th>Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>M</td>
<td>F0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>M</td>
<td>F0</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>M</td>
<td>F0</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>M</td>
<td>F0</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>M</td>
<td>F0</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>M</td>
<td>F0</td>
<td>13</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

fit<-glm.nb(formula = Days ~ Sex + Age %in% Sex + Eth %in% Sex + Lrn %in% Sex +
Eth:Lrn %in% Sex, data = quine, init.theta = 1.59806882991095, link = log) # %in%
refers to nesting

An estimate of theta is obtained from the fit:

\[
\text{c(theta=fit$theta, SE=fit$SE)}
\]

\[
\begin{array}{ll}
\text{theta} & 1.598044 \\
\text{SE} & 0.2128835
\end{array}
\]

D. Extended Example (problem 4.7)

This example illustrates some details in using Splus for a logit model. The analysis is patterned after the section on binomial data in V&R (1994, 1997, p. 230).

First, we set the type of contrast to treatment contrasts for factors.

\[
\text{options(contrasts=c("contr.treatment"},"contr.poly"))}
\]

Now, we get the data set up:

\[
\text{view<-rep(1:7,2)}
\]
view
[1] 1 2 3 4 5 6 7 1 2 3 4 5 6 7

Reagan<-c(1,13,44,155,92,100,18,0,0,2,1,0,2,0)
race<-factor(rep(c("white","nonwhite"),c(7,7)))
race
[1] white white white white white white white nonwhite
[9] nonwhite nonwhite nonwhite nonwhite nonwhite nonwhite nonwhite

Carter<-c(12,57,71,146,61,41,8,6,16,23,31,8,7,4)
presvotes<-cbind(Reagan,Carter)

Now, fit a binomial glm with interaction:

presvotes.lg<-glm(presvotes ~ race*view, family=binomial)
summary(presvotes.lg)

Call: glm(formula = presvotes ~ race * view, family = binomial)
Deviance Residuals:
Min       1Q    Median        3Q      Max
-1.443006 -1.07026 -0.492815 0.2732216 1.083453

Coefficients:
                        Value     Std. Error   t value
(Intercept)  -4.40223924  1.40305708 -3.1376054
race       2.35759965  1.42833712  1.6505906
view       0.36575214  0.30485056  1.1997751
race:view   0.12995204  0.31080087  0.4181197

(Dispersion Parameter for Binomial family taken to be 1 )

Null Deviance: 185.1567 on 13 degrees of freedom
Residual Deviance: 12.29655 on 10 degrees of freedom
Number of Fisher Scoring Iterations: 4

Correlation of Coefficients:

            (Intercept)    race      view
race -0.9823011
view -0.9441010  0.9273914
race:view  0.9260258 -0.9447185 -0.9808546

Plot the interaction:

plot(c(1,7),c(0,1), type="n", xlab="political view", ylab="prob")
text(view,Reagan/(Carter+Reagan),as.character(race))
ld<-seq(1,7,0.1)

Add lines for white and non-white:

lines(ld,predict(presvotes.lg,data.frame(view=ld,race=factor(rep("white",length(ld)), levels=levels(race))), type="response"))
We can test for a race difference at a particular view, say view=5. The `I()` function is used so that (view-5) is interpreted as is, meaning as a number, here.

```r
presvotes.lgA <- glm(presvotes ~ race * I(view - 5), family = binomial)
summary(presvotes.lgA)
```

Call: glm(formula = presvotes ~ race * I(view - 5), family = binomial)

Deviance Residuals:
Min       1Q    Median       3Q      Max
-1.443006 -1.07026 -0.492815  0.2732216  1.083453

Coefficients:
                     Value Std. Error    t value
(Intercept)    -2.5734788  0.5037669 -5.1084713
race           3.0073596  0.5110801  5.8843217
I(view - 5)    0.3657521  0.3048505  1.1997751
race: I(view - 5)  0.1299520  0.3108009  0.4181197

(Dispersion Parameter for Binomial family taken to be 1)

Null Deviance: 185.1567 on 13 degrees of freedom
Residual Deviance: 12.29655 on 10 degrees of freedom
Number of Fisher Scoring Iterations: 4

Correlation of Coefficients:
                     (Intercept)   race   I(view - 5)
(Intercept)    -0.9856907
race           0.3905948
I(view - 5)    -0.3886784  0.4003841 -0.9808546

`step` is a function for doing a stepwise analysis. The component `anova` is the most useful. We use it here to find out whether it picks a model with interaction or not. The model with the lowest AIC is considered best.

```r
step(presvotes.lg, trace=F)$anova
```

Stepwise Model Path
Analysis of Deviance Table
Initial Model:
presvotes ~ race * view

Final Model:
presvotes ~ race + view

<table>
<thead>
<tr>
<th></th>
<th>Step Df</th>
<th>Deviance</th>
<th>Resid. Df</th>
<th>Resid. Dev</th>
<th>AIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>10</td>
<td>12.29655</td>
<td>20.29655</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>- race:view</td>
<td>1</td>
<td>0.1737404</td>
<td>11</td>
<td>12.47030</td>
</tr>
</tbody>
</table>

The independence model Race + view is chosen. Here is the Analysis of deviance:

```r
anova(glm(presvotes~race + view, family=binomial), test="Chi")
```

Analysis of Deviance Table

Binomial model

Response: presvotes

Terms added sequentially (first to last)

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>Deviance</th>
<th>Resid. Df</th>
<th>Resid. Dev</th>
<th>Pr(Chi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>13</td>
<td>185.1567</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>race</td>
<td>1</td>
<td>95.49444</td>
<td>12</td>
<td>89.6622</td>
<td>0</td>
</tr>
<tr>
<td>view</td>
<td>1</td>
<td>77.19194</td>
<td>11</td>
<td>12.4703</td>
<td>0</td>
</tr>
</tbody>
</table>

Now, we test a model with parallel lines for each race (common slope), but different intercepts. The term `-1' in the formula removes the common intercept (forces it through 0) so that the coefficient estimate on the 1-coded level of race becomes the new nonzero intercept for that level.

```r
presvotes.lgB<-glm(presvotes~race + view - 1, family=binomial)
```

```r
summary(presvotes.lgB)$coefficients
```

<table>
<thead>
<tr>
<th>Value</th>
<th>Std. Error</th>
<th>t value</th>
</tr>
</thead>
<tbody>
<tr>
<td>racenonwhite</td>
<td>-4.9608574</td>
<td>0.53948823</td>
</tr>
<tr>
<td>racewhite</td>
<td>-2.0239493</td>
<td>0.26239118</td>
</tr>
<tr>
<td>view</td>
<td>0.4908373</td>
<td>0.05926813</td>
</tr>
</tbody>
</table>

One result we can obtain from setting different intercepts is a prediction of the viewpoint at which prob(vote for Reagan) = .25, .5, .75 for nonwhites/whites. We can use the function `dose.p` from the MASS library. The second argument to the function (i.e., c(1, 3) or c(2, 3) ) refers to the coefficients specifying the common slope and separate intercept. For nonwhites, they are the racenonwhite coefficient and view coefficient. The third argument refers to the probability points (.25, .5, .75).

For nonwhites, we have the following predictions:

```r
dose.p(presvotes.lgB,c(1,3),(1:3)/4)
```

<table>
<thead>
<tr>
<th>Dose</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>p = 0.25:</td>
<td>7.868887</td>
</tr>
<tr>
<td>p = 0.50:</td>
<td>10.106928</td>
</tr>
<tr>
<td>p = 0.75:</td>
<td>12.345169</td>
</tr>
</tbody>
</table>

And, for whites, we have:

```r
dose.p(presvotes.lgB,c(2,3),(1:3)/4)
```

<table>
<thead>
<tr>
<th>Dose</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>p = 0.25:</td>
<td>1.885221</td>
</tr>
<tr>
<td>p = 0.50:</td>
<td>4.123463</td>
</tr>
<tr>
<td>p = 0.75:</td>
<td>6.361704</td>
</tr>
</tbody>
</table>
To print residual plots for the model: presvotes ~ race + view, type:

```r
par(mfrow=c(2, 2))
plot(presvotes.lg, ask=T)
```

Make a plot selection (or 0 to exit):

1: plot: All
2: plot: Residuals vs Fitted Values
3: plot: Sqrt of abs(Residuals) vs Predictions
4: plot: Response vs Fitted Values
5: plot: Normal QQplot of Std. Residuals
Selection: 1
Chapter 5 - Agresti

A. Demonstration of Simpson’s Paradox

The set of commands in this section illustrates how to get from Table 5.1 to 5.2 (i.e., collapse a cross-classified table output from `crosstabs`)

To set up the data:
```r
DeathP<-c("Yes", "No")
Victim.race<-c("White", "Black")
Defend.race<-c("White", "Black")

table.5.1<-expand.grid(DeathP, Victim.race, Defend.race) # compare this to the function fac.design

names(table.5.1)<-c("DeathP", "Defendent", "Victim")
```

Next, add the counts to `table.5.1` and put both into a data frame:
```r
table.5.1<-data.frame(table.5.1, counts=c(19, 132, 11, 52, 0, 9, 6, 97))

crosstabs(counts~DeathP+Defendent+Victim,table.5.1, margin=list()) # (margin=list() causes only the counts to show in the cells – see the help file for more, as well as the default)
```

<table>
<thead>
<tr>
<th>Victim=White</th>
<th>DeathP</th>
<th>Defendent</th>
<th>RowTotl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>19</td>
<td>11</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.140</td>
</tr>
<tr>
<td>No</td>
<td>132</td>
<td>52</td>
<td>184</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.860</td>
</tr>
<tr>
<td>ColTotl</td>
<td>151</td>
<td>63</td>
<td>214</td>
</tr>
<tr>
<td></td>
<td>0.71</td>
<td>0.29</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Victim=Black</th>
<th>DeathP</th>
<th>Defendent</th>
<th>RowTotl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>0</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.054</td>
</tr>
<tr>
<td>No</td>
<td>9</td>
<td>97</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.946</td>
</tr>
<tr>
<td>ColTotl</td>
<td>9</td>
<td>103</td>
<td>112</td>
</tr>
<tr>
<td></td>
<td>0.08</td>
<td>0.92</td>
<td></td>
</tr>
</tbody>
</table>

Test for independence of all factors

Chi^2 = 122.3975 d.f.= 4 (p=0)

Yates' correction not used
Now, to collapse Table 5.1 to Table 5.2, use

crosstabs(counts ~ DeathP + Defendant, table.5.1, margin = list())

<table>
<thead>
<tr>
<th>DeathP</th>
<th>Defendent</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>White</td>
<td>Black</td>
</tr>
<tr>
<td>Yes</td>
<td>19</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>141</td>
<td>149</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ColTotl</td>
<td>160</td>
<td>166</td>
</tr>
<tr>
<td></td>
<td>0.49</td>
<td>0.51</td>
</tr>
</tbody>
</table>

Test for independence of all factors
Chi^2 = 0.2214463 d.f. = 1 (p=0.6379401)
Yates' correction not used
A. Fitting Loglinear Models

If the data are already in a data frame with counts as one variable, as table.5.1 is, the easiest method to use to fit a loglinear model is `glm` with family=poisson and a log link (called a surrogate poisson). There is also a function in the MASS library called `loglm` which allows input from a data frame (using a formula). The function `loglin`, which comes with Splus, requires input in the form of an array such as output from `table` or `matrix` or `array`. (`loglm` can also take an array as input). It is easy to get an array from a data frame: issue the `design.table` command with the data frame as the argument. (Note, also, that the function `as.data.frame.array` achieves the opposite effect: that is, returning a data frame as the equivalent expression of an array)

Agresti uses the data in Table 6.3 (Death Penalty Example, p.171-174) to illustrate fitting of a loglinear model.

Since we already have table.5.1 in a data frame, let's use `loglm` from the MASS library. (I will illustrate the other functions later, especially the use of `glm`).

The next set of commands fits the models in table 6.3 (not in the same order)

Model (D,V,P):
```
fitDVP<-loglm(counts~.,data=table.5.1,param=T,fit=T) # '.' = DeathP+Defendent+Victim
```

Obtain expected frequencies:
```
fitted.values(fitDVP)
```
```
,, White             White     Black
Yes  11.59848       12.03342
  No  93.43220       96.93591
```

```
summary(fit)
```
```
X^2 df P(> X^2)
Likelihood Ratio 137.9294  4        0
  Pearson 122.3975  4        0
```

Model (VP,DV) :
```
fitVP.DV<-update(fitDVP, .~ DeathP*Victim + Defendent*Victim)
```
```
fitted.values(fitVP.DV)
```
```
,, White               White     Black
Yes  21.16822        0.4821429
  No 129.83177       8.5178576
```

```
summary(fitVP.DV)

Statistics:

<table>
<thead>
<tr>
<th></th>
<th>X^2</th>
<th>df</th>
<th>P(&gt; X^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood Ratio</td>
<td>1.881909</td>
<td>2</td>
<td>0.3902552</td>
</tr>
<tr>
<td>Pearson</td>
<td>1.431339</td>
<td>2</td>
<td>0.4888648</td>
</tr>
</tbody>
</table>

Model (DVP) - Saturated:

fit.DVP<-update(fitDVP, .~. ^2 + Defendent*Victim*DeathP)

fitted.values(fit.DVP)

summary(fit.DVP)

Statistics:

<table>
<thead>
<tr>
<th></th>
<th>X^2</th>
<th>df</th>
<th>P(&gt; X^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood Ratio</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Pearson</td>
<td>NA</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Model (P,DV):

fitP.DV<-loglm(counts~ DeathP + Defendent*Victim,data=table.5.1,param=T,fit=T)

same as

fitP.DV<-update(fitDVP, .~. + Defendent*Victim)

Model (DP,VP,DV) - Note how the formula is written

fitDP.VP.DV<-loglm(counts~(Defendent+Victim+DeathP)^2,data=table.5.1,param=T,fit=T)

same as

fitDP.VP.DV<-update(fitDVP, .~. ^2)

B. Partial and Marginal Odds Ratios

Obtaining partial and marginal odds ratios is illustrated in Table 6.4 (p. 173). Here, I show how to reproduce that table.

All 1.0 odds ratios are 1.0 by design. For example, in the independence model, when the victim is white, the odds of a white defendant being given the death penalty are the same as for a black defendant. The non-unity odds ratios must be computed either by using the estimated frequencies (easiest) or by using the parameter estimates. In the latter case, you must derive the expression for the odds ratio, which can be very tedious if the model is complicated. Thus, I have only used one of the models in Table 6.4.

Compute partial and marginal odds ratios for association D-V under model (VP,DV):
Get estimated parameters:

\[
\text{lambdaD} \leftarrow \text{fitVP.DV$param$Defendent} \\
\text{lambdaV} \leftarrow \text{fitVP.DV$param$Victim} \\
\text{lambdaP} \leftarrow \text{fitVP.DV$param$DeathP} \\
\text{lambdaDV} \leftarrow \text{fitVP.DV$param$"Victim X Defendent"} \\
\text{lambdaVP} \leftarrow \text{fitVP.DV$param$"DeathP X Victim"}
\]

Partial association (D-V):

\[
\exp(\lambda_{DV}[1,1] + \lambda_{DV}[2,2] - \lambda_{DV}[1,2] - \lambda_{DV}[2,1])
\]

[1] 27.43033

Marginal association (D-V) - sum over levels of P

\[
\text{num} \leftarrow \exp(\lambda_D[1] + \lambda_V[1] + \lambda_P[1] + \lambda_{DV}[1,1] + \lambda_{VP}[1,1]) + \\
\exp(\lambda_D[1] + \lambda_V[1] + \lambda_P[2] + \lambda_{DV}[1,1] + \lambda_{VP}[1,2]) + \\
\exp(\lambda_D[2] + \lambda_V[2] + \lambda_P[1] + \lambda_{DV}[2,2] + \lambda_{VP}[2,1]) + \\
\exp(\lambda_D[2] + \lambda_V[2] + \lambda_P[2] + \lambda_{DV}[2,2] + \lambda_{VP}[2,2])
\]

\[
\text{den} \leftarrow \exp(\lambda_D[2] + \lambda_V[1] + \lambda_P[1] + \lambda_{DV}[2,1] + \lambda_{VP}[1,1]) + \\
\exp(\lambda_D[2] + \lambda_V[1] + \lambda_P[2] + \lambda_{DV}[2,1] + \lambda_{VP}[1,2]) + \\
\exp(\lambda_D[1] + \lambda_V[2] + \lambda_P[1] + \lambda_{DV}[1,2] + \lambda_{VP}[2,1]) + \\
\exp(\lambda_D[1] + \lambda_V[2] + \lambda_P[2] + \lambda_{DV}[1,2] + \lambda_{VP}[2,2])
\]

\[
\text{as.numeric( num/den )}
\]

[1] 27.43033

Note: I'm pretty certain this calculation is unnecessarily complicated.

C. Comparing nested models using LR tests

Table 6.6 (p.176) in Agresti displays a comparison of the set of hierarchical models (DVP), (DV,P), (VP,DV), (DP,VP,DV) using LR tests. One can reproduce this table using the \texttt{anova} function.

\[
\text{anova(fitDVP,fitP.DV,fitVP.DV,fitDP.VP.DV)}
\]

Model 1:
\[
\text{counts} \sim \text{DeathP} + \text{Victim} + \text{Defendent}
\]
Model 2:
\[
\text{counts} \sim \text{Defendent:Victim} + \text{DeathP}
\]
Model 3:
\[
\text{counts} \sim \text{DeathP} + \text{Victim} + \text{Defendent} + \text{DeathP:Victim} + \text{Defendent:Victim}
\]
Model 4:
\[
\text{counts} \sim (\text{Defendent} + \text{Victim} + \text{DeathP})^2
\]

<table>
<thead>
<tr>
<th>Model</th>
<th>Deviance</th>
<th>df</th>
<th>Delta(Dev)</th>
<th>Delta(df)</th>
<th>P(&gt; Delta(Dev))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>137.9293816</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model 2</td>
<td>8.1316088</td>
<td>3</td>
<td>129.7977728</td>
<td>1</td>
<td>0.00000</td>
</tr>
<tr>
<td>Model 3</td>
<td>1.8819090</td>
<td>2</td>
<td>6.2496997</td>
<td>1</td>
<td>0.01242</td>
</tr>
<tr>
<td>Model 4</td>
<td>0.7007495</td>
<td>1</td>
<td>1.1811596</td>
<td>1</td>
<td>0.27712</td>
</tr>
<tr>
<td>Saturated</td>
<td>0.0000000</td>
<td>0</td>
<td>0.7007495</td>
<td>1</td>
<td>0.40253</td>
</tr>
</tbody>
</table>

Based on the results, we should keep Model 3, (PV, DV). We cannot reject Model 3 in favor of a simpler model, Model 2 (DV,P).

D. Loglinear Models as Logit Models
To illustrate the relationship between loglinear and logit models, when one factor is treated as the response, I use the data from table 5.1. Here, death penalty is the response.

<table>
<thead>
<tr>
<th>DeathP</th>
<th>Defendant</th>
<th>Victim</th>
<th>counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>White</td>
<td>White</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>White</td>
<td>White</td>
</tr>
<tr>
<td>3</td>
<td>Yes</td>
<td>Black</td>
<td>White</td>
</tr>
<tr>
<td>4</td>
<td>No</td>
<td>Black</td>
<td>White</td>
</tr>
<tr>
<td>5</td>
<td>Yes</td>
<td>White</td>
<td>Black</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>White</td>
<td>Black</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Black</td>
<td>Black</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Black</td>
<td>Black</td>
</tr>
</tbody>
</table>

```r
yes<-table.5.1[DeathP=="Yes",]$counts  # gives (19, 11, 0, 6)
no<-table.5.1[DeathP=="No",]$counts
attach(table.5.1)
```

To fit the logit model:

```r
fit<-glm(cbind(no,yes) ~ Victim[c(2,4,6,8)], family=binomial, link="logit")
Victim[c(2,4,6,8)] gives (White, White, Black, Black)
```

Coefficients:

```
(Intercept) Victim[c(2, 4, 6, 8)]
  2.342709             0.5289706
```

Degrees of Freedom: 4 Total; 2 Residual
Residual Deviance: 1.881895

The relationship between the parameters in the logit model and loglinear model (VP,DV) is displayed here. Note the value for the coefficient on Victim in the logit model:

```r
fitVP.DV$param$"DeathP X Victim"[1,1] - fitVP.DV$param$"DeathP X Victim"[1,2]
[1] 0.5289706
```

### E. Asymptotic Standard Errors and Wald tests

Standard errors and Wald tests can be obtained from the summary of a `glm.object` but not (directly) from a `loglin` object (or `loglm` object). However, one can always extract the model matrix, X, and use the estimated frequencies to compute (6.12) or (6.13) in Agresti.

I will use the Death Penalty data to show part of Table 6.7.

To obtain the model matrix, use the function `model.matrix` with the first argument a `loglm` fit.

```r
X<-model.matrix(fitVP.DV, data=table.5.1)
```

Get the total number of cases

```r
n<-sum(fitVP.DV$frequencies)
```
and the vector of expected frequencies:
\[
mhat = \text{matrix(fitted.values(fitVP.DV), ncol=1)}
\]

Then, the asymptotic covariance matrix (6.12) is computed as
\[
covbeta = \text{solve(t(X)} %*% (\text{diag(as.numeric(mhat))}) %*% X) \quad \text{# poisson sampling approximation}
\]

And, the ASEs in Table 6.7 should be (?)
\[
sqrt(diag(covbeta))[5:6]
\]
\[
[1] 0.09070108 0.09496805 \quad \text{# Table 6.7 has .116 and .095 (the first does not match)}
\]

F. Probability Estimates

Starting with the fitted values from the model (DV,VP) of the Death Penalty data,

\[
\begin{array}{cccc}
& & & \\
\text{White} & \text{Black} & \\
\text{Yes} & 21.16822 & 0.4821429 \\
\text{No} & 129.83177 & 8.5178576 \\
\end{array}
\]

\[
\begin{array}{cccc}
& & & \\
\text{White} & \text{Black} & \\
\text{Yes} & 8.831776 & 5.517857 \\
\text{No} & 54.168224 & 97.482140 \\
\end{array}
\]

we calculate the estimated probabilities of death penalty (Table 6.8) as
\[
\begin{array}{c}
\text{fit} = \text{fitted.values(fitVP.DV)} \\
\text{den} = \text{apply(fit,3,colSums)} \\
\text{fit}[1,1,1]/\text{den}[1,1] \\
\text{fit}[1,2,1]/\text{den}[2,1] \\
\text{fit}[1,1,2]/\text{den}[1,2] \\
\text{fit}[1,2,2]/\text{den}[2,2]
\end{array}
\]

\[
\begin{array}{c}
[1] 0.1401869 \\
[1] 0.05357143 \\
[1] 0.1401869 \\
[1] 0.05357143
\end{array}
\]

Asymptotic standard errors can be computed using equation (6.15)
\[
\begin{array}{c}
p = \text{mhat}/n \\
\text{Dpp} = \text{diag(as.numeric(p)} - p%*%t(p) \\
\text{covp} = \text{(Dpp}%*%X%*%solve(t(X))%*%Dpp%*%X%*%t(X))%*%Dpp)/n \\
\text{sqrt(diag(covp))}
\end{array}
\]

G. Analyzing survival rates using loglinear models

For this section, Agresti uses the data in Table 6.12 (p. 190) - Heart valve operations.

I will use the function \texttt{glm} to fit the model in Section 6.6.1 because I want to include the offset term, exposure.

First, set up the data
\[
\text{table.6.12<-expand.grid(c("Aortic","Mitral"), c("<55","55+"))} \\
\text{table.6.12<-data.frame(table.6.12, Deaths=c(4,1,7,9), Exposure=c(1259,2082,1417,1647))} \\
\text{names(table.6.12)[1:2]<-c("Valve","Age")} \\
\text{attach(table.6.12)}
\]
Add Risks to Table 6.12:

```r
table.6.12 <- data.frame(table.6.12, Risk = Deaths/Exposure)
```

<table>
<thead>
<tr>
<th>Valve</th>
<th>Age</th>
<th>Deaths</th>
<th>Exposure</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aortic</td>
<td>&lt;55</td>
<td>4</td>
<td>1259</td>
<td>0.0031771247</td>
</tr>
<tr>
<td>Mitral</td>
<td>&lt;55</td>
<td>1</td>
<td>2082</td>
<td>0.0004803074</td>
</tr>
<tr>
<td>Aortic</td>
<td>55+</td>
<td>7</td>
<td>1417</td>
<td>0.0049400141</td>
</tr>
<tr>
<td>Mitral</td>
<td>55+</td>
<td>9</td>
<td>1647</td>
<td>0.0054644809</td>
</tr>
</tbody>
</table>

Fit a glm with an offset term for log(Exposure):

```r
fit <- glm(Deaths ~ Valve + Age + offset(log(Exposure)), family = poisson, link = log)
```

Coefficients:

```
(Intercept)    Valve    Age
  -5.866556  -0.1649332  0.610474
```

Degrees of Freedom: 4 Total; 1 Residual
Residual Deviance: 3.222511

To get the chi-squared statistic, use `resid`:

```r
sum(resid(fit, type = "pearson")^2)
```

```
[1] 3.113503
```

Obtain the log odds of risk for older group and its asymptotic ASE (p. 191) using,

```r
c(logit = 2 * coef(fit)["Age"], ASE = 2 * summary(fit)$coefficients["Age", "Std. Error"])
```

```
logit.Age    ASE
  1.220948  0.5136586
```

To reproduce the fitted values for the glm in Table 6.13 (p. 192), use the following commands.

```r
attach(table.6.12)
mhat <- fitted(fit)
exphat <- fitted(fit)/Exposure
temp <- rbind(mhat, exphat)
array(temp, dim = c(2, 2, 2), dimnames = list(c("Deaths", "Risk"), Valve = c("Aortic", "Mitral"), Age = c("<55", "55+")))
```

```
, , <55
  Aortic  Mitral
Deaths 2.284108702 2.715892496
Risk   0.001814225 0.001304463

, , 55+
  Aortic  Mitral
Deaths 8.715891783 7.284108228
Risk   0.006150947 0.004422652
```

Now, Agresti compares the first model with a model that omits the Valve factor:

```r
fit2 <- glm(Deaths ~ Age + offset(log(Exposure)), family = poisson, link = log)
```

Model comparison test:
anova(fit2, fit, test="Chi")

Analysis of Deviance Table
Response: Deaths

<table>
<thead>
<tr>
<th>Terms</th>
<th>Resid. Df</th>
<th>Resid. Dev</th>
<th>Test Df</th>
<th>Deviance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age + offset(log(Exposure))</td>
<td>2</td>
<td>3.789717</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valve + Age + offset(log(Exposure))</td>
<td>1</td>
<td>3.222511</td>
<td>1</td>
<td>0.5672062</td>
</tr>
</tbody>
</table>

Pr(Chi)
1 0.4513712

We keep the model Count~Valve + Age + Offset(log(Exposure)).

To test a large set of models in a stepwise manner, one can use the function step with a lower and upper model specified. Cp and AIC criteria are given. Only the final model is printed unless trace=T is specified.

step(fit, list(lower = formula(fit), upper = ~ . + Valve*Age), scale=1, trace=T, direction="both")

The above specifies both forward and backward stepwise selection of terms with the starting model that was obtained from “fit” (in this case, “fit” is the output from the last glm we ran). The lower bound model is also the model represented by “fit”. The upper bound model includes a valve x age interaction. Since there is only one term separating the starting model from the upper model, a stepwise selection is not really necessary. Examples using step are given below.

H. Modeling survival times using loglinear models

Agresti illustrates fitting survival models as loglinear models using the data in Table 6.14. Since we have an offset, we'll have to use glm again to fit the model.

First, form the data set:

```r
table.6.14<-expand.grid(Stage=factor(c(1,2,3)), Histology=c("I","II","III"), Time=factor(c(0,2,4,6,8,10,12)))
table.6.14<-
cbind.data.frame(table.6.14,Deaths=c(9,12,42,5,4,28,1,1,19,2,7,26,2,3,1,0,3,5,12,3,5,10,1,3,7,10,10,10,10,10,10,2,4,5,1,1,6,1,4,5,2,2,0,0,0,3,3,3,4,2,1,3,1,0,3,1,4,1,2,4,2,0,2,3)),
Exposure=c(157,134,212,77,71,130,21,22,101,139,110,136,68,63,72,17,18,63,126,96,90,6,3,58,42,14,14,43,102,86,64,55,42,21,12,10,32,88,66,47,50,35,14,10,8,21,82,59,39,4,5,32,13,8,8,14,76,51,29,42,28,7,6,10))
```

Now, perform a loglinear analysis using glm:

```r
fit<-glm(Deaths~Histology+Time+Stage+offset(log(Exposure)),data=table.6.14, family=poisson,link=log)
```

summary(fit)

Coefficients:

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Std. Error</th>
<th>t value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>-2.503854618</td>
<td>0.07788986</td>
<td>-32.14609023</td>
</tr>
<tr>
<td>Histology1</td>
<td>0.081221146</td>
<td>0.06097426</td>
<td>1.33205626</td>
</tr>
<tr>
<td>Histology2</td>
<td>0.008773679</td>
<td>0.04708072</td>
<td>0.18635396</td>
</tr>
</tbody>
</table>
We choose a model using `step`.

```r
fit2<-glm(Deaths~Time+offset(log(Exposure)),data=table.6.14,family=poisson,link=log)
# (lower model)
step(fit,scope=list(lower=formula(fit2),upper=~.^2),direction="both",trace=T)$anova
```

Selecting the “anova” component gives the Analysis of Deviance table.

```
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
Deaths ~ Histology + Time + Stage + offset(log(Exposure))

Final Model:
Deaths ~ Time + Stage + offset(log(Exposure))

<table>
<thead>
<tr>
<th>Step</th>
<th>Df Deviance</th>
<th>Resid. Df</th>
<th>Resid. Dev</th>
<th>AIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>52</td>
<td>43.92253</td>
<td>63.9225</td>
</tr>
<tr>
<td>2</td>
<td>1.876473</td>
<td>54</td>
<td>45.79901</td>
<td>63.7990</td>
</tr>
</tbody>
</table>
```

Thus, `step` chooses a model without Histology.

**Illustration of using function `loglin`**

Finally, I will use the data in Table 6.16 to illustrate the Splus function `loglin` — loglinear analysis using an iterative proportional fitting algorithm. `loglm` actually calls `loglin`. The difference between the two is the way in which each handles structural zeros (`loglm` tries to correct the df, but doesn’t always get it right) and that `loglm` can take input in the form of a formula or an array. `loglin` requires input in the form of an array, and requires one to specify the marginal totals to be fit. Other differences, as well as further discussion, can be found in the *On-line Complements* to V&R (1997).

Here is the full call to `loglin`:

```r
loglin(table, margin, start = rep(1, length(table)), fit = FALSE, eps = 0.1, iter = 20,param = FALSE, print = TRUE)
```

Some examples of specifying marginal totals for three variables are:
- conditional independence model: `margin=list(c(1,2),c(1,3))`
- independence model: `margin=list(1,2,3)`
For table 6.16, first create an array of frequencies:

```r
temp <- c("G.Dis", "Mid.Pos", "G.App")
Att.Abortion <- rep(rep(temp, 3), c(209, 101, 237, 151, 126, 426, 16, 21, 138))
Schooling <- rep(ordered(c("LessHS", "HS", "MoreHS"), levels = c("LessHS", "HS", "MoreHS")), 
c(209+101+237, 151+126+426, 16+21+138))
table.6.16 <- table(Schooling, Att.Abortion)
```

Here I perform a loglinear analysis testing an independence model. The marginal totals to fit are those for the variables one and two. Thus, `margin=list(1, 2)`. I also request parameter estimates (param=T) and fitted frequencies (fit=T)

```r
loglin(table.6.16, margin=list(1, 2), param=T, fit=T)
```

2 iterations: deviation 0

```
$lrt:
[1] 96.52666

$pearson:
[1] 93.03375

$df:
[1] 4

$margin:
$margin[[1]]: 
[1] 1

$margin[[2]]: 
[1] 2

$fit: 
  G.Dis Mid.Pos G.App
LessHS 144.33122  95.19719 307.47156
   HS 185.49333 122.34666 395.16000
MoreHS  46.17544  30.45614  98.36842

$param: 
$param$constant: 
[1] 4.78923

$param$"1": 
  LessHS    HS    MoreHS
0.2962515 0.5471598 -0.8434113

$param$"2": 
  G.Dis Mid.Pos G.App
-0.1133705 -0.5295308  0.6429013
```
For weighted cells, the help file for loglin says, “The start argument can be used to produce analyses when the cells are assigned different weights, see Clogg and Eliason (1988). The start should be one over the weights.”
Chapter 7 - Agresti

A. Stepwise Model Selection

As discussed above, the function `step` is used for stepwise selection of glm models. Criteria for selection are Mallow’s $C_p$ and RSS. The component “anova” gives a summary of the trace path. To incorporate the sampling design in a stepwise selection, one could make use of the scope option, which allows one to give a lower and upper model.

To illustrate stepwise procedures, I follow Agresti’s Section 7.2.4, using the data in Table 7.3-5 (p. 219).

data:

```r
table.7.3 <- expand.grid(EMS=c("yes", "no"), PMS=c("yes", "no"), Gender=c("women", "men"), M.Status=c("divorced", "married"))
table.7.3 <- cbind.data.frame(table.7.3, count=c(17, 54, 36, 214, 28, 60, 17, 68, 4, 25, 4, 322, 11, 42, 4, 130))
```

Forward stepwise:

Fit the independence model (this is our initial model)

```r
fit <- glm(count ~ (EMS + PMS + Gender + M.Status), data=table.7.3, family=poisson, link="log")
res <- step(fit, direction="forward", scope=list(lower=formula(fit), upper=formula(count ~ (EMS + PMS + Gender + M.Status)^3)), trace=T, fit=T)
res$anova
```

<table>
<thead>
<tr>
<th>Resid. Df</th>
<th>Resid. Dev</th>
<th>AIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>232.1398</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>156.8804</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>92.4739</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>46.4576</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>21.0676</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>8.1535</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>5.2463</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>0.7569</td>
</tr>
</tbody>
</table>

Compare this table to Table 7.4.

Backward stepwise:

Fit initial model - all 2-way interactions:

```r
fit <- glm(count ~ (EMS + PMS + Gender + M.Status)^3, data=table.7.3, family=poisson, link="log")
res <- step(fit, direction="backward", scope=list(lower=formula(count ~ (EMS + PMS + Gender + M.Status)), upper=formula(count ~ (EMS + PMS + Gender + M.Status)^3)), trace=T, fit=T)
res$anova
```

<table>
<thead>
<tr>
<th>Resid. Df</th>
<th>Resid. Dev</th>
<th>AIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.1463554</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.1854976</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.3674045</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.7569237</td>
</tr>
</tbody>
</table>

Compare output to Table 7.5.
B. Adjusting a model using indicator functions

I use the data in Table 7.7 (p. 226) - Graduate school admissions - for this section.

```r
table.7.7 <- expand.grid(admit=c("yes","no"), gender=c("male","female"), dept=LETTERS[1:6])
table.7.7 <- cbind.data.frame(table.7.7, count=c(512,313,89,19,353,207,17,8,120,205,202,391,138,279,131,244,53,138,94,299,22,351,24,317))
```

I will fit the model in equation (7.7).

Start with model (AD,DG,GA):

```r
fit <- glm(count~.^2, data=table.7.7, family=poisson, link="log")
X <- model.matrix(fit)
```

Make the indicator in equation (7.7)

```r
X[table.7.7[,"dept"] == "A",][,"admit:gender"] <- 1
X[table.7.7[,"dept"] != "A",][,"admit:gender"] <- 0
ind <- X[, "admit:gender"]
```

Now, fit the new model

```r
fit <- glm(count~ gender:admit:ind + dept*admit + dept*gender, data=table.7.7, family=poisson, link="log")
```

Degrees of Freedom: 24 Total; 5 Residual
Residual Deviance: 2.681497

To obtain fitted values in an array, use the following command (notice dept. A):

```r
array(fitted(fit), dim=c(2,2,6), dimnames=list(c("yes","no"), c("male","female"), LETTERS[1:6]))
```

, , A
male female
yes 512 89
no 313 19

, , B
male female
yes 354.188 15.811966
no 205.812 9.188034

, , C
male female
yes 113.9978 208.0022
no 211.0022 384.9978

, , D
male female
yes 141.6326 127.3674
no 275.3674 247.6326

, , E
male female
yes 48.07705 98.92295
no 142.92295 294.07705

, , F
male female
yes 24.03081 21.96919
no 348.96919 319.03081
The same analysis can be done using the suggestion in section 7.3.4 (p.228) with structural zeroes. 
\texttt{loglin} (and \texttt{loglm}) will handle structural zeroes if you specify initial values of 0 for the appropriate cells and 1's for the remaining cells.

Create an array out of table.7.7
\begin{verbatim}
\texttt{table.7.17<-design.table(table.7.7)}
\end{verbatim}

Repeat the (AD,DG) fit:
\begin{verbatim}
\texttt{loglin(table.7.17,margin=list(c(1,3),c(2,3)),param=T,fit=T)}
\end{verbatim}

Now add the structural zeros and fit equation (7.7). Notice we start the subtable with structural zeroes at zero. The subtable values remain at zero through the fitting process.

\begin{verbatim}
\texttt{table.7.17[,1]<-0}
\texttt{loglin(table.7.17,margin=list(c(1,3),c(2,3)),start=c(rep(0,4),rep(1,20)),fit=T)}
\end{verbatim}

2 iterations: deviation 0

\begin{verbatim}
$\text{lrt:}
[1] 2.681459

$\text{pearson:}
[1] 2.6904

$\text{df:}
[1] 6 # Ripley (1997) warns that using loglin with structural zeroes sometimes gives
incorrect df, as here

$\text{margin:}
$\text{margin[[1]]:}
[1] "admit" "dept"

$\text{margin[[2]]:}
[1] "gender" "dept"

$\text{fit:}

\begin{verbatim}
, , A
  male female
   yes  0   0
   no  0   0

, , B
  male female
   yes 354.188 15.811966
   no 205.812  9.188034

, , C
  male female
   yes 113.9978 208.0022
   no 211.0022 384.9978

, , D
  male female
   yes 141.6326 127.3674
   no 275.3674 247.6326

, , E
  male female
   yes  48.07705 98.922946
   no 142.92294 294.07706
\end{verbatim}

male  female
yes  24.03081  21.96919
no   348.96918 319.03082

Compare the fit with the above example.

C. Cochran-Mantel-Haenszel Test

To demonstrate this test, I use the data from Table 7.9 (p. 233) and the function `mantelhaen.test`.

```r
table.7.9 <- array(c(0, 0, 6, 5, 3, 0, 3, 6, 6, 2, 0, 4, 5, 6, 1, 0, 2, 5, 0, 0),
                  dim = c(2, 2, 5),
                  dimnames = list(delay = c("none", "some"),
                                 resp = c("cured", "died"),
                                 pen.level = c(1/8, 1/4, 1/2, 1, 4)))
mantelhaen.test(table.7.9)  # this takes in variables from a data frame too
```

Warning messages:

Warning in mantelhaen.test(table.7.9): Columns in some subtables x[,k] contain only zeroes.

Mantel-Haenszel chi-square test with continuity correction
data:  table.7.9
Mantel-Haenszel chi-square = 3.9286, df = 1, p-value = 0.0475

Table 7.10 (columns 2 & 3)

For column 2: extract the LRT statistic for model (CD, PD, PC)
```r
loglin(table.7.9, margin = list(c(1, 2), c(2, 3), c(1, 3)))$lrt
```

[1] 7.501059

For column 3: model (PD, PC)
```r
loglin(table.7.9, margin = list(c(2, 3), c(1, 3)))$lrt
```

[1] 14.29378

Mantel-Haenszel odds ratios can be obtained with the help of the following function, `OR.MH`, applied to an array:

```r
OR.MH <- function(x) {
    n11k <- x[1, 1,]
    n22k <- x[2, 2,]
    n12k <- x[1, 2,]
    n21k <- x[2, 1,]
    nk <- n11k + n22k + n12k + n21k
    sum(n11k * n22k / nk) / sum(n12k * n21k / nk)
}
```

Applied to Table 7.9, the function `OR.MH` gives the common log odds ratio

```r
log(OR.MH(table.7.9))
```

[1] 1.94591

Standard errors can be computed using the same techniques.
D. Sample size and power calculations

Frank Harrell's Hmisc library has many functions for power and sample size calculations. In particular, \texttt{bpower} calculates the formula on page 240. Here it is applied to the example at the end of p. 240.

\begin{verbatim}
\texttt{bpower(p1=.63,p2=.57,n=50)}  \# .63-.57 = .06 absolute difference
  \texttt{Power}
  \texttt{0.07128458}

\texttt{bpower(p1=.63,p2=.57,n=200)}
  \texttt{Power}
  \texttt{0.1388583}
\end{verbatim}

Power for a chi-squared test can be computed using the following function:

\begin{verbatim}
\texttt{chipower.f<-function(p,pm,n,alpha,df)}
  \texttt{(nc<-n*my.pearson.x2(observed=p,expected=pm)$X2)}
  \texttt{1-pchisq(qchisq(1-alpha,df),df=df,ncp=nc)}
\end{verbatim}

I demonstrate its use with the example at the end of p. 241.

\begin{verbatim}
\texttt{fit<- loglin(matrix(c(.315,.185,.285,.215),byrow=T,ncol=2),margin=list(1,2),fit=T)}
\texttt{chipower.f(p=matrix(c(.315,.185,.285,.215),byrow=T,ncol=2), pm=fit$fit,}
  \texttt{n=200, alpha=.05, df=fit$df)}
\end{verbatim}

\begin{verbatim}
[1] 0.1393477
\end{verbatim}
Chapter 8 - Agresti

A. Uniform Association model

I will use the Job satisfaction data (table.2.4) for this section.

<table>
<thead>
<tr>
<th>income</th>
<th>jobsat</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;6000</td>
<td>very dissat</td>
<td>20</td>
</tr>
<tr>
<td>upto15000</td>
<td>very dissat</td>
<td>22</td>
</tr>
<tr>
<td>upto25000</td>
<td>very dissat</td>
<td>13</td>
</tr>
<tr>
<td>&gt;25000</td>
<td>very dissat</td>
<td>7</td>
</tr>
<tr>
<td>&lt;6000</td>
<td>little</td>
<td>24</td>
</tr>
<tr>
<td>upto15000</td>
<td>little</td>
<td>38</td>
</tr>
<tr>
<td>upto25000</td>
<td>little</td>
<td>28</td>
</tr>
<tr>
<td>&gt;25000</td>
<td>little</td>
<td>18</td>
</tr>
<tr>
<td>&lt;6000</td>
<td>moderate</td>
<td>80</td>
</tr>
<tr>
<td>upto15000</td>
<td>moderate</td>
<td>104</td>
</tr>
<tr>
<td>upto25000</td>
<td>moderate</td>
<td>81</td>
</tr>
<tr>
<td>&gt;25000</td>
<td>moderate</td>
<td>54</td>
</tr>
<tr>
<td>&lt;6000</td>
<td>very sat</td>
<td>82</td>
</tr>
<tr>
<td>upto15000</td>
<td>very sat</td>
<td>125</td>
</tr>
<tr>
<td>upto25000</td>
<td>very sat</td>
<td>113</td>
</tr>
<tr>
<td>&gt;25000</td>
<td>very sat</td>
<td>92</td>
</tr>
</tbody>
</table>

I add equal interval scores for income and jobsat to the table using the function codes to get the scores:

```
attach(table.2.4)
table.2.4.new<-cbind.data.frame(table.2.4,c.income=codes(ordered(income)),
                               c.jobsat=codes(ordered(jobsat)) )
```

I fit the Uniform Association model using scores for the variables only in the interaction. Note the use of ':' in the interaction term.

```
fit<-glm(count ~ income + jobsat + c.income:c.jobsat, data=table.2.4.new,
          family=poisson,link="log")
```

Coefficients:

```
(Intercept)    income1    income2    income3    jobsat1    jobsat2    jobsat3
3.025165 -0.00526504 -0.1905522 -0.2670179 0.1537626 0.325433 0.1581947
```

```
c.income:c.jobsat
0.1119394
```

Degrees of Freedom: 16 Total; 8 Residual
Residual Deviance: 2.385921

Notice the coefficient on c.income:c.jobsat. This is Agresti’s beta-hat.

To obtain a test of the association parameter, beta, use the `summary` command:

```
summary(fit)$coef["c.income:c.jobsat",]
```

```
         Value  Std. Error   t value
0.1119394   0.03640746  3.074628
```
Obtain the fitted values for the uniform association model using:

```r
matrix(fitted(fit),byrow=F,ncol=4,dimnames=list(Income=c(levels(income)),JobSat=c(leve
lons(jobsat))))
```

<table>
<thead>
<tr>
<th>very dissat</th>
<th>little</th>
<th>moderate</th>
<th>very sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;6000</td>
<td>19.347234</td>
<td>29.42999</td>
<td>74.92518</td>
</tr>
<tr>
<td>upto15000</td>
<td>21.412156</td>
<td>36.42892</td>
<td>103.72863</td>
</tr>
<tr>
<td>upto25000</td>
<td>13.592345</td>
<td>25.86395</td>
<td>82.36865</td>
</tr>
<tr>
<td>&gt;25000</td>
<td>7.648265</td>
<td>16.27714</td>
<td>57.97755</td>
</tr>
</tbody>
</table>

Now, Agresti fits the Independence model and compares it to the Uniform Association model.

Fit the Independence model:
```
fit2<-glm(count ~ income + jobsat, data=table.2.4.new, family=poisson,link="log")
```

Compare the two models (Uniform Association v. Independence):
```
anova(fit2,fit)
```

Analysis of Deviance Table

<table>
<thead>
<tr>
<th>Terms Resid. Df Resid. Dev</th>
<th>Test Df</th>
<th>Pr(Chi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>income + jobsat</td>
<td>9</td>
<td>12.03686</td>
</tr>
<tr>
<td>+c.income:c.jobsat</td>
<td>8</td>
<td>2.38592</td>
</tr>
</tbody>
</table>

The deviance above is G-squared for the Independence model given uniform association holds.

**B. Row Effects model - (same effect for an entire row)**

Agresti fits a row effects model to the data in Table 8.4 - p. 273.

```
table.8.4<-data.frame(expand.grid(Affil=c("Democrat","Independent","Republican"), Ideology=c("Liberal","Moderate","Conservative")), c.Ideo=codes(ordered(rep(c(1,2,3),c(3,3,3)))))
count=c(143,119,15,156,210,72,100,141,127))
attach(table.8.4)
```

To fit Row Effects model, type:
```
fit<-glm(count~Affil+Ideology+c.Ideo:Affil,family=poisson,link="log",data=table.8.4)
```

```
Degrees of Freedom: 9 Total; 2 Residual
Residual Deviance: 2.814931
```

To obtain fitted values for the Row Effects model, type:
```
matrix(fitted(fit),byrow=F,ncol=3,dimnames=list(Affiliation=c(levels(Affil)),Ideolog y=c(levels(Ideology))))
```

<table>
<thead>
<tr>
<th>Liberal</th>
<th>Moderate</th>
<th>Conservative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Democrat</td>
<td>136.63414</td>
<td>168.73171</td>
</tr>
<tr>
<td>Independent</td>
<td>123.79454</td>
<td>200.41091</td>
</tr>
<tr>
<td>Republican</td>
<td>16.57131</td>
<td>68.85738</td>
</tr>
</tbody>
</table>

To obtain the row effects themselves, we use the interaction parameter estimates.
res<-coef(fit)

(Intercept) Affil1 Affil2 Ideology1 Ideology2 AffilDemocratc.Ideo
6.057586 -0.1847135 -1.046094 0.7121822 0.4455428 -1.213361
AffilIndependentc.Ideo
-0.9426178

mu1-mu2:
temp<-res["AffilDemocratc.Ideo"]-res["AffilIndependentc.Ideo"]

mu2:
mu2<-(1/3)*(res["AffilIndependentc.Ideo"]-temp)

Getting the ASE is then also a matter of manipulation.

res<-summary(fit)$coef

<table>
<thead>
<tr>
<th>Term</th>
<th>Value</th>
<th>Std. Error</th>
<th>t value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>6.057586</td>
<td>0.14456061</td>
<td>41.903433</td>
</tr>
<tr>
<td>Affil1</td>
<td>-0.184714</td>
<td>0.09528375</td>
<td>-1.938563</td>
</tr>
<tr>
<td>Affil2</td>
<td>-1.046094</td>
<td>0.10073181</td>
<td>-10.38494</td>
</tr>
<tr>
<td>Ideology1</td>
<td>0.7121822</td>
<td>0.07042806</td>
<td>10.112194</td>
</tr>
<tr>
<td>Ideology2</td>
<td>0.4455428</td>
<td>0.05381862</td>
<td>8.278601</td>
</tr>
<tr>
<td>AffilDemocratc.Ideo</td>
<td>-1.2133612</td>
<td>0.13041946</td>
<td>-9.303529</td>
</tr>
<tr>
<td>AffilIndependentc.Ideo</td>
<td>-0.9426178</td>
<td>0.12602316</td>
<td>-7.479719</td>
</tr>
</tbody>
</table>

SE of mu2:

SE2<-res["AffilIndependentc.Ideo","Std. Error"]
SE1<-res["AffilDemocratc.Ideo","Std. Error"]
res<-summary(fit)$cor[6,7]*SE1*SE2
sqrt( ( (4/9)*(SE2^2)+(1/9)*SE1^2 ) - (4/9)*res )

[1] 0.05885542

To compare independence and row effects models, use anova:

fit2<-glm(count~Affil+Ideology,family=poisson,link="log",data=table.8.4)
anova(fit2,fit)

Analysis of Deviance Table

Response: count

<table>
<thead>
<tr>
<th>Terms Resid. Df Resid. Dev</th>
<th>Test Df</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deviance</td>
<td>1</td>
</tr>
<tr>
<td>Affil + Ideology</td>
<td>4</td>
</tr>
<tr>
<td>Affil + Ideology + c.Ideo:Affil</td>
<td>2</td>
</tr>
</tbody>
</table>

Now, I get predicted logits using fit$linear.predictors:

res<-matrix(fit$linear.predictors,byrow=F,ncol=3)
mat<-matrix(c(res[,2]-res[,1],res[,3]-res[,2]),byrow=T,ncol=3)

[,1]    [,2]     [,3]
[1,] 0.2110031 0.4817465 1.4243643
[2,] -0.5889149 -0.3181714 0.6244464

You can use the function matplot to create Figure 8.2 (p. 274).
C. Modeling three-factor interaction

Agresti uses the data in Table 8.10 p. 280 (effect of smoking) to illustrate modeling a three-factor interaction.

```r
table.8.10<-expand.grid(Status=c("Never","Former","Current"),
Breath=c("Normal","Borderline","Abnormal"), Age=c("< 40","40-50"))
table.8.10<-cbind.data.frame(table.8.10,count=c(577,192,682,27,20,46,7,3,11,164,
145,245,4,15,47,0,7,27))
attach(table.8.10)

I fit model (8.16) to Table 8.10 using:

```r
c.Age:c.Status, data=table.8.10, family=poisson, link="log")
```n
Degrees of Freedom: 18 Total; 7 Residual
Residual Deviance: 10.81581

To recreate Table 8.11, use the following set of commands:

```r
res<-summary(fit,cor=F)$coefficients

Coefficients:           Value Std. Error    t value
(Intercept)  4.69087634 1.15591459  4.0581514
Breath1 -0.98551429 0.28466131 -3.4620591
Breath2 -0.57776113 0.28454270 -2.0304901
Status1 -0.50384721 0.05397706 -9.3344686
Status2 -0.03229618 0.04871983 -0.6628959
Age -0.58570923 0.08588420 -6.8197550
Status1Age  0.08054713 0.05378014  1.4977114
Status2Age -0.20715182 0.04848249 -4.2727143
c.Status:c.Breath -0.54788559 0.22174479 -2.4708583
c.Age:c.Breath -1.01492541 0.43958629 -2.3088195
c.Age:c.Status  0.66312565 0.16467956  4.0267635

The coefficients from the fitted model (fit as stated in the command) do not correspond exactly to those in Agresti. Instead they are linear combinations of them as follows:

```r
beta1<-res["c.Status:c.Breath","Value"]
smoke<-beta1+beta3
Age<-beta2 + 2*beta3
smokeXage<-beta3

2*summary(fit)$cor[9,11]* (res["c.Status:c.Breath","Std. Error"]*
)

SEAge <- sqrt( (res["c.Age:c.Breath","Std. Error"])^2 +
2*2*summary(fit)$cor[10,11]* (res["c.Age:c.Breath","Std. Error"]*
)

matrix(c(smoke,Age,smokeXage,SEsmoke,SEAge,res["c.Age:c.Breath:c.Status","Std.
Error"]),byrow=F,ncol=2, dimnames=list(Parameter=c("Smoking","Age","Smoking x
age"),c("Estimate","SE")))
```
D. Uniform interaction model

The following command fits a uniform interaction model to the data in Table 8.10:

```r
fit2<-glm(count~(Status+Age+Breath)^2 + c.Breath:\c.Age:\c.Status, data=table.8.10,
          family=poisson, link="log")
```

Degrees of Freedom: 18 Total; 3 Residual
Residual Deviance: 2.74473

```r
summary(fit2,cor=F)$coefficients["c.Breath:\c.Age:\c.Status",]
```

<table>
<thead>
<tr>
<th>Value</th>
<th>Std. Error</th>
<th>t value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.831125</td>
<td>0.1945378</td>
<td>4.272242</td>
</tr>
</tbody>
</table>

E. Mantel Score Test of Conditional Independence

To illustrate Mantel's Score Test, Agresti uses the data in Table 8.13.

```r
dose<-(c(0,5,30,75,150,300))
survive<-c(0,1)
nijk<-array(c(0,7,3,1,4,11,5015,10752,2989,694,418,387,5,4,6,1,3,6,5973,11811,
              2620,771,792,820,2,8,3,1,3,7,5669,10828,2798,797,596,624,3,19,4,2,1,10,6158,12645
              ,3566,972,694,608,3,7,3,2,2,6,3695,9053,2415,555,393,289),dim=c(6,2,5))
```

I transpose each stratum (so that they look like Table 8.13)

```r
nijk<-array(apply(nijk,3,t),dim=c(2,6,5))
```

Now, get the marginal sums:

```r
n.jk<-array(apply(nijk,3,rowSums),c(2,1,5))
ni.k<-array(apply(nijk,3,colSums),c(1,6,5))
n..k<-apply(nijk,3,sum)
```

Get intermediate quantities in order to compute equation (8.17):

```r
u.k<-apply(ni.k,3,function(x,u){u%*%t(x)},u=dose)
v.k<-apply(n.jk,3,function(x,v){v%*%x},v=survive)
E.k<-(u.k)*(v.k)/n..k
u2.k<-apply(ni.k,3,function(x,u){(u^2)%*%t(x)},u=dose)
v2.k<-apply(n.jk,3,function(x,v){(v^2)%*%x},v=survive)
var.k<-(1/n..k)*((u2.k-(u.k)^2)/n..k)*((v2.k-(v.k)^2)/n..k)
grid<as.matrix(survive)%*%dose
uv.k<-colSums(apply(nijk,3,function(x,uv){x*uv},uv=grid))
```

Equation (8.17):

```r
((sum(uv.k-E.k))^2)/sum(var.k)
```

```
[1] 397.0276
```

The answer should be 426.3. So, there is an error somewhere.
F. Correspondence Analysis

The mutliv library contains a function, ca, which performs correspondence analysis. To add the library, type

```r
library(multiv)
```

I will use the data in Table 8.14 to follow Agresti’s example in Section 8.5.4.

```r
table.8.14<-data.frame(expand.grid(MH=c("well", "mild", "moderate", "impaired"),
                                   SES=LETTERS[1:6]),
                      count=c(64, 94, 58, 46, 57, 94, 54, 40, 57, 105, 65, 60, 71, 141, 77, 94, 36, 97, 54, 78, 21, 71, 54, 71))
attach(table.8.14)
MH SES count
1 well A 64
2 mild A 94
3 moderate A 58
4 impaired A 46
5 well B 57
6 mild B 94
7 moderate B 54
8 impaired B 40
...
```

I now put the data frame above into an array and transpose it:

```r
temp<-t(design.table(table.8.14))
```

To do the correspondence analysis, I use the `ca` function:

```r
fit<-ca(temp, nf=3)
```

Now, I extract the squared correlations for the three dimensions

```r
fit$evals
[1] 0.0260861516 0.0013649208 0.0002818179
```

To get the row and column projections (i.e., Table 8.15), use the following columns:

**column projections**

```r
fit$cproj
"Factor1"  "Factor2"  "Factor3"
[1,] 0.26031950 0.01059587 0.022159025
[2,] 0.02943672 0.02487361 -0.018911108
[3,] -0.01373850 -0.06926306 -0.004133451
[4,] -0.23754509 0.01763202 0.015694946
```

**row projections**

```r
fit$rproj
"Factor1"  "Factor2"  "Factor3"
[1,] 0.18204767 -0.01965235 0.027711924
[2,] 0.18603650 -0.01028790 -0.026940031
[3,] 0.06004478 -0.02157942 -0.010481284
[4,] -0.01218749 0.04121657 0.009749501
[5,] -0.16456713 0.04381708 -0.008189335
[6,] -0.28677478 -0.06237175 0.003613453
```

To plot the first and second factors (Figure 8.3), use
plot(fit$rproj[,1], fit$rproj[,2], type="n", ylim=c(-.1,.1), xlim=c(-.3,.3), xlab="", ylab="", axes=F)
text(fit$rproj[,1], fit$rproj[,2], labels=dimnames(temp)$SES)
text(fit$cproj[,1], fit$cproj[,2], labels=dimnames(temp)$MH)

One can place additional axes through x=0 and y=0 using a modified version of the function plaxes in library multiv:

my.plaxes.f(fit$rproj[,1], fit$rproj[,2], size=.15)

my.plaxes.f<-
function(a, b, size = 0.1)
{
  arrows(min(a), 0, max(a), 0, size = size)
  arrows(0, min(b), 0, max(b), size = size)
}

Note that the MASS library also has a multivariate CA function (as well as a univariate one). It is called mca. I had to alter it to fit three factors. With the above dataset, the new function, my.mca gives

counts<-c(64, 94, 58, 46, 57, 46, 54, 40, 57, 105, 65, 60, 71, 141, 77, 94, 36, 97, 54, 78, 21, 71, 54, 71)
table.8.14<-apply(table.8.14[,c(1,2)],2,function(x){rep(x,counts)})
fit<-my.mca(as.data.frame(table.8.14),nf=3)

MASS has a univariate CA, which estimates a single factor model. Using it on Table 8.14,

temp<-t(design.table(table.8.14))  # transposed the default

fit<-corresp(temp)

$cor:
[1] 0.1615119

$rscore:
A         B          C          D        E        F
-1.127146 -1.151843 -0.3717666 0.07545902 1.018916 1.775564

$cscore:
well       mild   moderate impaired
-1.611765 -0.1822571 0.08506316 1.470761

The rscore and cscore match the scores given on p. 293, for a one-dimensional model.

For canonical correlation model, note that Splus has a cancor function
A. Generalized logit model for multiple nominal response categories

To fit a generalized logit model for multiple nominal response categories, I will use Agresti's use of Table 9.1 (Alligator food choice). To set up the data, type:

```r
food.labs<-c("fish","invert","rep","bird","other")
size.labs<-c("<2.3",">2.3")
gender.labs<-c("m","f")
lake.labs<-c("hancock","oklawaha","trafford","george")

table.9.1<-
  expand.grid(food=food.labs, size=size.labs, gender=gender.labs, lake=lake.labs)
temp<-(7,1,0,0,5,4,0,0,1,2,16,3,2,2,3,3,0,1,2,3,2,2,0,0,1,13,7,6,0,0,3,9,1,0,2,0,1,0
  ,1,0,3,7,1,0,1,8,6,6,3,5,2,4,1,1,4,0,1,0,0,0,13,10,0,2,2,9,0,0,1,2,3,9,1,0,1,8,1,0,0
  ,1)
table.9.1<-apply(table.9.1,2,function(x){rep(x,temp)})

First, I collapse over gender (as on page 310):

table91.nogender<-table(category(table.9.1[,1],levels=food.labs),
  category(table.9.1[,2],levels=size.labs),category(table.9.1[,4],levels=lake.labs)
)

(The reason for using the category function above is to give the labels in an order that is not alphabetical)

Now, I first fit the model (FS, FL, LS) (see Table.9.3):
To fit the model as a generalized logit model, we can use B. Ripley’s `multinom` function in the nnet library. This function fits multinomial logit models with nominal response categories.

To use the function `multinom`, we transform the format of the data, collapsing over Gender. It is important that the level “fish” be coded as 1. Thus, I use an ordered factor instead of a factor. This will not affect the analysis.

```r
library(nnet)
```
table.9.1<-data.frame(food=codes(ordered(table.9.1[,"food"],lev=food.labs)),
                      table.9.1[,c(2,4)])

Now, set the contrasts option so that coefficient estimates sum to zero:

options(contrasts=c("contr.sum","contr.poly"))

Fit the model:

fit<-multinom(food~lake+size,data=table.9.1)

summary(fit,cor=F)$coefficients

(Intercept)      lake1      lake2      lake3       size
2  -0.7197536 -0.1001507 -1.7585930  0.8370343  0.7290775
3  -1.8309445 -1.6592416 -0.4164767  0.7996359 -0.1755950
4  -2.1260480 -0.2824301  0.4127130 -0.9356402 -0.3154096
5  -1.1514310 -0.5870533  0.2391164 -0.5814067  0.1657686

Note that the lakes are numbered in a different order than that in Table 9.4.

Conditional logit models can be handled by multinom or by the function coxph, for Cox proportional hazards models. Also, all of the models discussed in section 9.2 can be handled by multinom.

B. Multinomial Logits for Ordinal Responses (Continuation-Ratio logit model)

For this section, I will use the set of libraries Design and Hmisc by F. Harrell. In particular, the library fits forward continuation-ratio models and proportional odds models, and many other models which use multinomial ordinal responses.

Continuation-Ratio logit model

For this model, I use the data in Table 9.7 (p. 320), following Agresti's example. A continuation-ratio model is fit using cr.setup and lrm from the Design library.

First, it is important to add the libraries in this order:

library(Hmisc,T)
library(Design,T)

Now, set up the data:

y<-ordered(c("non-live","malformed","normal"),levels=c("non-
             live","malformed","normal"))
y<-rep(rep(y,5),c(15,1,281,17,0,225,22,7,283,38,59,202,144,132,9))
x<-rep(c(0,62.5,125,250,500),c(15+1+281,17+0+225,22+7+283,38+59+202,144+132+9))

table.9.7<-table(x,y)

      non-live  malformed normal
0          15         1     281
62.5       17         0     225
125        22         7     283
250        38         59    202
500        144        132     9

cr.setup will transform the response variable so that it can be used for a continuation-ratio model. In particular, it will create the new variables, y and cohort. The newly created variables are longer (have
more observations) than the old response variable. Cohort defines the denominator categories for each logit (see equation (9.8) in Agresti). \( y \) is the transformed response variable taking on values 0 or 1 depending on whether the observation is a success or not within its cohort.

For example, for the data in table 9.7, there are two cohorts. The first cohort (\( j=1 \)) is the set of all three categories: non-live, malformation, and normal, where an observation is considered a success if it falls in non-live versus either of the other two categories. The second cohort (\( j=2 \)) is the set of the last two categories, malformed and normal, where an observation is considered a success if it falls in malformed over normal.

Here is how to fit the model:

First set up the response:

```r
u<-cr.setup(y)
y.u<-u$y
x.u<-x[u$subs]  # this ensures that the covariate is the correct length
```

I will do separate fits first before showing how to fit both models (\( j=1 \) and \( j=2 \)) together. After the fit of the \( j=1 \) model, I will discuss some of the output from `lrm` and compare it to Agresti’s results and to results using `glm`.

To fit the \( j=1 \) model:

```r
fit<-lrm(y.u[u$cohort=="all"]~x.u[u$cohort=="all"])
```

Logistic Regression Model

Frequencies of Responses

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1199</td>
</tr>
<tr>
<td>1</td>
<td>236</td>
</tr>
</tbody>
</table>

Obs Max Deriv Model L.R. d.f. P C Dxy Gamma Tau-a R2 Brier

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1435</td>
<td>2e-009</td>
<td>253.33</td>
<td>1</td>
<td>0</td>
<td>0.781</td>
<td>0.561</td>
<td>0.667</td>
</tr>
</tbody>
</table>

Coef S.E. Wald Z P

Intercept -3.247934 0.1576602 -20.6 0
x.u 0.006389 0.0004348 14.7 0

The "Model L.R." given above is supposed to be the model likelihood ratio chisquare according to Harrell (1998). However, if you examine what `glm` gives and what Agresti says is the model deviance (5.77), you can see that model L.R. is actually equal to -2LogLH(model with intercept + x). The d.f. above gives the number of d.f. used with the addition of x in the model (i.e., 1). What Agresti gives (p. 321) is the model residual deviance. That is, he gives -2LogLH(model with intercept only)-(-2LogLH(model with intercept + x)). His d.f. correspond to the resulting d.f. when going from an intercept model (df=4) to a model with x (df=3). These are the df and LR statistic given directly by `glm` when modeling a linear logit, as shown later.

To fit the \( j=2 \) model:

```r
fit<-lrm(y.u[u$cohort=="y>=malformed"]~x.u[u$cohort=="y>=malformed"])
```

Logistic Regression Model

Frequencies of Responses

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1000</td>
</tr>
<tr>
<td>1</td>
<td>199</td>
</tr>
</tbody>
</table>
Obs Max Deriv Model L.R. d.f. P     C   Dxy Gamma Tau-a    R2 Brier
1199  5e-006     646.52    1 0 0.948 0.895 0.97 0.248 0.703 0.052

Coef      S.E. Wald Z P
Intercept -5.70190 0.332249 -17.16 0
x.u  0.01737 0.001227 14.16 0

See Harrell (1998) or the help file for the library for a full discussion of the other statistics produced by lrm.

To fit both models together (j=1 and j=2), fit an interaction term, as in the following.

```r
fit<-lrm(y~cohort*x)
```

Logistic Regression Model

Frequencies of Responses
0   1 2199 435

Obs Max Deriv Model L.R. d.f. P     C   Dxy Gamma Tau-a    R2 Brier
2634  4e-006     899.86    3 0 0.884 0.768 0.819 0.212 0.489 0.083

Coef      S.E. Wald Z P
cohort=y>=malformed 2.453968 0.3677581 -6.67 0
x  0.006389 0.0004348 14.70 0
cohort=y>=malformed * x  0.010986 0.0013020   8.44 0

When y>=malformed (j=2), the linear logit is -5.70 + .017x.

When y=all (j=1), the linear logit is -3.247 + .0064x.

Notice that the values for model L.R.in the individual model sum to the model L.R. for the interaction model above. However, the d.f. do not add.

Odds Ratios

To get selected odds ratios for the j=2 model, first issue the datadist command and reissue the lrm call, as follows:

```r
x.u<-x.u[u$cohort=="y>=malformed"]
dd<-datadist(x.u)
options(datadist='dd')
fit<-lrm(y.u[u$cohort=="y>=malformed"]~x.u)
```

Using summary(fit) will give odds ratios comparing the default levels of x.u (the lowest and highest nonzero values)

```r
summary(fit)
```

<table>
<thead>
<tr>
<th>Factor</th>
<th>Low</th>
<th>High</th>
<th>Diff.</th>
<th>Effect</th>
<th>S.E.</th>
<th>Lower 0.95</th>
<th>Upper 0.95</th>
</tr>
</thead>
<tbody>
<tr>
<td>x.u</td>
<td>62.5</td>
<td>250</td>
<td>187.5</td>
<td>3.26</td>
<td>0.23</td>
<td>2.81</td>
<td>3.71</td>
</tr>
<tr>
<td>Odds Ratio</td>
<td>62.5</td>
<td>250</td>
<td>187.5</td>
<td>25.99</td>
<td>NA</td>
<td>16.56</td>
<td>40.80</td>
</tr>
</tbody>
</table>

Thus, given that a fetus was alive, the estimated odds of it being malformed versus normal is 26 times higher when a mouse is exposed to 500 mg/kg per day of the toxic substance than when it is exposed to 62.5 mg/kg per day. The NA for SE is apparently not a mistake. Also the value 3.26 = (.0174)*(250-62.5) is the log odds.
To get an odds ratio comparing specific levels of x, for example comparing levels x=125 and x=250:

```
summary(fit, x=c(125, 250))
```

<table>
<thead>
<tr>
<th>Effects</th>
<th>Response : y.u[u$cohort == &quot;y&gt;=malformed&quot;]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor Low High Diff. Effect S.E. Lower 0.95 Upper 0.95</td>
<td></td>
</tr>
<tr>
<td>x.u</td>
<td>125 250 125 2.17 0.15 1.87 2.47</td>
</tr>
<tr>
<td>Odds Ratio 125 250 125 8.77 NA 6.50 11.85</td>
<td></td>
</tr>
</tbody>
</table>

Or, levels x=250 and x=500

```
summary(fit, x=c(250, 500))
```

<table>
<thead>
<tr>
<th>Effects</th>
<th>Response : y.u[u$cohort == &quot;y&gt;=malformed&quot;]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor Low High Diff. Effect S.E. Lower 0.95 Upper 0.95</td>
<td></td>
</tr>
<tr>
<td>x.u</td>
<td>250 500 250 4.34 0.31 3.74 4.95</td>
</tr>
<tr>
<td>Odds Ratio 250 500 250 76.99 NA 42.20 140.48</td>
<td></td>
</tr>
</tbody>
</table>

Using `glm`

We can instead estimate the continuation ratio model using `glm`, as we did the linear logit model in chapter 4.

First detach Design and Hmisc, or else `glm` will not work right:

```
detach("C:\Program Files\splus45\library\Design\_Data")
detach("C:\Program Files\splus45\library\Hmisc\_Data")
```

Now set x (the covariate) and the weights for the first linear logit model

```
x<-c(0, 62.5, 125, 250, 500)
n1<-rowSums(table.9.7) # use the whole table
```

For model j=1, take the first column of table.9.7 as the “success”, as follows:

```
fit<-glm(table.9.7[,1]/n1~x, family=binomial,link=logit,weights=n1)
```

```
Coefficients:
(Intercept) x
-3.247934 0.006389069

Degrees of Freedom: 5 Total; 3 Residual
Residual Deviance: 5.777478
```

```
fit$null.deviance
[1] 259.1073
```

```
fit$null.deviance-fit$deviance # this is what lrm gave us as model L.R.
[1] 253.3298
```

```
summary(fit)

Null Deviance: 259.1073 on 4 degrees of freedom
Residual Deviance: 5.777478 on 3 degrees of freedom
```

The difference of the above sets of values gives Null Deviance-Residual Deviance = 253.3298 and df=1. These are the Model L.R. and Model df reported by `lrm`. 
For the $j=2$ model, take the second and third columns of table.9.7, and use the second column as the success:

$$n2<-\text{rowSums(table.9.7[,c(2,3)])}$$

$$\text{glm(table.9.7[,2]/n2~x, family=binomial, link=logit, weights=n2)}$$

Coefficients:

(Intercept)          x  
 -5.701891 0.01737464

Degrees of Freedom: 5 Total; 3 Residual
Residual Deviance: 6.060908

Even though \texttt{glm} can also fit proportional odds models, there are advantages to using \texttt{lrm} for these types of models because of the built-in features, like the odds ratios above. See Harrell(1998) or the web site http://hesweb1.med.virginia.edu/biostat (under Statistical Computing Tools) for more information.

As mentioned in Harrell (1998) as well as Agresti (chapter 9 notes), the continuation-ratio model is a discrete version of the Cox proportional hazards model. Thus, one could probably fit these models using either \texttt{coxph}, which comes with Splus, or \texttt{cph}, which is in the Design library.

### C. Cumulative Logit Models

For these models (proportional odds, uniform association for cumulative odds, row effects model for nominal explanatory variables), we will use the Design library again. To load the library, issue the following commands in the order specified.

```r
library(Hmisc, T)
library(Design, T)
```

#### 1. Proportional Odds Model

For this model I use the data in Table 9.8. To set it up, type

```r
mental.imp<-rep(c("well","mild","moderate","impaired"),c(12,12,7,9))
mental.imp<-ordered(mental.imp,levels=rev(c("well","mild","moderate","impaired")))
SES<--c(1,1,1,1,0,1,0,1,1,0,0,1,0,1,0,1,1,0,0,1,0,1,0,1,0,1,1,0,0,0,1,0,0,1,1,0,0,0,1,0,0,1,1,0,0,0,1,0,0,0,1,0,0,1,1,0,0,0,1,0,0,1,1,0,0,0,1,0,0,1,1,0,0)
LE<--c(1,9,4,3,2,0,1,3,3,7,1,2,5,6,3,1,8,2,5,5,9,3,3,1,0,4,3,9,6,4,3,8,2,7,5,4,4,8,8,9)
table.9.8<-data.frame(mental.imp=mental.imp,SES=SES,LE=LE)
```

(The minus signs in SES and life events are required to estimate the parameters for the main effects model on page 325).

\texttt{lrm} fits proportional odds models automatically if the response is ordinal:

```r
lrm(mental.imp~SES+LE, data=table.9.8)
```

Frequencies of Responses

<table>
<thead>
<tr>
<th>impaired</th>
<th>moderate</th>
<th>mild</th>
<th>well</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>7</td>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>
To add the interaction, we want to prevent the minuses from canceling out in the interaction between SES and LE. So, we can use the `I` function, which interprets its argument literally:
```
`lrm(mental.imp ~ SES + LE + I(-SES * LE), data = table.9.8)`
```

Logistic Regression Model

Frequencies of Responses

impaired moderate mild well
9 7 12 12

The last coefficient is the interaction term.

To get odds ratios for this last model (the interaction model), we have to explicitly give the interaction term a name in `datadist`, and we should also note that the levels of LE and SES are negated:
```
IA<-I(-SES*LE)  # name the interaction
dd<-datadist(SES=-SES,LE=-LE,IA=IA)  # register variables in datadist
options(datadist='dd')
fit<-lrm(mental.imp~SES+LE+IA)  # fit the model
summary(fit)
```

Effects

Response : mental.imp

```
Factor Low High Diff. Effect S.E. Lower 0.95 Upper 0.95
SES 0.00 1.00 1.00 -0.37 1.14 -2.60 1.86
Odds Ratio 0.00 1.00 1.00 0.69 NA 0.07 6.40
LE 2.00 6.25 4.25 1.79 0.79 0.23 3.34
Odds Ratio 2.00 6.25 4.25 5.97 NA 1.26 28.22
IA -4.25 0.00 4.25 -0.77 1.01 -2.76 1.21
Odds Ratio -4.25 0.00 4.25 0.46 NA 0.06 3.37
```

See the function `popower` in library Hmisc for computing power of proportional odds models.

2. Cumulative logit uniform association model

I use the data in Table 9.9 in Agresti and follow Section 9.4.5. To set up the data, type:
```
operation<-factor(c("A","B","C","D"))
operation<-rep(rep(operation,12),c(23,23,20,24,7,10,13,10,2,5,5,6,18,18,13,9,6,6,13,15,1,2,2,2,8,12,11,7,6,4,6,7,3,4,2,4,12,15,14,13,9,3,8,6,1,2,3,4))
```
To fit model (9.16) (uniform association model), I use scores (codes) for operation and exclude hospital:

\[ \text{lrm(dump}\sim I(-\text{codes(operation)}), \text{data=table.8.6)} \]

Logistic Regression Model

Frequencies of Responses

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>S</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>240</td>
<td>129</td>
<td>48</td>
</tr>
</tbody>
</table>

Obs Max Deriv Model L.R. d.f. P C Dxy Gamma Tau-a R2 Brier
417 4e-008 6.61 1 0.0102 0.564 0.128 0.17 0.072 0.019 0.241

<table>
<thead>
<tr>
<th>Coef</th>
<th>S.E.</th>
<th>Wald</th>
<th>Z</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>y&gt;=S</td>
<td>-0.8820 0.24966</td>
<td>-3.53</td>
<td>0.0004</td>
<td></td>
</tr>
<tr>
<td>y&gt;=M</td>
<td>-2.6362 0.28389</td>
<td>-9.29</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>operation</td>
<td>-0.2247 0.08798</td>
<td>2.55</td>
<td>0.0106</td>
<td></td>
</tr>
</tbody>
</table>

The model L.R. and df give the reduction in \( \chi^2 \) and in df from the fit model and from one assuming independence (i.e., it is a test that the coefficient on operation is zero).

To obtain the predicted logits, probabilities, and expected frequencies, we can use the linear.predictor attribute (or, predict.lrm function) to get linear predictors, and take the anti-logit to
get cumulative probabilities. However, you must note that \texttt{lrm} fits the probabilities, \( \Pr(y > j) \) (or, \( 1 - \Pr(y \leq j) \)), not \( \Pr(y \leq j) \). Thus, to reproduce Table 9.9, I type

```r
probs<-unique(exp(fit$linear.predictors)/(1+exp(fit$linear.predictors)))
```

```
[1] 0.3413441 0.3935044 0.4482114 0.5042011
```

cumprobs<-1-probs

```
[1] 0.6586559 0.6064956 0.5517886 0.4957989
```

```r
rowSums(table(operation,dump))*(cumprobs)
```

```
A  B  C  D
63.23096 63.07554 60.69675 53.05048
```
to get the first column of Table 9.9. The rest follow similarly.

### 3. Cumulative logit row effects model

Fitting the row effects is just a matter of adding hospital to the formula.

```r
fit<-lrm(dump~I(-codes(operation))+hospital,data=table.8.6)
```

Logistic Regression Model

Frequencies of Responses

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>S</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>240</td>
<td>129</td>
<td>48</td>
</tr>
</tbody>
</table>

Obs Max Deriv Model L.R. d.f.      P     C   Dxy Gamma Tau-a    R2 Brier

```
417 2e-007 9.16 4 0.0572 0.575 0.149 0.159 0.084 0.026 0.239
```

<table>
<thead>
<tr>
<th>Coef</th>
<th>S.E.</th>
<th>Wald</th>
<th>Z</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y &gt; S )</td>
<td>0.99551</td>
<td>0.28612</td>
<td>-3.48</td>
<td>0.00005</td>
</tr>
<tr>
<td>( y &gt; M )</td>
<td>-2.75744</td>
<td>0.31808</td>
<td>-8.67</td>
<td>0.00000</td>
</tr>
<tr>
<td>operation</td>
<td>-0.22666</td>
<td>0.08814</td>
<td>2.57</td>
<td>0.0101</td>
</tr>
<tr>
<td>hospital=H2</td>
<td>0.10383</td>
<td>0.25037</td>
<td>0.41</td>
<td>0.6783</td>
</tr>
<tr>
<td>hospital=H3</td>
<td>0.42884</td>
<td>0.28066</td>
<td>1.53</td>
<td>0.1265</td>
</tr>
<tr>
<td>hospital=H4</td>
<td>0.02693</td>
<td>0.26901</td>
<td>0.10</td>
<td>0.9202</td>
</tr>
</tbody>
</table>

Note that the model L.R. = 31.64-22.48 = 9.16, and d.f. = 30-26= 4

For an interaction model, just use * in the formula:

```r
lrm(dump~I(-codes(operation))*hospital,data=table.8.6)
```

### D. Cumulative Link Models

#### Proportional Hazards Models

Although we don't actually have the data in Table 9.10, we can substitute the percentages multiplied by 10 as estimates of the numbers in each category, for illustration purposes. Then, we can use lifelength as the ordered response, or time of death, in a Cox proportional hazards model.

To set up the data, type:

```r
lifelength<-.ordered(c("0=20","20-40","40-50","50-65","over 65"))
```
lifelength<-rep( rep(lifelength,4),c(24, 34, 175, 27, 29, 44, 163, 737))
race<-rep(race,2),c(24+34+175+27+29+44+163+737))
sex<-rep(c("male","female"),c(24+34+175+27+29+44+163+737))

To fit a coxph model, we need to set up a Surv object. The arguments to Surv are the time of death for each individual and whether each time was censored (the status). We will assume that all deaths indicated in Table 9.10 were deaths and censor the over-65 category. Thus, our status variable looks like:

status<-ifelse(codes(lifelength)<5,1,0)

To fit a Cox PH model, type:

fit<-coxph(Surv(codes(lifelength), status )~race+sex)

The coefficients are off somewhat because we didn't use the real data. But, they are roughly the same. (It is important to note that Agresti's differences of coefficients are equivalent to coxph's actual coefficients). Thus,

Pr(survive until lifelength category $j$ | male) =

Pr(survive until lifelength category $j$ | female)$^{exp(0.656)}$

as in Agresti, p. 333.

E. Regression Model for Ordered Responses

Agresti uses the Olive Preference data in Table 9.11 (p. 335) to illustrate regression on ordered responses, where the conditional mean is linearly related to the explanatory variables. I first tried both of the functions glm and aov, but can't reproduce the estimates in Agresti exactly. The fitting algorithm used for glm, however, is iteratively reweighted least squares, and not weighted least squares, as was used by Agresti. I also, then, compute the weighted least squares estimates of the regression coefficients using the development on p. 458-462 of Agresti. These estimates, of course, match those of Agresti.

The first analysis is via the functions glm and aov.

First, set up the data:

location<-factor(c("MW","NE","SW"))
 pref<-c(1,2.5,4.5,6,7,8.5)
 Urban<-factor(c("Rural","Urban"))
 table.9.11<-data.frame(expand.grid(pref=pref,location=location,Urban=Urban),
 count=c(20,15,12,17,16,28,18,17,18,18,6,25,12,9,23,21,19,30,30,22,21,17,8,12,23,
 18,20,18,10,15,11,9,26,19,17,24))
I set the contrasts option so that contrasts on the levels of a factor sum to zero.

```
options(contrasts=c("contr.sum","contr.poly"))
```

Using `glm` we get:

```
fit<-glm(pref~location+Urban,family=poisson(identity),weights=table.9.11$counts,
data=table.9.11)
fit$coefficients

(Intercept)  location1  location2    Urban
 4.97535  -0.3923726  -0.2691034  0.330591
```

```
predict(fit)[seq(along=predict(fit))%%6==1]
   1        7       13       19       25       31
 4.913568 5.036838  5.967417  4.252387  4.375656  5.306235
```

Using `aov` we get:

```
fit<-aov(pref ~ Urban + location, data = table.9.11, weights = counts)
fit$coefficients

(Intercept)       Urban location1     location2
 4.975370085  0.3146362183  -0.3715660835  -0.2757134231
```

```
predict(fit)[seq(along=predict(fit))%%6==1]
   1           7         13          19          25          31
 4.91844022 5.014292881  5.93728581  4.289167784  4.385020444  5.308013374
```

The two give similar results. Now, using the development on p. 458 ff., first I set up the data and calculate the sample proportions responding in each preference category by population (i.e., location x urbanization combination).

```
table.9.11<-matrix(c(20,18,12,30,23,11,15,17,9,22,18,9,12,18,23,21,20,26,17,18,21,
                17,18,19,16,6,19,8,10,17,28,25,30,12,15,24),
                nc=6, nr=6, byrow=F)
n<-rowSums(table.9.11)  # sample size for each population
p<-sweep(table.9.11,1,n,FUN="/")
p1<-p[1,]                # sample proportions for each of the six categories
p2<-p[2,]
p3<-p[3,]
p4<-p[4,]
p5<-p[5,]
p6<-p[6,]
J<-6  # Six response categories
I<-6  # Six populations
```

Now, I compute the block diagonal variance-covariance matrix, $V$, using $V_1$, $V_2$, $V_3$, $V_4$, $V_5$, $V_6$ (the population specific covariance matrices). In the for-loop below, the object `Vnames` stores the names of the objects $V_1$ through $V_6$, and the object `pnames` stores the names of the vectors of sample proportions for each group. First I calculate each $V_i$, then I calculate $V$.

```
Vnames<-numeric(length=I)
pnames<-Vnames
```
for(i in 1:I)
{
    Vnames[i] <- paste("V", i, collapse = " ", sep = "")
    pnames[i]<-paste("p",i,collapse=" ",sep="")
    V<-matrix(0,nc=J,nr=J)
    p<-eval(parse(text = pnames[i])) # Use the name ‘p’ for the current proportion
    diag(V)<-p*(1-p)
    p<-as.matrix(p)
    junk<-matrix(-kronecker(p,p),nc=J,nr=J,byrow=T)
    V[lower.tri(diag(J))]<-junk[lower.tri(junk)]
    V<-t(V)
    V[lower.tri(V)]<-junk[lower.tri(junk,diag=F)]
    assign(Vnames[i], matrix(V/n[i], ncol = J, byrow = T)) # assign V to
    Vnames[i]
}

I construct V using V1 through V6

zero<-matrix(0,J,J)
V<-rbind(
    cbind(V1,zero,zero,zero,zero,zero),
    cbind(zero,V2,zero,zero,zero,zero),
    cbind(zero,zero,V3,zero,zero,zero),
    cbind(zero,zero,zero,V4,zero,zero),
    cbind(zero,zero,zero,zero,V5,zero),
    cbind(zero,zero,zero,zero,zero,V6),
)

Now, the model for this example has form $F(\pi) = X\beta$, where $\pi$ is a 6 x 6 matrix of response probability distributions for each population, $\beta = (\alpha, \beta_1^U, \beta_1^L, \beta_2^L)^T$, and $F(\pi)$ is a 6 x 1 vector of the 6 response functions

$$F_1(\pi) = \alpha + \beta_1^U + \beta_1^L = V^T (\pi_{1(1)}, \ldots, \pi_{6(1)}) = \sum_{j=1}^{6} \frac{\exp(\alpha + \beta_1^U + \beta_1^L)}{1 + \exp(\alpha + \beta_1^U + \beta_1^L)}$$

$$
\vdots
$$

$$F_6(\pi) = \alpha - \beta_1^U - \beta_1^L - \beta_2^L = V^T (\pi_{1(2,3)}, \ldots, \pi_{6(2,3)}) = \sum_{j=1}^{6} \frac{\exp(\alpha - \beta_1^U - \beta_1^L - \beta_2^L)}{1 + \exp(\alpha - \beta_1^U - \beta_1^L - \beta_2^L)}$$

representing the 2 x 3 combinations of Location and Urbanization (see equation 9.18 in Agresti). The 6 x 1 vector V contains the scores for the preference categories.

Now, note that the matrix $Q = \frac{\partial F_i(\pi)}{\partial \pi_{jk}}$ on page 459 contains the 1 x 36 row vectors

$$\frac{\partial F_i(\pi)}{\partial \pi_{jk}} = (0_1, \ldots, 0_{(k-1)6}, V_1, \ldots, V_6, 0, \ldots, 0_{36}) ,$$

and the design matrix is
\[
X = \begin{bmatrix}
1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 \\
1 & 1 & -1 & -1 \\
1 & -1 & 1 & 0 \\
1 & -1 & 0 & 1 \\
1 & -1 & -1 & -1 
\end{bmatrix}
\]

All of the above is input into Splus via the following commands. For the \( F(\ ) \) functions, we use the sample proportions instead of the probabilities, \( \pi \).

```r
pref<-c(1,2.5,4.5,6,7,8.5) # the vector nu
Q<-rbind(
  c(pref,rep(0,30)),
  c(rep(0,6),pref,rep(0,24)),
  c(rep(0,12),pref,rep(0,18)),
  c(rep(0,18),pref,rep(0,12)),
  c(rep(0,24),pref,rep(0,6)),
  c(rep(0,30),pref)
)
VF<-Q%*%V%*%t(Q) # transformed covariance matrix (p. 460 in Agresti)

# Design matrix:
X<-cbind(rep(1,J),rep(c(1,-1),c(3,3)),rep(c(1,0,-1),2),rep(c(0,1,-1),2))

# Functions
Fp<-c(pref%*%p1,pref%*%p2,pref%*%p3,pref%*%p4,pref%*%p5,pref%*%p6)

Now, I estimate beta using the formula in section 13.3.2, the weighted least squares estimator.

```r
InvVF<-solve(VF)
Covb<-solve(t(X)%*%InvVF%*%X)
b<-as.numeric(Covb%*%t(X)%*%InvVF%*%Fp)
[1]  4.9680696  0.2965415 -0.3933309 -0.2661780
```

The asymptotic standard errors:

```r
sqrt(diag(Covb))
[1] 0.1011683 0.1005547 0.1448348 0.1467234
```

I compute a Wald statistic for entire model using the formula on p. 460 in Agresti.

```r
as.numeric(t(Fp-X%*%b)%*%InvVF%*%(Fp-X%*%b))
[1] 4.807433
```

To test \( H_0: \beta_1^L = \beta_2^L = \beta_3^L \), we can use the coefficient matrix

```r
C<-rbind(c(0,0,1,-1),c(0,0,1,-1)
and the Wald statistic
```

```r
t(C%*%b)%*%solve(C%*%solve(t(X)%*%solve(VF)%*%X)%*%t(C))%*%C%*%b
t(C%*%b)%*%solve(C%*%solve(t(X)%*%solve(VF)%*%X)%*%t(C))%*%C%*%b
```
Obtaining predicted mean response values is straightforward using the regression coefficients.

\[
X \, %*% \, b
\]

\[
\begin{bmatrix}
[1,] & 4.871280128 \\
[2,] & 4.998433076 \\
[3,] & 5.924119931 \\
[4,] & 4.278197184 \\
[5,] & 4.405350131 \\
[6,] & 5.331036986 \\
\end{bmatrix}
\]

F. (Supplement) Nonlinear Ordinal Models (Using libraries nolr and lrm)

The following supplement demonstrates the function \texttt{nolr}, available in Mathieson’s library of the same name. I use the synthetic data provided by Mathieson in the library and compare his analysis in the postscript file (included with the library) with something similar using \texttt{lrm}.

For an ordinal model, we might not want to assume that the log odds of \( \Pr(y \geq j | X) \) is a linear function of the vector of covariates, \( X \), as is done with the proportional odds models in section 9.4.1 in Agresti. If we do not have a specific form of function in mind, we can fit the model as a feed-forward layered neural network and estimate it using MLE. This can be done using library \texttt{nolr}.

A feed-forward neural network is (from Bishop, 1995) a general framework for representing nonlinear functional mappings between a set of input variables and a set of output variables. This is achieved by representing a nonlinear function of a multivariate vector in terms of compositions of nonlinear functions of the scalar elements in the vector. The nonlinear functions are called \textit{activation functions}.

A familiar example of a \textit{single-layered} neural network is logistic discrimination. Let \( x \) be a \( d \)-dimensional observation assumed to come from one of two Gaussian populations with common covariance. If \( a = w^T x + w_0 \) represents its linear discriminant, and \( g(a) = \frac{1}{1 + \exp(-a)} \), is a logistic sigmoid activation function, then the output unit \( y = g(a) \) is the posterior probability that \( x \) belongs in class 1. Thus, the output units are nonlinear functions of weighted linear combinations of the input units. In the expression for \( a \), \( w_0 \) is called the bias term.

A neural network with an additional layer of \textit{hidden units} intermediate between the input units and output units is called a multilayered network. The units in a multi-layered feed-forward network have one-way connections to other units, and each unit is only connected to units in higher layers of the network, where the layers are ordered from input to hidden to output. Suppose we have a network with \( d \) input units, \( c \) output units and \( M \) hidden units, where the number of input units represents the dimension of each input variable and the number of output units represents the dimension of each output variable. Following the discussion and notation in Bishop, the output for the \( j \)th hidden unit is obtained by first forming a weighted linear combination of the \( d \) input values, and adding a bias to give

\[
a_j = \sum_{i=1}^{d} w_{ij}^{(1)} x_i + w_{j0}^{(1)}
\]

where \( w_{ij}^{(1)} \) represents a weight in layer 1 going from input unit \( i \) to hidden unit \( j \). Hidden unit \( j \) is then \textit{activated} by evaluating a (nonlinear) activation function, \( g(\cdot) \), at \( a_j \) to get \( z_j = g(a_j) \). Then, the output for the \( k \)th output unit is obtained by first forming a weighted linear combination of the outputs of the \( M \) hidden units, \( a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \), and activating the output unit by evaluating a nonlinear function, \( \tilde{g}(\cdot) \), at \( a_k \) to get \( y_k = \tilde{g}(a_k) \). The entire process of obtaining the output, \( y_k \), from the layered network can be summarized in one expression


\[ y_k = \tilde{g}\left( \sum_{j=1}^{M} w_{kj}^{(2)} \tilde{g}\left( \sum_{i=1}^{d} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right) \]

Note that the bias terms can be absorbed into the weighted linear combinations by introducing variables, \( x_0 \) and \( z_0 \), which take the value 1.

Usual choices for the activation functions are linear, logistic, or threshold (step) functions. Also, as might be suspect, the multilayered network can have many intermediate layers, not just one, as considered here.

**Using nolr**

An example for using nolr comes from the author, M. Mathieson in the file example.ps, a postscript file which comes with the library. Mathieson generates a dataset containing patterns from four ordered classes. The object `x.train` (part of the library) is an 80 by 2 matrix of patterns. Each row is a training pattern, and there are 20 from each class. The object `y.train` (also part of the library) is a vector of class labels (i.e., numbers from 1 to 4). Thus, there are 2 input units and 1 output unit. In example.ps, a NOLR model is fit by maximum likelihood using a single-layered network with 3 hidden units. The likelihood is derived in the file example.ps.

```r
library(nolr)
net <- nolr(x.train, y, size=3, decay=.1)  # fit network
summary(net())
```

a 2-3-1 network for nolr with 14 weights for 4 classes

options were - skip-layer connections
linear output units
Weight decay = 0.1
Parameters estimated by maximum likelihood

Weights (decays):
  0->3  1->3  2->3  0->4  1->4  2->4
  0.01  0.01 -0.01 -0.20  1.03  2.12
  0->5  1->5  2->5  3->6  4->6  5->6
  2.45  2.06  0.66 -0.01  2.20  3.26
  1->6  2->6
  -1.52  1.36

Cut-points:
[1]  -1.271153  2.798747  7.696411

\[-\log(L) = 39.3935594757335\]

The information from `summary(net)` tells us the number of input-hidden-output units we used and the number of weights estimated. The weights are listed under Weights. We are also told that the activation function for the output unit was linear (in nolr, one cannot change this. In nnet, one can, to a sigmoid activation function).

The weights are rather cryptic. 0 represents the input layer bias, and 1 and 2 represent the two input units. 3, 4, and 5 represent the hidden units, and 6 is the output unit. Thus the optimal weights and associated paths are shown. The skip-layer option allows the network to “skip” the hidden layer and go right from input to output, if necessary. The two weights, 1->6 and 2->6 indicate the weights associated with those paths. The cut-points are defined in the likelihood in the file example.ps.

We can plot the 3-dimensional surface indicating the effect of \( x \) on the output from the network or on the probabilities that \( x \) comes from a particular class. Below, I show both fitted surfaces. The second plot shows the probability that \( x \) comes from class 1 (i.e., its response is 1).
Using lrm

To incorporate continuous nonlinear predictors in an ordinal logistic model estimated using \texttt{lrm}, one can transform the predictors using a restricted cubic spline with \(k\) knots (a natural cubic spline). For example, using the synthetic data above, we fit a proportional odds model with continuous nonlinear effects of \(x_1\) and \(x_2\).

In order to take advantage of the \texttt{predict.lrm} function, we must do the following two commands (this is part of the use of the Design library only):

\begin{verbatim}
library("Hmisc", T)  # if not already in the 2nd search position
library("Design", T)
dd<-datadist(x.train$x1,x.train$x2)
options(datadist='dd')
\end{verbatim}

Now fit an additive model with each \(x\) element transformed using a restricted cubic spline with 3 knots:

\begin{verbatim}
fit<-lrm(y.train~rcs(x.train$x1,3) + rcs(x.train$x2,3))
\end{verbatim}

To view the marginal effects of the nonlinear transformations on the log odds that \(y=1\), we can plot them.

\begin{verbatim}
par(mfrow=c(1,2))
rcspline.plot(x1,y,nk=3,show="xbeta",xlab="x1",statloc="none")
rcspline.plot(x2,y,nk=3,show="xbeta",xlab="x2",statloc="none")
\end{verbatim}
The arrows facing downward on the horizontal axis are the knots chosen by `rcs`. Alternatively, one can set them. From the looks of the confidence limits around the transformations, these are not very accurate.

Or, we could look at the effect of each splined x on the probability that $y=1$

```r
rcspline.plot(x1, y, nk=3, show="prob", xlab="x1", statloc="none")
rcspline.plot(x2, y, nk=3, show="prob", xlab="x2", statloc="none")
```

The joint effect of $rcs(x_1)$ and $rcs(x_2)$ is seen in the following two plots. The first plot shows the effect on the "linear" predictor, $rcs(X)^T \beta$, the common part of each log odds $Pr(y \geq j | X)$. Note that $rcs(X)$ represents the splined x's. The second plot shows the joint effect of $rcs(x_1)$ and $rcs(x_2)$ on $Pr(y = 1 | X)$. 
grid <- list(x1=seq(-5,5,0.2),x2=seq(-6,6,0.2))
pred <- matrix(predict(fit,expand.grid(grid),type="lp"),
              length(grid$x1))
persp(grid$x1, grid$x2, pred, xlab="x1", ylab="x2",
     zlab="eta(x)")

pred <- matrix(predict(fit,expand.grid(grid),type="fitted.ind")[,1],
              length(grid$x1))
persp(grid$x1, grid$x2, pred, xlab="x1", ylab="x2",
     zlab="prob(y=1)")

These plots are very similar to those obtained from nolr.
Chapter 10 - Agresti

A. Comparing dependent proportions

McNemar’s test is easily conducted on the data in Table 10.1 using the built-in Splus function, `mcnemar.test`

```r
mcnemar.test(matrix(c(794, 150, 86, 570), byrow=T, ncol=2), correct=F)
```

McNemar's chi-square test without continuity correction

data: matrix(c(794, 150, 86, 570), byrow = T, ncol = 2)
McNemar's chi-square = 17.3559, df = 1, p-value = 0

The statistic reported is Agresti's $z_0^2$, which indicates a significant change in presidential approval rating. The direction of the change can, of course, be determined by computing the statistic, $d$, on p.348.

B. Conditional Logistic Regression (CLR)

In this section, I demonstrate how one might use CLR to fit a logistic model for binary matched pairs in Section 10.1.4 in Agresti. I use the data from problem 10.1 on p. 375. This method was originally introduced by T. Therneau and uses his `coxph` function, which comes with Splus.

Here is how I set up the data:

```r
group<-rep(1:500,rep(2,500))  # each distinct number represents a pair
status<-rep(c(1,0),500)       # status indicates husband(1) or wife (0)
party<-c(rep("Demo",400),rep(c("Demo","Rep"),25),rep(c("Rep","Demo"),
75),rep("Rep",400))
```

To fit the model, I use the function `coxph`. The “time” setting for the function `Surv` is just a vector of ones (`rep(1,1000)`) with as many values as persons.

```r
fit<-coxph(Surv(rep(1,1000),status)~party+strata(group),method="exact")
```

Call:
`coxph(formula = Surv(rep(1, 1000), status) ~ party + strata(group), method = "exact")`

```r
n= 1000
coef exp(coef) se(coef)    z     p
party 1.1        3    0.231 4.76 2e-006
```

```
exp(coef) exp(-coef) lower .95 upper .95
party 3.033 0.333 1.91 4.72
```

Rsquare= 0.026   (max possible= 0.5 )
Likelihood ratio test= 26.2  on 1 df,  p=3.14e-007
Wald test         = 22.6  on 1 df,  p=1.96e-006
Efficient score test = 25  on 1 df,  p=5.73e-007

From the output, we obtain that the estimate of the odds ratio is 3. The odds of being a Democrat are three times more likely for the wife of the matched pair than for the husband. The effect of political party is significant. Thus, marginal homogeneity is rejected.
C. Symmetry Models

To fit symmetry models, Agresti uses the migration data in Table 10.2, obtained in Splus using the following commands.

```r
residence80 <- c("NE", "MW", "S", "W")
residence85 <- c("NE", "MW", "S", "W")

table.10.2 <- expand.grid(res80 = residence80, res85 = residence85, symm = c("1", "2"))

table.10.2$counts <-
270, 63, 176, 286, 10192, 11607, 87, 172, 63, 100, 13677, 225, 176, 366, 515,
17819, 286, 124, 302, 270, 10192

The "symm" factor in data frame table.10.2 is used as a result of the suggestion in problem 10.27 (p. 382), which explains how quasi-symmetry models can be fit using loglinear models. "symm" is factor "Z", with two levels.

Ordinary Symmetry Model

An ordinary symmetry model is simply fit as a two-factor log-linear model, as shown here.

```r
fit <- glm(counts ~ res80 * res85, family = poisson(link = log), data = table.10.2)
```

Coefficients:

<table>
<thead>
<tr>
<th></th>
<th>res801</th>
<th>res802</th>
<th>res803</th>
<th>res851</th>
<th>res852</th>
<th>res853</th>
</tr>
</thead>
<tbody>
<tr>
<td>res801res851</td>
<td>6.329792</td>
<td>0.1776753</td>
<td>0.18193</td>
<td>-0.03731356</td>
<td>0.1776753</td>
<td>0.18193</td>
</tr>
<tr>
<td>2.451728</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>res802res851</td>
<td>res803res851</td>
<td>res801res852</td>
<td>res802res852</td>
<td>res803res852</td>
<td>res801res853</td>
<td></td>
</tr>
<tr>
<td>0.03945671</td>
<td>0.09719178</td>
<td>0.03945671</td>
<td>0.5854987</td>
<td>0.008290874</td>
<td>0.09719178</td>
<td></td>
</tr>
<tr>
<td>0.008290874</td>
<td>0.3470497</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Degrees of Freedom: 32 Total; 16 Residual

Residual Deviance: 487.1004

As it says in problem 10.27, we need to divide the Residual Deviance by 2 to get the true Goodness of Fit statistic:

```r
fit$deviance/2
[1] 243.5502
```

It is also apparent that one must compute the df by hand since the residual df for the symmetry model is 6, not 16, and the total is half of 32.

Another way to fit the symmetry model, which does not require the adjustment of df or of GoF statistics, is via a method proposed by A. Zaslavsky. The data is set up in the same way, with the exception of the "symm" factor.

```r
table.10.2 <- expand.grid(res80 = residence80, res85 = residence85)

table.10.2$counts <-
(11607, 100, 366, 124, 87, 13677, 515, 302, 172, 225, 17819, 270, 63, 176, 286, 10192)

table.10.2$symm <- paste(
pmin(as.numeric(table.10.2$res80), as.numeric(table.10.2$res85)),
pmax(as.numeric(table.10.2$res80), as.numeric(table.10.2$res85)), sep = ",")
```

<table>
<thead>
<tr>
<th>res80</th>
<th>res85</th>
<th>counts</th>
<th>symm</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE</td>
<td>NE</td>
<td>11607</td>
<td>1,1</td>
</tr>
<tr>
<td>MW</td>
<td>NE</td>
<td>100</td>
<td>1,2</td>
</tr>
</tbody>
</table>

```r
res80 85 counts  symm
1  NE  NE  11607  1,1
2  MW  NE  100    1,2
```
To fit the symmetry model, just use "symm" as the factor:

```r
fit<-glm(counts~symm,family=poisson(log),data=table.10.2)
```

Coefficients:

<table>
<thead>
<tr>
<th></th>
<th>(Intercept)</th>
<th>symm1</th>
<th>symm2</th>
<th>symm3</th>
<th>symm4</th>
<th>symm5</th>
<th>symm6</th>
<th>symm7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6.958844</td>
<td>-2.410701</td>
<td>-0.4513163</td>
<td>-0.4898459</td>
<td>0.7031941</td>
<td>-0.1328645</td>
<td>-0.157338</td>
<td>0.4209411</td>
</tr>
<tr>
<td></td>
<td>0.4209411</td>
<td>symm8</td>
<td>symm9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-0.134868</td>
<td>0.2522793</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Degrees of Freedom: 16 Total; 6 Residual
Residual Deviance: 243.5502

Note that one does not need to adjust the Goodness of Fit statistic nor the df.

Quasi-Symmetry Model

Using the suggestion in problem 10.27, and the first definition of "symm", we have the quasi-symmetry fit:

```r
fit<-glm(counts~symm*res80+symm*res85+res80*res85,family=poisson(link=log),
data=table.10.2)
```

Coefficients:

<table>
<thead>
<tr>
<th></th>
<th>(Intercept)</th>
<th>symm</th>
<th>res801</th>
<th>res802</th>
<th>res803</th>
<th>res804</th>
<th>res805</th>
<th>res851</th>
<th>res852</th>
<th>res853</th>
<th>symmres801</th>
<th>symmres802</th>
<th>symmres803</th>
<th>symmres805</th>
<th>symmres852</th>
<th>symmres853</th>
<th>symmres855</th>
<th>symmres856</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6.298585</td>
<td>1.77305e-016</td>
<td>0.1797539</td>
<td>0.1800794</td>
<td>-0.03565628</td>
<td>0.1797539</td>
<td>0.1800794</td>
<td>-0.03565628</td>
<td>-0.01224104</td>
<td>-0.128141</td>
<td>-0.04887195</td>
<td>0.01224104</td>
<td>0.128141</td>
<td>0.04887195</td>
<td>-0.03565628</td>
<td>-0.01224104</td>
<td>-0.128141</td>
<td>-0.04887195</td>
</tr>
<tr>
<td></td>
<td>2.451878</td>
<td>0.04095189</td>
<td>0.09777525</td>
<td>0.04095189</td>
<td>0.6015282</td>
<td>0.01442996</td>
<td>0.3494123</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Degrees of Freedom: 32 Total; 9 Residual
Residual Deviance: 5.971925

```r
fit$deviance/2
[1] 2.985962
```

Fitted values for the quasi-symmetry model are obtained as follows:

```r
matrix(fitted(fit),ncol=4,byrow=T,dimnames=list(rep(residence80,2),
residence85))[1:4,]
```

# we take only the first set of fitted values
Using Zaslavsky's definition of "symm", a fit of the quasi-symmetry model is obtained using

\[
\text{fit<}-\text{glm(counts~symm+res80, family=poisson(link=log), data=table.10.2)}
\]

Coefficients:

(Intercept)     symm1      symm2      symm3     symm4      symm5      symm6 

symm7  

6.933879 -2.423092 -0.6049717 -0.5461458 0.7378705 -0.1830642 -0.1816523

0.354548

symm8    symm9     res801   res802    res803

-0.1799573 0.222472 0.02448208 0.256282 0.0977439

Degrees of Freedom: 16 Total; 3 Residual
Residual Deviance: 2.985962

To obtain fitted values, type:

\[
\text{matrix(fitted(fit2), ncol=4, byrow=T, dimnames=list(residence80, residence85))[1:4,]}
\]

we take only the first set of fitted values

\[
\begin{array}{cccc}
\text{NE} & \text{MW} & \text{S} & \text{W} \\
11607.00000 & 95.78862 & 370.4375 & 123.7739 \\
91.21138 & 13677.00000 & 501.6825 & 311.1061 \\
167.56253 & 238.31746 & 17819.00000 & 261.1200 \\
63.22609 & 166.89392 & 294.8800 & 10192.00000 \\
\end{array}
\]

Quasi-independence Model

The quasi-independence model is fit in two ways. First, I use the method in GLIM described in Table A.20 of Appendix A of Agresti. Then, I use the method of SAS PROC CATMOD, also mentioned in Appendix A.

The method in GLIM adds to the independence model four indicator variables that represent the main diagonal cells.

\[
\text{table.10.2<-expand.grid(res80=residence80, res85=residence85)}
\]

\[
\text{table.10.2$counts<-c(11607,100,366,124,87,13677,501,302,172,225,17819,270,63,176,286,10192)}
\]

I use these Indicator variables:

\[
\text{table.10.2$D1<-c(1,rep(0,15))}
\]
\[
\text{table.10.2$D2<-c(rep(0,5),1,rep(0,10))}
\]
\[
\text{table.10.2$D3<-c(rep(0,10),1,rep(0,5))}
\]
\[
\text{table.10.2$D4<-c(rep(0,15),1)}
\]

Now, fit the quasi-independence model:

\[
\text{fit<-glm(counts~res80+res85+D1+D2+D3+D4, family=poisson(link=log), data=table.10.2)}
\]

Degrees of Freedom: 16 Total; 5 Residual
Residual Deviance: 69.5094

The fitted values for the quasi-independence model are obtained as follows:
matrix(fitted(fit),ncol=4,byrow=T,dimnames=list(residence80, residence85))[1:4,]  
we take only the first set of fitted values

<table>
<thead>
<tr>
<th></th>
<th>NE</th>
<th>MW</th>
<th>S</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE</td>
<td>11607.0000</td>
<td>126.5622</td>
<td>312.9086</td>
<td>150.5292</td>
</tr>
<tr>
<td>MW</td>
<td>117.3881</td>
<td>13677.0000</td>
<td>531.1125</td>
<td>255.4994</td>
</tr>
<tr>
<td>S</td>
<td>133.2261</td>
<td>243.8025</td>
<td>17819.0000</td>
<td>289.9714</td>
</tr>
<tr>
<td>W</td>
<td>71.3858</td>
<td>130.6353</td>
<td>322.9789</td>
<td>10192.0000</td>
</tr>
</tbody>
</table>

Appendix A in Agresti (p. 500) also suggests fitting the quasi-independence model as SAS PROC CATMOD does. That is, zero-out the main diagonal of the contingency table, and fit an independence log-linear model, treating the main diagonal zeroes as structural zeroes. I do that here using loglin with the ‘start’ option.

```r
table.10.2<-expand.grid(res80=residence80,res85=residence85)
table.10.2$counts<-c(0,100,366,124,87,0,515,302,172,225,0,270,63,176,286,0)  # note the zeroes
table.10.2<-t(design.table(table.10.2))  # create a table out of the data frame
loglin(table.10.2,margin=list(1,2),start=abs(as.numeric(diag(4))-1),fit=T)
```

7 iterations: deviation 0.0811768

$lrt:
[1] 69.50941

$pearson:
[1] 71.15848

$df:
[1] 9

$margin:
$margin[[1]]:
[1] "res80"

$margin[[2]]:
[1] "res85"

$fit:

<table>
<thead>
<tr>
<th></th>
<th>NE</th>
<th>MW</th>
<th>S</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE</td>
<td>0.0000</td>
<td>126.5636</td>
<td>312.9075</td>
<td>150.5306</td>
</tr>
<tr>
<td>MW</td>
<td>117.38965</td>
<td>0.0000</td>
<td>531.1134</td>
<td>255.5030</td>
</tr>
<tr>
<td>S</td>
<td>133.22371</td>
<td>243.7990</td>
<td>0.0000</td>
<td>289.9664</td>
</tr>
<tr>
<td>W</td>
<td>71.38664</td>
<td>130.6374</td>
<td>322.9791</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

As noted in a previous section, when we have structural zeroes in loglin, we must adjust the df for the zeroes. Thus, we can subtract 4 to get df = 5.

**D. Marginal Homogeneity Models**

In this section, the marginal homogeneity model is fit to the migration data using the model matrix in Table A.19 of Appendix A in Agresti.

First, I set up the dummy matrix in Table A.19 as a matrix, then I make it a data frame:

```r
dummies<-matrix(0,nr=16,nc=13)
dummies[cbind(c(1:3,5:7,9:11),c(1:3,4:6,7:9))]<-1
dummies[cbind(rep(c(4,8,12),rep(3,3)),c(1:3,4:6,7:9))]<-1
dummies[cbind(rep(13:15,3),c(1:3,4:6,7:9))]<-1
dummies[16,10]<-1
diag(dummies[c(4,8,12),c(11,12,13)])<-1
```
diag(dummies[c(13,14,15),c(11,12,13)])<-1
dummies<-data.frame(counts=c(11607,100,366,124,87,13677,515,302,172,225,17819,270,63,176,286,10192),dummies)
names(dummies)<-c("counts","m11","m12","m13","m21","m22","m23","m31","m32","m33","m44","m1","m2","m3")

| counts m11 m12 m13 m21 m22 m23 m31 m32 m33 m44 m1 m2 m3 |
|--------------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 1 11607 1 0 0 0 0 0 0 0 0 0 0 0 0 |
| 2 100 0 1 0 0 0 0 0 0 0 0 0 0 0 |
| 3 366 0 0 1 0 0 0 0 0 0 0 0 0 0 |
| 4 124 -1 -1 -1 0 0 0 0 0 0 0 1 0 0 |
| 5 87 0 0 0 1 0 0 0 0 0 0 0 0 0 |
| 6 13677 0 0 0 0 1 0 0 0 0 0 0 0 0 |
| 7 515 0 0 0 0 0 1 0 0 0 0 0 0 0 |
| 8 302 0 0 0 -1 -1 -1 0 0 0 0 0 1 0 |
| 9 172 0 0 0 0 0 0 1 0 0 0 0 0 0 |
| 10 225 0 0 0 0 0 0 0 1 0 0 0 0 0 |
| 11 17819 0 0 0 0 0 0 0 0 1 0 0 0 0 |
| 12 270 0 0 0 0 0 0 -1 -1 -1 0 0 0 1 |
| 13 63 -1 0 0 -1 0 0 -1 0 0 0 1 0 0 |
| 14 176 0 -1 0 0 -1 0 0 -1 0 0 0 1 0 |
| 15 286 0 0 -1 0 0 -1 0 0 -1 0 0 0 1 |
| 16 10192 0 0 0 0 0 0 0 0 0 1 0 0 0 |

The above matrix is supposed to encompass the constraints on the expected cell entries.

Now, I fit the marginal homogeneity model using a poisson family with identity link:

```r
fit<-glm(counts~.-1,family=poisson(identity),data=dummies)
```

Degrees of Freedom: 16 Total; 3 Residual
Residual Deviance: 240.7458

```r
matrix(fitted(fit),byrow=T,nc=4,dimnames=list(residence80,residence85))
```

<table>
<thead>
<tr>
<th>NE</th>
<th>MW</th>
<th>S</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE 11607.00000</td>
<td>98.08285</td>
<td>265.6867</td>
<td>93.96975</td>
</tr>
<tr>
<td>MW 88.73298</td>
<td>13677.00000</td>
<td>379.0733</td>
<td>232.34523</td>
</tr>
<tr>
<td>S 276.47479</td>
<td>350.80060</td>
<td>17819.0000</td>
<td>287.25936</td>
</tr>
<tr>
<td>W 92.53157</td>
<td>251.26811</td>
<td>269.7747</td>
<td>10192.0000</td>
</tr>
</tbody>
</table>

Check that the row and column sums match:

```r
rowSums(matrix(fitted(fit),byrow=T,nc=4))
[1] 12064.74 14377.15 18733.53 10805.57
```

```r
colSums(matrix(fitted(fit),byrow=T,nc=4))
[1] 12064.74 14377.15 18733.53 10805.57
```

Another way to test marginal homogeneity is to use Bhapkar's (1966) and Stuart's (1955) asymptotic test. However, instead of using Lipsitz's (1988) matrix, we can use a matrix that is more transparent in terms of how easily it is to see the constraints. The development is found in Wickens (1989).

The constraints that all cross marginal totals are equal is equivalent to the following three equations:

\[
\pi_{12} + \pi_{13} + \pi_{14} = \pi_{21} + \pi_{31} + \pi_{41}
\]

\[
\pi_{12} + \pi_{23} + \pi_{24} = \pi_{21} + \pi_{23} + \pi_{24}
\]

\[
\pi_{13} + \pi_{23} + \pi_{34} = \pi_{31} + \pi_{32} + \pi_{34}
\]
which we can put into a contrast matrix directly:

\[
A <- \text{matrix(c(0,1,1,-1,0,0,-1,0,0,1,0,0,0,-1,0,0,0,0,1,0,0,1,0,0,0,-1,0,0,0,-1,0,0,0,0,1,0,0,1,0,0,0,-1,0,0,0,0,1,0,0,1,0), nc=16, nr=3, byrow=T)}
\]

Now, get the sample proportions and compute the linear combinations imposed by the constraint matrix, A.

\[
p <- (\text{dummies$counts/sum(dummies$counts)})
\]

\[
y <- A %*% p
\]

Now, get Sp, the var/cov matrix of p. (This is a complicated way to do it. I’m sure there is a simpler way.)

\[
Sp <- \text{diag(p*(1-p))}
\]

\[
\text{oldres} <- \text{kronecker(p[1], p[-1])}
\]

\[
\text{for(i in 2:15){ res} <- kronecker(p[i], p[-(1:i)]); res} <- \text{c(oldres, res); oldres} <- res 
\]

\[
Sp[\text{lower.tri(Sp)}] <- -res
\]

\[
\text{temp} <- t(Sp)
\]

\[
\text{temp[lower.tri(Sp)]} <- -res
\]

\[
Sp <- \text{temp/sum(dummies$counts)}
\]

Now, get Sy, the covariance matrix of the linear combinations:

\[
Sy <- A %*% Sp %*% t(A)
\]

And, compute W:

\[
\text{as.numeric(t(y) *%*% solve(Sy) *%*% y)}
\]

[1] 236.4906

**E. Square Tables with Ordered Categories**

Using the data in Table 10.4, I fit the symmetry and conditional symmetry models.

\[
table.10.4 <- \text{data.frame(expand.grid(case=c(0,1,2,3), control=c(0,1,2,3)), counts=c(31,12,14,6,5,1,1,5,0,2,1,0,0,1,0))}
\]

**Symmetry model**

I modify table.10.4 by adding a symmetry factor:

\[
table.10.4$symm <- paste(\text{pmin(as.numeric(table.10.4$case), as.numeric(table.10.4$control))}, \text{pmax(as.numeric(table.10.4$case), as.numeric(table.10.4$control))}, sep="", )
\]

and obtain the chi-squared statistic using .pearson.x2

\[
\text{fit} <- \text{glm(counts ~ symm, data=table.10.4, family=poisson(log))}
\]

\[
.pearson.x2(\text{observed=table.10.4$counts, expected=fitted(fit)})$X2
\]

[1] 15.14592

**Conditional Symmetry Model**

To fit a conditional symmetry model, I first obtain a vector that represents tau:

\[
temp <- \text{matrix(0, nr=4, nc=4)}
\]

\[
tau <- \text{as.numeric(ifelse(row(temp)<col(temp), 1, 0))}
\]
Then, I add tau as a factor in the model:

```r
fit<-glm(counts~symm+tau,family=poisson(log),data=table.10.4)
```

Coefficients:

<table>
<thead>
<tr>
<th></th>
<th>(Intercept)</th>
<th>symm1</th>
<th>symm2</th>
<th>symm3</th>
<th>symm4</th>
<th>symm5</th>
<th>symm6</th>
<th>symm7</th>
<th>symm8</th>
<th>symm9</th>
<th>tau</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1960172</td>
<td>-0.4370336</td>
<td>-0.1086027</td>
<td>-0.3424712</td>
<td>-0.5091759</td>
<td>-0.3849995</td>
<td>-0.2749997</td>
<td>-0.08544468</td>
<td>-0.0968229</td>
<td>-0.9984199</td>
<td>-1.157453</td>
</tr>
</tbody>
</table>

Degrees of Freedom: 16 Total; 5 Residual
Residual Deviance: 5.336852

The model has a chi-squared GOF value:

```r
.pearson.x2(observed=table.10.4$counts,expected=fitted(fit))$X2
```

[1] 3.605283

To obtain the fitted values for the conditional symmetry model, one may type:

```r
matrix(round(fitted(fit),1),nr=4,nc=4,byrow=F,dimnames=list(case=c(0,1,2,3),control=c(0,1,2,3)))
```

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31.0</td>
<td>4.1</td>
<td>4.5</td>
<td>1.4</td>
</tr>
<tr>
<td>1</td>
<td>12.9</td>
<td>1.0</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>14.5</td>
<td>0.8</td>
<td>2.0</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>4.6</td>
<td>0.8</td>
<td>1.5</td>
<td>0.0</td>
</tr>
</tbody>
</table>

F. Measuring Agreement

For this section, I use the Pathologist data in Table 10.5. Here is how I set it up.

```r
PathA<-factor(c(1,2,3,4,5))
PathB<-PathA
table.10.5<-data.frame(expand.grid(PathA=PathA,PathB=PathB),
count=c(22,5,0,0,0,2,7,2,1,0,2,14,36,14,3,0,0,0,7,0,0,0,0,0,0))
table.10.5<-design.table(table.10.5)
```

Cohen’s Kappa

One way to compute Kappa is with the following function:

```r
Kappa<-function(table){
  pij<-table/sum(table)
  pio<-sum(diag(pij))
  pie<-sum(rowSums(pij)*colSums(pij))
  (pio-pie)/(1-pie)
}
```

Applied to Table 10.5, we get

```r
Kappa(table.10.5)
```

[1] 0.4984183
Uniform Association Plus Extra Agreement Model

To fit a linear-by-linear association plus extra agreement model (10.28) to Table 10.5, I will use unit-spaced scores.

```r
delta<-as.numeric(diag(5)) # Gives 1 for diagonal entries, 0 otherwise
table.10.5<-data.frame(expand.grid(PathA=factor(PathA), PathB=factor(PathB)), count=c(22,5,0,0,2,7,2,1,0,2,14,14,3,0,0,0,7,0,0,0,0,0,3))
fit<-glm(count~PathA+PathB+delta+codes(PathA):codes(PathB), family=poisson(log), data=table.10.5) # The codes function gives the unit-spaced scores.
```

```r
summary(fit, cor=F)
```

Coefficients:

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Std. Error</th>
<th>t value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>-11.4248226</td>
<td>3.5087037</td>
<td>-3.256138</td>
</tr>
<tr>
<td>PathA1</td>
<td>-0.8506102</td>
<td>0.3803249</td>
<td>-2.236536</td>
</tr>
<tr>
<td>PathA2</td>
<td>-1.4361990</td>
<td>0.3928591</td>
<td>-3.655761</td>
</tr>
<tr>
<td>PathA3</td>
<td>-1.5228694</td>
<td>0.5070563</td>
<td>-3.003353</td>
</tr>
<tr>
<td>PathA4</td>
<td>-1.9647665</td>
<td>0.5572738</td>
<td>-3.5267567</td>
</tr>
<tr>
<td>PathB1</td>
<td>-1.3273248</td>
<td>0.3009011</td>
<td>-4.411166</td>
</tr>
<tr>
<td>PathB2</td>
<td>-0.7410464</td>
<td>0.3822727</td>
<td>-1.938528</td>
</tr>
<tr>
<td>PathB3</td>
<td>-2.0525682</td>
<td>0.4964779</td>
<td>-4.134259</td>
</tr>
<tr>
<td>PathB4</td>
<td>-2.4092934</td>
<td>0.6539193</td>
<td>-3.684389</td>
</tr>
<tr>
<td>delta</td>
<td>1.0668241</td>
<td>0.4037553</td>
<td>2.642254</td>
</tr>
<tr>
<td>codes(PathA):codes(PathB)</td>
<td>1.1498870</td>
<td>0.3418667</td>
<td>3.363554</td>
</tr>
</tbody>
</table>

(Dispersion Parameter for Poisson family taken to be 1 )

Null Deviance: 267.6605 on 24 degrees of freedom
Residual Deviance: 8.411956 on 14 degrees of freedom

The underlined values in the list of coefficients are estimates of \( \delta \) and \( \beta \), respectively, in Agresti.

To obtain a matrix of the fitted values, one may type:

```r
matrix(round(fitted(fit),1),nr=5,nc=5, dimnames=list(PathA, PathB))
```

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22.1</td>
<td>1.7</td>
<td>2.2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>4.4</td>
<td>8.9</td>
<td>12.5</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>0.4</td>
<td>1.0</td>
<td>36.1</td>
<td>0.5</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>0.1</td>
<td>0.4</td>
<td>15.6</td>
<td>5.4</td>
<td>0.5</td>
</tr>
<tr>
<td>5</td>
<td>0.0</td>
<td>0.0</td>
<td>2.6</td>
<td>1.0</td>
<td>2.4</td>
</tr>
</tbody>
</table>

G. Bradley-Terry Model

We fit the Bradley-Terry model for paired comparisons using the data in Table 10.6. There are several ways to fit this model, given a set of paired comparisons. I will fit it first as a quasi-symmetry model, then as a logit model.

To set up the data, type

```r
losing.team<-c("Milwaukee", "Detroit", "Toronto", "NY", "Boston", "Cleveland", "Baltimore")
win.team<-losing.team
table.10.6<-expand.grid(losing=factor(losing.team), winning=factor(win.team))
table.10.6$counts<-c(0, 7, 9, 7, 9, 11, 6, 0, 7, 5, 11, 9, 9, 4, 6, 0, 7, 7, 8, 12, 6, 8, 6, 0, 6, 7, 10, 6, 2, 6, 7, 0, 7, 12, 4, 4, 5, 6, 6, 0, 6, 2, 4, 1, 3, 1, 7, 0)
```
Quasi-symmetry fit:

```r
table.10.6$symm <- paste(pmin(as.numeric(factor(table.10.6$winning, levels=win.team)),
as.numeric(factor(table.10.6$losing, levels=losing.team))),
pmax(as.numeric(factor(table.10.6$winning, levels=win.team)),
as.numeric(factor(table.10.6$losing, levels=losing.team))), sep = "",
)
```

Quasi-symmetry fit:

```r
fit <- glm(counts ~ symm + losing, data = table.10.6, family = poisson(log))
```

Degrees of Freedom: 49 Total; 15 Residual
Residual Deviance: 15.73729

Fitted values are obtained by

```r
matrix(round(fitted(fit), 1), nr = 7, nc = 7, byrow = T, dimnames = list(win.team, losing.team))
```

# byrow = T matches the data entry of the counts above

<table>
<thead>
<tr>
<th></th>
<th>Milwaukee</th>
<th>Detroit</th>
<th>Toronto</th>
<th>NY</th>
<th>Boston</th>
<th>Cleveland</th>
<th>Baltimore</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milwaukee</td>
<td>0.0</td>
<td>7.0</td>
<td>7.4</td>
<td>7.6</td>
<td>8.0</td>
<td>9.2</td>
<td>10.8</td>
</tr>
<tr>
<td>Detroit</td>
<td>6.0</td>
<td>0.0</td>
<td>7.0</td>
<td>7.1</td>
<td>7.6</td>
<td>8.8</td>
<td>10.5</td>
</tr>
<tr>
<td>Toronto</td>
<td>5.6</td>
<td>6.0</td>
<td>0.0</td>
<td>6.7</td>
<td>7.1</td>
<td>8.4</td>
<td>10.2</td>
</tr>
<tr>
<td>NY</td>
<td>5.4</td>
<td>5.9</td>
<td>6.3</td>
<td>0.0</td>
<td>7.0</td>
<td>8.3</td>
<td>10.1</td>
</tr>
<tr>
<td>Boston</td>
<td>5.0</td>
<td>5.4</td>
<td>5.9</td>
<td>6.0</td>
<td>0.0</td>
<td>7.9</td>
<td>9.8</td>
</tr>
<tr>
<td>Cleveland</td>
<td>3.8</td>
<td>4.2</td>
<td>4.6</td>
<td>4.7</td>
<td>5.1</td>
<td>0.0</td>
<td>8.6</td>
</tr>
<tr>
<td>Baltimore</td>
<td>2.2</td>
<td>2.5</td>
<td>2.8</td>
<td>2.9</td>
<td>3.2</td>
<td>4.4</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Logit model fit (see Appendix A in Agresti):

Appendix A states that one can fit a Bradley-Terry model by creating seven artificial variables, corresponding to the seven teams, and then fitting a logit model to the counts of wins. Here, X1 - X7 are the artificial variables, which equal 1 if the particular team won the game represented by the cell, -1 if the team lost the game, and 0 if the team did not play.

```r
X1 <- c(-1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
X2 <- c(1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
X3 <- c(0, 1, 0, 0, 0, 0, 1, 0, 0, 0, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0)
X4 <- c(0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, -1, -1, -1, 0, 0, 0, 0)
X5 <- c(0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, -1, 0)
X6 <- c(0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0)
X7 <- c(0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0)
```

```r
response <- cbind(6, 4, 6, 4, 2, 6, 8, 2, 4, 4, 6, 6, 5, 1, 7, 6, 3, 6, 1, 7),
                13 - c(6, 4, 6, 4, 2, 6, 8, 2, 4, 4, 6, 6, 5, 1, 7, 6, 3, 6, 1, 7))
```

The first column of “response” is the number of wins in each of the 21 games; the second is the number of losses.
Here is the fit of a logit model:

\[
\text{fit} \leftarrow \text{glm}(\text{response} \sim -1 + \text{X1} + \text{X2} + \text{X3} + \text{X4} + \text{X5} + \text{X6}, \text{family} = \text{binomial}) \quad \# \text{exclude intercept}
\]

Degrees of Freedom: 21 Total; 15 Residual
Residual Deviance: 15.7365

and its fitted values:

\[
\text{junk} \leftarrow \text{matrix}(0, \text{nc}=7, \text{nr}=7, \text{dimnames} = \text{list} (\text{win.team}, \text{win.team}))
\]

\[
\text{junk}[\text{lower.tri} (\text{junk})] \leftarrow 13 \times \text{fitted} (\text{fit}), 1
\]

\[
\text{junk} \leftarrow 13 - \text{round} (13 \times \text{fitted} (\text{fit}), 1)
\]

\[
\begin{array}{cccccccc}
\text{Milwaukee} & \text{Detroit} & \text{Toronto} & \text{NY} & \text{Boston} & \text{Cleveland} & \text{Baltimore} \\
0.0 & 7.0 & 7.4 & 7.6 & 8.0 & 9.2 & 10.8 \\
6.0 & 0.0 & 7.0 & 7.1 & 7.6 & 8.8 & 10.5 \\
5.6 & 6.0 & 0.0 & 6.7 & 7.1 & 8.4 & 10.2 \\
5.4 & 5.9 & 6.3 & 0.0 & 7.0 & 8.3 & 10.1 \\
5.0 & 5.4 & 5.9 & 6.0 & 0.0 & 7.9 & 9.8 \\
3.8 & 4.2 & 4.6 & 4.7 & 5.1 & 0.0 & 8.6 \\
2.2 & 2.5 & 2.8 & 2.9 & 3.2 & 4.4 & 0.0 \\
\end{array}
\]

Issuing the command \text{fitted} (\text{fit}) gives the \( \Pi_{ij} \)’s.

### H. Bradley-Terry Model-with Order Effect (rough draft)

Apparently, this model can be fit in many ways. Of all the ways I tried, the logit model fit (as per p. 373, Agresti) gave the closest answers. However, I still can’t get this to work.

As mentioned in Agresti, the 42 pair sets (where order matters) can be viewed as 42 independent binomial samples, where response is the number of successes (wins) out of the total played.

\[
\text{response} \leftarrow \text{cbind}(c(4,4,4,4,6,4,6,4,4,6,4,2,4,4,6,4,6,4,6,5,6,2, \\
3,2,3,5,2,2,4,5,2,3,1,2,3,3,1,4,4,2,4,1,3), \\
c(3,2,3,1,2,0,2,3,0,1,3,4,3,2,0,3,2,1,2,0,4, \\
3,5,3,1,5,5,3,1,5,3,5,5,3,4,6,2,3,4,2,6,4))
\]

I create the “home advantage” variable by giving each of the 42 game sets a 1 if a home team won the set and 0 if an away team won or if there was a tie in the number of games won.

\[
\text{Home} \leftarrow c(1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,0, \\
0,0,0,1,0,0,1,1,1,0,0,0,0,0,0,0,1,1,0,0,0) \quad \# \text{ties give no advantage}
\]

Then, the seven artificial variables (similar to those above, except now there are 42) are set up as follows, with a 1 for a win by that team, -1 for a loss, and 0 for no-play. There are several ways I could have set up ties; but no way gave correct answers in the model fit.
Model fit:

```r
glm(response~1+X1+X2+X3+X4+X5+X6+Home, family=binomial)
```

Degrees of Freedom: 42 Total; 35 Residual
Residual Deviance: 40.38968
Chapter 11 - Agresti

A. Symmetry Models for Repeated Categorical Responses

Agresti fits two symmetry models to the data in Table 11.1 (Attitudes towards legalized abortion): Complete symmetry and Quasi-symmetry.

First, set up the data

```r
library(tidyverse)

table.11.1 <- data.frame(expand.grid(Poor=c("y", "n"), Unmarried=c("y", "n"), Rape=c("y", "n"), Defect=c("y", "n")),
                          counts=c(605, 68, 91, 320, 1, 0, 3, 45, 7, 3, 7, 54, 2, 0, 3, 125))
```

Complete Symmetry

To fit a complete symmetry model to Table 11.1, I use the suggestion by A. Zaslavsky, but generalize it to a model with T=4 factors (conditions) and I=2 responses under each factor.

Here is the resulting table. Note that I label "symm" as "4" for all rows (subject profiles) with 4 yeses. I give "symm" the label "3" for all rows with 3 yeses, etc. Thus, the complete symmetry model has 5 parameters. This model assumes that all orders of the responses are equal.

```r
table.11.1$symm <- as.factor(rowSums(cbind(table.11.1$Poor, table.11.1$Unmarried, table.11.1$Rape, table.11.1$Defect)-1))
```

```r
library(dplyr)
table.11.1 %>% mutate(symm)
```

<table>
<thead>
<tr>
<th>Poor</th>
<th>Unmarried</th>
<th>Rape</th>
<th>Defect</th>
<th>counts</th>
<th>symm</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>n</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>68</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>91</td>
<td>1</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>320</td>
<td>2</td>
</tr>
<tr>
<td>y</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>45</td>
<td>3</td>
</tr>
<tr>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>y</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>54</td>
<td>3</td>
</tr>
<tr>
<td>y</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>y</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>125</td>
<td>4</td>
</tr>
</tbody>
</table>

Now, I fit the model:

```r
fit <- glm(counts ~ symm, data=table.11.1, family=poisson(log))
```

Coefficient Estimates:

```
  (Intercept)        symm1        symm2        symm3        symm4
  4.445258 -1.336765 -0.348698 -0.370272  0.09576387
```

Degrees of Freedom: 16 Total; 11 Residual
Residual Deviance: 1334.931

Quasi-symmetry Model

Generalizing the same method as was used in Chapter 10 to a higher dimensional table, we get the fit,
fit<-glm(counts~symm+Unmarried+Rape+Defect,family=poisson(link=log),
data=table.11.1)

Degrees of Freedom: 16 Total; 8 Residual
Residual Deviance: 33.23141

Note that I have excluded the “Poor” variable from the model formula above. If I had kept all variables in the formula, the last mentioned variable would not be estimated.

No-three-factor interaction model

Agresti compares the complete symmetry model and quasi-symmetry model to a No-three-factor interaction model and finds large differences in the estimated conditional odds ratios for the latter model.

Here is the model fit.

fit<-glm(counts~(Unmarried+Rape+Poor+Defect)^2,family=poisson(link=log),
data=table.11.1)

Coefficients:
(Intercept) Unmarried      Rape      Poor     Defect Unmarried:Rape
2.439943 0.9590131 -1.278875 0.3275534 -0.7938695 0.6603908
Unmarried:Poor Unmarried:Defect Rape:Poor Rape:Defect Poor:Defect
0.8637749 0.3821812 0.3306227 0.7118181 0.2291614

Degrees of Freedom: 16 Total; 5 Residual
Residual Deviance: 6.567354

And, the conditional odds ratios can be obtained using, for example,

exp(4*fit$coefficients[10])

for the DR odds ratio.

**B. Marginal Homogeneity**

Marginal homogeneity implies T identical response distributions across T time periods. Agresti tests marginal homogeneity of the binary response distributions (success/failure) across three drugs for the data in Table 11.2.

table.11.2<-data.frame(expand.grid(C=c("y","n"),B=c("y","n"),A=c("y","n")),
counts=c(6,16,2,4,2,4,6,6))

<table>
<thead>
<tr>
<th>C</th>
<th>B</th>
<th>A</th>
<th>counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>y</td>
<td>y</td>
<td>6</td>
</tr>
<tr>
<td>y</td>
<td>y</td>
<td>n</td>
<td>16</td>
</tr>
<tr>
<td>n</td>
<td>y</td>
<td>y</td>
<td>2</td>
</tr>
<tr>
<td>y</td>
<td>n</td>
<td>y</td>
<td>4</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
<td>y</td>
<td>2</td>
</tr>
<tr>
<td>n</td>
<td>y</td>
<td>n</td>
<td>4</td>
</tr>
<tr>
<td>y</td>
<td>n</td>
<td>n</td>
<td>6</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
<td>n</td>
<td>6</td>
</tr>
</tbody>
</table>
First, I compute Bhapkar’s W statistic, following the same logic (i.e., from Wickens, 1989) as in Chapter 10.

The constraint that all marginals (A, B, C) are equal is equivalent to the following two equations.

```r
A <- matrix(c(0, 0, 1, 1, -1, -1, 0, 0,
              0, 1, -1, 0, 0, 1, -1, 0),
             nc = 8, nr = 2, byrow = T)
```

Then, the following commands will calculate W:

```r
prob <- (table.11.2$counts / sum(table.11.2$counts))
y <- A %*% prob
Sp <- diag(prob * (1 - prob))
oldres <- kronecker(prob[1], prob[-1])
for (i in 2:7) {
  res <- kronecker(prob[i], prob[-(1:i)]); res <- c(oldres, res); oldres <- res
}
Sp[lower.tri(Sp)] <- -res
temp <- t(Sp)
temp[lower.tri(Sp)] <- -res
Sp <- temp / sum(table.11.2$counts)

Sy <- A %*% Sp %*% t(A)
```

Compute W:

```r
as.numeric(t(y) %*% solve(Sy) %*% y)
[1] 6.584493
```

For Madansky’s LRT, we can use the equations already set up in Table A.21 in Agresti for each of the eight expected frequencies, subject to the constraint of marginal homogeneity.

```r
dummies <- matrix(c(1, 0, 0, 0, 0, 0,
                    -1, 0, 0, 0, 0, 0,
                    -1, 0, 0, 0, 0, 0,
                    1, -1, -1, 0, 1, 0,
                    -1, 0, 0, 1, 0, 0,
                    1, -1, -1, 0, 1, 0,
                    1, 0, -1, -1, 0, 0,
                    0, 0, 0, 0, 0, 1),
                   byrow = T, nr = 8)
```

```r
table.11.2 <- data.frame(counts = table.11.2$counts, dummies)
```

```r
names(table.11.2) <- c("counts", "m111", "m11p", "m1p1", "mp11", "m1pp", "m222")
counts m111 m11p m1p1 mp11 m1pp m222
1 6 1 0 0 0 0 0
2 16 -1 1 0 0 0 0
3 2 -1 0 1 0 0 0
4 4 1 -1 -1 0 1 0
5 2 -1 0 0 1 0 0
6 4 1 -1 0 -1 1 0
7 6 1 0 -1 -1 1 0
8 6 0 0 0 0 0 1
```

Now, perform the fit including all of the dummy variables (the ‘.’ in the formula stands for m111 + m11p + m1p1 + mp11 + m1pp + m222) and excluding an intercept:

```r
fit <- glm(counts ~ . - 1, family = poisson(identity), data = table.11.2)
```

Degrees of Freedom: 8 Total; 2 Residual
Residual Deviance: 5.945079
Cochran’s Q-statistic

If we had the individual responses, we could calculate Cochran’s Q statistic (or the CMH statistic) by setting up n 3x2 matrices like that in Table 11.3 as a 3x2xn array, which is then directly sent into `mantelhaen.test`.

C. Modeling a Repeated Categorical Response

I will use the functions in two Splus libraries to analyze the data in Table 11.4 in Agresti: Rmtools and YAGS. Both are used expressly for repeated categorical (and continuous, Gaussian) responses. Each requires the data to be set up differently.

**RMTOOLS**

Table 11.4 displays data from a repeated binary response (normal/abnormal suffering from mental depression).

```r
table.11.4<-expand.grid(Treatment=c("standard","new"), diagnosis=c("mild","severe"),
                      type=c("NNN","NNA","NAN","NAA","ANN","ANA","AAN","AAA"))
```

In the data frame below, y1, y2, and y3 represent the 0/1 responses at each occasion. The variable, “type”, is actually unnecessary.

```r
temp<-matrix(c(c(0,0,0),c(0,0,1),c(0,1,0),c(0,1,1),c(1,0,0),c(1,0,1),c(1,1,0),
               c(1,1,1)), nc=3, byrow=T)
temp<-temp[rep(1:nrow(temp),rep(4,nrow(temp))),]
table.11.4<-cbind.data.frame(table.11.4,temp)
table.11.4<-cbind.data.frame(table.11.4,
                              counts=c(16,31,2,7,13,0,2,2,9,6,8,5,3,0,9,2,14,22,9,31,4,2,15,5,15,9,27,32,
                                      6,0,28,6))
names(table.11.4)[4:6]<-c("y1","y2","y3")
```

<table>
<thead>
<tr>
<th>Treatment</th>
<th>diagnosis</th>
<th>type</th>
<th>y1</th>
<th>y2</th>
<th>y3</th>
<th>counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>standard</td>
<td>mild</td>
<td>NNN</td>
<td>0</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>new</td>
<td>mild</td>
<td>NNN</td>
<td>0</td>
<td>0</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>standard</td>
<td>severe</td>
<td>NNN</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>new</td>
<td>severe</td>
<td>NNN</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>standard</td>
<td>mild</td>
<td>NNA</td>
<td>0</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>new</td>
<td>mild</td>
<td>NNA</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>standard</td>
<td>severe</td>
<td>NNA</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>new</td>
<td>severe</td>
<td>NNA</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>standard</td>
<td>mild</td>
<td>NAN</td>
<td>0</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>new</td>
<td>mild</td>
<td>NAN</td>
<td>0</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>standard</td>
<td>severe</td>
<td>NAA</td>
<td>0</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>new</td>
<td>severe</td>
<td>NAA</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>standard</td>
<td>mild</td>
<td>NAA</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>new</td>
<td>mild</td>
<td>NAA</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>standard</td>
<td>severe</td>
<td>NAA</td>
<td>0</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>16</td>
<td>new</td>
<td>severe</td>
<td>NAA</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>standard</td>
<td>mild</td>
<td>ANN</td>
<td>1</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>18</td>
<td>new</td>
<td>mild</td>
<td>ANN</td>
<td>1</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>19</td>
<td>standard</td>
<td>severe</td>
<td>ANN</td>
<td>1</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>20</td>
<td>new</td>
<td>severe</td>
<td>ANN</td>
<td>1</td>
<td>0</td>
<td>31</td>
</tr>
<tr>
<td>21</td>
<td>standard</td>
<td>mild</td>
<td>ANN</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>22</td>
<td>new</td>
<td>mild</td>
<td>ANN</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>23</td>
<td>standard</td>
<td>severe</td>
<td>ANN</td>
<td>1</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>24</td>
<td>new</td>
<td>severe</td>
<td>ANN</td>
<td>1</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>25</td>
<td>standard</td>
<td>mild</td>
<td>AAN</td>
<td>1</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>26</td>
<td>new</td>
<td>mild</td>
<td>AAN</td>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>27</td>
<td>standard</td>
<td>severe</td>
<td>AAN</td>
<td>1</td>
<td>1</td>
<td>27</td>
</tr>
</tbody>
</table>
library(rmtools)

To use \texttt{rm.fit}, we must put the data frame into list format, with the counts labeled as "counts" in the list, and the time increments correctly specified. First, I designate the time increments as 1, 2, and 4 weeks. Then, I construct the list.

\begin{verbatim}
time.vec<-c(1,2,4)
table.11.4<-list(y=as.matrix(table.11.4[,4:6]), treatment=factor(table.11.4[,1],
levels=c("new","standard")), diagnosis=factor(table.11.4[,2],
levels=c("severe","mild")), time=time.vec, counts=table.11.4[,7])
\end{verbatim}

Below is the fit for the first model on p. 397. This model contains main effects only. (In the formula specification, I use 1-y as the response variable because of the way I coded an N and A above. I want to model the probability of a \textit{normal} response, not an abnormal response.

\begin{verbatim}
options("contrasts"=c("contr.treatment","contr.poly"))
fit<-rm.fit(I(1-y)~diagnosis+treatment+log(time,2),data=table.11.4,family="binary")
\end{verbatim}

Coefficients:
\begin{verbatim}
Value Std. Error t value
(Intercept) -0.8857 0.1478  -5.994
diagnosis  1.288 0.1468   8.773
treatment -0.8833 0.144  -6.135
log(time, 2)  0.901 0.0901  10.007
\end{verbatim}

Log likelihood:  -596.2552

Here is the fit of the second model at the bottom of p. 397, which adds an occasion x treatment interaction. Again, I used 1-y as the response because of the way I coded an N and A above. I want to model the probability of a \textit{normal} response.

\begin{verbatim}
fit<-rm.fit(I(1-y)~diagnosis+treatment*log(time,2),data=table.11.4,family="binary")
\end{verbatim}

Coefficients:
\begin{verbatim}
Value Std. Error t value
(Intercept) -1.4002 0.1825  -7.672
diagnosis  1.313 0.1481   8.868
treatment  0.06 0.2232   0.264
log(time, 2)  1.499 0.152  10.007
treatment:log(time, 2) -1.016 0.189  -5.387
\end{verbatim}

Log likelihood:  -580.8718

Message: RELATIVE FUNCTION CONVERGENCE

\texttt{rm.fit} fits the model using maximum likelihood subject to constraints on the marginal distributions. Thus, the coefficients are not exactly the same.

Here are the fitted probabilities:

\begin{verbatim}
 junk<-fit$fitted.values
 probs<-junk[1:4,]  # take only the first four
dimnames(probs)<-list(trtcombn=c("mild-std","mild-new","sev-std","sev-new"),
week=c("1","2","3"))
\end{verbatim}
Now, I illustrate the same analyses using library YAGS (Yet Another GEE Solver).

First, set up the data:

```r
table.11.4<-cbind(rep(c("mild","severe"),c(16+13+9+14+15+6+31+0+6+0+22+2+9+0,
2+2+8+9+15+27+28+7+2+5+2+31+5+32+6)),
c(rep(c("standard","new"), c(16+13+9+3+14+15+6,31+0+6+0+22+2+9+0)),
rep(c("standard","new"),c(2+2+8+9+15+27+28,7+2+5+2+31+5+32+6))))
```

```r
resp<-matrix(c(0,0,0, 0,0,1, 0,1,0, 0,1,1, 1,0,0, 1,0,1, 1,1,0, 1,1,1),byrow=T,nc=3)
resp<-rbind(resp,resp,resp,resp)
n<-c(16,13,9,14,4,15,6,31,0,6,0,22,2,9,0,2,2,8,9,9,15,27,28,7,2,5,2,31,5,32,6)
resp<-cbind(resp,n)
```

```r
temp<-matrix(unlist(apply(resp,1,function(x){
  if(x[4]!=0) t(matrix(rep(x[1:3],x[4]),nr=x[4],nc=3,byrow=T))
})),nc=3,byrow=T)
table.11.4<-data.frame(table.11.4,temp)
names(table.11.4)<-c("Diagnosis","Treatment","y1","y2","y3")
```

The first 100 values of the data frame are shown below. Again, y1, y2, and y3 are the three responses at the three occasions.
<table>
<thead>
<tr>
<th></th>
<th>mild</th>
<th>standard</th>
<th>new</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>mild</td>
<td>standard</td>
<td>0 1 0</td>
</tr>
<tr>
<td>35</td>
<td>mild</td>
<td>standard</td>
<td>0 1 0</td>
</tr>
<tr>
<td>36</td>
<td>mild</td>
<td>standard</td>
<td>0 1 0</td>
</tr>
<tr>
<td>37</td>
<td>mild</td>
<td>standard</td>
<td>0 1 0</td>
</tr>
<tr>
<td>38</td>
<td>mild</td>
<td>standard</td>
<td>0 1 0</td>
</tr>
<tr>
<td>39</td>
<td>mild</td>
<td>standard</td>
<td>0 1 1</td>
</tr>
<tr>
<td>40</td>
<td>mild</td>
<td>standard</td>
<td>0 1 1</td>
</tr>
<tr>
<td>41</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>42</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>43</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>44</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>45</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>46</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>47</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>48</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>49</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>50</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>51</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>52</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>53</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>54</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>55</td>
<td>mild</td>
<td>standard</td>
<td>1 0 0</td>
</tr>
<tr>
<td>56</td>
<td>mild</td>
<td>standard</td>
<td>1 0 1</td>
</tr>
<tr>
<td>57</td>
<td>mild</td>
<td>standard</td>
<td>1 0 1</td>
</tr>
<tr>
<td>58</td>
<td>mild</td>
<td>standard</td>
<td>1 0 1</td>
</tr>
<tr>
<td>59</td>
<td>mild</td>
<td>standard</td>
<td>1 0 1</td>
</tr>
<tr>
<td>60</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>61</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>62</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>63</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>64</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>65</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>66</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>67</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>68</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>69</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>70</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>71</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>72</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>73</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>74</td>
<td>mild</td>
<td>standard</td>
<td>1 1 0</td>
</tr>
<tr>
<td>75</td>
<td>mild</td>
<td>standard</td>
<td>1 1 1</td>
</tr>
<tr>
<td>76</td>
<td>mild</td>
<td>standard</td>
<td>1 1 1</td>
</tr>
<tr>
<td>77</td>
<td>mild</td>
<td>standard</td>
<td>1 1 1</td>
</tr>
<tr>
<td>78</td>
<td>mild</td>
<td>standard</td>
<td>1 1 1</td>
</tr>
<tr>
<td>79</td>
<td>mild</td>
<td>standard</td>
<td>1 1 1</td>
</tr>
<tr>
<td>80</td>
<td>mild</td>
<td>standard</td>
<td>1 1 1</td>
</tr>
<tr>
<td>81</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>82</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>83</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>84</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>85</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>86</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>87</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>88</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>89</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>90</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>91</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>92</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>93</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>94</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>95</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>96</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>97</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>98</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>99</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
<tr>
<td>100</td>
<td>mild</td>
<td>new</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>
Now, I have to add the subject “id” and “week” factors. “id” indicates the subject tested. The “week” number is either 1, 2, or 4.

```r
temp <- table.11.4[rep(1:nrow(table.11.4), rep(3, nrow(table.11.4))), 1:2]
row.names(temp) <- 1:nrow(temp)
table.11.4.yags <- cbind.data.frame(temp, t(data.matrix(table.11.4)[, 3:5]))

# add id factor:

add.id.factor <- cbind.data.frame(rep(1:nrow(table.11.4), rep(3, nrow(table.11.4))),
                                   table.11.4.yags)

# add time (in weeks) factor:

add.week.factor <- cbind.data.frame(add.id.factor, rep(c(1, 2, 4), nrow(table.11.4)))
names(add.week.factor)[c(1, 4, 5)] <- c("id", "suffering", "week")

Here are the first 100 values:

<table>
<thead>
<tr>
<th>id</th>
<th>Diagnosis</th>
<th>Treatment</th>
<th>suffering</th>
<th>week</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>19</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>21</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>22</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>24</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>25</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>27</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>28</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>30</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>31</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>33</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>34</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>35</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>36</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>37</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>38</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>39</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>40</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>41</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>42</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>43</td>
<td>mild</td>
<td>standard</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
```
Now, fit the second model:

```r
attach(table.11.4.yags)
time<-log(week,2)
library(yags)
```
fit<-yags(I(1-suffering)~factor(Diagnosis,levels=c("severe","mild"))
+factor(Treatment,levels=c("new","standard"))*time,id=id,cor.met=week,corstr="unstructured",family=binomial)

summary(fit)

Coefficients:

<table>
<thead>
<tr>
<th>Estimate</th>
<th>Naive S.E.</th>
<th>Naive z</th>
<th>Robust S.E.</th>
<th>Robust z</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>-1.38484307</td>
<td>0.1836080</td>
<td>-7.5423910</td>
<td>-7.8141883</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>1.30489234</td>
<td>0.1448783</td>
<td>9.0068171</td>
<td>8.9980583</td>
</tr>
<tr>
<td>Treatment</td>
<td>0.05441048</td>
<td>0.2261665</td>
<td>0.2405771</td>
<td>0.2395461</td>
</tr>
<tr>
<td>time</td>
<td>1.48890188</td>
<td>0.1493473</td>
<td>9.9693953</td>
<td>10.3392898</td>
</tr>
<tr>
<td>Treatment:time</td>
<td>-1.01299682</td>
<td>0.1870708</td>
<td>-5.4150437</td>
<td>-5.4302367</td>
</tr>
</tbody>
</table>

Estimated Scale Parameter: 0.982367
Number of Iterations: 2

And, I obtain the fitted values as follows:

(fitted(fit))[1:3,]
[1] 0.2002323 0.5259913 0.8310322

(fitted(fit))[340:342,]
[1] 0.2090878 0.2984840 0.4064592

(fitted(fit))[679:681,]
[1] 0.4800230 0.8036005 0.9477402

fitted(fit)[1018:1020,]
[1] 0.4936153 0.6107260 0.7163179

D. Modeling a Repeated Ordinal Response

Agresti uses the Insomnia data on p. 402 (Table 11.7) to illustrate modeling a repeated ordinal response. As with the ordered mean response models in Chapter 9, I could not find a package that could implement both the ordinal and repeated nature of the response. Thus, I compute the weighted least squares estimates of the coefficients in (11.11) directly using the development on pages 458-462 of Agresti.

First, I set up the data and sample proportion vectors for each Occasion x Treatment combination.

table.11.7<-
| matrix(c(7,4,1,0,11,5,2,2,13,23,3,1,9,17,13,8,7,4,2,1,14,5,1,0,6,9,18,2,4,11,14,22),nc=4, nr=8, byrow=T)
p1<-(rowSums(table.11.7[1:4,1:4])/sum(table.11.7[1:4,1:4]))
p2<-(colSums(table.11.7[1:4,1:4])/sum(table.11.7[1:4,1:4]))
p3<-(rowSums(table.11.7[5:8,1:4])/sum(table.11.7[5:8,1:4]))
p4<-(colSums(table.11.7[5:8,1:4])/sum(table.11.7[5:8,1:4]))
n1<-119; n2<-n1; n3<-120; n4<-n3 # sample sizes for each “population”
n<-c(n1,n2,n3,n4)
J<-4 # We have four response variables
I<-4 # We have four partly dependent multinomial samples

The 4 x 1 vector \( \mathbf{p} \) represents the sample distribution of the response for the ith population. The populations are ordered 1 through 4 according to their presentation in Table 11.8 on p. 403. Thus, group Active/Initial is group 1, and Passive/Follow-up is group 4.
The variance-covariance matrix on p. 459 in Agresti is modified here to account for the fact that the first and second populations and the third and fourth populations are dependent (they are the same individuals). Thus, the covariance matrix becomes

\[
V = \begin{pmatrix}
V_1 & V_{12} & 0 & 0 \\
V_{12} & V_2 & 0 & 0 \\
0 & 0 & V_3 & V_{34} \\
0 & 0 & V_{34} & V_4
\end{pmatrix}
\]

where \( V_i \) is as described on p. 459 in Agresti, and

\[
n_i V_{ii} = \begin{pmatrix}
\text{cov}(y_{1j}, y_{1j'}) & \text{cov}(y_{1j}, y_{2j'}) & \text{cov}(y_{1j}, y_{3j'}) & \text{cov}(y_{1j}, y_{4j'}) \\
\text{cov}(y_{2j}, y_{1j'}) & \text{cov}(y_{2j}, y_{2j'}) & \text{cov}(y_{2j}, y_{3j'}) & \text{cov}(y_{2j}, y_{4j'}) \\
\text{cov}(y_{3j}, y_{1j'}) & \text{cov}(y_{3j}, y_{2j'}) & \text{cov}(y_{3j}, y_{3j'}) & \text{cov}(y_{3j}, y_{4j'}) \\
\text{cov}(y_{4j}, y_{1j'}) & \text{cov}(y_{4j}, y_{2j'}) & \text{cov}(y_{4j}, y_{3j'}) & \text{cov}(y_{4j}, y_{4j'})
\end{pmatrix}
\]

where \( i' \neq i \). The covariance terms in the above matrix are

\[
\text{cov}(y_{kj}, y_{kj'}) = E(y_{kj}y_{kj'}) - E(y_{kj})E(y_{kj'}) = \Pr(y_{kj} = 1, y_{kj'} = 1) - \Pr(y_{kj} = 1)\Pr(y_{kj'} = 1)
\]

These terms can be estimated using the sample proportions.

Thus, the Splus commands to input the covariance matrix \( V \) are

```splus
Vnames<-numeric(length=I)
pnames<-Vnames
for(i in 1:I)
{
    Vnames[i] <- paste("V", i, collapse = " ", sep = "")
pnames[i]<-paste("p",i,collapse=" ",sepe="")
V<-matrix(0,nc=J,nr=J)
p<-eval(parse(text = pnames[i]))
diag(V)<-p*(1-p)
p<-as.matrix(p)
junk<-matrix(-kronecker(p,p),nc=J,nr=J,byrow=T)
V[lower.tri(diag(J))]<-junk[lower.tri(junk)]
V<-t(V)
V[lower.tri(V)]<-junk[lower.tri(junk,diag=F)]
assign(Vnames[i], matrix(V/n[i], ncol = J, byrow = T))
}
# Compute V12
active<-table.11.7[1:4,1:4]
active<-sweep(active,1,rowSums(active),FUN="/")
p<-cbind(p1,p2,p3,p4)
V12<-matrix(c(p1[1]*(active[1,]-p[1,2]),p1[2]*(active[2,]-p[1,2]),p1[3]*(active[3,]-
p[2,2]),p1[4]*(active[4,]-p[2,2])), nc=J, nr=J, byrow=T)
V12<-V12/n1
# Compute V34
passive<-table.11.7[5:8,1:4]
```
passive<-sweep(passive,1,rowSums(passive),FUN="/"
V34<-V34/n3
# Compute V
zero<-matrix(0,J,J)
V<-rbind(
cbind(V1,V12,zero,zero),
cbind(t(V12),V2,zero,zero),
cbind(zero,zero,V3,V34),
cbind(zero,zero,t(V34),V4))
The objects Vnames and pnames have the same usage as in Section 9.d.
The model (equation 13.11) for this example encompasses the 12 equations

\[
F_1(\pi) = \alpha_1 - \beta_1^0 - \beta_1^T - \gamma_{11} = \log\left(\frac{\pi_{11}}{\pi_{11}+\pi_{12}+\pi_{14}}\right)
\]

\[
F_2(\pi) = \alpha_2 - \beta_1^0 - \beta_2^T - \gamma_{11} = \log\left(\frac{\pi_{12}}{\pi_{12}+\pi_{13}+\pi_{14}}\right)
\]

\[
\vdots
\]

\[
F_{12}(\pi) = \alpha_5 = \log\left(\frac{\pi_{44}}{\pi_{14}+\pi_{24}+\pi_{34}}\right),
\]

one for each of the four treatment combinations and three response cutoffs. The matrix \( Q \) of first derivatives of the response functions \( F \), has the following form

\[
Q = \begin{pmatrix}
\frac{\partial}{\partial \pi_{11}} F_1(\pi) & \cdots & \frac{\partial}{\partial \pi_{14}} F_1(\pi) \\
\vdots & \ddots & \vdots \\
\frac{\partial}{\partial \pi_{11}} F_{12}(\pi) & \cdots & \frac{\partial}{\partial \pi_{44}} F_{12}(\pi)
\end{pmatrix}
\]

and is straightforward to compute using the function deriv or deriv3 available from the MASS library. The response functions and the matrix of derivatives can be estimated using sample values. Here are the Splus commands to compute them for this example.

library(MASS)
# Definitions of the derivative functions in the Q matrix:
d1F<-deriv3(~log(x/(a+b+k)),c("x"),function(x,a,b,k){},hessian=F)
d2F<-deriv3(~log(a/(x+b+k)),c("x"),function(x,a,b,k){},hessian=F)
d3F<-deriv3(~log((x+a)/(b+k)),c("x"),function(x,a,b,k){},hessian=F)
d4F<-deriv3(~log((b+a)/(x+k)),c("x"),function(x,a,b,k){},hessian=F)
d5F<-deriv3(~log((x+a+b)/k),c("x"),function(x,a,b,k){},hessian=F)
d6F<-deriv3(~log((k+a+b)/x),c("x"),function(x,a,b,k){},hessian=F)

# Now, we plug in the specific estimated probabilities:
Q<-c(
as.numeric(attr(d1F(p1[1],p1[2],p1[3],p1[4]),"gradient")),
as.numeric(attr(d2F(p1[2],p1[1],p1[3],p1[4]),"gradient")),
...
as.numeric(attr(d2F(p1[3], p1[1], p1[2], p1[4]), "gradient")),
as.numeric(attr(d2F(p1[4], p1[1], p1[2], p1[3]), "gradient")),
rep(0, 12),
as.numeric(attr(d3F(p1[1], p1[2], p1[3], p1[4]), "gradient")),
as.numeric(attr(d3F(p1[2], p1[1], p1[3], p1[4]), "gradient")),
as.numeric(attr(d4F(p1[3], p1[1], p1[2], p1[4]), "gradient")),
as.numeric(attr(d4F(p1[4], p1[1], p1[2], p1[3]), "gradient")),
rep(0, 12),
as.numeric(attr(d5F(p1[1], p1[2], p1[3], p1[4]), "gradient")),
as.numeric(attr(d5F(p1[2], p1[1], p1[3], p1[4]), "gradient")),
as.numeric(attr(d5F(p1[3], p1[1], p1[2], p1[4]), "gradient")),
as.numeric(attr(d5F(p1[4], p1[1], p1[2], p1[3]), "gradient")),
rep(0, 12),
ap(0, 4),
as.numeric(attr(d1F(p2[1], p2[2], p2[3], p2[4]), "gradient")),
as.numeric(attr(d2F(p2[2], p2[1], p2[3], p2[4]), "gradient")),
as.numeric(attr(d2F(p2[3], p2[1], p2[2], p2[4]), "gradient")),
as.numeric(attr(d2F(p2[4], p2[1], p2[2], p2[3]), "gradient")),
rep(0, 8),
ap(0, 4),
as.numeric(attr(d3F(p2[1], p2[2], p2[3], p2[4]), "gradient")),
as.numeric(attr(d3F(p2[2], p2[1], p2[3], p2[4]), "gradient")),
as.numeric(attr(d4F(p2[3], p2[1], p2[2], p2[4]), "gradient")),
as.numeric(attr(d4F(p2[4], p2[1], p2[2], p2[3]), "gradient")),
rep(0, 8),
ap(0, 4),
as.numeric(attr(d5F(p2[1], p2[2], p2[3], p2[4]), "gradient")),
as.numeric(attr(d5F(p2[2], p2[1], p2[3], p2[4]), "gradient")),
as.numeric(attr(d5F(p2[3], p2[1], p2[2], p2[4]), "gradient")),
as.numeric(attr(d5F(p2[4], p2[1], p2[2], p2[3]), "gradient")),
rep(0, 8),
ap(0, 4),
as.numeric(attr(d1F(p3[1], p3[2], p3[3], p3[4]), "gradient")),
as.numeric(attr(d2F(p3[2], p3[1], p3[3], p3[4]), "gradient")),
as.numeric(attr(d2F(p3[3], p3[1], p3[2], p3[4]), "gradient")),
as.numeric(attr(d2F(p3[4], p3[1], p3[2], p3[3]), "gradient")),
rep(0, 4),
ap(0, 8),
as.numeric(attr(d3F(p3[1], p3[2], p3[3], p3[4]), "gradient")),
as.numeric(attr(d3F(p3[2], p3[1], p3[3], p3[4]), "gradient")),
as.numeric(attr(d4F(p3[3], p3[1], p3[2], p3[4]), "gradient")),
as.numeric(attr(d4F(p3[4], p3[1], p3[2], p3[3]), "gradient")),
rep(0, 8),
ap(0, 8),
as.numeric(attr(d5F(p3[1], p3[2], p3[3], p3[4]), "gradient")),
as.numeric(attr(d5F(p3[2], p3[1], p3[3], p3[4]), "gradient")),
as.numeric(attr(d5F(p3[3], p3[1], p3[2], p3[4]), "gradient")),
as.numeric(attr(d6F(p3[4], p3[1], p3[2], p3[3]), "gradient")),
rep(0, 8),
ap(0, 8),
as.numeric(attr(d1F(p4[1], p4[2], p4[3], p4[4]), "gradient")),
as.numeric(attr(d2F(p4[2], p4[1], p4[3], p4[4]), "gradient")),
as.numeric(attr(d2F(p4[3], p4[1], p4[2], p4[4]), "gradient")),
rep(0, 12),
as.numeric(attr(d2F(p4[4], p4[1], p4[2], p4[3]), "gradient")),
rep(0, 12),
as.numeric(attr(d3F(p4[1], p4[2], p4[3], p4[4]), "gradient")),
as.numeric(attr(d3F(p4[2], p4[1], p4[3], p4[4]), "gradient")),
as.numeric(attr(d4F(p4[3], p4[1], p4[2], p4[4]), "gradient")),
as.numeric(attr(d4F(p4[4], p4[1], p4[2], p4[3]), "gradient")),
rep(0, 12),
as.numeric(attr(d5F(p4[1], p4[2], p4[3], p4[4]), "gradient")),
as.numeric(attr(d5F(p4[2], p4[1], p4[3], p4[4]), "gradient")),
as.numeric(attr(d5F(p4[3], p4[1], p4[2], p4[4]), "gradient")),
as.numeric(attr(d6F(p4[4], p4[1], p4[2], p4[3]), "gradient"))

Q <- matrix(Q, nr=12, nc=16, byrow=T)

# Definitions of the response functions, afterward evaluated at the sample proportions:
f1 <- function(x, a, b, k) log(x/(a+b+k))
f2 <- function(x, a, b, k) log((x+a)/(b+k))
f3 <- function(x, a, b, k) log((x+a+b)/k)

Fp <- c(f1(p1[1], p1[2], p1[3], p1[4]), f2(p1[1], p1[2], p1[3], p1[4]), f3(p1[1], p1[2], p1[3], p1[4]), f1(p2[1], p2[2], p2[3], p2[4]), f2(p2[1], p2[2], p2[3], p2[4]), f3(p2[1], p2[2], p2[3], p2[4]), f1(p3[1], p3[2], p3[3], p3[4]), f2(p3[1], p3[2], p3[3], p3[4]), f3(p3[1], p3[2], p3[3], p3[4]), f1(p4[1], p4[2], p4[3], p4[4]), f2(p4[1], p4[2], p4[3], p4[4]), f3(p4[1], p4[2], p4[3], p4[4]))

The transformed covariance matrix is estimated as before:
VF <- Q %*% V %*% t(Q)

The design matrix reflects the model equation, \( F(\pi) = X\beta \), with \( \beta = (\alpha_1, \alpha_2, \alpha_3, \beta_1^T, \beta_2^T, \gamma_1^T)^T \).

X <- rbind(diag(3), diag(3), diag(3), diag(3))
X <- cbind(X, rep(rep(c(-1, 0), c(3, 3)), 2), rep(c(-1, 0), c(6, 6)), rep(c(-1, 0), c(3, 9)))

To estimate beta, we use the weighted least squares formula:

InvVF <- solve(VF)
Covb <- solve(t(X) %*% InvVF %*% X)
b <- InvVF %*% t(X) %*% solve(VF) %*% Fp

The residual chi-squared statistic is then computed as
\( t(Fp-X\times b) \times \text{InvVF} \times (Fp-X\times b) \)

\[
[1] \quad 7.417129
\]

The ASEs for all estimates are

\[
\text{sqrt(diag(Covb))}
\]

\[
[1] \quad 0.1728720 \quad 0.1705049 \quad 0.1829844 \quad 0.1604752 \quad 0.2345762 \quad 0.2460542
\]

Obtaining predicted values requires just a little more thought than usual. Because we are modeling cumulative logits, the probabilities estimated by

\[
cprob<-\exp(X\times b)/(1+\exp(X\times b))
\]

are the cumulative probabilities: \( \Pr(y \leq j \mid x) \). So, to get the probabilities \( \Pr(y = j \mid x) \), we just take successive differences of cumulative probabilities where appropriate.

\[
\begin{align*}
\text{prob} & \leftarrow \text{matrix}(c(cprob[1], \text{diff(cprob[1:3])}, 1-cprob[3], \text{cprob[4]}, \text{diff(cprob[4:6])}, 1-\text{cprob[6]}), \text{byrow}=\text{T}, \text{nc}=4) \\
\text{prob2} & \leftarrow \text{matrix}(c(cprob[7], \text{diff(cprob[7:9])}, 1-cprob[9], \text{cprob[10]}, \text{diff(cprob[10:12])}, 1-\text{cprob[12]}), \text{byrow}=\text{T}, \text{nc}=4) \\
\text{prob} & \leftarrow \text{array}(\text{c(prob, prob2)}, \text{dim}=c(2, 4, 2)) \\
\text{dimnames(prob)} & \leftarrow \text{list}(\text{c("Initial", "Follow-up"), c("<20", "20-30", "30-60", ">60"), e("Active", "Passive"))}
\end{align*}
\]

\[
\begin{array}{cccc}
& <20 & 20-30 & 30-60 & >60 \\
\text{Initial} & 0.1020006 & 0.1837122 & 0.3028913 & 0.4113959 \\
\text{Follow-up} & 0.3854300 & 0.3029012 & 0.1993055 & 0.1123633 \\
\end{array}
\]

\[
\begin{array}{cccc}
& <20 & 20-30 & 30-60 & >60 \\
\text{Initial} & 0.09839743 & 0.1792289 & 0.3012663 & 0.4211074 \\
\text{Follow-up} & 0.23851225 & 0.2859779 & 0.2732992 & 0.2022106 \\
\end{array}
\]

**E. Markov Chain Models**

Agresti fits several Markov chain models to the respiratory illness data in Table 11.9. These can be fit as ordinary log-linear models with a function like \text{glm}.

\[
\text{table.11.9} \leftarrow \text{expand.grid(twelve=(c(1,2)), eleven=(c(1,2)), ten=(c(1,2)), nine=(c(1,2)))} \\
\text{table.11.9} \leftarrow \text{cbind.data.frame(table.11.9, counts=c(94, 30, 15, 28, 14, 9, 12, 63, 19, 15, 10, 44, 17, 42, 35, 572 )))}
\]

\[
\begin{array}{cccc}
\text{twelve} & \text{eleven} & \text{ten} & \text{nine} & \text{counts} \\
1 & 1 & 1 & 1 & 94 \\
2 & 2 & 1 & 1 & 30 \\
3 & 1 & 2 & 1 & 15 \\
4 & 2 & 2 & 1 & 28 \\
5 & 1 & 1 & 2 & 14 \\
6 & 2 & 1 & 2 & 9 \\
7 & 1 & 2 & 2 & 12 \\
8 & 2 & 2 & 2 & 63 \\
9 & 1 & 1 & 2 & 19 \\
10 & 2 & 1 & 1 & 15 \\
11 & 1 & 2 & 1 & 10 \\
\end{array}
\]
Test a first-order model:

```
glm(counts~nine*ten+ten*eleven+eleven*twelve, data=table.11.9, family=poisson(log))
```

Degrees of Freedom: 16 Total; 8 Residual  
Residual Deviance: 122.9025

Test a second-order model:

```
glm(counts~nine*ten*eleven+ten*eleven*twelve, data=table.11.9, family=poisson(log))
```

Degrees of Freedom: 16 Total; 4 Residual  
Residual Deviance: 23.86324

Test a model that permits all pairwise associations:

```
options(contrasts=c("contr.sum","contr.poly"))
glm(counts~nine*nine+eleven+nine*eleven*twelve+ten*eleven+ten*twelve+eleven*twelve, data=table.11.9, family=poisson(log))
```

Coefficients:

<table>
<thead>
<tr>
<th>Term</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>19.50572</td>
</tr>
<tr>
<td>nine</td>
<td>-5.391511</td>
</tr>
<tr>
<td>ten</td>
<td>-6.488897</td>
</tr>
<tr>
<td>eleven</td>
<td>-6.244144</td>
</tr>
<tr>
<td>twelve</td>
<td>-5.207749</td>
</tr>
<tr>
<td>nine:ten</td>
<td>1.806361</td>
</tr>
<tr>
<td>nine:eleven</td>
<td>0.9478092</td>
</tr>
<tr>
<td>nine:twelve</td>
<td>1.053104</td>
</tr>
<tr>
<td>ten:eleven</td>
<td>1.645769</td>
</tr>
<tr>
<td>ten:twelve</td>
<td>1.074206</td>
</tr>
<tr>
<td>eleven:twelve</td>
<td>1.849747</td>
</tr>
</tbody>
</table>

Degrees of Freedom: 16 Total; 5 Residual  
Residual Deviance: 1.458497

Test a model that restricts pairwise associations to be one of two values (high or low) depending on whether the association is first- or second-order:

I first change the coding in table.11.9 to be 0/1 instead of 1/2:

```
table.11.9[,1:4] <- table.11.9[,1:4] - 1
attach(table.11.9)
```

The objects pairs and thirdpairs represent the common effects of first- and second-order associations.

```
pairs <- cbind((nine*ten), (ten*eleven), (eleven*twelve))
thirdpairs <- cbind((ten*twelve), nine*eleven, nine*twelve)
pairs <- as.numeric(rowSums(pairs))
thirdpairs <- as.numeric(rowSums(thirdpairs))
glm(counts=nine + ten + eleven + twelve + I(thirdpairs) + I(pairs), data=table.11.9, family=poisson(log))
```

Coefficients:

<table>
<thead>
<tr>
<th>Term</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>4.546055</td>
</tr>
<tr>
<td>nine</td>
<td>-1.596589</td>
</tr>
<tr>
<td>ten</td>
<td>-1.970608</td>
</tr>
<tr>
<td>eleven</td>
<td>-1.853073</td>
</tr>
<tr>
<td>twelve</td>
<td>-1.13977</td>
</tr>
<tr>
<td>I(thirdpairs)</td>
<td>1.035713</td>
</tr>
<tr>
<td>I(pairs)</td>
<td>1.75203</td>
</tr>
</tbody>
</table>
Degrees of Freedom: 16 Total; 9 Residual
Residual Deviance: 2.268529

The estimated log odds ratios are highlighted.