APPENDIX:

Software Details for Examples in *Categorical Data Analysis*

In this appendix we provide details about how to use R, SAS, Stata, and SPSS statistical software for categorical data analysis, illustrating for the examples in the text. This supplements the brief description found in Appendix A of the text *Categorical Data Analysis* by Alan Agresti, 3rd edition, published by Wiley, 2012). For each package, the material is organized by chapter of presentation and refers to datasets analyzed in those chapters. For convenience, data for examples are entered in the form of the contingency table displayed in the text. In practice, the data would usually be entered at the subject level. The full data sets are available at

`www.stat.ufl.edu/~aa/cda/cda.html`

A.1 SAS EXAMPLES

SAS is general-purpose software for a wide variety of statistical analyses. The main procedures (PROCs) for categorical data analyses are FREQ, GENMOD, LOGISTIC, NLMIXED, GLIMMIX, and CATMOD. PROC FREQ performs basic analyses for two-way and three-way contingency tables. PROC GENMOD fits generalized linear models using ML or Bayesian methods, cumulative link models for ordinal responses, zero-inflated Poisson regression models for count data, and GEE analyses for marginal models. PROC LOGISTIC gives ML fitting of binary response models, cumulative link models for ordinal responses, and baseline-category logit models for nominal responses. (PROC SURVEYLOGISTIC fits binary and multi-category regression models to survey data by incorporating the sample design into the analysis and using the method of pseudo ML.) PROC CATMOD fits baseline-category logit models and can fit a variety of other models using weighted least squares. PROC NLMIXED gives ML fitting of generalized linear mixed models, using adaptive Gauss–Hermite quadrature. PROC GLIMMIX also fits such models with a variety of fitting methods.

The examples in this appendix show SAS code for version 9.22. We focus on basic model fitting rather than the great variety of options. For more detail, see

Stokes, Davis, and Koch (2012) *Categorical Data Analysis Using SAS*, 3rd ed. Cary, NC: SAS Institute.

Allison (1999) *Logistic Regression Using the SAS System.* Cary, NC: SAS Institute.

For examples of categorical data analyses with SAS for many data sets in my text *An Introduction to Categorical Data Analysis*, see the useful site

`www.ats.ucla.edu/stat/examples/icda/`

set up by the UCLA Statistical Computing Center. A useful SAS site on-line with details about the options as well as many examples for each PROC is at

`support.sas.com/rnd/app/da/stat/procedures/CategoricalDataAnalysis.html`.

In SAS, The @@ symbol in an input line indicates that each line of data contains more than one observation. Input of a variable as characters rather than numbers requires an accompanying $ label in the INPUT statement.

Chapter 1: Introduction

With PROC FREQ for a 1×2 table of counts of successes and failures for a binomial variate, confidence limits for the binomial proportion include Agresti-Coull, Jeffreys (Bayes), score (Wilson), and Clopper–Pearson exact method. The keyword BINOMIAL and the EXACT statement yields binomial tests. Table 1 shows code for confidence intervals for the example in the text section 1.4.3 about estimating the proportion of people who are vegetarians, when 0 of 25 in a sample are vegetarian.

Table 1: SAS Code for Confidence Intervals for a Proportion

```
---------------------------------------------------------------------------------
data veg;
input response $ count;
datalines;
no   25
yes  0
;
proc freq data=veg; weight count;
tables response / binomial(ac wilson exact jeffreys) alpha=.05;
run;
---------------------------------------------------------------------------------
```

Chapters 2–3: Two-Way Contingency Tables

Table 2 uses SAS to analyze Table 3.2 in *Categorical Data Analysis*, on education and belief in God. PROC FREQ forms the table with the TABLES statement, ordering row and column categories alphanumerically. To use instead the order in which the categories appear in the data set (e.g., to treat the variable properly in an ordinal analysis), use the ORDER = DATA option in the PROC statement. The WEIGHT statement is needed when you enter the cell counts instead of subject-level data. PROC FREQ can conduct chi-squared tests of independence (CHISQ option), show its estimated expected frequencies (EXPECTED), provide a wide assortment of measures of association and their standard errors (MEASURES), and provide ordinal statistic (3.16) with a "nonzero correlation" test (CMH1). You can also perform chi-squared tests using PROC GENMOD (using loglinear models discussed in Chapters 9-10), as shown. Its RESIDUALS option provides cell residuals. The output labeled "StReschi" is the standardized residual.

For creating mosaic plots in SAS, see `www.datavis.ca` and `www.datavis.ca/books/vcd/`.

Table 3 analyzes the tea tasting data in Table 3.9 of the textbook. With PROC FREQ, for 2×2 tables the MEASURES option in the TABLES statement provides confidence intervals for the odds ratio (labeled "case-control" on output) and the relative risk, and the RISKDIFF option provides intervals for the proportions and their difference. For tables having small cell counts, the EXACT statement can provide various exact analyses. These include Fisher's exact test and its generalization for $I \times J$ tables, treating variables as nominal, with keyword FISHER. The OR keyword

Table 2: **SAS Code for Chi-Squared, Measures of Association, and Residuals for Data on Education and Belief in God in Table 3.2**

```
-----------------------------------------------------------------------------
data table;
    input degree belief $ count @@;
datalines;
1 1  9   1 2  8   1 3 27   1 4  8   1 5  47   1 6 236
2 1 23   2 2 39   2 3 88   2 4 49   2 5 179   2 6 706
3 1 28   3 2 48   3 3 89   3 4 19   3 5 104   3 6 293
  ;
proc freq  order=data; weight count;
  tables degree*belief / chisq expected measures cmh1;
proc genmod  order=data;  class degree belief;
    model count = degree belief / dist=poi link=log residuals;
-----------------------------------------------------------------------------
```

gives the odds ratio and its large-sample Wald confidence interval based on (3.2) and the small-sample interval based on the noncentral hypergeometric distribution (16.28). Other EXACT statement keywords include unconditional exact confidence limits for the difference of proportions (keyword RISKDIFF), exact trend tests for $I \times 2$ tables (TREND), and exact chi-squared tests (CHISQ) and exact correlation tests for $I \times J$ tables (MHCHI). You can use Monte Carlo simulation (option MC) to estimate exact P-values when the exact calculation is too time-consuming. Table 3 also uses PROC LOGISTIC to get a profile-likelihood confidence interval for the odds ratio (CLODDS = PL). PROC LOGISTIC uses FREQ to weight counts, serving the same purpose for which PROC FREQ uses WEIGHT.

Table 3: **SAS Code for Fisher's Exact Test and Confidence Intervals for Odds Ratio for Tea-Tasting Data in Table 3.9**

```
-----------------------------------------------------------------------------
data fisher;
input poured guess count @@;
datalines;
1 1 3    1 2 1    2 1 1    2 2 3
;
proc freq;  weight count;
   tables poured*guess / measures riskdiff;
   exact fisher or / alpha=.05;
proc logistic descending;  freq count;
    model guess = poured / clodds=pl;
-----------------------------------------------------------------------------
```

Chapter 4: Generalized Linear Models

PROC GENMOD fits GLMs. It specifies the response distribution in the DIST option ("poi" for Poisson, "bin" for binomial, "mult" for multinomial, "negbin" for negative binomial) and specifies the link in the LINK option. For binomial models with grouped data, the response in the model statements takes the form of the number of "successes" divided by the number of cases. Table 4 illustrates for the snoring data in Table 4.2 of the textbook. Profile likelihood confidence intervals are provided in PROC GENMOD with the LRCI option.

Table 4: **SAS Code for Binary GLMs for Snoring Data in Table 4.2**

```
-------------------------------------------------------------------------
data glm;
input snoring disease total @@;
datalines;
0 24 1379    2 35 638    4 21 213    5 30 254
;
proc genmod; model disease/total = snoring  /  dist=bin  link=identity;
proc genmod; model disease/total = snoring  /  dist=bin  link=logit;
proc genmod; model disease/total = snoring  /  dist=bin  link=probit;
-------------------------------------------------------------------------
```

Table 5 uses PROC GENMOD for count modeling of the horseshoe crab data in Table 4.3 of the textbook. Each observation refers to a single crab. Using width as the predictor, the first two models use Poisson regression and the third model assumes a negative binomial distribution.

Table 6 uses PROC GENMOD for the overdispersed data of Table 4.7 of the textbook. A CLASS statement requests indicator (dummy) variables for the groups. With no intercept in the model (option NOINT) for the identity link, the estimated parameters are the four group probabilities. The ESTIMATE statement provides an estimate, confidence interval, and test for a contrast of model parameters, in this case the difference in probabilities for the first and second groups. The second analysis uses the Pearson statistic to scale standard errors to adjust for overdispersion. PROC LOGISTIC can also provide overdispersion modeling of binary responses; see Table 29 in the Chapter 14 part of this appendix for SAS.

The final PROC GENMOD run in Table 7 fits the Poisson regression model with log link for the grouped data of Tables 4.4 and 5.2. It models the total number of satellites at each width level (variable "satell"), using the log of the number of cases as offset.

Chapters 5–7: Logistic Regression and Binary Response Analyses

You can fit logistic regression models using either software for GLMs or specialized software for logistic regression. PROC GENMOD uses Newton-Raphson, whereas

Table 5: **SAS Code for Poisson and Negative Binomial GLMs for Horseshoe Crab Data in Table 4.3**

```
---------------------------------------------------------------
data crab;
input color spine width satell weight;
datalines;
3  3  28.3  8  3.05
4  3  22.5  0  1.55
...
3  2  24.5  0  2.00
;
proc genmod;
   model satell = width / dist=poi link=log ;
proc genmod;
   model satell = width / dist=poi link=identity ;
proc genmod;
   model satell = width / dist=negbin link=identity ;
----------------------------------------------------------------
```

Table 6: **SAS Code for Overdispersion Modeling of Teratology Data in Table 4.7**

```
-------------------------------------------------------------------------
data moore;
  input litter group n y @@;
datalines;
1  1 10 1    2 1 11 4    3 1 12 9     4 1  4 4     5 1 10 10
...
55 4 14 1   56 4  8 0   57 4  6 0    58 4 17 0
;
proc genmod;  class group;
  model y/n = group / dist=bin link=identity noint;
estimate 'pi1-pi2' group 1 -1 0 0;
proc genmod;  class group;
  model y/n = group / dist=bin link=identity noint scale=pearson;
-------------------------------------------------------------------------
```

PROC LOGISTIC uses Fisher scoring. Both yield ML estimates, but the *SE* values use the inverted observed information matrix in PROC GENMOD and the inverted expected information matrix in PROC LOGISTIC. These are the same for the logit link because it is the canonical link function for the binomial, but differ for other links.

Table 7 applies PROC GENMOD and PROC LOGISTIC to Table 5.2 of the textbook, when “y” out of “n” crabs had satellites at a given width level. Profile likelihood confidence intervals are provided in PROC GENMOD with the LRCI option and in PROC LOGISTIC with the PLCL option. In PROC GENMOD, the ALPHA = option can specify an error probability other than the default of 0.05, and the TYPE3 option provides likelihood-ratio tests for each parameter. (In the Chapter 9–10 section we discuss the second GENMOD analysis of a loglinear model.)

Table 7: **SAS Code for Modeling Grouped Crab Data in Tables 4.4 and 5.2**

```
----------------------------------------------------------------------
data crab;
input width y n satell;  logcases=log(n);
datalines;
22.69  5 14  14
...
30.41 14 14  72
;
proc genmod;
  model y/n = width / dist=bin link=logit lrci alpha=.01 type3;
proc logistic;
  model y/n = width / influence stb;
  output out=predict  p=pi_hat  lower=LCL  upper=UCL;
proc print  data=predict;
proc genmod;
  model satell = width / dist=poi link=log offset=logcases residuals;
----------------------------------------------------------------------
```

With PROC LOGISTIC, logistic regression is the default for binary data. PROC LOGISTIC has a built-in check of whether logistic regression ML estimates exist. It can detect complete separation of data points with 0 and 1 outcomes, in which case at least one estimate is infinite. PROC LOGISTIC can also apply other links, such as the probit. Its INFLUENCE option provides Pearson and deviance residuals and diagnostic measures (Pregibon 1981). The STB option provides standardized estimates by multiplying by $s_{x_j}\sqrt{3}/\pi$ (text Section 5.4.7). Following the model statement, Table 7 requests predicted probabilities and lower and upper 95% confidence limits for the probabilities.

Table 8 uses PROC GENMOD and PROC LOGISTIC to fit a logistic model with qualitative predictors to the AIDS and AZT study of Table 5.6. In PROC GENMOD, the OBSTATS option provides various “observation statistics,” including predicted values and their confidence limits. The RESIDUALS option requests residuals such as the Pearson and standardized residuals (labeled “Reschi” and “StReschi”). A CLASS statement requests indicator variables for the factor. By default, in PROC GENMOD the parameter estimate for the last level of each factor equals 0. In PROC LOGISTIC, estimates sum to zero. That is, dummies take the effect coding $(1, -1)$, with values of 1 when in the category and -1 when not, for which parameters sum to

0. In the CLASS statement in PROC LOGISTIC, the option PARAM = REF requests (1, 0) indicator variables with the last category as the reference level. Putting REF = FIRST next to a variable name requests its first category as the reference level. The CLPARM = BOTH and CLODDS = BOTH options provide Wald and profile likelihood confidence intervals for parameters and odds ratio effects of explanatory variables. With AGGREGATE SCALE = NONE in the model statement, PROC LOGISTIC reports Pearson and deviance tests of fit; it forms groups by aggregating data into the possible combinations of explanatory variable values, without overdispersion adjustments. Adding variables in parentheses after AGGREGATE (as in the second use of PROC LOGISTIC in Table 8) specifies the predictors used for forming the table on which to test fit, even when some predictors may have no effect in the model.

Table 8: **SAS Code for Logistic Modeling of AIDS Data in Table 5.6**

```
-----------------------------------------------------------------------
data aids;
input race $ azt $ y n @@;
datalines;
 White Yes 14 107   White No 32 113   Black Yes 11 63   Black No 12 55
;
proc genmod; class race azt;
  model y/n = azt race / dist=bin type3 lrci residuals obstats;
proc logistic; class race azt / param=reference;
  model y/n = azt race / aggregate scale=none clparm=both clodds=both;
  output out=predict p=pi_hat lower=lower upper=upper;
proc print data=predict;
proc logistic; class race azt (ref=first) / param=ref;
  model y/n = azt / aggregate=(azt race) scale=none;
-----------------------------------------------------------------------
```

Table 9 shows logistic regression analyses for the horseshoe crab data of Table 4.3. The models refer to a constructed binary variable Y that equals 1 when a horseshoe crab has satellites and 0 otherwise. With binary data entry, PROC GENMOD and PROC LOGISTIC order the levels alphanumerically, forming the logit with (1, 0) responses as $\log[P(Y = 0)/P(Y = 1)]$. Invoking the procedure with DESCENDING following the PROC name reverses the order. The first two PROC GENMOD statements use both color and width as predictors; color is qualitative in the first model (by the CLASS statement) and quantitative in the second. A CONTRAST statement tests contrasts of parameters, such as whether parameters for two levels of a factor are identical. The statement shown contrasts the first and fourth color levels. The third PROC GENMOD statement uses an indicator variable for color, indicating whether a crab is light or dark (color = 4). The fourth PROC GENMOD statement fits the main effects model using all the predictors. PROC LOGISTIC has options for stepwise selection of variables, as the final model statement shows. The LACKFIT option yields the Hosmer–Lemeshow statistic. Using the OUTROC option, PROC LOGISTIC can output a data set for plotting a ROC curve.

Table 9: **SAS Code for Logistic Regression Models with Horseshoe Crab Data in Table 4.3**

```
--------------------------------------------------------------------------------------
data crab;
input  color  spine  width  satell  weight;
if satell>0 then y=1; if satell=0 then y=0;
if color=4 then light=0; if color < 4 then light=1;
datalines;
2  3  28.3  8  3.05
...
2  2  24.5  0  2.00
;
proc genmod descending; class color;
  model y = width color / dist=bin link=logit lrci type3 obstats;
  contrast 'a-d' color 1  0  0  -1;
proc genmod descending;
  model y = width color / dist=bin link=logit;
proc genmod descending;
  model y = width light / dist=bin link=logit;
proc genmod descending; class color spine;
  model y = width weight color spine / dist=bin link=logit type3;
proc logistic descending; class color spine / param=ref;
  model y = width weight color spine / selection=backward lackfit outroc=classif1;
proc plot data=classif1; plot _sensit_ * _1mspec_  ;
--------------------------------------------------------------------------------------
```

Table 10 analyzes the clinical trial data of Table 6.9 of the textbook. The CMH option in PROC FREQ specifies the CMH statistic, the Mantel–Haenszel estimate of a common odds ratio and its confidence interval, and the Breslow–Day statistic. FREQ uses the two rightmost variables in the TABLES statement as the rows and columns for each partial table; the CHISQ option yields chi-square tests of independence for each partial table. For $I \times 2$ tables the TREND keyword in the TABLES statement provides the Cochran–Armitage trend test. The EQOR option in an EXACT statement provides an exact test for equal odds ratios proposed by Zelen (1971). O'Brien (1986) gave a SAS macro for computing powers using the noncentral chi-squared distribution.

Models with probit and complementary log-log (CLOGLOG) links are available with PROC GENMOD, PROC LOGISTIC, or PROC PROBIT. PROC SURVEYLOGISTIC fits binary regression models to survey data by incorporating the sample design into the analysis and using the method of pseudo ML (with a Taylor series approximation). It can use the logit, probit, and complementary log-log link functions.

For the logit link, PROC GENMOD can perform exact conditional logistic analyses, with the EXACT statement. It is also possible to implement the small-sample tests with mid-P-values and confidence intervals based on inverting tests using mid-P-values. The option $CLTYPE = EXACT|MIDP$ requests either the exact or mid-P

Table 10: **SAS Code for Cochran–Mantel–Haenszel Test and Related Analyses of Clinical Trial Data in Table 6.9**

```
-----------------------------------------------------------------------------
data cmh;
input center $ treat response count @@ ;
datalines;
a 1 1 11     a 1 2 25     a 2 1 10     a 2 2 27
...
h 1 1  4     h 1 2  2     h 2 1  6     h 2 2  1
;
proc freq; weight count;
  tables center*treat*response / cmh chisq;
-----------------------------------------------------------------------------
```

confidence intervals for the parameter estimates. By default, the exact intervals are produced.

Exact conditional logistic regression is also available in PROC LOGISTIC with the EXACT statement.

PROC GAM fits generalized additive models.

Chapter 8: Multinomial Response Models

PROC LOGISTIC fits baseline-category logit models using the LINK = GLOGIT option. The final response category is the default baseline for the logits. Exact inference is also available using the conditional distribution to eliminate nuisance parameters. PROC CATMOD also fits baseline-category logit models, as Table 11 shows for the text example on alligator food choice (Table 8.1). CATMOD codes estimates for a factor so that they sum to zero. The PRED = PROB and PRED = FREQ options provide predicted probabilities and fitted values and their standard errors. The POPULATION statement provides the variables that define the predictor settings. For instance, with "gender" in that statement, the model with lake and size effects is fitted to the full table also classified by gender.

PROC GENMOD can fit the proportional odds version of cumulative logit models using the DIST = MULTINOMIAL and LINK = CLOGIT options. Table 12 fits it to the data shown in Table 8.5 on happiness, number of traumatic events, and race. When the number of response categories exceeds 2, by default PROC LOGISTIC fits this model. It also gives a score test of the proportional odds assumption of identical effect parameters for each cutpoint. Both procedures use the $\alpha_j + \beta x$ form of the model. Cox (1995) used PROC NLIN for the more general model having a scale parameter.

Both PROC GENMOD and PROC LOGISTIC can use other links in cumulative link models. PROC GENMOD uses LINK = CPROBIT for the cumulative probit model and LINK = CCLL for the cumulative complementary log-log model. PROC LOGISTIC fits a cumulative probit model using LINK = PROBIT.

Table 11: **SAS Code for Baseline-Category Logit Models with Alligator Data in Table 8.1**

```
----------------------------------------------------------------------
data gator;
input lake gender size food count;
datalines;
1 1 1 1 7
1 1 1 2 1
1 1 1 3 0
1 1 1 4 0
1 1 1 5 5
...
4 2 2 1 8
4 2 2 2 1
4 2 2 3 0
4 2 2 4 0
4 2 2 5 1 ;
proc logistic; freq count;
class lake size / param=ref;
model food(ref='1') = lake size / link=glogit aggregate scale=none;
proc catmod; weight count;
population lake size gender;
model food = lake size / pred=freq pred=prob;
----------------------------------------------------------------------
```

PROC SURVEYLOGISTIC described above for incorporating the sample design into the analysis can also fit multicategory regression models to survey data, with links such as the baseline-category logit and cumulative logit.

You can fit adjacent-categories logit models in CATMOD by fitting equivalent baseline-category logit models. Table 13 uses it for Table 8.5 from the textbook, on happiness, number of traumatic events, and race. Each line of code in the model statement specifies the predictor values (for the two intercepts, trauma, and race) for the two logits. The trauma and race predictor values are multiplied by 2 for the first logit and 1 for the second logit, to make effects comparable in the two models. PROC CATMOD has options (CLOGITS and ALOGITS) for fitting cumulative logit and adjacent-categories logit models to ordinal responses; however, those options provide weighted least squares (WLS) rather than ML fits. A constant must be added to empty cells for WLS to run. CATMOD treats zero counts as structural zeros, so they must be replaced by small constants when they are actually sampling zeros.

With the CMH option, PROC FREQ provides the generalized CMH tests of conditional independence. The statistic for the "general association" alternative treats X and Y as nominal [statistic (8.18) in the text], the statistic for the "row mean scores differ" alternative treats X as nominal and Y as ordinal, and the statistic for the "nonzero correlation" alternative treats X and Y as ordinal [statistic (8.19)].

Table 12: **SAS Code for Cumulative Logit and Probit Models with Happiness Data in Table 8.5**

```
-------------------------------------------------------------------------
data gss;
    input race $ trauma happy;  * race is 0=white, 1=black;
datalines;
    0 0 1
    0 0 1
    0 0 1
    0 0 1
    0 0 1
    0 0 1
    0 0 1
    0 0 2
...
    1 3 3
        ;
ods graphics on;
ods html;
proc genmod; class race;
model happy = trauma race / dist=multinomial link=clogit lrci type3;
effectplot slicefit;
run;
proc logistic data=gss plots=effect(at(race=all)); class race;
model happy = trauma race ;
 output out=predict p=hi_hat lower=LCL upper=UCL;
 proc print data=predict;
run;
proc logistic data=gss plots=all; class race;
model happiness = trauma race;
run;
ods html close;
ods graphics off;
-------------------------------------------------------------------------
```

PROC MDC fits multinomial discrete choice models, with logit and probit links. One can also use PROC PHREG, which is designed for the Cox proportional hazards model for survival analysis, because the partial likelihood for that analysis has the same form as the likelihood for the multinomial model (Allison 1999, Chap. 7; Chen and Kuo 2001).

Table 13: SAS Code for Adjacent-Categories Logit Model Fitted to Table 8.5 on Happiness, Traumatic Events, and Race

```
-----------------------------------------------------------------------------
data gss;
input race trauma happy count;
count2 = count + 0.00000001;
datalines;
0 0 1  7
0 0 2 15
...
1 5 2  0
1 5 3  0
    ;
proc catmod order=data; weight count2;
population race trauma;
model happy = (1 0 0 0,  0 1 0 0,
               1 0 2 0,  0 1 1 0,
               1 0 4 0,  0 1 2 0,
               1 0 6 0,  0 1 3 0,
               1 0 8 0,  0 1 4 0,
               1 0 10 0, 0 1 5 0,
               1 0 0 2,  0 1 0 1,
               1 0 2 2,  0 1 1 1,
               1 0 4 2,  0 1 2 1,
               1 0 6 2,  0 1 3 1,
               1 0 8 2,  0 1 4 1,
               1 0 10 2, 0 1 5 1) / ML NOGLS;
run;
-----------------------------------------------------------------------------
```

Chapters 9–10: Loglinear Models

For details on the use of SAS (mainly with PROC GENMOD) for loglinear modeling of contingency tables and discrete response variables, see *Advanced Log-Linear Models Using SAS* by D. Zelterman (published by SAS, 2002).

Table 14 uses PROC GENMOD to fit loglinear model (AC, AM, CM) to Table 9.3 from the survey of high school students about using alcohol, cigarettes, and marijuana.

Table 15 uses PROC GENMOD for table raking of Table 9.?? from the textbook. Note the artificial pseudo counts used for the response, to ensure the smoothed margins.

Table 16 uses PROC GENMOD to fit the linear-by-linear association model (10.5) and the row effects model (10.7) to Table 10.3 in the textbook (with column scores 1, 2, 4, 5). The defined variable "assoc" represents the cross-product of row and column

Table 14: **SAS Code for Fitting Loglinear Models to High School Drug Survey Data in Table 9.3**

```
--------------------------------------------------------------------------
data drugs;
input a c m count @@;
datalines;
1 1 1 911   1 1 2 538   1 2 1 44   1 2 2 456
2 1 1   3   2 1 2  43   2 2 1  2   2 2 2 279
;
proc genmod; class a c m;
  model count = a c m a*m a*c c*m / dist=poi link=log lrci type3 obstats;
---------------------------------------------------------------------------
```

Table 15: **SAS Code for Raking Table 9.15**

```
------------------------------------------------------------------------
data rake;
input partyid polviews count;
log_ct = log(count); pseudo = 100/3;
cards;
1 1 306
1 2 279
1 3 116
2 1 185
2 2 312
2 3 194
3 1  26
3 2 134
3 3 338
;
proc genmod; class partyid polviews;
  model pseudo = partyid polviews / dist=poi link=log offset=log_ct
     obstats;
------------------------------------------------------------------------
```

scores, which has β parameter as coefficient in model (10.5).

Correspondence analysis is available in SAS with PROC CORRESP.

Table 16: **SAS Code for Fitting Association Models to GSS Data in Table 10.??**

```
-------------------------------------------------------------------
data sex;
input premar birth u v count @@;  assoc = u*v ;
datalines;
1 1 1 1  81   1 2 1 2  68   1 3 1 4 60   1 4 1 5 38
...
;
proc genmod; class premar birth;
  model  count = premar birth assoc / dist=poi link=log;
proc genmod; class premar birth;
  model  count = premar birth premar*v / dist=poi link=log;
-------------------------------------------------------------------
```

Chapter 11: Models for Matched Pairs

Table 17 analyzes Table 11.1 on presidential voting in two elections. For square tables, the AGREE option in PROC FREQ provides the McNemar chi-squared statistic for binary matched pairs, the X^2 test of fit of the symmetry model (also called *Bowker's test*), and Cohen's kappa and weighted kappa with SE values. The MCNEM keyword in the EXACT statement provides a small-sample binomial version of McNemar's test. PROC CATMOD can provide the Wald confidence interval for the difference of proportions. The code forms a model for the marginal proportions in the first row and the first column, specifying a model matrix in the model statement that has an intercept parameter (the first column) that applies to both proportions and a slope parameter that applies only to the second; hence the second parameter is the difference between the second and first marginal proportions.

PROC LOGISTIC can conduct conditional logistic regression.

Table 18 shows ways of testing marginal homogeneity for the migration data in Table 11.5 of the textbook. The PROC GENMOD code shows the Lipsitz et al. (1990) approach, expressing the I^2 expected frequencies in terms of parameters for the $(I-1)^2$ cells in the first $I-1$ rows and $I-1$ columns, the cell in the last row and last column, and $I-1$ marginal totals (which are the same for rows and columns). Here, m11 denotes expected frequency μ_{11}, m1 denotes $\mu_{1+} = \mu_{+1}$, and so on. This parameterization uses formulas such as $\mu_{14} = \mu_{1+} - \mu_{11} - \mu_{12} - \mu_{13}$ for terms in the last column or last row. CATMOD provides the Bhapkar test (11.15) of marginal homogeneity, as shown.

Table 19 shows various square-table analyses of Table 11.6 of the textbook on premarital and extramarital sex. The "symm" factor indexes the pairs of cells that have the same association terms in the symmetry and quasi-symmetry models. For instance, "symm" takes the same value for cells (1, 2) and (2, 1). Including this term as a factor in a model invokes a parameter λ_{ij} satisfying $\lambda_{ij} = \lambda_{ji}$. The first model fits this factor alone, providing the symmetry model. The second model looks like the

Table 17: **SAS Code for McNemar's Test and Comparing Proportions for Matched Samples in Table 11.1**

```
--------------------------------------------------------------
data matched;
input first second count @@;
datalines;
 1 1 175   1 2 16   2 1  54   2 2 188
;
proc freq; weight count;
    tables first*second / agree;  exact mcnem;
proc catmod; weight count;
    response marginals;
    model first*second =  (1 0 ,
                            1 1 ) ;
--------------------------------------------------------------
```

third except that it identifies "premar" and "extramar" as class variables (for quasi-symmetry), whereas the third model statement does not (for ordinal quasi-symmetry). The fourth model fits quasi-independence. The "qi" factor invokes the δ_i parameters. It takes a separate level for each cell on the main diagonal and a common value for all other cells. The fifth model fits a quasi-uniform association model that takes the uniform association version of the linear-by-linear association model and imposes a perfect fit on the main diagonal.

The bottom of Table 19 fits square-table models as logit models. The pairs of cell counts (n_{ij}, n_{ji}), labeled as "above" and "below" with reference to the main diagonal, are six sets of binomial counts. The variable defined as "score" is the distance $(u_j - u_i) = j - i$. The first two cases are symmetry and ordinal quasi-symmetry. Neither model contains an intercept (NOINT), and the ordinal model uses "score" as the predictor. The third model allows an intercept and is the conditional symmetry model mentioned in Note 11.2.

Table 20 uses PROC GENMOD for logit fitting of the Bradley–Terry model (11.30) to the baseball data of Table 11.10, forming an artificial explanatory variable for each team. For a given observation, the variable for team i is 1 if it wins, -1 if it loses, and 0 if it is not one of the teams for that match. Each observation lists the number of wins ("wins") for the team with variate-level equal to 1 out of the number of games ("games") against the team with variate-level equal to -1. The model has these artificial variates, one of which is redundant, as explanatory variables with no intercept term. The COVB option provides the estimated covariance matrix of parameter estimators.

Table 21 uses PROC GENMOD for fitting the complete symmetry and quasi-symmetry models to Table 11.13 on attitudes toward legalized abortion.

Table 22 shows the likelihood-ratio test of marginal homogeneity for the attitudes toward abortion data of Table 11.13, where for instance *m11p* denotes μ_{11+}. The marginal homogeneity model expresses the eight cell expected frequencies in terms

Table 18: **SAS Code for Testing Marginal Homogeneity with Migration Data in Table 11.5**

```
---------------------------------------------------------------------------
data migrate;
input then $ now $ count m11 m12 m13 m21 m22 m23 m31 m32 m33 m44 m1 m2 m3;
datalines;
  ne ne 266  1  0  0  0  0  0  0  0  0  0  0  0 1  1
  ne mw  15  0  1  0  0  0  0  0  0  0  0  0  0 2  5
  ne  s  61  0  0  1  0  0  0  0  0  0  0  0  0 3  5
  ne  w  28 -1 -1 -1  0  0  0  0  0  0  1  0  0 4  5
  mw ne  10  0  0  0  1  0  0  0  0  0  0  0  0 2  5
  mw mw 414  0  0  0  0  1  0  0  0  0  0  0  0 5  2
  mw  s  50  0  0  0  0  0  1  0  0  0  0  0  0 6  5
  mw  w  40  0  0  0 -1 -1 -1  0  0  0  0  1  0 7  5
   s ne   8  0  0  0  0  0  0  1  0  0  0  0  0 3  5
   s mw  22  0  0  0  0  0  0  0  1  0  0  0  0 6  5
   s  s 578  0  0  0  0  0  0  0  0  1  0  0  0 8  3
   s  w  22  0  0  0  0  0  0 -1 -1 -1  0  0  1 9  5
   w ne   7 -1  0  0 -1  0  0 -1  0  0  1  0  0 4  5
   w mw   6  0 -1  0  0 -1  0  0 -1  0  0  1  0 7  5
   w  s  27  0  0 -1  0  0 -1  0  0 -1  0  0  1 9  5
   w  w 301  0  0  0  0  0  0  0  0  0  1  0  0 10 4
;
proc genmod;
  model count = m11 m12 m13 m21 m22 m23 m31 m32 m33 m44 m1 m2 m3 /
  dist=poi link=identity;
proc catmod; weight count; response marginals;
  model then*now = _response_ / freq;
  repeated time 2;
---------------------------------------------------------------------------
```

of $\mu_{111}, \mu_{11+}, \mu_{1+1}, \mu_{+11}, \mu_{1++}$, and μ_{222} (since $\mu_{+1+} = \mu_{++1} = \mu_{1++}$). Note, for instance, that $\mu_{112} = \mu_{11+} - \mu_{111}$ and $\mu_{122} = \mu_{111} + \mu_{1++} - \mu_{11+} - \mu_{1+1}$. CATMOD provides the generalized Bhapkar test (11.37) of marginal homogeneity.

Chapter 12: Clustered Categorical Responses: Marginal Models

Table 23 uses PROC GENMOD to analyze Table 12.1 from the textbook on depression, using GEE. Possible working correlation structures are TYPE = EXCH for exchangeable, TYPE = AR for autoregressive, TYPE = INDEP for independence, and TYPE = UNSTR for unstructured. Output shows estimates and standard errors under the naive working correlation and based on the sandwich matrix incorporating the

Table 19: **SAS Code Showing Square-Table Analysis of Table 11.6 on Premarital and Extramarital Sex**

```
---------------------------------------------------------------------------
data sex;
input premar  extramar  symm  qi  count @@;
unif = premar*extramar;
datalines;
1 1 1 1 144   1 2 2 5  2   1 3 3 5  0   1 4  4 5 0
2 1 2 5  33   2 2 5 2  4   2 3 6 5  2   2 4  7 5 0
3 1 3 5  84   3 2 6 5 14   3 3 8 3  6   3 4  9 5 1
4 1 4 5 126   4 2 7 5 29   4 3 9 5 25   4 4 10 4 5
;
proc genmod; class symm;
  model  count = symm / dist=poi link=log; * symmetry;
proc genmod; class extramar premar symm;
  model  count = symm extramar premar / dist=poi link=log; *QS;
proc genmod; class symm;
  model  count = symm extramar premar / dist=poi link=log; * ordinal QS;
proc genmod; class extramar premar qi;
  model count = extramar premar qi / dist=poi link=log; * quasi indep;
proc genmod;  class extramar premar;
  model count = extramar premar qi unif / dist=poi link=log;
data sex2; * quasi uniform assoc.
input score below above @@; trials = below + above;
datalines;
1 33 2    1 14 2    1 25 1    2 84 0    2 29 0    3 126 0
;
proc genmod data=sex2;
  model above/trials = score / dist=bin link=logit noint;
proc genmod data=sex2;
  model above/trials = / dist=bin link=logit noint;
proc genmod data=sex2;
  model above/trials = / dist=bin link=logit;
---------------------------------------------------------------------------
```

empirical dependence. Alternatively, the working association structure in the binary case can use the log odds ratio (e.g., using LOGOR = EXCH for exchangeability). The type 3 option in GEE provides score-type tests about effects. See Stokes et al. (2012) for the use of GEE with missing data. PROC GENMOD also provides deletion and diagnostics statistics for its GEE analyses and provides graphics for these statistics.

Table 24 uses PROC GENMOD to implement GEE for a cumulative logit model for the insomnia data of Table 12.3. For multinomial responses, independence is currently the only working correlation structure.

Table 20: **SAS Code for Fitting Bradley–Terry Model to Baseball Data of Table 11.10**

```
--------------------------------------------------------------------------
data baseball;
input wins games boston newyork tampabay toronto baltimore ;
datalines;
 12 18  1 -1  0  0  0
 6  18  1  0 -1  0  0
 10 18  1  0  0 -1  0
 10 18  1  0  0  0 -1
 9  18  0  1 -1  0  0
 11 18  0  1  0 -1  0
 13 18  0  1  0  0 -1
 12 18  0  0  1 -1  0
 9  18  0  0  1  0 -1
 12 18  0  0  0  1 -1
    ;
proc genmod;
   model wins/games = boston newyork tampabay toronto baltimore /
   dist=bin link=logit noint covb obstats ;
run;
--------------------------------------------------------------------------
```

Chapter 13: Clustered Categorical Responses: Random Effects Models

PROC NLMIXED extends GLMs to GLMMs by including random effects. Table 25 analyzes the matched pairs model (13.3) for the change in presidential voting data in Table 13.1.

Table 26 analyzes the Presidential voting data in Table 13.2 of the text, using a one-way random effects model.

Table 27 fits model (13.11) to the attitudes on legalized abortion data of Table 13.3. This shows how to set initial values and set the number of quadrature points for Gauss–Hermite quadrature (e.g., QPOINTS =). One could let SAS fit without initial values but then take that fit as initial values in further runs, increasing QPOINTS until estimates and standard errors converge to the necessary precision.

Table 23 above uses PROC NLMIXED for Table 12.1 on depression. Table 24 uses PROC NLMIXED for ordinal modeling of Table 12.3, defining a general multinomial log likelihood.

Table 28 shows a correlated bivariate random effect analysis of Table 13.8 on attitudes toward the leading crowd.

Agresti et al. (2000) showed PROC NLMIXED examples for clustered data, Agresti and Hartzel (2000) showed code for multicenter trials such as Table 13.7, and Hartzel et

Table 21: SAS Code for Fitting Symmetry and Quasi-Symmetry Models to Attitudes toward Legalized Abortion Data of Table 11.13

```
-----------------------------------------------------------------------------
data gss;
    input absingle abhlth abpoor count;
    sum = absingle + abhlth + abpoor;
 datalines;
 1 1  1 466
 1 1  0  39
 1 0  1   3
 1 0  0   1
 0 1  1  71
 0 1  0 423
 0 0  1   3
 0 0  0 147
     ;
proc genmod data=gss; class sum;
    model count = sum / dist=poisson link=log;
run;
proc genmod data=gss; class sum;
    model count = sum absingle abhlth abpoor / dist=poisson link=log
        obstats ;
run;
-----------------------------------------------------------------------------
```

al. (2001a) showed code for multicenter trials with an ordinal response. The Web site for the journal *Statistical Modelling* shows PROC NLMIXED code for an adjacent-categories logit model and a nominal model at the data archive for Hartzel et al. (2001b). See

`stat.uibk.ac.at/smij/1.2-HartzelAgrestiCaffo-SASCode.txt`

Chen and Kuo (2001) discussed fitting multinomial logit models, including discrete-choice models, with random effects.

PROC NLMIXED allows only one RANDOM statement, which makes it difficult to incorporate random effects at different levels. PROC GLIMMIX has more flexibility. It also fits random effects models and provides built-in distributions and associated variance functions as well as link functions for categorical responses. It can provide a variety of fitting methods, including pseudo likelihood methods, but not ML. See

`support.sas.com/rnd/app/da/stat/procedures/glimmix.html`

Table 22: **SAS Code for Testing Marginal Homogeneity with Attitudes toward Legalized Abortion Data of Table 11.13**

```
----------------------------------------------------------------------------
data abortion;
input a b c count m111 m11p m1p1 mp11 m1pp m222 @@;
datalines;
 1 1 1 466   1  0  0  0  0  0      1 1 2   3  -1  1  0  0  0  0
 1 2 1  71  -1  0  1  0  0  0      1 2 2   3   1 -1 -1  0  1  0
 2 1 1  39  -1  0  0  1  0  0      2 1 2   1   1 -1  0 -1  1  0
 2 2 1 423   1  0 -1 -1  1  0      2 2 2 147   0  0  0  0  0  1
;
proc genmod;
   model count = m111 m11p m1p1 mp11 m1pp m222 / dist=poi link=identity;
proc catmod; weight count; response marginals;
    model a*b*c =  _response_  / freq;
    repeated item 3;
----------------------------------------------------------------------------
```

Table 23: **SAS Code for Marginal Modeling of Depression Data in Table 12.1**

```
----------------------------------------------------------------------------
data depress;
input case diagnose drug time outcome @@; * outcome=1 is normal;
datalines;
 1   0  0  0  1       1  0  0  1  1       1  0  0  2  1
...
340  1  1  0  0     340  1  1  1  0     340  1  1  2  0
;
proc genmod descending;  class case;
  model outcome = diagnose drug time drug*time / dist=bin link=logit type3;
  repeated subject=case / type=exch corrw;
proc nlmixed  qpoints=200;
  parms alpha=-.03 beta1=-1.3 beta2=-.06 beta3=.48 beta4=1.02 sigma=.066;
  eta = alpha + beta1*diagnose + beta2*drug + beta3*time + beta4*drug*time + u;
  p = exp(eta)/(1 + exp(eta));
  model outcome ~ binary(p);
  random u ~ normal(0, sigma*sigma) subject = case;
----------------------------------------------------------------------------
```

Chapter 14: Other Mixture Models for Categorical Data

PROC LOGISTIC provides two overdispersion approaches for binary data. The SCALE = WILLIAMS option uses variance function of the beta-binomial form (14.10),

Table 24: **SAS Code for GEE and Random Intercept Cumulative Logit Analysis of Insomnia Data in Table 12.3**

```
--------------------------------------------------------------------------
data francom;
 input case treat time outcome @@;
 y1=0;y2=0;y3=0;y4=0;
 if outcome=1 then y1=1;
 if outcome=2 then y2=1;
 if outcome=3 then y3=1;
 if outcome=4 then y4=1;
datalines;
    1  1  0  1       1  1  1  1
...
  239  0  0  4     239  0  1  4
;
proc genmod;  class case;
  model outcome = treat time treat*time / dist=multinomial link=clogit;
  repeated subject=case / type=indep corrw;
proc nlmixed  qpoints=40;
  bounds i2 > 0;  bounds i3 > 0;
  eta1 = i1 + treat*beta1 + time*beta2 + treat*time*beta3 + u;
  eta2 = i1 + i2 + treat*beta1 + time*beta2 + treat*time*beta3 + u;
  eta3 = i1 + i2 + i3 + treat*beta1 + time*beta2 + treat*time*beta3 + u;
  p1 = exp(eta1)/(1 + exp(eta1));
  p2 = exp(eta2)/(1 + exp(eta2)) - exp(eta1)/(1 + exp(eta1));
  p3 = exp(eta3)/(1 + exp(eta3)) - exp(eta2)/(1 + exp(eta2));
  p4 = 1 - exp(eta3)/(1 + exp(eta3));
  ll = y1*log(p1) + y2*log(p2) + y3*log(p3) + y4*log(p4);
  model y1 ~ general(ll);
  estimate 'interc2' i1+i2; * this is alpha_2 in model, and i1 is alpha_1;
  estimate 'interc3' i1+i2+i3;  * this is alpha_3 in model;
  random u ~ normal(0, sigma*sigma) subject=case;
--------------------------------------------------------------------------
```

and SCALE = PEARSON uses the scaled binomial variance (14.11). Table 29 illustrates for Table 4.7 from a teratology study. That table also uses PROC NLMIXED for adding litter random intercepts.

For Table 14.6 on homicides, Table 30 uses PROC GENMOD to fit a negative binomial model and a quasi-likelihood model with scaled Poisson variance using the Pearson statistic, and PROC NLMIXED to fit a Poisson GLMM. PROC NLMIXED can also fit negative binomial models.

The PROC GENMOD procedure fits zero-inflated Poisson regression models.

Table 25: **SAS Code for Fitting Model (13.3) for Matched Pairs to Table 13.1**

```
----------------------------------------------------------------------------
data kerryobama;
 input case occasion response count;
datalines;
 1  0 1 175
 1  1 1 175
 2  0 1 16
 2  1 0 16
 3  0 0 54
 3  1 1 54
 4  0 0 188
 4  1 0 188
;
proc nlmixed data=kerryobama qpoints=1000;
  eta = alpha + beta*occasion + u;
  p = exp(eta)/(1 + exp(eta));
  model  response ~ binary(p);
  random u ~ normal(0, sigma*sigma) subject = case;
  replicate count;
run;
----------------------------------------------------------------------------
```

Chapter 15: Non-Model-Based Classification and Clustering

PROC DISCRIM in SAS can perform discriminant analysis. For example, for the ungrouped horseshoe crab as analyzed above in Table 9, you can add code such as

```
-------------------------------------
proc discrim data=crab crossvalidate;
    priors prop;
    class y;
    var width color;
run;
-------------------------------------
```

the statement “priors prop” sets the prior probabilities equal to the sample size. Alternatively “priors equal” would have equal prior proportions in the two categories.

PROC DISTANCE can form distances such as the Jaccard index between pairs of variables. Then, PROC CLUSTER can perform a cluster analysis. Table 31 illustrates for Table 15.6 of the textbook on statewide grounds for divorce, using the average linkage method for pairs of clusters with the Jaccard dissimilarity index.

Table 26: **SAS Code for GLMM Analysis of Election Data in Table 13.2**

```
---------------------------------------------------------------------------------
data vote;
input y n;
case = _n_;
datalines;
1   5
16  32
...
1   4
;
proc nlmixed;
   eta = alpha + u;  p = exp(eta) / (1 + exp(eta));
   model y ~ binomial(n,p);
   random u ~ normal(0,sigma*sigma) subject=case;
   predict p out=new;
proc print data=new;
---------------------------------------------------------------------------------
```

Chapter 16: Large- and Small-Sample Theory for Multinomial Models

Exact conditional logistic regression is available in PROC LOGISTIC with the EXACT statement. It provides ordinary and mid-P-values as well as confidence limits for each model parameter and the corresponding odds ratio with the ESTIMATE = BOTH option. Or, you can pick the type of confidence interval you want by specifying CLTYPE=EXACT or CLTYPE=MIDP. In particular, this enables you to get the Cornfield exact interval for an odds ratio, or its mid-P adaptation. You can also conduct the exact conditional version of the Cochran–Armitage test using the TREND option in the EXACT statement with PROC FREQ. One can also conduct an asymptotic conditional logistic regression, using a STRATA statement to indicate the stratification parameters to be conditioned out. PROC PHREG can also do this (Stokes et al. 2012). For a $2\times2\times K$ table, using the EQOR option in an EXACT statement in PROC FREQ provides an exact test for equal odds ratios proposed by Zelen (1971).

Table 27: **SAS Code for GLMM Modeling of Opinions in Table 13.3**

```
-----------------------------------------------------------------------------
data new;
input sex poor single any count;
datalines;
1 1 1 1 342
...
2 0 0 0 457
;
data new;  set new;
   sex = sex-1; case = _n_;
   q1=1; q2=0; resp = poor; output;
   q1=0; q2=1; resp = single; output;
   q1=0; q2=0; resp = any;  output;
drop poor single any;
proc nlmixed  qpoints = 50;
   parms alpha=0  beta1=.8  beta2=.3  gamma=0  sigma=8.6;
   eta = alpha + beta1*q1 + beta2*q2 + gamma*sex + u;
   p = exp(eta)/(1 + exp(eta));
   model resp ~ binary(p);
   random u ~ normal(0,sigma*sigma) subject = case;
   replicate count;
-----------------------------------------------------------------------------
```

A.2 R AND S-PLUS EXAMPLES

R is free software maintained and regularly updated by many volunteers. It is an open-source version using the S programming language, and many S-Plus functions also work in R. See `www.r-project.org`, at which site you can download R and find various documentation. Our discussion in this Appendix refers to R version 2.13.0.

Dr. Laura Thompson has prepared an excellent, detailed manual on the use of R and S-Plus to conduct the analyses shown in the second edition of *Categorical Data Analysis*. You can access this at

`https://home.comcast.net/~lthompson221/Splusdiscrete2.pdf`

A good introductory resource about R functions for various basic types of categorical data analyses is material prepared by Dr. Brett Presnell at the University of Florida. The sites

`www.stat.ufl.edu/~presnell/Courses/sta4504-2000sp/R`

and in particular,

`www.stat.ufl.edu/~presnell/Courses/sta4504-2000sp/R/R-CDA.pdf`

have details for an introductory course on categorical data analysis with many of the examples from my books.

Table 28: **SAS Code for GLMM for Leading Crowd Data in Table 13.8**

```
--------------------------------------------------------------------------------
data crowd;
input mem1 att1 mem2 att2 count;
datalines;
  1 1 1 1 458
  ...
  0 0 0 0 554
;
data new; set crowd;
  case=_n_;
  x1m=1; x1a=0; x2m=0; x2a=0; var=1; resp=mem1; output;
  x1m=0; x1a=1; x2m=0; x2a=0; var=0; resp=att1; output;
  x1m=0; x1a=0; x2m=1; x2a=0; var=1; resp=mem2; output;
  x1m=0; x1a=0; x2m=0; x2a=1; var=0; resp=att2; output;
  drop mem1 att1 mem2 att2;
proc nlmixed data=new;
  eta=beta1m*x1m + beta1a*x1a + beta2m*x2m + beta2a*x2a + um*var + ua*(1-var);
  p=exp(eta)/(1+exp(eta));
  model resp ~ binary(p);
  random um ua ~ normal([0,0],[s1*s1, cov12, s2*s2]) subject=case;
  replicate count;
  estimate 'mem change' beta2m-beta1m; estimate 'att change' beta2a-beta1a;
--------------------------------------------------------------------------------
```

Another useful resource is the website of Dr. Chris Bilder

`statistics.unl.edu/faculty/bilder/stat875`

where the link to R has examples of the use of R for most chapters of my introductory text, *An Introduction to Categorical Data Analysis.* The link to Schedule at Bilder's website for Statistics 875 at the University of Nebraska has notes for a course on this topic following that text as well as R code and output imbedded within the notes. Thanks to Dr. Bilder for this outstanding resource.

Another good source of examples for Splus and R is Dr. Pat Altham's at Cambridge, UK,

`www.statslab.cam.ac.uk/~pat`

Texts that contain examples of the use of R for various categorical data methods include *Statistical Modelling in R* by M. Aitkin, B. Francis, J. Hinde, and R. Darnell (Oxford 2009), *Modern Applied Statistics With S-PLUS*, 4th ed., by W. N. Venables and B. D. Ripley (Springer, 2010), *Analyzing Medical Data Using S-PLUS* by B. Everitt and S. Rabe-Hesketh (Springer, 2001), *Regression Modeling Strategies* by F. E. Harrell (Springer, 2001), and *Bayesian Computation with R* by J. Albert (Springer, 2009).

Table 29: **SAS Code for Overdispersion Analysis of Teratology Study Data of Table 4.7**

```
----------------------------------------------------------------------------
data moore;
input litter group n y @@;
  z2=0; z3=0; z4=0;
  if group=2 then z2=1; if group=3 then z3=1; if group=4 then z4=1;
datalines;
1  1 10 1     2 1 11 4     3 1 12 9     4 1  4 4
....
55 4 14 1    56 4  8 0    57 4  6 0    58 4 17 0
;
proc logistic;
   model y/n = z2 z3 z4 / scale=williams;
proc logistic;
   model y/n = z2 z3 z4 / scale=pearson;
proc nlmixed  qpoints=200;
  eta = alpha + beta2*z2 + beta3*z3 + beta4*z4 + u ;
  p = exp(eta)/(1 + exp(eta));
  model y ~ binomial(n,p) ;
  random u ~ normal(0, sigma*sigma) subject=litter;
----------------------------------------------------------------------------
```

Chapter 1: Introduction

Univariate binomial and multinomial inference

The function *dbinom* can generate binomial probabilities, for example, *dbinom(6, 10, 0.5)* gives the probability of 6 successes in 10 trials with "probability of success" parameter $\pi = 0.50$.

The function *prop.test* gives the Pearson (score) test and score confidence interval for a binomial proportion, for example, *prop.test(6, 10, p=.5, correct=FALSE)*, where "correct=FALSE" turns off the continuity correction, which is the default. The function *binom.test* gives a small-sample binomial test, for example *binom.test(8, 12, p=0.5, alternative = c("two.sided"))* gives a two-sided test of H_0: $\pi = 0.50$ with 8 successes in 12 trials.

The *table* function constructs contingency tables.

The function *chisq.test* can perform the Pearson chi-squared test of goodness-of-fit of a set of multinomial probabilities. For example, with 3 categories and hypothesized values (0.4, 0.3, 0.3) and observed counts (12, 8, 10),

```
> x <- c(12, 8, 10)
> p <- c(0.4, 0.3, 0.3)
> chisq.test(x, p=p)
```

Table 30: **SAS Code for Fitting Models to Murder Data in Table 14.6**

```
---------------------------------------------------------------------------
data new;
input white black other response;
datalines;
1070 119 55  0
  60  16  5  1
...
   1   0  0  6
;
data new; set new; count = white; race = 0; output;
   count = black; race = 1; output; drop white black other;
data new2; set new; do i = 1 to count; output; end; drop i;
proc genmod data=new2;
   model response = race / dist=negbin link=log;
proc genmod data=new2;
   model response = race / dist=poi link=log scale=pearson;
data new; set new; case = _n_;
proc nlmixed  data = new  qpoints=400;
   parms alpha=-3.7 beta=1.90 sigma=1.6;
   eta = alpha + beta*race + u;  mu = exp(eta);
   model response ~ poisson(mu);
   random  u ~ normal(0, sigma*sigma) subject=case;
   replicate count;
---------------------------------------------------------------------------
```

```
        Chi-squared test for given probabilities

data:  x
X-squared = 0.2222, df = 2, p-value = 0.8948

> chisq.test(x, p=p, simulate.p.value=TRUE, B=10000)

     Chi-squared test for given probabilities with
     simulated p-value (based on 10000 replicates)

data:  x
X-squared = 0.2222, df = NA, p-value = 0.8763
```

The argument "simulate.p.value = TRUE" requests simulation of the exact small-sample test of goodness of fit, with B replicates. So, the second run above uses simulation of 10,000 multinomials with the hypothesized probabilities and finds the sample proportion of them having X^2 value at least as large as the observed value of 0.2222.

Table 31: SAS Code for Discriminant Analysis for Table 15.?? on Statewide Grounds for Divorce

```
data divorce;
  input state $ incompat cruelty desertn non_supp alcohol
               felony impotenc insanity separate ;
   datalines;
   California     1 0 0 0 0 0 0 1 0
   Florida        1 0 0 0 0 0 0 1 0
   Illinois       0 1 1 0 1 1 1 0 0
   Massachusetts  1 1 1 1 1 1 1 0 1
   Michigan       1 0 0 0 0 0 0 0 0
   NewYork        0 1 1 0 0 1 0 0 1
   Texas          1 1 1 0 0 1 0 1 1
   Washington     1 0 0 0 0 0 0 0 1
 ;
 title Grounds for Divorce;
 proc distance  data=divorce  method=djaccard  absent=0  out=distjacc;
   var anominal(incompat--separate);
   id state;
 run;
proc print data=distjacc(obs=8);
  id state;
  title2 Only 8 states;
run;
proc cluster  data=distjacc  method=average
  pseudo outtree=tree;
  id state;
run;
proc tree data=tree horizontal n=7 out=out ;
id state;
run;
```

For special R functions for confidence intervals for a binomial parameter, see

`www.stat.ufl.edu/~aa/cda/R/one-sample/R1/index.html`

The confidence intervals include the score (Wilson) CI, Blaker's exact CI, the small-sample Clopper-Pearson interval and its mid-P adaptation discussed in Section 16.6 of the textbook, and the Agresti–Coull CI and its add-4 special case.

Bayesian inference

See logitnorm.r-forge.r-project.org/ for utilities such as a quantile function for the logit-normal distribution.

The *hpd* function in the *TeachingDemos* library can construct HPD intervals from a posterior distribution. The package *hdrcde* is a more sophisticated package for such methods. For the informative analysis of the vegetarians example at the end of Section 1.6.4:

```
library("TeachingDemos")
y <- 0; n <- 25
a1 <- 3.6; a2 <- 41.4
a <- a1 + y; b <- a2 + n
h <- hpd(qbeta, shape1=a, shape2=b)
```

Chapters 2–3: Two-Way Contingency Tables

For creating mosaic plots in R, see `www.datavis.ca` and also the *mosaic* functions in the *vcd* and *vcdExtra* libraries; see Michael Friendly's tutorial at `cran.us.r-project.org/web/packages/vcdExtra/vignettes/vcd-tutorial.pdf`, which also is useful for basic descriptive and inferential statistics for contingency tables. To construct a mosaic plot for the data in Table 3.2, one can enter

```
> x<- c(9,8,27,8,47,236,23,39,88,49,179,706,28,48,89,19,104,293)
> data <- matrix(x, nrow=3,ncol=6, byrow=TRUE)
> dimnames(data) = list(Degree=c("< HS","HS","College"),Belief=c("1","2","3","4","5","6"))
> install.packages("vcdExtra")
> library("vcdExtra")
> StdResid <- c(-0.4,-2.2,-1.4,-1.5,-1.3,3.6,-2.5,-2.6,-3.3,1.8,0.0,3.4,3.1,4.7,4.8,-0.8,1.1,-6.7)
> StdResid <- matrix(StdResid,nrow=3,ncol=6,byrow=TRUE)
> mosaic(data,residuals = StdResid, gp=shading_Friendly)
```

Chi-squared and Fisher's exact test; Residuals

The function *chisq.test* also can perform the Pearson chi-squared test of independence in a two-way contingency table. For example, for Table 3.2 of the text, using also the textitstdres component for providing standardized residuals,

```
>  data <- matrix(c(9,8,27,8,47,236,23,39,88,49,179,706,28,48,89,19,104,293),
   ncol=6,byrow=TRUE)
> chisq.test(data)

        Pearson's Chi-squared test

data:  data
X-squared = 76.1483, df = 10, p-value = 2.843e-12

> chisq.test(data)$stdres
          [,1]      [,2]      [,3]      [,4]       [,5]      [,6]
[1,] -0.368577 -2.227511 -1.418621 -1.481383 -1.3349600  3.590075
[2,] -2.504627 -2.635335 -3.346628  1.832792  0.0169276  3.382637
[3,]  3.051857  4.724326  4.839597 -0.792912  1.0794638 -6.665195
```

As shown above, you can simulate the exact conditional distribution to estimate the P-value whenever the chi-squared asymptotic approximation is suspect.

Here is code to obtain the profile likelihood confidence interval for the odds ratio for Table 3.1 on seat-belt use and traffic accidents (using the fact that the log odds ratio is the parameter in a simple logistic model):

```
> yes <- c(54,25)
> n <- c(10379,51815)
> x <- c(1,0)
> fit <- glm(yes/n ~ x, weights=n, family=binomial(link=logit))
> summary(fit)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -7.6361     0.2000  -38.17   <2e-16 ***
x             2.3827     0.2421    9.84   <2e-16 ***
-
> confint(fit)
Waiting for profiling to be done...
                2.5 %    97.5 %
(Intercept) -8.055554 -7.268025
x            1.919634  2.873473
> exp(1.919634); exp(2.873473)
[1] 6.818462
[1] 17.69838
```

The function *fisher.test* performs Fisher's exact test. For example, for the tea tasting data of Table 3.9 in the text,

```
> tea <- matrix(c(3,1,1,3),ncol=2,byrow=TRUE)
> fisher.test(tea)
```

The P-value is the sum of probabilities of tables with the given margins that have probability no greater than the observed table. The output also shows the conditional ML estimate of the odds ratio (see Sec. 16.6.4) and a corresponding exact confidence interval based on noncentral hypergeometric probabilities. Use *fisher.test(tea,alternative="greater")* for the one-sided test. For an $I \times J$ table called "table," using

```
> fisher.test(table, simulate.p.value=TRUE, B=10000)
```

generates Monte Carlo simulation with B replicates to estimate the exact P-value based on the exact conditional multiple hypergeometric distribution obtained by conditioning on all row and column marginal totals.

On page 10 of

`www.stat.ufl.edu/~presnell/Courses/sta4504-2000sp/R/R-CDA.pdf`

Brett Presnell shows a simple function for finding also the likelihood-ratio statistic G^2.

Confidence intervals for association measures

For parameters comparing two binomial proportions such as the difference of proportions, relative risk, and odds ratio, a good general-purpose method for constructing confidence intervals is to invert the score test. Such intervals are not available in the standard software packages. See

`www.stat.ufl.edu/~aa/cda/R/two-sample/R2/index.html`

for R functions for confidence intervals comparing two proportions with independent samples, and

`www.stat.ufl.edu/~aa/cda/R/matched/R2_matched/index.html`

for R functions for confidence intervals comparing two proportions with dependent samples. Most of these were written by my former graduate student, Yongyi Min, who also prepared the Bayesian intervals mentioned below. Please quote this site if you use one of these R functions for confidence intervals for association parameters. We believe these functions are dependable, but no guarantees or support are available, so use them at your own risk.

For examples of using R to obtain mid-P confidence intervals for the odds ratio, see the link to Laura Thompson's manual at `www.stat.ufl.edu/~aa/cda/cda.html`.

Ralph Scherer at the Institute for Biometry in Hannover, Germany, has prepared a package *PropCIs* on CRAN incorporating many of these confidence interval functions for proportions and comparisons of proportions. It can be downloaded at

`cran.r-project.org/web/packages/PropCIs/index.html`

Fay (2010a) described an R package that constructs a small-sample confidence interval for the odds ratio by inverting the test using the P-value (mentioned in Section 16.6.1) that was suggested by Blaker (2000), which equals the minimum one-tail probability plus an attainable probability in the other tail that is as close as possible to, but not greater than, that one-tailed probability. See

`journal.r-project.org/archive/2010-1/RJournal_2010-1_Fay.pdf`

Euijung Ryu, a former PhD student of mine who is now at Mayo Clinic, has prepared R functions for various confidence intervals for the ordinal measure $[P(Y1 > Y2) + (1/2)P(Y1 = Y2)]$ that is useful for comparing two multinomial distributions on an ordinal scale. See

`www.stat.ufl.edu/~aa/cda/R/stochastic/ryu-stochastic-code.pdf`

for the functions, including the Wald confidence interval as well as score, pseudo-score, and profile likelihood intervals that are computationally more complex and require using Joe Lang's *mph.fit* function (see below). Also, Euijung has prepared an R function for multiple comparisons of proportions with independent samples using simultaneous confidence intervals for the difference of proportions or the odds ratio, based on the Studentized-range inversion of score tests proposed by Agresti et al. (2008). See

`www.stat.ufl.edu/~aa/cda/R/multcomp/ryu-simultaneous.pdf`

Joseph Lang's *mph.fit* function just mentioned is a general purpose and very powerful function that can provide ML fitting of generalized loglinear models (Section 10.5.1) and other much more general "multinomial-Poisson homogeneous" models such as covered in Lang (2004, 2005). These include models that can be specified in terms of constraints of the form $h(\boldsymbol{\mu}) = 0$, such as the marginal homogeneity model and the calf infection example in Section 1.5.6 of the text. For details, see

`www.stat.uiowa.edu/~jblang/mph.fitting/index.htm`

Joe has also prepared an R program, *ci.table*, for computing (among other things) score and likelihood-ratio-test-based (i.e., profile likelihood) intervals for contingency table parameters. See

`www.stat.uiowa.edu/~jblang/ci.table.documentation/ci.table.examples.htm`

Bayesian inference for two-way tables

Surveys of Bayesian inference using R were given by J. H. Park,

`cran.r-project.org/web/views/Bayesian.html`

and by Jim Albert,

`bayes.bgsu.edu/bcwr`

The latter is a website for the text *Bayesian Computation with R* by Albert. It shows examples of some categorical data analyses, such as Bayesian inference for a 2×2 table, a Bayesian test of independence in a contingency table, and probit regression.

Yongyi Min has prepared some R functions for Bayesian confidence intervals for 2×2 tables using independent beta priors for two binomial parameters, for the difference of proportions, odds ratio, and relative risk. See

`www.stat.ufl.edu/~aa/cda/R/bayes/index.html`

These are evaluated and compared to score confidence intervals in Agresti and Min (2005).

Chapter 4: Generalized Linear Models

Generalized linear models can be fitted with the *glm* function:

`stat.ethz.ch/R-manual/R-patched/library/stats/html/glm.html`

`www.statmethods.net/advstats/glm.html`

That function can be used for such things as logistic regression, Poisson regression, and loglinear models.

Consider a binomial variate y based on n successes with explanatory variable x and a $N \times 2$ data matrix with columns consisting of the values of y and $n - y$. For example, for the logit link with the snoring data in Table 4.2 of the text, using scores (0, 2, 4, 5), showing also a residual analysis,

```
> snoring <- matrix(c(24,1355,35,603,21,192,30,224), ncol=2, byrow=TRUE)
> scores <- c(0,2,4,5)
> snoring.fit <- glm(snoring ~ scores, family=binomial(link=logit))
> summary(snoring.fit)

Call:
glm(formula = snoring ~ scores, family = binomial(link = logit))
```

```
Deviance Residuals:
      1        2        3        4
-0.8346   1.2521   0.2758  -0.6845

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.86625    0.16621 -23.261  < 2e-16 ***
scores       0.39734    0.05001   7.945 1.94e-15 ***
---
Signif. codes:  0  1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 65.9045  on 3  degrees of freedom
Residual deviance:  2.8089  on 2  degrees of freedom
AIC: 27.061

Number of Fisher Scoring iterations: 4

> pearson <- summary.lm(snoring.fit)$residuals   #  Pearson residuals
> hat <- lm.influence(snoring.fit)$hat  # hat or leverage values
> stand.resid <- pearson/sqrt(1 - hat)  # standardized residuals
> cbind(scores, snoring, fitted(snoring.fit), pearson, stand.resid)
  scores                       pearson stand.resid
1      0 24 1355 0.02050742 -0.8131634  -1.6783847
2      2 35  603 0.04429511  1.2968557   1.5448873
3      4 21  192 0.09305411  0.2781891   0.3225535
4      5 30  224 0.13243885 -0.6736948  -1.1970179
```

For the identity link with data in the form of Bernoulli observations, use code such as

```
>   fit <- glm(y ~ x, family=quasi(variance="mu(1-mu)"),start=c(0.5, 0))
>   summary(fit, dispersion=1)
```

The fitting procedure will not converge if at some stage of the fitting process, probability estimates fall outside the permissible (0, 1) range.

The profile likelihood confidence interval is available with the *confint* function in R, which is applied to the model fit object.

The *glm* function can be used to fit Poisson loglinear models and counts and for rates. For negative binomial models, you can use the *glm.nb* function in the MASS library.

`stat.ethz.ch/R-manual/R-patched/library/MASS/html/glm.nb.html`

However, in the notation of Sec. 4.3.4, this function identifies the dispersion parameter (which it calls "theta") as k, not its reciprocal γ. Negative binomial regression can also be handled by Thomas Yee's *VGAM* package mentioned for Chapter 8 below and by the *negbin* function in the *aod* package:

`cran.r-project.org/web/packages/aod/aod.pdf`

To illustrate R for models for counts, for the data in Sec. 4.3 on numbers of satellites for a sample of horseshoe crabs,

```
> crabs <- read.table("crab.dat",header=T)
> crabs
   color spine width satellites weight
 1     3     3  28.3          8   3050
 2     4     3  22.5          0   1550
 3     2     1  26.0          9   2300
 4     4     3  24.8          0   2100
 5     4     3  26.0          4   2600
 6     3     3  23.8          0   2100
....
173     3     2  24.5          0   2000

> weight <- weight/1000  #  weight in kilograms rather than grams
> fit <- glm(satellites ~ weight, family=poisson(link=log), data=crabs)
> summary(fit)

> library(MASS)
> fit.nb <- glm.nb(satell ~ weight, link=log)
> summary(fit.nb)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.8647     0.4048  -2.136   0.0327 *
weight2       0.7603     0.1578   4.817 1.45e-06 ***
---
    Null deviance: 216.43  on 172  degrees of freedom
Residual deviance: 196.16  on 171  degrees of freedom
AIC: 754.64

              Theta:  0.931
          Std. Err.:  0.168
2 x log-likelihood:  -748.644
```

The function *rstandard.glm* has a type argument that can be used to request standardized residuals. That is, you can type

```
>   fit <- glm(... model formula, family, data, etc ...)
>   rstandard(fit, type="pearson")
```

to get standardized Pearson residuals for a fitted GLM. Without the type argument, rstandard(fit) returns the standardized deviance residuals.

The *statmod* library at CRAN contains a function *glm.scoretest* that computes score test statistics for adding explanatory variables to a GLM.

Statistical Models in S by J. M. Chambers and T. J. Hastie (Wadsworth, Belmont, California, 1993, p. 227) showed the use of S-Plus in quasi-likelihood analyses using the *quasi* and *make.family* functions.

Following is an example of the analyses shown for the teratology data, including the quasi-likelihood approach:

```
 # This borrows heavily from Laura Thompson's manual at
 # https://home.comcast.net/~lthompson221/Splusdiscrete2.pdf
> rats <- read.table("teratology.dat", header = T)
> rats  #  Full data set of 58 litters at course website
   litter group  n  y
1       1     1 10  1
2       2     1 11  4
3       3     1 12  9

57     57     4  6  0
58     58     4 17  0
> rats$group <- as.factor(rats$group)
> fit.bin <- glm(y/n ~ group - 1, weights = n, data=rats, family=binomial)
> summary(fit.bin)

Coefficients:      # these are the sample logits
       Estimate Std. Error z value Pr(>|z|)
group1   1.1440     0.1292   8.855  < 2e-16 ***
group2  -2.1785     0.3046  -7.153 8.51e-13 ***
group3  -3.3322     0.7196  -4.630 3.65e-06 ***
group4  -2.9857     0.4584  -6.514 7.33e-11 ***
---

    Null deviance: 518.23  on 58  degrees of freedom
Residual deviance: 173.45  on 54  degrees of freedom
AIC: 252.92

> (pred <- unique(predict(fit.bin, type="response")))
[1] 0.75840979 0.10169492 0.03448276 0.04807692  # sample proportions
> (SE <- sqrt(pred*(1-pred)/tapply(rats$n,rats$group,sum)))
         1          2          3          4
0.02367106 0.02782406 0.02395891 0.02097744      # SE's of proportions

> (X2 <- sum(resid(fit.bin, type="pearson")^2))  # Pearson stat.
[1] 154.707
> phi <- X2/(58 - 4)                        # estimate of phi for QL analysis
> phi
[1] 2.864945
> SE*sqrt(phi)
         1          2          3          4
0.04006599 0.04709542 0.04055320 0.03550674  # adjusted SE's for proportions

> fit.ql <- glm(y/n ~ group - 1, weights=n, data=rats, family=quasi(link=identity,
            variance="mu(1-mu)"),start=unique(predict(fit.bin,type="response")))
> summary(fit.ql)  # This shows another way to get the QL results

Coefficients:
       Estimate Std. Error t value Pr(>|t|)
group1  0.75841    0.04007  18.929   <2e-16 ***
group2  0.10169    0.04710   2.159   0.0353 *
group3  0.03448    0.04055   0.850   0.3989
group4  0.04808    0.03551   1.354   0.1814
---
(Dispersion parameter for quasi family taken to be 2.864945)
```

Chapters 5–7: Logistic Regression and Binary Response Analyses

Logistic Regression

Since logistic regression is a generalized linear model, it can be fitted with the *glm* function, as mentioned above.

If y is a binary variable (i.e., ungrouped binomial data with each $n = 1$), the vector of y values (0 and 1) can be entered as the response variable. Following is an example with the horseshoe crab data as a data frame, declaring color to be a factor in order to set up indicator variables for it (which, by default, choose the first category as the baseline without its own indicator variable).

```
> crabs <- read.table("crabs.dat",header=TRUE)
> crabs
   color spine width satellites weight
 1     3     3  28.3          8   3050
 2     4     3  22.5          0   1550
 3     2     1  26.0          9   2300
....
173     3     2  24.5          0   2000

> y <- ifelse(crabs$satellites > 0, 1, 0)  #  y = a binary indicator of satellites
> crabs$weight <- crabs$weight/1000  #  weight in kilograms rather than grams

> fit <- glm(y ~ weight, family=binomial(link=logit), data=crabs)
> summary(fit)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.6947     0.8802  -4.198 2.70e-05 ***
weight        1.8151     0.3767   4.819 1.45e-06 ***
---

    Null Deviance: 225.7585 on 172 degrees of freedom
Residual Deviance: 195.7371 on 171 degrees of freedom
AIC: 199.74

> crabs$color <- crabs$color - 1  #  color now takes values 1,2,3,4
> crabs$color <- factor(crabs$color)  #  treat color as a factor
> fit2 <- glm(y ~ weight + color, family=binomial(link=logit), data=crabs)
> summary(fit2)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.2572     1.1985  -2.718  0.00657 **
weight        1.6928     0.3888   4.354 1.34e-05 ***
color2        0.1448     0.7365   0.197  0.84410
color3       -0.1861     0.7750  -0.240  0.81019
color4       -1.2694     0.8488  -1.495  0.13479
---

(Dispersion Parameter for Binomial family taken to be 1 )

    Null Deviance: 225.7585 on 172 degrees of freedom
Residual Deviance: 188.5423 on 168 degrees of freedom
AIC: 198.54
```

For grouped data, rather than defining the response as the set of success and failure counts as was done in the Chapter 4 discussion above for the snoring data, one can instead enter the response in the form y/n for y successes in n trials, entering the number of trials as the weight. For example, again for the snoring data of Table 4.2,

```
> yes <- c(24,35,21,30)
> n <- c(1379,638,213,254)
> scores <- c(0,2,4,5)
> fit <- glm(yes/n ~ scores, weights=n, family=binomial(link=logit))
> fit

Coefficients:
(Intercept)       scores
    -3.8662       0.3973

Degrees of Freedom: 3 Total (i.e. Null);  2 Residual
Null Deviance:      65.9
Residual Deviance: 2.809        AIC: 27.06
```

The R package *glmnet* can apparently fit logistic regression to data sets with very large numbers of variables or observations, and as mentioned below can use regularization methods such as the lasso:

`cran.r-project.org/web/packages/glmnet/index.html`

Cochran–Mantel–Haenszel test

The function *mantelhaen.test* can perform Cochran-Mantel-Haenszel tests for $I \times J \times K$ tables:

`stat.ethz.ch/R-manual/R-patched/library/stats/html/mantelhaen.test.html`

For example, for the clinical trials data in Table 6.9,

```
> beitler <- c(11,10,25,27,16,22,4,10,14,7,5,12,2,1,14,16,6,0,11,12,1,0,10,10,1,1,4,8,4,6,2,1)
> beitler <- array(beitler, dim=c(2,2,8))
> mantelhaen.test(beitler, correct=FALSE)

        Mantel-Haenszel chi-squared test without continuity correction

data:  beitler
Mantel-Haenszel X-squared = 6.3841, df = 1, p-value = 0.01151
alternative hypothesis: true common odds ratio is not equal to 1
95 percent confidence interval:
 1.177590 3.869174
sample estimates:
common odds ratio
         2.134549
```

When $I = 2$ and $J = 2$, enter "correct=FALSE" so as not to use the continuity correction. In that case, the output also shows the Mantel–Haenszel estimate $\hat{\theta}_{MH}$ and the corresponding confidence interval for the common odds ratio. With the exact option,

```
> mantelhaen.test(beitler, correct=FALSE, exact=TRUE)
```

R provides the exact conditional test (Sec. 7.3.5) and the conditional ML estimate of the common odds ratio (Sec. 16.6.6). When $I > 2$ and/or $J > 2$, this function provides the generalized test that treats X and Y as nominal scale (i.e., df = $(I-1)(J-1)$, given in equation (8.18) in the text).

Other binary response models

For binary data, alternative links are possible. For example, continuing with the horseshoe crab data from above,

```
> fit.probit <- glm(y ~ weight, family=binomial(link=probit), data=crabs)
> summary(fit.probit)
Coefficients:
                Value Std. Error   t value
(Intercept) -2.238245  0.5114850 -4.375974
     weight  1.099017  0.2150722  5.109989

Residual Deviance: 195.4621 on 171 degrees of freedom
```

For the complementary log-log link with the beetle data of Table 7.1, showing also the construction of standardized residuals and profile likelihood confidence intervals,

```
> beetles <- read.table("beetle.dat", header=T)
> beetles
   dose number killed
1 1.691     59      6
2 1.724     60     13
```

```
3 1.755     62     18
4 1.784     56     28
5 1.811     63     52
6 1.837     59     53
7 1.861     62     61
8 1.884     60     60

> binom.dat <- matrix(append(killed,number-killed),ncol=2)
> fit.cloglog <- glm(binom.dat ~ dose, family=binomial(link=cloglog),
    data=beetles)
> summary(fit.cloglog)  #  much better fit than logit

               Value Std. Error   t value
(Intercept) -39.52250   3.232269 -12.22748
       dose  22.01488   1.795086  12.26397

    Null Deviance: 284.2024 on 7 degrees of freedom
Residual Deviance: 3.514334 on 6 degrees of freedom

> pearson.resid <- resid(fit.cloglog, type="pearson")
> std.resid <- pearson.resid/sqrt(1-lm.influence(fit.cloglog)$hat)
> cbind(dose, killed/number, fitted(fit.cloglog), pearson.resid, std.resid)
   dose                      pearson.resid  std.resid
1 1.691 0.1016949 0.09582195     0.1532583  0.1772659
2 1.724 0.2166667 0.18802653     0.5677671  0.6694966
3 1.755 0.2903226 0.33777217    -0.7899738 -0.9217717
4 1.784 0.5000000 0.54177644    -0.6274464 -0.7041154
5 1.811 0.8253968 0.75683967     1.2684541  1.4855799
6 1.837 0.8983051 0.91843509    -0.5649292 -0.7021989
7 1.861 0.9838710 0.98575181    -0.1249636 -0.1489834
8 1.884 1.0000000 0.99913561     0.2278334  0.2368981
> confint(fit.cloglog)
               2.5 %    97.5 %
(Intercept) -46.13984 -33.49923
dose         18.66945  25.68877
```

Bayesian fitting

Jim Albert in *Bayesian Computation with R* (Springer 2009, pp. 216-219) presented an R function, *bayes.probit*, for implementing his algorithm for fitting probit models with a Bayesian approach.

Penalized likelihood

The Copas smoothing method can be implemented with the R function *ksmooth*, with lambda=bandwidth. For example, for the kyphosis example of Sec. 7.4.3,

```
> x <- c(12, 15, 42, 52, 59, 73, 82, 91, 96, 105, 114, 120, 121, 128, 130,
         139, 139, 157, 1, 1, 2, 8, 11, 18, 22, 31, 37, 61, 72, 81, 97,
        112, 118, 127, 131, 140, 151, 159, 177, 206)
> y <- c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0)
> k1 <- ksmooth(x,y,"normal",bandwidth=25)
> k2 <- ksmooth(x,y,"normal",bandwidth=100)
> plot(x,y)
> lines(k1)
> lines(k2, lty=2)
```

The *brglm* function in the MASS library can implement bias reduction using the Firth penalized likelihood approach for binary regression models, including models with logit, probit, and complementary log-log links:

`cran.r-project.org/web/packages/brglm/index.html`

Lasso for binary and count models is available in the R packages *glmnet* and *glmpath*:

`cran.r-project.org/web/packages/glmnet/index.html`

`cran.r-project.org/web/packages/glmpath/index.html`

The group lasso is available with the *grplasso* package:

`cran.r-project.org/web/packages/grplasso/index.html`

Generalized additive models

For a generalized additive model, R has a *gam* package:

`cran.r-project.org/web/packages/gam/index.html`

Thomas Yee's *VGAM* library can also handle GAMs:

`www.stat.auckland.ac.nz/~yee/VGAM/doc/glmgam.pdf`

`rss.acs.unt.edu/Rdoc/library/VGAM/html/vgam.html`

For example, for the ungrouped horseshoe crab data,

```
> library(vgam)
> gam.fit <- vgam(y ~ s(weight), family=binomialff(link=logit), data=crabs)
> plot(weight, fitted(gam.fit))
```

GAMs can also be fitted with the *gam* function in the *mgcv* library:

`cran.r-project.org/web/packages/mgcv/mgcv.pdf`

False discovery rate (FDR)

R packages for FDR are listed at

`strimmerlab.org/notes/fdr.html`

Chapter 8: Multinomial Response Models

For baseline-category logit models, one can use the *multinom* function in the *nnet* library that has been provided by Venables and Ripley to do various calculations by neural nets (see, e.g., p. 230 of Venables and Ripley, 3rd ed.):

`cran.r-project.org/web/packages/nnet/nnet.pdf`

Statements have the form

```
> fit <- multinom(y ~ x + factor(z),weights=freq, data=gators)
```

The VGAM package

Especially useful for modeling multinomial responses is the *VGAM* package and *vglm* function developed by Thomas Yee at Auckland, New Zealand,

`www.stat.auckland.ac.nz/~yee/VGAM`

This package has functions that can also can fit a wide variety of models including multinomial logit models for nominal responses and cumulative logit models, adjacent-categories models, and continuation-ratio models for ordinal responses. For more details, see "The VGAM package for categorical data analysis," in *Journal of Statistical Software*, vol. 32, pp. 1-34 (2010), `www.jstatsoft.org/v32/i10`. See also `www.stat.auckland.ac.nz/~yee/VGAM/doc/categorical.pdf` for some basic examples of its use for categorical data analysis.

Following is an example of the use of *vglm* for fitting a baseline-category logit model to the alligator food choice data in Table 8.1 of the textbook. The data file has the five multinomial counts for the food choices identified as $y1$ through $y5$, with $y1$ being fish as in the text. The *vglm* function uses the final category as the baseline, so to use fish as the baseline, in the model statement we identify the response categories as (y_2, y_3, y_4, y_5, y_1). By contrast, the *multinom* function in the *nnet* library picks the first category of the response variable as the baseline. The following also shows output using it. For both functions, a predictor identified as a factor in the model statement has its first category as the baseline, so the lake estimates shown here differ from those in the book, which used the last lake level as the baseline.

```
> alligators <- read.table("alligators.dat",header=TRUE)
> alligators
   lake gender size y1 y2 y3 y4 y5
1     1      1    1  7  1  0  0  5
2     1      1    0  4  0  0  1  2
3     1      0    1 16  3  2  2  3
4     1      0    0  3  0  1  2  3
5     2      1    1  2  2  0  0  1
6     2      1    0 13  7  6  0  0
7     2      0    1  0  1  0  1  0
8     2      0    0  3  9  1  0  2
9     3      1    1  3  7  1  0  1
10    3      1    0  8  6  6  3  5
11    3      0    1  2  4  1  1  4
12    3      0    0  0  1  0  0  0
13    4      1    1 13 10  0  2  2
14    4      1    0  9  0  0  1  2
15    4      0    1  3  9  1  0  1
16    4      0    0  8  1  0  0  1
```

```
> library(VGAM)
> vglm(formula = cbind(y2,y3,y4,y5,y1) ~ size + factor(lake),
family=multinomial, data=alligators)

Coefficients:
  (Intercept):1   (Intercept):2   (Intercept):3   (Intercept):4          size:1
     -3.2073772      -2.0717560      -1.3979592      -1.0780754       1.4582046
         size:2          size:3          size:4 factor(lake)2:1 factor(lake)2:2
     -0.3512628      -0.6306597       0.3315503       2.5955779       1.2160953
factor(lake)2:3 factor(lake)2:4 factor(lake)3:1 factor(lake)3:2 factor(lake)3:3
     -1.3483253      -0.8205431       2.7803434       1.6924767       0.3926492
factor(lake)3:4 factor(lake)4:1 factor(lake)4:2 factor(lake)4:3 factor(lake)4:4
      0.6901725       1.6583586      -1.2427766      -0.6951176      -0.8261962

Degrees of Freedom: 64 Total; 44 Residual
Residual Deviance: 52.47849
Log-likelihood: -74.42948

> library(nnet)
> fit2 <- multinom(cbind(y1,y2,y3,y4,y5) ~ size + factor(lake), data=alligators)
> summary(fit2)
Call:
multinom(formula = cbind(y1, y2, y3, y4, y5) ~ size + factor(lake),
    data = alligators)

Coefficients:
   (Intercept)       size factor(lake)2 factor(lake)3 factor(lake)4
y2   -3.207394  1.4582267     2.5955898     2.7803506     1.6583514
y3   -2.071811 -0.3512070     1.2161555     1.6925186    -1.2426769
y4   -1.397976 -0.6306179    -1.3482294     0.3926516    -0.6951107
y5   -1.078137  0.3315861    -0.8204767     0.6902170    -0.8261528

Std. Errors:
   (Intercept)      size factor(lake)2 factor(lake)3 factor(lake)4
y2   0.6387317 0.3959455     0.6597077     0.6712222     0.6128757
y3   0.7067258 0.5800273     0.7860141     0.7804482     1.1854024
y4   0.6085176 0.6424744     1.1634848     0.7817677     0.7812585
y5   0.4709212 0.4482539     0.7296253     0.5596752     0.5575414

Residual Deviance: 540.0803
AIC: 580.0803
```

The vglm function for ordinal models

The *vglm* function in the *VGAM* library can also fit a wide variety of ordinal models. Many examples of the use of *vglm* for various ordinal-response analyses are available at the website for my book, *Analysis of Ordinal Categorical Data* (2nd ed., 2010), `www.stat.ufl.edu/~aa/ordinal/ord.html`, and several of these are also shown below. For example, for the cumulative logit model fitted to the happiness data of Table 8.5

of the textbook, entering each multinomial observation as a set of indicators that indicates the response category, letting race = 0 for white and 1 for black, and letting traumatic be the number of traumatic events,

```
> happy <- read.table("happy.dat", header=TRUE)
> happy
   race traumatic y1 y2 y3
1     0         0  1  0  0
2     0         0  1  0  0
3     0         0  1  0  0
4     0         0  1  0  0
5     0         0  1  0  0
6     0         0  1  0  0
7     0         0  1  0  0
8     0         0  0  1  0
...
94    1         2  0  0  1
95    1         3  0  1  0
96    1         3  0  1  0
97    1         3  0  0  1
> library(VGAM)
> fit <- vglm(cbind(y1,y2,y3) ~ race + traumatic,
         family=cumulative(parallel=TRUE), data=happy)
> summary(fit)

Coefficients:
                 Value Std. Error t value
(Intercept):1 -0.51812    0.33819 -1.5320
(Intercept):2  3.40060    0.56481  6.0208
race          -2.03612    0.69113 -2.9461
traumatic     -0.40558    0.18086 -2.2425

Names of linear predictors: logit(P[Y<=1]), logit(P[Y<=2])

Residual Deviance: 148.407 on 190 degrees of freedom
Log-likelihood: -74.2035 on 190 degrees of freedom
Number of Iterations: 5

> fit.inter <- vglm(cbind(y1,y2,y3) ~ race + traumatic + race*traumatic,
     family=cumulative(parallel=TRUE), data=happy)
> summary(fit.inter)
Coefficients:
                  Value Std. Error t value
(Intercept):1  -0.43927    0.34469 -1.2744
(Intercept):2   3.52745    0.58737  6.0055
race           -3.05662    1.20459 -2.5375
traumatic      -0.46905    0.19195 -2.4436
race:traumatic  0.60850    0.60077  1.0129

Residual Deviance: 147.3575 on 189 degrees of freedom
```

```
Log-likelihood: -73.67872 on 189 degrees of freedom
Number of Iterations: 5
```

The parallel=TRUE option requests the proportional odds version of the model with the same effects for each cumulative logit. Then entering *fitted(fit)* would produce the estimated probabilities for each category for each observation. Here, we also fitted the model with an interaction term, which does not provide a significantly better fit.

To use *vglm* to fit the cumulative logit model not having the proportional odds assumption, we take out the parallel=TRUE option. Then, we do a likelihood-ratio test to see if it gives a better fit:

```
> fit2 <- vglm(cbind(y1,y2,y3) ~ race + traumatic, family=cumulative,
          data=happy)
> summary(fit2)

Coefficients:
                  Value Std. Error   t value
(Intercept):1  -0.56605    0.36618 -1.545821
(Intercept):2   3.48370    0.75950  4.586850
race:1        -14.01877  322.84309 -0.043423
race:2         -1.84673    0.76276 -2.421095
traumatic:1    -0.34091    0.21245 -1.604644
traumatic:2    -0.48356    0.27524 -1.756845

Residual Deviance: 146.9951 on 188 degrees of freedom
Log-likelihood: -73.49755 on 188 degrees of freedom
Number of Iterations: 14

> pchisq(deviance(fit)-deviance(fit2),df=df.residual(fit)-df.residual(fit2),lower.tail=FALSE)
[1] 0.4936429
```

Note that the ML effect estimate of race for the first logit is actually $-\infty$, reflecting the lack of any black subjects in the first happiness category.

For the same data, to fit the cumulative probit model with common effects for each probit, we use

```
fit.probit <- vglm(cbind(y1,y2,y3) ~ race + traumatic,
    family=cumulative(link=probit, parallel=TRUE), data=happy)
> summary(fit.probit)

Coefficients:
                 Value Std. Error t value
(Intercept):1 -0.34808   0.200147 -1.7391
(Intercept):2  1.91607   0.282872  6.7736
race          -1.15712   0.378716 -3.0554
traumatic     -0.22131   0.098973 -2.2361

Residual Deviance: 148.1066 on 190 degrees of freedom
Log-likelihood: -74.0533 on 190 degrees of freedom
Number of Iterations: 5
```

To fit the adjacent-categories logit model to the same data, we use

```
> fit.acat <- vglm(cbind(y1,y2,y3) ~ race + traumatic,
     family=acat(reverse=TRUE, parallel=TRUE), data=happy)

> summary(fit.acat)

Coefficients:
                 Value Std. Error t value
(Intercept):1 -0.49606    0.31805 -1.5597
(Intercept):2  3.02747    0.57392  5.2751
race          -1.84230    0.64190 -2.8701
traumatic     -0.35701    0.16396 -2.1775

Names of linear predictors: log(P[Y=1]/P[Y=2]), log(P[Y=2]/P[Y=3])
Residual Deviance: 148.1996 on 190 degrees of freedom
Log-likelihood: -74.09982 on 190 degrees of freedom
Number of Iterations: 5
```

To fit the continuation-ratio logit model to the same data, one direction for forming the sequential logits yields the results:

```
> fit.cratio <- vglm(cbind(y1,y2,y3) ~ race + traumatic,
       family=cratio(reverse=TRUE, parallel=TRUE), data=happy)

> summary(fit.cratio)

Coefficients:
                 Value Std. Error t value
(Intercept):1 -0.45530    0.32975 -1.3808
(Intercept):2  3.34108    0.56309  5.9335
race          -2.02555    0.67683 -2.9927
traumatic     -0.38504    0.17368 -2.2170

Names of linear predictors: logit(P[Y<2|Y<=2]), logit(P[Y<3|Y<=3])
Residual Deviance: 148.1571 on 190 degrees of freedom
Log-likelihood: -74.07856 on 190 degrees of freedom
Number of Iterations: 5
```

The more common form of continuation-ratio logit is obtained by instead using REVERSE=FALSE in the model-fitting statement.

Other multinomial functions

For the proportional odds version of cumulative logit models, you can alternatively use the *polr* function in the MASS library, with syntax shown next. However, the data file then needs the response as a factor vector, so we first put the data from the above examples in that form.

```
> library(MASS)
> response <- matrix(0,nrow=97,ncol=1)
> response <- ifelse(y1==1,1,0)
> response <- ifelse(y2==1,2,resp)
> response <- ifelse(y3==1,3,resp)
> y <- factor(response)
> polr(y ~ race + traumatic, data=happy)
Call:
polr(formula = y ~ race + traumatic, data=happy)

Coefficients:
     race traumatic
2.0361187 0.4055724

Intercepts:
       1|2        2|3
-0.5181118  3.4005955

Residual Deviance: 148.407
AIC: 156.407
```

Chapters 9–10: Loglinear Models

Since loglinear models are special cases of generalized linear models with Poisson random component and log link function, they can be fitted with the *glm* function. To illustrate this, the following code shows fitting the models (A, C, M) and (AC, AM, CM) for Table 9.3 for the high school survey about use of alcohol, cigarettes and marijuana. The code also shows forming Pearson and standardized residuals for the homogeneous association model, (AC, AM, CM). For factors, R sets the value equal to 0 at the first category rather than the last as in the text examples.

```
> drugs <- read.table("drugs.dat",header=TRUE)
> drugs
    a   c   m count
1 yes yes yes   911
2 yes yes  no   538
3 yes  no yes    44
4 yes  no  no   456
5  no yes yes     3
6  no yes  no    43
7  no  no yes     2
8  no  no  no   279
> alc <- factor(a); cig <- factor(c); mar <- factor(m)
> indep <- glm(count ~ alc + cig + mar, family=poisson(link=log), data=drugs)
> summary(indep) % loglinear model (A, C, M)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  6.29154    0.03667 171.558  < 2e-16 ***
alc2        -1.78511    0.05976 -29.872  < 2e-16 ***
```

```
cig2        -0.64931    0.04415 -14.707  < 2e-16 ***
mar2         0.31542    0.04244   7.431 1.08e-13 ***
---

    Null deviance: 2851.5  on 7  degrees of freedom
Residual deviance: 1286.0  on 4  degrees of freedom
AIC: 1343.1

Number of Fisher Scoring iterations: 6

> homo.assoc <- update(indep, .~. + alc:cig + alc:mar + cig:mar)
> summary(homo.assoc) # loglinear model (AC, AM, CM)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  6.81387    0.03313 205.699  < 2e-16 ***
alc2        -5.52827    0.45221 -12.225  < 2e-16 ***
cig2        -3.01575    0.15162 -19.891  < 2e-16 ***
mar2        -0.52486    0.05428  -9.669  < 2e-16 ***
alc2:cig2    2.05453    0.17406  11.803  < 2e-16 ***
alc2:mar2    2.98601    0.46468   6.426 1.31e-10 ***
cig2:mar2    2.84789    0.16384  17.382  < 2e-16 ***
---

    Null deviance: 2851.46098  on 7  degrees of freedom
Residual deviance:    0.37399  on 1  degrees of freedom
AIC: 63.417

Number of Fisher Scoring iterations: 4

> pearson <- summary.lm(homo.assoc)$residuals   #  Pearson residuals
> sum(pearson^2)  # Pearson goodness-of-fit statistic
[1] 0.4011006
> leverage <- lm.influence(homo.assoc)$hat # leverage values
> std.resid <- pearson/sqrt(1 - leverage) # standardized residuals
> expected <- fitted(homo.assoc) # estimated expected frequencies
> cbind(count, expected, pearson, std.resid)
  count   expected     pearson  std.resid
1   911 910.383170  0.02044342  0.6333249
2   538 538.616830 -0.02657821 -0.6333249
3    44  44.616830 -0.09234564 -0.6333249
4   456 455.383170  0.02890528  0.6333249
5     3   3.616830 -0.32434090 -0.6333251
6    43  42.383170  0.09474777  0.6333249
7     2   1.383170  0.52447895  0.6333251
8   279 279.616830 -0.03688791 -0.6333249
```

Association models

Following is an example for the linear-by-linear association model and the row effects and columns effects models (with scores 1, 2, 4, 5) fitted to Table 10.3 on premarital sex and teenage birth control.

```
> sexdata <- read.table("sex.dat", header=TRUE)
> attach(sexdata)
> uv <- premar*birth
> premar <- factor(premar); birth <- factor(birth)
> LL.fit <- glm(count ~ premar + birth + uv, family=poisson)
> summary(LL.fit)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.10684    0.08951  45.881  < 2e-16 ***
premar2     -1.64596    0.13473 -12.216  < 2e-16 ***
premar3     -1.77002    0.16464 -10.751  < 2e-16 ***
premar4     -1.75369    0.23432  -7.484 7.20e-14 ***
birth2      -0.46411    0.11952  -3.883 0.000103 ***
birth3      -0.72452    0.16201  -4.472 7.74e-06 ***
birth4      -1.87966    0.24910  -7.546 4.50e-14 ***
uv           0.28584    0.02824  10.122  < 2e-16 ***
    Null deviance: 431.078  on 15  degrees of freedom
Residual deviance:  11.534  on  8  degrees of freedom
AIC: 118.21

Number of Fisher Scoring iterations: 4

> u <- c(1,1,1,1,2,2,2,2,4,4,4,4,5,5,5,5)
> v <- c(1,2,4,5,1,2,4,5,1,2,4,5,1,2,4,5)
> row.fit <- glm(count ~ premar + birth + u:birth, family=poisson)
> summary(row.fit)

Coefficients: (1 not defined because of singularities)
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.98722    0.14624  34.102  < 2e-16 ***
premar2     -0.65772    0.13124  -5.011 5.40e-07 ***
premar3      0.46664    0.16266   2.869 0.004120 **
premar4      1.50195    0.17952   8.366  < 2e-16 ***
birth2      -0.31939    0.19821  -1.611 0.107103
birth3      -0.72688    0.20016  -3.632 0.000282 ***
birth4      -1.49032    0.23745  -6.276 3.47e-10 ***
birth1:u    -0.59533    0.06555  -9.082  < 2e-16 ***
birth2:u    -0.40543    0.06068  -6.681 2.37e-11 ***
birth3:u    -0.12975    0.05634  -2.303 0.021276 *
birth4:u          NA         NA      NA       NA

    Null deviance: 431.078  on 15  degrees of freedom
Residual deviance:   8.263  on  6  degrees of freedom
```

```
AIC: 118.94

Number of Fisher Scoring iterations: 4

> column.fit <- glm(count ~ premar + birth + premar:v, family=poisson)
> summary(column.fit)

Coefficients: (1 not defined because of singularities)
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.40792    0.26947   5.225 1.74e-07 ***
premar2     -0.68466    0.29053  -2.357 0.018444 *
premar3      0.78235    0.22246   3.517 0.000437 ***
premar4      2.11167    0.18958  11.138  < 2e-16 ***
birth2       0.54590    0.11723   4.656 3.22e-06 ***
birth3       1.59262    0.14787  10.770  < 2e-16 ***
birth4       1.51018    0.16420   9.197  < 2e-16 ***
premar1:v    0.58454    0.05930   9.858  < 2e-16 ***
premar2:v    0.49554    0.07990   6.202 5.57e-10 ***
premar3:v    0.20315    0.06538   3.107 0.001890 **
premar4:v         NA         NA      NA       NA

    Null deviance: 431.0781  on 15  degrees of freedom
Residual deviance:   7.5861  on 6  degrees of freedom
AIC: 118.26

Number of Fisher Scoring iterations: 4
----------------------------------------------------------------
```

The *loglin* function can fit loglinear models using iterative proportional fitting. Joseph Lang's mph.fit function can fit generalized loglinear models (Section 10.5.1) and other much more general "multinomial-Poisson homogeneous" models such as covered in Lang (2004, 2005):

`www.stat.uiowa.edu/~jblang/mph.fitting/index.htm`

Multiplicative models such as RC and stereotype

The gnm add-on package for R, developed by David Firth and Heather Turner at the Univ. of Warwick, can fit multiplicative models such as Goodman's RC association model for two-way contingency tables and Anderson's stereotype model for ordinal multinomial responses:

`www2.warwick.ac.uk/fac/sci/statistics/staff/academic-research/firth/software`

Thomas Yee's VGAM package mentioned for Chapter 8 above can also fit Goodman's RC association model and Anderson's stereotype model, as well as bivariate logistic and probit models for bivariate binary responses.

Nenadic and Greenacre have developed the ca package for correspondence analysis:

`www.statmethods.net/advstats/ca.html`

Chapter 11: Models for Matched Pairs

McNemar test

The function *mcnemar.test* can conduct McNemar's test for matched pairs. For example, for Table 11.1,

```
ratings <- matrix(c(175, 16, 54, 188), ncol=2, byrow=TRUE,
+ dimnames = list("2004 Election" = c("Democrat", "Republican"),
+                        "2008 Election" = c("Democrat", "Republican")))
> mcnemar.test(ratings, correct=FALSE)
```

where a continuity correction is made unless "correct=FALSE" is specified.

Bradley–Terry models

The Bradley–Terry model can be fitted using the *glm* function by treating it as a generalized linear model. It can also be fitted using specialized functions, such as with the *brat* function in Thomas Yee's *VGAM* library mentioned above:

`rss.acs.unt.edu/Rdoc/library/VGAM/html/brat.html`

or by Prof. David Firth as described at

`www2.warwick.ac.uk/fac/sci/statistics/staff/academic-research/firth/software/bradleyterry`

`www.jstatsoft.org/v12/i01`

Chapter 12: Clustered Categorical Responses: Marginal Models

GEE methods

Laura Thompson's manual describes several packages for doing GEE analyses. For instance, in the following code we use the *gee* function in the *gee* library to analyze the opinions about abortion data analyzed in Sec. 13.3.2 with both marginal models and random effects models.

```
> abortion
     gender response question case
1         1        1        1    1
2         1        1        2    1
3         1        1        3    1
4         1        1        1    2
5         1        1        2    2
6         1        1        3    2
7         1        1        1    3
8         1        1        2    3
9         1        1        3    3
...
5545      0        0        1 1849
```

```
5546      0        0        2 1849
5547      0        0        3 1849
5548      0        0        1 1850
5549      0        0        2 1850
5550      0        0        3 1850

> z1 <- ifelse(abortion$question==1,1,0)
> z2 <- ifelse(abortion$question==2,1,0)
> z3 <- ifelse(abortion$question==3,1,0)

> library(gee)

> fit.gee <- gee(response ~ gender + z1 + z2, id=case, family=binomial,
+            corstr="exchangeable", data=abortion)

> summary(fit.gee)

 GEE:  GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
 gee S-function, version 4.13 modified 98/01/27 (1998)

Model:
 Link:                      Logit
 Variance to Mean Relation: Binomial
 Correlation Structure:     Exchangeable

Coefficients:
                Estimate Naive S.E.     Naive z Robust S.E.    Robust z
(Intercept) -0.125325730 0.06782579 -1.84775925  0.06758212 -1.85442135
gender       0.003437873 0.08790630  0.03910838  0.08784072  0.03913758
z1           0.149347107 0.02814374  5.30658404  0.02973865  5.02198729
z2           0.052017986 0.02815145  1.84779075  0.02704703  1.92324179

Working Correlation
          [,1]      [,2]      [,3]
[1,] 1.0000000 0.8173308 0.8173308
[2,] 0.8173308 1.0000000 0.8173308
[3,] 0.8173308 0.8173308 1.0000000

> fit.gee2 <- gee(response ~ gender + z1 + z2, id=case, family=binomial,
+            corstr="independence", data=abortion)

> summary(fit.gee2)

 Link:                      Logit
 Variance to Mean Relation: Binomial
 Correlation Structure:     Independent

Coefficients:
                Estimate Naive S.E.     Naive z Robust S.E.    Robust z
(Intercept) -0.125407576 0.05562131 -2.25466795  0.06758236 -1.85562596
```

```
gender       0.003582051 0.05415761  0.06614123  0.08784012  0.04077921
z1           0.149347113 0.06584875  2.26803253  0.02973865  5.02198759
z2           0.052017989 0.06586692  0.78974374  0.02704704  1.92324166

Working Correlation
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

From the *geepack* library, the function *geeglm* performs fitting of clustered data using the GEE method. See

`www.jstatsoft.org/v15/i02/paper`

for details, including an example for a binary response. Possible working correlation structures include independence, exchangeable, autoregressive (ar1), and unstructured. In addition to the sandwich covariance matrix (which is the default), when the number of clusters is small one can find a jackknife estimator. Fitting statements have the form:

```
> geeglm(y ~ x1 + x2, family=binomial, id=subject, corst=''exchangeable'')
```

The library *repolr* has a function *repolr* for GEE methods with ordinal responses:

`cran.r-project.org/web/packages/repolr/repolr.pdf`

Here is an example for the insomnia data of Table 12.3, using the independence working correlation structure (Thanks to Anestis Touloumis).

```
> insomnia<-read.table("insomnia.dat",header=TRUE)
> insomnia<-as.data.frame(insomnia)
> insomnia
     case treat occasion outcome
      1      1     0        1
      1      1     1        1
      2      1     0        1
      2      1     1        1
      3      1     0        1
      3      1     1        1
      4      1     0        1
      4      1     1        1
      5      1     0        1
...
    239      0     0        4
    239      0     1        4

> library(repolr)
> fit <- repolr(formula = outcome ~ treat + occasion + treat * occasion,
     + subjects="case", data=insomnia, times=c(1,2), categories=4,
      corstr = "independence")
> summary(fit$gee)
```

```
Coefficients:
                  Estimate Naive S.E.     Naive z Robust S.E.    Robust z
factor(cuts)1  -2.26708899  0.2027367 -11.1824294   0.2187606 -10.3633343
factor(cuts)2  -0.95146176  0.1784822  -5.3308499   0.1809172  -5.2591017
factor(cuts)3   0.35173977  0.1726860   2.0368745   0.1784232   1.9713794
treat           0.03361002  0.2368973   0.1418759   0.2384374   0.1409595
occasion        1.03807641  0.2375992   4.3690229   0.1675855   6.1943093
treat:occasion  0.70775891  0.3341759   2.1179234   0.2435197   2.9063728
```

ML for marginal models

Joseph Lang at the Univ. of Iowa has R and S-Plus functions such as *mph.fit* for ML fitting of marginal models (when the explanatory variables are categorical and not numerous) through the generalized loglinear model (10.10). This uses the constraint approach with Lagrange multipliers.

Chapters 13–14: Clustered Categorical Responses: Random Effects Models

Generalized linear mixed models

The function *lmer* (linear mixed effects in R) in the R package *Matrix* can be used to fit generalized linear mixed models. See the Gelman and Hill (2007) text, such as Sec. 12.4. See also the *lme4* package, described in `http://cran.r-project.org/web/packages/lme4/vignettes/Theory.pdf` These use adaptive Gauss–Hermite quadrature.

The function *glmm* in the *repeated* library can fit generalized linear mixed models using Gauss–Hermite quadrature methods, for families including the binomial and Poisson:

`rss.acs.unt.edu/Rdoc/library/repeated/html/glmm.html`

The package glmmAK can also fit them, with a Bayesian approach with priors for the fixed effects parameters:

`cran.r-project.org/web/packages/glmmAK/glmmAK.pdf`

The function glmmML in the *glmmML* package can fit GLMMs with random intercepts by adaptive Gauss–Hermite quadrature. For instance, in the following code we use it to analyze the opinions about abortion data analyzed in Sec. 13.3.2 with random effects models, employing Gauss-Hermite quadrature with 50 quadrature points and a starting value of 9 for the estimate of σ.

```
> abortion
     gender response question case
1         1        1        1    1
2         1        1        2    1
3         1        1        3    1
4         1        1        1    2
5         1        1        2    2
6         1        1        3    2
```

```
...
5548      0         0         1 1850
5549      0         0         2 1850
5550      0         0         3 1850

> z1 <- ifelse(abortion$question==1,1,0)
> z2 <- ifelse(abortion$question==2,1,0)
> z3 <- ifelse(abortion$question==3,1,0)

> library(glmmML)

> fit.glmmML <- glmmML(response ~ gender + z1 + z2,
+            cluster=abortion$case, family=binomial, data=abortion,
+            method = 'ghq', n.points=50, start.sigma=9)
> summary(fit.glmmML)

Call:  glmmML(formula = response ~ gender + z1 + z2, family = binomial,
              data = abortion, cluster = abortion$case, start.sigma = 9,
              method = "ghq", n.points = 50)

                coef se(coef)        z Pr(>|z|)
(Intercept) -0.62222   0.3811 -1.63253 1.03e-01
gender       0.01272   0.4936  0.02578 9.79e-01
z1           0.83587   0.1599  5.22649 1.73e-07
z2           0.29290   0.1568  1.86822 6.17e-02

Scale parameter in mixing distribution:  8.788 gaussian
Std. Error:                              0.5282

        LR p-value for H_0: sigma = 0:  0

Residual deviance: 4578 on 5545 degrees of freedom      AIC: 4588
```

The function *glmmPQL* in the MASS library can fit GLMMs using penalized quasi-likelihood. The R package *MCMCglmm* can fit them with Markov Chain Monte Carlo methods:

`cran.r-project.org/web/packages/MCMCglmm/vignettes/CourseNotes.pdf`

For a text on GLMMs using R, see *Multivariate Generalized Linear Mixed Models* by D. M. Berridge and R. Crouchley, published 2011 by CRC Press. The emphasis is on multivariate models, using the Sabre software package in R.

Item response models

Dimitris Rizopoulos from Leuven, Belgium has prepared a package *ltm* for Item Response Theory analyses. This package can fit the Rasch model, the two-parameter logistic model, Birnbaum's three-parameter model, the latent trait model with up to two latent variables, and Samejima's graded response model:

`med.kuleuven.be/biostat/software/software.htm#LatentIRT`

Latent class models

Steve Buyske at Rutgers has prepared a library for fitting latent class models with the EM algorithm:

`www.stat.rutgers.edu/home/buyske/software.html`

Beta-binomial and quasi-likelihood analyses

The following shows the beta-binomial and quasi-likelihood analyses of the teratology data presented in Sec. 14.3.4, continuing with the analyses shown above at the end of the R discussion for Chapter 4. Beta-binomial modeling is an option with the *vglm* function in the VGAM library (using Fisher scoring) and the *betabin* function in the aod library. It seems that *vglm* in VGAM uses Fisher scoring and hence reports SE values based on the expected information matrix, whereas *betabin* in aod uses the observed information matrix. Quasi-likelihood with the beta-binomial type variance is available with the *quasibin* function in the *aod* library. (In the following example, the random part of the statement specifies the same overdispersion for each group). For details about the *aod* package, see

`cran.r-project.org/web/packages/aod/aod.pdf`

Again, we borrow heavily from Laura Thompson's excellent manual.

```
> group <- rats$group
> library(VGAM)  #  We use Thomas Yee's VGAM library
> fit.bb <- vglm(cbind(y,n-y) ~ group, betabinomial(zero=2,irho=.2),
          data=rats)
  # two parameters, mu and rho, and zero=2 specifies 0 covariates for 2nd
  # parameter (rho);  irho is the initial guess for rho in beta-bin variance.
> summary(fit.bb)  # fit of beta-binomial model

Coefficients:
                Value Std. Error t value
(Intercept):1  1.3458    0.24412  5.5130
(Intercept):2 -1.1458    0.32408 -3.5355  # This is logit(rho)
group2        -3.1144    0.51818 -6.0103
group3        -3.8681    0.86285 -4.4830
group4        -3.9225    0.68351 -5.7387

Names of linear predictors: logit(mu), logit(rho)

Log-likelihood: -93.45675 on 111 degrees of freedom

> logit(-1.1458, inverse=T)    # This is a function in VGAM
[1] 0.2412571             # The estimate of rho in beta-bin variance

> install.packages("aod")  # another way to fit beta-binomial models
> library(aod)
> betabin(cbind(y,n-y) ~ group, random=~1,data=rats)
Beta-binomial model
```

```
-------------------
betabin(formula = cbind(y, n - y) ~ group, random = ~1, data = rats)

Fixed-effect coefficients:
              Estimate Std. Error    z value Pr(> |z|)
(Intercept)  1.346e+00  2.481e-01  5.425e+00 5.799e-08
group2      -3.115e+00  5.020e-01 -6.205e+00 5.485e-10
group3      -3.869e+00  8.088e-01 -4.784e+00 1.722e-06
group4      -3.924e+00  6.682e-01 -5.872e+00 4.293e-09

Overdispersion coefficients:
                 Estimate Std. Error   z value   Pr(> z)
phi.(Intercept) 2.412e-01  6.036e-02 3.996e+00 3.222e-05

> quasibin(cbind(y,n-y) ~ group, data=rats) # QL with beta-bin variance
Quasi-likelihood generalized linear model
-----------------------------------------
quasibin(formula = cbind(y, n - y) ~ group, data = rats)

Fixed-effect coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.2124     0.2233  5.4294   < 1e-4
group2       -3.3696     0.5626 -5.9893   < 1e-4
group3       -4.5853     1.3028 -3.5197    4e-04
group4       -4.2502     0.8484 -5.0097   < 1e-4

Overdispersion parameter:
   phi
0.1923

Pearson's chi-squared goodness-of-fit statistic = 54.0007
```

Negative binomial and other count models

As shown above in the Chapter 4 description for R, the *glm.nb* function in the MASS library is a modification of the *glm* function to handle negative binomial regression models:

`stat.ethz.ch/R-manual/R-patched/library/MASS/html/glm.nb.html`

The *negbin* function in the *aod* package can also handle negative binomial regression:

`cran.r-project.org/web/packages/aod/aod.pdf`

Thomas Yee's *VGAM* package can also fit zero-inflated Poisson models and negative binomial models.

Chapter 15: Non-Model-Based Classification and Clustering

Discriminant analysis

In the MASS library there is a *lda* function for linear discriminant analysis and a *qda* function for quadratic discriminant analysis:

`stat.ethz.ch/R-manual/R-patched/library/MASS/html/lda.html`

`stat.ethz.ch/R-manual/R-patched/library/MASS/html/qda.html`

Classification trees

In the *tree* library,

`cran.r-project.org/web/packages/tree/tree.pdf`

there is a *tree* function for binary recursive partitioning, and a *prune.tree* function for pruning them.

See also the *rpart* package and its *rpart* function for recursive partitioning to construct classification trees and *prune* function for pruning them:

`cran.r-project.org/web/packages/rpart/index.html`

For example, for the horseshoe crab data with width and quantitative color as predictors,

```
> library(tree)
> attach(crabs)
> fit <- rpart(y ~ color + width, method="class")
> plot(fit)
> text(fit)
> printcp(fit)

Classification tree:
rpart(formula = y ~ color + width, method = "class")

Variables actually used in tree construction:
[1] color width

Root node error: 62/173 = 0.35838

n= 173

        CP nsplit rel error  xerror     xstd
1 0.161290      0   1.00000 1.00000 0.101728
2 0.080645      1   0.83871 1.03226 0.102421
3 0.064516      2   0.75806 0.96774 0.100972
4 0.048387      3   0.69355 0.93548 0.100149
5 0.016129      4   0.64516 0.85484 0.097794
6 0.010000      6   0.61290 0.82258 0.096728
> plotcp(fit)
> summary(fit)
> plot(fit, uniform=TRUE,
   main="Classification Tree for Crabs")
> pfit2 <- prune(fit, cp= 0.02)
> plot(pfit2, uniform=TRUE,
   main="Pruned Classification Tree for Crabs")
plot(pfit2, uniform=TRUE,
+    main="Pruned Classification Tree for Crabs")
> text(pfit2, use.n=TRUE, all=TRUE, cex=.8)
> post(pfit2, file = "ptree2.ps",
   title = "Pruned Classification Tree for Crabs")
post(pfit2, file = "ptree2.ps",
+    title = "Pruned Classification Tree for Crabs")
```

Cluster analysis

The *dist* function in R computes distances to be used in a cluster analysis:

```
stat.ethz.ch/R-manual/R-patched/library/stats/html/dist.html
```

The *method="binary"* option invokes the Jaccard-type dissimilarity distance discussed in the text. The *method="manhattan"* option invokes L1-norm distance, which for binary data is the total number of variables that do not match. The *hclust* function can perform basic hierarchical cluster analysis, using inputted distances:

```
stat.ethz.ch/R-manual/R-patched/library/stats/html/hclust.html
```

For example, for the text example on election clustering using only the states in Table 15.5, with the manhattan distance and the average linkage method for summarizing dissimilarities between clusters,

```
> x <- read.table("election.dat", header=F)
> x
  V1 V2 V3 V4 V5 V6 V7 V8 V9
1   0  0  0  0  1  0  0  0
2   0  0  0  1  1  1  1  1
3   0  0  0  1  0  0  0  1
4   0  0  0  0  1  0  0  1
5   0  0  0  1  1  1  1  1
6   0  0  1  1  1  1  1  1
7   1  1  1  1  1  1  1  1
8   0  0  0  1  1  0  0  0
9   0  0  0  1  1  1  0  1
10  0  0  1  1  1  1  1  1
11  0  0  0  1  1  0  0  1
12  0  0  0  0  0  0  0  0
13  0  0  0  0  0  0  0  1
14  0  0  0  0  0  0  0  0
> distances <- dist(x,method="manhattan")
> states <- c("AZ", "CA", "CO", "FL", "IL", "MA", "MN",
              "MO", "NM", "NY", "OH", "TX", "VA", "WY")
> democlust <- hclust(distances,"average")
> postscript(file="dendrogram-election.ps")
> plot(democlust, labels=states)
> graphics.off()
```

Chapter 16: Large- and Small-Sample Theory for Multinomial Models

See the discussion for Chapters 1–3 above for information about special R functions for small-sample confidence intervals for association measures in contingency tables.

Alessandra Brazzale has prepared the *hoa* package for higher-order asymptotic analyses, including approximate conditional analysis for logistic and loglinear models:

`cran.r-project.org/web/packages/cond/vignettes/Rnews-paper.pdf`

`www.isib.cnr.it/~brazzale/lib.html`

A.3 Stata

For examples of categorical data analyses with Stata for many data sets in my text *An Introduction to Categorical Data Analysis*, see the useful site

`www.ats.ucla.edu/stat/examples/icda/`

set up by the UCLA Statistical Computing Center. Specific examples are linked below. See also *A Handbook of Statistical Analyses Using Stata*, 4th ed., by S. Rabe-Hesketh and B. Everitt (CRC Press, 2006). A listing of the extensive selection of categorical data methods available as of 2002 in Stata was given in Table 3 of the article by R. A. Oster in the August 2002 issue of *The American Statistician* (pp. 235-246); the main focus of that article is on methods for small-sample exact analysis.

Chapter 1: Introduction

The *ci* command can construct confidence intervals for proportions, including Wald, score (Wilson), Agresti–Coull, Jeffreys Bayes, and Clopper–Pearson small-sample methods. See

`www.stata.com/help.cgi?ci`

The *bitest* command can conduct small-sample tests about a binomial parameter. See

`www.stata.com/help.cgi?bitest`

Chapters 2–3: Two-Way Contingency Tables

The *tabulate* command can construct two-way contingency tables, conduct chi-squared tests and Fisher's exact test, and find various measures of association and their standard errors, including Goodman and Kruskal's gamma and Kendall's tau-b. See

`www.stata.com/help.cgi?tabulate_twoway`

for a summary and a list of options. See

`www.ats.ucla.edu/stat/stata/examples/icda/icdast2.htm`

for an example.

The *cs* command can construct confidence intervals for the difference of proportions, relative risk, and odds ratio. See

`www.stata.com/help.cgi?cs`

and for an example, see `www.ats.ucla.edu/stat/stata/examples/icda/icdast2.htm`, which also shows how to use *logit* to obtain an interval for the odds ratio. The *cc* command can also construct confidence intervals for odds ratios. See

`www.stata.com/help.cgi?cc`

and for an example, see `www.ats.ucla.edu/stat/stata/examples/icda/icdast3.htm`.

For a Stata module for three-way tables, one can use the *tab3way* command,

`ideas.repec.org/c/boc/bocode/s425301.html`

with an example at `www.ats.ucla.edu/stat/stata/examples/icda/icdast3.htm`. See also `www.stata.com/statalist/archive/2009-04/msg00893.html` and `www.stata.com/statalist/archive/2010-04/msg00800.html`.

Nicholas Cox has a package *tabchi* for basic analyses of contingency tables. See

`http://www.nd.edu/~rwilliam/stats1/Categorical-Stata.pdf`

for a document by Richard Williams that describes this and the use of Stata for basic analyses for categorical data analysis. In particular, it can generate standardized (adjusted) residuals, as shown in the example in `www.ats.ucla.edu/stat/stata/examples/icda/icdast2.htm`.

Chapter 4: Generalized Linear Models

The *glm* command can fit generalized linear models such as logistic regression and loglinear models:

`www.stata.com/help.cgi?glm`

The link functions (with keywords in parentheses) include log (log), identity (i), logit (l), probit (p), complementary log-log (c). The families include binomial (b), Poisson (p), and negative binomial (nb). Newton-Raphson fitting is the default. Code takes the form

.glm y x1 x2, family(poisson) link(log) lnoffset(time)

for a Poisson model with explanatory variables $x1$ and $x2$, and for a binomial variate y based on n successes,

.glm y x1 x2, family(binomial n) link(logit)

for a logistic model. For examples, see `www.ats.ucla.edu/stat/stata/examples/icda/icdast4.htm`.

Profile likelihood confidence intervals are available with the *pllf* and *logprof* (for logistic regression) commands in Stata. For *pllf*, see article by P. Royston in *Stata Journal*, vol. 7, pp. 376–387:

`www.stata-journal.com/sjpdf.html?articlenum=st0132`

Chapters 5–7: Logistic Regression and Binary Response Methods

For a summary of all the Stata commands that can perform logistic regression, see

`www.stata.com/capabilities/logistic.html`.

Once a model has been fitted, the *predict* command has various options, including fitted values, the Hosmer–Lemeshow statistic, standardized residuals, and influence diagnostics.

In particular, other than with the *glm* command, logistic models can be fitting using the *logistic* and *logit* commands. See

`www.stata.com/help.cgi?logistic` and `www.stata.com/help.cgi?logit`.

Code has the form

.logit y x [fw=count]

with the option of frequency weights. For examples, see `www.ats.ucla.edu/stat/stata/examples/icda/icdast4.htm`,

and for the horseshoe crab data and AIDS/AZT examples of Chapter 5, see `www.ats.ucla.edu/stat/stata/examples/icda/icdast5.htm`.
For a special command for grouped data, see `www.stata.com/help.cgi?glogit`

In the *glm* command, other links, such as probit and cloglog, can be substituted for the logit. Probit models can also be fitting using *probit*. See

`www.stata.com/help.cgi?probit`

Conditional logistic regression can be conducted using the *clogit* command. See `www.stata.com/help.cgi?clogit`.
The *exlogistic* command performs exact conditional logistic regression. See `www.stata.com/help.cgi?exlogistic`

FIRTHLOGIT is a Stata module to use Firth's method for bias reduction in logistic regression. See

`ideas.repec.org/c/boc/bocode/s456948.html`

See also `http://www.homepages.ucl.ac.uk/~ucakgam/stata.html` for information about a package of penalized logistic regression programs that also includes the lasso as a special case.

Stata does not seem to currently have Bayesian capability.

Chapter 8: Multinomial Response Models

The command *mlogit* can fit baseline-category logit models:

`www.stata.com/help.cgi?mlogit`

The code for a baseline-category logit model takes the form

```
. mlogit y x1 x2 [fweight=freq]
```

For the alligator food choice example of the text, but using three outcome categories, see `www.ats.ucla.edu/stat/stata/examples/icda/icdast8.htm`.

The command *mprobit* fits multinomial probit models, for the case of independent normal error terms. See

`http://www.stata.com/help.cgi?mprobit`

for details. The command *asmprobit* allows more general structure for the error terms.

The command *ologit* can fit ordinal models, such as cumulative logit models:

`www.stata.com/help.cgi?ologit`

The code for the proportional odds version of cumulative logit models has form

```
. ologit y x [fweight=freq]
```

For an example, see `www.ats.ucla.edu/stat/stata/examples/icda/icdast8.htm`. The corresponding command *oprobit* can fit cumulative probit models. See
`www.nd.edu/~rwilliam/oglm`
for discussion of a new *oglm* command by Richard Williams for ordinal models that include as a special case cumulative link models with logit, probit, and complementary log-log link. Continuation-ratio logit models can be fitted with the *ocratio* module. See
`www.stata.com/search.cgi?query=ocratio`.
A command *omodel* is available from the Stata website for fitting these models and

testing the assumption of the same effects for each cumulative probability (i.e., the proportional odds assumption for cumulative logit models).

Chapters 9–10: Loglinear Models

Loglinear models can be fitted as generalized linear models using the *glm* command. For examples, including the high school student survey of alcohol, cigarette, and marijuana use from Chapter 9, see
`www.ats.ucla.edu/stat/stata/examples/icda/icdast6.htm`.
That source also describes use of a special *ipf* command for iterative proportional fitting.

For an example of using *glm* to fit an association model such as linear-by-linear association, see `www.ats.ucla.edu/stat/stata/examples/icda/icdast7.htm`. An example shown is the text example from Chapter 10 on opinions about premarital sex and birth control.

Chapter 11: Models for Matched Pairs

Most models in this chapter can be fitted as special cases of logistic or loglinear models, which are themselves special cases of generalized linear models with the *glm* command. Some specialized commands are also available. For example, *symmetry* tests symmetry and marginal homogeneity in square tables, and thus gives McNemar's test for the special case of 2×2 tables. See

`http://www.stata.com/help.cgi?symmetry`

and see also `www.ats.ucla.edu/stat/stata/examples/icda/icdast9.htm` for an example and the use of the *mcc* command for McNemar's test. That location also shows analyses of the coffee choice example from the text, and also the use of *glm* for fitting the Bradley–Terry model, with a tennis example.

The command *clogit* performs conditional logistic regression.

Chapters 12–14: Clustered Categorical Responses

For information about using GEE in Stata, see

`www.stata.com/meeting/1nasug/gee.pdf`

by Nicholas Horton (in 2001). The GEE method can be conducted using the *xtgee* command, see

`www.stata.com/help.cgi?xtgee` and `www.stata.com/capabilities/xtgee.html`,

with the usual distributions and link functions for the marginal models. Code has form such as

```
. xtgee y x1 x2, family(poisson) link(log) corr(exchangeable) robust
```

For ML fitting of generalized linear mixed models, the *GLLAMM* module described at `www.gllamm.org` can fit a very wide variety of models, including logistic and cumulative logit models with random effects. For further details, see
`www.stata.com/search.cgi?query=gllamm`
and Chapter 5 of *Multilevel and Longitudinal Modeling Using Stata* by S. Rabe-Hesketh

and A. Skrondal (Stata Press, 2005). For a discussion of its use of adaptive Gauss-Hermite quadrature, see `www.stata-journal.com/sjpdf.html?articlenum=st0005`.

Negative binomial regression models can be fitted with the *nbreg* command. See

`www.stata.com/help.cgi?nbreg` and `www.ats.ucla.edu/stat/stata/dae/nbreg.htm`.

It is also possible to fit these models with the *glm* command, with the nbinomial option for the family. See `www.ats.ucla.edu/stat/stata/library/count.htm`.

Chapter 15: Non-Model-Based Classification and Clustering

There is a *cart* module for classification trees, prepared by Wim van Putten. See `econpapers.repec.org/software/bocbocode/s456776.htm`.

Discriminant analysis is available with the *discrim* command. Options include linear discriminant analysis (subcommand *lda*, that is, the full command is *discrim lda*), quadratic discriminant analysis with subcommand *qda*, *k* nearest neighbor with subcommand *knn*, and logistic with subcommand *logistic*. See

`www.stata.com/help.cgi?discrim`

and `http://www.stata.com/help.cgi?candisc` for the canonical linear discriminant function.

For a summary of Stata capabilities for cluster analysis with the *cluster* command, see

`www.stata.com/capabilities/cluster.html` and `www.stata.com/help.cgi?cluster`.

Chapter 16: Large- and Small-Sample Theory for Multinomial Models

The *ci* command can construct small-sample confidence intervals for proportions, including Clopper–Pearson intervals. See

`www.stata.com/help.cgi?ci`

The *cc* command constructs small-sample confidence intervals for the odds ratio, unless one requests a different option. See

`www.stata.com/help.cgi?cc`

The *exlogistic* command performs exact conditional logistic regression. See `www.stata.com/help.cgi?exlogistic`

A.4 SPSS

Chapters 1–3: Introduction, Two-Way Contingency Tables

The DESCRIPTIVE STATISTICS option on the ANALYZE menu has a suboption called CROSSTABS, which provides several methods for contingency tables. After identifying the row and column variables in CROSSTABS, clicking on STATISTICS provides a wide variety of options, including the chi-squared test and measures of association. The output lists the Pearson statistic, its degrees of freedom, and its P-value (labeled Asymp. Sig.). If any expected frequencies in a 2×2 table are less than 5, Fisher's exact test results. It can also be requested by clicking on Exact in the CROSSTABS dialog box and selecting the exact test. SPSS also has an advanced module for small-sample inference (called SPSS Exact Tests) that provides exact P-values for various tests in CROSSTABS and NPAR TESTS procedures. For instance, the Exact Tests module provides exact tests of independence for $I \times J$ contingency tables with nominal or ordinal classifications. See the publication *SPSS Exact Tests for Windows*.

In CROSSTABS, clicking on CELLS provides options for displaying observed and expected frequencies, as well as the standardized residuals, labeled as "Adjusted standardized". Clicking on STATISTICS in CROSSTABS provides options of a wide variety of statistics other than chi-squared, including gamma and Kendall's tau-b. The output shows the measures and their standard errors (labeled Asymp. Std. Error), which you can use to construct confidence intervals. It also provides a test statistic for testing that the true measure equals zero, which is the ratio of the estimate to its standard error. This test uses a simpler standard error that only applies under independence and is inappropriate for confidence intervals. One option in the list of statistics, labeled Risk, provides as output the odds ratio and its confidence interval.

Suppose you enter the data as cell counts for the various combinations of the two variables, rather than as responses on the two variables for individual subjects; for instance, perhaps you call COUNT the variable that contains these counts. Then, select the WEIGHT CASES option on the DATA menu in the Data Editor window, instruct SPSS to weight cases by COUNT.

Chapter 4: Generalized Linear Models

To fit generalized linear models, on the ANALYZE menu select the GENERALIZED LINEAR MODELS option and the GENERALIZED LINEAR MODELS suboption. Select the Dependent Variable and then the Distribution and Link Function. Click on the Predictors tab at the top of the dialog box and then enter quantitative variables as Covariates and categorical variables as Factors. Click on the Model tab at the top of the dialog box and enter these variables as main effects, and construct any interactions that you want in the model. Click on OK to run the model.

Chapters 5–7: Logistic Regression and Binary Response Methods

To fit logistic regression models, on the ANALYZE menu select the REGRESSION option and the BINARY LOGISTIC suboption. In the LOGISTIC REGRESSION dialog box, identify the binary response (dependent) variable and the explanatory predictors (covariates). Highlight variables in the source list and click on a*b to create an interaction term. Identify the explanatory variables that are categorical and for which you want indicator variables by clicking on Categorical and declaring such a covariate to be a Categorical Covariate in the LOGISTIC REGRESSION: DEFINE CATEGORICAL VARIABLES dialog box. Highlight the categorical covariate and under Change Contrast you will see several options for setting up indicator variables. The Simple contrast constructs them as in this text, in which the final category is the baseline.

In the LOGISTIC REGRESSION dialog box, click on Method for stepwise model selection procedures, such as backward elimination. Click on Save to save predicted probabilities, measures of influence such as leverage values and DFBETAS, and standardized residuals. Click on Options to open a dialog box that contains an option to construct confidence intervals for exponentiated parameters.

Another way to fit logistic regression models is with the GENERALIZED LINEAR MODELS option and suboption on the ANALYZE menu. You pick the binomial distribution and logit link function. It is also possible there to enter the data as the number of successes out of a certain number of trials, which is useful when the data are in contingency table form. One can also fit such models using the LOGLINEAR option with the LOGIT suboption in the ANALYZE menu. One identifies the dependent variable, selects categorical predictors as factors, and selects quantitative predictors as cell covariates. The default fit is the saturated model for the factors, without including any covariates. To change this, click on Model and select a Custom model, entering the predictors and relevant interactions as terms in a customized (unsaturated) model. Clicking on Options, one can also display standardized residuals (called adjusted residuals) for model fits. This approach is well suited for logit models with categorical predictors, since standard output includes observed and expected frequencies. When the data file contains the data as cell counts, such as binomial numbers of successes and failures, one weights each cell by the cell count using the WEIGHT CASES option in the DATA menu.

Chapter 8: Multinomial Response Models

SPSS can fit logistic models for multinomial response variables. On the ANALYZE menu, choose the REGRESSION option and then the ORDINAL suboption for a cumulative logit model. Select the MULTINOMIAL LOGISTIC suboption for a baseline-category logit model. In the latter, click on Statistics and check Likelihood-ratio tests under Parameters to obtain results of likelihood-ratio tests for the effects of the predictors.

Chapters 9–10: Loglinear Models

For loglinear models, one uses the LOGLINEAR option with GENERAL suboption in the ANALYZE menu. One enters the factors for the model. The default is the saturated model, so click on Model and select a Custom model. Enter the factors as terms in a customized (unsaturated) model and then select additional interaction effects. Click on Options to show options for displaying observed and expected frequencies and adjusted residuals. When the data file contains the data as cell counts for the various combinations of factors rather than as responses listed for individual subjects, weight each cell by the cell count using the WEIGHT CASES option in the DATA menu.

Chapter 11: Models for Matched Pairs

The models discussed in this chapter are almost all generalized linear models and can be fitted as described above for Chapter 4. The LOGLINEAR option just mentioned for Chapters 9–10 can also be used.

Chapters 12–14: Clustered Categorical Responses

For GEE methods, on the ANALYZE menu choose the GENERALIZED LINEAR MODELS option and the GENERALIZED ESTIMATING EQUATIONS suboption. You can then select structure for the working correlation matrix and identify the between-subject and within-subject variables.

For random effects models, on the ANALYZE menu choose the MIXED MODELS option and the GENERALIZED LINEAR suboption.

Version 19 apparently has capability of fitting generalized linear mixed models:

`www.unileon.es/ficheros/servicios/informatica/spss/english/IBM-SPSS_advanced_statistics.pdf`

`www-01.ibm.com/software/analytics/spss/products/statistics/advanced-statistics`

Chapter 15: Non-Model-Based Classification and Clustering

SPSS Categories is an add-on module that provides optimal scaling procedures such as categorical principal components analysis and multidimensional scaling, and some reduction-dimension techniques such as correspondence analysis, biplots, and canonical correlation analysis.

A.5 StatXact and LogXact

For certain analyses, specialized software is better than the major packages. A good example is *StatXact* (Cytel Software, Cambridge, Massachusetts), which provides exact analysis for categorical data methods and some nonparametric methods. See `www.cytel.com/Software/StatXact.aspx` for details Among its procedures are small-sample confidence intervals for a binomial parameter, the difference of proportions, relative risk, and odds ratio, and Fisher's exact test and its generalizations for $I \times J$ tables. It can also conduct exact tests of conditional independence and of equality of odds ratios in $2 \times 2 \times K$ tables, and exact confidence intervals for the common odds ratio in several 2×2 tables. StatXact uses Monte Carlo methods to approximate exact P-values and confidence intervals when a data set is too large for exact inference to be computationally feasible. A listing of the extensive selection of small-sample methods available in StatXact as of 2002 was given in Table 1 of the article by R. A. Oster in the August 2002 issue of *The American Statistician* (pp. 235-246)

Its companion, *LogXact*, performs exact conditional logistic regression. It also provides exact conditional analyses for baseline-category logit models. See `www.cytel.com/Software/LogXact.aspx` for details.

A.6 OTHER SOFTWARE

HLM

HLM, from Scientific Software International (Chicago), fits multilevel models. See

`www.ssicentral.com/hlm`

For examples, see the useful site

`www.ats.ucla.edu/stat/hlm/examples/default.htm`

set up by the UCLA Statistical Computing Center.

Latent Gold

The Latent Gold program, marketed by Statistical Innovations (Belmont, MA), can fit a wide variety of finite mixture models such as latent class models (i.e. the latent variable is categorical), nonparametric mixtures of logistic regression, and some Rasch mixture models. It can handle binary, nominal, ordinal, and count response variables and can include random effects that are treated in a nonparametric method rather than assumed to have a normal distribution. See

`www.statisticalinnovations.com/products/latentgold.html`

See also

`www.ats.ucla.edu/stat/latent_gold/default.htm`

set up by the UCLA Statistical Computing Center.

LEM

LEM is a general program for latent class modeling. See www.tilburguniversity.edu/nl/over-tilburg-university/schools/socialsciences/organisatie/departement

and also the site

www.ats.ucla.edu/stat/lem/default.htm

set up by the UCLA Statistical Computing Center.

LIMDEP and NLOGIT

LIMDEP is designed for modeling limited dependent variables, including multinomial discrete choice models and count data models. NLOGIT is designed for nested logit models and multinomial logit models, and can handle extended discrete choice models that do not appear in LIMDEP. See

www.limdep.com/

For examples of LIMDEP from Greene's *Econometric Analysis*, see the useful site

www.ats.ucla.edu/stat/limdep/examples/default.htm

set up by the UCLA Statistical Computing Center.

MAREG

The program MAREG (Kastner et al. 1997) provides GEE fitting and ML fitting of marginal models with the Fitzmaurice and Laird (1993) approach, allowing multicategory responses. See

www.stat.uni-muenchen.de/sfb386/software/mareg/winmareg.html

MLwiN

MLwiN is a software package for fitting multilevel models. See

www.cmm.bristol.ac.uk/MLwiN

For examples, see the useful site

www.ats.ucla.edu/stat/mlwin/examples

set up by the UCLA Statistical Computing Center.

PASS

PASS, marketed by NCSS Statistical Software (Kaysville, Utah), provides power analyses and sample size determination.

SUDAAN

SUDAAN, from the Research Triangle Institute (Research Triangle Park, North Carolina), provides analyses for categorical and continuous data from stratified multi-stage cluster designs. See

www.rti.org/sudaan/

It has facility (MULTILOG procedure) for GEE analyses of marginal models for nominal and ordinal responses. See

`www.rti.org/sudaan/onlinehelp/sudaanMULTILOG_Procedure.html`

For examples, see the useful site

`www.ats.ucla.edu/stat/sudaan/examples/default.htm`

set up by the UCLA Statistical Computing Center.

SuperMix

SuperMix, distributed by Scientific Software International, provides ML fitting of generalized linear mixed models, including count responses, nominal responses, and ordinal responses using cumulative links including the cumulative logit, cumulative probit, and cumulative complementary log-log. This program is based on software developed over the years by Donald Hedeker and Robert Gibbons, who have also done considerable research on mixed models. For multilevel models, the program is supposed to be much faster than PROC MIXED or PROC NLMIXED in SAS and make it possible to fit relatively complex models using ML rather than approximations such as penalized quasi likelihood (communication from Robert Gibbons). See

`www.ssicentral.com/supermix/index.html`

Other Software

For software for the Berger - Boos test and other small-sample unconditional tests for 2×2 tables, see

`www.west.asu.edu/rlberge1/software.html`

For a variety of permutation analyses for categorical and continuous variables, including some multivariate analyses, using SAS macros constructed by Luigi Salmaso and Fortunato Pesarin and others at the University of Padova, see

`homes.stat.unipd.it/pesarin/software.html`

Robert Newcombe at the University of Wales in Cardiff provides an Excel spreadsheet for forming various confidence intervals for a proportion and for comparing two proportions with independent or with matched samples. His website also has SPSS and Minitab macros for doing this. See

`medicine.cf.ac.uk/en/research/research-groups/clinical-epidemiology/resources/`