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 `http://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.

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`

An upcoming excellent resource of the use of R for contingency table analysis is a soon-to-appear book by Dr. Maria Kateri. That text also provides functions for many methods not readily available in R.

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).

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 *pbinom(6, 10, 0.5)* would give the corresponding cumulative probability of 6 or fewer successes.

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 *proportion* package contains a great variety of confidence intervals for a binomial parameter, including Wald, likelihood-ratio, and score intervals. For instance, for 95% confidence intervals based on 0 successes in 10 trials, the package reports:

```
------------------------------------------------------------------------
> library(proportion)

> ciAllx(0, 10, 0.05)
      method x   LowerLimit UpperLimit
1       Wald 0 0.000000e+00 0.00000000
3 Likelihood 0 2.525061e-05 0.17481827
4      Score 0 2.005249e-17 0.27753280
------------------------------------------------------------------------
```

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. Some of these are also available in the *PropCIs* package prepared by Ralph Scherer at the Institute for Biometry in Hannover, Germany. For instance:

```
------------------------------------------------------------------------------
> library(PropCIs)

> addz2ci(9, 10, 0.95) # Agresti-Coull CI adding z^2 to success and failures
95 percent confidence interval:
 0.5740323 1.0000000

> add4ci(9, 10, 0.95) # Wald CI after add 2 successes and 2 failures
95 percent confidence interval:
 0.5707764 1.0000000

> scoreci(9, 10, 0.95) # score CI
```

```
95 percent confidence interval:
 0.5958 0.9821
------------------------------------------------------------------------
```

The confidence interval based on the test using mid P-value is available with the *midPci* function in the `PropCIs` package.

```
------------------------------------------------------------------------
> library(PropCIs)

> midPci(9, 10, 0.95) # 9 successes in 10 trials
95 percent confidence interval:
 0.5966 0.9946
------------------------------------------------------------------------
```

Binomial tests and confidence intervals using the mid P-value are available with the *exactci* package. Other available inferences with that package include the Blaker exact confidence interval.

```
------------------------------------------------------------------------
> library(exactci)

> binom.exact(9, 10, 0.50, alternative=c("greater"), midp=TRUE)
number of successes = 9, number of trials = 10, p-value = 0.005859

> binom.exact(9, 10, conf.level=0.95, midp=TRUE)
95 percent confidence interval:
 0.5965206 0.9950264

> binom.exact(9, 10, conf.level=0.95, tsmethod=c("blaker"))
95 percent confidence interval:
 0.5555 0.9949
------------------------------------------------------------------------
```

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)

        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.

Bayesian inference

For a Bayesian posterior interval based on a beta(α_1, α_2) prior distribution and y successes and $n-y$ failures, we find percentiles of the beta distribution with parameters α_1+y and α_2+n-y using the quantile function *qbeta*. We find a 95% posterior interval here based on the uniform prior ($\alpha_1 = \alpha_2 = 1.0$) and $y = 0$ and $n - y = 25$. Likewise, we can use the *pbeta* function of cumulative probabilities to find a tail probability, such as the posterior probability that π is at least 0.50:

```
------------------------------------------------------------------------
> qbeta(0.025, 1, 26); qbeta(0.975, 1, 26)
[1] 0.0009732879
[1] 0.1322746

> pbeta(0.50, 1, 26)  # posterior beta cumulative prob. at 0.50
[1] 1
> 1 - pbeta(0.50, 1, 26) # right-tail prob. above 0.50
[1] 1.490116e-08

> library(proportion}
> ciBAx(0, 25, 0.05, 1.0, 1.0) # y, n, CI error prob, beta prior values
  x        LBAQx     UBAQx        LBAHx     UBAHx # 2nd set are HPD
1 0 0.0009732879 0.1322746 2.440083e-10 0.1088304
------------------------------------------------------------------------
```

One can also use the *proportion* package for Bayesian posterior intervals based on beta priors, as shown above.

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 *stdres* 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
```

Likewise, replacing the *stdres* component by *expected* would generate the expected frequency estimates. 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 an example of using Mantel's ordinal test for a two-way table with the M^2 statistic of Section 3.4 applied to the infant birth defects example:

```
------------------------------------------------------------------------------
> Malform <- matrix(c(17066, 14464, 788, 126, 37, 48, 38, 5, 1, 1), ncol=2)
> Malform
      [,1] [,2]
[1,] 17066   48
[2,] 14464   38
[3,]   788    5
[4,]   126    1
[5,]    37    1
> library(vcdExtra)
> CMHtest(Malform, rscores=c(0, 0.5, 1.5, 4.0, 7.0))
Cochran-Mantel-Haenszel Statistics # test was proposed by Nathan Mantel
                 AltHypothesis   Chisq Df     Prob
cor        Nonzero correlation  6.5699  1 0.010372
------------------------------------------------------------------------------
```

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)

Fisher's Exact Test for Count Data

data:  tea
p-value = 0.4857
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
   0.2117329 621.9337505
sample estimates:
odds ratio
  6.408309

> fisher.test(tea,alternative="greater")

Fisher's Exact Test for Count Data

data:  tea
p-value = 0.2429
alternative hypothesis: true odds ratio is greater than 1
```

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 (proposed by Agresti, Wackerly, and Boyett, 1979).

For mid-P values, you can use the *exact2x2* package:

```
------------------------------------------------------------------------
> library(exact2x2)
> fisher.exact(tea,midp=TRUE,conf.int=FALSE,alternative="greater")
p-value = 0.1286
alternative hypothesis: true odds ratio is greater than 1
> fisher.exact(tea,midp=TRUE,conf.level=0.95)
p-value = 0.2571
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
   0.3100451 311.4363036
------------------------------------------------------------------------
```

This also shows a confidence interval based on inverting the exact conditional nonnull hypergeometric distribution, but using the mid P-value.

Confidence intervals for association measures

For a 2×2 table, the function *prop.test* provides the Wald confidence interval for the difference of proportions, where one uses the option *correct=FALSE* to suppress the continuity correction.

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. 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. For example, Here, we illustrate for the example on aspirin and heart attacks

```
------------------------------------------------------------------------------
> library(PropCIs)

> diffscoreci(189, 11034, 104, 11037, 0.95) # CI for difference of proportions
95 percent confidence interval:# score CI
 0.004716821 0.010788501

> riskscoreci(189, 11034, 104, 11037, 0.95) # score CI for relative risk
95 percent confidence interval:
 1.433904 2.304713

> orscoreci(189, 11034, 104, 11037, 0.95) # score CI for odds ratio
95 percent confidence interval:
 1.440802 2.329551
------------------------------------------------------------------------------
```

Some of these functions are taken from

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

where former students of mine prepared R functions for confidence intervals comparing two proportions with independent samples (difference of proportions, relative risk, odds ratio), and the site

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

which has 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. Note that the score CI for the difference of proportions is based on the formula on p. 79 of the text based on the work of Mee (1984). The slightly different score interval suggested by Miettinen and Nurminen (1985) incorporates a bias correction factor in the variance of $(n_1 + n_2)/(n_1 + n_2 - 1)$ that can result in even better coverage properties according to an article by Newcombe and Nurminen (2011, *Communications in Statistics*, **40**: 1271–1282). Bernhard Klingenberg has written an R function for both of these score intervals, with the Miettinen and Nurminen as the default. For details, go to `http://sites.williams.edu/bklingen`. Here is Bernhard's code, with examples:

```
# from Bernhard Klingenberg
# returns score test statistic, p-value and restricted MLEs for diff
of prop p1 - p2:
score.test <- function(delta.null, y,n, alternative = "two.sided",
MN=TRUE) {
  N=sum(n)
  C=sum(y)
  L3=N
  L2=(n[1]+2*n[2])*delta.null-N-C
  L1=(n[2]*delta.null-N-2*y[2])*delta.null+C
  L0=y[2]*delta.null*(1-delta.null)
  c=L2^3/(3*L3)^3 - L1*L2/(6*L3^2) + L0/(2*L3)
  b=ifelse(c>=0,1,-1)*sqrt(L2^2/(3*L3)^2 - L1/(3*L3))
  d <- min(c/b^3,1)
  d <- max(d, -1)
  a=(3.14159265358979+acos(d))/3
  p2=2*b*cos(a)-L2/(3*L3)
  p2 = max(p2, 0)
  p2 = min(p2,1)
  p1=p2+delta.null
  p1 = max(p1, 0)
  p1 = min(p1,1)
  se0 <- sqrt(p1*(1 - p1)/n[1] + p2*(1 - p2)/n[2])
    if (!MN) z=(y[1]/n[1]-y[2]/n[2]-delta.null)/se0 #Mee
  else z=(y[1]/n[1]-y[2]/n[2]-delta.null)/(se0*N/(N-1)) #Mietinnen and
Nurminen
    if (se0 == 0){
  z = 0
  }
```

```
  pvalue <- switch(alternative,
                   "two.sided" = 1 - pchisq(z^2, df=1),
                   "less" = pnorm(z),
                   "greater" = pnorm(z, lower.tail = FALSE)
  )
  return(list(test.stat = z, p.value = pvalue, p1.null = p1, p2.null =
p2))
}

#returns lower and upper score bounds for diff of prop p1 - p2:
score.int <- function(y,n,conflev=0.95, type="two.sided", MN=TRUE) {
  if (type=="two.sided") c=qnorm(1-(1-conflev)/2)^2 else
c=qnorm(conflev)^2
  delta1=(y[1]+1)/(n[1]+2) - (y[2]+1)/(n[2]+2) #starting point
  if (any(type=="lower",type=="two.sided")) {
    delta2=-1
    while( abs(delta1-delta2)>10^(-6) ) {#Bisection for LB
      delta=(delta1+delta2)/2
      z=score.test(delta,y,n, MN = MN)$test.stat^2
      if (z>c) delta2=delta else delta1=delta
    }
  }
  delta.LB=delta1
  delta1=(y[1]+1)/(n[1]+2) - (y[2]+1)/(n[2]+2) #starting point
  if (any(type=="upper",type=="two.sided")) {
    delta2=1;
    while( abs(delta1-delta2)>10^(-6) ) {#Bisection for UB
      delta=(delta1+delta2)/2
      z=score.test(delta,y,n, MN = MN)$test.stat^2
      if (z>c) delta2=delta else delta1=delta
    }
  }
  delta.UB=delta1
  return(switch(type,
                "lower"=delta.LB,
                "upper"=delta.UB,
                "two.sided"=cbind(delta.LB,delta.UB)
  ))
}

### Example:
> y <- c(10,5)
> n <- c(20,20)
> score.int(y,n)
        delta.LB  delta.UB
[1,] -0.05793719 0.5161737
> score.int(y,n, MN=FALSE)
        delta.LB  delta.UB
[1,] -0.05026568 0.5104669
> score.test(delta.null=0,y,n)
```

```
$test.stat
[1] 1.592168
$p.value
[1] 0.1113469
$p1.null
[1] 0.375
$p2.null
[1] 0.375
> score.test(delta.null=0,y,n, MN=FALSE)
$test.stat
[1] 1.632993
$p.value
[1] 0.1024704
$p1.null
[1] 0.375
$p2.null
[1] 0.375
```

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
```

Fay (2010a) described an R package *exact2x2* 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

and

cran.r-project.org/web/packages/exact2x2/index.html

For example, for a 2×2 table called *data*, the command *exact2x2(data, tsmethod = "blaker")* provides an exact test using Blaker's P-value and the confidence interval based on inverting that test.

You can construct a small-sample confidence interval for the odds ratio based on the exact conditional test with mid P-value using the *epitools* package,

https://cran.r-project.org/web/packages/epitools/epitools.pdf

For the data in the tea-tasting example, we have:

```
------------------------------------------------------------------------------
> install.packages("epitools")
> library(epitools)
> ormidp.test(3, 1, 1, 3, or=1)
  one.sided two.sided
1 0.1285714 0.2571429
> or.midp(c(3,1,1,3), conf.level=0.95)
$conf.int
[1]   0.3100508 306.6338538
------------------------------------------------------------------------------
```

This is also available in the *exact2x2* package, although results differ slightly:

```
------------------------------------------------------------------------
> library(exact2x2)
> fisher.exact(tea,midp=TRUE,conf.level=0.95)
p-value = 0.2571
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
   0.3100451 311.4363036
------------------------------------------------------------------------
```

In a 2002 paper in the journal *Biostatistics*, Agresti and Min showed that for independent binomial samples, one can obtain shorter exact confidence intervals for the odds ratio using unconditional methods (inverting a score test) instead of the conditional method. Their exact unconditional confidence interval for an odds ratio is also available in the *exact2x2* package, using the *uncondExact2x2* function. For the example in the Agresti and Min paper with sample sizes of 26 and success counts of 1 and 2:

```
------------------------------------------------------------------------------
> library(exact2x2)
> uncondExact2x2(1,26,2,26,parmtype="oddsratio",conf.level=0.95,
+                conf.int=TRUE,method="score",tsmethod="square")

        Unconditional Exact Test on Odds Ratio, method= score, squared

data:  x1/n1=(1/26) and x2/n2= (2/26)
proportion 1 = 0.038462, proportion 2 = 0.076923, p-value = 0.6794
alternative hypothesis: true p2(1-p1)/[p1(1-p2)] is not equal to 1
95 percent confidence interval:
```

```
  0.229944 29.342162
-------------------------------------------------------------------------
```

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`

The *vcd* package can construct the ordinal measure of association gamma and its standard error, as well as many other things for contingency tables, such as odds ratios, mosaics plots, and CMH tests for stratified tables. See

`http://cran.us.r-project.org/web/packages/vcdExtra/vignettes/vcd-tutorial.pdf`

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). Here is an example for the difference of proportions and the odds ratio using Jeffreys priors for a table with counts (3, 1) in row 1 and (1, 3) in row 2.

```
-------------------------------------------------------------------------------
> diffCI(3, 4, 1, 4, 0.5, 0.5, 0.5, 0.5, 0.95)
[1] -0.1093044  0.8812513

> orCI(3, 4, 1, 4, 0.5, 0.5, 0.5, 0.5, 0.95)
[1]   0.4712263 249.6543515
-------------------------------------------------------------------------------
```

Missing data

The *ACD* package can conduct some analyses of categorical data (e.g. loglinear models by ML, functions of count data using weighted least squares) when data are missing. See

cran.r-project.org/web/packages/ACD/ACD.pdf

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. It is also available with the *profilelike.glm* function in the ProfileLikelihood library prepared by Leena Choi. See `cran.r-project.org/web/packages/ProfileLikelihood/ProfileLikelihood.pdf`.

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 (Note: The complete data set is in the Datasets link `www.stat.ufl.edu/~aa/cda/data.html` at this website),

```
> 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). (Note that the complete data set is in the Datasets link `www.stat.ufl.edu/~aa/cda/data.html` at this website.)

```
> 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
```

For the AIDS and AZT use example:

```
> race <- c(1,1,0,0)
> azt <- c(1,0,1,0)
> symptoms <- c(14,32,11,12)
> n <- c(107, 113,63,55)
> response <- matrix(c(symptoms, n-symptoms), ncol=2)
> fit <- glm(response ~ race + azt, family=binomial(link=logit))
> summary(fit)

Call:
glm(formula = response ~ race + azt, family = binomial(link = logit))

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept) -1.07357    0.26294  -4.083 4.45e-05 ***
race         0.05548    0.28861   0.192  0.84755
azt         -0.71946    0.27898  -2.579  0.00991 **
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8.3499  on 3  degrees of freedom
Residual deviance: 1.3835  on 1  degrees of freedom
AIC: 24.86

Number of Fisher Scoring iterations: 4
```

Hosmer-Lemeshow: One place I've seen a function for the Hosmer-Lemeshow test is `http://sas-and-r.blogspot.com/2010/09/example-87-hosmer-and-lemeshow-goodness.html`

The function *rstandard* can be used to request standardized residuals, after a binary glm fit. For example, for the Berkeley admissions data shown on p. 63, the model assuming no gender effect fits well except for the first department:

```
-----------------------------------------------------------------
> data <- read.table("berkeley.dat",header=TRUE)
> data
   dept gender yes  no
1     A   male 512 313
2     A female  89  19
3     B   male 353 207
4     B female  17   8
5     C   male 120 205
6     C female 202 391
7     D   male 138 279
8     D female 131 244
9     E   male  53 138
10    E female  94 299
11    F   male  22 351
12    F female  24 317
> attach(data)
> n <- yes + no
> fit <- glm(yes/n ~ factor(dept), weights=n, family=binomial)
> summary(fit)
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    0.59346    0.06838   8.679   <2e-16 ***
factor(dept)B -0.05059    0.10968  -0.461    0.645
factor(dept)C -1.20915    0.09726 -12.432   <2e-16 ***
factor(dept)D -1.25833    0.10152 -12.395   <2e-16 ***
factor(dept)E -1.68296    0.11733 -14.343   <2e-16 ***
factor(dept)F -3.26911    0.16707 -19.567   <2e-16 ***
---
```

```
    Null deviance: 877.056  on 11  degrees of freedom
Residual deviance:  21.736  on  6  degrees of freedom
AIC: 102.68

> rstandard(fit, type="pearson")
         1          2          3          4          5          6          7
-4.1530728  4.1530728 -0.5037077  0.5037077  0.8680662 -0.8680662 -0.5458732
         8          9         10         11         12
 0.5458732  1.0005342 -1.0005342 -0.6197526  0.6197526
-----------------------------------------------------------------------
```

Big data: 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`

Exact Conditional Logistic Regression

See the package *logistiX* at

`https://cemsiis.meduniwien.ac.at/kb/wf/software/statistische-software/logistiX/`

and the package *elrm* for MCMC approximation of exact conditional distributions.

ROC curves

ROC curves can be constructed with the ROCR library. For example, for a probit model for the beetle mortality data of Section 7.1.4,

```
> dose <- c(rep(1.691,59),rep(1.724,60),rep(1.755,62),rep(1.784,56),
+ rep(1.811,63),rep(1.837,59),rep(1.861,62),rep(1.884,60))
> y <- c(rep(1,6),rep(0,53),rep(1,13),rep(0,47),rep(1,18),rep(0,44),
+ rep(1,28),rep(0,28),rep(1,52),rep(0,11),rep(1,53),rep(0,6),
+ rep(1,61),rep(0,1),rep(1,60))
> fit.probit <- glm(y ~ dose, family=binomial(link=probit))
> summary(fit.probit)

            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -34.956      2.649  -13.20   <2e-16
dose          19.741      1.488   13.27   <2e-16
---

> library("ROCR")  # to construct ROC curve
> pred <- prediction(fitted(fit.probit),y)
> perf <- performance(pred, "tpr", "fpr")
> plot(perf)
> performance(pred,"auc")
Slot "y.values":
[[1]]
```

```
[1] 0.9010852      # area under ROC curve
```

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).

Infinite estimates

Here is an example of the use of R for Table 6.11, in which center effect ML estimates for centers 1 and 3 are actually $-\infty$.

```
-----------------------------------------------------------------
> data <- read.table("fungal.dat",header=TRUE)
> data
   center treatment y  n
1       1         1 0  5
2       1         0 0  9
3       2         1 1 13
```

```
4       2         0 0 10
5       3         1 0  7
6       3         0 0  5
7       4         1 6  9
8       4         0 2  8
9       5         1 5 14
10      5         0 2 14
> attach(data)

> fit <- glm(y/n ~ treatment + factor(center), weights=n, family=binomial)
> summary(fit)
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)     -2.459e+01  2.330e+04  -0.001   0.9992
treatment        1.546e+00  7.017e-01   2.203   0.0276 *
factor(center)2  2.039e+01  2.330e+04   0.001   0.9993
factor(center)3  4.809e-03  3.172e+04   0.000   1.0000
factor(center)4  2.363e+01  2.330e+04   0.001   0.9992
factor(center)5  2.257e+01  2.330e+04   0.001   0.9992
---
    Null deviance: 28.53202  on 9  degrees of freedom
Residual deviance:  0.50214  on 4  degrees of freedom
Number of Fisher Scoring iterations: 21

> fit2 <- glm(y/n ~ treatment + factor(center) -1, weights=n, family=binomial)
> summary(fit2)
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
treatment           1.5460     0.7017   2.203 0.027569 *
factor(center)1   -24.5922 23296.3959  -0.001 0.999158
factor(center)2    -4.2025     1.1891  -3.534 0.000409 ***
factor(center)3   -24.5874 21523.6453  -0.001 0.999089
factor(center)4    -0.9592     0.6548  -1.465 0.142956
factor(center)5    -2.0223     0.6700  -3.019 0.002540 **
---
    Null deviance: 73.07369  on 10  degrees of freedom
Residual deviance:  0.50214  on  4  degrees of freedom
Number of Fisher Scoring iterations: 21
-----------------------------------------------------------------------
```

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 (which can also be obtained by requesting rstandard(fit.cloglog, type="pearson")) 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`

The Firth penalized likelihood approach can also be done using the R package *logistf*. For example, for Table 6.11 analyzed above in the "infinite estimates" subsection:

```
> fit3 <- logistf(y/n ~ treatment + factor(center) -1, weights=n, family=binomial)
> summary(fit3)
logistf(formula = y/n ~ treatment + factor(center) - 1, weights = n,
    family = binomial)

Model fitted by Penalized ML
Confidence intervals and p-values by Profile Likelihood

                      coef  se(coef) lower 0.95 upper 0.95       Chisq
treatment        1.3678143 0.6436197  -3.125353  5.9101373  0.34269584
factor(center)1 -4.0036677 1.5193002  -8.998902 -1.6994870 17.89776479
factor(center)2 -3.6351503 1.0063781  -8.204822 -0.9953539 11.19907751
factor(center)3 -4.1707188 1.5811491  -9.187891 -1.6107831 14.20192563
factor(center)4 -0.8487087 0.6264638  -5.897048  4.2538020  0.03158963
factor(center)5 -1.8328467 0.6200202  -6.599538  2.9956561  0.00000000
                           p
treatment       5.582773e-01
factor(center)1 2.330947e-05
factor(center)2 8.183801e-04
```

```
factor(center)3 1.642024e-04
factor(center)4 8.589313e-01
factor(center)5 1.000000e+00

Likelihood ratio test=40.72184 on 6 df, p=3.28493e-07, n=10
Wald test = 26.05109 on 6 df, p = 0.000217816

Covariance-Matrix:
           [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
[1,]  0.4142463 -0.2747484 -0.3377549 -0.3456519 -0.2304370 -0.2758511
[2,] -0.2747484  2.3082730  0.2240156  0.2292533  0.1528371  0.1829579
[3,] -0.3377549  0.2240156  1.0127969  0.2818266  0.1878864  0.2249146
[4,] -0.3456519  0.2292533  0.2818266  2.5000323  0.1922793  0.2301733
[5,] -0.2304370  0.1528371  0.1878864  0.1922793  0.3924569  0.1534505
[6,] -0.2758511  0.1829579  0.2249146  0.2301733  0.1534505  0.3844251
```

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 multiple capabilities for modeling categorical data.

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.

This function can also fit the partial proportional odds model. Here is an example with cumulative logit link for the mental impairment data on p. 62 of the 2nd edition of my book, Analysis of Ordinal Categorical Data (with proportional odds for the life events effect):

```
> fit < -vglm(impair ~ ses + life, family=cumulative(parallel=FALSE~ses))

> summary(fit)
```

```
Coefficients:
              Estimate Std. Error  z value
(Intercept):1 -0.17660    0.69506 -0.25408
(Intercept):2  1.00567    0.66327  1.51623
(Intercept):3  2.39555    0.77894  3.07539
ses:1          0.98237    0.76430  1.28531
ses:2          1.54149    0.73732  2.09066
ses:3          0.73623    0.81213  0.90655
life          -0.32413    0.12017 -2.69736

Names of linear predictors: logit(P[Y<=1]), logit(P[Y<=2]), logit(P[Y<=3])

Residual deviance: 97.36467 on 113 degrees of freedom
Log-likelihood: -48.68234 on 113 degrees of freedom
```

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
```

The *profilelike.polr* function in the ProfileLikelihood library can provide profile likelihood confidence intervals for the proportional odds version of the cumulative logit model. See
`cran.r-project.org/web/packages/ProfileLikelihood/ProfileLikelihood.pdf`.

The *ordinal* package at CRAN can fit cumulative link models. See
`www.cran.r-project.org/package=ordinal/`. Apparently it can also fit models with random effects, using Gauss-Hermite quadrature or a Laplace approximation.

The *MNP* package can fit multinomial probit models using Bayesian methods. See
`imai.princeton.edu/research/files/MNPjss.pdf`

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
> A <- factor(a); C <- factor(c); M <- factor(m)
> indep <- glm(count ~ A + C + M, 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 ***
A2          -1.78511    0.05976 -29.872  < 2e-16 ***
C2          -0.64931    0.04415 -14.707  < 2e-16 ***
M2           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, .~. + A:C + A:M + C:M)
> 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 ***
A2          -5.52827    0.45221 -12.225  < 2e-16 ***
C2          -3.01575    0.15162 -19.891  < 2e-16 ***
M2          -0.52486    0.05428  -9.669  < 2e-16 ***
A2:C2        2.05453    0.17406  11.803  < 2e-16 ***
A2:M2        2.98601    0.46468   6.426 1.31e-10 ***
C2:M2        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

> library(car)
> Anova(homo.assoc) # likelihood-ratio tests for pairwise conditional associations
    LR Chisq Df Pr(>Chisq)
A:C   187.38  1  < 2.2e-16
A:M    91.64  1  < 2.2e-16
C:M   497.00  1  < 2.2e-16

> pearson.resid <- resid(homo.assoc, type="pearson") # Pearson residuals
> sum(pearson.resid^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
> std.resid <- rstandard(homo.assoc, type="pearson")
    # other way to get standardized residuals
> expected <- fitted(homo.assoc) # estimated expected frequencies
> cbind(count, expected, pearson.resid, std.resid)
  count  expected pearson.resid  std.resid
1   911 910.38317    0.02044342  0.6333249
2   538 538.61683   -0.02657821 -0.6333249
3    44  44.61683   -0.09234564 -0.6333249
4   456 455.38317    0.02890528  0.6333249
5     3   3.61683   -0.32434086 -0.6333250
6    43  42.38317    0.09474777  0.6333249
7     2   1.38317    0.52447888  0.6333250
8   279 279.61683   -0.03688791 -0.6333249
```

By the results of Sec. 9.5, we get the same results for the association between marijuana use and each of alcohol use and cigarette use if we treat the data as four binomials (instead of eight Poissons) and model the logit for marijuana use in terms

of additive effects for alcohol use and cigarette use.

```
------------------------------------------------------------------------
> drugs2 <- read.table("drugs_binomial.dat", header=TRUE)
> drugs2
    A    C  M_yes  M_no     n
1 yes  yes    911   538  1449
2 yes   no     44   456   500
3  no  yes      3    43    46
4  no   no      2   279   281
> attach(drugs2)
> alc <- factor(A); cig <- factor(C)
> fit.logistic <- glm(M_yes/n ~ alc + cig, weights=n,
                      family=binomial(link=logit))
> summary(fit.logistic)
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -5.3090     0.4752 -11.172  < 2e-16
alcyes        2.9860     0.4647   6.426 1.31e-10
cigyes        2.8479     0.1638  17.382  < 2e-16
---
    Null deviance: 843.82664  on 3  degrees of freedom
Residual deviance:   0.37399  on 1  degrees of freedom
------------------------------------------------------------------------
```

The *loglin* function in the MASS library can fit loglinear models using iterative proportional fitting, reporting parameter estimates using constraints whereby they sum to zero (rather than a baseline equaling 0). The *loglm* function allows the models to be specified and fitted in a manner similar to using *glm*.

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
--------------------------------------------------------------
```

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.

Greenacre and Nenadic have developed the *ca* package for simple, multiple, and joint correspondence analysis:

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

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

The *ACD* package can fit loglinear models when data are missing. See

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

Chapter 11: Models for Matched Pairs

Confidence interval for difference of proportions with matched pairs

For the score CI due to Tango (1998) and the adjusted Wald CI proposed by Agresti and Min (2005) that forms the ordinary Wald CI after adding 0.50 to each cell, see

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

The are also available in the *PropCIs* package.

```
-------------------------------------------------------------------------------
> library(PropCIs)
> scoreci.mp(54, 16, 433, 0.95); diffpropci.Wald.mp(54, 136, 433, 0.95)
95 percent confidence interval:
 -0.12651019 -0.05148492 95% score CI for difference of marginal probabilities
```

```
95 percent confidence interval:
 0.1295873 0.2491655 95% Wald CI for difference of marginal probabilities
> diffpropci.mp(54, 136, 433, 0.95)
95 percent confidence interval:
 0.1288091 0.2482024 95% adjusted CI for difference of marginal probabilities
```

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 at

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

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. The function *hmmm* at CRAN developed by R. Colombi, S. Giordano, M. Cazzaro, and J. Lang can fit hierarchical multinomial marginal models (Bergsma and Rudas 2002). The models can impose inequality constraints on the parameters. For details, see

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

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 75 quadrature points and a starting value of 9 for the estimate of σ.

```
> abortion <- read.table("abortion.dat",header=TRUE)
> 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$question==3,1,0)
> library(glmmML)
> fit.glmm <- glmmML(response ~ gender + z1 + z2,
+           cluster=abortion$case, family=binomial, data=abortion,
+           method = "ghq", n.points=70, start.sigma=9)
> summary(fit.glmm)
               coef se(coef)        z Pr(>|z|)
(Intercept) -0.61874   0.3777 -1.63839 1.01e-01
gender       0.01259   0.4888  0.02575 9.79e-01
z1           0.83470   0.1601  5.21347 1.85e-07
z2           0.29240   0.1567  1.86622 6.20e-02

Scale parameter in mixing distribution:  8.736 gaussian
Std. Error:                              0.5421
        LR p-value for H_0: sigma = 0:  0
```

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`

For example, for the horseshoe crab example in the text, you can use the code

```
> lda(y ~ width + color, data=Crabs)
```

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`