Package ‘vcdExtra’

January 25, 2021

Type Package
Title 'vcd' Extensions and Additions
Version 0.7-5
Date 2021-01-22
Author Michael Friendly [aut, cre],
      Heather Turner [ctb],
      Achim Zeileis [ctb],
      Duncan Murdoch [ctb],
      David Firth [ctb]
Maintainer Michael Friendly <friendly@yorku.ca>
Depends R (>= 2.10), vcd, gnm (>= 1.0-3), grid
Suggests gmodels, Fahrmeir, effects, VGAM, plyr, lmtest, nnet,
      ggplot2, Sleuth2, car, lattice, stats4, rgl, AER
Imports MASS, grDevices, stats, utils, ca
Description Provides additional data sets, methods and documentation to complement the 'vcd' pack-
      age for Visualizing Categorical Data
      and the 'gnm' package for Generalized Nonlinear Models.
      In particular, 'vcdExtra' extends mosaic, assoc and sieve plots from 'vcd' to han-
      dle 'glm()' and 'gnm()' models and
      adds a 3D version in 'mosaic3d'. Additionally, methods are provided for comparing and visualiz-
      ing lists of
      'glm' and 'loglm' objects. This package is now a support package for the book, `Dis-
      crete Data Analysis with R' by
      Michael Friendly and David Meyer.
License GPL (>= 2)
URL https://CRAN.R-project.org/package=vcdExtra
BugReports https://github.com/friendly/vcdExtra
LazyLoad yes
LazyData yes
NeedsCompilation no
Repository CRAN
Date/Publication 2021-01-25 17:00:02 UTC
R topics documented:

vcdExtra-package .................................................. 3
Abortion ............................................................... 5
Accident ............................................................... 6
AirCrash .............................................................. 8
Alligator ............................................................. 10
Bartlett .............................................................. 11
blogits ............................................................... 12
Burt ................................................................. 13
Caesar ................................................................. 14
Cancer ............................................................... 16
CMHtest ............................................................ 17
collapse.table ...................................................... 20
Cormorants ........................................................ 22
Crossings ........................................................... 24
cutfac ............................................................... 25
CyclingDeaths ...................................................... 27
datasets ............................................................. 28
DaytonSurvey ....................................................... 29
Depends ............................................................ 30
Detergent ........................................................... 31
Donner .............................................................. 32
Draft1970 .......................................................... 34
Draft1970table ...................................................... 35
Dyke ................................................................. 37
expand.dft .......................................................... 39
Fungicide .......................................................... 40
Geissler ............................................................. 41
Gilby ................................................................. 43
GKgamma ........................................................... 44
Glass ............................................................... 45
glmlist ............................................................. 46
GSS ................................................................. 48
HairEyePlace ....................................................... 49
Hauser79 ........................................................... 50
Heart ............................................................... 52
Heckman ............................................................ 53
HLtest ............................................................... 54
HospVisits .......................................................... 56
Hoyt ................................................................. 57
ICU ................................................................. 59
JobSat .............................................................. 61
Kway ............................................................... 62
logLik.loglm ....................................................... 64
loglin-utilities .................................................... 65
logseries .......................................................... 68
LRstats ............................................................ 70
vcdExtra-package

Index

<table>
<thead>
<tr>
<th>vcdExtra-package</th>
<th>Extensions and additions to vcd: Visualizing Categorical Data</th>
</tr>
</thead>
</table>

Description

This package provides additional data sets, documentation, and a few functions designed to extend the vcd package for Visualizing Categorical Data and the gnm package for Generalized Nonlinear Models. In particular, vcdExtra extends mosaic, assoc and sieve plots from vcd to handle glm() and gnm() models and adds a 3D version in mosaic3d.

The main purpose of this package is to serve as a sandbox for introducing extensions of mosaic plots and related graphical methods that apply to loglinear models fitted using glm() and related, generalized nonlinear models fitted with gnm() in the gnm-package package. A related purpose is to fill in some holes in the analysis of categorical data in R, not provided in base R, the vcd, or other commonly used packages.

The method mosaic.glm extends the mosaic.loglm method in the vcd package to this wider class of models. This method also works for the generalized nonlinear models fit with the gnm-package package, including models for square tables and models with multiplicative associations.

mosaic3d introduces a 3D generalization of mosaic displays using the rgl package.

In addition, there are several new data sets, a tutorial vignette,

vcd-tutorial  Working with categorical data with R and the vcd package, vignette("vcd-tutorial",package = "vcdExtra")

and a few functions for manipulating categorical data sets and working with models for categorical data.

A new class, glmlist, is introduced for working with collections of glm objects, e.g., Kway for fitting all K-way models from a basic marginal model, and LRstats for brief statistical summaries of goodness-of-fit for a collection of models.

For square tables with ordered factors, Crossings supplements the specification of terms in model formulas using Symm, Diag, Topo, etc. in the gnm-package.

Some of these extensions may be migrated into vcd or gnm.

A collection of demos is included to illustrate fitting and visualizing a wide variety of models:

mental-glm  Mental health data: mosaics for glm() and gnm() models

occStatus  Occupational status data: Compare mosaic using expected= to mosaic.glm

ucb-glm  UCBAAdmissions data: Conditional independence via loglm() and glm()

vision-quasi  VisualAcuity data: Quasi- and Symmetry models

yaish-unidiff  Yaish data: Unidiff model for 3-way table

Wong2-3  Political views and support for women to work (U, R, C, R+C and RC(1) models)

Wong3-1  Political views, support for women to work and national welfare spending (3-way, marginal, and conditional independence models)

housing  Visualize glm(), multinom() and polr() models from example(housing,package="MASS")
Use `demo(package="vcdExtra")` for a complete current list.

The `vcdExtra` package now contains a large number of data sets illustrating various forms of categorical data analysis and related visualizations, from simple to advanced. Use `data(package="vcdExtra")` for a complete list, or `datasets(package="vcdExtra")` for an annotated one showing the class and `dim` for each data set.

**Author(s)**

Michael Friendly
Maintainer: Michael Friendly <friendly AT yorku.ca>

**References**


**See Also**

`gnm-package`, for an extended range of models for contingency tables

`mosaic` for details on mosaic displays within the strucplot framework.

**Examples**

```r
example(mosaic.glm)
demo("mental-glm")
```

---

**Abortion**

*Abortion Opinion Data*

**Description**

Opinions about abortion classified by gender and SES

**Usage**

`data(Abortion)`
**Accident**

**Format**

A 3-dimensional array resulting from cross-tabulating 3 variables for 1100 observations. The variable names and their levels are:

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sex</td>
<td>&quot;Female&quot;, &quot;Male&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Status</td>
<td>&quot;Lo&quot;, &quot;Hi&quot;</td>
</tr>
<tr>
<td>3</td>
<td>Support_Abortion</td>
<td>&quot;Yes&quot;, &quot;No&quot;</td>
</tr>
</tbody>
</table>

**Details**

The combinations of Sex and Status represent four independent samples, having fixed Sex:Status marginal totals. Thus the Sex:Status association must be included in any loglinear model. Support_Abortion is a natural response variable.

**Source**


**Examples**

data(Abortion)

# example goes here
ftable(Abortion)
mosaic(Abortion, shade=TRUE)

# stratified by Sex
fourfold(aperm(Abortion, 3:1))
# stratified by Status
fourfold(aperm(Abortion, c(3,1,2)))

---

**Accident**

*Traffic Accident Victims in France in 1958*

**Description**

Bertin (1983) used these data to illustrate the cross-classification of data by numerous variables, each of which could have various types and could be assigned to various visual attributes.

For modeling and visualization purposes, the data can be treated as a 4-way table using loglinear models and mosaic displays, or as a frequency-weighted data frame using a binomial response for result ("Died" vs. "Injured") and plots of predicted probabilities.
Usage
data(Accident)

Format
A data frame in frequency form (comprising a 5 x 2 x 4 x 2 table) with 80 observations on the following 5 variables.

age  an ordered factor with levels 0-9 < 10-19 < 20-29 < 30-49 < 50+
result a factor with levels Died Injured
mode mode of transportation, a factor with levels 4-Wheeled Bicycle Motorcycle Pedestrian
gender a factor with levels Female Male
Freq a numeric vector

Details
age is an ordered factor, but arguably, mode should be treated as ordered, with levels Pedestrian < Bicycle < Motorcycle < 4-Wheeled as Bertin does. This affects the parameterization in models, so we don’t do this directly in the data frame.

Source
Bertin (1983), p. 30; original data from the Ministere des Travaux Publics

References

Examples
# examples
data(Accident)
head(Accident)

# for graphs, reorder mode
Accident$mode <- ordered(Accident$mode, 
   levels=levels(Accident$mode)[c(4,2,3,1)])

# Bertin's table
accident_tab <- xtabs(Freq ~ gender+mode+age+result, data=Accident)
structable(mode+gender ~ age+result, data=accident_tab)

## Loglinear models
## ----------------
# mutual independence
acc.mod0 <- glm(Freq ~ age+result+mode+gender, data=Accident, family=poisson)
LRstats(acc.mod0)
mosaic(acc.mod0, ~mode+age+gender+result)
# result as a response
acc.mod1 <- glm(Freq ~ age*mode*gender + result, data=Accident, family=poisson)
LRstats(acc.mod1)
mosaic(acc.mod1, ~mode+age+gender+result,
  labeling_args = list(abbreviate = c(gender=1, result=4)))

# allow two-way association of result with each explanatory variable
acc.mod2 <- glm(Freq ~ age*mode*gender + result*(age+mode+gender), data=Accident, family=poisson)
LRstats(acc.mod2)
mosaic(acc.mod2, ~mode+age+gender+result,
  labeling_args = list(abbreviate = c(gender=1, result=4)))

acc.mods <- glmlist(acc.mod0, acc.mod1, acc.mod2)
LRstats(acc.mods)

## Binomial (logistic regression) models for result
## -----------------------------------------------
library(car) # for Anova()
acc.bin1 <- glm(result=="Var Died Var" ~ age+mode+gender,
  weights=Freq, data=Accident, family=binomial)
Anova(acc.bin1)
acc.bin2 <- glm(result=="Var Died Var" ~ (age+mode+gender)^2,
  weights=Freq, data=Accident, family=binomial)
Anova(acc.bin2)
acc.bin3 <- glm(result=="Var Died Var" ~ (age+mode+gender)^3,
  weights=Freq, data=Accident, family=binomial)
Anova(acc.bin3)

# compare models
anova(acc.bin1, acc.bin2, acc.bin3, test="Chisq")

# visualize probability of death with effect plots
## Not run:
library(effects)
plot(allEffects(acc.bin1), ylab="Pr (Died)")
plot(allEffects(acc.bin2), ylab="Pr (Died)")
## End(Not run)

---

**Description**

Data on all fatal commercial airplane crashes from 1993–2015. Excludes small planes (less than 6 passengers) and non-commercial (cargo, military, private) aircraft.
Usage

```r
data("AirCrash")
```

Format

A data frame with 439 observations on the following 5 variables.

- **Phase**: phase of the flight, a factor with levels:
  - en route
  - landing
  - standing
  - take-off
  - unknown

- **Cause**: a factor with levels:
  - criminal
  - human error
  - mechanical
  - unknown
  - weather

- **date**: date of crash, a Date

- **Fatalities**: number of fatalities, a numeric vector

- **Year**: year, a numeric vector

Details

Phase of the flight was cleaned by combining related variants, spelling, etc.

Source

Originally from David McCandless, [http://www.informationisbeautiful.net/visualizations/](http://www.informationisbeautiful.net/visualizations/)
plane-truth-every-single-commercial-plane-crash-visualized/, with the data at [https://docs.google.com/spreadsheets/d/1OvDq4_BtbR6nSnnHnjD5hVC3HQ-ulZPGbo0RDGbzM3Q/edit?usp=drive_web](https://docs.google.com/spreadsheets/d/1OvDq4_BtbR6nSnnHnjD5hVC3HQ-ulZPGbo0RDGbzM3Q/edit?usp=drive_web), downloaded April 14, 2015.

References

Rick Wicklin, [http://blogs.sas.com/content/iml/2015/03/30/visualizing-airline-crashes.html](http://blogs.sas.com/content/iml/2015/03/30/visualizing-airline-crashes.html)

Examples

```r
data(AirCrash)
aircrash.tab <- xtabs(~Phase + Cause, data=AirCrash)
mosaic(aircrash.tab, shade=TRUE)

# fix label overlap
mosaic(aircrash.tab, shade=TRUE,
      labeling_args=list(rot_labels=c(30, 30, 30, 30)))

# reorder by Phase
phase.ord <- rev(c(3,4,1,2,5))
mosaic(aircrash.tab[phase.ord,,], shade=TRUE,
       labeling_args=list(rot_labels=c(30, 30, 30, 30)), offset_varnames=0.5)

# reorder by frequency
phase.ord <- order(rowSums(aircrash.tab), decreasing=TRUE)
cause.ord <- order(colSums(aircrash.tab), decreasing=TRUE)
mosaic(aircrash.tab[phase.ord,cause.ord], shade=TRUE,
        labeling_args=list(rot_labels=c(30, 30, 30, 30)))
```
library(ca)
aircrash.ca <- ca(aircrash.tab)
plot(aircrash.ca)

---

### Alligator

#### Alligator Food Choice

**Description**

The Alligator data, from Agresti (2002), comes from a study of the primary food choices of alligators in four Florida lakes. Researchers classified the stomach contents of 219 captured alligators into five categories: Fish (the most common primary food choice), Invertebrate (snails, insects, crayfish, etc.), Reptile (turtles, alligators), Bird, and Other (amphibians, plants, household pets, stones, and other debris).

**Usage**

data(Alligator)

**Format**

A frequency data frame with 80 observations on the following 5 variables.

- **lake**  a factor with levels George Hancock Oklawaha Trafford
- **sex**  a factor with levels female male
- **size**  alligator size, a factor with levels large (>2.3m) small (<=2.3m)
- **food**  primary food choice, a factor with levels bird fish invert other reptile
- **count**  cell frequency, a numeric vector

**Details**

The table contains a fair number of 0 counts.

food is the response variable. fish is the most frequent choice, and often taken as a baseline category in multinomial response models.

**Source**

data(Alligator)

# change from frequency data.frame to table
allitable <- xtabs(count~lake+sex+size+food, data=Alligator)
# Agresti's Table 7.1
structable(food~lake+sex+size, allitable)

plot(allitable, shade=TRUE)
# mutual independence model
mosaic(~food+lake+size, allitable, shade=TRUE)
# food jointly independent of lake and size
mosaic(~food+lake+size, allitable, shade=TRUE, expected=~lake:size+food)

if (require(nnet)) {
  # multinomial logit model
  mod1 <- multinom(food ~ lake+size+sex, data=Alligator, weights=count)
}

Bartlett data on plum root cuttings

In an experiment to investigate the effect of cutting length (two levels) and planting time (two levels) on the survival of plum root cuttings, 240 cuttings were planted for each of the 2 x 2 combinations of these factors, and their survival was later recorded.

Bartlett (1935) used these data to illustrate a method for testing for no three-way interaction in a contingency table.

Usage
data(Bartlett)

Format
A 3-dimensional array resulting from cross-tabulating 3 variables for 960 observations. The variable names and their levels are:

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alive</td>
<td>&quot;Alive&quot;,&quot;Dead&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Time</td>
<td>&quot;Now&quot;,&quot;Spring&quot;</td>
</tr>
<tr>
<td>3</td>
<td>Length</td>
<td>&quot;Long&quot;,&quot;Short&quot;</td>
</tr>
</tbody>
</table>
Source


References


Examples

data(Bartlett)

fourfold(Bartlett, mfrow=c(1,2))

mosaic(Bartlett, shade=TRUE)
pairs(Bartlett, gp=shading_Friendly)

---

**blogits**

*Bivariate Logits and Log Odds Ratio*

**Description**

This function calculates the log odds and log odds ratio for two binary responses classified by one or more stratifying variables.

It is useful for plotting the results of bivariate logistic regression models, such as those fit using `vglm` in the *VGAM*.

**Usage**

```r
blogits(Y, add, colnames, row.vars, rev=FALSE)
```

**Arguments**

- **Y**: A four-column matrix or data frame whose columns correspond to the 2 x 2 combinations of two binary responses.
- **add**: Constant added to all cells to allow for zero frequencies. The default is 0.5 if any(Y)==0 and 0 otherwise.
- **colnames**: Names for the columns of the results. The default is c("logit1","logit2","logOR"). If less than three names are supplied, the remaining ones are filled in from the default.
- **row.vars**: A data frame or matrix giving the factor levels of one or more factors corresponding to the rows of Y.
- **rev**: A logical, indicating whether the order of the columns in Y should be reversed.
Details

For two binary variables with levels 0,1 the logits are calculated assuming the columns in \( Y \) are given in the order 11, 10, 01, 00, so the logits give the log odds of the 1 response compared to 0. If this is not the case, either use \( \text{rev=TRUE} \) or supply \( Y[,4:1] \) as the first argument.

Value

A data frame with \( \text{nrow}(Y) \) rows and \( 3 + \text{ncol}(\text{row.vars}) \) columns

Author(s)

Michael Friendly

References


See Also

`vglm`

Examples

data(Toxaemia)

```r
tox.tab <- xtabs(Freq~class + smoke + hyper + urea, Toxaemia)

# reshape to 4-column matrix
toxaemia <- t(matrix(aperm(tox.tab), 4, 15))

colnames(toxaemia) <- c("hu", "hU", "Hu", "HU")

rowlabs <- expand.grid(smoke=c("0", "1-19", "20+"), class=factor(1:5))
toxaemia <- cbind(toxaemia, rowlabs)

# logits for H and U

logitsTox <- blogits(toxaemia[,4:1], add=0.5, colnames=c("logitH", "logitW"), row.vars=rowlabs)

logitsTox
```

---

**Burt**

*Burt (1950) Data on Hair, Eyes, Head and Stature*

---

Description

Cyril Burt (1950) gave these data, on a sample of 100 people from Liverpool, to illustrate the application of a method of factor analysis (later called multiple correspondence analysis) applied to categorical data.

He presented these data initially in the form that has come to be called a "Burt table", giving the univariate and bivariate frequencies for an n-way frequency table.
Usage

data("Burt")

Format

A frequency data frame (representing a 3 x 3 x 2 x 2 frequency table) with 36 observations on the following 5 variables.

Hair  hair color, a factor with levels Fair Red Dark
Eyes  eye color, a factor with levels Light Mixed Dark
Head  head shape, a factor with levels Narrow Wide
Stature  height, a factor with levels Tall Short
Freq  a numeric vector

Details

Burt says: "In all, 217 individuals were examined, about two-thirds of them males. But, partly to simplify the calculations and partly because the later observations were rather more trustworthy, I shall here restrict my analysis to the data obtained from the last hundred males in the series."

Head and Stature reflect a binary coding where people are classified according to whether they are below or above the average for the population.

Source


Examples

data(Burt)
mosaic(Freq ~ Hair + Eyes + Head + Stature, data=Burt, shade=TRUE)

# or
burt.tab <- xtabs(Freq ~ Hair + Eyes + Head + Stature, data=Burt)
mosaic(burt.tab, shade=TRUE)

---

Caesar

Risk Factors for Infection in Caesarian Births

Description

Data from infection from birth by Caesarian section, classified by Risk (two levels), whether Antibiotics were used (two levels) and whether the Caesarian section was Planned or not. The outcome is Infection (three levels).
Usage

data(Caesar)

Format

A 4-dimensional array resulting from cross-tabulating 4 variables for 251 observations. The variable names and their levels are:

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Infection</td>
<td>&quot;Type 1&quot;, &quot;Type 2&quot;, &quot;None&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Risk</td>
<td>&quot;Yes&quot;, &quot;No&quot; (presence of risk factors)</td>
</tr>
<tr>
<td>3</td>
<td>Antibiotics</td>
<td>&quot;Yes&quot;, &quot;No&quot; (were antibiotics given?)</td>
</tr>
<tr>
<td>4</td>
<td>Planned</td>
<td>&quot;Yes&quot;, &quot;No&quot; (was the C section planned?)</td>
</tr>
</tbody>
</table>

Details

Infection is regarded as the response variable here. There are quite a few 0 cells here, particularly when Risk is absent and the Caesarian section was unplanned. Should these be treated as structural or sampling zeros?

Source


See Also

cesar for the same data recorded as a frequency data frame with other variables.

Examples

data(Caesar)
#display table; note that there are quite a few 0 cells
structable(Caesar)
require(MASS)

# baseline model, Infection as response
Caesar.mod0 <- loglm(~Infection + (Risk*Antibiotics*Planned), data=Caesar)
# NB: Pearson chisq cannot be computed due to the 0 cells
Caesar.mod0

mosaic(Caesar.mod0, main="Baseline model")

# Illustrate handling structural zeros
zeros <- 0+ (Caesar >0)
zeros[1,,1,1] <- 1
structable(zeros)

# fit model excluding possible structural zeros
Caesar.mod0s <- loglm(~Infection + (Risk*Antibiotics*Planned), data=Caesar,
start=zeros)
Caesar.mod0s

anova(Caesar.mod0, Caesar.mod0s, test="Chisq")
mosaic(Caesar.mod0s)

# what terms to add?
add1(Caesar.mod0, ~.^2, test="Chisq")

# add Association of Infection:Antibiotics
Caesar.mod1 <- update(Caesar.mod0, ~.+Infection:Antibiotics)
anova(Caesar.mod0, Caesar.mod1, test="Chisq")
mosaic(Caesar.mod1, gp=shading_Friendly, main="Adding Infection:Antibiotics")

---

### Cancer

#### Survival of Breast Cancer Patients

<table>
<thead>
<tr>
<th>Cancer</th>
<th>Survival of Breast Cancer Patients</th>
</tr>
</thead>
</table>

**Description**

Three year survival of 474 breast cancer patients according to nuclear grade and diagnostic center.

**Usage**

```r
data(Cancer)
```

**Format**

A 3-dimensional array resulting from cross-tabulating 3 variables for 474 observations. The variable names and their levels are:

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Survival</td>
<td>&quot;Died&quot;, &quot;Surv&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Grade</td>
<td>&quot;Malignant&quot;, &quot;Benign&quot;</td>
</tr>
<tr>
<td>3</td>
<td>Center</td>
<td>&quot;Boston&quot;, &quot;Glamorgan&quot;</td>
</tr>
</tbody>
</table>

**Source**


Examples

data(Cancer)

# example goes here

---

**CMHtest**  
*Generalized Cochran-Mantel-Haenszel Tests*

**Description**

Provides generalized Cochran-Mantel-Haenszel tests of association of two possibly ordered factors, optionally stratified other factor(s). With strata, CMHtest calculates these tests for each level of the stratifying variables and also provides overall tests controlling for the strata.

For ordinal factors, more powerful tests than the test for general association (independence) are obtained by assigning scores to the row and column categories.

**Usage**

```r
CMHtest(x, ...)
```

## S3 method for class 'formula'

```r
CMHtest(formula, data = NULL, subset = NULL, na.action = NULL, ...)
```

## Default S3 method:

```r
CMHtest(x, strata = NULL, 
        rscores = 1:R, cscores = 1:C, 
        types = c("cor", "rmeans", "cmeans", "general"), 
        overall=FALSE, details=overall, ...)
```

## S3 method for class 'CMHtest'

```r
print(x, digits = max(getOption("digits") - 2, 3), ...)
```

**Arguments**

- `x`  
  A 2+ way contingency table in array form, or a class "table" object with optional category labels specified in the dimnames(x) attribute.

- `formula`  
  a formula specifying the variables used to create a contingency table from data. This should be a one-sided formula when data is in array form, and a two-sided formula with a response Freq if data is a data frame with a cell frequency variable. For convenience, conditioning formulas can be specified indicating strata.

- `data`  
  either a data frame, or an object of class "table" or "ftable".

- `subset`  
  an optional vector specifying a subset of observations to be used.

- `na.action`  
  a function which indicates what should happen when the data contain NAs. Ignored if data is a contingency table.
strata  For a 3- or higher-way table, the names or numbers of the factors to be treated as strata. By default, the first 2 factors are treated as the main table variables, and all others considered stratifying factors.

rscores  Row scores. Either a set of numbers (typically integers, 1:R) or the string "midrank" for standardized midrank scores, or NULL to exclude tests that depend on row scores.

cscores  Column scores. Same as for row scores.

types  Types of CMH tests to compute: Any one or more of c("cor", "cmeans", "rmeans", "general"), or "ALL" for all of these.

overall  logical. Whether to calculate overall tests, controlling for the stratifying factors.

details  logical. Whether to include computational details in the result

...  Other arguments passed to default method.

digits  Digits to print.

Details

The standard $\chi^2$ tests for association in a two-way table treat both table factors as nominal (unordered) categories. When one or both factors of a two-way table are quantitative or ordinal, more powerful tests of association may be obtained by taking ordinality into account using row and or column scores to test for linear trends or differences in row or column means.

The CMH analysis for a two-way table produces generalized Cochran-Mantel-Haenszel statistics (Landis et al., 1978).

These include the CMH correlation statistic ("cor"), treating both factors as ordered. For a given stratum, with equally spaced row and column scores, this CMH statistic reduces to $(n-1)r^2$, where $r$ is the Pearson correlation between X and Y. With "midrank" scores, this CMH statistic is analogous to $(n-1)r^2_s$, using the Spearman rank correlation.

The ANOVA (row mean scores and column mean scores) statistics, treat the columns and rows respectively as ordinal, and are sensitive to mean shifts over columns or rows. These are transforms of the $F$ statistics from one-way ANOVAs with equally spaced scores and to Kruskal-Wallis tests with "midrank" scores.

The CMH general association statistic treat both factors as unordered, and give a test closely related to the Pearson $\chi^2$ test. When there is more than one stratum, the overall general CMH statistic gives a stratum-adjusted Pearson $\chi^2$, equivalent to what is calculated by `mantelhaen.test`.

For a 3+ way table, one table of CMH tests is produced for each combination of the factors identified as strata. If overall=TRUE, an additional table is calculated for the same two primary variables, controlling for (pooling over) the strata variables.

These overall tests implicitly assume no interactions between the primary variables and the strata and they will have low power in the presence of interactions.

Value

An object of class "CMHtest", a list with the following 4 components:

table  A matrix containing the test statistics, with columns Chisq, Df and Prob

names  The names of the table row and column variables
**CMHtest**

rscore  Row scores  
cscore  Column scores  

If details==TRUE, additional components are included.  
If there are strata, the result is a list of "CMHtest" objects. If overall=TRUE another component, labeled ALL is appended to the list.

**Author(s)**

Michael Friendly

**References**


**See Also**

- `cmh_test` provides the CMH test of general association;  
- `lbl_test` provides the CMH correlation test of linear by linear association.  
- `mantelhaen.test` provides the overall general Cochran-Mantel-Haenszel chi-squared test of the null that two nominal variables are conditionally independent in each stratum, assuming that there is no three-way interaction.

**Examples**

data(JobSat, package="vcdExtra")  
CMHtest(JobSat)  
CMHtest(JobSat, rscores="midrank", cscores="midrank")

# formula interface  
CMHtest(~ ., data=JobSat)

# A 3-way table (both factors ordinal)  
data(MSPatients, package="vcd")  
CMHtest(MSPatients)

# also calculate overall tests, controlling for Patient  
CMHtest(MSPatients, overall=TRUE)
# compare with mantelhaen.test
mantelhaen.test(MSPatients)

# formula interface
CMHtest(~ ., data=MSPatients, overall=TRUE)

# using a frequency data.frame
CMHtest(xtabs(Freq~ses*mental, data=Mental))
# or, more simply
CMHtest(Freq~ses+mental, data=Mental)

# conditioning formulae
CMHtest(Freq~right+left|gender, data=VisualAcuity)

CMHtest(Freq ~ attitude+memory|education+age, data=Punishment)

# Stokes et al, Table 5.1, p 92: two unordered factors
parties <- matrix(
c(221, 160, 360, 140,
  200, 291, 160, 311,
  208, 106, 316, 97), nrow=3, ncol=4, byrow=TRUE)
dimnames(parties) <- list(party=c("Dem", "Indep", "Rep"),
  neighborhood=c("Bayside", "Highland", "Longview", "Sheffield"))
CMHtest(parties, rscores=NULL, cscores=NULL)
# compare with Pearson chisquare
chisq.test(parties)

---

**collapse.table**  
**Collapse Levels of a Table**

**Description**
Collapse (or re-label) variables in a contingency table, array or ftable object by re-assigning levels of the table variables.

**Usage**
collapse.table(table, ...)

**Arguments**
- **table** A table, array or ftable object
- **...** A collection of one or more assignments of factors of the table to a list of levels

**Details**
Each of the ... arguments must be of the form variable = levels, where variable is the name of one of the table dimensions, and levels is a character or numeric vector of length equal to the corresponding dimension of the table.
Value

A `xtabs` and `table` object, representing the original table with one or more of its factors collapsed or rearranged into other levels.

Author(s)

Michael Friendly

See Also

`expand.dft` expands a frequency data frame to case form.
`margin.table` "collapses" a table in a different way, by summing over table dimensions.

Examples

```r
# create some sample data in table form
sex <- c("Male", "Female")
age <- letters[1:6]
education <- c("low", "med", "high")
data <- expand.grid(sex=sex, age=age, education=education)
counts <- rpois(36, 100)
data <- cbind(data, counts)
t1 <- xtabs(counts ~ sex + age + education, data=data)
structable(t1)

## age a b c d e f
## sex education
## Male low 119 101 109 85 99 93
## med 94 98 103 108 84 84
## high 81 88 96 110 100 92
## Female low 107 104 95 86 103 96
## med 104 98 94 95 110 106
## high 93 85 90 109 99 86

# collapse age to 3 levels
t2 <- collapse.table(t1, age=c("A", "A", "B", "B", "C", "C"))
structable(t2)

## age A B C
## sex education
## Male low 220 194 192
## med 192 211 168
## high 169 206 192
## Female low 211 181 199
## med 202 189 216
## high 178 199 185

# collapse age to 3 levels and pool education: "low" and "med" to "low"
t3 <- collapse.table(t1, age=c("A", "A", "B", "B", "C", "C"),
```

data("Cormorants")

Cormorants

Advertising Behavior by Males Cormorants

Description

Male double-crested cormorants use advertising behavior to attract females for breeding. In this study by Meagan Mc Rae (2015), cormorants were observed two or three times a week at six stations in a tree-nesting colony for an entire season, April 10, 2014-July 10, 2014. The number of advertising birds was counted and these observations were classified by characteristics of the trees and nests.

The goal is to determine how this behavior varies temporally over the season and spatially, as well as with characteristics of nesting sites.

Usage

data("Cormorants")
Cormorants

Format

A data frame with 343 observations on the following 8 variables.

- **category**  Time of season, divided into 3 categories based on breeding chronology, an ordered factor with levels Pre < Incubation < Chicks Present
- **week**  Week of the season
- **station**  Station of observations on two different peninsulas in a park, a factor with levels B1 B2 C1 C2 C3 C4
- **nest**  Type of nest, an ordered factor with levels no < partial < full
- **height**  Relative height of bird in the tree, an ordered factor with levels low < mid < high
- **density**  Number of other nests in the tree, an ordered factor with levels zero < few < moderate < high
- **tree_health**  Health of the tree the bird is advertising in, a factor with levels dead healthy
- **count**  Number of birds advertising, a numeric vector

Details

Observations were made on only 2 days in weeks 3 and 4, but 3 days in all other weeks. One should use log(days) as an offset, so that the response measures rate.

\[
\text{Cormorants$days <- ifelse(Cormorants$week %in% 3:4, 2, 3)}
\]

Source


Examples

data(Cormorants)
str(Cormorants)

if (require("ggplot2")) {
  print(ggplot(Cormorants, aes(count)) + geom_histogram(binwidth=0.5) +
         labs(x="Number of birds advertising"))

  # Quick look at the data, on the log scale, for plots of `count ~ week`,
  # stratified by something else.

  print(ggplot(Cormorants, aes(week, count, color=height)) + geom_jitter() +
       stat_smooth(method="loess", size=2) + scale_y_log10(breaks=c(1,2,5,10)) +
       geom_vline(xintercept=c(4.5, 9.5)))
}

# ### models using week
fit1 <- glm(count ~ week + station + nest + height + density + tree_health, data=Cormorants, family = poisson)
if (requireNamespace("car"))
car::Anova(fit1)

# plot fitted effects
if (requireNamespace("effects"))
  plot(effects::allEffects(fit1))

---

Crossings

Crossings Interaction of Factors

Description

Given two ordered factors in a square, n x n frequency table, Crossings creates an n-1 column matrix corresponding to different degrees of difficulty in crossing from one level to the next, as described by Goodman (1972).

Usage

Crossings(...)

Arguments

... Two factors

Value

For two factors of n levels, returns a binary indicator matrix of n*n rows and n-1 columns.

Author(s)

Michael Friendly and Heather Turner

References


See Also

glm, gnm for model fitting functions for frequency tables
Diag, Mult, Symm, Topo for similar extensions to terms in model formulas.
Examples

data(Hauser79)
# display table
strtable(~Father+Son, data=Hauser79)

hauser.indep <- gnm(Freq ~ Father + Son, data=Hauser79, family=poisson)
hauser.CR <- update(hauser.indep, ~ . + Crossings(Father,Son))
LRstats(hauser.CR)

hauser.CRdiag <- update(hauser.indep, ~ . + Crossings(Father,Son) + Diag(Father,Son))
LRstats(hauser.CRdiag)

cutfac

Cut a Numeric Variable to a Factor

Description

cutfac acts like cut, dividing the range of x into intervals and coding the values in x according in
which interval they fall. However, it gives nicer labels for the factor levels and by default chooses
convenient breaks among the values based on deciles.

It is particularly useful for plots in which one wants to make a numeric variable discrete for the
purpose of getting boxplots, spinograms or mosaic plots.

Usage

cutfac(x, breaks = NULL, q = 10)

Arguments

x a numeric vector which is to be converted to a factor by cutting
breaks either a numeric vector of two or more unique cut points or a single number
(greater than or equal to 2) giving the number of intervals into which x is to be
cut.
q the number of quantile groups used to define breaks, if that has not been speci-
ified.

Details

By default, cut chooses breaks by equal lengths of the range of x, whereas cutfac uses quantile
to choose breaks of roughly equal count.

Value

A factor corresponding to x is returned
Author(s)

Achim Zeileis

References


See Also

cut, quantile

Examples

```r
if (require(AER)) {
  data("NMES1988", package="AER")
  nmes <- NMES1988[, c(1, 6:8, 13, 15, 18)]

  plot(log(visits+1) ~ cutfac(chronic), data = nmes,
       ylab = "Physician office visits (log scale)",
       xlab = "Number of chronic conditions", main = "chronic")

  plot(log(visits+1) ~ cutfac(hospital, c(0:2, 8)), data = nmes,
       ylab = "Physician office visits (log scale)",
       xlab = "Number of hospital stays", main = "hospital")
}
```
CyclingDeaths

Description

A data frame containing the number of deaths of cyclists in London from 2005 through 2012 in each fortnightly period. Aberdein & Spiegelhalter (2013) discuss these data in relation to the observation that six cyclists died in London between Nov. 5 and Nov. 13, 2013.

Usage

data(CyclingDeaths)

Format

A data frame with 208 observations on the following 2 variables.

date a Date
deaths number of deaths, a numeric vector

Source


References


Examples

data(CyclingDeaths)

plot(deaths ~ date, data=CyclingDeaths, type="h", lwd=3, ylab="Number of deaths", axes=FALSE)
axis(1, at=seq(as.Date("2005-01-01"), by='years', length.out=9), labels=2005:2013)
axis(2, at=0:3)

# make a one-way frequency table
CyclingDeaths.tab <- table(CyclingDeaths$deaths)
gf <- goodfit(CyclingDeaths.tab)
gf
summary(gf)

rootogram(gf, xlab="Number of Deaths")
distplot(CyclingDeaths.tab)
# prob of 6 or more deaths in one fortnight
lambda <- gf$par$lambda
ppois(5, lambda, lower.tail=FALSE)

Description

The `data` function is used both to load data sets from packages, and give a display of the names and titles of data sets in one or more packages, however it does not return a result that can be easily used to get additional information about the nature of data sets in packages.

The `datasets()` function is designed to produce a more useful summary display of data sets in one or more packages. It extracts the class and dimension information (dim or codeLength) of each item, and formats these to provide additional descriptors.

Usage

```r
datasets(package, allClass=FALSE,
          incPackage=length(package) > 1,
          maxTitle=NULL)
```

Arguments

- `package` a character vector giving the package(s) to look in
- `allClass` include all classes of the item (TRUE) or just the last class (FALSE)?
- `incPackage` include the package name in result?
- `maxTitle` maximum length of data set Title

Details

The requested packages must be installed, and are silently loaded in order to extract class and size information.

Value

A `data.frame` whose rows correspond to data sets found in package. The columns (for a single package) are:

- `Item` data set name
- `class` class
- `dim` an abbreviation of the dimensions of the data set
- `Title` data set title

Author(s)

Michael Friendly, with R-help from Curt Seeliger
### Description
This data, from Agresti (2002), Table 9.1, gives the result of a 1992 survey in Dayton Ohio of 2276 high school seniors on whether they had ever used alcohol, cigarettes and marijuana.

### Usage
```r
data(DaytonSurvey)
```

### Format
A frequency data frame with 32 observations on the following 6 variables.

- **cigarette**: a factor with levels Yes No
- **alcohol**: a factor with levels Yes No
- **marijuana**: a factor with levels Yes No
- **sex**: a factor with levels female male
- **race**: a factor with levels white other
- **Freq**: a numeric vector

### Details
Agresti uses the letters G (sex), R (race), A (alcohol), C (cigarette), M (marijuana) to refer to the table variables, and this usage is followed in the examples below.

Background variables include sex and race of the respondent (GR), typically treated as explanatory, so that any model for the full table should include the term sex:race. Models for the reduced table, collapsed over sex and race are not entirely unreasonable, but don’t permit the estimation of the effects of these variables on the responses.

The full 5-way table contains a number of cells with counts of 0 or 1, as well as many cells with large counts, and even the ACM table collapsed over GR has some small cell counts. Consequently, residuals for these models in mosaic displays are best represented as standardized (adjusted) residuals.
Source


References


Examples

data(DaytonSurvey)

mod.GR <- glm(Freq ~ . + sex*race, data=DaytonSurvey, family=poisson) # mutual independence + GR
mod.homog.assoc <- glm(Freq ~ .^2, data=DaytonSurvey, family=poisson) # homogeneous association

# collapse over sex and race
Dayton.ACM <- aggregate(Freq ~ cigarette+alcohol+marijuana, data=DaytonSurvey, FUN=sum)

---

**Description**

This one-way table gives the type-token distribution of the number of dependencies declared in 4983 packages listed on CRAN on January 17, 2014.

**Usage**

data(Depends)

**Format**

The format is: 'table' int [1:15(1d)] 986 1347 993 685 375 298 155 65 32 19 ... - attr(*, "dim-names")=List of 1 ..$ Depends: chr [1:15] "0" "1" "2" "3" ...

**Source**

Examples

data(Depends)
plot(Depends, xlab="Number of Dependencies", ylab="Number of R Packages", lwd=8)

## Not run:
# The code below, from Joseph Rickert, downloads and tabulates the data
p <- as.data.frame(available.packages(), stringsAsFactors=FALSE)
names(p)

pkgs <- data.frame(p[,c(1,4)])  # Pick out Package names and Depends
row.names(pkgs) <- NULL  # Get rid of row names
pkgs <- pkgs[complete.cases(pkgs[,2]),]  # Remove NAs

pkgs$Depends2 <- strsplit(pkgs$Depends, ",")  # split list of Depends
pkgs$numDepends <- as.numeric(lapply(pkgs$Depends2, length))  # Count number of dependencies in list
zeros <- c(rep(0, dim(p)[1] - dim(pkgs)[1]))  # Account for packages with no dependencies
Depends <- as.vector(c(zeros, pkgs$numDepends))  # Set up to tablate

## End(Not run)

---

Detergent

**Detergent preference data**

Description

Cross-classification of a sample of 1008 consumers according to (a) the softness of the laundry water used, (b) previous use of detergent Brand M, (c) the temperature of laundry water used and (d) expressed preference for Brand X or Brand M in a blind trial.

Usage

data(Detergent)

Format

A 4-dimensional array resulting from cross-tabulating 4 variables for 1008 observations. The variable names and their levels are:

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Temperature</td>
<td>&quot;High&quot;, &quot;Low&quot;</td>
</tr>
<tr>
<td>2</td>
<td>M_User</td>
<td>&quot;Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td>3</td>
<td>Preference</td>
<td>&quot;Brand X&quot;, &quot;Brand M&quot;</td>
</tr>
<tr>
<td>4</td>
<td>Water_softness</td>
<td>&quot;Soft&quot;, &quot;Medium&quot;, &quot;Hard&quot;</td>
</tr>
</tbody>
</table>
Source

References

Examples
data(Detergent)

# example goes here
mosaic(Detergent, shade=TRUE)

require(MASS)
(det.mod0 <- loglm(~ Preference + Temperature + M_User + Water_softness, data=Detergent))
# examine addition of two-way terms
add1(det.mod0, ~ .^2, test="Chisq")

# model for Preference as a response
(det.mod1 <- loglm(~ Preference + (Temperature * M_User * Water_softness), data=Detergent))
mosaic(det.mod0)

---

Donner Survival in the Donner Party

Description
This data frame contains information on the members of the Donner Party, a group of people who attempted to migrate to California in 1846. They were trapped by an early blizzard on the eastern side of the Sierra Nevada mountains, and before they could be rescued, nearly half of the party had died.

What factors affected who lived and who died?

Usage
data(Donner)

Format
A data frame with 90 observations on the following 5 variables.

family family name, a factor with 10 levels
age age of person, a numeric vector
sex  a factor with levels Female Male
survived a numeric vector, 0 or 1
death  date of death for those who died before rescue, a POSIXct

Details
This data frame uses the person’s name as row labels. family reflects a recoding of the last names of individuals to reduce the number of factor levels. The main families in the Donner party were: Donner, Graves, Breen and Reed. The families of Murphy, Foster and Pike are grouped as 'MurFosPik', those of Fosdick and Wolfinger are coded as 'FosdWolf', and all others as 'Other'.

Source


References


See Also
donner in alr3, case2001 in Sleuth2 (adults only) provide similar data sets.

Examples
# conditional density plots
op <- par(mfrow=c(1,2), cex.lab=1.5)
cdplot(factor(survived) ~ age, subset=sex=='Male', data=Donner, main="Donner party: Males", ylevels=2:1, ylab="Survived", yaxlabels=c("yes", "no"))
with(Donner, rug(jitter(age[sex=='Male']), col="white", quiet=TRUE))
cdplot(factor(survived) ~ age, subset=sex=='Female', data=Donner, main="Donner party: Females", ylevels=2:1, ylab="Survived", yaxlabels=c("yes", "no"))
with(Donner, rug(jitter(age[sex=='Female']), col="white", quiet=TRUE))
par(op)

# fit some models
(mod1 <- glm(survived ~ age + sex, data=Donner, family=binomial))
(mod2 <- glm(survived ~ age * sex, data=Donner, family=binomial))
anova(mod2, test="Chisq")
(mod3 <- glm(survived ~ poly(age,2) * sex, data=Donner, family=binomial))
anova(mod3, test="Chisq")
LRstats(glmlist(mod1, mod2, mod3))

# plot fitted probabilities from mod2 and mod3
# idea from: http://www.ling.upenn.edu/~joseff/rstudy/summer2010_ggplot2_intro.html
library(ggplot2)

# separate linear fits on age for M/F
ggplot(Donner, aes(age, survived, color = sex)) +
  geom_point(position = position_jitter(height = 0.02, width = 0)) +
  stat_smooth(method = "glm", method.args = list(family = binomial), formula = y ~ x,
              alpha = 0.2, size=2, aes(fill = sex))

# separate quadratics
ggplot(Donner, aes(age, survived, color = sex)) +
  geom_point(position = position_jitter(height = 0.02, width = 0)) +
  stat_smooth(method = "glm", method.args = list(family = binomial), formula = y ~ poly(x,2),
              alpha = 0.2, size=2, aes(fill = sex))
of draft age (born 1944 to 1950) who shared a birthdate would be called to serve at once. The first 195 birthdates drawn were later called to serve in the order they were drawn; the last of these was September 24.

Source


References


See Also

*Draft1970table*

Examples

data(Draft1970)
# scatterplot
plot(Rank ~ Day, data=Draft1970)
with(Draft1970, lines(lowess(Day, Rank), col="red", lwd=2))
abline(lm(Rank ~ Day, data=Draft1970), col="blue")

# boxplots
plot(Rank ~ Month, data=Draft1970, col="bisque")

lm(Rank ~ Month, data=Draft1970)
anova(lm(Rank ~ Month, data=Draft1970))

# make the table version
with(Draft1970, table(Month, Risk))

---

*Draft1970table*  
**USA 1970 Draft Lottery Table**

Description

This data set gives the results of the 1970 US draft lottery, in the form of a frequency table. The rows are months of the year, Jan–Dec and columns give the number of days in that month which fall into each of three draft risk categories High, Medium, and Low, corresponding to the chances of being called to serve in the US army.
Usage

```
data(Draft1970table)
```

Format

```
The format is: 'table' int [1:12, 1:3] 9 7 5 8 9 11 12 13 10 9 ... - attr(*, "dimnames")=List of 2 ..$ Month: chr [1:12] "Jan" "Feb" "Mar" "Apr" ... ..$ Risk : chr [1:3] "High" "Med" "Low"
```

Details

The lottery numbers are divided into three categories of risk of being called for the draft – High, Medium, and Low – each representing roughly one third of the days in a year. Those birthdays having the highest risk have lottery numbers 1-122, medium risk have numbers 123-244, and the lowest risk category contains lottery numbers 245-366.

Source

This data is available in several forms, but the table version was obtained from

```
http://sas.uwaterloo.ca/~rwoldfor/software/eikosograms/data/draft-70
```

References


See Also

```
Draft1970
```

Examples

```
data(Draft1970table)
chisq.test(Draft1970table)
```

```
# plot.table -> graphics:::mosaicplot
plot(Draft1970table, shade=TRUE)
mosaic(Draft1970table, gp=shading_Friendly)
```

```
# correspondence analysis
if(require(ca)) {
  ca(Draft1970table)
  plot(ca(Draft1970table))
}
```

```
# convert to a frequency data frame with ordered factors
Draft1970df <- as.data.frame(Draft1970table)
Draft1970df <- within(Draft1970df, {
```
Sources of Knowledge of Cancer

Description

Observational data on a sample of 1729 individuals, cross-classified in a 2^5 table according to their sources of information (read newspapers, listen to the radio, do 'solid' reading, attend lectures) and whether they have good or poor knowledge regarding cancer. Knowledge of cancer is often treated as the response.

Usage

data(Dyke)

Format

A 5-dimensional array resulting from cross-tabulating 5 variables for 1729 observations. The variable names and their levels are:

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Knowledge</td>
<td>&quot;Good&quot;, &quot;Poor&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Reading</td>
<td>&quot;No&quot;, &quot;Yes&quot;</td>
</tr>
<tr>
<td>3</td>
<td>Radio</td>
<td>&quot;No&quot;, &quot;Yes&quot;</td>
</tr>
<tr>
<td>4</td>
<td>Lectures</td>
<td>&quot;No&quot;, &quot;Yes&quot;</td>
</tr>
<tr>
<td>5</td>
<td>Newspaper</td>
<td>&quot;No&quot;, &quot;Yes&quot;</td>
</tr>
</tbody>
</table>
Source


References


Examples

data(Dyke)

# independence model
mosaic(Dyke, shade=TRUE)

# null model, Knowledge as response, independent of others
require(MASS)
dyke.mod0 <- loglm(~ Knowledge + (Reading * Radio * Lectures * Newspaper), data=Dyke)
dyke.mod0
mosaic(dyke.mod0)

# view as doubledecker plot
Dyke <- Dyke[2:1,,,]  # make Good the highlighted value of Knowledge
doubledecker(Knowledge ~ ., data=Dyke)
# better version, with some options
doubledecker(Knowledge ~ Lectures + Reading + Newspaper + Radio, data=Dyke,
margins = c(1,6, length(dim(Dyke)) + 1, 1),
fill_boxes=list(rep(c("white", gray(.90)),4))
)

# separate (conditional) plots for those who attend lectures and those who do not
doubledecker(Knowledge ~ Reading + Newspaper + Radio, data=Dyke[,,,1,],
main="Do not attend lectures",
margins = c(1,6, length(dim(Dyke)) + 1, 1),
fill_boxes=list(rep(c("white", gray(.90)),3))
)
doubledecker(Knowledge ~ Reading + Newspaper + Radio, data=Dyke[,,,2,],
main="Attend lectures",
margins = c(1,6, length(dim(Dyke)) + 1, 1),
fill_boxes=list(rep(c("white", gray(.90)),3))
)

drop1(dyke.mod0, test="Chisq")
**expand.dft**

**Expand a frequency table to case form**

**Description**

Converts a frequency table, given either as a table object or a data frame in frequency form to a data frame representing individual observations in the table.

**Usage**

```r
expand.dft(x, var.names = NULL, freq = "Freq", ...)
expand.table(x, var.names = NULL, freq = "Freq", ...)
```

**Arguments**

- **x**: A table object, or a data frame in frequency form containing factors and one numeric variable representing the cell frequency for that combination of factors.
- **var.names**: A list of variable names for the factors, if you wish to override those already in the table.
- **freq**: The name of the frequency variable in the table.
- **...**: Other arguments passed down to `type.convert`. In particular, pay attention to `na.strings` (default: `na.strings=NA` if there are missing cells) and `as.is` (default: `as.is=FALSE`, converting character vectors to factors).

**Details**

`expand.table` is a synonym for `expand.dft`.

**Value**

A data frame containing the factors in the table and as many observations as are represented by the total of the `freq` variable.

**Author(s)**

Mark Schwarz

**References**


See Also

(type.convert, expandCategorical)

Examples

library(vcd)
art <- xtabs(~Treatment + Improved, data = Arthritis)
art
artdf <- expand.dft(art)
str(artdf)

# 1D case
(tab <- table(sample(head(letters), 20, replace=TRUE)))
expand.table(tab, var.names="letter")

Fungicide Carcinogenic Effects of a Fungicide

Description

Data from Gart (1971) on the carcinogenic effects of a certain fungicide in two strains of mice. Of interest is how the association between group (Control, Treated) and outcome (Tumor, No Tumor) varies with sex and strain of the mice.

Breslow (1976) used this data to illustrate the application of linear models to log odds ratios.

Usage

data(Fungicide)

Format

The data comprise a set of four 2 x 2 tables classifying 403 mice, either Control or Treated and whether or not a tumor was later observed. The four groups represent the combinations of sex and strain of mice. The format is: num [1:2, 1:2, 1:2] 5 4 74 12 3 2 84 14 10 4 ... - attr(*, "dimnames")=List of 4 ..$ group : chr [1:2] "Control" "Treated" ..$ outcome: chr [1:2] "Tumor" "NoTumor" ..$ sex : chr [1:2] "M" "F" ..$ strain : chr [1:2] "1" "2"

Details

All tables have some small cells, so a continuity correction is recommended.

Source

Geissler

References


Examples

```r
data(Fungicide)
# loddsratio was moved to vcd; requires vcd_1.3-3+
## Not run:
if (require(vcd)) {
  fung.lor <- loddsratio(Fungicide, correct=TRUE)
  fung.lor
  confint(fung.lor)
}
## End(Not run)

# visualize odds ratios in fourfold plots
cotabplot(Fungicide, panel=cotab_fourfold)
# -- fourfold() requires vcd >= 1.2-10
fourfold(Fungicide, p_adjust_method="none")
```

Geissler

*Geissler’s Data on the Human Sex Ratio*

Description

Geissler (1889) published data on the distributions of boys and girls in families in Saxony, collected for the period 1876-1885. The Geissler data tabulates the family composition of 991,958 families by the number of boys and girls listed in the table supplied by Edwards (1958, Table 1).

Usage

```r
data(Geissler)
```

Format

A data frame with 90 observations on the following 4 variables. The rows represent the non-NA entries in Edwards’ table.

- **boys**: number of boys in the family, 0:12
- **girls**: number of girls in the family, 0:12
- **size**: family size: boys+girls
- **Freq**: number of families with this sex composition
Details

The data on family composition was available because, on the birth of a child, the parents had to state the sex of all their children on the birth certificate. These family records are not necessarily independent, because a given family may have had several children during this 10 year period, included as multiple records.

Source


References


See Also

Saxony, containing the data for families of size 12.

Examples

data(Geissler)
## maybe str(Geissler) ; plot(Geissler) ...

# reproduce Saxony data, families of size 12
Saxony12<-subset(Geissler, size==12, select=c(boys, Freq))
rownames(Saxony12)<-NULL

# make a 1-way table
xtabs(Freq~boys, Saxony12)

# extract data for other family sizes
Saxony11<-subset(Geissler, size==11, select=c(boys, Freq))
rownames(Saxony11)<-NULL
Saxony10<-subset(Geissler, size==10, select=c(boys, Freq))
rownames(Saxony10)<-NULL
Description

Schoolboys were classified according to their clothing and to their teachers rating of "dullness" (lack of intelligence), in a 5 x 7 table originally from Gilby (1911). Anscombe (1981) presents a slightly collapsed 4 x 6 table, used here, where the last two categories of clothing were pooled as were the first two categories of dullness due to small counts.

Both Dullness and Clothing are ordered categories, so models and methods that examine their association in terms of ordinal categories are profitable.

Usage

data(Gilby)

Format

A 2-dimensional array resulting from cross-tabulating 2 variables for 1725 observations. The variable names and their levels are:

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dullness</td>
<td>&quot;Ment. defective&quot;, &quot;Slow&quot;, &quot;Slow Intell&quot;, &quot;Fairly Intell&quot;, &quot;Capable&quot;, &quot;V. Able&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Clothing</td>
<td>&quot;V. Well clad&quot;, &quot;Well clad&quot;, &quot;Passable&quot;, &quot;Insufficient&quot;</td>
</tr>
</tbody>
</table>

Source


References


Examples

data(Gilby)

mosaic(Gilby, shade=TRUE)

# correspondence analysis to see relations among categories
if(require(ca)){
ca(Gilby)
plot(ca(Gilby))
title(xlab="Dimension 1", ylab="Dimension 2")
}

Gilby

Clothing and Intelligence Rating of Children

Description

Schoolboys were classified according to their clothing and to their teachers rating of "dullness" (lack of intelligence), in a 5 x 7 table originally from Gilby (1911). Anscombe (1981) presents a slightly collapsed 4 x 6 table, used here, where the last two categories of clothing were pooled as were the first two categories of dullness due to small counts.

Both Dullness and Clothing are ordered categories, so models and methods that examine their association in terms of ordinal categories are profitable.

Usage

data(Gilby)

Format

A 2-dimensional array resulting from cross-tabulating 2 variables for 1725 observations. The variable names and their levels are:

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dullness</td>
<td>&quot;Ment. defective&quot;, &quot;Slow&quot;, &quot;Slow Intell&quot;, &quot;Fairly Intell&quot;, &quot;Capable&quot;, &quot;V. Able&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Clothing</td>
<td>&quot;V. Well clad&quot;, &quot;Well clad&quot;, &quot;Passable&quot;, &quot;Insufficient&quot;</td>
</tr>
</tbody>
</table>

Source


References


Examples

data(Gilby)

mosaic(Gilby, shade=TRUE)

# correspondence analysis to see relations among categories
if(require(ca)){
ca(Gilby)
plot(ca(Gilby))
title(xlab="Dimension 1", ylab="Dimension 2")
}
Calculate Goodman-Kruskal Gamma for ordered tables

Description

The Goodman-Kruskal $\gamma$ statistic is a measure of association for ordinal factors in a two-way table proposed by Goodman and Kruskal (1954).

Usage

GKgamma(x, level = 0.95)

Arguments

- **x**: A two-way frequency table, in matrix or table form. The rows and columns are considered to be ordinal factors
- **level**: Confidence level for a significance test of $\gamma \neq$

Value

Returns an object of class "GKgamma" with 6 components, as follows

- **gamma**: The gamma statistic
- **C**: Total number of concordant pairs in the table
- **D**: Total number of disconcordant pairs in the table
- **sigma**: Standard error of gamma
- **CIlevel**: Confidence level
- **CI**: Confidence interval

Author(s)

Michael Friendly; original version by Laura Thompson

References


See Also

assocstats, Kappa

Examples

data(JobSat)
GKgamma(JobSat)

Description

Glass(1954) gave this 5 x 5 table on the occupations of 3500 British fathers and their sons.

Usage

data("Glass")

Format

A frequency data frame with 25 observations on the following 3 variables representing a 5 x 5 table with 3500 cases.

father a factor with levels Managerial Professional Skilled Supervisory Unskilled
son a factor with levels Managerial Professional Skilled Supervisory Unskilled
Freq a numeric vector

Details

The occupational categories in order of status are: (1) Professional \& High Administrative (2) Managerial, Executive \& High Supervisory (3) Low Inspectional \& Supervisory (4) Routine Non-manual \& Skilled Manual (5) Semi- \& Unskilled Manual

However, to make the point that factors are ordered alphabetically by default, Friendly \& Meyer (2016) introduce this data set in the form given here.

Source

Glass, D. V. (1954), Social Mobility in Britain. The Free Press.

References


Examples

```r
data(Glass)
glass.tab <- xtabs(Freq ~ father + son, data=Glass)
largs <- list(set_varnames=list(father="Father's Occupation", son="Son's Occupation"), abbreviate=10)
gargs <- list(interpolate=c(1,2,4,8))
mosaic(glass.tab, shade=TRUE, labeling_args=largs, gp_args=gargs, main="Alphabetic order", legend=FALSE, rot_labels=c(20,90,0,70))

# reorder by status
ord <- c(2, 1, 4, 3, 5)
mosaic(glass.tab[ord, ord], shade=TRUE, labeling_args=largs, gp_args=gargs, main="Effect order", legend=FALSE, rot_labels=c(20,90,0,70))
```

---

`glmlist`  
**Create a Model List Object**

Description

`glmlist` creates a `glmlist` object containing a list of fitted `glm` objects with their names. `loglmlist` does the same for `loglm` objects.

The intention is to provide object classes to facilitate model comparison, extraction, summary and plotting of model components, etc., perhaps using `lapply` or similar.

There exists a `anova.glm` method for `glmlist` objects. Here, a `coef` method is also defined, collecting the coefficients from all models in a single object of type determined by `result`.

Usage

```r
glmlist(...)
loglmlist(...)
```

```r
## S3 method for class 'glmlist'
coef(object, result=c("list", "matrix", "data.frame"), ...)
```

Arguments

- `...`: One or more model objects, as appropriate to the function, optionally assigned names as in list.
- `object`: a `glmlist` object
- `result`: type of the result to be returned
Details

The arguments to glmlist or loglmlist are of the form value or name=value. Any objects which do not inherit the appropriate class glm or loglm are excluded, with a warning. In the coef method, coefficients from the different models are matched by name in the list of unique names across all models.

Value

An object of class glmlist loglmlist, just like a list, except that each model is given a name attribute.

Author(s)

Michael Friendly; coef method by John Fox

See Also

The function llist in package Hmisc is similar, but perplexingly more general.

The function anova.glm also handles glmlist objects LRstats gives LR statistics and tests for a glmlist object.

Examples

data(Mental)
indep <- glm(Freq ~ mental+ses,
   family = poisson, data = Mental)
Cscore <- as.numeric(Mental$ses)
Rscore <- as.numeric(Mental$mental)
coleff <- glm(Freq ~ mental + ses + Rscore:ses,
   family = poisson, data = Mental)
roweff <- glm(Freq ~ mental + ses + mental:Cscore,
   family = poisson, data = Mental)
linlin <- glm(Freq ~ mental + ses + Rscore:Cscore,
   family = poisson, data = Mental)

# use object names
mods <- glmlist(indep, coleff, roweff, linlin)
names(mods)

# assign new names
mods <- glmlist(Indep=indep, Col=coleff, Row=roweff, LinxLin=linlin)
names(mods)

LRstats(mods)

coef(mods, result='data.frame')

#extract model components
unlist(lapply(mods, deviance))
res <- lapply(mods, residuals)
boxplot(as.data.frame(res), main="Residuals from various models")

---

GSS  
**General Social Survey– Sex and Party affiliation**

---

**Description**

Data from the General Social Survey, 1991, on the relation between sex and party affiliation.

**Usage**

data(GSS)

**Format**

A data frame in frequency form with 6 observations on the following 3 variables.

- **sex** a factor with levels female male
- **party** a factor with levels dem indep rep
- **count** a numeric vector

**Source**

Agresti, A. Categorical Data Analysis John Wiley & Sons, 2002, Table 3.11, p. 106.

**Examples**

data(GSS)
## maybe str(GSS) ; plot(GSS) ...
(GSStab <- xtabs(count ~ sex + party, data=GSS))

mod.glm <- glm(count ~ sex + party, family = poisson, data = GSS)
Description

A three-way frequency table crossing eye color and hair color in two places, Caithness and Aberdeen, Scotland. These data were of interest to Fisher (1940) and others because there are mixtures of people of Nordic, Celtic and Anglo-Saxon origin. One or both tables have been widely analyzed in conjunction with RC and canonical correlation models for categorical data, e.g., Becker and Clogg (1989).

Usage

data(HairEyePlace)

Format

The format is: num [1:4, 1:5, 1:2] 326 688 343 98 38 116 84 48 241 584 ... - attr(*, "dimnames")=List of 3 ...
[ 1] "Eye" : chr [1:4] "Blue" "Light" "Medium" "Dark" ...
[ 2] "Hair" : chr [1:5] "Fair" "Red" "Medium" "Dark" ...

Details

The hair and eye colors are ordered as in the original source, suggesting that they form ordered categories.

Source

This data was taken from the colors data in logmult.

References


Examples

data(HairEyePlace)

# separate mosaics
mosaic(HairEyePlace[,,1], shade=TRUE, main="Caithness")
mosaic(HairEyePlace[,,2], shade=TRUE, main="Aberdeen")

# condition on Place
mosaic(~Hair + Eye |Place, data=HairEyePlace, shade=TRUE, legend=FALSE)
cotabplot(~Hair+Eye|Place, data=HairEyePlace, shade=TRUE, legend=FALSE)
Description

Hauser (1979) presented this two-way frequency table, cross-classifying occupational categories of sons and fathers in the United States.

Usage

data(Hauser79)

Format

A frequency data frame with 25 observations on the following 3 variables, representing the cross-classification of 19912 individuals by father’s occupation and son’s first occupation.

Son  a factor with levels UpNM LoNM UpM LoM Farm
Father a factor with levels UpNM LoNM UpM LoM Farm
Freq  a numeric vector

Source


References


Examples

data(Hauser79)
str(Hauser79)

# display table
structable(~Father+Son, data=Hauser79)

#Examples from Powers & Xie, Table 4.15
# independence model
mosaic(Freq ~ Father + Son, data=Hauser79, shade=TRUE)

hauser.indep <- gnm(Freq ~ Father + Son, data=Hauser79, family=poisson)
mosaic(hauser.indep, ~Father+Son, main="Independence model", gp=shading_Friendly)

hauser.quasi <- update(hauser.indep, ~ . + Diag(Father,Son))
mosaic(hauser.quasi, ~Father+Son, main="Quasi-independence model", gp=shading_Friendly)

hauser.qsymm <- update(hauser.indep, ~ . + Diag(Father, Son) + Symm(Father, Son))
mosaic(hauser.qsymm, ~Father+Son, main="Quasi-symmetry model", gp=shading_Friendly)

# numeric scores for row/column effects
Sscore <- as.numeric(Hauser79$Son)
Fscore <- as.numeric(Hauser79$Father)

# row effects model
hauser.roweff <- update(hauser.indep, ~ . + Father*Sscore)
LRstats(hauser.roweff)

# uniform association
hauser.UA <- update(hauser.indep, ~ . + Fscore*Sscore)
LRstats(hauser.UA)

# uniform association, omitting diagonals
hauser.UAdiag <- update(hauser.indep, ~ . + Fscore*Sscore + Diag(Father, Son))
LRstats(hauser.UAdiag)

# Levels for Hauser 5-level model
levels <- matrix(c(
2, 4, 5, 5, 5,
3, 4, 5, 5, 5,
5, 5, 5, 5, 5,
5, 5, 5, 4, 4,
5, 5, 5, 4, 1
), 5, 5, byrow=TRUE)

hauser.topo <- update(hauser.indep, ~ . + Topo(Father, Son, spec=levels))
mosaic(hauser.topo, ~Father+Son, main="Topological model", gp=shading_Friendly)

hauser.RC <- update(hauser.indep, ~ . + Mult(Father, Son), verbose=FALSE)
mosaic(hauser.RC, ~Father+Son, main="RC model", gp=shading_Friendly)
LRstats(hauser.RC)

# crossings models
hauser.CR <- update(hauser.indep, ~ . + Crossings(Father, Son))
mosaic(hauser.topo, ~Father+Son, main="Crossings model", gp=shading_Friendly)
LRstats(hauser.CR)

hauser.CRdiag <- update(hauser.indep, ~ . + Crossings(Father, Son) + Diag(Father, Son))
LRstats(hauser.CRdiag)

# compare model fit statistics
modlist <- glmlist(hauser.indep, hauser.roweff, hauser.UA, hauser.UAdiag,
hauser.quasi, hauser.qsymm, hauser.topo,
hauser.RC, hauser.CR, hauser.CRdiag)
sumry <- LRstats(modlist)
Heart

Summary

# or, more simply
LRstats(modlist, sortby="AIC")

mods <- substring(rownames(sumry),8)
with(sumry,
{
plot(Df, AIC, cex=1.3, pch=19, xlab='Degrees of freedom', ylab='AIC')
text(Df, AIC, mods, adj=c(0.5,-.5), col='red', xpd=TRUE)
})

Heart

Sex, Occupation and Heart Disease

Description

Classification of individuals by gender, occupational category and occurrence of heart disease

Usage

data(Heart)

Format

A 3-dimensional array resulting from cross-tabulating 3 variables for 21522 observations. The variable names and their levels are:

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Disease</td>
<td>&quot;Disease&quot;, &quot;None&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Gender</td>
<td>&quot;Male&quot;, &quot;Female&quot;</td>
</tr>
<tr>
<td>3</td>
<td>Occup</td>
<td>&quot;Unempl&quot;, &quot;WhiteCol&quot;, &quot;BlueCol&quot;</td>
</tr>
</tbody>
</table>

Source

Karger, (1980).

Examples

data(Heart)

# example goes here
Heckman


Description

1583 married women were surveyed over the years 1967-1971, recording whether or not they were employed in the labor force.

The data, originally from Heckman & Willis (1977) provide an example of modeling longitudinal categorical data, e.g., with Markov chain models for dependence over time.

Usage

```r
data(Heckman)
```

Format

A 5-dimensional array resulting from cross-tabulating 5 variables for 1583 observations. The variable names and their levels are:

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>e1971</td>
<td>&quot;71Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td>2</td>
<td>e1970</td>
<td>&quot;70Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td>3</td>
<td>e1969</td>
<td>&quot;69Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td>4</td>
<td>e1968</td>
<td>&quot;68Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td>5</td>
<td>e1967</td>
<td>&quot;67Yes&quot;, &quot;No&quot;</td>
</tr>
</tbody>
</table>

Details


Alternatively, one can examine Markov chain models of first-order (dependence on previous year), second-order (dependence on previous two years), etc.

Source


References


Examples

```r
data(Heckman)
```
# independence model
mosaic(Heckman, shade=TRUE)
# same, as a loglm()
require(MASS)
(heckman.mod0 <- loglm(~ e1971+e1970+e1969+e1968+e1967, data=Heckman))
mosaic(heckman.mod0, main="Independence model")

# first-order markov chain: bad fit
mosaic(heckman.mod1, main="1st order markov chain model")

# second-order markov chain: bad fit
mosaic(heckman.mod2, main="2nd order markov chain model")

# third-order markov chain: fits OK
mosaic(heckman.mod2, main="3rd order markov chain model")

---

**HLtest**

*Hosmer-Lemeshow Goodness of Fit Test*

**Description**

The **HLtest** function computes the classical Hosmer-Lemeshow (1980) goodness of fit test for a binomial glm object in logistic regression.

The general idea is to assesses whether or not the observed event rates match expected event rates in subgroups of the model population. The Hosmer-Lemeshow test specifically identifies subgroups as the deciles of fitted event values, or other quantiles as determined by the `g` argument. Given these subgroups, a simple chisquare test on `g-2 df` is used.

In addition to print and summary methods, a plot method is supplied to visualize the discrepancies between observed and fitted frequencies.

**Usage**

```
HosmerLemeshow(model, g = 10)
HLtest(model, g = 10)
```

```
## S3 method for class 'HLtest'
print(x, ...)
## S3 method for class 'HLtest'
summary(object, ...)
## S3 method for class 'HLtest'
plot(x, ...)
```
## S3 method for class 'HLtest'
rootogram(x, ...)

### Arguments
- **model**: A glm model object in the binomial family
- **g**: Number of groups used to partition the fitted values for the GOF test.
- **x, object**: A HLtest object
- **...**: Other arguments passed down to methods

### Value
A class HLtest object with the following components:
- **table**: A data.frame describing the results of partitioning the data into g groups with the following columns: cut, total, obs, exp, chi
- **chisq**: The chisquared statistics
- **df**: Degrees of freedom
- **p.value**: p value
- **groups**: Number of groups
- **call**: model call

### Author(s)
Michael Friendly

### References

### See Also
- `rootogram`, ~~~

### Examples
```r
data(birthwt, package="MASS")
# how to do this without attach?
attach(birthwt)
race = factor(race, labels = c("white", "black", "other"))```
HospVisits

Hospital Visits Data

Description

Length of stay in hospital for 132 schizophrenic patients, classified by visiting patterns, originally from Wing (1962).

Usage

data("HospVisits")

Format


Details

Both table variables can be considered ordinal. The variable visit refers to visiting patterns recorded in hospital. The category labels are abbreviations of those given by Goodman (1983); e.g., "Regular" is short for "received visitors regularly or patient went home". The variable stay refers to length of stay in hospital, in year groups.

---

ptd = factor(ptl > 0)
ftv = factor(ftv)
levels(ftv)[-c(1:2)] = "2+
bwt <- data.frame(low = factor(low), age, lwt, race,
smoke = (smoke > 0), ptd, ht = (ht > 0), ui = (ui > 0), ftv)
detach(birthwt)
options(contrasts = c("contr.treatment", "contr.poly"))
BWmod <- glm(low ~ ., family=binomial, data=bwt)

(hlt <- HLtest(BWmod))
str(hlt)
summary(hlt)
plot(hlt)

# basic model
BWmod0 <- glm(low ~ age, family=binomial, data=bwt)
(hlt0 <- HLtest(BWmod0))
str(hlt0)
summary(hlt0)
plot(hlt0)
Hoyt

Source


References


Examples

data(HospVisits)
mosaic(HospVisits, gp=shading_Friendly)
library(ca)
ca(HospVisits)
# surprisingly 1D!
plot(ca(HospVisits))

Hoyt

Minnesota High School Graduates

Description

Minnesota high school graduates of June 1930 were classified with respect to (a) Rank by thirds in their graduating class, (b) post-high school Status in April 1939 (4 levels), (c) Sex, (d) father’s Occupational status (7 levels, from 1=High to 7=Low).

The data were first presented by Hoyt et al. (1959) and have been analyzed by Fienberg(1980), Plackett(1974) and others.

Usage

data(Hoyt)

Format

A 4-dimensional array resulting from cross-tabulating 4 variables for 13968 observations. The variable names and their levels are:

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Status</td>
<td>&quot;College&quot;, &quot;School&quot;, &quot;Job&quot;, &quot;Other&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Rank</td>
<td>&quot;Low&quot;, &quot;Middle&quot;, &quot;High&quot;</td>
</tr>
<tr>
<td>3</td>
<td>Occupation</td>
<td>&quot;1&quot;, &quot;2&quot;, &quot;3&quot;, &quot;4&quot;, &quot;5&quot;, &quot;6&quot;, &quot;7&quot;</td>
</tr>
<tr>
<td>4</td>
<td>Sex</td>
<td>&quot;Male&quot;, &quot;Female&quot;</td>
</tr>
</tbody>
</table>
Details

Post high-school Status is natural to consider as the response. Rank and father’s Occupation are ordinal variables.

Source


References


See Also

minn38 provides the same data as a data frame.

Examples

data(Hoyt)

# display the table
structable(Status+Sex ~ Rank+Occupation, data=Hoyt)

# mosaic for independence model
plot(Hoyt, shade=TRUE)

# examine all pairwise mosaics
pairs(Hoyt, shade=TRUE)

# collapse Status to College vs. Non-College
Hoyt1 <- collapse.table(Hoyt, Status=c("College", rep("Non-College",3)))
plot(Hoyt1, shade=TRUE)

#################################################
# fitting models with loglm, plotting with mosaic
#################################################

# fit baseline log-linear model for Status as response
require(MASS)
hoyt.mod0 <- loglm(~ Status + (Sex*Rank*Occupation), data=Hoyt1)
hoyt.mod0
mosaic(hoyt.mod0, gp=shading_Friendly, main="Baseline model: Status + (Sex*Rank*Occ)"

# add one-way association of Status with factors
hoyt.mod1 <- loglm(~ Status * (Sex + Rank + Occupation) + (Sex*Rank*Occupation), data=Hoyt1)
hoyt.mod1
mosaic(hoyt.mod1, gp=shading_Friendly, main="Status * (Sex + Rank + Occ)"
# can we drop any terms?
drop1(hoyt.mod1, test="Chisq")

# assess model fit
anova(hoyt.mod0, hoyt.mod1)

# what terms to add?
add1(hoyt.mod1, ~.^2, test="Chisq")

# add interaction of Sex:Occupation on Status
hoyt.mod2 <- update(hoyt.mod1, ~.+Status:Sex:Occupation)
mosaic(hoyt.mod2, gp=shading_Friendly, main="Adding Status:Sex:Occupation")

# compare model fits
anova(hoyt.mod0, hoyt.mod1, hoyt.mod2)

# Alternatively, try stepwise analysis, heading toward the saturated model
steps <- step(hoyt.mod0, direction="forward", scope=~Status*Sex*Rank*Occupation)
# display anova
steps$anova

---

ICU data set

**Description**

The ICU data set consists of a sample of 200 subjects who were part of a much larger study on survival of patients following admission to an adult intensive care unit (ICU), derived from Hosmer, Lemeshow and Sturdivant (2013) and Friendly (2000).

The major goal of this study was to develop a logistic regression model to predict the probability of survival to hospital discharge of these patients and to study the risk factors associated with ICU mortality. The clinical details of the study are described in Lemeshow, Teres, Avrunin, and Pastides (1988).

This data set is often used to illustrate model selection methods for logistic regression.

**Usage**

data(ICU)

**Format**

A data frame with 200 observations on the following 22 variables.

died  Died before discharge?, a factor with levels No Yes
age   Patient age, a numeric vector
sex   Patient sex, a factor with levels Female Male
race  Patient race, a factor with levels Black Other White. Also represented here as white.
Service at ICU Admission, a factor with levels Medical Surgical
Cancer part of present problem?, a factor with levels No Yes
History of chronic renal failure?, a factor with levels No Yes
Infection probable at ICU admission?, a factor with levels No Yes
Patient received CPR prior to ICU admission?, a factor with levels No Yes
Systolic blood pressure at admission (mm Hg), a numeric vector
Heart rate at ICU Admission (beats/min), a numeric vector
Previous admission to an ICU within 6 Months?, a factor with levels No Yes
Type of admission, a factor with levels Elective Emergency
Admission with a long bone, multiple, neck, single area, or hip fracture? a factor with levels No Yes
PO2 from initial blood gases, a factor with levels >60 <=60
pH from initial blood gases, a factor with levels >=7.25 <7.25
PCO2 from initial blood gases, a factor with levels <=45 >45
Bicarbonate (HCO3) level from initial blood gases, a factor with levels >=18 <18
Creatinine, from initial blood gases, a factor with levels <=2 >2
Level of unconsciousness at admission to ICU, a factor with levels None Stupor Coma
a recoding of race, a factor with levels White Non-white
a recoding of coma a factor with levels No Yes

Details

Patient ID numbers are the rownames of the data frame.

Note that the last two variables white and uncons are a recoding of respectively race and coma to binary variables.

Source


References

Examples

```r
data(ICU)
# remove redundant variables (race, coma)
ICU1 <- ICU[,,-c(4,20)]

# fit full model
icu.full <- glm(died ~ ., data=ICU1, family=binomial)
summary(icu.full)

# simpler model (found from a "best" subsets procedure)
icu.mod1 <- glm(died ~ age + sex + cancer + systolic + admit + uncons, data=ICU1, family=binomial)
summary(icu.mod1)

# even simpler model
icu.mod2 <- glm(died ~ age + cancer + admit + uncons, data=ICU1, family=binomial)
summary(icu.mod2)

anova(icu.mod2, icu.mod1, icu.full, test="Chisq")

# Reproduce Fig 6.12 from VCD
icu.fit <- data.frame(ICU, prob=predict(icu.mod2, type="response"))

# combine categorical risk factors to a single string
risks <- ICU[, c("cancer", "admit", "uncons")]
risks[,1] <- ifelse(risks[,1]=="Yes", "Cancer", "")
risks[,3] <- ifelse(risks[,3]=="Yes", "Uncons", "")
risks <- apply(risks, 1, paste, collapse="")
risks[risks==""] <- "(none)"
icu.fit$risks <- risks

library(ggplot2)
ggplot(icu.fit, aes(x=age, y=prob, color=risks)) + geom_point(size=2) +
geom_line(size=1.25, alpha=0.5) +
theme_bw() + ylab("Probability of death")
```

Description

This data set is a contingency table of job satisfaction by income for a small sample of black males from the 1996 General Social Survey, as used by Agresti (2002) for an example.

Usage

data(JobSat)
Format

A 4 x 4 contingency table of income by satisfaction, with the following structure:

```r
table [1:4, 1:4] 1 2 1 0 3 3 6 1 10 10 ...
- attr(*, "dimnames")=List of 2
  ..$ income : chr [1:4] "< 15k" "15-25k" "25-40k" "> 40k"
  ..$ satisfaction: chr [1:4] "VeryD" "LittleD" "ModerateS" "VeryS"
```

Details

Both income and satisfaction are ordinal variables, and are so ordered in the table. Measures of association, visualizations, and models should take ordinality into account.

Source

Agresti, A. Categorical Data Analysis John Wiley & Sons, 2002, Table 2.8, p. 57.

Examples

```r
data(JobSat)
assocstats(JobSat)
GKgamma(JobSat)
```

---

**Kway**  
**Fit All K-way Models in a GLM**

Description

Generate and fit all 0-way, 1-way, 2-way, ... k-way terms in a glm.

This function is designed mainly for hierarchical loglinear models (or glm's in the poission family), where it is desired to find the highest-order terms necessary to achieve a satisfactory fit.

Using `anova` on the resulting `glmlist` object will then give sequential tests of the pooled contributions of all terms of degree \( k + 1 \) over and above those of degree \( k \).

This function is also intended as an example of a generating function for `glmlist` objects, to facilitate model comparison, extraction, summary and plotting of model components, etc., perhaps using `lapply` or similar.

Usage

```r
Kway(formula, family=poisson, data, ..., order = nt, prefix = "kway")
```
Arguments

- **formula**: a two-sided formula for the 1-way effects in the model. The LHS should be the response, and the RHS should be the first-order terms connected by + signs.

- **family**: a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See `family` for details of family functions.)

- **data**: an optional data frame, list or environment (or object coercible by `as.data.frame`) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `glm` is called.

- **...**: Other arguments passed to `glm`

- **order**: Highest order interaction of the models generated. Defaults to the number of terms in the model formula.

- **prefix**: Prefix used to label the models fit in the `glmlist` object.

Details

With y as the response in the formula, the 0-way (null) model is \( y \sim 1 \). The 1-way ("main effects") model is that specified in the `formula` argument. The k-way model is generated using the formula \( . \sim \cdot^k \). With the default `order = nt`, the final model is the saturated model.

As presently written, the function requires a two-sided formula with an explicit response on the LHS. For frequency data in table form (e.g., produced by `xtabs`) you the `data` argument is coerced to a data frame, so you should supply the formula in the form `Freq ~ ....`

Value

An object of class `glmlist`, of length `order+1` containing the 0-way, 1-way, ... models up to degree `order`.

Author(s)

Michael Friendly and Heather Turner

See Also

- `glmlist`, `Summarise` (soon to be deprecated), `LRstats`

Examples

```R
## artificial data
factors <- expand.grid(A=factor(1:3), B=factor(1:2), C=factor(1:3), D=factor(1:2))
Freq <- rpois(nrow(factors), lambda=40)
df <- cbind(factors, Freq)
mods3 <- Kway(Freq ~ A + B + C, data=df, family=poisson)
LRstats(mods3)
mods4 <- Kway(Freq ~ A + B + C + D, data=df, family=poisson)
LRstats(mods4)
```
# JobSatisfaction data
data(JobSatisfaction, package="vcd")
modSat <- Kway(Freq ~ management+supervisor+own, data=JobSatisfaction, family=poisson, prefix="JobSat")
LRstats(modSat)
anova(modSat, test="Chisq")

# Rochdale data: very sparse, in table form
data(Rochdale, package="vcd")
## Not run:
modRoch <- Kway(Freq~EconActive + Age + HusbandEmployed + Child +
                Education + HusbandEducation + Asian + HouseholdWorking,
data=Rochdale, family=poisson)
LRstats(modRoch)
## End(Not run)

---

**logLik.loglm**  
*Log-Likelihood of a loglm Object*

**Description**

Calculates the log-likelihood value of the loglm model represented by object evaluated at the estimated coefficients.

It allows the use of **AIC** and **BIC**, which require that a logLik method exists to extract the corresponding log-likelihood for the model.

**Usage**

```r
## S3 method for class 'loglm'
logLik(object, ..., zero=1E-10)
```

**Arguments**

- `object`  
  A loglm object

- `...`  
  For compatibility with the S3 generic; not used here

- `zero`  
  value used to replace zero frequencies in calculating the log-likelihood

**Details**

If cell frequencies have not been stored with the loglm object (via the argument `keep.frequencies = TRUE`), they are obtained using `update`.

This function calculates the log-likelihood in a way that allows for non-integer frequencies, such as the case where 0.5 has been added to all cell frequencies to allow for sampling zeros. If the frequencies still contain zero values, those are replaced by the value of `start`.

For integer frequencies, it gives the same result as the corresponding model fit using `glm`, whereas `glm` returns `-Inf` if there are any non-integer frequencies.
Value

Returns an object of class \texttt{logLik}. This is a number with one attribute, "df" (degrees of freedom), giving the number of (estimated) parameters in the model.

Author(s)

Achim Zeileis

See Also

\texttt{loglm}, \texttt{AIC}, \texttt{BIC},

Examples

data(Titanic, package="datasets")

require(MASS)
titanic.mod1 <- loglm(~ (Class * Age * Sex) + Survived, data=Titanic)
titanic.mod2 <- loglm(~ (Class * Age * Sex) + Survived*(Class + Age + Sex), data=Titanic)
titanic.mod3 <- loglm(~ (Class * Age * Sex) + Survived*(Class + Age * Sex), data=Titanic)

logLik(titanic.mod1)
AIC(titanic.mod1, titanic.mod2, titanic.mod3)
BIC(titanic.mod1, titanic.mod2, titanic.mod3)

# compare with models fit using glm()
titanic <- as.data.frame(Titanic)
titanic(glm1) <- glm(Freq ~ (Class * Age * Sex) + Survived, data=titanic, family=poisson)
titanic(glm2) <- glm(Freq ~ (Class * Age * Sex) + Survived*(Class + Age + Sex), data=titanic, family=poisson)
titanic(glm3) <- glm(Freq ~ (Class * Age * Sex) + Survived*(Class + Age * Sex), data=titanic, family=poisson)

logLik(titanic.glml)
AIC(titanic.glml, titanic.glm2, titanic.glm3)
BIC(titanic.glml, titanic.glm2, titanic.glm3)

---

Description

These functions generate lists of terms to specify a loglinear model in a form compatible with \texttt{loglin} and also provide for conversion to an equivalent \texttt{loglm} specification or a shorthand character string representation.
They allow for a more conceptual way to specify such models by a function for their type, as opposed to just an uninterpreted list of model terms and also allow easy specification of marginal models for a given contingency table. 
They are intended to be used as tools in higher-level modeling and graphics functions, but can also be used directly.

Usage

conditional(nf, table = NULL, factors = 1:nf, with = nf)

joint(nf, table = NULL, factors = 1:nf, with = nf)

markov(nf, factors = 1:nf, order = 1)

mutual(nf, table = NULL, factors = 1:nf)

saturated(nf, table = NULL, factors = 1:nf)

loglin2formula(x, env = parent.frame())

loglin2string(x, brackets = c("[", "]"), sep = ",", collapse = " ", abbrev)

Arguments

nf  number of factors for which to generate the model
table a contingency table used only for factor names in the model, typically the output from table and possibly permuted with aperm
factors names of factors used in the model formula when table is not specified
with For joint and conditional models, with gives the indices of the factors against which all others are considered jointly or conditionally independent
order For markov, this gives the order of the Markov chain model for the factors. An order=1 Markov chain allows associations among sequential pairs of factors, e.g., [A, B], [B, C], [C, D] ... An order=2 Markov chain allows associations among sequential triples.
x For the loglin2* functions, a list of terms in a loglinear model, such as returned by conditional, joint,...
env For loglin2formula, environment in which to evaluate the formula
brackets For loglin2string, characters to use to surround model terms. Either a single character string containing two characters (e.g., '[]') or a character vector of length two.
sep For loglin2string, the separator character string used for factor names within a given model term
collapse For loglin2string, the character string used between terms in the model string
For `loglin2string`, whether and how to abbreviate the terms in the string representation. This has not yet been implemented.

**Details**

The main model specification functions, `conditional`, `joint`, `markov`, ..., `saturated`, return a list of vectors indicating the marginal totals to be fit, via the margin argument to `loglin`. Each element of this list corresponds to a high-order term in a hierarchical loglinear model, where, e.g., a term like `c("A", "B")` is equivalent to the `loglm` term "A:B" and hence automatically includes all low-order terms.

Note that these can be used to supply the expected argument for the default `mosaic` function, when the data is supplied as a contingency table.

The table below shows some typical results in terms of the standard shorthand notation for loglinear models, with factors A, B, C, ..., where brackets are used to delimit the high-order terms in the loglinear model.

<table>
<thead>
<tr>
<th>function</th>
<th>3-way</th>
<th>4-way</th>
<th>5-way</th>
</tr>
</thead>
<tbody>
<tr>
<td>mutual</td>
<td>[A] [B] [C]</td>
<td>[A] [B] [C] [D]</td>
<td>[A] [B] [C] [D] [E]</td>
</tr>
<tr>
<td>joint</td>
<td>[AB] [C]</td>
<td>[ABCD] [D]</td>
<td>[ABCE] [E]</td>
</tr>
<tr>
<td>joint (with=1)</td>
<td>[AC] [BC]</td>
<td>[AD] [BD] [CD]</td>
<td>[AE] [BE] [CE] [DE]</td>
</tr>
<tr>
<td>conditional</td>
<td>[AB] [AC]</td>
<td>[AB] [AC] [AD]</td>
<td>[AB] [AC] [AD] [AE]</td>
</tr>
<tr>
<td>condit (with=1)</td>
<td>[ABC] [BCD]</td>
<td>[ABC] [BCD]</td>
<td>[ABC] [BCD] [CDE]</td>
</tr>
<tr>
<td>markov (order=1)</td>
<td>[AB] [BC]</td>
<td>[AB] [BC] [CD]</td>
<td>[AB] [BC] [CD] [DE]</td>
</tr>
<tr>
<td>markov (order=2)</td>
<td>[ABC] [BCD]</td>
<td>[ABC] [BCD]</td>
<td>[ABC] [BCD] [CDE]</td>
</tr>
<tr>
<td>saturated</td>
<td>[ABC]</td>
<td>[ABCD]</td>
<td>[ABCDE]</td>
</tr>
</tbody>
</table>

`loglin2formula` converts the output of one of these to a model formula suitable as the formula for `loglm`.

`loglin2string` converts the output of one of these to a string describing the loglinear model in the shorthand bracket notation, e.g., "[A,B] [A,C]".

**Value**

For the main model specification functions, `conditional`, `joint`, `markov`, ..., the result is a list of vectors (terms), where the elements in each vector are the names of the factors. The elements of the list are given names `term1`, `term2`, ...

**Author(s)**

Michael Friendly

**References**

These functions were inspired by the original SAS implementation of mosaic displays, described in the User’s Guide, [http://www.datavis.ca/mosaics/mosaics.pdf](http://www.datavis.ca/mosaics/mosaics.pdf)
See Also

loglin, loglm

Examples

    joint(3, table=HairEyeColor)
    # as a formula or string
    loglin2formula(joint(3, table=HairEyeColor))
    loglin2string(joint(3, table=HairEyeColor))

    joint(2, HairEyeColor)  # marginal model for [Hair] [Eye]

    # other possibilities
    joint(4, factors=letters, with=1)
    joint(5, factors=LETTERS)
    joint(5, factors=LETTERS, with=4:5)

    conditional(4)
    conditional(4, with=3:4)

    # use in mosaic displays or other strucplots
    mosaic(HairEyeColor, expected=joint(3))
    mosaic(HairEyeColor, expected=conditional(3))

    # use with MASS::loglm
    cond3 <- loglin2formula(conditional(3, table=HairEyeColor))
    cond3 <- loglin2formula(conditional(3))  # same, with factors 1,2,3
    require(MASS)
    loglm(cond3, data=HairEyeColor)

    saturated(3, HairEyeColor)
    loglin2formula(saturated(3, HairEyeColor))
    loglin2string(saturated(3, HairEyeColor))
    loglin2string(saturated(3, HairEyeColor), brackets='()', sep=' ', ')

---

logseries

The Logarithmic Series Distribution

Description

The logarithmic series distribution is a long-tailed distribution introduced by Fisher etal. (1943) in connection with data on the abundance of individuals classified by species.

These functions provide the density, distribution function, quantile function and random generation for the logarithmic series distribution with parameter prob.
usage

dlogseries(x, prob = 0.5, log = FALSE)
plogseries(q, prob = 0.5, lower.tail = TRUE, log.p = FALSE)
qlogseries(p, prob = 0.5, lower.tail = TRUE, log.p = FALSE, max.value = 10000)
rlogseries(n, prob = 0.5)

arguments

x, q vector of quantiles representing the number of events.
prob parameter for the distribution, 0 < prob < 1
log, log.p logical; if TRUE, probabilities p are given as log(p)
lower.tail logical; if TRUE (default), probabilities are P[X ≤ x], otherwise, P[X > x].
p vector of probabilities
max.value maximum value returned by qlogseries
n number of observations for rlogseries

details

The logarithmic series distribution with prob = p has density

\[ p(x) = \alpha p^x / x \]

for \( x = 1, 2, \ldots \), where \( \alpha = -1 / \log(1 - p) \) and \( 0 < p < 1 \). Note that counts \( x = 2 \) cannot occur.

value

dlogseries gives the density, plogseries gives the distribution function, qlogseries gives the quantile function, and rlogseries generates random deviates.

author(s)

Michael Friendly, using original code modified from the gmlss.dist package by Mikis Stasinopoulos.

references

https://en.wikipedia.org/wiki/Logarithmic_distribution


see also

Distributions, ~~~
Examples

```r
XL <- expand.grid(x=1:5, p=c(0.33, 0.66, 0.99))
lgd <- data.frame(XL, prob=dlogseries(XL[,"x"], XL[,"p"],))
lgd$p = factor(lgd$p)
str(lgd)
```

```r
require(lattice)
mycol <- palette()[2:4]
xyplot(prob ~ x, data=lgd, groups=p,
xlab=list("Number of events (k)", cex=1.25),
ylab=list("Probability", cex=1.25),
type='b', pch=15:17, lwd=2, cex=1.25, col=mycol,
key = list(
title = 'p',
points = list(pch=15:17, col=mycol, cex=1.25),
lines = list(lwd=2, col=mycol),
text = list(levels(lgd$p)),
x=0.9, y=0.98, corner=c(x=1, y=1)
)
```

```r
# random numbers
hist(rlogseries(200, prob=.4), xlab='x')
hist(rlogseries(200, prob=.8), xlab='x')
```

---

**LRstats**

**Brief Summary of Model Fit for glm and loglm Models**

**Description**

For `glm` objects, the print and summary methods give too much information if all one wants to see is a brief summary of model goodness of fit, and there is no easy way to display a compact comparison of model goodness of fit for a collection of models fit to the same data. All `loglm` models have equivalent `glm` forms, but the print and summary methods give quite different results.

`LRstats` provides a brief summary for one or more models fit to the same dataset for which `logLik` and `nobs` methods exist (e.g., `glm` and `loglm` models).

**Usage**

```r
LRstats(object, ...)
```

---

## S3 method for class 'glmlist'
```r
LRstats(object, ..., saturated = NULL, sortby = NULL)
## S3 method for class 'loglmlist'
LRstats(object, ..., saturated = NULL, sortby = NULL)
## Default S3 method:
LRstats(object, ..., saturated = NULL, sortby = NULL)
```
LRstats

Arguments

object a fitted model object for which there exists a logLik method to extract the corresponding log-likelihood
... optionally more fitted model objects
saturated saturated model log likelihood reference value (use 0 if deviance is not available)
sortby either a numeric or character string specifying the column in the result by which the rows are sorted (in decreasing order)

Details

The function relies on residual degrees of freedom for the LR chisq test being available in the model object. This is true for objects inheriting from lm, glm, loglm, polr and negbin.

Value

A data frame (also of class anova) with columns c("AIC","BIC","LR Chisq","Df","Pr(>Chisq)"). Row names are taken from the names of the model object(s).

Author(s)

Achim Zeileis

See Also

logLik, glm, loglm,
logLik.loglm, modFit

Examples

data(Mental)
indep <- glm(Freq ~ mental+ses,
            family = poisson, data = Mental)
LRstats(indep)
Cscore <- as.numeric(Mental$ses)
Rscore <- as.numeric(Mental$mental)
coleff <- glm(Freq ~ mental + ses + Rscore:ses,
             family = poisson, data = Mental)
roweff <- glm(Freq ~ mental + ses + mental:Cscore,
             family = poisson, data = Mental)
linlin <- glm(Freq ~ mental + ses + Rscore:Cscore,
             family = poisson, data = Mental)

# compare models
LRstats(indep, coleff, roweff, linlin)
Mammograms  Mammogram Ratings

Description

Kundel & Polansky (2003) give (possibly contrived) data on a set of 110 mammograms rated by two readers.

Usage

data(Mammograms)

Format


Source


Examples

data(Mammograms)
B <- agreementplot(Mammograms, main="Mammogram ratings") # agreement measures B Kappa(Mammograms)

## other displays
mosaic(Mammograms, shade=TRUE)
sieve(Mammograms, pop = FALSE, shade = TRUE)
labeling_cells(text = Mammograms, gp_text = gpar(fontface = 2, cex=1.75))(as.table(Mammograms))

mcaplot  Simple and enhanced plot of MCA solutions

Description

This function is intended as an alternative to `plot.mjca` for plotting multiple correspondence analysis solutions. It provides more flexibility for labeling factor levels and connecting them with lines. It does not support some features of `plot.mjca` (centroids, supplementary points, arrows, etc.)
Usage

mcaplot(obj, map = "symmetric", dim = 1:2,
        col = c("blue", "red", "brown", "black", "green3", "purple"),
        pch = 15:20, cex = 1.2, pos = 3, lines = TRUE, lwd = 2,
        legend = FALSE, legend.pos = "topright",
        xlab = "_auto_", ylab = "_auto_",
        rev.axes = c(FALSE, FALSE),
        ...)  

Arguments

obj An "mjca" object
map Character string specifying the map type, i.e., the scaling applied to coordinates for different types of MCA representations. Allowed options include: "symmetric" (default), "rowprincipal", "colprincipal", "symbiplot", "rowgab", "colgab", "rowgreen", "colgreen". See mjca for details.
dim Dimensions to plot, an integer vector of length 2
col Vector of colors, one for each factor in the MCA
pch Vector of point symbols for the category levels, one for each factor
cex Character size for points and level labels
pos Position of level labels relative to the category points; either a single number or a vector of length equal to the number of category points.
lines A logical or an integer vector indicating which factors are to be joined with lines using multilines
lwd Line width(s) for the lines
legend Logical; draw a legend for the factor names?
legend.pos Position of the legend in the plot, as in legend
xlab, ylab Labels for horizontal and vertical axes. The default, "_auto_" means that the function auto-generates a label of the form "Dimension X (xx.x %)"
rev.axes A logical vector of length 2, where TRUE reverses the direction of the corresponding axis
... Arguments passed down to plot

Value

Returns the coordinates of the category points invisibly

Author(s)

Michael Friendly

See Also

mjca, plot.mjca
cacoord returns CA and MCA coordinates, multilines draw multiple lines according to a factor,
Examples

```r
require(ca)
data(Titanic)
titanic.mca <- mjca(Titanic)
mcaplot(titanic.mca, legend=TRUE, legend.pos="topleft")

data(HairEyeColor)
haireye.mca <- mjca(HairEyeColor)
mcaplot(haireye.mca, legend=TRUE, cex.lab=1.3)
```

---

**Mental**

<table>
<thead>
<tr>
<th>Mental impairment and parents SES</th>
</tr>
</thead>
</table>

**Description**

A 6 x 4 contingency table representing the cross-classification of mental health status (mental) of 1660 young New York residents by their parents’ socioeconomic status (ses).

**Usage**

data(Mental)

**Format**

A data frame frequency table with 24 observations on the following 3 variables.

- **ses** an ordered factor with levels 1 < 2 < 3 < 4 < 5 < 6
- **mental** an ordered factor with levels Well < Mild < Moderate < Impaired
- **Freq** cell frequency: a numeric vector

**Details**

Both ses and mental can be treated as ordered factors or integer scores. For ses, 1="High" and 6="Low".

**Source**


**References**

Examples

data(Mental)
str(Mental)
(Mental.tab <- xtabs(Freq ~ ses+mental, data=Mental))

# mosaic and sieve plots
mosaic(Mental.tab, gp=shading_Friendly)
sieve(Mental.tab, gp=shading_Friendly)

library(ca)
plot(ca(Mental.tab), main="Mental impairment & SES")
title(xlab="Dim 1", ylab="Dim 2")

Description

Data from Kastenbaum and Lamphiear (1959). The table gives the number of depletions (deaths) in 657 litters of mice, classified by litter size and treatment. This data set has become a classic in the analysis of contingency tables, yet unfortunately little information on the details of the experiment has been published.

Usage

data("Mice")

Format

A frequency data frame with 30 observations on the following 4 variables, representing a 5 x 2 x 3 contingency table.

litter  litter size, a numeric vector
treatment treatment, a factor with levels A B
deaths  number of depletions, a factor with levels 0 1 2+
Freq   cell frequency, a numeric vector

Source


References

Examples

```r
data(Mice)
# make a table
ftable(mice.tab <- xtabs(Freq ~ litter + treatment + deaths, data=Mice))

library(vcd)
mosaic(mice.tab, shade=TRUE)
```

---

### Social Mobility data

**Description**

Data on social mobility, recording the occupational category of fathers and their sons.

**Usage**

```r
data(Mobility)
```

**Format**

A 2-dimensional array resulting from cross-tabulating 2 variables for 19912 observations. The variable names and their levels are:

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Son's_Occupation</td>
<td>\code{&quot;UpNonMan&quot;, &quot;LoNonMan&quot;, &quot;UpManual&quot;, &quot;LoManual&quot;, &quot;Farm&quot;}</td>
</tr>
</tbody>
</table>

**Source**


**Examples**

```r
data(Mobility)

# example goes here
```
modFit

Brief Summary of Model Fit for a glm or loglm Object

Description

Formats a brief summary of model fit for a glm or loglm object, showing the likelihood ratio Chisq (df) value and or AIC. Useful for inclusion in a plot title or annotation.

Usage

modFit(x, ...)
## S3 method for class 'glm'
modFit(x, stats="chisq", digits=2, ...)
## S3 method for class 'loglm'
modFit(x, stats="chisq", digits=2, ...)

Arguments

x A glm or loglm object
...
stats One or more of chisq or aic, determining the statistics displayed.
digits Number of digits after the decimal point in displayed statistics.

Value

A character string containing the formatted values of the chosen statistics.

Author(s)

Michael Friendly

See Also

Summarise (soon to be deprecated), LRstats

Examples

data(Mental)
require(MASS)
(Mental.tab <- xtabs(Freq ~ ses+mental, data=Mental))
(Mental.mod <- loglm(~ses+mental, Mental.tab))
Mental.mod
modFit(Mental.mod)

# use to label mosaic()
mosaic(Mental.mod, main=paste("Independence model,", modFit(Mental.mod)))
Mosaic plots for fitted generalized linear and generalized nonlinear models

Description

Produce mosaic plots (and other plots in the strucplot framework) for a log-linear model fitted with glm or for a generalized nonlinear model fitted with gnm.

These methods extend the range of strucplot visualizations well beyond the models that can be fit with loglm. They are intended for models for counts using the Poisson family (or quasi-poisson), but should be sensible as long as (a) the response variable is non-negative and (b) the predictors visualized in the strucplot are discrete factors.

Usage

## S3 method for class 'glm'
mosaic(x, formula = NULL, panel = mosaic,
       type = c("observed", "expected"),
       residuals = NULL,
       residuals_type = c("pearson", "deviance", "rstandard"),
       gp = shading_hcl, gp_args = list(), ...)

## S3 method for class 'glm'
sieve(x, ...)

## S3 method for class 'glm'
assoc(x, ...)

Arguments

x
A glm or gnm object. The response variable, typically a cell frequency, should be non-negative.

formula
A one-sided formula with the indexing factors of the plot separated by '+', determining the order in which the variables are used in the mosaic. A formula must be provided unless x$data inherits from class "table" – in which case the indexing factors of this table are used, or the factors in x$data (or model.frame(x) if x$data is an environment) exactly cross-classify the data – in which case this set of cross-classifying factors are used.

panel
Panel function used to draw the plot for visualizing the observed values, residuals and expected values. Currently, one of "mosaic", "assoc", or "sieve" in vcd.

type
A character string indicating whether the "observed" or the "expected" values of the table should be visualized by the area of the tiles or bars.

residuals
An optional array or vector of residuals corresponding to the cells in the data, for example, as calculated by residuals.glm(x), residuals.gnm(x).

residuals_type
If the residuals argument is NULL, residuals are calculated internally and used in the display. In this case, residual_type can be "pearson", "deviance" or
Otherwise (when residuals is supplied), residuals_type is used as a label for the legend in the plot.

gp
Object of class "gpar", shading function or a corresponding generating function (see strucplot Details and shadings). Ignored if shade = FALSE.

gp_args
A list of arguments for the shading-generating function, if specified.

...
Other arguments passed to the panel function e.g., mosaic

Details
For both poisson family generalized linear models and loglinear models, standardized residuals provided by rstandard (sometimes called adjusted residuals) are often preferred because they have constant unit asymptotic variance.

The sieve and assoc methods are simple convenience interfaces to this plot method, setting the panel argument accordingly.

Value
The structable visualized by strucplot is returned invisibly.

Author(s)
Heather Turner, Michael Friendly, with help from Achim Zeileis

See Also
glm, gnm, plot.loglm, mosaic

Examples

GSStab <- xtabs(count ~ sex + party, data=GSS)
# using the data in table form
mod.glm1 <- glm(Freq ~ sex + party, family = poisson, data = GSStab)
res <- residuals(mod.glm1)
std <- rstandard(mod.glm1)

# For mosaic.default(), need to re-shape residuals to conform to data
stdtab <- array(std, dim=dim(GSStab), dimnames=dimnames(GSStab))
mosaic(GSStab, gp=shading_Friendly, residuals=stdtab, residuals_type="Std
residuals", labeling = labeling_residuals)

# Using externally calculated residuals with the glm() object
mosaic.glm(mod.glm1, residuals=std, labeling = labeling_residuals, shade=TRUE)

# Using residuals_type
mosaic.glm(mod.glm1, residuals_type="rstandard", labeling = labeling_residuals, shade=TRUE)

## Ordinal factors and structured associations
data(Mental)
xtabs(Freq ~ mental+ses, data=Mental)
long.labels <- list(set_varnames = c(mental="Mental Health Status", ses="Parent SES"))

# fit independence model
# Residual deviance: 47.418 on 15 degrees of freedom
indep <- glm(Freq ~ mental + ses,
             family = poisson, data = Mental)
long.labels <- list(set_varnames = c(mental="Mental Health Status",
                                    ses="Parent SES"))
mosaic(indep, residuals_type="rstandard", labeling_args = long.labels, labeling=labeling_residuals)
# or, show as a sieve diagram
mosaic(indep, labeling_args = long.labels, panel=sieve, gp=shading_Friendly)

# fit linear x linear (uniform) association. Use integer scores for rows/cols
Cscore <- as.numeric(Mental$ses)
Rscore <- as.numeric(Mental$mental)
linlin <- glm(Freq ~ mental + ses + Rscore:Cscore,
              family = poisson, data = Mental)
mosaic(linlin, residuals_type="rstandard",
       labeling_args = long.labels, labeling=labeling_residuals, suppress=1, gp=shading_Friendly,
       main="Lin x Lin model")

## Goodman Row-Column association model fits even better (deviance 3.57, df 8)
if (require(gnm)) {
  Mental$mental <- C(Mental$mental, treatment)
  Mental$ses <- C(Mental$ses, treatment)
  RC1model <- gnm(Freq ~ ses + mental + Mult(ses, mental),
                  family = poisson, data = Mental)
  mosaic(RC1model, residuals_type="rstandard",
         labeling_args = long.labels, labeling=labeling_residuals, suppress=1, gp=shading_Friendly,
         main="RC1 model")
}

############### UCB Admissions data, fit using glm()
structable(Dept ~ Admit+Gender, UCBAdmissions)
berkeley <- as.data.frame(UCBAdmissions)
berk.glm1 <- glm(Freq ~ Dept * (Gender+Admit), data=berkeley, family="poisson")
summary(berk.glm1)
mosaic(berk.glm1, gp=shading_Friendly, labeling=labeling_residuals, formula=-Admit+Dept+Gender)
# the same, displaying studentized residuals; note use of formula to reorder factors in the mosaic
mosaic(berk.glm1, residuals_type="rstandard", labeling=labeling_residuals, shade=TRUE,
       formula=-Admit+Dept+Gender, main="Model: [DeptGender][DeptAdmit]")

## all two-way model
berk.glm2 <- glm(Freq ~ (Dept + Gender + Admit)^2, data=berkeley, family="poisson")
summary(berk.glm2)
mosaic.glm(berk.glm2, residuals_type="rstandard", labeling=labeling_residuals, shade=TRUE,
          formula=-Admit+Dept+Gender, main="Model: [DeptGender][DeptAdmit][AdmitGender]")
anova(berk.glm1, berk.glm2, test="Chisq")
# Add 1 df term for association of [GenderAdmit] only in Dept A
berkeley <- within(berkeley, dept1AG <- (Dept=='A')*(Gender=='Female')*(Admit=='Admitted'))
berkeley[1:6,]
berk glm3 <- glm(Freq ~ Dept * (Gender + Admit) + dept1AG, data=berkeley, family="poisson")
summary(berk glm3)
mosaic glm(berk glm3, residuals_type="rstandard", labeling = labeling_residuals, shade=TRUE,
formula=~Admit+Dept+Gender, main="Model: [DeptGender][DeptAdmit] + DeptA*[GA]"
anova(berk glm1, berk glm3, test="Chisq")

mosaic glmlist

Mosaic Displays for glmlist and logllmlist Objects

Description

This function provides a convenient interface for viewing mosaic displays associated with a collection of glm models for frequency tables that have been stored in a glmlist or loglmlist object. You can plot either selected models individually, or mosaics for all models in an array of viewports.

Usage

## S3 method for class 'glmlist'
mosaic(x, selection,
panel=mosaic,
type=c("observed", "expected"),
legend=ask | !missing(selection),
main=NULL,
ask=TRUE, graphics=TRUE, rows, cols, newpage=TRUE,
...)

## S3 method for class 'loglmlist'
mosaic(x, selection,
panel=mosaic,
type=c("observed", "expected"),
legend=ask | !missing(selection),
main=NULL,
ask=TRUE, graphics=TRUE, rows, cols, newpage=TRUE,
...)

Arguments

- **x**: a glmlist or loglmlist object
- **selection**: the index or name of one glm or loglm object in x. If no selection is specified, a menu of models is presented or all models are plotted.
- **panel**: a strucplot panel function, typically mosaic or sieve
### Details

Most details of the plots produced can be controlled via ...arguments as shown in some of the examples below. In particular, with panel=sieve you need to also pass gp=shading_Friendly to get a color version.

### Value

Returns the result of mosaic.glm.

### Author(s)

Michael Friendly

### References


### See Also

- glmlist, loglmlist, Kway
- mosaic.glm, mosaic, strucplot, for the many parameters that control the details of mosaic plots.

### Examples

```r
data(JobSatisfaction, package="vcd")

# view all pairwise mosaics
pairs(xtabs(Freq~management+supervisor+own, data=JobSatisfaction),
     shade=TRUE, diag_panel=pairs_diagonal_mosaic)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>a character string indicating whether the “observed” or the “expected” values of the table should be visualized</td>
</tr>
<tr>
<td>legend</td>
<td>logical: show a legend for residuals in the mosaic display(s)? The default behavior is to include a legend when only a single plot is shown, i.e., if ask is TRUE or a selection has been specified.</td>
</tr>
<tr>
<td>main</td>
<td>either a logical, or a vector of character strings used for plotting the main title. If main is a logical and TRUE, the name of the selected glm object is used.</td>
</tr>
<tr>
<td>ask</td>
<td>logical: should the function display a menu of models, when one is not specified in selection? If selection is not supplied and ask is TRUE (the default), a menu of model names is presented; if ask is FALSE, mosaics for all models are plotted in an array.</td>
</tr>
<tr>
<td>graphics</td>
<td>logical: use a graphic dialog box when ask=TRUE?</td>
</tr>
<tr>
<td>rows, cols</td>
<td>when ask=FALSE, the number of rows and columns in which to plot the mosaics.</td>
</tr>
<tr>
<td>newpage</td>
<td>start a new page? (only applies to ask=FALSE)</td>
</tr>
<tr>
<td>...</td>
<td>other arguments passed to mosaic.glm and ultimately to mosaic.</td>
</tr>
</tbody>
</table>
modSat <- Kway(Freq ~ management+supervisor+own, data=JobSatisfaction, family=poisson, prefix="JobSat")

names(modSat)

## Not run:
mosaic(modSat) # uses menu, if interactive()

## End(Not run)
mosaic(modSat, "JobSat.1") # model label
mosaic(modSat, 2) # model index

# supply a formula to determine the order of variables in the mosaic
mosaic(modSat, 2, formula=~own+supervisor+management)

mosaic(modSat, ask=FALSE) # uses viewports

# use a different panel function, label the observed valued in the cells
mosaic(modSat, 1, main=TRUE, panel=sieve, gp=shading_Friendly, labeling=labeling_values)

data(Mental)

indep <- glm(Freq ~ mental+ses, family = poisson, data = Mental)

Cscore <- as.numeric(Mental$ses)

Rscore <- as.numeric(Mental$mental)

coleff <- glm(Freq ~ mental + ses + Rscore:ses, family = poisson, data = Mental)

roveff <- glm(Freq ~ mental + ses + mental:Cscore, family = poisson, data = Mental)

linlin <- glm(Freq ~ mental + ses + Rscore:Cscore, family = poisson, data = Mental)

# assign names for the plot labels
modMental <- glmlist(Indep=indep, ColEff=coleff, RowEff=roveff, `Lin x Lin`=linlin)

mosaic(modMental, ask=FALSE, margins=c(3,1,1,2), labeling_args=list(abbreviate_labs=5))

---

**mosaic3d**

### 3D Mosaic Plots

**Description**

Produces a 3D mosaic plot for a contingency table (or a `link[MASS]{loglm}` model) using the **rgl-package**.

Generalizing the 2D mosaic plot, this begins with a given 3D shape (a unit cube), and successively sub-divides it along the X, Y, Z dimensions according to the table margins, generating a nested set of 3D tiles. The volume of the resulting tiles is therefore proportional to the frequency represented.
in the table cells. Residuals from a given loglinear model are then used to color or shade each of
the tiles.
This is a developing implementation. The arguments and details are subject to change.

Usage

```r
mosaic3d(x, ...)
```

## S3 method for class 'loglm'
mosaic3d(x, type = c("observed", "expected"),
  residuals_type = c("pearson", "deviance"), ...)

## Default S3 method:
mosaic3d(x, expected = NULL, residuals = NULL,
  type = c("observed", "expected"), residuals_type = NULL,
  shape = rgl::cube3d(alpha = alpha), alpha = 0.5,
  spacing = 0.1, split_dir = 1:3, shading = shading_basic, interpolate=c(2,4),
  zero_size=.05,
  label_edge,
  labeling_args = list(), newpage = TRUE, box=FALSE, ...)

Arguments

- `x`: A `link[MASS]{loglm}` model object. Alternatively, a multidimensional array or table or `structable` of frequencies in a contingency table. In the present implementation, the dimensions are taken in sequential order. Use `link[base]{aperm}` or `structable` to change this.
- `expected`: optionally, for contingency tables, an array of expected frequencies of the same dimension as `x`, or alternatively the corresponding loglinear model specification as used by `link[stats]{loglin}` or `link[MASS]{loglm}` (see `structable` for details).
- `residuals`: optionally, an array of residuals of the same dimension as `x` (see details).
- `type`: a character string indicating whether the "observed" or the "expected" frequencies in the table should be visualized by the volume of the 3D tiles.
- `residuals_type`: a character string indicating the type of residuals to be computed when none are supplied. If residuals is NULL, `residuals_type` must be one of "pearson" (default; giving components of Pearson’s chi-squared), "deviance" (giving components of the likelihood ratio chi-squared), or "FT" for the Freeman-Tukey residuals. The value of this argument can be abbreviated.
- `shape`: The initial 3D shape on which the mosaic is based. Typically this is a call to an `rgl` function, and must produce a `shape3d` object. The default is a "unit cube" on (-1, +1), with transparency specified by `alpha`.
- `alpha`: Specifies the transparency of the 3D tiles used to compose the 3D mosaic.
- `spacing`: A number or vector giving the total amount of space used to separate the 3D tiles along each of the dimensions of the table. The values specified are re-cycled to the number of table dimensions.
split_dir A numeric vector composed of the integers 1:3 or a character vector composed of c("x","y","z"), where split_dir[i] specifies the axis along which the tiles should be split for dimension i of the table. The values specified are recycled to the number of table dimensions.

shading A function, taking an array or vector of residuals for the given model, returning a vector of colors. At present, only the default shading=shading_basic is provided. This is roughly equivalent to the use of the shade argument in mosaicplot or to the use of gp=shading_Friendly in mosaic.

interpolate a vector of interpolation values for the shading function.

zero_size The radius of a small sphere used to mark zero cells in the display.

label_edge A character vector composed of c("-","+") indicating whether the labels for a given table dimension are to be written at the minima ("-"), or maxima ("+") of the other dimensions in the plot. The default is rep( c('-','+'),each=3,length=ndim), meaning that the first three table variables are labeled at the minima, and successive ones at the maxima.

labeling_args This argument is intended to be used to specify details of the rendering of labels for the table dimensions, but at present has no effect.

newpage logical indicating whether a new page should be created for the plot or not.

box logical indicating whether a bounding box should be drawn around the plot.

... Other arguments passed down to mosaic.default or 3D functions.

Details

Friendly (1995), Friendly [Sect. 4.5](2000) and Theus and Lauer (1999) have all used the idea of 3D mosaic displays to explain various aspects of loglinear models (the iterative proportional fitting algorithm, the structure of various models for 3-way and n-way tables, etc.), but no implementation of 3D mosaics was previously available.

For the default method, residuals, used to color and shade the 3D tiles, can be passed explicitly, or, more typically, are computed as needed from observed and expected frequencies. In this case, the expected frequencies are optionally computed for a specified loglinear model given by the expected argument. For the loglm method, residuals and observed frequencies are calculated from the model object.

Value

Invisibly, the list of shape3d objects used to draw the 3D mosaic, with names corresponding to the concatenation of the level labels, separated by ":".

Author(s)

Michael Friendly, with the help of Duncan Murdoch and Achim Zeileis

References


See Also

`strucplot, mosaic, mosaicplot`

`loglin, loglm` for details on fitting loglinear models

Examples

```r
# 2 x 2 x 2
if(requireNamespace("rgl")){
  mosaic3d(Bartlett, box=TRUE)
  # compare with expected frequencies under model of mutual independence
  mosaic3d(Bartlett, type="expected", box=TRUE)

  # 2 x 2 x 3
  mosaic3d(Heart, box=TRUE)
}

## Not run:
# 2 x 2 x 2 x 3
# illustrates a 4D table
mosaic3d(Detergent)

# compare 2D and 3D mosaics
demo("mosaic-hec")

## End(Not run)
```

---

**PhdPubs**

*Publications of PhD Candidates*

### Description

A data set giving the number of publications by doctoral candidates in biochemistry in relation to various predictors, originally from Long (1997).

There is a large number of zero counts. Is there evidence for a separate group of non-publishers?

### Usage

```r
data(PhdPubs)
```
Format

A data frame with 915 observations on the following 6 variables.

- articles: number of articles published in the final three years of PhD studies
- female: dummy variable for gender, coded 1 for female
- married: dummy variable for marital status, coded 1 for married
- kid5: number of young children, age 5 and under
- phdprestige: prestige of the PhD department. The higher the number the more prestigious the program.
- mentor: number of publications by the mentor in the preceding three years

Details

In this version of the data set, phdprestige had been rounded to the nearest integer. A Stata version with the continuous values was subsequently found at https://www.stata-press.com/data/lf2/couart2.dta

Source


Examples

data(PhdPubs)
# very uninformative
hist(PhdPubs$articles, breaks=0:19, col="pink", xlim=c(0,20),
    xlab="Number of Articles")

library(vcd)
rootogram(goodfit(PhdPubs$articles), xlab="Number of Articles")
# compare with negative binomial
rootogram(goodfit(PhdPubs$articles, type="nbinomial"),
    xlab="Number of Articles", main="Negative binomial")

Description

This is a replacement for the print.Kappa method in vcd, adding display of z values to the vcd version and optional confidence intervals.
Usage

## S3 method for class 'Kappa'
print(x, digits=max(getOption("digits") - 3, 3), CI=FALSE, level=0.95, ...)

Arguments

  x       A Kappa object
  digits  number of digits to print
  CI      Include confidence intervals in the display?
  level   confidence level
  ...     Other arguments

Value

  Returns the Kappa object, invisibly.

Author(s)

  Michael Friendly

See Also

  confint.Kappa

Examples

data("SexualFun")
Kappa(SexualFun)
print(Kappa(SexualFun), CI=TRUE)

# stratified 3-way table
apply(MSPatients, 3, Kappa)

seq_loglm

Sequential Loglinear Models for an N-way Table

Description

  This function takes an n-way contingency table and fits a series of sequential models to the 1-, 2-, ...
  n-way marginal tables, corresponding to a variety of types of loglinear models.

Usage

  seq_loglm(x,
            type = c("joint", "conditional", "mutual", "markov", "saturated"),
            marginals = 1:nf, vorder = 1:nf,
            k = NULL, prefix = "model", fitted = TRUE, ...)
Arguments

x  
a contingency table in array form, with optional category labels specified in the dimnames(x) attribute, or else a data.frame in frequency form, with the frequency variable named “Freq”.

type  
type of sequential model to fit, a character string. One of "joint", "conditional", "mutual", "markov", or "saturated".

marginals  
which marginal sub-tables to fit? A vector of a (sub)set of the integers, 1:nf where nf is the number of factors in the full n-way table.

vorder  
order of variables, a permutation of the integers 1:nf, used to reorder the variables in the original table for the purpose of fitting sequential marginal models.

k  
conditioning variable(s) for type = "joint", "conditional" or Markov chain order for type = "markov"

prefix  
prefix used to give names to the sequential models

fitted  
argument passed to loglm to store the fitted values in the model objects

...  
other arguments, passed down

Details

Sequential marginal models for an n-way tables begin with the model of equal-probability for the one-way margin (equivalent to a chisq.test) and add successive variables one at a time in the order specified by vorder.

All model types give the same result for the two-way margin, namely the test of independence for the first two factors.

Sequential models of joint independence (type="joint") have a particularly simple interpretation, because they decompose the likelihood ratio test for the model of mutual independence in the full n-way table, and hence account for "total" association in terms of portions attributable to the conditional probabilities of each new variable, given all prior variables.

Value

An object of class "loglmlist", each of which is a class "loglm" object

Note

One-way marginal tables are a bit of a problem here, because they cannot be fit directly using loglm. The present version uses loglin, and repairs the result to look like a loglm object (sort of).

Author(s)

Michael Friendly

References

These functions were inspired by the original SAS implementation of mosaic displays, described in the User’s Guide, http://www.datavis.ca/mosaics/mosaics.pdf
seq_mosaic

See Also

loglin-utilities for descriptions of sequential models, conditional, joint, mutual, ...
loglmlist,

Examples

data(Titanic, package="datasets")
# variables are in the order Class, Sex, Age, Survived
tt <- seq_loglm(Titanic)

seq_mosaic

Sequential Mosaics and Strucplots for an N-way Table

Description

This function takes an n-way contingency table and plots mosaics for series of sequential models to
the 1-, 2-, ... n-way marginal tables, corresponding to a variety of types of loglinear models.

Usage

seq_mosaic(x, panel = mosaic,
  type = c("joint", "conditional", "mutual", "markov", "saturated"),
  plots = 1:nf, vorder = 1:nf,
  k = NULL, ...)

Arguments

x a contingency table in array form, with optional category labels specified in
  the dimnames(x) attribute, or else a data.frame in frequency form, with the fre-
  quency variable named "Freq".
panel a strucplot panel function, typically mosaic or sieve. NOT yet implemented.
type type of sequential model to fit, a character string. One of "joint", "conditional",
  "mutual", "markov", or "saturated".
plots which marginal sub-tables to plot? A vector of a (sub)set of the integers, 1:nf
  where nf is the number of factors in the full n-way table.
vorder order of variables, a permutation of the integers 1:nf, used to reorder the vari-
  ables in the original table for the purpose of fitting sequential marginal models.
k conditioning variable(s) for type = "joint", "conditional" or Markov chain
  order for type = "markov"
... other arguments passed to mosaic.
Details

This function produces similar plots to the use of `mosaic.loglmlist`, called with the result of `seq_loglm`.

Value

None. Used for its side-effect of producing plots

Author(s)

Michael Friendly

References

These functions were inspired by the original SAS implementation of mosaic displays, described in the User's Guide, http://www.datavis.ca/mosaics/mosaics.pdf

See Also

`loglin-utilities` for descriptions of sequential models, `conditional`, `joint`, `mutual`, ...

`loglmlist`, `mosaic.loglmlist`, `seq_loglm`

`mosaic.glm`, `mosaic`, `strucplot`, for the many parameters that control the details of mosaic plots.

Examples

data(Titanic, package="datasets")

seq_mosaic(Titanic)  # models of joint independence, Survived last
seq_mosaic(Titanic, type="condit")
seq_mosaic(Titanic, type="mutual")

# other panel functions and options: presently BUGGED
## Not run:
seq_mosaic(Titanic, type="mutual", panel=sieve,
  gp=shading_FRIENDLY, labeling=labeling_values)

## End(Not run)
Description

This data set, from Efron and Thisted (1976), gives the number of distinct words types (Freq) of words that appeared exactly once, twice, etc. up to 100 times (count) in the complete works of Shakespeare. In these works, Shakespeare used 31,534 distinct words (types), comprising 884,647 words in total.

Efron & Thisted used this data to ask the question, "How many words did Shakespeare know?" Put another way, suppose another new corpus of works Shakespeare were discovered, also with 884,647 words. How many new word types would appear? The answer to the main question involves contemplating an infinite number of such new corpora.

Usage

data(ShakeWords)

Format

A data frame with 100 observations on the following 2 variables.

count  the number of times a word type appeared in Shakespeare’s written works
Freq   the number of different words (types) appearing with this count.

Details

In addition to the words that appear 1:100 times, there are 846 words that appear more than 100 times, not listed in this data set.

Source


Examples

data(ShakeWords)

## maybe str(ShakeWords) ; plot(ShakeWords) ...

---

split3d  Subdivide a 3D Object

Description

Subdivides a shape3d object or a list of shape3d objects into objects of the same shape along a given dimension according to the proportions or frequencies specified in vector(s).

split3d is the basic workhorse used in mosaic3d, but may be useful in other contexts.

range3d and center3d are utility functions, also useful in other contexts.
split3d

Usage

split3d(obj, ...)

## S3 method for class 'shape3d'
split3d(obj, p, dim, space = 0.1, ...)

## S3 method for class 'list'
split3d(obj, p, dim, space = 0.1, ...)

range3d(obj)

center3d(obj)

Arguments

obj A shape3d object, or a list composed of them

... Other arguments for split3d methods

p For a single shade3d object, a vector of proportions (or a vector of non-negative numbers which will be normed to proportions) indicating the number of subdivisions and their scaling along dimension dim. For a list of shade3d objects, a matrix whose columns indicate the subdivisions of each object.

dim The dimension along which the object is to be subdivided. Either an integer: 1, 2, or 3, or a character: "x", "y", or "z".

space The total space used to separate the copies of the object along dimension dim. The unit inter-object space is therefore space/(length(p)-1).

Details

The resulting list of shape3d objects is actually composed of copies of the input object(s), scaled according to the proportions in p and then translated to make their range along the splitting dimension equal to that of the input object(s).

Value

split3d returns a list of shape3d objects.

range3d returns a 2 x 3 matrix, whose first row contains the minima on dimensions x, y, z, and whose second row contains the maxima.

center3d returns a numeric vector containing the means of the minima and maxima on dimensions x, y, z.

Author(s)

Duncan Murdoch, with refinements by Michael Friendly
Summarise

Brief Summary of Model Fit for glm and loglm Models

Description

For glm objects, the print and summary methods give too much information if all one wants to see is a brief summary of model goodness of fit, and there is no easy way to display a compact comparison of model goodness of fit for a collection of models fit to the same data. All loglm models have equivalent glm forms, but the print and summary methods give quite different results.

Summarise provides a brief summary for one or more models fit to the same dataset for which logLik and nobs methods exist (e.g., glm and loglm models).

Usage

Summarise(object, ...)

## S3 method for class 'glmlist'
Summarise(object, ..., saturated = NULL, sortby = NULL)

## S3 method for class 'loglmlist'
Summarise(object, ..., saturated = NULL, sortby = NULL)

## Default S3 method:
Summarise(object, ..., saturated = NULL, sortby = NULL)

Arguments

object a fitted model object for which there exists a logLik method to extract the corresponding log-likelihood

... optionally more fitted model objects

See Also

mosaic3d
shapelist3d for the plotting of lists of shape3d objects.

Examples

if (require(rgl)) {
  open3d()
  cube <- cube3d(alpha=0.4)
  sl1 <- split3d(cube, c(0.2, 0.3, 0.5), 1)
  col <- c("#FF000080", "#E5E5E580", "#0000FF80")
  shapelist3d(sl1, col=col)

  open3d()
  p <- matrix(c(0.6, 0.4, 0.5, 0.5, 0.2, 0.8), nrow=2)
  sl2 <- split3d(sl1, p, 2)
  shapelist3d(sl2, col=col)
}
saturated  saturated model log likelihood reference value (use 0 if deviance is not available)
sortby   either a numeric or character string specifying the column in the result by which
         the rows are sorted (in decreasing order)

Details

The function relies on residual degrees of freedom for the LR chisq test being available in the model
object. This is true for objects inheriting from `lm`, `glm`, `loglm`, `polr` and `negbin`.

Value

A data frame (also of class `anova`) with columns `c("AIC","BIC","LR Chisq","Df","Pr(>Chisq)")`.  
Row names are taken from the names of the model object(s).

Author(s)

Achim Zeileis

See Also

`logLik`, `glm`, `loglm`, `logLik.loglm`, `modFit`

Examples

data(Mental)
  indep <- glm(Freq ~ mental+ses,
              family = poisson, data = Mental)
  Summarise(indep)
  Cscore <- as.numeric(Mental$ses)
  Rscore <- as.numeric(Mental$mental)

  coleff <- glm(Freq ~ mental + ses + Rscore:ses,
                family = poisson, data = Mental)
  roweff <- glm(Freq ~ mental + ses + mental:Cscore,
                family = poisson, data = Mental)
  linlin <- glm(Freq ~ mental + ses + Rscore:Cscore,
                family = poisson, data = Mental)

  # compare models
  Summarise(indep, coleff, roweff, linlin)
Passengers on the Titanic

Description
Data on passengers on the RMS Titanic, excluding the Crew and some individual identifier variables.

Usage
data(Titanicp)

Format
A data frame with 1309 observations on the following 6 variables.

pclass a factor with levels 1st 2nd 3rd
survived a factor with levels died survived
sex a factor with levels female male
age passenger age in years (or fractions of a year, for children), a numeric vector; age is missing for 263 of the passengers
sibsp number of siblings or spouses aboard, integer: 0:8
parch number of parents or children aboard, integer: 0:6

Details
There are a number of related versions of the Titanic data, in various formats. This version was derived from ptitanic in the rpart.plot package, modifying it to remove the Class 'labelled' attributes for some variables (inherited from Frank Harrell's titanic3 version) which caused problems with some applications, notably ggplot2.

Other versions:
Titanic is the 4-way frequency table of all 2201 people aboard the Titanic, including passengers and crew.

Source
The original R source for this dataset was compiled by Frank Harrell and Robert Dawson: https://biostat.app.vumc.org/wiki/pub/Main/DataSets/titanic.html, described in more detail in https://biostat.app.vumc.org/wiki/pub/Main/DataSets/titanic3info.txt

For this version of the Titanic data, passenger details were deleted, survived was cast as a factor, and the name changed to Titanicp to minimize confusion with other versions.

Examples
data(Titanicp)
## maybe str(Titanicp) ; plot(Titanicp) ...
Toxaemia

Toxaemia Symptoms in Pregnancy

Description

Brown et al (1983) gave these data on two signs of toxaemia, an abnormal condition during pregnancy characterized by high blood pressure (hypertension) and high levels of protein in the urine. If untreated, both the mother and baby are at risk of complications or death.

The data frame Toxaemia represents 13384 expectant mothers in Bradford, England in their first pregnancy, who were also classified according to social class and the number of cigarettes smoked per day.

Usage

data(Toxaemia)

Format

A data frame in frequency form representing a 5 x 3 x 2 x 2 contingency table, with 60 observations on the following 5 variables.

- **class**: Social class of mother, a factor with levels 1 2 3 4 5
- **smoke**: Cigarettes smoked per day during pregnancy, a factor with levels 0 1-19 20+
- **hyper**: Hypertension level, a factor with levels Low High
- **urea**: Protein urea level, a factor with levels Low High
- **Freq**: frequency in each cell, a numeric vector

Source


References


Examples

data(Toxaemia)

tox.tab <- xtabs(Freq~class+smoke+hyper+urea,Toxaemia)
ftable(tox.tab, row.vars=1)
# symptoms by smoking
mosaic(~smoke+hyper+urea, data=tox.tab, shade=TRUE)

# symptoms by social class
mosaic(~class+hyper+urea, data=tox.tab, shade=TRUE)

# predictors
mosaic(~smoke+class, data=tox.tab, shade=TRUE)

# responses
mosaic(~hyper+urea, data=tox.tab, shade=TRUE)

# log odds ratios for urea and hypertension, by class and smoke
## Not run:
LOR <- loddsratio(aperm(tox.tab))
LOR
## End(Not run)

---

**TV Viewing Data**

**Description**

This data set TV comprises a 5 x 11 x 3 contingency table based on audience viewing data from Neilsen Media Research for the week starting November 6, 1995.

**Usage**

data(TV)

**Format**

A 5 x 11 x 3 array of cell frequencies with the following structure:

```r
int [1:5, 1:11, 1:3] 146 244 233 174 294 151 181 161 183 281 ...
- attr(*, "dimnames")=List of 3
  ..$ Day : chr [1:5] "Monday" "Tuesday" "Wednesday" "Thursday" ...
  ..$ Time : chr [1:11] "8:00" "8:15" "8:30" "8:45" ...
  ..$ Network: chr [1:3] "ABC" "CBS" "NBC"
```

**Details**

The original data, tv.dat, contains two additional networks: "Fox" and "Other", with small frequencies. These levels were removed in the current version. There is also a fourth factor, transition State transition (turn the television Off, Switch channels, or Persist in viewing the current channel). The TV data here includes only the Persist observations.
Source

The original data, tv.dat, came from the initial implementation of mosaic displays in R by Jay Emerson (1998). Similar data had been used by Hartigan and Kleiner (1984) as an illustration.

References


Examples

data(TV)
structable(TV)
doubledecker(TV)

# reduce number of levels of Time
TV.df <- as.data.frame.table(TV)
levels(TV.df$Time) <- rep(c("8:00-8:59", "9:00-9:59", "10:00-10:44"), c(4, 4, 3))
TV2 <- xtabs(Freq ~ Day + Time + Network, TV.df)

# re-label for mosaic display
levels(TV.df$Time) <- c("8", "9", "10")
# fit mode of joint independence, showing association of Network with Day*Time
mosaic(~ Day + Network + Time, data = TV.df, expected = ~ Day:Time + Network, legend = FALSE)
# with doubledecker arrangement
mosaic(~ Day + Network + Time, data = TV.df, expected = ~ Day:Time + Network,
      split = c(TRUE, TRUE, FALSE), spacing = spacing_highlighting, legend = FALSE)

update.xtabs

Update method for a xtabs object

Description

Provides an update method for "xtabs" objects, typically by removing terms from the formula to collapse over them.

Usage

## S3 method for class 'xtabs'
update(object, formula., ..., evaluate = TRUE)
Arguments

- **object**: An existing "xtabs" object
- **formula**: Changes to the formula? see `update.formula` for details
- ... Additional arguments to the call, or arguments with changed values.
- **evaluate**: If TRUE, evaluate the new call else return the call

Value

If `evaluate == TRUE`, the new "xtabs" object, otherwise the updated call

Author(s)

Michael Friendly

See Also

- `update.formula` for details on updates to model formulae
- `margin.table` does something similar, `collapse.table` collapses category levels

Examples

```r
vietnam.tab <- xtabs(Freq ~ sex + year + response, data=Vietnam)
update(vietnam.tab, formula = ~ . -year)
```

---

**vcdExtra-deprecated**  
*Deprecated Functions in vcdExtra Package*

Description

These functions are provided for compatibility with older versions of the vcdExtra package only. They are replaced by LRstats.

Usage

```r
summarise(...)
```

Arguments

... pass arguments down.
Details

summarise.* have been replaced by LRstats functions.

---

**Vietnam**

**Student Opinion about the Vietnam War**

**Description**

A survey of student opinion on the Vietnam War was taken at the University of North Carolina at Chapel Hill in May 1967 and published in the student newspaper. Students were asked to fill in ballot papers stating which policy out of A, B, C or D they supported. Responses were cross-classified by gender/year.

The response categories were:

- A: Defeat North Vietnam by widespread bombing and land invasion
- B: Maintain the present policy
- C: De-escalate military activity, stop bombing and begin negotiations
- D: Withdraw military forces Immediately

**Usage**

data(Vietnam)

**Format**

A frequency data frame with 40 observations representing a 2 x 5 x 4 contingency table on the following 4 variables.

- sex: a factor with levels Female Male
- year: year of study, an ordered factor with levels Freshmen, Sophomore, Junior, Senior, Grad student
- response: a factor with levels A B C D
- Freq: cell frequency, a numeric vector

**Details**

For some analyses, it is useful to treat year as numeric, and possibly assign grad students a value year=7.

**Source**

Aitken, M. etal, 1989, *Statistical Modelling in GLIM*

**References**

Vote1980

Race and Politics in the 1980 Presidential Vote

Description

Data from the 1982 General Social Survey on votes in the 1980 U.S. presidential election in relation to race and political conservatism.

Usage

data(Vote1980)

Format

A frequency data frame representing a 2 x 7 x 2 table, with 28 observations on the following 4 variables.

- race: a factor with levels NonWhite White
- conservatism: a factor with levels 1 2 3 4 5 6 7, 1=most liberal, 7=most conservative
- votefor: a factor with levels Carter Reagan; Carter represents Jimmy Carter or other.
- Freq: a numeric vector

Details

The data contains a number of sampling zeros in the frequencies of NonWhites voting for Ronald Reagan.

Source


References


Examples

data(Vote1980)
fourfold(xtabs(Freq ~ race + votefor + conservatism, data=Vote1980), mfrow=c(2,4))
WorkerSat

Worker Satisfaction Data

Description

Blue collar workers job satisfaction from large scale investigation in Denmark in 1968 (Andersen, 1991).

Usage

data("WorkerSat")

Format

A frequency data frame with 8 observations on the following 4 variables, representing the 2 x 2 x 2 classification of 715 cases.

Manage  Quality of management, an ordered factor with levels bad < good
Super   Supervisor satisfaction, an ordered factor with levels low < high
Worker  Worker job satisfaction, an ordered factor with levels low < high
Freq    a numeric vector

Source

https://online.stat.psu.edu/stat504/lesson/10/10.2/10.2.9

References


Examples

data(WorkerSat)

worker.tab <- xtabs(Freq ~ Worker + Super + Manage, data=WorkerSat)
fourfold(worker.tab)
mosaic(worker.tab, shade=TRUE)
Yamaguchi (1987) presented this three-way frequency table, cross-classifying occupational categories of sons and fathers in the United States, United Kingdom and Japan. This data set has become a classic for models comparing two-way mobility tables across layers corresponding to countries, groups or time (e.g., Goodman and Hout, 1998; Xie, 1992).

The US data were derived from the 1973 OCG-II survey; those for the UK from the 1972 Oxford Social Mobility Survey; those for Japan came from the 1975 Social Stratification and Mobility survey. They pertain to men aged 20-64.

Usage

data(Yamaguchi87)

Format

A frequency data frame with 75 observations on the following 4 variables. The total sample size is 28887.

Son  a factor with levels UpNM LoNM UpM LoM Farm
Father a factor with levels UpNM LoNM UpM LoM Farm
Country a factor with levels US UK Japan
Freq  a numeric vector

Details

Five status categories – upper and lower nonmanuals (UpNM, LoNM), upper and lower manuals (UpM, LoM), and Farm are used for both fathers’ occupations and sons’ occupations.

Upper nonmanuals are professionals, managers, and officials; lower nonmanuals are proprietors, sales workers, and clerical workers; upper manuals are skilled workers; lower manuals are semiskilled and unskilled nonfarm workers; and farm workers are farmers and farm laborers.

Some of the models from Xie (1992), Table 1, are fit in demo(yamaguchi-xie).

Source

References


Examples

data(Yamaguchi87)
# reproduce Table 1
structable(~ Father + Son + Country, Yamaguchi87)
# create table form
Yama.tab <- xtabs(Freq ~ Son + Father + Country, data=Yamaguchi87)

# define mosaic labeling_args for convenient reuse in 3-way displays
largs <- list(rot_labels=c(right=0), offset_varnames = c(right = 0.6),
            offset_labels = c(right = 0.2),
            set_varnames = c(Son="Son's status", Father="Father's status")
)

# Fit some models & display mosaics
yama.indep <- glm(Freq ~ Son + Father + Country, data=Yamaguchi87, family=poisson)
anova(yama.indep)
mosaic(yama.indep, ~Son+Father, main="[S][F] ignoring country")
mosaic(yama.indep, ~Country + Son + Father, condvars="Country",
      labeling_args=largs,
      main="[S][F][C] Mutual independence'")

# no association between S and F given country ('perfect mobility')
# asserts same associations for all countries
yama.noRC <- glm(Freq ~ (Son + Father) * Country, data=Yamaguchi87, family=poisson)
anova(yama.noRC)
mosaic(yama.noRC, ~Country + Son + Father, condvars="Country",
      labeling_args=largs,
      main="[SC][FC] No [SF] (perfect mobility)"
)

# ignore diagonal cells
yama.quasi <- update(yama.noRC, ~ . + Diag(Son,Father):Country)
anova(yama.quasi)
mosaic(yama.quasi, ~Son+Father, main="Quasi [S][F]"

## see also:
# demo(yamaguchi-xie)
##
Description

Carries out a simple score test (van den Broek, 1995) for excess zeros in an otherwise Poisson distribution of counts. It gives a $\chi^2_1$ statistic on one degree of freedom.

Usage

`zero.test(x)`

Arguments

- `x` A vector of non-negative counts, or a one-way frequency table of such counts.

Details

The test first calculates the rate estimate from the mean, $\hat{\lambda} = \bar{x}$. The number of observed zeros, $n_0$, is then compared with the expected number, $n\hat{p}_0$, where $\hat{p}_0 = \exp[-\hat{\lambda}]$. Then the test statistic is calculated by the formula:

$$\frac{(n_0 - n\hat{p}_0)^2}{n\hat{p}_0(1 - \hat{p}_0) - n\bar{x}\hat{p}_0}$$

This test statistic has a $\chi^2_1$ distribution.

Value

Returns invisibly a list of three elements:

- `statistic` Description of ‘comp1’
- `df` Description of ‘comp2’
- `pvalue` Upper tail p-value

Author(s)

Michael Friendly

References

The original R code came from a Stackexchange question, [https://stats.stackexchange.com/questions/118322/how-to-test-for-zero-inflation-in-a-dataset](https://stats.stackexchange.com/questions/118322/how-to-test-for-zero-inflation-in-a-dataset)


Examples

# synthetic tests
zero.test(rpois(100, 1))
zero.test(rpois(100, 5))
# add some extra zeros
zero.test(c(rep(0, 20), rpois(100, 5)))

# Articles by Phd candidates
data(PhdPubs, package="vcdExtra")
zero.test(PhdPubs$articles)

phd.tab <- table(PhdPubs$articles)
zero.test(phd.tab)
Index

* array
  expand.dft, 39
* attribute
  collapse.table, 20
* category
  GKgamma, 44
  print.Kappa, 87
* datasets
  Abortion, 5
  Accident, 6
  AirCrash, 8
  Alligator, 10
  Bartlett, 11
  Burt, 13
  Caesar, 14
  Cancer, 16
  Cormorants, 22
  CyclingDeaths, 27
  DaytonSurvey, 29
  Depends, 30
  Detergent, 31
  Donner, 32
  Draft1970, 34
  Draft1970table, 35
  Dyke, 37
  Fungicide, 40
  Geissler, 41
  Gilby, 43
  Glass, 45
  GSS, 48
  HairEyePlace, 49
  Hauser79, 50
  Heart, 52
  Heckman, 53
  HospVisits, 56
  Hoyt, 57
  ICU, 59
  JobSat, 61
  Mammograms, 72
  Mental, 74
  Mice, 75
  Mobility, 76
  PhDinPubs, 86
  ShakeWords, 91
  Titanicp, 96
  Toxaemia, 97
  TV, 98
  Vietnam, 101
  Vote1980, 102
  WorkerSat, 103
  Yamaguchi87, 104
* data
  datasets, 28
* distribution
  logseries, 68
* dplot
  split3d, 92
* hplots
  seq_mosaic, 90
* hplot
  mcaplot, 72
  mosaic.glm, 78
  mosaic.glmlist, 81
  mosaic3d, 83
* htest
  CMHtest, 17
  GKgamma, 44
  HLtest, 54
  logLik.loglm, 64
  print.Kappa, 87
  zero.test, 106
* manip
  blogits, 12
  collapse.table, 20
  Crossings, 24
  cutfac, 25
  expand.dft, 39
* models
<table>
<thead>
<tr>
<th>Crossings</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>glmlist</td>
<td>46</td>
</tr>
<tr>
<td>Kway</td>
<td>62</td>
</tr>
<tr>
<td>logLik.loglm</td>
<td>64</td>
</tr>
<tr>
<td>loglin-utilities</td>
<td>65</td>
</tr>
<tr>
<td>LRstats</td>
<td>70</td>
</tr>
<tr>
<td>modFit</td>
<td>77</td>
</tr>
<tr>
<td>mosaic.glml</td>
<td>78</td>
</tr>
<tr>
<td>seq_loglm</td>
<td>88</td>
</tr>
<tr>
<td>Summarise</td>
<td>94</td>
</tr>
<tr>
<td>update.xtabs</td>
<td>99</td>
</tr>
</tbody>
</table>

* multivariate
  - mosaic.glml, 78

* package
  - datasets, 28
  - vcdExtra-package, 3

* utilities
  - glmlist, 46
  - modFit, 77

<table>
<thead>
<tr>
<th>Abortion</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accident</td>
<td>6</td>
</tr>
<tr>
<td>AIC</td>
<td>64, 65</td>
</tr>
<tr>
<td>AirCrash</td>
<td>8</td>
</tr>
<tr>
<td>Alligator</td>
<td>10</td>
</tr>
<tr>
<td>anova</td>
<td>62</td>
</tr>
<tr>
<td>anova.glm</td>
<td>46, 47</td>
</tr>
<tr>
<td>array</td>
<td>20</td>
</tr>
<tr>
<td>as.data.frame</td>
<td>63</td>
</tr>
<tr>
<td>assoc.glm</td>
<td>78</td>
</tr>
<tr>
<td>assocstats</td>
<td>43</td>
</tr>
</tbody>
</table>

| Bartlett        | 11  |
| BIC             | 64, 65 |
| blogits         | 12  |
| Burt            | 13  |

| cacoord         | 73  |
| Caesar          | 14  |
| caesar          | 15  |
| Cancer          | 16  |
| case2001        | 33  |
| center3d        | 92  |
| chisq.test      | 89  |
| cmh_test        | 19  |
| CMHtest         | 17  |
| Cochran Mantel Haenszel test (CMHtest) | 17 |
| coef.glmlist    | 46  |
| collapse.table  | 20, 100 |
| conditional     | 90, 91 |
| conditional (loglin-utilities) | 65 |
| confint.Kappa   | 88  |
| Cormorants      | 22  |
| Crossings       | 4, 24 |
| cut             | 25, 26 |
| cutfac          | 25  |
| CyclingDeaths   | 27  |
| data            | 28, 29 |
| datasets        | 28  |
| DaytonSurvey    | 29  |
| Depends         | 30  |
| Detergent       | 31  |
| Diag            | 4, 24 |
| Distributions   | 69  |
| dlogseries (logseries) | 68 |
| Donner          | 32  |
| Draft1970       | 34, 36 |
| Draft1970table  | 35, 35 |
| Dyke            | 37  |
| expand.dft      | 21, 39 |
| expand.table    | expand.dft, 39 |
| expandCategorical | 40 |
| factor          | 25  |
| family          | 63  |
| ftable          | 20  |
| Fungicide       | 40  |
| Geissler        | 41  |
| Gilby           | 43  |
| GKgamma         | 44  |
| Glass           | 45  |
| glm             | 24, 64, 71, 78, 79, 95 |
| glmlist         | 4, 46, 62, 63, 82 |
| gnm             | 24, 78, 79 |
| GSS             | 48  |
| HairEyePlace    | 49  |
| Hauser79        | 50  |
| Heart           | 52  |
| Heckman         | 53  |
| HLtest          | 54  |
| HosmerLemeshow (HLtest) | 54 |
| HospVisits      | 56  |
| Hoyt            | 57  |
ICU, 59
JobSat, 61
Joint, 90, 91
Joint (loglin-utilities), 65
Kappa, 45
Kway, 4, 62, 82
lapply, 46
lboot, 19
legend, 73
llist, 47
logLik, 71, 95
logLik.loglm, 64, 71, 95
loglin, 65, 67, 68, 86, 89
loglin-utilities, 65
loglin2formula (loglin-utilities), 65
loglin2string (loglin-utilities), 65
loglm, 65, 67, 68, 71, 78, 86, 89, 95
loglmlist, 82, 90, 91
loglmlist (glmList), 46
Logseries (logseries), 68
logseries, 68
LRstats, 4, 47, 63, 70, 77, 100, 101
Mammograms, 72
mantelhaen.test, 18, 19
margin.table, 21, 100
markov (loglin-utilities), 65
mcaplot, 72
Mental, 74
Mice, 75
minn38, 58
mjca, 73
Mobility, 76
modFit, 71, 77, 95
mosaic, 5, 67, 79, 81, 82, 85, 86, 90, 91
mosaic.glm, 4, 78, 82, 91
mosaic.glmList, 81
mosaic.loglm, 4
mosaic.loglmlist, 91
mosaic.loglmlist (mosaic.glmList), 81
mosaic3d, 3, 4, 83, 92, 94
mosaicplot, 85, 86
Mult, 24
multilines, 73
mutual, 90, 91
mutual (loglin-utilities), 65
PhdPubs, 86
plogseries (logseries), 68
plot.HLtest (HLtest), 54
plot.loglm, 79
plot.mjca, 72, 73
print.CMHtest (CMHtest), 17
print.GKgamma (GKgamma), 44
print.HLtest (HLtest), 54
print.Kappa, 87
qlogseries (logseries), 68
quantile, 25, 26
range3d (split3d), 92
rlogseries (logseries), 68
rootogram, 55
rootogram.HLtest (HLtest), 54
saturated (loglin-utilities), 65
Saxony, 42
seq.loglm, 88, 91
seq.mosaic, 90
shadings, 79
ShakeWords, 91
shapelist3d, 94
sieve, 81, 90
sieve.glm (mosaic.glm), 78
split3d, 92
structplot, 78, 79, 81, 82, 86, 90, 91
structtable, 84
Summarise, 63, 77, 94
summarise (vcdExtra-deprecated), 100
summary.HLtest (HLtest), 54
Symm, 4, 24
table, 20, 66
Titanic, 96
Titanic.p, 96
Topo, 4, 24
Toxaemia, 97
TV, 98
type.convert, 40
update.formula, 100
update.xtabs, 99
vcdExtra (vcdExtra-package), 3
vcdExtra-deprecated, 100
vcdExtra-package, 3
vglm, 12, 13
INDEX

Vietnam, 101
Vote1980, 102
WorkerSat, 103
Yamaguchi87, 104
zero.test, 106