Package ‘vcdExtra’

March 18, 2026

Type Package

Title 'vcd' Extensions and Additions

Version 0.9.3

Date 2026-03-06

Language en-US

Maintainer Michael Friendly <friendly@yorku.ca>

Depends R (>= 4.1.0), vcd, gnm (>= 1.0-3)

Suggests gmodels, Fahrmeir, effects, VGAM, plyr, lmtest, nnet, ggplot2, Sleuth2, car, lattice, stats4, AER, coin, Hmisc, rmarkdown, seriation, testthat (>= 3.0.0), tibble, glue, here, purrr, readxl, stringr, tidyr (>= 1.3.0), rgl

Imports MASS, grDevices, grid, stats, utils, dplyr, ca, igraph, colorspace, gt, scales, methods, knitr, htmlwidgets, webshot2

Description Provides additional data sets, methods and documentation to complement 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'. Additionally, methods are provided for comparing and visualizing lists of
'glm' and 'loglm' objects. This package is now a support package for the book, ``Discrete Data Analysis with R'' by
Michael Friendly and David Meyer.

License GPL (>= 2)

URL https://friendly.github.io/vcdExtra/,
https://github.com/friendly/vcdExtra

BugReports https://github.com/friendly/vcdExtra/issues

VignetteBuilder knitr, rmarkdown

LazyLoad yes

LazyData yes

Encoding UTF-8

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation no

Author Michael Friendly [aut, cre] (ORCID:
<https://orcid.org/0000-0002-3237-0941>),
David Meyer [ctb],
Achim Zeileis [ctb] (ORCID: <https://orcid.org/0000-0003-0918-3766>),
Duncan Murdoch [ctb],
Heather Turner [ctb] (ORCID: <https://orcid.org/0000-0002-1256-3375>),
David Firth [ctb],
Daniel Sabanes Bove [ctb],
Matt Kumar [ctb],
Shuguang Sun [ctb],
Gavin Klorfine [ctb]

Repository CRAN

Date/Publication 2026-03-18 06:10:15 UTC

Contents

`vcdExtra-package` *Extensions and additions to vcd: Visualizing Categorical Data*

Description

Details

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`.

This package is also a support package for the book, *Discrete Data Analysis with R* by Michael Friendly and David Meyer, Chapman & Hall/CRC, 2016, `https://www.routledge.com/Discrete-Data-Analysis-with-Friendly-Meyer/p/book/9781498725835` with a number of additional data sets, and functions. The web site for the book is `http://ddar.datavis.ca`.

In addition, I teach a course, *Psy 6136: Categorical Data Analysis*, `https://friendly.github.io/psy6136/` using this package.

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 UCBAdmissions 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 (ORCID)

References

Friendly, M. *Visualizing Categorical Data*, Cary NC: SAS Institute, 2000. Web materials: `http://www.datavis.ca/books/vcd/`.

Friendly, M. and Meyer, D. (2016). *Discrete Data Analysis with R: Visualization and Modeling Techniques for Categorical and Count Data*. Boca Raton, FL: Chapman & Hall/CRC. `http://ddar.datavis.ca`.

Meyer, D.; Zeileis, A. & Hornik, K. The Strucplot Framework: Visualizing Multi-way Contingency Tables with vcd *Journal of Statistical Software*, 2006, **17**, 1-48. Available in R via `vignette("strucplot", package = "vcd")`

Turner, H. and Firth, D. *Generalized nonlinear models in R: An overview of the gnm package*, 2007, `http://eprints.ncrm.ac.uk/472/`. Available in R via `vignette("gnmOverview", package = "gnm")`.

See Also

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

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

Examples

```
example(mosaic.glm)

demo("mental-glm")
```

`Abortion` *Abortion Opinion Data*

Description

Opinions about abortion classified by gender and SES

Format

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

No	Name	Levels
1	`Sex`	`"Female", "Male"`
2	`Status`	`"Lo", "Hi"`
3	`Support_Abortion`	`"Yes", "No"`

Details

`Support_Abortion` is a natural response variable.

The combinations of `Sex` and `Status` represent four independent samples, having fixed `Sex-Status` marginal totals. There were 500 females and 600 males. Within the female group, 250 of low status and 250 of high status were sampled. Similarly for the males, with 300 in each of the low and hgh status sub-groups.

This is an example of a product-multinomial sampling scheme. the `Sex:Status` association must be included in any loglinear model where the goal is to determine how attitude toward abortion depends on the others.

Alternatively, a logit model for abortion support may provide a simpler analysis.

Source

Christensen, R. (1990). *Log-Linear Models*, New York, NY: Springer-Verlag, p. 92, Example 3.5.2.

Christensen, R. (1997). *Log-Linear Models and Logistic Regression*, New York, NY: Springer, p. 100, Example 3.5.2.

Examples

```
data(Abortion)


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.

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

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.

`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

Bertin, J. (1983), *Semiology of Graphics*, University of Wisconsin Press.

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=='Died' ~ age + mode + gender,
    weights=Freq, data=Accident, family=binomial)
Anova(acc.bin1)

acc.bin2 <- glm(result=='Died' ~ (age + mode + gender)^2,
    weights=Freq, data=Accident, family=binomial)
Anova(acc.bin2)

acc.bin3 <- glm(result=='Died' ~ (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)


#
```

`AirCrash` *Air Crash Data*

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.

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, https://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, downloaded April 14, 2015.

References

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

Examples

```
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).

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

Agresti, A. (2002). *Categorical Data Analysis*, New York: Wiley, 2nd Ed., Table 7.1

Examples

```
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)
}
```

`Asbestos` *Effect of Exposure to Asbestos*

Description

A two-way contingency table formed from the cross-classification of the number of years of occupational exposure to asbestos and the diagnosed severity of asbestosis of 1117 New York workers. Asbestosis is a chronic lung disease that results in the lung tissue being scared due to contact with the fibers which can lead to severe breathing difficulties.

Format

The format is:

```
num [1:5, 1:4] 310 212 21 25 7 36 158 35 102 35 ...
- attr(*, "dimnames")=List of 2
 ..$ exposure: chr [1:5] "0-9" "10-19" "20-29" "30-39" ...
 ..$ grade   : chr [1:4] "None" "Grade 1" "Grade 2" "Grade 3"#'
```

Details

`exposure` and `grade` should be regarded as ordered factors. Beh and Lombardo (2022) use this data to illustrate a polynomial biplot for ordered categories.

The data summarized here was studied by Beh and Smith (2011) and comes from the original data collected and published by Selikoff (1981) who examined the link between asbestos exposure and asbestosis severity in 1963.

Source

Beh, E. J. & Lombardo, R. (2022). Features of the Polynomial Biplot for Ordered Contingency Tables, *Journal of Computational and Graphical Statistics*, 31:2, 403-412, DOI: 10.1080/10618600.2021.1990773, Table 1.

References

Beh, E. J., and D. R. Smith (2011b), Real World Occupational Epidemiology, Part 2: A Visual Interpretation of Statistical Significance, *Archives of Environmental & Occupational Health*, **66**, 245-248.

Selikoff, I. J. (1981), Household Risks With Inorganic Fibers, *Bulletin of the New York Academy of Medicine*, **57**, 947-961.

Examples

```
data(Asbestos)
# mosaic plot
vcd::mosaic(Asbestos, shade=TRUE, legend=FALSE)

# do the correspondence analysis
library(ca)
Asbestos.ca <- ca(Asbestos)

plot(Asbestos.ca, lines=TRUE)
```

---

assoc_graph *Association Graph for a Loglinear Model*

Description

Construct an undirected graph representing the associations in a loglinear model. Nodes represent variables and edges represent pairwise associations fitted in the model. If two variables are not connected by an edge, they are conditionally independent given the other variables.

Usage

```
assoc_graph(x, ...)

## S3 method for class 'list'
assoc_graph(x, result = c("igraph", "matrix", "edge_list"), ...)

## S3 method for class 'loglm'
assoc_graph(x, result = c("igraph", "matrix", "edge_list"), ...)

## S3 method for class 'glm'
assoc_graph(
  x,
  result = c("igraph", "matrix", "edge_list"),
  measure = c("none", "chisq", "cramer"),
  ...
)

## S3 method for class 'assoc_graph'
print(x, ...)
```

Arguments

`x` An object specifying the model. Can be:

- A `list` of character vectors (a margin/generating class list, as produced by `joint`, `conditional`, etc.)

- A fitted `loglm` object
- A fitted `glm` object (poisson family loglinear model)

`...`	Additional arguments (currently unused).
`result`	Type of result to return: `"igraph"` (default) returns an `igraph` object; `"matrix"` returns the adjacency matrix; `"edge_list"` returns a two-column character matrix of edges.
`measure`	Type of association measure for edge weights (only for `glm` method): `"none"` (default) produces an unweighted graph; `"chisq"` computes partial chi-squared statistics (deviance change when each edge is removed from the model); `"cramer"` computes Cramer's V from the marginal two-way table for each edge.

Details

Each high-order term (margin) in a hierarchical loglinear model defines a clique in the association graph. For example, the term `c("A", "B", "C")` generates edges A–B, A–C, and B–C. Single-variable terms (as in mutual independence) yield isolated nodes with no edges.

For `loglm` objects, the margins are extracted from the `$margin` component. For `glm` objects, the interaction terms are extracted from the model formula.

Value

Depending on `result`:

- `"igraph"`: An `igraph` undirected graph object of class `c("assoc_graph", "igraph")`, with vertex names corresponding to the variable names. When `measure != "none"`, edge weights are stored as `E(g)$weight` and the measure name as `g$measure`.
- `"matrix"`: A symmetric adjacency matrix with variable names as row and column names. Contains 0/1 when unweighted, or association strength values when `measure` is specified.
- `"edge_list"`: When unweighted, a two-column character matrix (from, to). When `measure` is specified, a data frame with columns from, to, and weight.

References

Khamis, H. J. (2011). *The Association Graph and the Multigraph for Loglinear Models*. SAGE Publications. doi:10.4135/9781452226521

Darroch, J. N., Lauritzen, S. L., & Speed, T. P. (1980). Markov Fields and Log-Linear Interaction Models for Contingency Tables. *The Annals of Statistics*, 8(3), 522–539. doi:10.1214/aos/1176345006

Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics*. John Wiley & Sons, Chichester.

See Also

`joint`, `conditional`, `mutual`, `saturated`, `loglin2string`, `seq_loglm`, `plot.assoc_graph`

Other loglinear models: `get_model()`, `glmlist()`, `joint()`, `plot.assoc_graph()`, `seq_loglm()`

Examples

```
# Structural graphs from margin lists (3-way: A, B, C)
mutual(3, factors = c("A", "B", "C"))      |> assoc_graph()
joint(3, factors = c("A", "B", "C"))       |> assoc_graph()
conditional(3, factors = c("A", "B", "C")) |> assoc_graph()
saturated(3, factors = c("A", "B", "C"))   |> assoc_graph()

# Adjacency matrix form
conditional(3, factors = c("A", "B", "C")) |> assoc_graph(result = "matrix")

# From a fitted loglm model (Berkeley admissions)
## Not run:
mod <- MASS::loglm(~ (Admit + Gender) * Dept, data = UCBAdmissions)
assoc_graph(mod)
plot(assoc_graph(mod), main = "Berkeley: [AD] [GD]")

## End(Not run)

# From glm models (Dayton Survey: cigarette, alcohol, marijuana, sex, race)
data(DaytonSurvey)

# Mutual independence + sex*race: one edge only
mod.SR <- glm(Freq ~ . + sex*race, data = DaytonSurvey, family = poisson)
assoc_graph(mod.SR)
plot(assoc_graph(mod.SR), main = "Mutual indep. + [SR]")

# [AM][AC][MC][AR][AS][RS]: {race, sex} indep {marijuana, cigs} | alcohol
mod.cond <- glm(Freq ~ (cigarette + alcohol + marijuana)^2 +
                        (alcohol + sex + race)^2,
                data = DaytonSurvey, family = poisson)

# define groups for the model
gps <- list(c("cigarette", "marijuana"),
            "alcohol",
            c("sex", "race"))

assoc_graph(mod.cond)
plot(assoc_graph(mod.cond),
     groups = gps,
     layout = igraph::layout_nicely,
     main = "{R,S} indep {M,C} | A")

# Weighted graph: partial chi-squared
g <- assoc_graph(mod.cond, measure = "chisq")
g
plot(g, edge.label = TRUE,
     groups = gps,
     layout = igraph::layout_nicely,
     main = "Partial chi-squared weights")

# Cramer's V (marginal)
g2 <- assoc_graph(mod.cond, measure = "cramer")
```

```
g2
plot(g2, edge.label = TRUE,
     groups = gps,
     layout = igraph::layout_nicely,
     main = "Cramer's V weights")
```

as_array *Convert frequency, case, or table form data into an array*

Description

Converts object (obj) in frequency, case or table form into an array. The column containing the frequencies (freq) must be supplied if obj is in frequency form.

Usage

```
as_array(obj, freq = NULL, dims = NULL)
```

Arguments

obj	object to be converted to an array
freq	If obj is in frequency form, this is the name of the frequency column. Leave as NULL if obj is in any other form.
dims	A character vector of dimensions. If not specified, all variables apart from freq will be used as dimensions

Details

Unclasses the as_table() function to return an object in array form.

Value

object in array form

Author(s)

Gavin M. Klorfine

Examples

```
library(vcdExtra)

data("HairEyeColor")

freqForm <- as.data.frame(HairEyeColor) # Generate frequency form data
tidy_freqForm <- dplyr::as_tibble(HairEyeColor) # Generate tidy frequency form data
caseForm <- expand.dft(freqForm) # Generate case form data
```

```
# Frequency form -> array form
as_array(freqForm, freq = "Freq") |> str()

# Warned if forgot to specify freq
as_array(freqForm) |> str()

# Case form -> array form
as_array(caseForm) |> str()

# Frequency (tibble) form -> array form
as_array(tidy_freqForm, freq = "n") |> str()

# For specific dimensions
as_array(tidy_freqForm, freq = "n", dims = c("Hair", "Eye")) |> str()
```

`as_caseform` *Convert frequency or table form into case form.*

Description

Converts object (`obj`) in frequency or table form into case form. The column containing the frequencies (`freq`) must be supplied if `obj` is in frequency form. Returns a tibble if `tidy` is set to `TRUE`.

Usage

```
as_caseform(obj, freq = "Freq", dims = NULL, tidy = TRUE)
```

Arguments

`obj`	object to be converted to case form
`freq`	If `obj` is in frequency form, this is the name of the frequency column. If `obj` is in any other form, do not supply an argument (see "Details")
`dims`	A character vector of dimensions. If not specified, all variables apart from `freq` will be used as dimensions
`tidy`	returns a tibble if set to TRUE

Details

A wrapper for `expand.dft()` that is able to handle arrays.

If a frequency column is not supplied, this function defaults to "Freq" just like `expand.dft()`. Converts `obj` to a table using `as_table()` before converting to case form.

Value

object in case form.

Author(s)

Gavin M. Klorfine

Examples

```
library(vcdExtra)

data("HairEyeColor")

freqForm <- as.data.frame(HairEyeColor) # Generate frequency form data
tidy_freqForm <- dplyr::as_tibble(HairEyeColor) # Generate tidy frequency form data
tableForm <- as_table(HairEyeColor) # Generate table form data
arrayDat <- as_array(HairEyeColor) # Generate an array

# Frequency form -> case form
as_caseform(freqForm) |> str()

# Frequency form (tibble) -> case form
as_caseform(tidy_freqForm, freq = "n") |> str()

# Array -> case form
as_caseform(arrayDat) |> str()

# Optionally specify dims
as_caseform(tableForm, dims = c("Hair", "Eye")) |> str()
```

`as_freqform` *Convert any form (case or table form) into frequency form.*

Description

A wrapper for `as.data.frame()` that is able to properly handle arrays. Converts object (`obj`) in case or table form into frequency form. The column containing the frequencies (`freq`) must be supplied if `obj` is already in frequency form (and you are using this function to select dimensions). Returns a tibble if `tidy` is set to `TRUE`.

Usage

```
as_freqform(obj, freq = NULL, dims = NULL, tidy = TRUE)
```

Arguments

`obj`	object to be converted to frequency form
`freq`	If `obj` is already in frequency form, this is the name of the frequency column. If `obj` is in any other form, do not supply an argument (see "Details")
`dims`	A character vector of dimensions. If not specified, all variables apart from `freq` will be used as dimensions
`tidy`	returns a tibble if set to TRUE

Details

Converts `obj` to a table using `as_table()` before converting to frequency form

Value

object in frequency form.

Author(s)

Gavin M. Klorfine

Examples

```
library(vcdExtra)

data("HairEyeColor")

freqForm <- as.data.frame(HairEyeColor) # Generate frequency form data
tableForm <- as_table(HairEyeColor) # Generate table form data
arrayDat <- as_array(HairEyeColor) # Generate an array
caseForm <- as_caseform(HairEyeColor) # Generate case form data

# array -> frequency form
as_freqform(arrayDat) |> str()

# table -> frequency form
as_freqform(tableForm) |> str()

# case -> frequency form
as_freqform(caseForm) |> str()

# Selecting dimensions (optional)
as_freqform(freqForm, freq = "Freq", dims = c("Hair", "Eye")) |> str()

as_freqform(tableForm, dims = c("Hair", "Eye")) |> str()
```

`as_table` *Convert frequency or case form data into table form*

Description

Converts object (`obj`) in frequency or case form into table form. The column containing the frequencies (`freq`) must be supplied if `obj` is in frequency form. Optionally returns a table of proportions with (optionally) specified margins.

Usage

```
as_table(obj, freq = NULL, dims = NULL, prop = NULL)
```

Arguments

obj	object to be converted to table form
freq	If `obj` is in frequency form, this is the name of the frequency column. Leave as `NULL` if `obj` is in any other form.
dims	A character vector of dimensions. If not specified, all variables apart from `freq` will be used as dimensions
prop	If set to TRUE, returns a table of proportions. May also be set to a character or numeric vector of margins.

Details

If `obj` was in table form to begin with, it is returned to the user as-is unless dimensions were specified (in which case it returns a table with entries summed over excluded dimensions). When `prop` is set to `TRUE`, the returned table will have proportions that sum to one, whereas if a character or numerical vector of table dimensions is supplied to `prop`, proportions will be marginalized across the specified dimensions.

Value

object in table form

Author(s)

Gavin M. Klorfine

Examples

```
library(vcdExtra)

data("HairEyeColor")

freqForm <- as.data.frame(HairEyeColor) # Generate frequency form data
tidy_freqForm <- dplyr::as_tibble(HairEyeColor) # Generate tidy frequency form data
caseForm <- expand.dft(freqForm) # Generate case form data

# Frequency form -> table form
as_table(freqForm, freq = "Freq") |> str()

# Warned if forgot to specify freq
as_table(freqForm) |> str()

# Frequency form (tibble) -> table form
as_table(tidy_freqForm, freq = "n") |> str()

# Case form -> table form
as_table(caseForm) |> str()

# For specific dimensions
as_table(tidy_freqForm, freq = "n", dims = c("Hair", "Eye")) |> str()
```

```
#-----For proportions-----#

as_table(freqForm, freq = "Freq", prop = TRUE) |> head(c(4,4,1)) # print only Sex == Male rows

# Marginalize proportions along "Sex" (i.e., male proportions sum to 1, female proportions sum to 1)
as_table(freqForm, freq = "Freq", prop = "Sex") |> head(c(4,4,1))

as_table(freqForm, freq = "Freq", prop = 3) |> head(c(4,4,1)) # Same as above

# Marginalize proportions along multiple variables
as_table(freqForm, freq = "Freq", prop = c("Hair", "Sex")) |> head(c(4,4,1))

as_table(freqForm, freq = "Freq", prop = c(1, 3)) |> head(c(4,4,1)) # Same as above

# Using dims and prop arguments in tandem
as_table(freqForm, freq = "Freq", dims = c("Hair", "Eye"), prop = TRUE)
```

Bartlett *Bartlett Data on Plum Root Cuttings*

Description

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.

Format

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

dim	Name	Levels
1	`Alive`	`"Alive", "Dead"`
2	`Time`	`"Now", "Spring"`
3	`Length`	`"Long", "Short"`

Details

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

Source

Hand, D. and Daly, F. and Lunn, A. D.and McConway, K. J. and Ostrowski, E. (1994). *A Handbook of Small Data Sets*. London: Chapman & Hall, p. 15, # 19.

References

Bartlett, M. S. (1935). Contingency Table Interactions *Journal of the Royal Statistical Society*, Supplement, 1935, 2, 248-252.

Examples

```
data(Bartlett)

# measures of association
assocstats(Bartlett)
oddsratio(Bartlett)

# Test models

## Independence
MASS::loglm(formula = ~Alive + Time + Length, data = Bartlett)

## No three-way association
MASS::loglm(formula = ~(Alive + Time + Length)^2, data = Bartlett)

# Use woolf_test() for a formal test of homogeneity of odds ratios
vcd::woolf_test(Bartlett)


# Plots
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.

Usage

```
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

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

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 `rev=TRUE` or supply `Y[,4:1]` as the first argument.

Value

A data frame with `nrow(Y)` rows and `3 + ncol(row.vars)` columns

Author(s)

Michael Friendly

References

Friendly, M. and Meyer, D. (2016). *Discrete Data Analysis with R: Visualization and Modeling Techniques for Categorical and Count Data*. Boca Raton, FL: Chapman & Hall/CRC. `http://ddar.datavis.ca`.

See Also

`vglm`

Examples

```
data(Toxaemia)
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.

Format

A frequency data frame (representing a 3 x 3 x 2 x 2 frequency table) with 36 cells 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

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.

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

Burt, C. (1950). The factorial analysis of qualitative data, *British Journal of Statistical Psychology*, **3**(3), 166-185. Table IX.

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).

Format

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

dim	Name	Levels
1	`Infection`	`"Type 1"`, `"Type 2"`, `"None"`
2	`Risk`	`"Yes"`, `"No"` (presence of risk factors)
3	`Antibiotics`	`"Yes"`, `"No"` (were antibiotics given?)
4	`Planned`	`"Yes"`, `"No"` (was the C section planned?)

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

% Fahrmeir:94 Fahrmeir, L. & Tutz, G. (1994). Multivariate Statistical Modelling Based on Generalized Linear Models New York: Springer Verlag, Table 1.1.

See Also

`caesar` 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*

Description

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

Format

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

dim	Name	Levels
1	`Survival`	`"Died", "Surv"`
2	`Grade`	`"Malignant", "Benign"`
3	`Center`	`"Boston", "Glamorgan"`

Source

Lindsey, J. K. (1995). Analysis of Frequency and Count Data Oxford, UK: Oxford University Press. p. 38, Table 2.5.

Whittaker, J. (1990) Graphical Models in Applied Multivariate Statistics New York: John Wiley and Sons, p. 220.

Examples

```
data(Cancer)

MASS::loglm(~Survival + Grade + Center, data = Cancer)

vcd::mosaic(Cancer, shade=TRUE)
```

`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.

Usage

```
CMHtest(x, ...)

## S3 method for class 'formula'
CMHtest(formula, data = NULL, subset = NULL, na.action = NULL, ...)

## Default S3 method:
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'
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.
`...`	Other arguments passed to default method.
`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 `NA`s. 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
`digits`	Digits to print.

Details

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

The standard χ^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 etal., 1978).

These include the CMH **correlation** statistic (`"cor"`), treating both factors as ordered. For a given statum, 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 $(n-1)r_S^2$, 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 χ^2 test. When there is more than one stratum, the overall general CMH statistic gives a stratum-adjusted Pearson χ^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.

Note that strata combinations with insufficient data (less than 2 observations) are automatically omitted from the analysis.

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
`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

Stokes, M. E. & Davis, C. S. & Koch, G., (2000). *Categorical Data Analysis using the SAS System*, 2nd Ed., Cary, NC: SAS Institute, pp 74-75, 92-101, 124-129. Details of the computation are given at: `http://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_freq_a0000000648.htm`

Cochran, W. G. (1954), Some Methods for Strengthening the Common χ^2 Tests, *Biometrics*, 10, 417-451.

Landis, R. J., Heyman, E. R., and Koch, G. G. (1978). Average Partial Association in Three-way Contingency Tables: A Review and Discussion of Alternative Tests, *International Statistical Review*, **46**, 237-254.

Mantel, N. (1963), Chi-square Tests with One Degree of Freedom: Extensions of the Mantel-Haenszel Procedure," *Journal of the American Statistical Association*, 58, 690-700.

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

Other association tests: GKgamma(), HLtest(), woolf_test(), zero.test()

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 etal, 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 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

```
# 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"),
    education=c("low", "low", "high"))
structable(t3)

##                  age   A   B   C
## sex    education
## Male   low           412 405 360
##        high          169 206 192
## Female low           413 370 415
##        high          178 199 185



# change labels for levels of education to 1:3
t4 <- collapse.table(t1,  education=1:3)
structable(t4)
structable(t4)
##                  age   a   b   c   d   e   f
## sex    education
## Male   1             119 101 109  85  99  93
##        2              94  98 103 108  84  84
##        3              81  88  96 110 100  92
## Female 1             107 104  95  86 103  96
##        2             104  98  94  95 110 106
##        3              93  85  90 109  99  86
```

color_table *Smart Display of Frequency Table with Colored Cell Backgrounds*

Description

color_table() creates a formatted, semi-graphic "heatmap" table display of frequency data with cell backgrounds colored according to observed frequencies or their residuals from a loglinear model. The goal is to provide a "smart" tabular display of cross-classified frequency data, with some abilities to highlight possibly interesting patterns or unusual cells. This is an S3 generic function, providing methods for different input types: "table", "xtabs", "matrix", "ftable", "structable" or "data.frame".

Usage

```
color_table(x, ...)

## S3 method for class 'table'
color_table(
  x,
  formula = NULL,
  values = c("freq", "residuals"),
  shade = c("residuals", "freq", "pearson", "deviance"),
  model = NULL,
  expected = NULL,
  palette = NULL,
  legend = FALSE,
  margins = TRUE,
  digits = 0,
  title = NULL,
  filename = NULL,
  ...
)
```

```
## S3 method for class 'ftable'
color_table(
  x,
  values = c("freq", "residuals"),
  shade = c("residuals", "freq", "pearson", "deviance"),
  model = NULL,
  expected = NULL,
  palette = NULL,
  legend = FALSE,
  margins = TRUE,
  digits = 0,
  title = NULL,
  filename = NULL,
  ...
)

## S3 method for class 'structable'
color_table(
  x,
  values = c("freq", "residuals"),
  shade = c("residuals", "freq", "pearson", "deviance"),
  model = NULL,
  expected = NULL,
  palette = NULL,
  legend = FALSE,
  margins = TRUE,
  digits = 0,
  title = NULL,
  filename = NULL,
  ...
)

## S3 method for class 'data.frame'
color_table(
  x,
  formula = NULL,
  freq_col = NULL,
  values = c("freq", "residuals"),
  shade = c("residuals", "freq", "pearson", "deviance"),
  model = NULL,
  expected = NULL,
  palette = NULL,
  legend = FALSE,
  margins = TRUE,
  digits = 0,
  title = NULL,
  filename = NULL,
  ...
)

## S3 method for class 'matrix'
color_table(
  x,
  values = c("freq", "residuals"),
  shade = c("residuals", "freq", "pearson", "deviance"),
  model = NULL,
  expected = NULL,
  palette = NULL,
  legend = FALSE,
  margins = TRUE,
  digits = 0,
  title = NULL,
  filename = NULL,
  ...
)

## Default S3 method:
color_table(x, ...)
```

Arguments

`x`	A `"table"`, `"xtabs"`, `"matrix"`, `"ftable"`, `"structable"`, or `"data.frame"` object
`...`	Additional arguments passed to methods
`formula`	Formula specifying a `row_vars ~ col_vars` layout (for multi-way tables) to make them "flat" as defined for `vcd::structable()` and `stats::ftable()`.
`values`	What values to display in cells: `"freq"` for observed frequencies (default), or `"residuals"` to display the residual values. When `values = "residuals"`, margins are suppressed since residuals don't have meaningful totals.
`shade`	What values determine cell shading: `"residuals"` (default), `"freq"`, `"pearson"`, or `"deviance"`
`model`	A fitted model (loglm or glm) to compute residuals from. If NULL and shade involves residuals, uses an independence model for all factors.
`expected`	Expected frequencies (alternative to `model`), a data structure of the same shape as `x`
`palette`	Color palette function or vector for background colors. Default depends on shade type. When `shade = "freq"` the default is `palette = c("white", "firebrick")`; otherwise `c("#B2182B", "white", , "#2166AC")` ranging from red to blue for negative and positive residuals. The background colors are computed by interpolation using `scales::col_numeric()`.
`legend`	Controls display of shading interpretation note: `TRUE` or `"note"` (default) adds a source note explaining the shading; `FALSE` (default) suppresses the note, but a message is printed in the console.
`margins`	Logical, include row/column totals?

`digits`	Number of decimal places for displayed values
`title`	Optional table title
`filename`	Optional filename to save the table as an image. If provided, the table is saved using `gtsave`. Supported formats include `.png`, `.pdf`, `.html`, `.rtf`, and `.docx`. The file format is determined by the file extension. Other arguments can be passed to `gtsave` via `...`.
`freq_col`	Name of the frequency column. If NULL, looks for "Freq" or "count".

Details

This function provides a heatmap-style representation of a frequency table, where background coloring is used to visualize patterns and anomalies in the data. When shading by *residuals* (the default), cells with large positive residuals (more observations than expected) are shaded red, while cells with large negative residuals (fewer than expected) are shaded blue. This makes it easy to identify cells that deviate substantially from what would be expected under a given model (by default, the independence model).

For multi-way tables (3 or more dimensions), residuals are computed from the model of complete independence among all factors using `loglm`. But you can specify a model using the `model` or `expected` arguments, in a way similar to that provided by `mosaic.glm()`. A message is printed showing the chi-squared statistic, degrees of freedom, and p-value for this test.

Contrast shading

For cells with dark background colors, black text can be difficult to read. This function automatically selects white or black text for each cell based on which provides better contrast against the background color. If the **colorspace** package is available, `contrast_ratio` is used to determine the optimal text color according to WCAG 2.1 guidelines. Otherwise, a fallback based on relative luminance (ITU-R BT.709) is used.

Use in documents

In R Markdown (`.Rmd`) or Quarto (`.qmd`) documents, **gt** tables render natively in **HTML output** — simply return the `gt` object from a chunk and knitr renders it automatically via **gt**'s built-in `knit_print` method. No `filename` argument is needed.

For **PDF or Word output**, **gt** does not render natively. Use the `filename` argument to save the table as a `.png` image, then include it with `include_graphics`:

```
color_table(my_table, filename = "my_table.png")
knitr::include_graphics("my_table.png")
```

The `vwidth` and `vheight` arguments (passed via `...`) control the image viewport size in pixels. Supported save formats are `.png`, `.pdf`, `.html`, `.rtf`, and `.docx`.

For documents that target **multiple output formats**, a small helper that branches on `is_html_output` avoids duplicating code:

```
gt_obj <- color_table(my_table)
if (knitr::is_html_output()) {
  gt_obj
} else {
  gt::gtsave(gt_obj, "my_table.png")
  knitr::include_graphics("my_table.png")
}
```

If you need a caption or cross-reference label, use `gt::tab_caption()` on the returned object:

```
    color_table(my_table) |>
      gt::tab_caption("Table 1: Pattern of association in MyTable")
```

Value

A gt table object that can be further customized

Methods (by class)

- `color_table(table)`: Method for table objects (including result of xtabs)
- `color_table(ftable)`: Method for ftable objects
- `color_table(structable)`: Method for structable objects (vcd package)
- `color_table(data.frame)`: Method for data.frame in frequency form
- `color_table(matrix)`: Method for matrix objects
- `color_table(default)`: Default method

Examples

```
## Not run:
# Basic usage with 2-way table - shade by residuals from independence
data(HairEyeColor)
HEC <- margin.table(HairEyeColor, 1:2)  # 2-way: Hair x Eye
color_table(HEC)

# Shade by frequencies instead (no message printed)
color_table(HEC, shade = "freq")

# 3-way table - using a formula to specify layout
color_table(HairEyeColor, formula = Eye ~ Hair + Sex)

# Display residual values in cells instead of frequencies
color_table(HEC, values = "residuals")

# From a data.frame in frequency form (2-way)
hec_df <- as.data.frame(HEC)
color_table(hec_df)

# Save table as an image file
color_table(HEC, filename = "hair_eye_table.png")

## End(Not run)
```

Cormorants *Advertising Behavior by Males Cormorants*

Description

Male double-crested cormorants use advertising behavior to attract females for breeding. In this study by Meagan McRae (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.

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

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

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.

```
Cormorants$days <- ifelse(Cormorants$week \%in\% 3:4, 2, 3)
```

Source

McRae, M. (2015). Spatial, Habitat and Frequency Changes in Double-crested Cormorant Advertising Display in a Tree-nesting Colony. Unpublished MA project, Environmental Studies, York University.

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))
```

`CrabSatellites` *Horseshoe Crab Mating*

Description

Determinants of mating for male satellites to nesting horseshoe crabs. The number of `satellites` is a natural outcome variable. This dataset is useful for exploring various count data models.

Format

A data frame containing 173 observations on 5 variables.

`color` Ordered factor indicating color (light medium, medium, dark medium, dark).

`spine` Ordered factor indicating spine condition (both good, one worn or broken, both worn or broken).

`width` Carapace width (cm).

`weight` Weight (kg).

`satellites` Number of satellites.

Details

Brockmann (1996) investigates horseshoe crab mating. The crabs arrive on the beach in pairs to spawn. Furthermore, unattached males also come to the beach, crowd around the nesting couples and compete with attached males for fertilizations. These so-called satellite males form large groups around some couples while ignoring others. Brockmann (1996) shows that the groupings are not driven by environmental factors but by properties of the nesting female crabs. Larger females that are in better condition attract more satellites.

Agresti (2002, 2013) reanalyzes the number of satellites using count models. Explanatory variables are the female crab's color, spine condition, weight, and carapace width. Color and spine condition are ordered factors but are treated as numeric in some analyses.

Source

Table 4.3 in Agresti (2002). This dataset was taken from the **countreg** package, which is not on CRAN

References

Agresti A (2002). Categorical Data Analysis, 2nd ed., John Wiley & Sons, Hoboken.

Agresti A (2013). Categorical Data Analysis, 3rd ed., John Wiley & Sons, Hoboken. Brockmann HJ (1996). "Satellite Male Groups in Horseshoe Crabs, Limulus polyphemus", Ethology, 102(1), 1–21.

Examples

```
## load data, use ordered factors as numeric, and grouped factor version of width
data("CrabSatellites", package = "vcdExtra")
CrabSatellites <- transform(CrabSatellites,
  color = as.numeric(color),
  spine = as.numeric(spine),
  cwidth = cut(width, c(-Inf, seq(23.25, 29.25), Inf))
)

## Agresti, Table 4.4
aggregate(CrabSatellites$satellites,
          list(CrabSatellites$cwidth), function(x)
  round(c(Number = length(x), Sum = sum(x), Mean = mean(x), Var = var(x)), digits = 2))

## Agresti, Figure 4.4
plot(tapply(satellites, cwidth, mean) ~ tapply(width, cwidth, mean),
  data = CrabSatellites,
  ylim = c(0, 6), pch = 19, cex = 1.5,
  xlab = "Mean carapace width (cm)",
  ylab = "Mean number of satellites")

## More examples: ?countreg::CrabSatellites` has examples of other plots and count data models
```

`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

Details

Instead of treating all mobility as equal, this model posits that the difficulty of moving between categories increases with the number of boundaries (or "crossings") that must be crossed, and that associations between categories decrease with their separation.

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

Goodman, L. (1972). Some multiplicative models for the analysis of cross-classified data. In: *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, CA: University of California Press, pp. 649-696.

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
structable(~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)

# what does Crossings do?
cr <- with(Hauser79, Crossings(Father, Son))
head(cr)
# Show the codings for varying Crossings levels
matrix(cr[,1], nrow=5)
matrix(cr[,2], nrow=5)
matrix(cr[,3], nrow=5)
matrix(cr[,4], nrow=5)
```

---

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 specified.

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

Friendly, M. and Meyer, D. (2016). *Discrete Data Analysis with R: Visualization and Modeling Techniques for Categorical and Count Data*. Boca Raton, FL: Chapman & Hall/CRC. http://ddar.datavis.ca.

See Also

cut, quantile

Examples

```
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")
}

data("CrabSatellites", package = "vcdExtra")

# jittered scatterplot
plot(jitter(satellites) ~ width, data=CrabSatellites,
  ylab="Number of satellites (jittered)",
  xlab="Carapace width",
  cex.lab=1.25)
with(CrabSatellites,
     lines(lowess(width, satellites), col="red", lwd=2))

# boxplot, using deciles
plot(satellites ~ cutfac(width), data=CrabSatellites,
     ylab="Number of satellites",
     xlab="Carapace width (deciles)")
```

CyclingDeaths *London Cycling Deaths*

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.

Format

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

`date` a Date

`deaths` number of deaths, a numeric vector

Source

https://www.data.gov.uk/dataset/cb7ae6f0-4be6-4935-9277-47e5ce24a11f/road-accidents-safety-data, STATS 19 data, 2005-2012, using the files `Casualty0512.csv` and `Accidents0512.csv`

References

Aberdein, Jody and Spiegelhalter, David (2013). Have London's roads become more dangerous for cyclists? *Significance*, 10(6), 46–48.

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)
```

`datasets` *Information on Data Sets in Packages*

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.

Usage

```
datasets(
  package = "vcdExtra",
  allClass = FALSE,
  incPackage = length(package) > 1,
  maxTitle = NULL,
  ndim = FALSE
)
```

Arguments

`package`	a character vector giving the package(s) to look in
`allClass`	a logical variable. 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
`ndim`	logical; include the number of dimensions in the result?

Details

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.

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, a character variable
`class`	class, the object class of the data set, typically one of `"data.frame"`, `"table"`, `"array"` ...
`dim`	an abbreviation of the dimensions of the data set, in a form like `"36x3"` for a data.frame or matrix with 36 rows and 3 columns.
`Title`	data set title

Note

In Rmd documents, `datasets("package") |> knitr::kable()` can be used to create a more pleasing display.

Author(s)

Michael Friendly, with R-help from Curt Seeliger

See Also

`data`, `kable`

Examples

```
datasets("vcdExtra")
# datasets(c("vcd", "vcdExtra"))
datasets("datasets", maxTitle=50)

# just list dataset names in a package
datasets("vcdExtra")[,"Item"]
datasets("vcd")[,"Item"]

# Find one-way tables in vcd (dim doesn't contain "x")
datasets("vcd") |>
  dplyr::filter(class=="table",
                !stringr::str_detect(dim, "x"))
```

DaytonSurvey *Dayton Student Survey on Substance Use*

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.

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

Agresti, A. (2002). *Categorical Data Analysis*, 2nd Ed., New York: Wiley-Interscience, Table 9.1, p. 362.

References

Thompson, L. (2009). *R (and S-PLUS) Manual to Accompany Agresti's Categorical Data*, http://www.stat.ufl.edu/~aa/cda/Th

Examples

```
data(DaytonSurvey)

# mutual independence
mod.0  <- glm(Freq ~ ., data=DaytonSurvey, family=poisson)

# mutual independence + GR
mod.GR <- glm(Freq ~ . + sex*race, data=DaytonSurvey, family=poisson)
anova(mod.GR, test = "Chisq")

# all two-way terms
mod.all2way <- glm(Freq ~ .^2, data=DaytonSurvey, family=poisson)
anova(mod.all2way, test = "Chisq")

# compare models
LRstats(mod.0, mod.GR, mod.all2way)

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

Depends *Dependencies of R Packages*

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.

Format

The format is a one-way frequency table of counts of packages with 0, 1, 2, ... dependencies.

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

Source

Using code from https://www.r-bloggers.com/2013/12/a-look-at-the-distribution-of-r-package-dependenci

Examples

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

# what type of distribution?
# Ord_plot can't classify this!
Ord_plot(Depends)

## 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
Deps <- as.vector(c(zeros,pkgs$numDepends))     # Set up to tablate
Depends <- table(Deps)


## 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.

Format

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

dim	Name	Levels
1	`Temperature`	`"High", "Low"`
2	`M_User`	`"Yes", "No"`
3	`Preference`	`"Brand X", "Brand M"`
4	`Water_softness`	`"Soft", "Medium", "Hard"`

Source

Fienberg, S. E. (1980). *The Analysis of Cross-Classified Categorical Data* Cambridge, MA: MIT Press, p. 71.

References

Ries, P. N. & Smith, H. (1963). The use of chi-square for preference testing in multidimensional problems. *Chemical Engineering Progress*, 59, 39-43.

Examples

```
data(Detergent)

# basic mosaic plot
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)
```

`dlogseries` *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.

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 $P[X \le x]$, otherwise, $P[X > x]$.
`p`	vector of probabilities
`max.value`	maximum value returned by `qlogseries`
`n`	number of observations for `rlogseries`

Details

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

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 cumulative 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

Fisher, R. A. and Corbet, A. S. and Williams, C. B. (1943). The relation between the number of species and the number of individuals *Journal of Animal Ecology*, 12, 42-58.

See Also

`Distributions`

Examples

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

require(lattice)
```

```
mycol <- palette()[2:4]
xyplot( prob ~ x, data=lgs.df, 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(lgs.df$p)),
x=0.9, y=0.98, corner=c(x=1, y=1)
)
)


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

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.

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

What factors affected who lived and who died?

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'`.

`survived` is the response variable. What kind of models should be used here?

Source

D. K. Grayson, 1990, "Donner party deaths: A demographic assessment", *J. Anthropological Research*, **46**, 223-242.

Johnson, K. (1996). *Unfortunate Emigrants: Narratives of the Donner Party*. Logan, UT: Utah State University Press. Additions, and dates of death from `http://user.xmission.com/~octa/DonnerParty/Roster.htm`.

References

Ramsey, F.L. and Schafer, D.W. (2002). *The Statistical Sleuth: A Course in Methods of Data Analysis*, (2nd ed), Duxbury.

Friendly, M. and Meyer, D. (2016). *Discrete Data Analysis with R: Visualization and Modeling Techniques for Categorical and Count Data*. Boca Raton, FL: Chapman & Hall/CRC. `http://ddar.datavis.ca`.

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))
```

Draft1970 *USA 1970 Draft Lottery Data*

Description

This data set gives the results of the 1970 US draft lottery, in the form of a data frame.

Format

A data frame with 366 observations on the following 3 variables.

`Day` day of the year, 1:366

`Rank` draft priority rank of people born on that day

`Month` an ordered factor with levels `Jan` < `Feb` ...< `Dec`

Details

The draft lottery was used to determine the order in which eligible men would be called to the Selective Service draft. The days of the year (including February 29) were represented by the numbers 1 through 366 written on slips of paper. The slips were placed in separate plastic capsules that were mixed in a shoebox and then dumped into a deep glass jar. Capsules were drawn from the jar one at a time.

The first number drawn was 258 (September 14), so all registrants with that birthday were assigned lottery number `Rank` 1. The second number drawn corresponded to April 24, and so forth. All men 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

Starr, N. (1997). Nonrandom Risk: The 1970 Draft Lottery, *Journal of Statistics Education*, v.5, n.2 doi:10.1080/10691898.1997.11910534

References

Fienberg, S. E. (1971), "Randomization and Social Affairs: The 1970 Draft Lottery," *Science*, 171, 255-261.

`https://en.wikipedia.org/wiki/Draft_lottery_(1969)`

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
Draft1970$Risk <- cut(Draft1970$Rank, breaks=3, labels=c("High", "Med", "Low"))
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.

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

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

References

Fienberg, S. E. (1971), "Randomization and Social Affairs: The 1970 Draft Lottery," *Science*, 171, 255-261.

Starr, N. (1997). Nonrandom Risk: The 1970 Draft Lottery, *Journal of Statistics Education*, v.5, n.2 doi:10.1080/10691898.1997.11910534

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, {
  Month <- ordered(Month)
  Risk <- ordered(Risk, levels=rev(levels(Risk)))
  })
str(Draft1970df)

# similar model, as a Poisson GLM
indep <- glm(Freq ~ Month + Risk, family = poisson, data = Draft1970df)

mosaic(indep, residuals_type="rstandard", gp=shading_Friendly)

# numeric scores for tests of ordinal factors
Cscore <- as.numeric(Draft1970df$Risk)
Rscore <- as.numeric(Draft1970df$Month)

# linear x linear association between Month and Risk
linlin <- glm(Freq ~ Month + Risk + Rscore:Cscore, family = poisson, data = Draft1970df)

# compare models
anova(indep, linlin, test="Chisq")
mosaic(linlin, residuals_type="rstandard", gp=shading_Friendly)
```

Dyke *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.

Format

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

dim	Name	Levels
1	`Knowledge`	`"Good", "Poor"`
2	`Reading`	`"No", "Yes"`
3	`Radio`	`"No", "Yes"`
4	`Lectures`	`"No", "Yes"`
5	`Newspaper`	`"No", "Yes"`

Source

Fienberg, S. E. (1980). *The Analysis of Cross-Classified Categorical Data* Cambridge, MA: MIT Press, p. 85, Table 5-6.

References

Dyke, G. V. and Patterson, H. D. (1952). Analysis of factorial arrangements when the data are proportions. *Biometrics*, 8, 1-12.

Lindsey, J. K. (1993). *Models for Repeated Measurements* Oxford, UK: Oxford University Press, p. 57.

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

```
expand.dft(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

Originally posted on R-Help, Jan 20, 2009, http://tolstoy.newcastle.edu.au/R/e6/help/09/01/1873.html

Friendly, M. and Meyer, D. (2016). *Discrete Data Analysis with R: Visualization and Modeling Techniques for Categorical and Count Data*. Boca Raton, FL: Chapman & Hall/CRC. http://ddar.datavis.ca.

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.

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, 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

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

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

Source

Gart, J. J. (1971). The comparison of proportions: a review of significance tests, confidence intervals and adjustments for stratification. *International Statistical Review*, 39, 148-169.

References

Breslow, N. (1976), Regression analysis of the log odds ratio: A method for retrospective studies, *Biometrics*, 32(3), 409-416.

Examples

```
data(Fungicide)
# loddsratio was moved to vcd; requires vcd_1.3-3+
## Not run:
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).

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

Edwards, A. W. F. (1958). An Analysis Of Geissler's Data On The Human Sex Ratio. *Annals of Human Genetics*, 23, 6-15.

References

Friendly, M. and Meyer, D. (2016). *Discrete Data Analysis with R: Visualization and Modeling Techniques for Categorical and Count Data*. Boca Raton, FL: Chapman & Hall/CRC. http://ddar.datavis.ca.

Geissler, A. (1889). *Beitrage zur Frage des Geschlechts verhaltnisses der Geborenen* Z. K. Sachsischen Statistischen Bureaus, 35, n.p.

Lindsey, J. K. & Altham, P. M. E. (1998). Analysis of the human sex ratio by using overdispersion models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 47, 149-157.

See Also

`Saxony`, containing the data for families of size 12.

Examples

```
data(Geissler)
str(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
```

---

get_model — *Extract Model Formulas from Model Objects or Model Lists*

Description

`get_model()` extracts the model formula or bracket notation from a single `loglm` or `glm` object.

Usage

```
get_model(x, type = c("brackets", "formula"), abbrev = FALSE, ...)

get_models(x, type = c("brackets", "formula"), abbrev = FALSE, ...)
```

Arguments

`x`	For `get_model()`: a `loglm` or `glm` object. For `get_models()`: a `glmlist` or `loglmlist` object.
`type`	Type of output: `"brackets"` for loglinear bracket notation (e.g., `"[AB] [C]"`), or `"formula"` for R formula notation. For `glm` and `glmlist` objects, only `"formula"` is meaningful.
`abbrev`	Logical or integer. If `TRUE` or a positive integer, abbreviate factor names to that many characters (default 1 when `TRUE`). Only applies to bracket notation.
`...`	Additional arguments passed to `loglin2string` such as `sep` and `collapse`.

Details

`get_models()` does the same for each model in a `loglmlist` or `glmlist` object. These are useful for labeling models in summaries and plots.

For `loglm` objects created by `seq_loglm`, the bracket notation is stored in the `model.string` component. For other `loglm` objects, it is constructed from the `margin` component using `loglin2string`.

For `glm` objects, the formula is extracted using `formula()`.

Model notation

For `loglmlist` objects created by `seq_loglm`, the bracket notation distinguishes between models fit to marginal sub-tables and models fit to the full table. Parentheses are used for marginal sub-tables, e.g., `"(Class) (Sex)"`, while square brackets are used for the full table, e.g., `"[Class,Sex,Age] [Survived]"`.

The `abbrev` argument can be used to abbreviate factor names for more compact display, e.g., `"[C,S,A] [S]"`.

Value

For `get_model()`: a character string with the model formula or bracket notation.

```
For `get_models()`: a named character vector with the model formulas or bracket notations.
```

See Also

glmlist, loglmlist, loglin2string, LRstats, seq_mosaic

Other glmlist functions: Kway(), LRstats(), glmlist(), mosaic.glmlist()

Other loglinear models: assoc_graph(), glmlist(), joint(), plot.assoc_graph(), seq_loglm()

Examples

```
data(Titanic)
tit.joint <- seq_loglm(Titanic, type = "joint")

# Single model
get_model(tit.joint[[2]])
get_model(tit.joint[[4]])
get_model(tit.joint[[4]], abbrev = TRUE)
get_model(tit.joint[[4]], type = "formula")

# Model list
get_models(tit.joint)
get_models(tit.joint, type = "formula")

# With abbreviated factor names
get_models(tit.joint, abbrev = TRUE)
get_models(tit.joint, abbrev = 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.

Format

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

No	Name	Levels

1	Dullness	"Ment. defective", "Slow", "Slow Intell", "Fairly Intell", "Capable", "V.Able"
2	Clothing	"V.Well clad", "Well clad", "Passable", "Insufficient"

Details

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

Source

Anscombe, F. J. (1981). *Computing in Statistical Science Through APL*. New York: Springer-Verlag, p. 302

References

Gilby, W. H. (1911). On the significance of the teacher's appreciation of general intelligence. *Biometrika*, 8, 93-108 (esp. p. 94). (Quoted by Kendall (1943,..., 1953) Table 13.1, p 320.)

Examples

```
data(Gilby)

# CMH tests treating row/column variables as ordinal
CMHtest(Gilby)

mosaic(Gilby, shade=TRUE)

# correspondence analysis to see relations among categories
if(require(ca)){
ca(Gilby)
plot(ca(Gilby), lines=TRUE)

}
```

GKgamma *Calculate Goodman-Kruskal Gamma for ordered tables*

Description

The Goodman-Kruskal γ 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)

## S3 method for class 'GKgamma'
print(x, digits = 3, ...)
```

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 $\gamma \neq =$
`digits`	number to digits to use in the print method
`...`	other arguments (unused), for conformity with the print generic

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 discordant 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

Agresti, A. *Categorical Data Analysis*. John Wiley & Sons, 2002, pp. 57–59.

Goodman, L. A., & Kruskal, W. H. (1954). Measures of association for cross classifications. *Journal of the American Statistical Association*, 49, 732-764.

Goodman, L. A., & Kruskal, W. H. (1963). Measures of association for cross classifications III: Approximate sampling theory. *Journal of the American Statistical Association*, 58, 310-364.

See Also

`assocstats`, Kappa

Other association tests: `CMHtest()`, `HLtest()`, `woolf_test()`, `zero.test()`

Examples

```
data(JobSat)
GKgamma(JobSat)
```

Glass *British Social Mobility from Glass(1954)*

Description

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

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 Nonmanual & 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

Bishop, Y. M. M. and Fienberg, S. E. and Holland, P. W. (1975). *Discrete Multivariate Analysis: Theory and Practice*, MIT Press.

Friendly, M. and Meyer, D. (2016). *Discrete Data Analysis with R: Visualization and Modeling Techniques for Categorical and Count Data*. Boca Raton, FL: Chapman & Hall/CRC. http://ddar.datavis.ca.

Examples

```
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.

Usage

```
glmlist(...)

loglmlist(...)

## 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 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.

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.

Other glmlist functions: Kway(), LRstats(), get_model(), mosaic.glmlist()

Other loglinear models: assoc_graph(), get_model(), joint(), plot.assoc_graph(), seq_loglm()

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.

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*, 2nd E., John Wiley & Sons, 2002, Table 3.11, p. 106.

Examples

```
data(GSS)
str(GSS)

# use xtabs to show the table in a compact form
(GSStab <- xtabs(count ~ sex + party, data=GSS))

# fit the independence model
(mod.glm <- glm(count ~ sex + party, family = poisson, data = GSS))

# display all the residuals in a mosaic plot
mosaic(mod.glm,
  formula = ~ sex + party,
  labeling = labeling_residuals,
  suppress=0)
```

HairEyePlace *Hair Color and Eye Color in Caithness and Aberdeen*

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.

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
..$ Eye  : chr [1:4] "Blue" "Light" "Medium" "Dark"
..$ Hair : chr [1:5] "Fair" "Red" "Medium" "Dark" ...
..$ Place: chr [1:2] "Caithness" "Aberdeen"
```

Details

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).

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

Becker, M. P., and Clogg, C. C. (1989). Analysis of Sets of Two-Way Contingency Tables Using Association Models. *Journal of the American Statistical Association*, 84(405), 142-151.

Fisher, R.A. (1940) The precision of discriminant functions. *Annals of Eugenics*, 10, 422-429.

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)
```

`Hauser79` *Hauser (1979) Data on Social Mobility*

Description

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

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

Details

It is a good example for exploring a variety of models for square tables: quasi-independence, quasi-symmetry, row/column effects, uniform association, etc., using the facilities of the **gnm**.

Hauser's data was first presented in 1979, and then published in 1980. The name of the dataset reflects the earliest use.

It reflects the "frequencies in a classification of son's first full-time civilian occupation by father's (or other family head's) occupation at son's sixteenth birthday among American men who were aged 20 to 64 in 1973 and were not currently enrolled in school".

As noted in Hauser's Table 1, "Counts are based on observations weighted to estimate population counts and compensate for departures of the sampling design from simple random sampling. Broad occupation groups are upper nonmanual: professional and kindred workers, managers and officials, and non-retail sales workers; lower nonmanual: proprietors, clerical and kindred workers, and retail sales workers; upper manual: craftsmen, foremen, and kindred workers; lower manual: service workers, operatives and kindred workers, and laborers (except farm); farm: farmers and farm managers, farm laborers, and foremen. density of mobility or immobility in the cells to which they refer."

The table levels for `Son` and `Father` have been arranged in order of decreasing status as is common for mobility tables.

Source

R.M. Hauser (1979), Some exploratory methods for modeling mobility tables and other cross-classified data. In: K.F. Schuessler (Ed.), *Sociological Methodology*, 1980, Jossey-Bass, San Francisco, pp. 413-458. Table 1.

References

Powers, D.A. and Xie, Y. (2008). *Statistical Methods for Categorical Data Analysis*, Bingley, UK: Emerald.

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)

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

# Quasi-symmetry
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)

# RC model
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)
sumry[order(sumry$AIC, decreasing=TRUE),]
# 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

Format

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

No	Name	Levels
1	`Disease`	`"Disease", "None"`
2	`Gender`	`"Male", "Female"`
3	`Occup`	`"Unempl", "WhiteCol", "BlueCol"`

Source

% Karger, 1980 Karger, (1980).

Examples

```
data(Heart)
str(Heart)

# Display the frequencies for occupational categories.
# Each row is a 2 x 2 table
vcd::structable(Disease + Gender ~ Occup, data=Heart)

# display as fourfold plots
vcd::cotabplot(~ Disease + Gender | Occup, data=Heart, panel = cotab_fourfold)
```

`Heckman` *Labour Force Participation of Married Women 1967-1971*

Description

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

Format

A 5-dimensional 2^5 array resulting from cross-tabulating 5 binary variables for 1583 observations. The variable names and their levels are:

No	Name	Levels
1	`e1971`	`"71Yes", "No"`
2	`e1970`	`"70Yes", "No"`
3	`e1969`	`"69Yes", "No"`
4	`e1968`	`"68Yes", "No"`
5	`e1967`	`"67Yes", "No"`

Details

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.

Lindsey (1993) fits an initial set of logistic regression models examining the dependence of employment in 1971 (`e1971`) on successive subsets of the previous years, `e1970`, `e1969`, ... `e1967`.

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

Source

Lindsey, J. K. (1993). *Models for Repeated Measurements* Oxford, UK: Oxford University Press, p. 185.

References

% HeckmanWillis:77 Heckman, J.J. & Willis, R.J. (1977). "A beta-logistic model for the analysis of sequential labor force participation by married women." *Journal of Political Economy*, 85: 27-58

Examples

```
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
(heckman.mod1 <- loglm(~ e1971*e1970 + e1970*e1969 +e1969*e1968 + e1968*e1967, data=Heckman))
mosaic(heckman.mod1, main="1st order markov chain model")

# second-order markov chain: bad fit
(heckman.mod2 <- loglm(~ e1971*e1970*e1969 + e1970*e1969*e1968 +e1969*e1968*e1967, data=Heckman))
mosaic(heckman.mod2, main="2nd order markov chain model")

# third-order markov chain: fits OK
(heckman.mod3 <- loglm(~ e1971*e1970*e1969*e1968 + e1970*e1969*e1968*e1967, data=Heckman))
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

Usage

```
HLtest(model, g = 10)

## S3 method for class 'HLtest'
print(x, ...)

## S3 method for class 'HLtest'
summary(object, ...)

## S3 method for class 'HLtest'
rootogram(x, ...)

## S3 method for class 'HLtest'
plot(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

Details

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.

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 chisquare statistics
`df`	Degrees of freedom
`p.value`	p value
`groups`	Number of groups

`call`	model call

%% ...

Author(s)

Michael Friendly

References

Hosmer, David W., Lemeshow, Stanley (1980). A goodness-of-fit test for multiple logistic regression model. *Communications in Statistics, Series A*, 9, 1043-1069.

Hosmer, David W., Lemeshow, Stanley (2000). *Applied Logistic Regression*, New York: Wiley, ISBN 0-471-61553-6

Lemeshow, S. and Hosmer, D.W. (1982). A review of goodness of fit statistics for use in the development of logistic regression models. *American Journal of Epidemiology*, 115(1), 92-106.

See Also

`rootogram`, ~~~

Other association tests: `CMHtest()`, `GKgamma()`, `woolf_test()`, `zero.test()`

Examples

```
data(birthwt, package="MASS")
# how to do this without attach?
attach(birthwt)
race = factor(race, labels = c("white", "black", "other"))
ptd = factor(ptl > 0)
ftv = factor(ftv)
levels(ftv)[-(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)
```

HospVisits *Hospital Visits Data*

Description

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

Format

A 3 by 3 frequency table, with format:

```
table [1:3, 1:3] 43 6 9 16 11 18 3 10 16
- attr(*, "dimnames")=List of 2
..$ visit: chr [1:3] "Regular" "Infrequent" "Never"
..$ stay : chr [1:3] "2-9" "10-19" "20+"
```

Details

Both table variables can be considered ordinal. The variable `visit` refers to visiting patterns recorded 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.

Source

Goodman, L. A. (1983) The analysis of dependence in cross-classifications having ordered categories, using log-linear models for frequencies and log-linear models for odds. *Biometrics*, 39, 149-160.

References

Wing, J. K. (1962). Institutionalism in Mental Hospitals, *British Journal of Social and Clinical Psychology*, 1 (1), 38-51.

See Also

`ca`

Examples

```
data(HospVisits)
mosaic(HospVisits, gp=shading_Friendly)

if(require(ca)){
  ca(HospVisits)
  # surprisingly 1D !
  plot(ca(HospVisits))
  }
```

HouseTasks *Household Tasks Performed by Husbands and Wives*

Description

A 13 x 4 table of frequencies of household tasks performed by couples, either by the Husband, Wife, Alternating or Jointly.

Format

The format is:

```
  'table' int [1:13, 1:4] 36 11 24 51 13 1 1 14 20 46 ...
- attr(*, "dimnames")=List of 2
..$ Task: chr [1:13] "Breakfast" "Dinner" "Dishes" "Driving" ...
..$ Who : chr [1:4] "Alternating" "Husband" "Jointly" "Wife"
```

Source

This data set was taken from housetasks, a 13 x 4 data.frame. In this table version, the rows and columns were sorted alphabetically (and a typo was corrected).

Examples

```
data(HouseTasks)
str(HouseTasks)

chisq.test(HouseTasks)

# mosaic plot, illustrating some tweaks to handle overlapping labels
require(vcd)
mosaic(HouseTasks, shade = TRUE,
       labeling = labeling_border(rot_labels = c(45,0, 0, 0),
                                  offset_label =c(.5,5,0, 0),
                                  varnames = c(FALSE, TRUE),
                                  just_labels=c("center","right"),
                                  tl_varnames = FALSE),
       legend = FALSE)
```

```
# use seriation package to permute rows & cols using correspondence analysis
if(require(seriation)) {
order <- seriate(HouseTasks, method = "CA")
# the permuted row and column labels
rownames(HouseTasks)[order[[1]]]
colnames(HouseTasks)[order[[2]]]

# do the permutation
HT_perm <- permute(HouseTasks, order, margin=1)

mosaic(HT_perm, shade = TRUE,
       labeling = labeling_border(rot_labels = c(45,0, 0, 0),
                                  offset_label =c(.5,5,0, 0),
                                  varnames = c(FALSE, TRUE),
                                  just_labels=c("center","right"),
                                  tl_varnames = FALSE),
       legend = FALSE)
}
```

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).

Format

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

No	Name	Levels
1	`Status`	`"College", "School", "Job", "Other"`
2	`Rank`	`"Low", "Middle", "High"`
3	`Occupation`	`"1", "2", "3", "4", "5", "6", "7"`
4	`Sex`	`"Male", "Female"`

Details

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

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

Source

Fienberg, S. E. (1980). *The Analysis of Cross-Classified Categorical Data.* Cambridge, MA: MIT Press, p. 91-92.

R. L. Plackett, (1974). *The Analysis of Categorical Data.* London: Griffin.

References

Hoyt, C. J., Krishnaiah, P. R. and Torrance, E. P. (1959) Analysis of complex contingency tables, *Journal of Experimental Education* 27, 187-194.

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 *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).

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` Service at ICU Admission, a factor with levels `Medical` `Surgical`

`cancer` Cancer part of present problem?, a factor with levels `No` `Yes`

`renal` History of chronic renal failure?, a factor with levels `No` `Yes`

`infect` Infection probable at ICU admission?, a factor with levels `No` `Yes`

`cpr` Patient received CPR prior to ICU admission?, a factor with levels `No` `Yes`

`systolic` Systolic blood pressure at admission (mm Hg), a numeric vector

`hrtrate` Heart rate at ICU Admission (beats/min), a numeric vector

`previcu` Previous admission to an ICU within 6 Months?, a factor with levels `No` `Yes`

`admit` Type of admission, a factor with levels `Elective` `Emergency`

`fracture` Admission with a long bone, multiple, neck, single area, or hip fracture? a factor with levels `No` `Yes`

`po2` PO2 from initial blood gases, a factor with levels `>60` `<=60`

`ph` pH from initial blood gases, a factor with levels `>=7.25` `<7.25`

`pco` PCO2 from initial blood gases, a factor with levels `<=45` `>45`

`bic` Bicarbonate (HCO3) level from initial blood gases, a factor with levels `>=18` `<18`

`creatin` Creatinine, from initial blood gases, a factor with levels `<=2` `>2`

`coma` Level of unconsciousness at admission to ICU, a factor with levels `None` `Stupor` `Coma`

`white` a recoding of `race`, a factor with levels `White` `Non-white`

`uncons` a recoding of `coma` a factor with levels `No` `Yes`

Details

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.

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

M. Friendly (2000), *Visualizing Categorical Data*, Appendix B.4. SAS Institute, Cary, NC.

Hosmer, D. W. Jr., Lemeshow, S. and Sturdivant, R. X. (2013) *Applied Logistic Regression*, NY: Wiley, Third Edition.

References

Lemeshow, S., Teres, D., Avrunin, J. S., Pastides, H. (1988). Predicting the Outcome of Intensive Care Unit Patients. *Journal of the American Statistical Association*, 83, 348-356.

Examples

```
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[,2] <- ifelse(risks[,2]=="Emergency", "Emerg", "")
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")
```

JobSat *Cross-classification of job satisfaction by income*

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.

Format

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

```
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

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

`joint` *Loglinear Model Utilities*

Description

These functions generate lists of terms to specify a loglinear model in a form compatible with `loglin` and also provide for conversion to an equivalent `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

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

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

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

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

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

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

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

Arguments

`nf`	number of factors for which to generate model
`table`	a contingency table used for factor names, typically the output from `table`
`factors`	names of factors used in the model when `table` is not specified
`with`	indices of the factors against which others are considered conditionally independent
`order`	order of the markov chain
`x`	a list of terms in a loglinear model, such as returned by `joint`, `conditional`, ...
`env`	environment in which to evaluate the formula
`brackets`	characters to use to surround model terms. Either a single character string containing two characters or a character vector of length two.
`sep`	characters used to separate factor names within a term
`collapse`	characters used to separate terms
`abbrev`	Unused as yet

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.

function	**3-way**	**4-way**	**5-way**
`mutual`	[A] [B] [C]	[A] [B] [C] [D]	[A] [B] [C] [D] [E]
`joint`	[AB] [C]	[ABC] [D]	[ABCE] [E]
`joint (with=1)`	[A] [BC]	[A] [BCD]	[A] [BCDE]
`conditional`	[AC] [BC]	[AD] [BD] [CD]	[AE] [BE] [CE] [DE]
`condit (with=1)`	[AB] [AC]	[AB] [AC] [AD]	[AB] [AC] [AD] [AE]
`markov (order=1)`	[AB] [BC]	[AB] [BC] [CD]	[AB] [BC] [CD] [DE]
`markov (order=2)`	[A] [B] [C]	[ABC] [BCD]	[ABC] [BCD] [CDE]
`saturated`	[ABC]	[ABCD]	[ABCDE]

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

Source

Code from Henrique Dallazuanna, wwwhsd@gmail.com, R-help 7-4-2013

See Also

Other loglinear models: `assoc_graph()`, `get_model()`, `glmlist()`, `plot.assoc_graph()`, `seq_loglm()`

`knit_include` *Include an HTML-renderable object in any knitr output format*

Description

A pipe-friendly helper in Rmarkdown, Quarto, and other documents for objects that render natively in HTML output (RStudio Viewer, HTML documents) but need a convenient image fallback for PDF, Word, and other non-HTML formats. It was designed to solve a problem in rendering the result of `color_table()`, and then generalized to provide for other similar cases.

Usage

```
knit_include(
  x,
  file = tempfile("ki_", tmpdir = "."),
  width = 700,
  height = 400,
  ...
)
```

Arguments

`x`	Any R object. Specialised handling for `gt_tbl` and `htmlwidget`; all other classes are passed through unchanged.
`file`	Base filename (no extension) for temporary files written when output is not HTML. Defaults to a session-unique temp file in the current directory so multiple calls don't collide.
`width`	Viewport / screenshot width in pixels (non-HTML only).
`height`	Viewport / screenshot height in pixels (non-HTML only).
`...`	Additional arguments forwarded to `gtsave` or `webshot`.

Details

Supported classes (handled internally) solve this problem by saving the image to a file and then inserting it in the document using `knitr::include_graphics()`:

- `"gt_tbl"` (gt package) – saved via `gtsave`
- `"htmlwidget"` (htmlwidgets family: plotly, DT, leaflet, dygraphs, ...) – saved via `saveWidget` + `webshot`

Packages that handle cross-format output natively (**flextable**, **kableExtra**, **huxtable**) do NOT need this helper.

For any other class, `x` is returned as-is in all output formats and knitr's normal printing applies. This means the function is safe to use in a pipe on any object: it only intervenes when it knows how to help!

Value

For `gt_tbl` / `htmlwidget` in non-HTML output: the result of `include_graphics`. Otherwise: `x` unchanged.

Examples

```
## Not run:
data(HairEyeColor)
HEC <- margin.table(HairEyeColor, 1:2)

## gt_tbl from color_table()
color_table(HEC, title = "Hair x Eye Color") |> knit_include()

## DT htmlwidget
DT::datatable(mtcars) |> knit_include(width = 900, height = 500)

## plotly htmlwidget
plotly::plot_ly(mtcars, x = ~wt, y = ~mpg) |> knit_include()

## Any other object passes through unchanged
lm(mpg ~ wt, data = mtcars) |> knit_include()

## End(Not run)
```

`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.

Usage

```
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` to a 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

This function is designed mainly for hierarchical loglinear models (or `glms` in the poisson 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.

With `y` as the response in the `formula`, the 0-way (null) model is `y ~ 1`. The 1-way ("main effects") model is that specified in the `formula` argument. The k-way model is generated using the formula `. ~ .^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

Other glmlist functions: LRstats(), get_model(), glmlist(), mosaic.glmlist()

Examples

```
## 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)
```

labeling_points   *Labeling Function for Mosaic Displays Using Points*

Description

This labeling function for use with strucplot displays, such as mosaic, draws random points within each cell of a mosaic plot, where the number of points represents observed or expected frequencies. This creates a "dot-density" visualization that provides a direct visual representation of cell frequencies.

Usage

```
labeling_points(
  labels = TRUE,
  varnames = labels,
  value_type = c("observed", "expected"),
  scale = 1,
  pch = 19,
  size = unit(0.5, "char"),
  gp_points = gpar(col = "black", alpha = 0.7),
  margin = unit(0.05, "npc"),
  seed = NULL,
  jitter = 1,
  clip = FALSE,
  ...
)
```

Arguments

`labels`	Logical vector or scalar indicating whether labels should be drawn for the table dimensions via `labeling_border`. Defaults to `TRUE`.
`varnames`	Logical vector or scalar indicating whether variable names should be drawn. Defaults to `labels`.
`value_type`	Character string specifying whether to display `"observed"` or `"expected"` frequencies as points.
`scale`	Numeric scaling factor. The number of points drawn equals `round(frequency / scale)`. Use larger values for tables with large counts.
`pch`	Point character (plotting symbol). Default is 19 (filled circle).
`size`	Point size as a `unit` object. Default is `unit(0.5, "char")`.
`gp_points`	A `gpar` object controlling point appearance (color, alpha, etc.), for example: `gp_points = gpar(col = "red")`.
`margin`	Margin inside cells as a `unit` object. Points are drawn within this inset area. Default is `unit(0.05, "npc")`.
`seed`	Optional integer seed for reproducible point placement across several similar plots. If `NULL` (default), no seed is set.
`jitter`	Numeric jitter amount (0-1) for point placement. Default is 1 (full random). Values < 1 create more regular patterns.
`clip`	Logical indicating whether to clip points at cell boundaries. Default is `FALSE`.
`...`	Additional arguments passed to `labeling_border` for axis labels.

Details

This function follows the "grapcon_generator" pattern used by vcd labeling functions. It returns a function that can be passed to the `labeling` argument of `strucplot`, `mosaic`, or related functions as the argument `labeling = labeling_poionts(...)`.

The visualization is inspired by the conceptual model described in Friendly (1995), where cell frequencies are represented as physical counts of objects. Under independence, point density would be uniform across cells (adjusted for marginals), so departures from independence become visible as density variations.

This approach is related to sieve diagrams (`sieve`), which also use density to represent association.

Value

A function of class `"grapcon_generator"` suitable for use as the `labeling` argument in `strucplot` and related functions like `mosaic`.

References

Friendly, M. (1995). Conceptual and Visual Models for Categorical Data. *The American Statistician*, **49**, 153-160. doi:10.1080/00031305.1995.10476131

Friendly, M. (1997). Conceptual Models for Visualizing Contingency Table Data. In M. Greenacre & J. Blasius (Eds.), *Visualization of Categorical Data* (pp. 17–35). Academic Press. `https://www.datavis.ca/papers/koln/kolnpapr.pdf`

Meyer, D., Zeileis, A., and Hornik, K. (2006). The Strucplot Framework: Visualizing Multi-way Contingency Tables with vcd. *Journal of Statistical Software*, **17**(3), 1-48. doi:10.18637/jss.v017.i03

See Also

`labeling_cells`, `labeling_border`, `mosaic`, `sieve`

Examples

```
library(vcd)

# 2-way table of Hair and Eye color
HairEye <- margin.table(HairEyeColor, 2:1)

# Basic usage - observed frequencies as points
mosaic(HairEye,
       labeling = labeling_points(scale = 1))

# Show expected frequencies instead of observed
mosaic(HairEye,
       labeling = labeling_points(
                    value_type = "expected",
                    scale = 2,
                    seed = 42)
      )

# Combine with residual shading
mosaic(HairEye,
       shade = TRUE, legend = FALSE,
       labeling = labeling_points(scale = 2, seed = 42))
```

```
# Make tiles show expected frequencies, show points for observed frequencies
# Reproduces: Fig 6 in Friendly (1995)
mosaic(HairEye,
       type = "expected",
       shade = TRUE, legend = FALSE,
       labeling = labeling_points(scale = 2, seed = 42))
```

`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.

Usage

```
## 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

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

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 `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

loglm, AIC, 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.glm1)
AIC(titanic.glm1, titanic.glm2, titanic.glm3)
BIC(titanic.glm1, titanic.glm2, titanic.glm3)
```

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.

Usage

```
LRstats(object, ...)

## S3 method for class 'glmlist'
LRstats(
  object,
  ...,
  saturated = NULL,
  sortby = NULL,
  label = c("name", "formula"),
  label.args = list()
)

## S3 method for class 'loglmlist'
LRstats(
  object,
  ...,
  saturated = NULL,
  sortby = NULL,
  label = c("name", "formula"),
  label.args = list()
)

## Default S3 method:
LRstats(
  object,
  ...,
  saturated = NULL,
  sortby = NULL,
  label = c("name", "formula"),
  label.args = list()
)
```

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)
`label`	character string specifying how to label the rows: `"name"` (default) uses the model object names; `"formula"` uses model formulas or bracket notation obtained from `get_models` (for `glmlist` and `loglmlist` objects) or `get_model` (for individual model objects passed to the default method).
`label.args`	a list of additional arguments passed to `get_models` when `label = "formula"`. Useful arguments include `abbrev` (logical or integer) to abbreviate factor names and `sep` to change the separator in bracket notation.

Details

`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).

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) or their model formulas.

Author(s)

Achim Zeileis, Michael Friendly

See Also

`logLik`, `glm`, `loglm`,

`logLik.loglm`, `modFit`, `get_models`, `get_model`

Other glmlist functions: `Kway()`, `get_model()`, `glmlist()`, `mosaic.glmlist()`

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 using object names (default)
LRstats(indep, coleff, roweff, linlin)

# compare models in a glmlist, using formula labels
mods <- glmlist(indep, coleff, roweff, linlin)
LRstats(mods, label = "formula")

# loglmlist example with bracket notation labels
data(Titanic)
tit.joint <- seq_loglm(Titanic, type = "joint")
LRstats(tit.joint)
LRstats(tit.joint, label = "formula")
LRstats(tit.joint, label = "formula", label.args = list(abbrev = TRUE))
```

Mammograms *Mammogram Ratings*

Description

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

Format

A frequency table in matrix form. The format is:

```
  num [1:4, 1:4] 34 6 2 0 10 8 5 1 2 8 ...
- attr(*, "dimnames")=List of 2
..$ Reader2: chr [1:4] "Absent" "Minimal" "Moderate" "Severe"
..$ Reader1: chr [1:4] "Absent" "Minimal" "Moderate" "Severe"
```

Source

Kundel, H. L. & Polansky, M. (2003), "Measurement of Observer Agreement", *Radiology*, **228**, 303-308, Table A1

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

```
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` *Mental Impairment and Parents SES*

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`).

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

Haberman, S. J. *The Analysis of Qualitative Data: New Developments*, Academic Press, 1979, Vol. II, p. 375.

Srole, L.; Langner, T. S.; Michael, S. T.; Kirkpatrick, P.; Opler, M. K. & Rennie, T. A. C. *Mental Health in the Metropolis: The Midtown Manhattan Study*, NYU Press, 1978, p. 289

References

Friendly, M. *Visualizing Categorical Data*, Cary, NC: SAS Institute, 2000, Appendix B.7.

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)

if(require(ca)){
  plot(ca(Mental.tab), main="Mental impairment & SES", lines=TRUE)
}
```

`Mice` *Mice Depletion Data*

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.

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

Goodman, L. A. (1983) The analysis of dependence in cross-classifications having ordered categories, using log-linear models for frequencies and log-linear models for odds. *Biometrics*, 39, 149-160.

References

Kastenbaum, M. A. & Lamphiear, D. E. (1959) Calculation of chi-square to calculate the no three-factor interaction hypothesis. *Biometrics*, 15, 107-115.

Examples

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

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

`Mobility` *Social Mobility data*

Description

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

Format

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

No	Name	Levels
1	`Son's_Occupation`	`"UpNonMan", "LoNonMan", "UpManual", "LoManual", "Farm"`
2	`Father's_Occupation`	`"UpNonMan", "LoNonMan", "UpManual", "LoManual", "Farm"`

Source

Falguerolles, A. de and Mathieu, J. R. (1988). *Proceedings of COMPSTAT 88*, Copenhagen, Denmark, Springer-Verlag.

% FeathermanHauser:78

Featherman, D. L. and Hauser, R. M. Occupations and social mobility in the United States. *Sociological Microjournal*, 12, Fiche 62. Copenhagen: Sociological Institute.

See Also

`Glass`, `Hauser79`, `Yamaguchi87` for other examples of mobility data.

Examples

```
data(Mobility)
Mobility

# independence model
MASS::loglm(~Father_Occupation + Son_Occupation, data = Mobility)

vcd::mosaic(Mobility, shade=TRUE, legend = FALSE)
```

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
`...`	Arguments passed down
`stats`	statistics to print: one or more of `"chisq"`, `"aic"`
`digits`	number to digits to use in the print method

Value

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

Author(s)

Michael Friendly

See Also

`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.glm` *Mosaic plots for fitted generalized linear and generalized nonlinear models*

Description

Produces 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`.

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 `"rstandard"`. 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

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.

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`

Other mosaic plots: `mosaic.glmlist()`, `mosaic3d()`

Examples

```
library(vcdExtra)

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\nresiduals",
       labeling = labeling_residuals)


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

# Using residuals_type
mosaic(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,
       formula = ~ses + mental,
       residuals_type="rstandard",
       labeling_args = long.labels,
       labeling=labeling_residuals)

# or, show as a sieve diagram
```

```
mosaic(indep,
       formula = ~ses + mental,
       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,
       formula = ~ses + mental,
       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,
       formula = ~ses + mental,
       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(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(berk.glm3,
       residuals_type = "rstandard",
       labeling = labeling_residuals,
       shade=TRUE,
      formula = ~Admit+Dept+Gender,
      main = "Model: [DeptGender][DeptAdmit] + DeptA*[GA]")

# compare models
anova(berk.glm1, berk.glm3, test="Chisq")
```

mosaic.glmlist *Mosaic Displays for glmlist and loglmlist 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`
`type`	a character string indicating whether the `"observed"` or the `"expected"` values of the table should be visualized
`legend`	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.
`main`	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.

`ask`	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.
`graphics`	logical: use a graphic dialog box when `ask=TRUE`?
`rows, cols`	when `ask=FALSE`, the number of rows and columns in which to plot the mosaics.
`newpage`	start a new page? (only applies to `ask=FALSE`)
`...`	other arguments passed to `mosaic.glm` and ultimately to `mosaic`.

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

David Meyer, Achim Zeileis, and Kurt Hornik (2006). The Strucplot Framework: Visualizing Multi-Way Contingency Tables with vcd. *Journal of Statistical Software*, 17(3), 1-48. `https://www.jstatsoft.org/v17/i03/`, available as `vignette("strucplot", package="vcd")`.

See Also

`glmlist`, `loglmlist`, `Kway`

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

Other mosaic plots: `mosaic.glm()`, `mosaic3d()`

Other glmlist functions: `Kway()`, `LRstats()`, `get_model()`, `glmlist()`

Examples

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

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

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)
roweff <- 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=roweff, `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`.

Usage

```
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 = 0.05,
  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.
`...`	Other arguments passed down to `mosaic.default` or 3D functions.
`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.
`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).
`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 re-cycled 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.

Details

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.

Friendly (1995), Friendly (2000, Sect. 4.5) 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

Friendly, M. (1995). Conceptual and Visual Models for Categorical Data, *The American Statistician*, **49**, 153-160.

Friendly, M. *Visualizing Categorical Data*, Cary NC: SAS Institute, 2000. Web materials: http://www.datavis.ca/books/vcd/.

Theus, M. & Lauer, S. R. W. (1999) Visualizing Loglinear Models. *Journal of Computational and Graphical Statistics*, **8**, 396-412.

See Also

`strucplot`, `mosaic`, `mosaicplot` for aspects of mosaic plots.

`loglin`, `loglm` for details on fitting loglinear models

`play3d`, `movie3d` for what you can do with a 3D mosaic plot

Other mosaic plots: `mosaic.glm()`, `mosaic.glmlist()`

Examples

```
# 2 x 2 x 2
if (!rgl::rgl.useNULL() && interactive()){
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)
}

# Make a dynamic display
## Not run:
  mosaic3d(Heart, box=TRUE, alpha = 0.3, interpolate = c(1, 2, 4))
  play3d(spin3d(axis = c(0, 1, 0), rpm = 10), duration = 5)

## End(Not run)

## 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)
```

pairs_diagonal_mosaic *Diagonal Panel Function for Pairs Plots of Contingency Tables*

Description

An enhanced replacement for `pairs_diagonal_mosaic` from the **vcd** package. This version fixes two bugs in the original (the `labeling` and `alternate_labels` arguments were hardcoded and ignored) and changes the default labeling to `labeling_border`.

Usage

```
pairs_diagonal_mosaic(
  split_vertical = TRUE,
  margins = unit(0, "lines"),
  offset_labels = -0.4,
  offset_varnames = 0,
  gp = NULL,
  fill = "grey",
  labeling = labeling_border,
  alternate_labels = TRUE,
  ...
)

## S3 method for class 'table'
pairs(x, diag_panel = pairs_diagonal_mosaic, ...)
```

Arguments

`split_vertical`	Logical; passed to `mosaic`. Default is `TRUE`.
`margins`	A `unit` object giving the margins around the mosaic. Default is `unit(0, "lines")`.
`offset_labels`	Numeric; offset for the cell labels. Default is `-0.4`.
`offset_varnames`	Numeric; offset for the variable name labels. Default is `0`.
`gp`	A `gpar` object for the mosaic tiles, or `NULL` (default) to use `fill`.
`fill`	Either a color name or a function mapping an integer to a vector of colors (e.g. a shading function). Default is `"grey"`.
`labeling`	A labeling function such as `labeling_border` or `labeling_values`. Default is `labeling_border` (no cell counts shown).
`alternate_labels`	Logical; whether to alternate label positions on the axes. Default is `TRUE`.
`...`	Additional arguments passed to the **vcd** `pairs.table` method.
`x`	A contingency table (object of class `"table"`).
`diag_panel`	A `"grapcon_generator"` function or a panel function for the diagonal cells. Defaults to the vcdExtra version of `pairs_diagonal_mosaic`.

Details

A companion `pairs.table` method is also provided that uses this improved function as the default diagonal panel.

The function follows the `"grapcon_generator"` pattern used throughout **vcd**: calling `pairs_diagonal_mosaic(...)` returns a *function* suitable for passing as the `diag_panel` argument of `pairs.table`.

The original `vcd::pairs_diagonal_mosaic` has two bugs: (1) the `labeling` argument is accepted but then hardcoded to `labeling_values` inside the returned function, and (2) `alternate_labels` is similarly hardcoded to `TRUE`. This version fixes both, making those arguments work as documented.

The default labeling scheme was changed from `labeling_values` to `labeling_border`, so as to disable cell counts by default. To enable cell counts, use `labeling_values`.

`pairs.table` is an S3 method for `pairs` that overrides the version in **vcd** solely to change the default `diag_panel` to the improved `pairs_diagonal_mosaic` defined in this package. All actual rendering is delegated to the **vcd** implementation, retrieved via `utils::getFromNamespace()` to avoid a `:::` check NOTE.

Value

A function of class `"grapcon_generator"` suitable for use as the `diag_panel` argument in `pairs.table`.

See Also

`pairs.table`, `pairs_diagonal_mosaic`, `labeling_border`, `labeling_values`

Examples

```
data(Hoyt, package = "vcdExtra")
pred_tab <- margin.table(Hoyt, c("Rank", "Occupation", "Sex"))

# Default: no cell counts in diagonal panels
pairs(pred_tab)

# Show cell counts in diagonal panels
pairs(pred_tab, diag_panel_args = list(labeling = labeling_values))

# Use residual shading for off-diagonal panels
pairs(pred_tab, gp = shading_Friendly)
```

`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).

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 preceeding three years

Details

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

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

Long, J. S. (1997). *Regression Models for Categorical and Limited Dependent Variables*, Sage.

Long, J. S. & Freese, J. (2006). *Regression Models for Categorical Dependent Variables Using Stata*, 2nd Ed., Stata Press.

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")
```

plot.assoc_graph *Plot an Association Graph*

Description

Plot method for assoc_graph objects, displaying the association structure of a loglinear model as a network diagram.

Usage

```
## S3 method for class 'assoc_graph'
plot(
  x,
  layout = NULL,
  groups = NULL,
  colors = c("lightblue", "lightyellow", "lightgreen", "lightsalmon", "plum"),
  vertex.size = 30,
  vertex.label.cex = 1.2,
  edge.width = 2,
  edge.label = NULL,
  ...
)
```

Arguments

`x`	An `assoc_graph` object, as returned by `assoc_graph`.
`layout`	Layout function or coordinate matrix for node positions. Defaults to `layout_in_circle` for up to 6 nodes, `layout_with_fr` otherwise.
`groups`	Optional named list assigning variables to groups for coloring, e.g., `list(response = "Survived", predictors = c("Class", "Sex", "Age"))`.
`colors`	Character vector of colors for groups. Recycled as needed.
`vertex.size`	Vertex size (default 30).
`vertex.label.cex`	Label size for vertex names (default 1.2).
`edge.width`	Edge width (default 2). If edge weights are present, widths are scaled from the weights automatically.
`edge.label`	Optional edge labels. If `TRUE` and edge weights are present, the weight values are used as labels.
`...`	Additional arguments passed to `plot.igraph`, such as `main` for a title.

Value

The `assoc_graph` object `x`, returned invisibly.

See Also

assoc_graph, plot.igraph

Other loglinear models: assoc_graph(), get_model(), glmlist(), joint(), seq_loglm()

Examples

```
# Basic structural plot
g <- conditional(3, factors = c("A", "B", "C")) |> assoc_graph()
plot(g, main = "Conditional independence: [AC] [BC]")

# With grouped node colors
g <- saturated(4, factors = c("A", "B", "C", "D")) |> assoc_graph()
plot(g, groups = list(c("A", "B"), c("C", "D")),
     main = "Saturated model")
```

---

print.Kappa *Print Kappa*

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)
```

`Reinis` *Risk Factors for Coronary Heart Disease*

Description

Data from the beginning of a 15-year follow-up study of probable risk factors for coronary thrombosis. The data are from 1841 men employed in a Czechoslovakian car factory and represent all possible combinations of the six risk factors.

Format

A 6-dimensional array resulting from cross-tabulating 6 binary variables for 1841 observations. The variable names and their levels are:

No	Name	Levels (meaning)
1	`smoke`	`"y", "n"` (smoking)
2	`mental`	`"y", "n"` (strenuous mental work)
3	`phys`	`"y", "n"` (strenuous physical work)
4	`systol`	`"y", "n"` (systolic blood pressure > 140)
5	`protein`	`"y", "n"` (ratio of beta to alpha lipoproteins > 3)
6	`family`	`"y", "n"` (family history of CHD)

Details

The study was conducted to examine risk factors for coronary heart disease (CHD) and collected information on smoking habits, mental and physical work strain, systolic blood pressure, ratio of lipoproteins, and family history of CHD.

The six variables form a $2^6 = 64$ contingency table. The data have been used extensively to illustrate model search procedures for high-dimensional contingency tables.

Source

Originally from the **gRbase** package as `data(reinis, package = "gRbase")`

References

Edwards, D. and Havranek, T. (1985). A fast procedure for model search in multidimensional contingency tables. *Biometrika*, 72, 339-351.

Gauraha, N. and Parui, S. K. (2020). Mutual conditional independence and its applications to model selection in Markov networks. *Annals of Mathematics and Artificial Intelligence*, 88, 951-972. doi:10.1007/s10472020096907

Reinis, Z., Pokorny, J., Basika, V., Tiserova, J., Gorican, K., Horakova, D., Stuchlikova, E., Havranek, T., and Hrabovsky, F. (1981). Prognostic significance of the risk profile in the prevention of coronary heart disease. *Bratis. lek. Listy*, 76, 137-150.

Examples

```
data(Reinis)
str(Reinis)
ftable(Reinis, row.vars = 1:3)

# Fit all 0-way through 6-way models using Kway()
Reinis.gmodels <- Kway(Freq ~ smoke + mental + phys + systol + protein + family,
                       data = Reinis, family = poisson)

# Examine fit statistics
LRstats(Reinis.gmodels)

# Sequential tests for k vs k+1 way effects
anova(Reinis.gmodels, test = "Chisq")

# Fit sequential models of joint independence
Reinis.seqjoint <- seq_loglm(Reinis, type = "joint", prefix="joint")
LRstats(Reinis.seqjoint)

# Fit sequential models of conditional independence
Reinis.seqcond <- seq_loglm(Reinis, type = "conditional", prefix = "cond")
LRstats(Reinis.seqcond)
```

`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 = NULL,
  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. If `NULL` (the default), uses an abbreviation of `type`: `"joint"`, `"cond"`, `"mutual"`, `"markov"`, or `"sat"`.
`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

See Also

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

loglmlist

Other loglinear models: assoc_graph(), get_model(), glmlist(), joint(), plot.assoc_graph()

Examples

```
data(Titanic, package="datasets")
# variables are in the order Class, Sex, Age, Survived

# Models of joint independence
tit.joint <- seq_loglm(Titanic, type = "joint")

# compare the models
LRstats(tit.joint)

#' # Models of conditional independence
tit.cond <- seq_loglm(Titanic, type = "conditional")

# compare the models
LRstats(tit.cond)
```

`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 frequency 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 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"`
`...`	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 for Mosaics*, `http://www.datavis.ca/mosaics/mosaics.pdf`

See Also

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

```
   \code{\link{mosaic.glm}}, \code{\link[vcd]{mosaic}},
 \code{\link[vcd]{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)
```

ShakeWords *Shakespeare's Word Type Frequencies*

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.

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

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.

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

Bradley Efron and Ronald Thisted (1976). Estimating the Number of Unseen Species: How Many Words Did Shakespeare Know? *Biometrika*, Vol. 63, No. 3, pp. 435-447, %http://www.jstor.org/stable/2335721

Examples

```
data(ShakeWords)
str(ShakeWords)

plot(sqrt(Freq) ~ count, data=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).

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

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.

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

See Also

mosaic3d

shapelist3d for the plotting of lists of shape3d objects.

Examples

```
if (!rgl::rgl.useNULL() && interactive()){
  open3d()
  cube <- cube3d(alpha=0.4)
  sl1 <- split3d(cube, c(.2, .3, .5), 1)
  col <- c("#FF000080", "#E5E5E580", "#0000FF80")
  shapelist3d(sl1, col=col)

  open3d()
  p <- matrix(c(.6, .4, .5, .5, .2, .8), nrow=2)
  sl2 <- split3d(sl1, p, 2)
  shapelist3d(sl2, col=col)
  }
```

`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.

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
`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

`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).

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, LRstats

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)
```

Titanicp *Passengers on the Titanic*

Description

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

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://hbiostat.org/data/repo/titanic.txt`, described in more detail in `https://hbiostat.org/data/repo/titanic`

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.

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

Details

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.

Source

Brown, P. J., Stone, J. and Ord-Smith, C. (1983), Toxaemic signs during pregnancy. *JRSS, Series C, Applied Statistics*, 32, 69-72

References

Friendly, M. (2000), *Visualizing Categorical Data*, SAS Institute, Cary, NC, Example 7.15.

Friendly, M. and Meyer, D. (2016). *Discrete Data Analysis with R: Visualization and Modeling Techniques for Categorical and Count Data*. Boca Raton, FL: Chapman & Hall/CRC. http://ddar.datavis.ca. Example 10.10.

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 *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.

Format

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

```
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

Friendly, M. and Meyer, D. (2016). *Discrete Data Analysis with R: Visualization and Modeling Techniques for Categorical and Count Data*. Boca Raton, FL: Chapman & Hall/CRC. http://ddar.datavis.ca.

Emerson, John W. Mosaic Displays in S-PLUS: A General Implementation and a Case Study. *Statistical Graphics and Computing Newsletter*, 1998, 9(1), 17–23, http://www.stat.yale.edu/~jay/R/mosaic/v91.pdf

Hartigan, J. A. & Kleiner, B. A Mosaic of Television Ratings. *The American Statistician*, 1984, 38, 32-35.

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

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

update(vietnam.tab, formula = ~ . -year)
```

`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.

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

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

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

Friendly, M. (2000), *Visualizing Categorical Data*, SAS Institute, Cary, NC, Example 7.9.

Examples

```
data(Vietnam)
## maybe str(Vietnam) ; plot(Vietnam) ...
```

`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.

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

Clogg, C. & Shockey, J. W. (1988). In Nesselroade, J. R. & Cattell, R. B. (ed.) Multivariate Analysis of Discrete Data, *Handbook of Multivariate Experimental Psychology*, New York: Plenum Press.

References

Agresti, A. (1990) *Categorical Data Analysis*, Table 4.12 New York: Wiley-Interscience.

Friendly, M. (2000) *Visualizing Categorical Data*, Example 7.5 Cary, NC: SAS Institute.

Examples

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

`woolf_test` *Woolf Test for Homogeneity of Odds Ratios*

Description

Test for homogeneity on $2 \times 2 \times k$ tables over strata (i.e., whether the log odds ratios are the same in all strata). Generalized to handle tables of any dimensionality beyond 3. For 4-dimensional tables, optionally provides a two-way decomposition of the homogeneity test into row effects, column effects, and residual.

Usage

```
woolf_test(x, decompose = FALSE)

## S3 method for class 'woolf_test'
print(x, ...)
```

Arguments

x	An object of class "woolf_test"
decompose	Logical. If TRUE and x is 4-dimensional (a $2 \times 2 \times R \times C$ table), the test is decomposed into row effects, column effects, and residual (interaction). Defaults to FALSE. Ignored for non-4-dimensional tables.
...	Additional arguments (currently unused)

Details

The Woolf test (Woolf, 1955) tests the hypothesis that the odds ratios θ_i are equal across all k strata. The test statistic is computed as the weighted sum of squared deviations of the log odds ratios from their weighted mean:

$$\chi^2_W = \sum_{i=1}^{k} w_i[\log(\theta_i) - \log(\bar{\theta}_w)]^2 = \sum_{i=1}^{k} w_i \log^2(\theta_i/\bar{\theta}_w)$$

where $\theta_i = (n_{11i}n_{22i})/(n_{12i}n_{21i})$ is the odds ratio in stratum i, and $\bar{\theta}_w$ is the weighted average odds ratio (computed as the exponential of the weighted mean of the log odds ratios).

The weights w_i are the inverse variances of the log odds ratios:

$$w_i = 1/\text{Var}(\log\theta_i) = 1/(1/n_{11i} + 1/n_{12i} + 1/n_{21i} + 1/n_{22i})$$

Under the null hypothesis of homogeneous odds ratios, χ^2_W follows a chi-squared distribution with $k-1$ degrees of freedom.

Decomposition for 4-way tables: For a $2\times2\times R\times C$ table, the strata form an $R\times C$ two-way layout with odds ratios θ_{ij} for row i and column j. This suggests that overall Woolf test of homogeneity can be decomposed into three components, conceptually analogous to a two-way ANOVA with one observation per cell:

$$\chi^2_{\text{W:Total}} = \chi^2_{\text{W:Rows}} + \chi^2_{\text{W:Cols}} + \chi^2_{\text{W:Residual}}$$

where:

- $\chi^2_{\text{W:Rows}}$ tests whether the odds ratios differ among row levels (pooling over columns), with $R-1$ df
- $\chi^2_{\text{W:Cols}}$ tests whether the odds ratios differ among column levels (pooling over rows), with $C-1$ df
- $\chi^2_{\text{W:Residual}}$ tests the row $\times$ column interaction (deviation from additivity on the log odds scale), with $(R-1)(C-1)$ df

The row effect test compares the marginal log odds ratios $\log\bar{\theta}_{i+}$ (pooled over columns) to the overall weighted mean. Similarly, the column effect test compares $\log\bar{\theta}_{+j}$ (pooled over rows). The residual tests whether the cell log odds ratios $\log\theta_{ij}$ are additive in the row and column effects.

Note: The two-way ANOVA-like decomposition for 4-dimensional tables appears to be a novel extension introduced in this package. The existing literature on the Woolf test (and related Breslow-Day test) treats strata as unstructured, testing only whether all k odds ratios are equal. This decomposition exploits the factorial structure of $R\times C$ strata to provide more detailed insight into the sources of heterogeneity.

Value

A list of class `"woolf_test"` (also inheriting from `"htest"`) containing the following components:

`statistic`	the chi-squared test statistic.
`parameter`	degrees of freedom of the approximate chi-squared distribution of the test statistic.
`p.value`	p-value for the test.
`method`	a character string indicating the type of test performed.
`data.name`	a character string giving the name of the data.
`or_vars`	names of the first two dimensions (the 2x2 table variables).
`strata_vars`	names of the stratifying variables (dimensions 3 and beyond).
`observed`	the observed log odds ratios.
`expected`	the expected log odds ratio under the null hypothesis (weighted mean).
`decomposed`	logical indicating if decomposition was performed.

When `decompose = TRUE` (only for 4-dimensional tables), additional components:

`rows`	list with statistic, df, p.value, observed, expected for row effects.
`cols`	list with statistic, df, p.value, observed, expected for column effects.
`residual`	list with statistic, df, p.value for residual (interaction).

References

Woolf, B. (1955). On estimating the relation between blood group and disease. *Annals of Human Genetics*, **19**, 251-253.

See Also

`mantelhaen.test`, `BreslowDayTest`

Other association tests: `CMHtest()`, `GKgamma()`, `HLtest()`, `zero.test()`

Examples

```
# 3-way tables
data(CoalMiners, package = "vcd")
woolf_test(CoalMiners)

data(Heart, package = "vcdExtra")
woolf_test(Heart)

# 4-way table without decomposition
data(Fungicide, package = "vcdExtra")
woolf_test(Fungicide)

# 4-way table with decomposition
woolf_test(Fungicide, decompose = TRUE)

# 4-way table, but need to rearrange dimensions
# How does association between `Preference` and `M_User` vary over strata?
data(Detergent, package = "vcdExtra")
dimnames(Detergent) |> names()
Detergent <- aperm(Detergent, c(3, 2, 1, 4))
woolf_test(Detergent)
```

`WorkerSat` *Worker Satisfaction Data*

Description

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

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

Originally from https://online.stat.psu.edu/stat504/Lesson10

References

Andersen, E. B. (1991) Statistical Analysis of Categorical Data, 2nd Ed., Springer-Verlag.

Examples

```
data(WorkerSat)

worker.tab <- xtabs(Freq ~ Worker + Super + Manage, data=WorkerSat)
fourfold(worker.tab)
mosaic(worker.tab, shade=TRUE)
```

`Yamaguchi87` *Occupational Mobility in Three Countries*

Description

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).

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

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.

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 semi-skilled 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

Yamaguchi, K. (1987). Models for comparing mobility tables: toward parsimony and substance, *American Sociological Review*, vol. 52 (Aug.), 482-494, Table 1

References

Goodman, L. A. and Hout, M. (1998). Statistical Methods and Graphical Displays for Analyzing How the Association Between Two Qualitative Variables Differs Among Countries, Among Groups, Or Over Time: A Modified Regression-Type Approach. *Sociological Methodology*, 28 (1), 175-230.

Xie, Yu (1992). The log-multiplicative layer effect model for comparing mobility tables. *American Sociological Review*, 57 (June), 380-395.

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

# Mutual independence
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)
##
```

`zero.test` *Score test for zero inflation in Poisson data*

Description

Carries out a simple score test (van den Broek, 1995) for excess zeros in an otherwise Poisson distribution of counts. It gives a χ^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:

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

. This test statistic has a χ^2_1 distribution.

Value

Returns invisibly a list of three elements:

`statistic`	Value of the test statistic
`df`	Degrees of freedom
`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`

Van den Broek, J. (1995). A Score Test for Zero Inflation in a Poisson Distribution. *Biometrics*, **51**(2), 738-743. https://www.jstor.org/stable/2532959

Yang, Zhao, James W. Hardin, and Cheryl L. Addy (2010). Score Tests for Zero-Inflation in Overdispersed Count Data. *Communications in Statistics - Theory and Methods* **39** (11) 2008-2030. DOI: 10.1080/03610920902948228

See Also

Other association tests: `CMHtest()`, `GKgamma()`, `HLtest()`, `woolf_test()`

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