

# Package ‘vcd’

February 21, 2012

**Version** 1.2-13

**Date** 2012-02-19

**Title** Visualizing Categorical Data

**Author** David Meyer [aut, cre], Achim Zeileis [aut], Kurt Hornik [aut], Michael Friendly [ctb]

**Maintainer** David Meyer <David.Meyer@R-project.org>

**Description** Visualization techniques, data sets, summary and inference procedures aimed particularly at categorical data. Special emphasis is given to highly extensible grid graphics. The package was inspired by the book ‘‘Visualizing Categorical Data’’ by Michael Friendly.

**LazyLoad** yes

**LazyData** yes

**Depends** R (>= 2.4.0), MASS, grid, colorspace

**Suggests** KernSmooth, mvtnorm, kernlab, HSAUR, coin

**Imports** stats, utils, MASS, grDevices

**License** GPL-2

**Repository** CRAN

**Date/Publication** 2012-02-20 11:37:19

## R topics documented:

agreementplot . . . . .	3
Arthritis . . . . .	5
assoc . . . . .	6
assocstats . . . . .	9
Baseball . . . . .	10
BrokenMarriage . . . . .	12
Bundesliga . . . . .	12

Bundestag2005	14
Butterfly	15
cd_plot	16
CoalMiners	18
coindep_test	19
cotabplot	21
cotab_panel	23
co_table	25
DanishWelfare	26
distplot	27
doubledecker	28
Employment	30
Federalist	31
fourfold	33
goodfit	35
grid_barplot	38
grid_legend	39
Hitters	40
hls	41
HorseKicks	42
Hospital	43
independence_table	44
JobSatisfaction	44
JointSports	45
Kappa	46
labeling_border	48
labeling_cells_list	52
legends	54
Lifeboats	56
mar_table	57
mosaic	58
MSPatients	61
NonResponse	62
oddsratio	63
Ord_plot	65
OvaryCancer	67
Pairs plot panel functions for diagonal cells	68
Pairs plot panel functions for off-diagonal cells	70
pairs.table	72
plot.loglm	74
PreSex	75
Punishment	77
RepVict	78
Rochdale	79
rootogram	80
Saxony	81
SexualFun	82
shadings	83

sieve . . . . .	87
SpaceShuttle . . . . .	89
spacings . . . . .	91
spine . . . . .	92
strucplot . . . . .	94
structable . . . . .	98
struc_assoc . . . . .	100
struc_mosaic . . . . .	102
struc_sieve . . . . .	104
Suicide . . . . .	105
table2d_summary . . . . .	106
ternaryplot . . . . .	107
tile . . . . .	109
Trucks . . . . .	111
UKSoccer . . . . .	112
VisualAcuity . . . . .	113
VonBort . . . . .	114
WeldonDice . . . . .	115
WomenQueue . . . . .	116
woolf_test . . . . .	117

**Index****118**


---

agreementplot	<i>Bangdiwala's Observer Agreement Chart</i>
---------------	--

---

**Description**

Representation of a  $k \times k$  confusion matrix, where the observed and expected diagonal elements are represented by superposed black and white rectangles, respectively. The function also computes a statistic measuring the strength of agreement (relation of respective area sums).

**Usage**

```
## Default S3 method:
agreementplot(x, reverse_y = TRUE, main = NULL,
              weights = c(1, 1 - 1/(ncol(x) - 1)^2), margins = par("mar"),
              newpage = TRUE, pop = TRUE, xlab = names(dimnames(x))[2],
              ylab = names(dimnames(x))[1],
              xlab_rot = 0, xlab_just = "center",
              ylab_rot = 90, ylab_just = "center",
              fill_col = function(j) gray((1 - (weights[j]) ^ 2) ^ 0.5),
              line_col = "red", xscale = TRUE, yscale = TRUE,
              ...)
## S3 method for class 'formula'
agreementplot(formula, data = NULL, ..., subset)
```

**Arguments**

<code>x</code>	a confusion matrix, i.e., a table with equal-sized dimensions.
<code>reverse_y</code>	if TRUE, the y axis is reversed (i.e., the rectangles' positions correspond to the contingency table).
<code>main</code>	user-specified main title.
<code>weights</code>	vector of weights for successive larger observed areas, used in the agreement strength statistic, and also for the shading. The first element should be 1.
<code>margins</code>	vector of margins (see <a href="#">par</a> ).
<code>newpage</code>	logical; if TRUE, the plot is drawn on a new page.
<code>pop</code>	logical; if TRUE, all newly generated viewports are popped after plotting.
<code>xlab, ylab</code>	labels of x- and y-axis.
<code>xlab_rot, ylab_rot</code>	rotation angle for the category labels.
<code>xlab_just, ylab_just</code>	justification for the category labels.
<code>fill_col</code>	a function, giving the fill colors used for exact and partial agreement
<code>line_col</code>	color used for the diagonal reference line
<code>formula</code>	a formula, such as $y \sim x$ . For details, see <a href="#">xtabs</a> .
<code>data</code>	a data frame (or list), or a contingency table from which the variables in <code>formula</code> should be taken.
<code>subset</code>	an optional vector specifying a subset of the rows in the data frame to be used for plotting.
<code>xscale, yscale</code>	logicals indicating whether the marginals should be added on the x-axis/y-axis, respectively.
<code>...</code>	further graphics parameters (see <a href="#">par</a> ).

**Details**

Weights can be specified to allow for partial agreement, taking into account contributions from off-diagonal cells. A weight vector of length 1 means strict agreement only, each additional element increases the maximum number of disagreement steps.

**Value**

Invisibly returned, a list with components

<code>Bangdiwala</code>	the unweighted agreement strength statistic.
<code>Bangdiwala_Weighted</code>	the weighted statistic.
<code>weights</code>	the weight vector used.

**Author(s)**

David Meyer <David.Meyer@R-project.org>

## References

Michael Friendly (2000), *Visualizing Categorical Data*. SAS Institute, Cary, NC.

## Examples

```
data("SexualFun")
agreementplot(t(SexualFun))

data("MSPatients")
## best visualized using a resized device, e.g. using:
## get(getOption("device"))(width = 12)
pushViewport(viewport(layout = grid.layout(ncol = 2)))
pushViewport(viewport(layout.pos.col = 1))
agreementplot(t(MSPatients[,1]), main = "Winnipeg Patients",
               newpage = FALSE)
popViewport()
pushViewport(viewport(layout.pos.col = 2))
agreementplot(t(MSPatients[,2]), main = "New Orleans Patients",
               newpage = FALSE)
popViewport(2)
dev.off()
```

---

Arthritis

*Arthritis Treatment Data*

---

## Description

Data from Koch & Edwards (1988) from a double-blind clinical trial investigating a new treatment for rheumatoid arthritis.

## Usage

```
data("Arthritis")
```

## Format

A data frame with 84 observations and 5 variables.

**ID** patient ID.

**Treatment** factor indicating treatment (Placebo, Treated).

**Sex** factor indicating sex (Female, Male).

**Age** age of patient.

**Improved** ordered factor indicating treatment outcome (None, Some, Marked).

## Source

Michael Friendly (2000), *Visualizing Categorical Data*: <http://euclid.psych.yorku.ca/ftp/sas/vcd/catdata/arthritis.sas>

## References

- G. Koch & S. Edwards (1988), Clinical efficiency trials with categorical data. In K. E. Peace (ed.), *Biopharmaceutical Statistics for Drug Development*, 403–451. Marcel Dekker, New York.
- M. Friendly (2000), *Visualizing Categorical Data*. SAS Institute, Cary, NC.

## Examples

```
data("Arthritis")
art <- xtabs(~ Treatment + Improved, data = Arthritis, subset = Sex == "Female")
art

mosaic(art, gp = shading_Friendly)
mosaic(art, gp = shading_max)
```

---

assoc

*Extended Association Plots*

---

## Description

Produce an association plot indicating deviations from a specified independence model in a possibly high-dimensional contingency table.

## Usage

```
## Default S3 method:
assoc(x, row_vars = NULL, col_vars = NULL, compress = TRUE,
      xlim = NULL, ylim = NULL,
      spacing = spacing_conditional(sp = 0), spacing_args = list(),
      split_vertical = NULL, keep_aspect_ratio = FALSE,
      xscale = 0.9, yspace = unit(0.5, "lines"), main = NULL, sub = NULL,
      ..., residuals_type = "Pearson", gp_axis = gpar(lty = 3))
## S3 method for class 'formula'
assoc(formula, data = NULL, ..., subset = NULL, na.action = NULL, main = NULL, sub = NULL)
```

## Arguments

- |                       |   |
|-----------------------|---|
| <code>x</code>        | a contingency table in array form with optional category labels specified in the <code>dimnames(x)</code> attribute, or an object inheriting from the "ftable" class (such as "structable" objects).  |
| <code>row_vars</code> | a vector of integers giving the indices, or a character vector giving the names of the variables to be used for the rows of the association plot.   |
| <code>col_vars</code> | a vector of integers giving the indices, or a character vector giving the names of the variables to be used for the columns of the association plot.  |
| <code>compress</code> | logical; if FALSE, the space between the rows (columns) are chosen such that the <i>total</i> heights (widths) of the rows (columns) are all equal. If TRUE, the space between rows and columns is fixed and hence the plot is more "compressed". |

<code>xlim</code>	a $2 \times k$ matrix of doubles, $k$ number of total columns of the plot. The columns of <code>xlim</code> correspond to the columns of the association plot, the rows describe the column ranges (minimums in the first row, maximums in the second row). If <code>xlim</code> is NULL, the ranges are determined from the residuals according to <code>compress</code> (if TRUE: widest range from each column, if FALSE: from the whole association plot matrix).
<code>ylim</code>	a $2 \times k$ matrix of doubles, $k$ number of total rows of the plot. The columns of <code>ylim</code> correspond to the rows of the association plot, the rows describe the column ranges (minimums in the first row, maximums in the second row). If <code>ylim</code> is NULL, the ranges are determined from the residuals according to <code>compress</code> (if TRUE: widest range from each row, if FALSE: from the whole association plot matrix).
<code>spacing</code>	a spacing object, a spacing function, or a corresponding generating function (see <a href="#">strucplot</a> for more information). The default is the spacing-generating function <a href="#">spacing_conditional</a> that is (by default) called with the argument list <code>spacing_args</code> (see <a href="#">spacings</a> for more details).
<code>spacing_args</code>	list of arguments for the spacing-generating function, if specified (see <a href="#">strucplot</a> for more information).
<code>split_vertical</code>	vector of logicals of length $k$ , where $k$ is the number of margins of <code>x</code> (default: FALSE). Values are recycled as needed. A TRUE component indicates that the corresponding dimension is folded into the columns, FALSE folds the dimension into the rows.
<code>keep_aspect_ratio</code>	logical indicating whether the aspect ratio should be fixed or not.
<code>residuals_type</code>	a character string indicating the type of residuals to be computed. Currently, only Pearson residuals are supported.
<code>xscale</code>	scale factor resizing the tile's width, thus adding additional space between the tiles.
<code>yspace</code>	object of class "unit" specifying additional space separating the rows.
<code>gp_axis</code>	object of class "gpar" specifying the visual aspects of the tiles' baseline.
<code>formula</code>	a formula object with possibly both left and right hand sides specifying the column and row variables of the flat table.
<code>data</code>	a data frame, list or environment containing the variables to be cross-tabulated, or an object inheriting from class <code>table</code> .
<code>subset</code>	an optional vector specifying a subset of observations to be used. Ignored if <code>data</code> is a contingency table.
<code>na.action</code>	an optional function which indicates what should happen when the data contain NAs. Ignored if <code>data</code> is a contingency table.
<code>main, sub</code>	either a logical, or a character string used for plotting the main (sub) title. If logical and TRUE, the name of the data object is used.
<code>...</code>	other parameters passed to <a href="#">strucplot</a>

## Details

Association plots have been suggested by Cohen (1980) and extended by Friendly (1992) and provide a means for visualizing the residuals of an independence model for a contingency table.

`assoc` is a generic function and currently has a default method and a formula interface. Both are high-level interfaces to the `strucplot` function, and produce (extended) association plots. Most of the functionality is described there, such as specification of the independence model, labeling, legend, spacing, shading, and other graphical parameters.

For a contingency table, the signed contribution to Pearson's  $\chi^2$  for cell  $\{ij \dots k\}$  is

$$d_{ij\dots k} = \frac{(f_{ij\dots k} - e_{ij\dots k})}{\sqrt{e_{ij\dots k}}}$$

where  $f_{ij\dots k}$  and  $e_{ij\dots k}$  are the observed and expected counts corresponding to the cell. In the association plot, each cell is represented by a rectangle that has (signed) height proportional to  $d_{ij\dots k}$  and width proportional to  $\sqrt{e_{ij\dots k}}$ , so that the area of the box is proportional to the difference in observed and expected frequencies. The rectangles in each row are positioned relative to a baseline indicating independence ( $d_{ij\dots k} = 0$ ). If the observed frequency of a cell is greater than the expected one, the box rises above the baseline, and falls below otherwise.

Additionally, the residuals can be colored depending on a specified shading scheme (see Meyer et al., 2003). Package `vcd` offers a range of *residual-based* shadings (see the shadings help page). Some of them allow, e.g., the visualization of test statistics.

Unlike the `assocplot` function in the `graphics` package, this function allows the visualization of contingency tables with more than two dimensions. Similar to the construction of ‘flat’ tables (like objects of class `"ftable"` or `"structable"`), the dimensions are folded into rows and columns.

The layout is very flexible: the specification of shading, labeling, spacing, and legend is modularized (see `strucplot` for details).

## Value

The `"structable"` visualized is returned invisibly.

## Author(s)

David Meyer <David.Meyer@R-project.org>

## References

Cohen, A. (1980), On the graphical display of the significant components in a two-way contingency table. *Communications in Statistics—Theory and Methods*, **A9**, 1025–1041.

Friendly, M. (1992), Graphical methods for categorical data. *SAS User Group International Conference Proceedings*, **17**, 190–200. <http://datavis.ca/papers/sugi/sugi17.pdf>

Meyer, D., Zeileis, A., Hornik, K. (2003), Visualizing independence using extended association plots. *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, K. Hornik, F. Leisch, A. Zeileis (eds.), ISSN 1609-395X. <http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/>

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. URL <http://www.jstatsoft.org/v17/i03/> and available as `vignette("strucplot")`.

### See Also

[mosaic](#), [strucplot](#), [structable](#)

### Examples

```
data("HairEyeColor")
## Aggregate over sex:
(x <- margin.table(HairEyeColor, c(1, 2)))

## Ordinary assocplot:
assoc(x)
## and with residual-based shading (of independence)
assoc(x, main = "Relation between hair and eye color", shade = TRUE)

## Aggregate over Eye color:
(x <- margin.table(HairEyeColor, c(1, 3)))
chisq.test(x)
assoc(x, main = "Relation between hair color and sex", shade = TRUE)

# Visualize multi-way table
assoc(aperm(HairEyeColor), expected = ~ (Hair + Eye) * Sex,
      labeling_args = list(just_labels = c(Eye = "left"),
                          offset_labels = c(right = -0.5),
                          offset_varnames = c(right = 1.2),
                          rot_labels = c(right = 0),
                          tl_varnames = c(Eye = TRUE))
      )

assoc(aperm(UCBAdmissions), expected = ~ (Admit + Gender) * Dept, compress = FALSE,
      labeling_args = list(abbreviate = c(Gender = TRUE), rot_labels = 0)
      )
```

---

assocstats

*Association Statistics*

---

### Description

Computes the Pearson chi-Squared test, the Likelihood Ratio chi-Squared test, the phi coefficient, the contingency coefficient and Cramer's V.

### Usage

```
assocstats(x)
```

**Arguments**

`x` an  $r \times c$  table.

**Value**

A list with components:

`chisq_tests` a  $2 \times 3$  table with the chi-squared statistics.  
`phi` The phi coefficient.  
`cont` The contingency coefficient.  
`cramer` Cramer's V.

**Author(s)**

David Meyer <David.Meyer@R-project.org>

**References**

Michael Friendly (2000), *Visualizing Categorical Data*. SAS Institute, Cary, NC.

**Examples**

```
data("Arthritis")
tab <- xtabs(~Improved + Treatment, data = Arthritis)
summary(assocstats(tab))
```

---

Baseball

*Baseball Data*

---

**Description**

Baseball data.

**Usage**

```
data("Baseball")
```

**Format**

A data frame with 322 observations and 25 variables.

**name1** player's first name.

**name2** player's last name.

**atbat86** times at Bat: number of official plate appearances by a hitter. It counts as an official at-bat as long as the batter does not walk, sacrifice, get hit by a pitch or reach base due to catcher's interference.

**hits86** hits.

**homer86** home runs.

**runs86** the number of runs scored by a player. A run is scored by an offensive player who advances from batter to runner and touches first, second, third and home base in that order without being put out.

**rbi86** Runs Batted In: A hitter earns a run batted in when he drives in a run via a hit, walk, sacrifice (bunt or fly) fielder's choice, hit-batsman or on an error (when the official scorer rules that the run would have scored anyway).

**walks86** A "walk" (or "base on balls") is an award of first base granted to a batter who receives four pitches outside the strike zone.

**years** Years in the Major Leagues. Seems to count all years a player has actually played in the Major Leagues, not necessarily consecutive.

**atbat** career times at bat.

**hits** career hits.

**homeruns** career home runs.

**runs** career runs.

**rbi** career runs batted in.

**walks** career walks.

**league86** player's league.

**div86** player's division.

**team86** player's team.

**posit86** player's position (see [Hitters](#)).

**outs86** number of putouts (see [Hitters](#))

**assist86** number of assists (see [Hitters](#))

**error86** number of errors (see [Hitters](#))

**sal87** annual salary on opening day (in USD 1000).

**league87** league in 1987.

**team87** team in 1987.

### Source

SAS System for Statistical Graphics, First Edition, page A2.3

### References

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

### See Also

[Hitters](#)

### Examples

```
data("Baseball")
```

---

BrokenMarriage

*Broken Marriage Data*

---

### Description

Data from the Danish Welfare Study about broken marriages or permanent relationships depending on gender and social rank.

### Usage

```
data("BrokenMarriage")
```

### Format

A data frame with 20 observations and 4 variables.

**Freq** frequency.

**gender** factor indicating gender (male, female).

**rank** factor indicating social rank (I, II, III, IV, V).

**broken** factor indicating whether the marriage or permanent relationship was broken (yes, no).

### Source

E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*, page 177.

### References

E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*. 2nd edition. Springer-Verlag, Berlin.

### Examples

```
data("BrokenMarriage")
structable(~ ., data = BrokenMarriage)
```

---

Bundesliga

*Ergebnisse der Fussball-Bundesliga*

---

### Description

Results from the first German soccer league (1963-2008).

### Usage

```
data("Bundesliga")
```

**Format**

A data frame with 14018 observations and 7 variables.

**HomeTeam** factor. Name of the home team.

**AwayTeam** factor. Name of the away team.

**HomeGoals** number of goals scored by the home team.

**AwayGoals** number of goals scored by the away team.

**Round** round of the game.

**Year** year in which the season started.

**Date** starting time of the game (in "POSIXct" format).

**Details**

The data comprises all games in the first German soccer league since its foundation in 1963. The data have been queried online from the official Web page of the DFB and prepared as a data frame in R by Daniel Dekic, Torsten Hothorn, and Achim Zeileis (replacing earlier versions of the data in the package containing only subsets of years).

Each year/season comprises 34 rounds (except 1963, 1964, 1991) so that all 18 teams play twice against each other (switching home court advantage). In 1963/64, there were only 16 teams, hence only 30 rounds. In 1991, after the German unification, there was one season with 20 teams and 38 rounds.

**Source**

Homepage of the Deutscher Fussball-Bund (DFB, German Football Association): <http://www.dfb.de/>

**References**

Leonhard Knorr-Held (1999), Dynamic rating of sports teams. SFB 386 "Statistical Analysis of Discrete Structures", Discussion paper **98**.

**See Also**

[UKSoccer](#)

**Examples**

```
data("Bundesliga")

## number of goals per game poisson distributed?
ngoals1 <- xtabs(~ HomeGoals, data = Bundesliga, subset = Year == 1995)
ngoals2 <- xtabs(~ AwayGoals, data = Bundesliga, subset = Year == 1995)
ngoals3 <- table(apply(subset(Bundesliga, Year == 1995)[,3:4], 1, sum))

gf1 <- goodfit(ngoals1)
gf2 <- goodfit(ngoals2)
gf3 <- goodfit(ngoals3)
```

```
summary(gf1)
summary(gf2)
summary(gf3)
plot(gf1)
plot(gf2)
plot(gf3)

Ord_plot(ngoals1)
distplot(ngoals1)
```

---

 Bundestag2005

*Votes in German Bundestag Election 2005*


---

### Description

Number of votes by province in the German Bundestag election 2005 (for the parties that eventually entered the parliament).

### Usage

```
data("Bundestag2005")
```

### Format

A 2-way "table" giving the number of votes for each party (Fraktion) in each of the 16 German provinces (Bundesland):

No	Name	Levels
1	Bundesland	Schleswig-Holstein, Mecklenburg-Vorpommern, ...
2	Fraktion	SPD, CDU/CSU, Gruene, FDP, Linke

### Details

In the election for the German parliament "Bundestag", five parties obtained enough votes to enter the parliament: the social democrats SPD, the conservative CDU/CSU, the liberal FDP, the green party "Die Gruenen" and the leftist party "Die Linke". The table Bundestag2005 gives the number of votes for each party (Fraktion) in each of the 16 German provinces (Bundesland). The provinces are ordered from North to South.

The data have been obtained from the German statistical office (Statistisches Bundesamt) from the Web page given below.

Note that the number of seats in the parliament cannot be computed from the number of votes alone. The examples below show the distribution of seats that resulted from the election.

### Source

Der Bundeswahlleiter, Statistisches Bundesamt. <http://www.bundeswahlleiter.de/bundestagswahl2005/>

**Examples**

```
## The outcome of the election in terms of seats in the
## parliament was:
seats <- structure(c(226, 61, 54, 51, 222),
  .Names = c("CDU/CSU", "FDP", "Linke", "Gruene", "SPD"))

## Hues are chosen as metaphors for the political parties
## CDU/CSU: blue, FDP: yellow, Linke: purple, Gruene: green, SPD: red
## using the respective hues from a color wheel with
## chroma = 60 and luminance = 75
parties <- rainbow_hcl(6, c = 60, l = 75)[c(5, 2, 6, 3, 1)]
names(parties) <- names(seats)
parties

## The pie chart shows that neither the SPD+Gruene coalition nor
## the opposition of CDU/CSU+FDP could assemble a majority.
## No party would enter a coalition with the leftists, leading to a
## big coalition.
pie(seats, clockwise = TRUE, col = parties)

## The regional distribution of the votes, stratified by province,
## is shown in a mosaic display: first for the 10 Western then the
## 6 Eastern provinces.
data("Bundestag2005")
votes <- Bundestag2005[c(1, 3:5, 9, 11, 13:16, 2, 6:8, 10, 12),
  c("CDU/CSU", "FDP", "SPD", "Gruene", "Linke")]
mosaic(votes, gp = gpar(fill = parties[colnames(votes)]),
  spacing = spacing_highlighting, labeling = labeling_left,
  labeling_args = list(rot_labels = c(0, 90, 0, 0), pos_labels = "center",
  just_labels = c("center", "center", "center", "right"), varnames = FALSE),
  margins = unit(c(2.5, 1, 1, 12), "lines"),
  keep_aspect_ratio = FALSE)
```

---

Butterfly

*Butterfly Species in Malaya*


---

**Description**

Data from Fisher et al. (1943) giving the number of tokens found for each of 501 species of butterflies collected in Malaya.

**Usage**

```
data("Butterfly")
```

**Format**

A 1-way table giving the number of tokens for 501 species of butterflies. The variable and its levels are

No	Name	Levels
1	nTokens	0, 1, ..., 24

### Source

Michael Friendly (2000), *Visualizing Categorical Data*, pages 21–22.

### References

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

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

### Examples

```
data("Butterfly")
Ord_plot(Butterfly)
```

---

cd\_plot

*Conditional Density Plots*

---

### Description

Computes and plots conditional densities describing how the distribution of a categorical variable  $y$  changes over a numerical variable  $x$ .

### Usage

```
cd_plot(x, ...)
## Default S3 method:
cd_plot(x, y,
  plot = TRUE, ylab_tol = 0.05,
  bw = "nrd0", n = 512, from = NULL, to = NULL,
  main = "", xlab = NULL, ylab = NULL, margins = c(5.1, 4.1, 4.1, 3.1),
  gp = gpar(), name = "cd_plot", newpage = TRUE, pop = TRUE,
  ...)
## S3 method for class 'formula'
cd_plot(formula, data = list(),
  plot = TRUE, ylab_tol = 0.05,
  bw = "nrd0", n = 512, from = NULL, to = NULL,
  main = "", xlab = NULL, ylab = NULL, margins = c(5.1, 4.1, 4.1, 3.1),
  gp = gpar(), name = "cd_plot", newpage = TRUE, pop = TRUE,
  ...)
```

**Arguments**

x	an object, the default method expects either a single numerical variable.
y	a "factor" interpreted to be the dependent variable
formula	a "formula" of type $y \sim x$ with a single dependent "factor" and a single numerical explanatory variable.
data	an optional data frame.
plot	logical. Should the computed conditional densities be plotted?
ylab_tol	convenience tolerance parameter for y-axis annotation. If the distance between two labels drops under this threshold, they are plotted equidistantly.
bw, n, from, to, ...	arguments passed to <a href="#">density</a>
main, xlab, ylab	character strings for annotation
margins	margins when calling <a href="#">plotViewport</a>
gp	a "gpar" object controlling the grid graphical parameters of the rectangles. It should specify in particular a vector of fill colors of the same length as <code>levels(y)</code> . The default is to call <a href="#">gray.colors</a> .
name	name of the plotting viewport.
newpage	logical. Should <a href="#">grid.newpage</a> be called before plotting?
pop	logical. Should the viewport created be popped?

**Details**

cd\_plot computes the conditional densities of x given the levels of y weighted by the marginal distribution of y. The densities are derived cumulatively over the levels of y.

This visualization technique is similar to spinograms (see [spine](#)) but they do not discretize the explanatory variable, but rather use a smoothing approach. Furthermore, the original x axis and not a distorted x axis (as for spinograms) is used. This typically results in conditional densities that are based on very few observations in the margins: hence, the estimates are less reliable there.

**Value**

The conditional density functions (cumulative over the levels of y) are returned invisibly.

**Author(s)**

Achim Zeileis <Achim.Zeileis@R-project.org>

**References**

Hofmann, H., Theus, M. (2005), *Interactive graphics for visualizing conditional distributions*, Unpublished Manuscript.

**See Also**

[spine](#), [density](#)

**Examples**

```
## Arthritis data
data("Arthritis")
cd_plot(Improved ~ Age, data = Arthritis)
cd_plot(Improved ~ Age, data = Arthritis, bw = 3)
cd_plot(Improved ~ Age, data = Arthritis, bw = "SJ")
## compare with spinogram
spine(Improved ~ Age, data = Arthritis, breaks = 3)

## Space shuttle data
data("SpaceShuttle")
cd_plot(Fail ~ Temperature, data = SpaceShuttle, bw = 2)

## scatter plot with conditional density
cdens <- cd_plot(Fail ~ Temperature, data = SpaceShuttle, bw = 2, plot = FALSE)
plot(I(-1 * (as.numeric(Fail) - 2)) ~ jitter(Temperature, factor = 2), data = SpaceShuttle,
     xlab = "Temperature", ylab = "Failure")
lines(53:81, cdens[[1]](53:81), col = 2)
```

CoalMiners

*Breathlessness and Wheeze in Coal Miners***Description**

Data from Ashford & Snowden (1970) given by Agresti (1990) on the association between two pulmonary conditions, breathlessness and wheeze, in a large sample of coal miners.

**Usage**

```
data("CoalMiners")
```

**Format**

A 3-dimensional array resulting from cross-tabulating variables for 16,330 coal miners. The variables and their levels are as follows:

No	Name	Levels
1	Wheeze	W, NoW
2	Breathlessness	B, NoB
3	Age	25-29, 30-34, ..., 60-64

**Source**

Michael Friendly (2000), *Visualizing Categorical Data*, pages 82–83, 319–322.

**References**

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

J. R. Ashford & R. D. Snowdon (1970), Multivariate probit analysis, *Biometrics*, **26**, 535–546.  
 M. Friendly (2000), *Visualizing Categorical Data*. SAS Institute, Cary, NC.

### Examples

```
data("CoalMiners")

## Fourfold display, both margins equated
fourfold(CoalMiners, mfc01 = c(2,4))

## Log Odds Ratio Plot
summary(l <- oddsratio(CoalMiners))
g <- seq(25, 60, by = 5)
plot(l,
      xlab = "Age Group",
      main = "Breathlessness and Wheeze in Coal Miners")
m <- lm(l ~ g + I(g^2))
lines(fitted(m), col = "red")

## Fourfold display, strata equated
fourfold(CoalMiners, std = "ind.max", mfc01 = c(2,4))
```

---

 coindep\_test

*Test for (Conditional) Independence*


---

### Description

Performs a test of (conditional) independence of 2 margins in a contingency table by simulation from the marginal distribution of the input table under (conditional) independence.

### Usage

```
coindep_test(x, margin = NULL, n = 1000,
            indepfun = function(x) max(abs(x)), aggfun = max,
            alternative = c("greater", "less"),
            pearson = TRUE)
```

### Arguments

x	a contingency table.
margin	margin index(es) or corresponding name(s) of the conditioning variables. Each resulting conditional table has to be a 2-way table.
n	number of (conditional) independence tables to be drawn.
indepfun	aggregation function capturing independence in (each conditional) 2-way table.
aggfun	aggregation function aggregating the test statistics computed by indepfun.
alternative	a character string specifying the alternative hypothesis; must be either "greater" (default) or "less" (and may be abbreviated.)
pearson	logical. Should the table of Pearson residuals under independence be computed and passed to indepfun (default) or the raw table of observed frequencies?

## Details

If `margin` is `NULL` this computes a simple independence statistic in a 2-way table. Alternatively, `margin` can give several conditioning variables and then conditional independence in the resulting conditional table is tested.

By default, this uses a (double) maximum statistic of Pearson residuals. By changing `indepfun` or `aggfun` a (maximum of) Pearson Chi-squared statistic(s) can be computed or just the usual Pearson Chi-squared statistics and so on. Other statistics can be computed by changing `pearson` to `FALSE`.

The function uses [r2dtable](#) to simulate the distribution of the test statistic under the null.

## Value

A list of class "coindep\_test" inheriting from "hstest" with following components:

<code>statistic</code>	the value of the test statistic.
<code>p.value</code>	the $p$ value for the test.
<code>method</code>	a character string indicating the type of the test.
<code>data.name</code>	a character string giving the name(s) of the data.
<code>observed</code>	observed table of frequencies
<code>expctd</code>	expected table of frequencies
<code>residuals</code>	corresponding Pearson residuals
<code>margin</code>	the margin used
<code>dist</code>	a vector of size $n$ with simulated values of the distribution of the statistic under the null.
<code>qdist</code>	the corresponding quantile function (for computing critical values).
<code>pdist</code>	the corresponding distribution function (for computing $p$ values).

## Author(s)

Achim Zeileis <Achim.Zeileis@R-project.org>

## See Also

[chisq.test](#), [fisher.test](#), [r2dtable](#)

## Examples

```
TeaTasting <- matrix(c(3, 1, 1, 3), nr = 2,
                    dimnames = list(Guess = c("Milk", "Tea"),
                                    Truth = c("Milk", "Tea")))
)
## compute maximum statistic
coindep_test(TeaTasting)
## compute Chi-squared statistic
coindep_test(TeaTasting, indepfun = function(x) sum(x^2))
## use unconditional asymptotic distribution
chisq.test(TeaTasting, correct = FALSE)
```

```

chisq.test(TeaTasting)

data("UCBAdmissions")
## double maximum statistic
coindep_test(UCBAdmissions, margin = "Dept")
## maximum of Chi-squared statistics
coindep_test(UCBAdmissions, margin = "Dept", indepfun = function(x) sum(x^2))
## Pearson Chi-squared statistic
coindep_test(UCBAdmissions, margin = "Dept", indepfun = function(x) sum(x^2), aggfun = sum)
## use unconditional asymptotic distribution
loglm(~ Dept * (Gender + Admit), data = UCBAdmissions)

```

---

cotabplot

*Coplot for Contingency Tables*


---

## Description

cotabplot is a generic function for creating trellis-like coplots (conditional plots) for contingency tables.

## Usage

```

cotabplot(x, ...)
## Default S3 method:
cotabplot(x, cond = NULL,
  panel = cotab_mosaic, panel_args = list(),
  margins = rep(1, 4), layout = NULL,
  text_gp = gpar(fontsize = 12), rect_gp = gpar(fill = grey(0.9)),
  pop = TRUE, newpage = TRUE,
  ...)
## S3 method for class 'formula'
cotabplot(formula, data = NULL, ...)

```

## Arguments

x	an object. The default method can deal with contingency tables in array form.
cond	margin index(es) or corresponding name(s) of the conditioning variables.
panel	panel function applied for each conditioned plot, see details.
panel_args	list of arguments passed to panel if this is a panel-generating function inheriting from class "grapcon_generator".
margins	either an object of class "unit" of length 4, or a numeric vector of length 4. The elements are recycled as needed. giving the margins around the whole plot.
layout	integer vector (of length two), giving the number of rows and columns for the panel.
text_gp	object of class "gpar" used for the text in the panel titles.

rect_gp	object of class "gpar" used for the rectangles with the panel titles.
pop	logical indicating whether the generated viewport tree should be removed at the end of the drawing or not.
newpage	logical controlling whether a new grid page should be created.
...	further arguments passed to the panel-generating function.
formula	a formula specifying the variables used to create a contingency table from data. It has to be of type $\sim x + y \mid z$ where $z$ is/are the conditioning variable(s) used.
data	either a data frame, or an object of class "table" or "ftable".

### Details

cotabplot is a generic function designed to create coplots or conditional plots (see Cleveland, 1993, and Becker, Cleveland, Shyu, 1996) similar to [coplot](#) but for contingency tables.

cotabplot takes on computing the conditioning information and setting up the trellis display, and then relies on a panel function to create plots from the full table and the conditioning information. A simple example would be a contingency table `tab` with margin names "x", "y" and "z". To produce this plot either the default interface can be used or the formula interface via

```
cotabplot(tab, "z") cotabplot(~ x + y | z, data = tab)
```

The panel function needs to be of the form

```
panel(x, condlevels)
```

where  $x$  is the *full* table (`tab` in the example above) and `condlevels` is a named vector with the levels (e.g., `c(z = "z1")` in the example above).

Alternatively, `panel` can also be a panel-generating function of class "grapcon\_generator" which creates a function with the interface described above. The panel-generating function is called with the interface

```
panel(x, condvars, ...)
```

where again  $x$  is the full table, `condvars` is now only a vector with the names of the conditioning variables (and not their levels, e.g., "z" in the example above). Further arguments can be passed to the panel-generating function via `...` which also includes the arguments set in `panel_args`.

Suitable panel-generating functions for mosaic, association and sieve plots can be found at [cotab\\_mosaic](#).

A description of the underlying ideas is given in Zeileis, Meyer, Hornik (2005).

### Author(s)

Achim Zeileis <Achim.Zeileis@R-project.org>

### References

- Becker, R.A., Cleveland, W.S., Shyu, M.-J. (1996), The visual design and control of trellis display. *Journal of Computational and Graphical Statistics*, **5**, 123–155.
- Cleveland, W.S. (1993), *Visualizing Data*, Summit, New Jersey: Hobart Press.

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. URL <http://www.jstatsoft.org/v17/i03/> and available as `vignette("strucplot")`.

Zeileis, A., Meyer, D., Hornik K. (2007), *Residual-based shadings for visualizing (conditional) independence*, *Journal of Computational and Graphical Statistics*, **16**, 507–525.

### See Also

[cotab\\_mosaic](#), [cotab\\_coindep](#), [co\\_table](#), [coindep\\_test](#)

### Examples

```
data("UCBAdmissions")

cotabplot(~ Admit + Gender | Dept, data = UCBAdmissions)
cotabplot(~ Admit + Gender | Dept, data = UCBAdmissions, panel = cotab_assoc)

ucb <- cotab_coindep(UCBAdmissions, condvars = "Dept", type = "assoc", n = 5000, margins = c(3, 1, 1, 3))
cotabplot(~ Admit + Gender | Dept, data = UCBAdmissions, panel = ucb)
```

---

cotab\_panel

*Panel-generating Functions for Contingency Table Coplots*

---

### Description

Panel-generating functions visualizing contingency tables that can be passed to `cotabplot`.

### Usage

```
cotab_mosaic(x = NULL, condvars = NULL, ...)
cotab_assoc(x = NULL, condvars = NULL, ylim = NULL, ...)
cotab_sieve(x = NULL, condvars = NULL, ...)
cotab_fourfold(x = NULL, condvars = NULL, ...)
cotab_coindep(x, condvars,
  test = c("doublemax", "maxchisq", "sumchisq"),
  level = NULL, n = 1000, interpolate = c(2, 4),
  h = NULL, c = NULL, l = NULL, lty = 1,
  type = c("mosaic", "assoc"), legend = FALSE, ylim = NULL, ...)
```

### Arguments

<code>x</code>	a contingency tables in array form.
<code>condvars</code>	margin name(s) of the conditioning variables.
<code>ylim</code>	y-axis limits for assoc plot. By default this is computed from <code>x</code> .
<code>test</code>	character indicating which type of statistic should be used for assessing conditional independence.

level, n, h, c, l, lty, interpolate  
 variables controlling the HCL shading of the residuals, see [shadings](#) for more details.

type  
 character indicating which type of plot should be produced.

legend  
 logical. Should a legend be produced in each panel?

...  
 further arguments passed to the plotting function (such as [mosaic](#) or [assoc](#) or [sieve](#) respectively).

## Details

These functions of class "panel\_generator" are panel-generating functions for use with [cotabplot](#), i.e., they return functions with the interface

```
panel(x, condlevels)
```

required for [cotabplot](#). The functions produced by [cotab\\_mosaic](#), [cotab\\_assoc](#) and [cotab\\_sieve](#) essentially only call [co\\_table](#) to produce the conditioned table and then call [mosaic](#), [assoc](#) or [sieve](#) respectively with the arguments specified.

The function [cotab\\_coinddep](#) is similar but additionally chooses an appropriate residual-based shading visualizing the associated conditional independence model. The conditional independence test is carried out via [coinddep\\_test](#) and the shading is set up via [shading\\_hcl](#).

A description of the underlying ideas is given in Zeileis, Meyer, Hornik (2005).

## Author(s)

Achim Zeileis <Achim.Zeileis@R-project.org>

## References

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. URL <http://www.jstatsoft.org/v17/i03/> and available as `vignette("strucplot")`.

Zeileis, A., Meyer, D., Hornik K. (2007), *Residual-based shadings for visualizing (conditional) independence*, *Journal of Computational and Graphical Statistics*, **16**, 507–525.

## See Also

[cotabplot](#), [mosaic](#), [assoc](#), [sieve](#), [co\\_table](#), [coinddep\\_test](#), [shading\\_hcl](#)

## Examples

```
data("UCBAdmissions")

cotabplot(~ Admit + Gender | Dept, data = UCBAdmissions)
cotabplot(~ Admit + Gender | Dept, data = UCBAdmissions, panel = cotab_assoc)
cotabplot(~ Admit + Gender | Dept, data = UCBAdmissions, panel = cotab_fourfold)

ucb <- cotab_coinddep(UCBAdmissions, condvars = "Dept", type = "assoc", n = 5000, margins = c(3, 1, 1, 3))
cotabplot(~ Admit + Gender | Dept, data = UCBAdmissions, panel = ucb)
```

---

co_table	<i>Compute Conditional Tables</i>
----------	-----------------------------------

---

**Description**

For a contingency table in array form, compute a list of conditional tables given some margins.

**Usage**

```
co_table(x, margin, collapse = ".")
```

**Arguments**

x	a contingency table in array form.
margin	margin index(es) or corresponding name(s) of the conditioning variables.
collapse	character used when collapsing level names (if more than 1 margin is specified).

**Details**

This is essentially an interface to `[` which is more convenient for arrays of arbitrary dimension.

**Value**

A list of the resulting conditional tables.

**Author(s)**

Achim Zeileis <Achim.Zeileis@R-project.org>

**Examples**

```
data("HairEyeColor")
co_table(HairEyeColor, 1)
co_table(HairEyeColor, c("Hair", "Eye"))
co_table(HairEyeColor, 1:2, collapse = "")
```

---

DanishWelfare

*Danish Welfare Study Data*

---

### Description

Data from the Danish Welfare Study.

### Usage

```
data("DanishWelfare")
```

### Format

A data frame with 180 observations and 5 variables.

**Freq** frequency.

**Alcohol** factor indicating daily alcohol consumption: less than 1 unit (<1), 1-2 units (1-2) or more than 2 units (>2). 1 unit is approximately 1 bottle of beer or 4cl 40% alcohol.

**Income** factor indicating income group in 1000 DKK (0-50, 50-100, 100-150, >150).

**Status** factor indicating marriage status (Widow, Married, Unmarried).

**Urban** factor indicating urbanization: Copenhagen (Copenhagen), Suburban Copenhagen (Sub-Copenhagen), three largest cities (LargeCity), other cities (City), countryside (Country).

### Source

E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*, page 205.

### References

E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*. 2nd edition. Springer-Verlag, Berlin.

### Examples

```
data("DanishWelfare")  
ftable(xtabs(Freq ~ ., data = DanishWelfare))
```

**Description**

Diagnostic distribution plots: poissonness, binomialness and negative binomialness plots.

**Usage**

```
distplot(x, type = c("poisson", "binomial", "nbinomial"),
         size = NULL, lambda = NULL, legend = TRUE, xlim = NULL, ylim = NULL,
         conf_int = TRUE, conf_level = 0.95, main = NULL,
         xlab = "Number of occurrences", ylab = "Distribution metameter",
         gp = gpar(cex = 0.5), name = "distplot", newpage = TRUE, pop = TRUE, ...)
```

**Arguments**

x	either a vector of counts, a 1-way table of frequencies of counts or a data frame or matrix with frequencies in the first column and the corresponding counts in the second column.
type	a character string indicating the distribution.
size	the size argument for the binomial and negative binomial distribution. If set to NULL and type is "binomial", then size is taken to be the maximum count. If set to NULL and type is "nbinomial", then size is estimated from the data.
lambda	parameter of the poisson distribution. If type is "poisson" and lambda is specified a leveled poissonness plot is produced.
legend	logical. Should a legend be plotted?
xlim	limits for the x axis.
ylim	limits for the y axis.
conf_int	logical. Should confidence intervals be plotted?
conf_level	confidence level for confidence intervals.
main	a title for the plot.
xlab	a label for the x axis.
ylab	a label for the y axis.
gp	a "gpar" object controlling the grid graphical parameters of the points.
name	name of the plotting viewport.
newpage	logical. Should <code>grid.newpage</code> be called before plotting?
pop	logical. Should the viewport created be popped?
...	further arguments passed to <code>grid.points</code> .

**Details**

distplot plots the number of occurrences (counts) against the distribution metameter of the specified distribution. If the distribution fits the data, the plot should show a straight line. See Friendly (2000) for details.

**Author(s)**

Achim Zeileis <Achim.Zeileis@R-project.org>

**References**

- D. C. Hoaglin (1980), A poissonness plot, *The American Statistician*, **34**, 146–149.
- D. C. Hoaglin & J. W. Tukey (1985), Checking the shape of discrete distributions. In D. C. Hoaglin, F. Mosteller, J. W. Tukey (eds.), *Exploring Data Tables, Trends and Shapes*, chapter 9. John Wiley & Sons, New York.
- M. Friendly (2000), *Visualizing Categorical Data*. SAS Institute, Cary, NC.

**Examples**

```
## Simulated data examples:
dummy <- rnbinom(1000, size = 1.5, prob = 0.8)
distplot(dummy, type = "nbinomial")

## Real data examples:
data("HorseKicks")
data("Federalist")
data("Saxony")
distplot(HorseKicks, type = "poisson")
distplot(HorseKicks, type = "poisson", lambda = 0.61)
distplot(Federalist, type = "poisson")
distplot(Federalist, type = "nbinomial", size = 1)
distplot(Federalist, type = "nbinomial")
distplot(Saxony, type = "binomial", size = 12)
```

---

doubledecker

*Doubledecker Plot*


---

**Description**

This function creates a doubledecker plot visualizing a classification rule.

**Usage**

```
## S3 method for class 'formula'
doubledecker(formula, data = NULL, ..., main = NULL)
## Default S3 method:
doubledecker(x, depvar = length(dim(x)),
  margins = c(1,4, length(dim(x)) + 1, 1),
```

```
gp = gpar(fill = rev(gray.colors(tail(dim(x), 1)))),
labeling = labeling_doubledecker,
spacing = spacing_highlighting,
main = NULL, keep_aspect_ratio = FALSE, ...)
```

### Arguments

formula	a formula specifying the variables used to create a contingency table from data. The dependent variable is used last for splitting.
data	either a data frame, or an object of class "table" or "fTable".
x	a contingency table in array form, with optional category labels specified in the dimnames(x) attribute.
devar	dimension index or character string specifying the dependent variable. That will be sorted last in the table.
margins	margins of the plot. Note that by default, all factor names (except the last one) and their levels are visualized <i>as a block</i> under the plot.
gp	object of class "gpar" used for the tiles of the last variable.
labeling	labeling function or corresponding generating function (see <a href="#">strucplot</a> for details).
spacing	spacing object, spacing function or corresponding generating function (see <a href="#">strucplot</a> for details).
main	either a logical, or a character string used for plotting the main title. If main is TRUE, the name of the data object is used.
keep_aspect_ratio	logical indicating whether the aspect ratio should be maintained or not.
...	Further parameters passed to mosaic.

### Details

Doubledecker plots visualize the dependence of one categorical (typically binary) variable on further categorical variables. Formally, they are mosaic plots with vertical splits for all dimensions (antecedents) except the last one, which represents the dependent variable (consequent). The last variable is visualized by horizontal splits, no space between the tiles, and separate colors for the levels.

### Value

The "structable" visualized is returned invisibly.

### Author(s)

David Meyer <David.Meyer@R-project.org>

**References**

H. Hoffmann (2001), Generalized odds ratios for visual modeling. *Journal of Computational and Graphical Statistics*, **10**, 4, 628–640.

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. URL <http://www.jstatsoft.org/v17/i03/> and available as vignette("strucplot").

**See Also**

[strucplot](#), [mosaic](#)

**Examples**

```
data("Titanic")
doubledecker(Titanic)
doubledecker(Titanic, depvar = "Survived")
doubledecker(Survived ~ ., data = Titanic)
```

---

Employment

*Employment Status*

---

**Description**

Data from a 1974 Danish study given by Andersen (1991) on the employees who had been laid off. The workers are classified by their employment status on 1975-01-01, the cause of their layoff and the length of employment before they were laid off.

**Usage**

```
data("Employment")
```

**Format**

A 3-dimensional array resulting from cross-tabulating variables for 1314 employees. The variables and their levels are as follows:

No	Name	Levels
1	EmploymentStatus	NewJob, Unemployed
2	EmploymentLength	<1Mo, 1-3Mo, 3-12Mo, 1-2Yr, 2-5Yr, >5Yr
3	LayoffCause	Closure, Replaced

**Source**

Michael Friendly (2000), *Visualizing Categorical Data*, pages 126–129.

## References

- E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*. Springer-Verlag, Berlin.
- M. Friendly (2000), *Visualizing Categorical Data*. SAS Institute, Cary, NC.

## Examples

```
data("Employment")

## Employment Status
mosaic(Employment,
       expected = ~ LayoffCause * EmploymentLength + EmploymentStatus,
       main = "Layoff*EmployLength + EmployStatus")

mosaic(Employment,
       expected = ~ LayoffCause * EmploymentLength + LayoffCause * EmploymentStatus,
       main = "Layoff*EmployLength + Layoff*EmployStatus")

## Stratified view

grid.newpage()
pushViewport(viewport(layout = grid.layout(ncol = 2)))
pushViewport(viewport(layout.pos.col = 1))

## Closure
mosaic(Employment[,1], main = "Layoff: Closure", newpage = FALSE)

popViewport(1)
pushViewport(viewport(layout.pos.col = 2))

## Replaced
mosaic(Employment[,2], main = "Layoff: Replaced", newpage = FALSE)
popViewport(2)
```

---

Federalist

*'May' in Federalist Papers*

---

## Description

Data from Mosteller & Wallace (1984) investigating the use of certain keywords ('may' in this data set) to identify the author of 12 disputed 'Federalist Papers' by Alexander Hamilton, John Jay and James Madison.

## Usage

```
data("Federalist")
```

**Format**

A 1-way table giving the number of occurrences of 'may' in 262 blocks of text. The variable and its levels are

No	Name	Levels
1	nMay	0, 1, ..., 6

### Source

Michael Friendly (2000), *Visualizing Categorical Data*, page 19.

### References

F. Mosteller & D. L. Wallace (1984), *Applied Bayesian and Classical Inference: The Case of the Federalist Papers*. Springer-Verlag, New York, NY.

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

### Examples

```
data("Federalist")
gf <- goodfit(Federalist, type = "nbinomial")
summary(gf)
plot(gf)
```

---

fourfold

*Fourfold Plots*

---

### Description

Creates an (extended) fourfold display of a  $2 \times 2 \times k$  contingency table, allowing for the visual inspection of the association between two dichotomous variables in one or several populations (strata).

### Usage

```
fourfold(x,
  color = c("#99CCFF", "#6699CC", "#FFA0A0", "#A0A0FF", "#FF0000", "#000080"),
  conf_level = 0.95, std = c("margins", "ind.max", "all.max"),
  margin = c(1, 2), space = 0.2, main = NULL, sub = NULL,
  mfrow = NULL, mfc0l = NULL, extended = TRUE, ticks = 0.15,
  p_adjust_method = p.adjust.methods, newpage = TRUE,
  fontsize = 12)
```

### Arguments

**x** a  $2 \times 2 \times k$  contingency table in array form, or a  $2 \times 2$  matrix if  $k$  is 1. If  $\text{length}(\text{dim}(x)) > 3$ , dimensions 3: $\text{length}(\text{dim}(x))$  are silently raveled into a combined strata dimension with  $k = \text{prod}(\text{dim}(x)[-(1:2)])$ .

color	a vector of length 6 specifying the colors to use for the smaller and larger diagonals of each $2 \times 2$ table. The first pair is used for the standard (non-extended) plots, the other two for the extended version: the second/third pair is used for tables with non-significant/significant log-odds ratios, respectively, the latter being visualized in brighter colors.
conf_level	confidence level used for the confidence rings on the odds ratios. Must be a single non-negative number less than 1; if set to 0, confidence rings are suppressed.
std	a character string specifying how to standardize the table. Must be one of "margins", "ind.max", or "all.max", and can be abbreviated by the initial letter. If set to "margins", each $2 \times 2$ table is standardized to equate the margins specified by margin while preserving the odds ratio. If "ind.max" or "all.max", the tables are either individually or simultaneously standardized to a maximal cell frequency of 1.
margin	a numeric vector with the margins to equate. Must be one of 1, 2, or c(1, 2) (the default), which corresponds to standardizing only the row, only column, or both row and column in each $2 \times 2$ table. Only used if std equals "margins".
space	the amount of space (as a fraction of the maximal radius of the quarter circles) used for the row and column labels.
main, sub	character string for the fourfold plot title/subtitle.
mfrow, mfcoll	a numeric vector with two components: <i>nr</i> and <i>nc</i> , indicating that the displays for the $2 \times 2$ tables should be arranged in an <i>nr</i> by <i>nc</i> layout, filled by rows/columns. The defaults are calculated to give a collection of plots in landscape orientation when <i>k</i> is not a perfect square.
extended	logical; if TRUE, extended plots are plotted, i.e., colors are brighter for significant log-odds ratios, and ticks are plotted showing the direction of association for positive log-odds.
ticks	the length of the ticks. If set to 0, no ticks are plotted.
p_adjust_method	method to be used for p-value adjustments for multi-stratum plots, as provided by <code>link[stats]{p.adjust}</code> . Use <code>p_adjust_method="none"</code> to disable this adjustment. The p-values are used for the 'visual' significance tests of the odds ratios.
newpage	logical; if TRUE, <code>grid.newpage()</code> is called before plotting.
fontsize	fontsize of main title. Other labels are scaled relative to this.

## Details

The fourfold display is designed for the display of  $2 \times 2 \times k$  tables.

Following suitable standardization, the cell frequencies  $f_{ij}$  of each  $2 \times 2$  table are shown as a quarter circle whose radius is proportional to  $\sqrt{f_{ij}}$  so that its area is proportional to the cell frequency. An association (odds ratio different from 1) between the binary row and column variables is indicated by the tendency of diagonally opposite cells in one direction to differ in size from those in the other direction; color is used to show this direction. Confidence rings for the odds ratio allow a visual test of the null of no association; the rings for adjacent quadrants overlap iff the observed counts are consistent with the null hypothesis.

Typically, the number  $k$  corresponds to the number of levels of a stratifying variable, and it is of interest to see whether the association is homogeneous across strata. The fourfold display visualizes the pattern of association. Note that the confidence rings for the individual odds ratios are not adjusted for multiple testing.

## References

Friendly, M. (1994), *A fourfold display for 2 by 2 by k tables*. Technical Report 217, York University, Psychology Department, <http://datavis.ca/papers/4fold/4fold.pdf>.

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

## See Also

`mosaic`, `assoc`

`link[stats]{p.adjust}` for methods of p value adjustment

## Examples

```
data("UCBAdmissions")
## Use the Berkeley admission data as in Friendly (1995).
x <- aperm(UCBAdmissions, c(2, 1, 3))
dimnames(x)[[2]] <- c("Yes", "No")
names(dimnames(x)) <- c("Sex", "Admit?", "Department")
ftable(x)

## Fourfold display of data aggregated over departments, with
## frequencies standardized to equate the margins for admission
## and sex.
## Figure 1 in Friendly (1994).
fourfold(margin.table(x, c(1, 2)))

## Fourfold display of x, with frequencies in each table
## standardized to equate the margins for admission and sex.
## Figure 2 in Friendly (1994).
fourfold(x)
cotabplot(x, panel = cotab_fourfold)

## Fourfold display of x, with frequencies in each table
## standardized to equate the margins for admission. but not
## for sex.
## Figure 3 in Friendly (1994).
fourfold(x, margin = 2)
```

## Description

Fits a discrete (count data) distribution for goodness-of-fit tests.

**Usage**

```
goodfit(x, type = c("poisson", "binomial", "nbinomial"),
        method = c("ML", "MinChisq"), par = NULL)
## S3 method for class 'goodfit'
predict(object, newcount = NULL, type = c("response", "prob"), ...)
```

**Arguments**

x	either a vector of counts, a 1-way table of frequencies of counts or a data frame or matrix with frequencies in the first column and the corresponding counts in the second column.
type	a character string indicating which distribution should be fit (for goodfit) or indicating the type of prediction (fitted response or probabilities in predict) respectively.
method	a character string indicating whether the distribution should be fit via ML (Maximum Likelihood) or Minimum Chi-squared.
par	a named list giving the distribution parameters (named as in the corresponding density function), if set to NULL, the default, the parameters are estimated. If the parameter size is not specified if type is "binomial" it is taken to be the maximum count. If type is "nbinomial", then parameter size can be specified to fix it so that only the parameter prob will be estimated (see the examples below).
object	an object of class "goodfit".
newcount	a vector of counts. By default the counts stored in object are used, i.e., the fitted values are computed. These can also be extracted by fitted(object).
...	<i>currently not used.</i>

**Details**

goodfit essentially computes the fitted values of a discrete distribution (either Poisson, binomial or negative binomial) to the count data given in x. If the parameters are not specified they are estimated either by ML or Minimum Chi-squared.

To fix parameters, par should be a named list specifying the parameters lambda for "poisson" and prob and size for "binomial" or "nbinomial", respectively. If for "binomial", size is not specified it is not estimated but taken as the maximum count.

The corresponding Pearson Chi-squared or likelihood ratio statistic, respectively, is computed and given with their  $p$  values by the summary method. The summary method always prints this information and returns a matrix with the printed information invisibly. The plot method produces a [rootogram](#) of the observed and fitted values.

In case of count distributions (Poisson and negative binomial), the minimum Chi-squared approach is somewhat ad hoc. Strictly speaking, the Chi-squared asymptotics would only hold if the number of cells were fixed or did not increase too quickly with the sample size. However, in goodfit the number of cells is data-driven: Each count is a cell of its own. All counts larger than the maximal count are merged into the cell with the last count for computing the test statistic.

**Value**

A list of class "goodfit" with elements:

observed	observed frequencies.
count	corresponding counts.
fitted	expected frequencies (fitted by ML).
type	a character string indicating the distribution fitted.
method	a character string indicating the fitting method (can be either "ML", "MinChisq" or "fixed" if the parameters were specified).
df	degrees of freedom.
par	a named list of the (estimated) distribution parameters.

**Author(s)**

Achim Zeileis <Achim.Zeileis@R-project.org>

**References**

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

**Examples**

```
## Simulated data examples:
dummy <- rnbinom(200, size = 1.5, prob = 0.8)
gf <- goodfit(dummy, type = "nbinomial", method = "MinChisq")
summary(gf)
plot(gf)

dummy <- rbinom(100, size = 6, prob = 0.5)
gf1 <- goodfit(dummy, type = "binomial", par = list(size = 6))
gf2 <- goodfit(dummy, type = "binomial", par = list(prob = 0.6, size = 6))
summary(gf1)
plot(gf1)
summary(gf2)
plot(gf2)

## Real data examples:
data("HorseKicks")
HK.fit <- goodfit(HorseKicks)
summary(HK.fit)
plot(HK.fit)

data("Federalist")
## try geometric and full negative binomial distribution
F.fit <- goodfit(Federalist, type = "nbinomial", par = list(size = 1))
F.fit2 <- goodfit(Federalist, type = "nbinomial")
summary(F.fit)
summary(F.fit2)
plot(F.fit)
plot(F.fit2)
```

---

`grid_barplot`*Barplot*

---

**Description**

Bar plots of 1-way tables in grid.

**Usage**

```
grid_barplot(height, width = 0.8, offset = 0,
  names = NULL, xlim = NULL, ylim = NULL, xlab = "", ylab = "", main = "",
  gp = gpar(fill = "lightgray"), name = "grid_barplot",
  newpage = TRUE, pop = FALSE)
```

**Arguments**

<code>height</code>	either a vector or a 1-way table of frequencies.
<code>width</code>	width of the bars (recycled if needed to the number of bars).
<code>offset</code>	offset of the bars (recycled if needed to the number of bars).
<code>names</code>	a vector of names for the bars, if set to NULL the names of <code>height</code> are used.
<code>xlim</code>	limits for the x axis.
<code>ylim</code>	limits for the y axis.
<code>xlab</code>	a label for the x axis.
<code>ylab</code>	a label for the y axis.
<code>main</code>	a title for the plot.
<code>gp</code>	a "gpar" object controlling the grid graphical parameters of the rectangles.
<code>name</code>	name of the plotting viewport.
<code>newpage</code>	logical. Should <code>grid.newpage</code> be called before plotting?
<code>pop</code>	logical. Should the viewport created be popped?

**Details**

`grid_barplot` mimics (some of) the features of `barplot`, but currently it only supports 1-way tables.

**Author(s)**

Achim Zeileis <Achim.Zeileis@R-project.org>

**Examples**

```
grid_barplot(sample(1:6), names = letters[1:6])
```

---

`grid_legend`*Legend Function for grid Graphics*

---

**Description**

This function can be used to add legends to *grid-based* plots.

**Usage**

```
grid_legend(x, y, pch, col, labels, frame = TRUE, hgap = unit(0.5, "lines"),
            vgap = unit(0.3, "lines"), default_units = "lines", gp = gpar(),
            draw = TRUE, title = "Legend:")
```

**Arguments**

<code>x, y</code>	coordinates of the legend
<code>pch</code>	integer vector of plotting symbols
<code>col</code>	character vector of colors for the symbols
<code>labels</code>	character vector of labels corresponding to the symbols
<code>frame</code>	logical indicating whether the legend should have a border or not.
<code>hgap</code>	object of class "unit" specifying the space between symbols and labels
<code>vgap</code>	object of class "unit" specifying the space between the lines
<code>default_units</code>	character string indicating the default unit
<code>gp</code>	object of class "gpar" used for the legend
<code>draw</code>	logical indicating whether the legend be drawn or not.
<code>title</code>	character string indicating the plot's title

**Value**

Invisibly, the legend as a "grob" object.

**Author(s)**

David Meyer <David.Meyer@R-project.org>

**See Also**

[legend](#)

**Examples**

```

data("Lifeboats")
attach(Lifeboats)
ternaryplot(Lifeboats[,4:6],
  pch = ifelse(side == "Port", 1, 19),
  col = ifelse(side == "Port", "red", "blue"),
  id = ifelse(men / total > 0.1, as.character(boat), NA),
  prop_size = 2,
  dimnames_position = "edge",
  main = "Lifeboats on Titanic")
grid_legend(0.8, 0.9, c(1, 19), c("red", "blue"),
  c("Port", "Starboard"), title = "SIDE")

```

---

Hitters

*Hitters Data*


---

**Description**

This data set is deduced from the [Baseball](#) fielding data set: fielding performance basically includes the numbers of Errors, Putouts and Assists made by each player. In order to reduce the number of observations, the was compressed by calculating the mean number of errors, putouts and assists for each team and for only 6 positions (1B, 2B, 3B, C, OF, SS and UT). In addition, each of these three variables was scaled to a common range by dividing each variable by the maximum of the variable.

**Usage**

```
data("Hitters")
```

**Format**

A data frame with 154 observations and 4 variables.

**Positions** factor indicating the field position (1B=first baseman, 2B=second baseman, 3B=third baseman, C=catcher, OF=outfielder, SS=Short Stop, UT=Utility Players).

**Putouts** occur when a fielder causes an opposing player to be tagged or forced out.

**Assists** are credited to other fielders involved in making that putout.

**Errors** count the errors made by a player.

**Source**

SAS System for Statistical Graphics, First Edition, Page A2.3

**References**

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

**Examples**

```
data("Hitters")
attach(Hitters)

colors <- c("black", "red", "green", "blue", "red", "black", "blue")
pch <- substr(levels(Positions), 1, 1)
ternaryplot(Hitters[,2:4],
  pch = as.character(Positions),
  col = colors[as.numeric(Positions)],
  main = "Baseball Hitters Data")
grid_legend(0.8, 0.9, pch, colors, levels(Positions),
  title = "POSITION(S)")

detach(Hitters)
```

---

hls

*HLS Color Specification*

---

**Description**

Create a HLS color from specifying hue, luminance and saturation.

**Usage**

```
hls(h = 1, l = 0.5, s = 1)
```

**Arguments**

h	hue value in [0, 1].
l	luminance value in [0, 1].
s	saturation value in [0, 1].

**Details**

HLS colors are a similar specification of colors as HSV colors, but using hue/luminance/saturation rather than hue/saturation/value.

**Author(s)**

Achim Zeileis <Achim.Zeileis@R-project.org>

**See Also**

[hsv](#), [hcl2hex](#), [polarLUV](#)

**Examples**

```
## an HLS color wheel
pie(rep(1, 12), col = sapply(1:12/12, function(x) hls(x)))
```

---

HorseKicks

*Death by Horse Kicks*

---

### Description

Data from von Bortkiewicz (1898), given by Andrews & Herzberg (1985), on number of deaths by horse or mule kicks in 10 (of 14 reported) corps of the Prussian army. 4 corps were not considered by Fisher (1925) as they had a different organization. This data set is a popular subset of the [VonBort](#) data.

### Usage

```
data("HorseKicks")
```

### Format

A 1-way table giving the number of deaths in 200 corps-years. The variable and its levels are

No	Name	Levels
1	nDeaths	0, 1, ..., 4

### Source

Michael Friendly (2000), *Visualizing Categorical Data*, page 18.

### References

D. F. Andrews & A. M. Herzberg (1985), *Data: A Collection of Problems from Many Fields for the Student and Research Worker*. Springer-Verlag, New York, NY.

R. A. Fisher (1925), *Statistical Methods for Research Workers*. Oliver & Boyd, London.

L. von Bortkiewicz (1898), *Das Gesetz der kleinen Zahlen*. Teubner, Leipzig.

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

### See Also

[VonBort](#)

### Examples

```
data("HorseKicks")
gf <- goodfit(HorseKicks)
summary(gf)
plot(gf)
```

---

Hospital

*Hospital data*

---

### Description

The table relates the length of stay (in years) of 132 long-term schizophrenic patients in two London mental hospitals with the frequency of visits.

### Usage

```
data("Hospital")
```

### Format

A 2-dimensional array resulting from cross-tabulating 132 patients. The variables and their levels are as follows:

No	Name	Levels
1	Visit Frequency	Regular, Less than monthly, Never
2	Length of Stay	2–9 years, 10–19 years, 20+ years

### Details

Wing (1962) who collected this data concludes that the longer the length of stay in hospital, the less frequent the visits.

Haberman (1974) notes that this pattern does not increase from the "Less than monthly" to the "Never" group, which are homogeneous.

### Source

S.J Haberman (1974): Log-linear models for frequency tables with ordered classifications. *Biometrics*, 30:689–700.

### References

J.K. Wing (1962): Institutionalism in mental hospitals. *British Journal of Social Clinical Psychology*, 1:38–51.

### Examples

```
data("Hospital")

mosaic(t(Hospital), shade = TRUE)
mosaic(Hospital, shade = TRUE)
sieve(Hospital, shade = TRUE)
assoc(Hospital, shade = TRUE)
```

independence\_table      *Independence Table*

---

**Description**

Computes table of expected frequencies (under the null hypotheses of independence) from an  $n$ -way table.

**Usage**

```
independence_table(x, frequency = c("absolute", "relative"))
```

**Arguments**

x                      a table.  
frequency            indicates whether absolute or relative frequencies should be computed.

**Value**

A table with either absolute or relative frequencies.

**Author(s)**

David Meyer <David.Meyer@R-project.org>

**Examples**

```
data("MSPatients")  
independence_table(MSPatients)  
independence_table(MSPatients, frequency = "relative")
```

---

JobSatisfaction      *Job Satisfaction Data*

---

**Description**

Data from Petersen (1968) about the job satisfaction of 715 blue collar workers, selected from Danish Industry in 1968.

**Usage**

```
data("JobSatisfaction")
```

**Format**

A data frame with 8 observations and 4 variables.

**Freq** frequency.

**management** factor indicating quality of management (bad, good).

**supervisor** factor indicating supervisor's job satisfaction (low, high).

**own** factor indicating worker's own job satisfaction (low, high).

**Source**

E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*, Table 5.4.

**References**

E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*. 2nd edition. Springer-Verlag, Berlin.

E. Petersen (1968), *Job Satisfaction in Denmark*. (In Danish). Mentalhygiejnisk Forlag, Copenhagen.

**Examples**

```
data("JobSatisfaction")
structable(~ ., data = JobSatisfaction)

mantelhaen.test(xtabs(Freq ~ own + supervisor + management,
                      data = JobSatisfaction))
```

---

JointSports

*Opinions About Joint Sports*

---

**Description**

Data from a Danish study in 1983 and 1985 about sports activities and the opinion about joint sports with the other gender among 16–19 year old high school students.

**Usage**

```
data("JointSports")
```

**Format**

A data frame with 40 observations and 5 variables.

**Freq** frequency.

**opinion** factor indicating opinion about sports joint with the other gender (very good, good, indifferent, bad, very bad).

**year** factor indicating year of study (1983, 1985).

**grade** factor indicating school grade (1st, 3rd).

**gender** factor indicating gender (Boy, Girl).

**Source**

E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*, page 210.

**References**

E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*. 2nd edition. Springer-Verlag, Berlin.

**Examples**

```
data("JointSports")
tab <- xtabs(Freq ~ gender + opinion + grade + year, data = JointSports)
doubledecker(opinion ~ gender + year + grade, data = tab)
loglm(~ opinion* (gender + grade+ year) + gender*year*grade, data = tab)
```

---

Kappa

*Cohen's Kappa and Weighted Kappa*

---

**Description**

Computes two agreement rates: Cohen's kappa and weighted kappa, and confidence bands.

**Usage**

```
Kappa(x, weights = c("Equal-Spacing", "Fleiss-Cohen"))
```

**Arguments**

x	a confusion matrix.
weights	either one of the character strings given in the default value, or a user-specified matrix with same dimensions as x.

**Details**

Cohen's kappa is the diagonal sum of the (possibly weighted) relative frequencies, corrected for expected values and standardized by its maximum value. The equal-spacing weights are defined by  $1 - |i - j| / (r - 1)$ ,  $r$  number of columns/rows, and the Fleiss-Cohen weights by  $1 - |i - j|^2 / (r - 1)^2$ . The latter one attaches greater importance to near disagreements.

**Value**

An object of class "Kappa" with three components:

Unweighted	numeric vector of length 2 with the kappa statistic (value component), along with Approximate Standard Error (ASE component)
Weighted	idem for the weighted kappa.
Weights	numeric matrix with weights used.

**Note**

The summary method also prints the weights.

There is a `confint` method for computing approximate confidence intervals.

**Author(s)**

David Meyer <David.Meyer@R-project.org>

**References**

Cohen, J. (1960), A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, **20**, 37–46.

Everitt, B.S. (1968), Moments of statistics kappa and weighted kappa. *The British Journal of Mathematical and Statistical Psychology*, **21**, 97–103.

Fleiss, J.L., Cohen, J., and Everitt, B.S. (1969), Large sample standard errors of kappa and weighted kappa. *Psychological Bulletin*, **72**, 332–327.

**See Also**

[agreementplot](#), [confint](#)

**Examples**

```
data("SexualFun")
Kappa(SexualFun)
```

labeling\_border

*Labeling Functions for Strucplots***Description**

These functions generate labeling functions used for strucplots.

**Usage**

```
labeling_border(labels = TRUE, varnames = labels,
  set_labels = NULL, set_varnames = NULL,
  tl_labels = NULL, alternate_labels = FALSE, tl_varnames = NULL,
  gp_labels = gpar(fontsize = 12),
  gp_varnames = gpar(fontsize = 12, fontface = 2),
  rot_labels = c(0, 90, 0, 90), rot_varnames = c(0, 90, 0, 90),
  pos_labels = "center", pos_varnames = "center",
  just_labels = "center", just_varnames = pos_varnames,
  boxes = FALSE, fill_boxes = FALSE,
  offset_labels = c(0, 0, 0, 0), offset_varnames = offset_labels,
  labbl_varnames = NULL, labels_varnames = FALSE, sep = ": ",
  abbreviate_labs = FALSE, rep = TRUE, clip = FALSE, ...)
labeling_values(value_type = c("observed", "expected", "residuals"),
  suppress = NULL, digits = 1, clip_cells = FALSE, ...)
labeling_residuals(suppress = NULL, digits = 1, clip_cells = FALSE, ...)
labeling_conditional(...)
labeling_left(rep = FALSE, pos_varnames = "left",
  pos_labels = "left", just_labels = "left", ...)
labeling_left2(tl_labels = TRUE, clip = TRUE, pos_varnames = "left",
  pos_labels = "left", just_labels = "left", ...)
labeling_cboxed(tl_labels = TRUE, boxes = TRUE, clip = TRUE,
  pos_labels = "center", ...)
labeling_lboxed(tl_labels = FALSE, boxes = TRUE, clip = TRUE,
  pos_labels = "left", just_labels = "left",
  labbl_varnames = FALSE, ...)
labeling_doubledecker(lab_pos = c("bottom", "top"), dep_varname = TRUE,
  boxes = NULL, clip = NULL, labbl_varnames = FALSE,
  rot_labels = rep.int(0, 4),
  pos_labels = c("left", "center", "left", "center"),
  just_labels = c("left", "left", "left", "center"),
  varnames = NULL, gp_varnames = gpar(fontsize = 12, fontface = 2),
  offset_varnames = c(0, -0.6, 0, 0), tl_labels = NULL, ...)
```

**Arguments**

**labels** vector of logicals indicating whether labels should be drawn for a particular dimension.

varnames	vector of logicals indicating whether variable names should be drawn for a particular dimension.
set_labels	An optional character vector with named components replacing the so-specified variable names. The component names must exactly match the variable names to be replaced.
set_varnames	An optional list with named components of character vectors replacing the labels of the so-specified variables. The component names must exactly match the variable names whose labels should be replaced.
tl_labels	vector of logicals indicating whether labels should be positioned on top (column labels) / left (row labels) for a particular dimension.
alternate_labels	vector of logicals indicating whether labels should be alternated on the top/bottom (left/right) side of the plot for a particular dimension.
tl_varnames	vector of logicals indicating whether variable names should be positioned on top (column labels) / on left (row labels) for a particular dimension.
gp_labels	list of objects of class "gpar" used for drawing the labels.
gp_varnames	list of objects of class "gpar" used for drawing the variable names.
rot_labels	vector of rotation angles for the labels for each of the four sides of the plot.
rot_varnames	vector of rotation angles for the variable names for each of the four sides of the plot.
pos_labels	character string of label positions ("left", "center", "right") for each of the variables.
pos_varnames	character string of variable names positions ("left", "center", "right") for each of the four sides of the plot.
just_labels	character string of label justifications ("left", "center", "right") for each of the variables.
just_varnames	character string of variable names justifications ("left", "center", "right") for each of the four sides of the plot.
boxes	vector of logicals indicating whether boxes should be drawn around the labels for a particular dimension.
fill_boxes	Either a vector of logicals, or a vector of characters, or a list of such vectors, specifying the fill colors for the boxes. "TRUE" and "FALSE" values are transformed into "grey" and "white", respectively. If fill_boxes is atomic, each component specifies a basic color for the corresponding dimension. This color is transformed into its HSV representation, and the value is varied from 50% to 100% to give a sequential color palette for the levels. For NA components, no palette is produced (no fill color). If fill_boxes is a list of vectors, each vector specifies the level colors of the corresponding dimension.
offset_labels, offset_varnames	numeric vector of length 4 indicating the offset of the labels (variable names) for each of the four sides of the plot.
labbl_varnames	vector of logicals indicating whether variable names should be drawn on the left (column variables) / on top (row variables) of the corresponding labels.

labels_varnames	vector of logicals indicating, for each dimension, whether the variable name should be added to the corresponding labels or not.
sep	separator used if any component of "labels_varnames" is TRUE.
abbreviate_labs	vector of integers or logicals indicating, for each dimension, the number of characters the labels should be abbreviated to. TRUE means 1 character, FALSE causes no abbreviation. Values are recycled as needed.
rep	vector of logicals indicating, for each dimension, whether labels should be repeated for all conditioning strata, or appear only once.
clip	vector of integers indicating, for each dimension, whether labels should be clipped to not overlap.
lab_pos	character string switching between "top" or "bottom" position of the labels (only used for labeling_doubledecker).
dep_varname	logical or character string. If logical, this is indicating whether the name of the dependent variable should be printed or not. A character string will be printed instead of the variable name taken from the dimnames.
value_type	character string specifying which values should be displayed in the cells.
suppress	numeric vector of length 2 specifying an interval of values that are not displayed. 0 values are never displayed. A single number, $k$ , is treated as $c(-k, k)$ . The default for labeling residuals is $c(-2, 2)$ . Use <code>suppress = 0</code> to show all non-zero values.
digits	integer specifying the number of digits used for rounding.
clip_cells	logical indicating whether the values should be clipped at the cell borders.
...	only used for labeling_conditional and labeling_doubledecker: parameters passed to labeling_cells and labeling_border.

## Details

These functions generate labeling functions called by `strucplot` for their side-effect of adding labels to the plot. They suppose that a `strucplot` has been drawn and the corresponding viewport structure is pushed, since the positions of the viewports are used for the label positioning. Note that the functions can also be used 'stand-alone' as shown in the examples.

All values supplied to vectorized arguments can be 'abbreviated' by using named components which override the default component values. In addition, these defaults can be overloaded by the sequence of non-named components which are recycled as needed (see examples).

This help page only documents `labeling_border` and derived functions, more functions are described on the help page for `labeling_cells` and `labeling_list`.

`labeling_left`, `labeling_left2`, `labeling_cboxed`, and `labeling_lboxed` are really just wrappers to `labeling_border`, and good examples for the parameter usage.

`labeling_residuals` is a trivial wrapper for `labeling_values`, which in turn calls `labeling_border` by additionally adding the observed or expected frequencies or residuals to the cells.

**Value**

A function with arguments:

`d` "dimnames" attribute from the visualized contingency table, or the visualized table itself from which the "dimnames" attributes will then be extracted.

`split_vertical` vector of logicals indicating the split directions.

`condvars` integer vector of conditioning dimensions.

**Author(s)**

David Meyer <David.Meyer@R-project.org>

**References**

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. URL <http://www.jstatsoft.org/v17/i03/> and available as vignette("strucplot").

**See Also**

[labeling\\_cells](#), [labeling\\_list](#), [structable](#), [grid.text](#)

**Examples**

```
data("Titanic")

mosaic(Titanic)

mosaic(Titanic, labeling = labeling_left)
labeling_left

mosaic(Titanic, labeling = labeling_cboxed)
labeling_cboxed

mosaic(Titanic, labeling = labeling_lboxed)
labeling_lboxed

data("PreSex")
mosaic(~ PremaritalSex + ExtramaritalSex | Gender + MaritalStatus,
      data = PreSex, labeling = labeling_conditional)

## specification of vectorized arguments
mosaic(Titanic, labeling_args = list(abbreviate_labs = c(Survived = TRUE)))

mosaic(Titanic, labeling_args = list(clip = TRUE, boxes = TRUE,
      fill_boxes = c(Survived = "green", "red")))

mosaic(Titanic, labeling_args = list(clip = TRUE, boxes = TRUE,
      fill_boxes = list(Sex = "red", "green")))

mosaic(Titanic, labeling_args = list(clip = TRUE, boxes = TRUE,
```

```

fill_boxes = list(Sex = c(Male = "red", "blue"), "green"))

## change variable names
mosaic(Titanic, labeling_args = list(set_varnames = c(Sex = "Gender")))

## change labels
mosaic(Titanic, labeling_args = list(set_varnames = c(Survived = "Status"),
  set_labels = list(Survived = c("Survived", "Not Survived")), rep = FALSE))

## show frequencies
mosaic(Titanic, labeling = labeling_values)

```

---

labeling\_cells\_list     *Labeling Functions for Strucplots*

---

## Description

These functions generate labeling functions that produce labels for strucplots.

## Usage

```

labeling_cells(labels = TRUE, varnames = TRUE,
  abbreviate_labels = FALSE, abbreviate_varnames = FALSE,
  gp_text = gpar(), lsep = ": ", lcollapse = "\n",
  just = "center", pos = "center", rot = 0,
  margin = unit(0.5, "lines"), clip_cells = TRUE,
  text = NULL, ...)
labeling_list(gp_text = gpar(), just = "left", pos = "left", lsep = ": ",
  sep = " ", offset = unit(c(2, 2), "lines"),
  varnames = TRUE, cols = 2, ...)

```

## Arguments

labels	vector of logicals indicating, for each dimension, whether labels for the factor levels should be drawn or not. Values are recycled as needed.
varnames	vector of logicals indicating, for each dimension, whether variable names should be drawn. Values are recycled as needed.
abbreviate_labels	vector of integers or logicals indicating, for each dimension, the number of characters the labels should be abbreviated to. TRUE means 1 character, FALSE causes no abbreviation. Values are recycled as needed.
abbreviate_varnames	vector of integers or logicals indicating, for each dimension, the number of characters the variable (i.e., dimension) names should be abbreviated to. TRUE means 1 character, FALSE causes no abbreviation. Values are recycled as needed.
gp_text	object of class "gpar" used for the text drawn.
lsep	character that separates variable names from the factor levels.

sep	character that separates the factor levels (only used for labeling_list).
offset	object of class "unit" of length 2 specifying the offset in x- and y-direction of the text block drawn under the strucplot (only used for labeling_list).
cols	number of text columns (only used for labeling_list).
lcollapse	character that separates several variable name/factor level-combinations. Typically a line break. (Only used for labeling_cells.)
just, pos	character string of length 1 (labeling_list) or at most 2 (labeling_cells) specifying the labels' horizontal position and justification (horizontal and vertical for labeling_cells).
rot	rotation angle in degrees, used for all labels (only used for labeling_cells).
margin	object of class "unit" (a numeric value is converted to "lines") specifying an offset from the cell borders (only used for labeling_cells).
clip_cells	logical indicating whether text should be clipped at the cell borders (only used for labeling_cells).
text	Optionally, a character table of the same dimensions than the contingency table whose entries will then be used instead of the labels. NA entries are not drawn. This allows custom cell annotations (see examples). Only used for labeling_cells.
...	Currently not used.

### Details

These functions generate labeling functions that can add different kinds of labels to an existing plot. Typically they are supplied to [strucplot](#) which then generates and calls the labeling function. They assume that a strucplot has been drawn and the corresponding viewport structure is pushed, so that by navigating through the viewport tree the labels can be positioned appropriately.

This help page only documents labeling\_list and labeling\_cells; more functions are described on the help page for [labeling\\_border](#).

The functions can also be used 'stand-alone' as shown in the examples.

Using labeling\_list will typically necessitate a bottom margin adjustment.

### Value

A function with arguments:

d	"dimnames" attribute from the visualized contingency table, or the visualized table itself from which the "dimnames" attributes will then be extracted.
split_vertical	vector of logicals indicating the split directions.
condvars	integer vector of conditioning dimensions

### Author(s)

David Meyer <David.Meyer@R-project.org>

## References

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. URL <http://www.jstatsoft.org/v17/i03/> and available as vignette("strucplot").

## See Also

[labeling\\_border](#), [structable](#), [grid.text](#)

## Examples

```
data("Titanic")

mosaic(Titanic, labeling = labeling_cells)
mosaic(Titanic, labeling = labeling_list)

## A more complex example, adding the observed frequencies
## to a mosaic plot:
tab <- ifelse(Titanic < 6, NA, Titanic)
mosaic(Titanic, pop = FALSE)
labeling_cells(text = tab, margin = 0)(Titanic)
```

---

legends

*Legend Functions for Strucplots*

---

## Description

These functions generate legend functions for residual-based shadings.

## Usage

```
legend_resbased(fontsize = 12,
  x = unit(1, "lines"), y = unit(0.1, "npc"),
  height = unit(0.8, "npc"),
  width = unit(0.7, "lines"),
  digits = 3, check_overlap = TRUE, text = NULL,
  steps = 200, ticks = 10, pvalue = TRUE, range = NULL)
legend_fixed(fontsize = 12, x = unit(1, "lines"), y = NULL,
  height = NULL, width = unit(1.5, "lines"), steps = 200,
  digits = 1, space = 0.05, text = NULL, range = NULL)
```

## Arguments

fontsize	fontsize of title and p-value text.
x, y	objects of class "unit" indicating the coordinates of the title. For <code>legend_fixed</code> , the default for y is computed as to leave enough space for the specified text.
height, width	object of class "unit" indicating the height/width of the legend. For <code>legend_fixed</code> , the default for y is computed as to align upper margins of legend and actual plot.

digits	number of digits for the scale labels.
check_overlap	logical indicating whether overlap of scale labels should be inhibited or not.
space	For legend_fixed only: proportion of space between the tiles.
text	character string indicating the title of the legend.
steps	granularity of the color gradient.
ticks	number of scale ticks.
pvalue	logical indicating whether the $p$ -value should be visualized under the scale or not.
range	Numeric vector of length 2 for setting the legend range. Computed from the residuals if omitted. NA values are replaced by the corresponding minimum / maximum of the residuals.

### Details

These functions generate legend functions for residual-based shadings, visualizing deviations from expected values of an hypothesized independence model. Therefore, the legend uses a supplied shading function to visualize the color gradient for the residuals range. legend\_fixed is inspired by the legend used in [mosaicplot](#). For more details on the shading functions and their return values, see [shadings](#).

### Value

A function with arguments:

residuals	residuals from the fitted independence model to be visualized.
shading	shading function computing colors from residuals (see details).
autotext	character vector indicating the title to be used when no text argument is specified. Allows strucplot to generate sensible defaults depending on the residuals type.

### Author(s)

David Meyer <David.Meyer@R-project.org>

### References

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. URL <http://www.jstatsoft.org/v17/i03/> and available as vignette("strucplot").

Meyer, D., Zeileis, A., Hornik, K. (2003), Visualizing independence using extended association plots. *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, K. Hornik, F. Leisch, A. Zeileis (eds.), ISSN 1609-395X. <http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/>

### See Also

[structable](#), [shadings](#)

## Examples

```
data("Titanic")

mosaic(Titanic, shade = TRUE, legend = legend_resbased)
mosaic(Titanic, shade = TRUE, legend = legend_fixed, gp = shading_Friendly)
```

---

Lifeboats

*Lifeboats on the Titanic*

---

## Description

Data from Mersey (1912) about the 18 (out of 20) lifeboats launched before the sinking of the S. S. Titanic.

## Usage

```
data("Lifeboats")
```

## Format

A data frame with 18 observations and 8 variables.

**launch** launch time in "POSIXt" format.

**side** factor. Side of the boat.

**boat** factor indicating the boat.

**crew** number of male crew members on board.

**men** number of men on board.

**women** number of women (including female crew) on board.

**total** total number of passengers.

**cap** capacity of the boat.

## Source

M. Friendly (2000), Visualizing Categorical Data: <http://euclid.psych.yorku.ca/ftp/sas/vcd/catdata/lifeboat.sas>

## References

L. Mersey (1912), Report on the loss of the "Titanic" (S. S.). Parliamentary command paper 6452.

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

**Examples**

```
data("Lifeboats")
attach(Lifeboats)
ternaryplot(
  Lifeboats[,4:6],
  pch = ifelse(side == "Port", 1, 19),
  col = ifelse(side == "Port", "red", "blue"),
  id = ifelse(men / total > 0.1, as.character(boat), NA),
  prop_size = 2,
  dimnames_position = "edge",
  main = "Lifeboats on the Titanic"
)
grid_legend(0.8, 0.9, c(1, 19), c("red", "blue"),
  c("Port", "Starboard"), title = "SIDE")
detach(Lifeboats)
```

---

mar\_table

*Table with Marginal Sums*

---

**Description**

Adds row and column sums to a two-way table.

**Usage**

```
mar_table(x)
```

**Arguments**

x                    a two-way table.

**Value**

A table with row and column totals added.

**Author(s)**

David Meyer <David.Meyer@R-project.org>

**Examples**

```
data("SexualFun")
mar_table(SexualFun)
```

**Description**

Plots (extended) mosaic displays.

**Usage**

```
## Default S3 method:
mosaic(x, condvars = NULL,
       split_vertical = NULL, direction = NULL, spacing = NULL,
       spacing_args = list(), gp = NULL, expected = NULL, shade = NULL,
       highlighting = NULL, highlighting_fill = grey.colors, highlighting_direction = NULL,
       zero_size = 0.5, zero_split = FALSE, zero_shade = NULL,
       zero_gp = gpar(col = 0), panel = NULL, main = NULL, sub = NULL, ...)
## S3 method for class 'formula'
mosaic(formula, data, highlighting = NULL,
       ..., main = NULL, sub = NULL, subset = NULL, na.action = NULL)
```

**Arguments**

x	a contingency table in array form, with optional category labels specified in the <code>dimnames(x)</code> attribute, or an object of class "structable".
condvars	vector of integers or character strings indicating conditioning variables, if any. The table will be permuted to order them first.
formula	a formula specifying the variables used to create a contingency table from data. For convenience, conditioning formulas can be specified; the conditioning variables will then be used first for splitting. If any, a specified response variable will be highlighted in the cells.
data	either a data frame, or an object of class "table" or "ftable".
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Ignored if data is a contingency table.
zero_size	size of the bullets used for zero entries (if 0, no bullets are drawn).
zero_split	logical controlling whether zero cells should be further split. If FALSE and zero_shade is FALSE, only one bullet is drawn (centered) for unsplit zero cells. If FALSE and zero_shade is TRUE, a bullet for each zero cell is drawn to allow, e.g., residual-based shadings to be effective also for zero cells.
zero_shade	logical controlling whether zero bullets should be shaded. The default is TRUE if shade is TRUE or expected is not null or gp is not null, and FALSE otherwise.
zero_gp	object of class "gpar" used for zero bullets in case they are <i>not</i> shaded.

<code>split_vertical</code>	vector of logicals of length $k$ , where $k$ is the number of margins of $x$ (default: FALSE). Values are recycled as needed. A TRUE component indicates that the tile(s) of the corresponding dimension should be split vertically, FALSE means horizontal splits. Ignored if <code>direction</code> is not NULL.
<code>direction</code>	character vector of length $k$ , where $k$ is the number of margins of $x$ (values are recycled as needed). For each component, a value of "h" indicates that the tile(s) of the corresponding dimension should be split horizontally, whereas "v" indicates vertical split(s).
<code>spacing</code>	spacing object, spacing function, or corresponding generating function (see <a href="#">strucplot</a> for more information). The default is <code>spacing_equal</code> if $x$ has two dimensions, <code>spacing_increase</code> for more dimensions, and <code>spacing_conditional</code> if conditioning variables are specified using <code>condvars</code> or the formula interface.
<code>spacing_args</code>	list of arguments for the generating function, if specified (see <a href="#">strucplot</a> for more information).
<code>gp</code>	object of class "gpar", shading function or a corresponding generating function (see details and <a href="#">shadings</a> ). Components of "gpar" objects are recycled as needed along the last splitting dimension. Ignored if <code>shade = FALSE</code> .
<code>shade</code>	logical specifying whether <code>gp</code> should be used or not (see <code>gp</code> ). If TRUE and <code>expected</code> is unspecified, a default model is fitted: if <code>condvars</code> (see <a href="#">strucplot</a> ) is specified, a corresponding conditional independence model, and else the total independence model.
<code>expected</code>	optionally, an array of expected values of the same dimension as $x$ , or alternatively the corresponding independence model specification as used by <a href="#">loglin</a> or <a href="#">loglm</a> (see <a href="#">strucplot</a> ).
<code>highlighting</code>	character vector or integer specifying a variable to be highlighted in the cells.
<code>highlighting_fill</code>	color vector or palette function used for a highlighted variable, if any.
<code>highlighting_direction</code>	Either "left", "right", "top", or "bottom" specifying the direction of highlighting in the cells.
<code>panel</code>	Optional function with arguments: <code>residuals</code> , <code>observed</code> , <code>expected</code> , <code>index</code> , <code>gp</code> , and <code>name</code> called by the <code>struc_mosaic</code> workhorse for each tile that is drawn in the mosaic. <code>index</code> is an integer vector with the tile's coordinates in the contingency table, <code>gp</code> a gpar object for the tile, and <code>name</code> a label to be assigned to the drawn grid object.
<code>main, sub</code>	either a logical, or a character string used for plotting the main (sub) title. If logical and TRUE, the name of the data object is used.
<code>...</code>	Other arguments passed to <a href="#">strucplot</a>

## Details

Mosaic displays have been suggested in the statistical literature by Hartigan and Kleiner (1984) and have been extended by Friendly (1994). [mosaicplot](#) is a base graphics implementation and `mosaic` is a much more flexible and extensible grid implementation.

`mosaic` is a generic function which currently has a default method and a formula interface. Both are high-level interfaces to the `strucplot` function, and produce (extended) mosaic displays. Most of the functionality is described there, such as specification of the independence model, labeling, legend, spacing, shading, and other graphical parameters.

A mosaic plot is an area proportional visualization of a (possibly higher-dimensional) table of expected frequencies. It is composed of tiles (corresponding to the cells) created by recursive vertical and horizontal splits of a square. The area of each tile is proportional to the corresponding cell entry, *given* the dimensions of previous splits.

An *extended* mosaic plot, in addition, visualizes the fit of a particular log-linear model. Typically, this is done by residual-based shadings where color and/or outline of the tiles visualize sign, size and possibly significance of the corresponding residual.

The layout is very flexible: the specification of shading, labeling, spacing, and legend is modularized (see `strucplot` for details).

In contrast to the `mosaicplot` function in **graphics**, the splits start with the *horizontal* direction by default to match the printed output of `structable`.

### Value

The "structable" visualized is returned invisibly.

### Author(s)

David Meyer <David.Meyer@R-project.org>

### References

Hartigan, J.A., and Kleiner, B. (1984), A mosaic of television ratings. *The American Statistician*, **38**, 32–35.

Emerson, J. W. (1998), Mosaic displays in S-PLUS: A general implementation and a case study. *Statistical Computing and Graphics Newsletter (ASA)*, **9**, 1, 17–23.

Friendly, M. (1994), Mosaic displays for multi-way contingency tables. *Journal of the American Statistical Association*, **89**, 190–200.

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. URL <http://www.jstatsoft.org/v17/i03/> and available as vignette("strucplot", package = "vcd").

The home page of Michael Friendly (<http://datavis.ca>) provides information on various aspects of graphical methods for analyzing categorical data, including mosaic plots. In particular, there are many materials for his course "Visualizing Categorical Data with SAS and R" at <http://datavis.ca/courses/VCD/>.

### See Also

[assoc](#), [strucplot](#), [mosaicplot](#), [structable](#), [doubledecker](#)

**Examples**

```

data("Titanic")
mosaic(Titanic)

## Formula interface for tabulated data plus shading and legend:
mosaic(~ Sex + Age + Survived, data = Titanic,
  main = "Survival on the Titanic", shade = TRUE, legend = TRUE)

data("HairEyeColor")
mosaic(HairEyeColor, shade = TRUE)
## Independence model of hair and eye color and sex. Indicates that
## there are significantly more blue eyed blond females than expected
## in the case of independence (and too few brown eyed blond females).

mosaic(HairEyeColor, shade = TRUE, expected = list(c(1,2), 3))
## Model of joint independence of sex from hair and eye color. Males
## are underrepresented among people with brown hair and eyes, and are
## overrepresented among people with brown hair and blue eyes, but not
## "significantly".

## Formula interface for raw data: visualize crosstabulation of numbers
## of gears and carburettors in Motor Trend car data.
data("mtcars")
mosaic(~ gear + carb, data = mtcars, shade = TRUE)

data("PreSex")
mosaic(PreSex, condvars = c(1,4))
mosaic(~ ExtramaritalSex + PremaritalSex | MaritalStatus + Gender,
  data = PreSex)

## Highlighting:
mosaic(Survived ~ ., data = Titanic)

data("Arthritis")
mosaic(Improved ~ Treatment | Sex, data = Arthritis, zero_size = 0)
mosaic(Improved ~ Treatment | Sex, data = Arthritis, zero_size = 0, highlighting_direction = "right")

```

---

MSPatients

*Diagnosis of Multiple Sclerosis*


---

**Description**

Data from Westlund & Kurland (1953) on the diagnosis of multiple sclerosis (MS): two samples of patients, one from Winnipeg and one from New Orleans, were each rated by two neurologists (one from each city) in four diagnostic categories.

**Usage**

```
data("MSPatients")
```

**Format**

A 3-dimensional array resulting from cross-tabulating 218 observations on 3 variables. The variables and their levels are as follows:

No	Name	Levels
1	New Orleans Neurologist	Certain, Probable, Possible, Doubtful
2	Winnipeg Neurologist	Certain, Probable, Possible, Doubtful
3	Patients	Winnipeg, New Orleans

**Source**

M. Friendly (2000), Visualizing Categorical Data: <http://euclid.psych.yorku.ca/ftp/sas/vcd/catdata/msdiag.sas>

**References**

K. B. Westlund & L. T. Kurland (1953), Studies on multiple sclerosis in Winnipeg, Manitoba and New Orleans, Louisiana, *American Journal of Hygiene*, **57**, 380–396.

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

**Examples**

```
data("MSPatients")
## best visualized using a resized device, e.g. using:
## get(getOption("device"))(width = 12)
pushViewport(viewport(layout = grid.layout(ncol = 2)))
pushViewport(viewport(layout.pos.col = 1))
agreementplot(t(MSPatients[, , 1]), main = "Winnipeg Patients",
               newpage = FALSE)
popViewport()
pushViewport(viewport(layout.pos.col = 2))
agreementplot(t(MSPatients[, , 2]), main = "New Orleans Patients",
               newpage = FALSE)
popViewport(2)
dev.off()
```

---

NonResponse

*Non-Response Survey Data*

---

**Description**

Data about non-response for a Danish survey in 1965.

**Usage**

```
data("NonResponse")
```

**Format**

A data frame with 12 observations and 4 variables.

**Freq** frequency.

**residence** factor indicating whether residence was in Copenhagen, in a city outside Copenhagen or at the countryside (Copenhagen, City, Country).

**response** factor indicating whether a response was given (yes, no).

**gender** factor indicating gender (male, female).

**Source**

E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*, Table 5.17.

**References**

E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*. 2nd edition. Springer-Verlag, Berlin.

**Examples**

```
data("NonResponse")
structable(~ ., data = NonResponse)
```

---

oddsratio	<i>(Log) Odds Ratios</i>
-----------	--------------------------

---

**Description**

Computes (log) odds ratios and their asymptotic standard errors for (possibly) stratified data.

**Usage**

```
oddsratio(x, stratum = NULL, log = TRUE)
## S3 method for class 'oddsratio'
plot(x, conf_level = 0.95, type = "o",
     xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL, whiskers = 0.1,
     baseline = TRUE, transpose = FALSE, ...)
```

**Arguments**

x	a 2 by 2 by ... table.
stratum	vector of strata dimensions.
log	if FALSE, ordinary odds ratios are computed.
conf_level	if not NULL or FALSE, conf_level-% confidence intervals are plotted for each data point.
type	plot type.

xlab	label for the x-axis. Defaults to "Strata" if transpose is FALSE.
ylab	label for the y-axis. Defaults to "Strata" if transpose is TRUE.
xlim	x-axis limits. Ignored if transpose is FALSE.
ylim	y-axis limits. Ignored if transpose is TRUE.
baseline	if TRUE, a red dashed line is plotted at a value of 1 (in case of odds) or 0 (in case of log-odds).
transpose	if TRUE, the plot is transposed.
whiskers	width of the confidence interval whiskers.
...	other graphics parameters (see <a href="#">par</a> ).

**Value**

An object of class "logoddsratio", which is simply a vector of (log) odds ratios with dimensionality depending on `stratum`, along with the following attributes:

ASE	a numeric vector with the asymptotic standard errors.
log	logical indicating whether log odds ratios or common odds ratios are computed.

**Note**

In case of zero entries, 0.5 will be added to the table.

The `summary` method prints the standard errors and—for log odds ratios—also computes and prints asymptotic  $z$  tests (standardized log odds ratios) and the corresponding  $p$  values.

There is a `confint` method for computing confidence intervals for the (log) odds ratios.

The `plot` method plots (log) odds ratios, computed by `oddsratio` for  $2 \times 2 \times k$  tables, along with confidence intervals.

**Author(s)**

David Meyer <David.Meyer@R-project.org>

**References**

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

**See Also**

[confint](#)

**Examples**

```
## load Coal Miners data
data("CoalMiners")

## compute log odds ratios
lor <- oddsratio(CoalMiners)
lor
```

```

## summary with z tests
summary(lor)

## confidence intervals
confint(lor)

## visualization
plot(lor,
      xlab = "Age Group",
      main = "Breathelessness and Wheeze in Coal Miners")

## add quadratic model
g <- seq(25, 60, by = 5)
m <- lm(lor ~ g + I(g^2))
lines(fitted(m), col = "red")

```

---

Ord\_plot

*Ord Plots*


---

## Description

Ord plots for diagnosing discrete distributions.

## Usage

```

Ord_plot(obj, legend = TRUE, estimate = TRUE, tol = 0.1, type = NULL,
         xlim = NULL, ylim = NULL, xlab = "Number of occurrences",
         ylab = "Frequency ratio", main = "Ord plot", gp = gpar(cex = 0.5),
         name = "Ord_plot", newpage = TRUE, pop = TRUE, ...)
Ord_estimate(x, type = NULL, tol = 0.1)

```

## Arguments

obj	either a vector of counts, a 1-way table of frequencies of counts or a data frame or matrix with frequencies in the first column and the corresponding counts in the second column.
legend	logical. Should a legend be plotted?.
estimate	logical. Should the distribution and its parameters be estimated from the data? See details.
tol	tolerance for estimating the distribution. See details.
type	a character string indicating the distribution, must be one of "poisson", "binomial", "nbinomial" or "log-series" or NULL. In the latter case the distribution is estimated from the data. See details.
xlim	limits for the x axis.
ylim	limits for the y axis.

xlab	a label for the x axis.
ylab	a label for the y axis.
main	a title for the plot.
gp	a "gpar" object controlling the grid graphical parameters of the points.
name	name of the plotting viewport.
newpage	logical. Should <code>grid.newpage</code> be called before plotting?
pop	logical. Should the viewport created be popped?
...	further arguments passed to <code>grid.points</code> .
x	a vector giving intercept and slope for the (fitted) line in the Ord plot.

### Details

The Ord plot plots the number of occurrences against a certain frequency ratio (see Friendly (2000) for details) and should give a straight line if the data comes from a poisson, binomial, negative binomial or log-series distribution. The intercept and slope of this straight line conveys information about the underlying distribution.

Ord\_plot fits a usual OLS line (black) and a weighted OLS line (red). From the coefficients of the latter the distribution is estimated by Ord\_estimate as described in Table 2.10 in Friendly (2000). To judge whether a coefficient is positive or negative a tolerance given by tol is used. If none of the distributions fits well, no parameters are estimated. Be careful with the conclusions from Ord\_estimate as it implements just some simple heuristics!

### Value

A vector giving the intercept and slope of the weighted OLS line.

### Author(s)

Achim Zeileis <Achim.Zeileis@R-project.org>

### References

J. K. Ord (1967), Graphical methods for a class of discrete distributions, *Journal of the Royal Statistical Society*, A **130**, 232–238.

Michael Friendly (2000), *Visualizing Categorical Data*. SAS Institute, Cary, NC.

### Examples

```
## Simulated data examples:
dummy <- rnbinom(1000, size = 1.5, prob = 0.8)
Ord_plot(dummy)

## Real data examples:
data("HorseKicks")
data("Federalist")
data("Butterfly")
data("WomenQueue")
```

```

grid.newpage()
pushViewport(viewport(layout = grid.layout(2, 2)))

pushViewport(viewport(layout.pos.col=1, layout.pos.row=1))
Ord_plot(HorseKicks, main = "Death by horse kicks", newpage = FALSE)
popViewport()

pushViewport(viewport(layout.pos.col=1, layout.pos.row=2))
Ord_plot(Federalist, main = "Instances of 'may' in Federalist papers", newpage = FALSE)
popViewport()

pushViewport(viewport(layout.pos.col=2, layout.pos.row=1))
Ord_plot(Butterfly, main = "Butterfly species collected in Malaya", newpage = FALSE)
popViewport()

pushViewport(viewport(layout.pos.col=2, layout.pos.row=2))
Ord_plot(WomenQueue, main = "Women in queues of length 10", newpage = FALSE)
popViewport(2)

```

---

OvaryCancer

*Ovary Cancer Data*


---

### Description

Data from Obel (1975) about a retrospective study of ovary cancer carried out in 1973. Information was obtained from 299 women, who were operated for ovary cancer 10 years before.

### Usage

```
data("OvaryCancer")
```

### Format

A data frame with 16 observations and 5 variables.

**Freq** frequency.

**stage** factor indicating the stage of the cancer at the time of operation (early, advanced).

**operation** factor indicating type of operation (radical, limited).

**survival** factor indicating survival status after 10 years (yes, no).

**xray** factor indicating whether X-ray treatment was received (yes, no).

### Source

E. B. Andersen (1991), The Statistical Analysis of Categorical Data, Table 6.4.

## References

- E. B. Obel (1975), A Comparative Study of Patients with Cancer of the Ovary Who Have Survived More or Less Than 10 Years. *Acta Obstetricia et Gynecologica Scandinavica*, **55**, 429-439.
- E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*. 2nd edition. Springer-Verlag, Berlin.

## Examples

```
data("OvaryCancer")
tab <- xtabs(Freq ~ xray + survival + stage + operation, data = OvaryCancer)
fable(tab, col.vars = "survival", row.vars = c("stage", "operation", "xray"))

## model: ~ xray * operation * stage + survival * stage
## interpretation: treat xray, operation, stage as fixed margins,
## the survival depends on stage, but not xray and operation.
doubledecker(survival ~ stage + operation + xray, data = tab)
mosaic(~ stage + operation + xray + survival,
  split = c(FALSE, TRUE, TRUE, FALSE), data = tab, keep = FALSE,
  gp = gpar(fill = rev(grey.colors(2))))
mosaic(~ stage + operation + xray + survival,
  split = c(FALSE, TRUE, TRUE, FALSE), data = tab, keep = FALSE,
  expected = ~ xray * operation * stage + survival*stage)
```

---

Pairs plot panel functions for diagonal cells

*Diagonal Panel Functions for Table Pairs Plot*

---

## Description

Diagonal panel functions for [pairs.table](#).

## Usage

```
pairs_barplot(gp_bars = NULL,
  gp_vartext = gpar(fontsize = 17),
  gp_leveltext = gpar(),
  just_leveltext = c("center", "bottom"),
  just_vartext = c("center", "top"),
  rot = 0, abbreviate = FALSE, check_overlap = TRUE, fill = "grey",
  var_offset = unit(1, "npc"), ...)
pairs_text(dimnames = TRUE, gp_vartext = gpar(fontsize = 17),
  gp_leveltext = gpar(), gp_border = gpar(), ...)
pairs_diagonal_text(varnames = TRUE, gp_vartext = gpar(fontsize = 17, fontface = "bold"),
  gp_leveltext = gpar(), gp_border = gpar(), pos = c("right", "top"),
  distribute = c("equal", "margin"), rot = 0, ...)
pairs_diagonal_mosaic(split_vertical = TRUE, margins = unit(0, "lines"),
  offset_labels = -0.4, offset_varnames = 0,
  gp = NULL, fill = "grey", ...)
```

**Arguments**

<code>dimnames</code>	vector of logicals indicating whether the factor levels should be displayed (only used for <code>pairs_text</code> ).
<code>varnames</code>	vector of logicals indicating whether the variable names should be displayed (only used for <code>pairs_text_diagonal</code> ).
<code>gp_bars</code>	object of class "gpar" used for bars (only used for <code>pairs_barplot</code> ). If unspecified, the default is to set the fill component of this object to the fill argument.
<code>gp_vartext</code>	object of class "gpar" used for the factor names.
<code>gp_leveltext</code>	object of class "gpar" used for the factor levels.
<code>gp_border</code>	object of class "gpar" used for the border (only used for <code>pairs_text</code> ).
<code>gp</code>	object of class "gpar" used for the tiles (only used for <code>pairs_diagonal_mosaic</code> ). If unspecified, the default is to set the fill component of this object to the fill argument.
<code>fill</code>	color vector or palette function used for the fill colors of bars (for <code>pairs_barplot</code> ) or tiles (for <code>pairs_diagonal_mosaic</code> ).
<code>just_leveltext</code> , <code>just_vartext</code>	character string indicating the justification for variable names and levels.
<code>pos</code>	character string of length 2 controlling the horizontal and vertical position of the variable names (only used for <code>pairs_text_diagonal</code> ).
<code>rot</code>	rotation angle for the variable levels.
<code>distribute</code>	character string indicating whether levels should be distributed equally or according to the margins (only used for <code>pairs_text_diagonal</code> ).
<code>abbreviate</code>	integer or logical indicating the number of characters the labels should be abbreviated to. TRUE means 1 character, FALSE causes no abbreviation.
<code>check_overlap</code>	If TRUE, some levels will be suppressed to avoid overlapping, if any.
<code>split_vertical</code>	vector of logicals of length $k$ , where $k$ is the number of margins of $x$ (values are recycled as needed). A TRUE component indicates that the tile(s) of the corresponding dimension should be split vertically, FALSE means horizontal splits. Default is FALSE.
<code>margins</code>	either an object of class "unit" of length 4, or a numeric vector of length 4. The elements are recycled as needed. The four components specify the top, right, bottom, and left margin of the plot, respectively. When a numeric vector is supplied, the numbers are interpreted as "lines" units. In addition, the unit or numeric vector may have named arguments ('top', 'right', 'bottom', and 'left'), in which case the non-named arguments specify the default values (recycled as needed), overloaded by the named arguments.
<code>offset_labels</code> , <code>offset_varnames</code>	numeric vector of length 4 indicating the offset of the labels (variable names) for each of the four sides of the plot.
<code>var_offset</code>	object of class "unit" specifying the offset of variable names from the bottom of the bar plots created by <code>pairs_barplot</code> . If numeric, the unit defaults to "npc".
<code>...</code>	other parameters passed to the underlying graphics functions.

**Details**

In the diagonal cells, the pairsplot visualizes statistics or information for each dimension (that is: the single factors) alone. `pairs_text` displays the factor's name, and optionally also the factor levels. `pairs_barplot` produces a bar plot of the corresponding factor, along with the factor's name.

**Value**

A function with one argument: the marginal table for the corresponding dimension.

**Author(s)**

David Meyer <David.Meyer@R-project.org>

**See Also**

[pairs.table](#), [pairs\\_assoc](#), [pairs\\_mosaic](#)

**Examples**

```
data("UCBAdmissions")

pairs(UCBAdmissions) # pairs_barplot is default
pairs(UCBAdmissions, diag_panel = pairs_text)

pairs(UCBAdmissions, diag_panel = pairs_diagonal_text)
pairs(Titanic, diag_panel = pairs_diagonal_text)
pairs(Titanic, diag_panel = pairs_diagonal_text(distribute = "margin"))
pairs(Titanic,
      diag_panel = pairs_diagonal_text(distribute = "margin", rot = 45))
```

---

Pairs plot panel functions for off-diagonal cells

*Off-diagonal Panel Functions for Table Pairs Plot*

---

**Description**

Off-diagonal panel functions for [pairs.table](#).

**Usage**

```
pairs_strucplot(panel = mosaic,
  type = c("pairwise", "total", "conditional", "joint"),
  legend = FALSE, margins = c(0, 0, 0, 0), labeling = NULL, ...)
pairs_assoc(...)
pairs_mosaic(...)
pairs_sieve(...)
```

**Arguments**

panel	function to be used for the plots in each cell, such as <code>pairs_assoc</code> , <code>pairs_mosaic</code> , and <code>pairs_sieve</code> .
type	character string specifying the type of independence model visualized in the cells.
legend	logical specifying whether a legend should be displayed in the cells or not.
margins	margins inside each cell (see <code>strucplot</code> ).
labeling	labeling function or labeling-generating function (see <code>strucplot</code> ).
...	<code>pairs_mosaic</code> , <code>pairs_assoc</code> , and <code>pairs_sieve</code> : parameters passed to <code>pairs_strucplot</code> . <code>pairs_strucplot</code> : other parameters passed to panel function.

**Details**

These functions really just wrap `assoc`, `sieve`, and `mosaic` by basically inhibiting labeling and legend-drawing and setting the margins to 0.

**Value**

A function with arguments:

x	contingency table.
i, j	cell coordinates.

**Author(s)**

David Meyer <David.Meyer@R-project.org>

**References**

Cohen, A. (1980), On the graphical display of the significant components in a two-way contingency table. *Communications in Statistics—Theory and Methods*, **A9**, 1025–1041.

Friendly, M. (1992), Graphical methods for categorical data. *SAS User Group International Conference Proceedings*, **17**, 190–200. <http://datavis.ca/sugi/sugi17.pdf>

**See Also**

`pairs.table`, `pairs_text`, `pairs_barplot`, `assoc`, `mosaic`

**Examples**

```
data("UCBAdmissions")
data("PreSex")

pairs(PreSex)
pairs(UCBAdmissions)
pairs(UCBAdmissions, upper_panel_args = list(shade = FALSE))
pairs(UCBAdmissions, lower_panel = pairs_mosaic(type = "conditional"))
pairs(UCBAdmissions, upper_panel = pairs_assoc)
```

---

pairs.table                      *Pairs Plot for Contingency Tables*

---

### Description

Produces a matrix of strucplot displays.

### Usage

```
## S3 method for class 'table'
pairs(x, upper_panel = pairs_mosaic, upper_panel_args = list(),
      lower_panel = pairs_mosaic, lower_panel_args = list(),
      diag_panel = pairs_barplot, diag_panel_args = list(),
      main = NULL, sub = NULL, main_gp = gpar(fontsize = 20),
      sub_gp = gpar(fontsize = 15), space = 0.3,
      newpage = TRUE, pop = TRUE, margins = unit(1, "lines"), ...)
```

### Arguments

x	a contingency table in array form, with optional category labels specified in the <code>dimnames(x)</code> attribute.
upper_panel	function for the upper triangle of the matrix, or corresponding generating function. If NULL, no panel is drawn.
upper_panel_args	list of arguments for the generating function, if specified.
lower_panel	function for the lower triangle of the matrix, or corresponding generating function. If NULL, no panel is drawn.
lower_panel_args	list of arguments for the panel-generating function, if specified.
diag_panel	function for the diagonal of the matrix, or corresponding generating function. If NULL, no panel is drawn.
diag_panel_args	list of arguments for the generating function, if specified.
main	either a logical, or a character string used for plotting the main title. If main is a logical and TRUE, the name of the object supplied as x is used.
sub	a character string used for plotting the subtitle. If sub is a logical and TRUE and main is unspecified, the name of the object supplied as x is used.
main_gp, sub_gp	object of class "gpar" containing the graphical parameters used for the main (sub) title, if specified.
space	double specifying the distance between the cells.
newpage	logical controlling whether a new grid page should be created.
pop	logical indicating whether all viewports should be popped after the plot has been drawn.

margins	either an object of class "unit" of length 4, or a numeric vector of length 4. The elements are recycled as needed. The four components specify the top, right, bottom, and left margin of the plot, respectively. When a numeric vector is supplied, the numbers are interpreted as "lines" units. In addition, the unit or numeric vector may have named arguments ('top', 'right', 'bottom', and 'left'), in which case the non-named arguments specify the default values (recycled as needed), overloaded by the named arguments.
...	For convenience, list of arguments for the panel-generating functions of upper and lower panels, if specified.

## Details

This is a `pairs` method for objects inheriting from class "table" or "structable". It plots a matrix of pairwise mosaic plots.

Four independence types are distinguished: "pairwise", "total", "conditional" and "joint". The pairwise mosaic matrix shows bivariate marginal relations, collapsed over all other variables. The total independence mosaic matrix shows mosaic plots for mutual independence, i.e., for marginal and conditional independence among all pairs of variables. The conditional independence mosaic matrix shows mosaic plots for marginal independence given all other variables. The joint independence mosaic matrix shows mosaic plots for joint independence of all pairs of variables from the others.

This method uses panel functions called for each cell of the matrix which can be different for upper matrix, lower matrix, and diagonal cells. Correspondingly, for each panel parameter `foo` (= 'upper', 'lower', or 'diag'), `pairs.table` takes two arguments: `foo\panel` and `foo\panel\_args`, which can be used to specify the parameters as follows:

1. Passing a suitable panel function to `foo\panel` which subsequently is called for each cell with the corresponding coordinates.
2. Passing a corresponding *generating function* (of class "panel\_generator") to `foo\panel`, along with parameters passed to `foo\panel\_args`, that generates such a function.

Hence, the second approach is equivalent to the first if `foo\panel(foo\panel\_args)` is passed to `foo\panel`.

## Author(s)

David Meyer <David.Meyer@R-project.org>

## References

- Cohen, A. (1980), On the graphical display of the significant components in a two-way contingency table. *Communications in Statistics—Theory and Methods*, **A9**, 1025–1041.
- Friendly, M. (1992), Graphical methods for categorical data. *SAS User Group International Conference Proceedings*, **17**, 190–200. <http://datavis.ca/sugi/sugi17.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. URL <http://www.jstatsoft.org/v17/i03/> and available as `vignette("strucplot")`.

**See Also**

[pairs\\_mosaic](#), [pairs\\_assoc](#), [pairs\\_sieve](#), [pairs\\_diagonal\\_text](#), [pairs\\_text](#), [pairs\\_barplot](#), [assoc](#), [sieve](#), [mosaic](#)

**Examples**

```
data("UCBAdmissions")
data("PreSex")
data(HairEyeColor)
hec = structable(Eye ~ Sex + Hair, data = HairEyeColor)

pairs(PreSex)
pairs(UCBAdmissions)
pairs(UCBAdmissions, upper_panel_args = list(shade = TRUE))
pairs(UCBAdmissions, lower_panel = pairs_mosaic(type = "conditional"))
pairs(UCBAdmissions, diag_panel = pairs_text)
pairs(UCBAdmissions, upper_panel = pairs_assoc, shade = TRUE)
pairs(hec, highlighting = 2, diag_panel_args = list(fill = grey.colors))
pairs(hec, highlighting = 2, diag_panel = pairs_diagonal_mosaic,
      diag_panel_args = list(fill = grey.colors, alternate_labels = TRUE))
```

---

plot.loglm

*Visualize Fitted Log-linear Models*


---

**Description**

Visualize fitted "loglm" objects by mosaic or association plots.

**Usage**

```
## S3 method for class 'loglm'
plot(x, panel = mosaic, type = c("observed", "expected"),
     residuals_type = c("pearson", "deviance"), gp = shading_hcl, gp_args = list(),
     ...)
```

**Arguments**

x	a fitted "loglm" object, see <a href="#">loglm</a> .
panel	a panel function for visualizing the observed values, residuals and expected values. Currently, <a href="#">mosaic</a> and <a href="#">assoc</a> in <a href="#">vcd</a> .
type	a character string indicating whether the observed or the expected values of the table should be visualized.
residuals_type	a character string indicating the type of residuals to be computed.
gp	object of class "gpar", shading function or a corresponding generating function (see details and <a href="#">shadings</a> ). Ignored if shade = FALSE.
gp_args	list of arguments for the shading-generating function, if specified.
...	Other arguments passed to the panel function.

**Details**

The plot method for "loglm" objects by default visualizes the model using a mosaic plot (can be changed to an association plot by setting `panel = assoc`) with a shading based on the residuals of this model. The legend also reports the corresponding p value of the associated goodness-of-fit test. The mosaic and assoc methods are simple convenience interfaces to this plot method, setting the `panel` argument accordingly.

**Value**

The "structable" visualized is returned invisibly.

**Author(s)**

Achim Zeileis <Achim.Zeileis@R-project.org>

**See Also**

[loglm](#), [assoc](#), [mosaic](#), [strucplot](#)

**Examples**

```
## mosaic display for PreSex model
data("PreSex")
fm <- loglm(~ PremaritalSex * ExtramaritalSex * (Gender + MaritalStatus),
  data = aperm(PreSex, c(3, 2, 4, 1)))
fm
## visualize Pearson statistic
plot(fm, split_vertical = TRUE)
## visualize LR statistic
plot(fm, split_vertical = TRUE, residuals_type = "deviance")

## conditional independence in UCB admissions data
data("UCBAdmissions")
fm <- loglm(~ Dept * (Gender + Admit), data = aperm(UCBAdmissions))

## use mosaic display
plot(fm, labeling_args = list(abbreviate = c(Admit = 3)))

## and association plot
plot(fm, panel = assoc)
assoc(fm)
```

---

PreSex

*Pre-marital Sex and Divorce*

---

**Description**

Data from Thornes & Collard (1979), reported in Gilbert (1981), on pre- and extra-marital sex and divorce.

**Usage**

```
data("PreSex")
```

**Format**

A 4-dimensional array resulting from cross-tabulating 1036 observations on 4 variables. The variables and their levels are as follows:

No	Name	Levels
1	MaritalStatus	Divorced, Married
2	ExtramaritalSex	Yes, No
3	PremaritalSex	Yes, No
4	Gender	Women, Men

**Source**

Michael Friendly (2000), *Visualizing Categorical Data*: <http://euclid.psych.yorku.ca/ftp/sas/vcd/catdata/marital.sas>

**References**

G. N. Gilbert (1981), *Modelling Society: An Introduction to Loglinear Analysis for Social Researchers*. Allen and Unwin, London.

B. Thornes & J. Collard (1979), *Who Divorces?*. Routledge & Kegan, London.

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

**Examples**

```
data("PreSex")

## Mosaic display for Gender and Premarital Sexual Experience
## (Gender Pre)
mosaic(margin.table(PreSex, c(3,4)),
       main = "Gender and Premarital Sex")

## (Gender Pre)(Extra)
mosaic(margin.table(PreSex, c(2,3,4)),
       expected = ~Gender * PremaritalSex + ExtramaritalSex ,
       main = "PreMaritalSex*Gender +Sex")

## (Gender Pre Extra)(Marital)
mosaic(PreSex,
       expected = ~Gender*PremaritalSex*ExtramaritalSex + MaritalStatus,
       main = "PreMarital*ExtraMarital + MaritalStatus")

## (GPE)(PEM)
mosaic(PreSex,
       expected = ~ Gender * PremaritalSex * ExtramaritalSex
              + MaritalStatus * PremaritalSex * ExtramaritalSex,
       main = "G*P*E + P*E*M")
```

---

Punishment

*Corporal Punishment Data*

---

### Description

Data from a study of the Gallup Institute in Denmark in 1979 about the attitude of a random sample of 1,456 persons towards corporal punishment of children.

### Usage

```
data("Punishment")
```

### Format

A data frame with 36 observations and 5 variables.

**Freq** frequency.

**attitude** factor indicating attitude: (no, moderate) punishment of children.

**memory** factor indicating whether the person had memories of corporal punishment as a child (yes, no).

**education** factor indicating highest level of education (elementary, secondary, high).

**age** factor indicating age group in years (15-24, 25-39, 40-).

### Note

Anderson (1991) erroneously indicates the total sum of respondents to be 783.

### Source

E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*, pages 207–208.

### References

E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*. 2nd edition. Springer-Verlag, Berlin.

### Examples

```
data("Punishment", package = "vcd")
pun <- xtabs(Freq ~ memory + attitude + age + education, data = Punishment)

## model: ~ (memory + attitude) * age * education
## use maximum sum-of-squares test/shading
cotabplot(~ memory + attitude | age + education, data = pun, panel = cotab_coindep,
  n = 5000, type = "assoc", test = "maxchisq", interpolate = 1:2)
```

RepVict

*Repeat Victimization Data***Description**

Data from Reiss (1980) given by Fienberg (1980) about instances of repeat victimization for households in the U.S. National Crime Survey.

**Usage**

```
data("RepVict")
```

**Format**

A 2-dimensional array resulting from cross-tabulating victimization. The variables and their levels are as follows:

No	Name	Levels
1	First Victimization	Rape, Assault, Robbery, Pickpocket, Personal Larceny, Burglary, Household Larceny, Auto Theft
2	Second Victimization	Rape, Assault, Robbery, Pickpocket, Personal Larceny, Burglary, Household Larceny, Auto Theft

**Source**

Michael Friendly (2000), *Visualizing Categorical Data*, page 113.

**References**

S. E. Fienberg (1980), *The Analysis of Cross-Classified Categorical Data*, MIT Press, Cambridge, 2nd edition.

A. J. J. Reiss (1980), Victim proneness by type of crime in repeat victimization. In S. E. Fienberg & A. J. J. Reiss (eds.), *Indicators of Crime and Criminal Justice*. U.S. Government Printing Office, Washington, DC.

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

**Examples**

```
data("RepVict")

mosaic(RepVict[-c(4,7),-c(4,7)], gp = shading_max,
       main = "Repeat Victimization Data")
```

---

Rochdale

*Rochdale Data*


---

**Description**

Information on 665 households of Rochdale, Lancashire, UK. The study was conducted to identify influence factors on economical activity of wives.

**Usage**

```
data("Rochdale")
```

**Format**

A 8-dimensional array resulting from cross-tabulating 665 observations on 8 variables. The variables and their levels are as follows:

No	Name	Levels
1	EconActive	yes, no
2	Age	<38, >38
3	HusbandEmployed	yes, no
4	Child	yes, no
5	Education	yes, no
6	HusbandEducation	yes, no
7	Asian	yes, no
8	HouseholdWorking	yes, no

**Note**

Many observations are missing: only 91 out of all 256 combinations contain information.

**Source**

Whittaker (1990).

**References**

H. Hofmann (2003). Constructing and reading mosaicplots. *Computational Statistics & Data Analysis*, **43**, 4, 565–580.

J. Whittaker (1990), *Graphical Models on Applied Multivariate Statistics*, Wiley, New York.

**Examples**

```
data("Rochdale")
mosaic(Rochdale)
```

---

 rootogram

*Rootograms*


---

## Description

Rootograms of observed and fitted values.

## Usage

```
## Default S3 method:
rootogram(x, fitted, names = NULL, scale = c("sqrt", "raw"),
  type = c("hanging", "standing", "deviation"),
  rect_gp = gpar(fill = "lightgray"), lines_gp = gpar(col = "red"),
  points_gp = gpar(col = "red"), pch = 19,
  xlab = NULL, ylab = NULL, ylim = NULL,
  name = "rootogram", newpage = TRUE, pop = TRUE, ...)
```

## Arguments

x	either a vector or a 1-way table of frequencies.
fitted	a vector of fitted frequencies.
names	a vector of names passed to <a href="#">grid_barplot</a> , if set to NULL the names of x are used.
scale	a character string indicating whether the values should be plotted on the raw or square root scale.
type	a character string indicating if the bars for the observed frequencies should be hanging or standing or indicate the deviation between observed and fitted frequencies.
rect_gp	a "gpar" object controlling the grid graphical parameters of the rectangles.
lines_gp	a "gpar" object controlling the grid graphical parameters of the lines.
points_gp	a "gpar" object controlling the grid graphical parameters of the points.
pch	plotting character for the points.
xlab	a label for the x axis.
ylab	a label for the y axis.
ylim	limits for the y axis.
name	name of the plotting viewport.
newpage	logical. Should <a href="#">grid.newpage</a> be called before plotting?
pop	logical. Should the viewport created be popped?
...	further arguments passed to <a href="#">grid_barplot</a> .

## Details

The observed frequencies are displayed as bars and the fitted frequencies as a line. By default a sqrt scale is used to make the smaller frequencies more visible.

**Author(s)**

Achim Zeileis <Achim.Zeileis@R-project.org>

**References**

J. W. Tukey (1977), *Exploratory Data Analysis*. Addison Wesley, Reading, MA.

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

**See Also**

[grid\\_barplot](#)

**Examples**

```
## Simulated data examples:
dummy <- rnbinom(200, size = 1.5, prob = 0.8)
observed <- table(dummy)
fitted1 <- dnbinom(as.numeric(names(observed)),
                  size = 1.5, prob = 0.8) * sum(observed)
fitted2 <- dnbinom(as.numeric(names(observed)),
                  size = 2, prob = 0.6) * sum(observed)
rootogram(observed, fitted1)
rootogram(observed, fitted2)

## Real data examples:
data("HorseKicks")
HK.fit <- goodfit(HorseKicks)
summary(HK.fit)
plot(HK.fit)
## or equivalently
rootogram(HK.fit)

data("Federalist")
F.fit <- goodfit(Federalist, type = "nbinomial")
summary(F.fit)
plot(F.fit)
```

---

Saxony

*Families in Saxony*

---

**Description**

Data from Geissler, cited in Sokal & Rohlf (1969) and Lindsey (1995) on gender distributions in families in Saxony in the 19th century.

**Usage**

```
data("Saxony")
```

**Format**

A 1-way table giving the number of male children in 6115 families of size 12. The variable and its levels are

No	Name	Levels
1	nMales	0, 1, ..., 12

**Source**

M. Friendly (2000), *Visualizing Categorical Data*, pages 40–42.

**References**

J. K. Lindsey (1995), *Analysis of Frequency and Count Data*. Oxford University Press, Oxford, UK.

R. R. Sokal & F. J. Rohlf (1969), *Biometry. The Principles and Practice of Statistics*. W. H. Freeman, San Francisco, CA.

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

**Examples**

```
data("Saxony")
gf <- goodfit(Saxony, type = "binomial")
summary(gf)
plot(gf)
```

---

SexualFun

*Sex is Fun*

---

**Description**

Data from Hout et al. (1987) given by Agresti (1990) summarizing the responses of married couples to the questionnaire item: Sex is fun for me and my partner: (a) never or occasionally, (b) fairly often, (c) very often, (d) almost always.

**Usage**

```
data("SexualFun")
```

**Format**

A 2-dimensional array resulting from cross-tabulating the ratings of 91 married couples. The variables and their levels are as follows:

No	Name	Levels
1	Husband	Never Fun, Fairly Often, Very Often, Always Fun
2	Wife	Never Fun, Fairly Often, Very Often, Always Fun

**Source**

M. Friendly (2000), *Visualizing Categorical Data*, page 91.

**References**

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

M. Hout, O. D. Duncan, M. E. Sobel (1987), Association and heterogeneity: Structural models of similarities and differences, *Sociological Methodology*, **17**, 145-184.

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

**Examples**

```
data("SexualFun")

## Kappa statistics
Kappa(SexualFun)

## Agreement Chart
agreementplot(t(SexualFun), weights = 1)
## Partial Agreement Chart and B-Statistics
agreementplot(t(SexualFun),
               xlab = "Husband's Rating",
               ylab = "Wife's Rating",
               main = "Husband's and Wife's Sexual Fun")
```

---

shadings

*Shading-generating Functions for Residual-based Shadings*


---

**Description**

Shading-generating functions for computing residual-based shadings for mosaic and association plots.

**Usage**

```
shading_hcl(observed, residuals = NULL, expected = NULL, df = NULL,
            h = NULL, c = NULL, l = NULL, interpolate = c(2, 4), lty = 1,
            eps = NULL, line_col = "black", p.value = NULL, level = 0.95, ...)
```

```
shading_hsv(observed, residuals = NULL, expected = NULL, df = NULL,
            h = c(2/3, 0), s = c(1, 0), v = c(1, 0.5),
            interpolate = c(2, 4), lty = 1, eps = NULL, line_col = "black",
            p.value = NULL, level = 0.95, ...)
```

```
shading_max(observed = NULL, residuals = NULL, expected = NULL, df = NULL,
            h = NULL, c = NULL, l = NULL, lty = 1, eps = NULL, line_col = "black",
            level = c(0.9, 0.99), n = 1000, ...)
```

```
shading_Friendly(observed = NULL, residuals = NULL, expected = NULL, df = NULL,
  h = c(2/3, 0), lty = 1:2, interpolate = c(2, 4),
  eps = 0.01, line_col = "black", ...)
```

```
shading_sieve(observed = NULL, residuals = NULL, expected = NULL, df = NULL,
  h = c(260, 0), lty = 1:2, interpolate = c(2, 4),
  eps = 0.01, line_col = "black", ...)
```

```
shading_binary(observed = NULL, residuals = NULL, expected = NULL, df = NULL,
  col = NULL)
```

```
hcl2hex(h = 0, c = 35, l = 85, fixup = TRUE)
```

### Arguments

observed	contingency table of observed values
residuals	contingency table of residuals
expected	contingency table of expected values
df	degrees of freedom of the associated independence model.
h	hue value in the HCL or HSV color description, has to be in [0, 360] for HCL and in [0, 1] for HSV colors. The default is to use blue and red for positive and negative residuals respectively. In the HCL specification it is c(260, 0) by default and for HSV c(2/3, 0).
c	chroma value in the HCL color description. This controls the maximum chroma for significant and non-significant results respectively and defaults to c(100, 20).
l	luminance value in the HCL color description. Defaults to c(90, 50) for small and large residuals respectively.
s	saturation value in the HSV color description. Defaults to c(1, 0) for large and small residuals respectively.
v	saturation value in the HSV color description. Defaults to c(1, 0.5) for significant and non-significant results respectively.
interpolate	a specification for mapping the absolute size of the residuals to a value in [0, 1]. This can be either a function or a numeric vector. In the latter case, a step function with steps of equal size going from 0 to 1 is used.
lty	a vector of two line types for positive and negative residuals respectively. Recycled if necessary.
eps	numeric tolerance value below which absolute residuals are considered to be zero, which is used for coding the border color and line type. If set to NULL (default), all borders have the default color specified by line_col. If set to a numeric value, all border colors corresponding to residuals with a larger absolute value are set to the full positive or negative color, respectively; borders corresponding to smaller residuals are drawn with line_col and lty[1]. This is used principally in shading_Friendly.

<code>line_col</code>	default border color (for <code>shading_sieve</code> : default sieve color).
<code>p.value</code>	the $p$ value associated with the independence model. By default, this is computed from a Chi-squared distribution with <code>df</code> degrees of freedom. <code>p.value</code> can be either a scalar or a function( <code>observed</code> , <code>residuals</code> , <code>expected</code> , <code>df</code> ) that computes the $p$ value from the data. If set to NA no inference is performed.
<code>level</code>	confidence level of the test used. If <code>p.value</code> is smaller than $1 - \text{level}$ , bright colors are used, otherwise dark colors are employed. For <code>shading_max</code> a vector of levels can be supplied. The corresponding critical values are then used as interpolate cut-offs.
<code>n</code>	number of permutations used in the call to <code>coindep_test</code> .
<code>col</code>	a vector of two colors for positive and negative residuals respectively.
<code>fixup</code>	logical. Should the color be corrected to a valid RGB value before correction?
<code>...</code>	Other arguments passed to <code>hcl2hex</code> or <code>hsv</code> , respectively.

## Details

These shading-generating functions can be passed to `strucplot` to generate residual-based shadings for contingency tables. `strucplot` calls these functions with the arguments `observed`, `residuals`, `expected`, `df` which give the observed values, residuals, expected values and associated degrees of freedom for a particular contingency table and associated independence model.

The shadings `shading_hcl` and `shading_hsv` do the same thing conceptually, but use HCL or HSV colors respectively. The former is usually preferred because they are perceptually based. Both shadings visualize the *sign* of the residuals of an independence model using two hues (by default: blue and red). The *absolute size* of the residuals is visualized by the colorfulness and the amount of grey, by default in three categories: very colorful for large residuals ( $> 4$ ), less colorful for medium sized residuals ( $< 4$  and  $> 2$ ), grey/white for small residuals ( $< 2$ ). More categories or a continuous scale can be specified by setting `interpolate`. Furthermore, the result of a significance test can be visualized by the amount of grey in the colors. If significant, a colorful palette is used, if not, the amount of color is reduced. See Zeileis, Meyer, and Hornik (2007) and `diverge_hcl` for more details.

The shading `shading_max` is applicable in 2-way contingency tables and uses a similar strategy as `shading_hcl`. But instead of using the cut-offs 2 and 4, it employs the critical values for the maximum statistic (by default at 90% and 99%). Consequently, color in the plot signals a significant result at 90% or 99% significance level, respectively. The test is carried out by calling `coindep_test`.

The shading `shading_Friendly` is very similar to `shading_hsv`, but additionally codes the sign of the residuals by different line types. See Friendly (1994) for more details. `shading_sieve` is similar, but uses HCL colors.

The shading `shading_binary` just visualizes the sign of the residuals by using two different colors (default: blue HCL(260, 50, 70) and red HCL(0, 50, 70)).

The color implementations employed are `hsv` from base R and `polarLUV` from the `colorspace` package, respectively. To transform the HCL coordinates to a hexadecimal color string (as returned by `hsv`), the function `hex` is employed. A convenience wrapper `hcl2hex` is provided.

**Value**

A shading function which takes only a single argument, interpreted as a vector/table of residuals, and returns a "gpar" object with the corresponding vector(s)/table(s) of graphical parameter(s).

**Author(s)**

Achim Zeileis <Achim.Zeileis@R-project.org>

**References**

Friendly M. (1994), Mosaic Displays for Multi-Way Contingency Tables. *Journal of the American Statistical Association*, **89**, 190–200.

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. URL <http://www.jstatsoft.org/v17/i03/>. See also `vignette("strucplot", package = "vcd")`.

Zeileis A., Meyer D., Hornik K. (2007), Residual-Based Shadings for Visualizing (Conditional) Independence. *Journal of Computational and Graphical Statistics*, **16**, 507–525.

Zeileis A., Hornik K. and Murrell P. (2008), Escaping RGBland: Selecting Colors for Statistical Graphics. *Computational Statistics & Data Analysis*, Forthcoming. Preprint available from <http://statmath.wu-wien.ac.at/~zeileis/papers/Zeileis+Hornik+Murrell-2008.pdf>.

**See Also**

[hex](#), [polarLUV](#), [hsv](#), [mosaic](#), [assoc](#), [strucplot](#), [diverge\\_hcl](#)

**Examples**

```
## load Arthritis data
data("Arthritis")
art <- xtabs(~Treatment + Improved, data = Arthritis)

## plain mosaic display without shading
mosaic(art)

## with shading for independence model
mosaic(art, shade = TRUE)
## which uses the HCL shading
mosaic(art, gp = shading_hcl)
## the residuals are too small to have color,
## hence the cut-offs can be modified
mosaic(art, gp = shading_hcl, gp_args = list(interpolate = c(1, 1.8)))
## the same with the Friendly palette
## (without significance testing)
mosaic(art, gp = shading_Friendly, gp_args = list(interpolate = c(1, 1.8)))

## assess independence using the maximum statistic
## cut-offs are now critical values for the test statistic
mosaic(art, gp = shading_max)

## association plot with shading as in base R
```

```
assoc(art, gp = shading_binary(col = c(1, 2)))
```

sieve

*Extended Sieve Plots*

## Description

(Extended) sieve displays for n-way contingency tables: plots rectangles with areas proportional to the expected cell frequencies and filled with a number of squares equal to the observed frequencies. Thus, the densities visualize the deviations of the observed from the expected values.

## Usage

```
## Default S3 method:
sieve(x, condvars = NULL, gp = NULL, shade = NULL,
      legend = FALSE, split_vertical = NULL, direction = NULL, spacing = NULL,
      spacing_args = list(), sievetype = c("observed", "expected"),
      gp_tile = gpar(), main = NULL, sub = NULL, ...)
## S3 method for class 'formula'
sieve(formula, data, ..., main = NULL, sub = NULL, subset = NULL)
```

## Arguments

x	a contingency table in array form, with optional category labels specified in the <code>dimnames(x)</code> attribute.
condvars	vector of integers or character strings indicating conditioning variables, if any. The table will be permuted to order them first.
formula	a formula specifying the variables used to create a contingency table from data. For convenience, conditioning formulas can be specified; the conditioning variables will then be used first for splitting. Formulas for sieve displays (unlike those for doubledecker plots) have no response variable.
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.
shade	logical specifying whether gp should be used or not (see gp). If TRUE and expected is unspecified, a default model is fitted: if condvars is specified, a corresponding conditional independence model, and else the total independence model. If shade is NULL (default), gp is used if specified.
sievetype	logical indicating whether rectangles should be filled according to observed or expected frequencies.
gp	object of class "gpar", shading function or a corresponding generating function (see details of <a href="#">strucplot</a> and <a href="#">shadings</a> ). Components of "gpar" objects are recycled as needed along the last splitting dimension. The default is a modified version of <a href="#">shading_Friendly</a> : if sievetype is "observed", cells with positive residuals are painted with a red sieve, and cells with negative residuals with a blue one. If sievetype is "expected", the sieves' color is gray. Ignored if shade = FALSE.

<code>gp_tile</code>	object of class "gpar", controlling the appearance of all <i>static</i> elements of the cells (e.g., border and fill color).
<code>legend</code>	either a legend-generating function, a legend function (see details of <a href="#">strucplot</a> and <a href="#">legends</a> ), or a logical value. If <code>legend</code> is NULL or TRUE and <code>gp</code> is a function, <code>legend</code> defaults to <a href="#">legend_resbased</a> .
<code>split_vertical</code>	vector of logicals of length $k$ , where $k$ is the number of margins of $x$ (default: FALSE). Values are recycled as needed. A TRUE component indicates that the tile(s) of the corresponding dimension should be split vertically, FALSE means horizontal splits. Ignored if <code>direction</code> is not NULL.
<code>direction</code>	character vector of length $k$ , where $k$ is the number of margins of $x$ (values are recycled as needed). For each component, a value of "h" indicates that the tile(s) of the corresponding dimension should be split horizontally, whereas "v" indicates vertical split(s).
<code>spacing</code>	spacing object, spacing function, or corresponding generating function (see <a href="#">strucplot</a> for more information). The default is no spacing at all if $x$ has two dimensions, and <code>spacing_increase</code> for more dimensions.
<code>spacing_args</code>	list of arguments for the generating function, if specified (see <a href="#">strucplot</a> for more information).
<code>main, sub</code>	either a logical, or a character string used for plotting the main (sub) title. If logical and TRUE, the name of the data object is used.
<code>...</code>	Other arguments passed to <a href="#">strucplot</a>

### Details

`sieve` is a generic function which currently has a default method and a formula interface. Both are high-level interfaces to the [strucplot](#) function, and produce (extended) sieve displays. Most of the functionality is described there, such as specification of the independence model, labeling, legend, spacing, shading, and other graphical parameters.

The layout is very flexible: the specification of shading, labeling, spacing, and legend is modularized (see [strucplot](#) for details).

### Value

The "structable" visualized is returned invisibly.

### Note

To be faithful to the original definition by Riedwyl & Schüpbach, the default is to have no spacing between the tiles for two-way tables.

### Author(s)

David Meyer <David.Meyer@R-project.org>

## References

H. Riedwyl & M. Schüpbach (1994), Parquet diagram to plot contingency tables. In F. Faulbaum (ed.), *Softstat '93: Advances in Statistical Software*, 293–299. Gustav Fischer, New York.

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

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. URL <http://www.jstatsoft.org/v17/i03/> and available as vignette("strucplot").

## See Also

[assoc](#), [strucplot](#), [mosaic](#), [structable](#), [doubledecker](#)

## Examples

```
data("HairEyeColor")

## aggregate over 'sex':
(tab <- margin.table(HairEyeColor, c(2,1)))

## plot expected values:
sieve(tab, sievetype = "expected", shade = TRUE)

## plot observed table:
sieve(tab, shade = TRUE)

## plot complete diagram:
sieve(HairEyeColor, shade = TRUE)

## an example for the formula interface:
data("VisualAcuity")
sieve(Freq ~ right + left, data = VisualAcuity)

## example with observed values in the cells:
sieve(Titanic, pop = FALSE, shade = TRUE)
labeling_cells(text = Titanic, gp_text = gpar(fontface = 2))(Titanic)
```

---

SpaceShuttle

*Space Shuttle O-ring Failures*

---

## Description

Data from Dalal et al. (1989) about O-ring failures in the NASA space shuttle program. The damage index comes from a discussion of the data by Tufte (1997).

## Usage

```
data("SpaceShuttle")
```

**Format**

A data frame with 24 observations and 6 variables.

**FlightNumber** Number of space shuttle flight.

**Temperature** temperature during start (in degrees F).

**Pressure** pressure.

**Fail** did any O-ring failures occur? (no, yes).

**nFailures** how many (of six) O-rings failed?.

**Damage** damage index.

**Source**

Michael Friendly (2000), Visualizing Categorical Data: <http://euclid.psych.yorku.ca/ftp/sas/vcd/catdata/orings.sas>

**References**

S. Dalal, E. B. Fowlkes, B. Hoadly (1989), Risk analysis of the space shuttle: Pre-Challenger prediction of failure, *Journal of the American Statistical Association*, **84**, 945–957.

E. R. Tufté (1997), *Visual Explanations: Images and Quantities, Evidence and Narrative*. Graphics Press, Cheshire, CT.

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

**Examples**

```
data("SpaceShuttle")
plot(nFailures/6 ~ Temperature, data = SpaceShuttle,
     xlim = c(30, 81), ylim = c(0,1),
     main = "NASA Space Shuttle O-Ring Failures",
     ylab = "Estimated failure probability",
     pch = 19, col = 4)
fm <- glm(cbind(nFailures, 6 - nFailures) ~ Temperature,
         data = SpaceShuttle,
         family = binomial)
lines(30 : 81,
      predict(fm, data.frame(Temperature = 30 : 81), type = "re"),
      lwd = 2)
abline(v = 31, lty = 3)
```

---

 spacings *Spacing-generating Functions*


---

**Description**

These functions generate spacing functions to be used with `strucplot` to obtain customized spaces between the elements of a `strucplot`.

**Usage**

```
spacing_equal(sp = unit(0.3, "lines"))
spacing_dimequal(sp)
spacing_increase(start = unit(0.3, "lines"), rate = 1.5)
spacing_conditional(sp = unit(0.3, "lines"), start = unit(2, "lines"), rate = 1.8)
spacing_highlighting(start = unit(0.2, "lines"), rate = 1.5)
```

**Arguments**

<code>start</code>	object of class "unit" indicating the start value for increasing spacings.
<code>rate</code>	increase rate for spacings.
<code>sp</code>	object of class "unit" specifying a fixed spacing.

**Details**

These generating functions return a function used by `strucplot` to generate appropriate spaces between tiles of a `strucplot`, using the `dimnames` information of the visualized table.

`spacing_equal` allows to specify one fixed space for *all* dimensions.

`spacing_dimequal` allows to specify a fixed space for *each* dimension.

`spacing_increase` creates increasing spaces for all dimensions, based on a starting value and an increase rate.

`spacing_conditional` combines `spacing_equal` and `spacing_increase` to create fixed spaces for conditioned dimensions, and increasing spaces for conditioning dimensions.

`spacing_highlighting` is essentially `spacing_conditional` but with the space of the last dimension set to 0. With a corresponding color scheme, this gives the impression of the last class being 'highlighted' in the penultimate class (as, e.g., in `doubledecker` plots).

**Value**

A spacing function with arguments:

<code>d</code>	"dim" attribute of a contingency table.
<code>condvars</code>	index vector of conditioning dimensions (currently only used by <code>spacing_conditional</code> ).

This function computes a list of objects of class "unit". Each list element contains the spacing information for the corresponding dimension of the table. The length of the "unit" objects is  $k - 1$ ,  $k$  number of levels of the corresponding factor.

**Author(s)**

David Meyer <David.Meyer@R-project.org>

**References**

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. URL <http://www.jstatsoft.org/v17/i03/> and available as vignette("strucplot").

**See Also**

[strucplot](#), [doubledecker](#)

**Examples**

```
data("Titanic")
strucplot(Titanic, spacing = spacing_increase(start = 0.5, rate = 1.5))
strucplot(Titanic, spacing = spacing_equal(1))
strucplot(Titanic, spacing = spacing_dimequal(1:4 / 4))
strucplot(Titanic, spacing = spacing_highlighting,
          gp = gpar(fill = c("light gray", "dark gray")))
data("PreSex")
strucplot(aperm(PreSex, c(1,4,2,3)), spacing = spacing_conditional,
          condvars = 2)
```

---

 spine

*Spine Plots and Spinograms*


---

**Description**

Spine plots are a special cases of mosaic plots, and can be seen as a generalization of stacked (or highlighted) bar plots. Analogously, spinograms are an extension of histograms.

**Usage**

```
spine(x, ...)
## Default S3 method:
spine(x, y = NULL,
      breaks = NULL, ylab_tol = 0.05, off = NULL,
      main = "", xlab = NULL, ylab = NULL, ylim = c(0, 1), margins = c(5.1, 4.1, 4.1, 3.1),
      gp = gpar(), name = "spineplot", newpage = TRUE, pop = TRUE,
      ...)
## S3 method for class 'formula'
spine(formula, data = list(),
      breaks = NULL, ylab_tol = 0.05, off = NULL,
      main = "", xlab = NULL, ylab = NULL, ylim = c(0, 1), margins = c(5.1, 4.1, 4.1, 3.1),
      gp = gpar(), name = "spineplot", newpage = TRUE, pop = TRUE,
      ...)
```

**Arguments**

x	an object, the default method expects either a single variable (interpreted to be the explanatory variable) or a 2-way table. See details.
y	a "factor" interpreted to be the dependent variable
formula	a "formula" of type $y \sim x$ with a single dependent "factor" and a single explanatory variable.
data	an optional data frame.
breaks	if the explanatory variable is numeric, this controls how it is discretized. breaks is passed to <a href="#">hist</a> and can be a list of arguments.
ylab_tol	convenience tolerance parameter for y-axis annotation. If the distance between two labels drops under this threshold, they are plotted equidistantly.
off	vertical offset between the bars (in per cent). It is fixed to 0 for spinograms and defaults to 2 for spine plots.
main, xlab, ylab	character strings for annotation
ylim	limits for the y axis
margins	margins when calling <a href="#">plotViewport</a>
gp	a "gpar" object controlling the grid graphical parameters of the rectangles. It should specify in particular a vector of fill colors of the same length as <code>levels(y)</code> . The default is to call <a href="#">gray.colors</a> .
name	name of the plotting viewport.
newpage	logical. Should <a href="#">grid.newpage</a> be called before plotting?
pop	logical. Should the viewport created be popped?
...	additional arguments passed to <a href="#">plotViewport</a> .

**Details**

`spine` creates either a spinogram or a spine plot. It can be called via `spine(x, y)` or `spine(y ~ x)` where `y` is interpreted to be the dependent variable (and has to be categorical) and `x` the explanatory variable. `x` can be either categorical (then a spine plot is created) or numerical (then a spinogram is plotted). Additionally, `spine` can also be called with only a single argument which then has to be a 2-way table, interpreted to correspond to `table(x, y)`.

Spine plots are a generalization of stacked bar plots where not the heights but the widths of the bars corresponds to the relative frequencies of `x`. The heights of the bars then correspond to the conditional relative frequencies of `y` in every `x` group. This is a special case of a mosaic plot with specific spacing and shading.

Analogously, spinograms extend stacked histograms. As for the histogram, `x` is first discretized (using [hist](#)) and then for the discretized data a spine plot is created.

**Value**

The table visualized is returned invisibly.

**Author(s)**

Achim Zeileis <Achim.Zeileis@R-project.org>

**References**

Hummel, J. (1996), Linked bar charts: Analysing categorical data graphically. *Computational Statistics*, **11**, 23–33.

Hofmann, H., Theus, M. (2005), *Interactive graphics for visualizing conditional distributions*, Unpublished Manuscript.

**See Also**

[cd\\_plot](#), [mosaic](#), [hist](#)

**Examples**

```
## Arthritis data (dependence on a categorical variable)
data("Arthritis")
(spine(Improved ~ Treatment, data = Arthritis))

## Arthritis data (dependence on a numerical variable)
(spine(Improved ~ Age, data = Arthritis, breaks = 5))
(spine(Improved ~ Age, data = Arthritis, breaks = quantile(Arthritis$Age)))
(spine(Improved ~ Age, data = Arthritis, breaks = "Scott"))

## Space shuttle data (dependence on a numerical variable)
data("SpaceShuttle")
(spine(Fail ~ Temperature, data = SpaceShuttle, breaks = 3))
```

---

strucplot

*Structured Displays of Contingency Tables*

---

**Description**

This modular function visualizes certain aspects of high-dimensional contingency tables in a hierarchical way.

**Usage**

```
strucplot(x, residuals = NULL, expected = NULL,
  condvars = NULL, shade = NULL, type = c("observed", "expected"),
  residuals_type = NULL, df = NULL, split_vertical = NULL,
  spacing = spacing_equal, spacing_args = list(),
  gp = NULL, gp_args = list(),
  labeling = labeling_border, labeling_args = list(),
  core = struc_mosaic, core_args = list(),
  legend = NULL, legend_args = list(),
  main = NULL, sub = NULL, margins = unit(3, "lines"),
```

```

title_margins = NULL, legend_width = NULL,
main_gp = gpar(fontsize = 20), sub_gp = gpar(fontsize = 15),
newpage = TRUE, pop = TRUE, keep_aspect_ratio = NULL, prefix = "", ...)

```

### Arguments

<code>x</code>	a contingency table in array form, with optional category labels specified in the <code>dimnames</code> attribute.
<code>residuals</code>	optionally, an array of residuals of the same dimension as <code>x</code> (see details).
<code>expected</code>	optionally, an array of expected values of the same dimension as <code>x</code> , or alternatively the corresponding independence model specification as used by <a href="#">loglin</a> or <a href="#">loglm</a> (see details).
<code>df</code>	degrees of freedom passed to the shading functions used for inference. Will be calculated (and overwritten if specified) if both <code>expected</code> and <code>residuals</code> are <code>NULL</code> , or if <code>expected</code> is given a formula.
<code>condvars</code>	number of conditioning variables, if any; those are expected to be ordered first in the table. This information is used for computing the expected values, and is also passed to the spacing functions (see <a href="#">spacings</a> ).
<code>shade</code>	logical specifying whether <code>gp</code> should be used or not (see <code>gp</code> ). If <code>TRUE</code> and <code>expected</code> is unspecified, a default model is fitted: if <code>condvars</code> is specified, a corresponding conditional independence model, and else the total independence model.
<code>residuals_type</code>	a character string indicating the type of residuals to be computed when none are supplied. If <code>residuals</code> is <code>NULL</code> , <code>residuals_type</code> 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. If <code>residuals</code> are specified, the value of <code>residuals_type</code> is just passed "as is" to the legend function.
<code>type</code>	a character string indicating whether the observed or the expected values of the table should be visualized.
<code>split_vertical</code>	vector of logicals of length $k$ , where $k$ is the number of margins of <code>x</code> (values are recycled as needed). A <code>TRUE</code> component indicates that the tile(s) of the corresponding dimension should be split vertically, <code>FALSE</code> means horizontal splits. Default is <code>FALSE</code> .
<code>spacing</code>	spacing object, spacing function, or a corresponding generating function (see details and <a href="#">spacings</a> ).
<code>spacing_args</code>	list of arguments for the spacing-generating function, if specified.
<code>gp</code>	object of class "gpar", shading function or a corresponding generating function (see details and <a href="#">shadings</a> ). Components of "gpar" objects are recycled as needed along the last splitting dimension. Ignored if <code>shade = FALSE</code> .
<code>gp_args</code>	list of arguments for the shading-generating function, if specified.
<code>labeling</code>	either a logical, or a labeling function, or a corresponding generating function (see details and <a href="#">labelings</a> . If <code>FALSE</code> or <code>NULL</code> , no labeling is produced.
<code>labeling_args</code>	list of arguments for the labeling-generating function, if specified.

core	either a core function, or a corresponding generating function (see details). Currently, generating functions for mosaic plots ( <code>struc_mosaic</code> ), association plots ( <code>struc_assoc</code> ), and sieve plots ( <code>struc_sieve</code> ) are provided.
core_args	list of arguments for the core-generating function, if specified.
legend	either a legend-generating function, or a legend function (see details and <a href="#">legends</a> ), or a logical. If legend is NULL or TRUE and gp is a function, legend defaults to <a href="#">legend_resbased</a> .
legend_args	list of arguments for the legend-generating function, if specified.
main	either a logical, or a character string used for plotting the main title. If main is a logical and TRUE, the name of the object supplied as x is used.
sub	a character string used for plotting the subtitle. If sub is a logical and TRUE and main is unspecified, the name of the object supplied as x is used.
margins	either an object of class "unit" of length 4, or a numeric vector of length 4. The elements are recycled as needed. The four components specify the top, right, bottom, and left margin of the plot, respectively. When a numeric vector is supplied, the numbers are interpreted as "lines" units. In addition, the unit or numeric vector may have named arguments ('top', 'right', 'bottom', and 'left'), in which case the non-named arguments specify the default values (recycled as needed), overloaded by the named arguments.
title_margins	either an object of class "unit" of length 2, or a numeric vector of length 2. The elements are recycled as needed. The two components specify the top and bottom <i>title</i> margin of the plot, respectively. The default for each <i>specified</i> title are 2 lines (and 0 else), except when a legend is plotted and keep_aspect_ratio is TRUE: in this case, the default values of both margins are set as to align the heights of legend and actual plot. When a numeric vector is supplied, the numbers are interpreted as "lines" units. In addition, the unit or numeric vector may have named arguments ('top' and 'bottom'), in which case the non-named argument specify the default value (recycled as needed), overloaded by the named arguments.
legend_width	An object of class "unit" of length 1 specifying the width of the legend (if any). Default: 5 lines.
pop	logical indicating whether the generated viewport tree should be removed at the end of the drawing or not.
main_gp, sub_gp	object of class "gpar" containing the graphical parameters used for the main (sub) title, if specified.
newpage	logical indicating whether a new page should be created for the plot or not.
keep_aspect_ratio	logical indicating whether the aspect ratio should be fixed or not. If unspecified, the default is TRUE for two-dimensional tables and FALSE otherwise.
prefix	optional character string used as a prefix for the generated viewport and grob names.
...	For convenience, list of arguments passed to the labeling-generating function used.

## Details

This function—usually called by higher-level functions such as `assoc` and `mosaic`—generates conditioning plots of contingency tables. First, it sets up a set of viewports for main- and subtitles, legend, and the actual plot region. Then, residuals are computed as needed from observed and expected frequencies, where the expected frequencies are optionally computed for a specified independence model. Finally, the specified functions for spacing, `gp`, main plot, legend, and labeling are called to produce the plot. The function invisibly returns the "structable" object visualized.

Most elements of the plot, such as the core function, the spacing between the tiles, the shading of the tiles, the labeling, and the legend, are modularized in graphical appearance control ("grapcon") functions and specified as parameters. For each element `foo` (= spacing, labeling, core, or legend), `strucplot` takes two arguments: `foo` and `foo_args`, which can be used to specify the parameters in the following alternative ways:

1. Passing a suitable function to `foo` which subsequently will be called from `strucplot` to compute shadings, labelings, etc.
2. Passing a corresponding *generating* function to `foo`, along with parameters passed to `foo_args`, that generates such a function. Generating functions must inherit from classes "grapcon\_generator" and "`}foo\code{"`.
3. Except for the shading functions (`shading_bar`), passing `foo(foo_args)` to the `foo` argument.
4. For shadings and spacings, passing the final parameter object itself; see the corresponding help pages for more details on the data structures.

If legends are drawn, a 'cinemascope'-like layout is used for the plot to preserve the 1:1 aspect ratio.

If `type = "expected"`, the expected values are passed to the observed argument of the core function, and the observed values to the expected argument.

Although the `gp` argument is typically used for shading, it can be used for arbitrary modifications of the tiles' graphics parameters (e.g., for highlighting particular cells, etc.).

## Value

Invisibly, an object of class "structable" corresponding to the plot.

## Note

The created viewports, as well as the tiles and bullets, are named and thus can conveniently be modified after a plot has been drawn (and `pop = FALSE`).

## Author(s)

David Meyer <David.Meyer@R-project.org>

## References

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. URL <http://www.jstatsoft.org/v17/i03/> and available as `vignette("strucplot")`.

**See Also**

[assoc](#), [mosaic](#), [sieve](#), [struc\\_assoc](#), [struc\\_sieve](#), [struc\\_mosaic](#), [structable](#), [doubledecker](#), [labelings](#), [shadings](#), [legends](#), [spacings](#)

**Examples**

```
data("Titanic")

strucplot(Titanic)
strucplot(Titanic, core = struc_assoc)
strucplot(Titanic, spacing = spacing_increase,
          spacing_args = list(start = 0.5, rate = 1.5))
strucplot(Titanic, spacing = spacing_increase(start = 0.5, rate = 1.5))

## modify a tile's color
strucplot(Titanic, pop = FALSE)
grid.edit("rect:Class=1st,Sex=Male,Age=Adult,Survived=Yes",
          gp = gpar(fill = "red"))
```

---

 structable

*Structured Contingency Tables*


---

**Description**

This function produces a ‘flat’ representation of a high-dimensional contingency table constructed by recursive splits (similar to the construction of mosaic displays).

**Usage**

```
## S3 method for class 'formula'
structable(formula, data,
           direction = NULL, split_vertical = NULL, ..., subset, na.action)
## Default S3 method:
structable(..., direction = NULL, split_vertical = FALSE)
```

**Arguments**

formula	a formula object with possibly both left and right hand sides specifying the column and row variables of the flat table.
data	a data frame, list or environment containing the variables to be cross-tabulated, or an object inheriting from class table.
subset	an optional vector specifying a subset of observations to be used. Ignored if data is a contingency table.
na.action	a function which indicates what should happen when the data contain NAs. Ignored if data is a contingency table

...	R objects which can be interpreted as factors (including character strings), or a list (or data frame) whose components can be so interpreted, or a contingency table object of class "table" or "ftable".
split_vertical	logical vector indicating, for each dimension, whether it should be split vertically or not (default: FALSE). Values are recycled as needed. If the argument is of length 1, the value is alternated for all dimensions. Ignored if direction is provided.
direction	character vector alternatively specifying the splitting direction ("h" for horizontal and "v" for vertical splits). Values are recycled as needed. If the argument is of length 1, the value is alternated for all dimensions.

### Details

This function produces textual representations of mosaic displays, and thus ‘flat’ contingency tables. The formula interface is quite similar to the one of `ftable`, but also accepts the `mosaic`-like formula interface (empty left-hand side). Note that even if the `ftable` interface is used, the `split_vertical` or `direction` argument is needed to specify the *order* of the horizontal and vertical splits. If pretabulated data with a `Freq` column is used, than the left-hand side should be left empty—the `Freq` column will be handled correctly.

"structable" objects can be subset using the `[]` and `[[` operators, using either level indices or names (see examples). The corresponding replacement functions are available as well. In addition, appropriate `aperm`, `cbind`, `rbind`, `length`, `dim`, and `is.na` methods do exist.

### Value

An object of class "structable", inheriting from class "ftable", with the splitting information ("split\_vertical") as additional attribute.

### Author(s)

David Meyer <David.Meyer@R-project.org>

### References

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. URL <http://www.jstatsoft.org/v17/i03/> and available as `vignette("strucplot")`.

### See Also

[strucplot](#), [mosaic](#), [ftable](#)

### Examples

```
structable(Titanic)
structable(Titanic, split_vertical = c(TRUE, TRUE, FALSE, FALSE))
structable(Titanic, direction = c("h", "h", "v", "v"))
structable(Sex + Class ~ Survived + Age, data = Titanic)
```

```

## subsetting of structable objects
(hec <- structable(aperm(HairEyeColor)))

## The "[" operator treats structables as a block-matrix and selects parts of the matrix:
hec[1]
hec[2]
hec[1,c(2,4)]
hec["Male",c("Blue","Green")]

## replacement funcion:
tmp <- hec
(tmp[1,2:3] <- tmp[2,c(1,4)])

## In contrast, the "[[" operator treats structables as two-dimensional
## lists. Indexing conditions on specified levels and thus reduces the dimensionality:

## seek subtables conditioning on levels of the first dimension:
hec[[1]]
hec[[2]]

## Seek subtable from the first two dimensions, given the level "Male"
## of the first variable, and "Brown" from the second
## (the following two commands are equivalent):
hec[["Male"]][["Brown"]]
hec[[c("Male","Brown")]]

## Seeking subtables by conditioning on row and/or column variables:
hec[["Male","Hazel"]]
hec[[c("Male","Brown"),]]
hec[[c("Male","Brown"),"Hazel"]]

## a few other operations
t(hec)
dim(hec)
dimnames(hec)
as.matrix(hec)
length(hec)
cbind(hec[,1],hec[,3])

as.vector(hec) ## computed on the _multiway_ table
as.vector(unclass(hec))

```

---

struc\_assoc

*Core-generating Function for Association Plots*


---

## Description

Core-generating function for strucplot returning a function producing association plots.

**Usage**

```
struc_assoc(compress = TRUE, xlim = NULL, ylim = NULL,
            yspace = unit(0.5, "lines"), xscale = 0.9, gp_axis = gpar(lty = 3))
```

**Arguments**

compress	logical; if FALSE, the space between the rows (columns) are chosen such that the <i>total</i> heights (widths) of the rows (column) are all equal. If TRUE, the space between the rows and columns is fixed and hence the plot is more “compressed”.
xlim	either a $2 \times k$ matrix of doubles, $k$ the number of total columns of the plot, or a recycled vector from which such a matrix will be constructed. The columns of xlim correspond to the columns of the association plot, the rows describe the column ranges (minimums in the first row, maximums in the second row). If xlim is NULL, the ranges are determined from the residuals according to compress (if TRUE: widest range from each column, if FALSE: from the whole association plot matrix).
ylim	either a $2 \times k$ matrix of doubles, $k$ the number of total rows of the plot, or a recycled vector from which such a matrix will be constructed. The columns of ylim correspond to the rows of the association plot, the rows describe the column ranges (minimums in the first row, maximums in the second row). If ylim is NULL, the ranges are determined from the residuals according to compress (if TRUE: widest range from each row, if FALSE: from the whole association plot matrix).
xscale	scale factor resizing the tile’s width, thus adding additional space between the tiles.
yspace	object of class “unit” specifying additional space separating the rows.
gp_axis	object of class “gpar” specifying the visual aspects of the tiles’ baseline.

**Details**

This function is usually called by strucplot (typically when called by assoc) and returns a function used by strucplot to produce association plots.

**Value**

A function with arguments:

residuals	table of residuals.
observed	not used by struc_assoc.
expected	table of expected frequencies.
spacing	object of class “unit” specifying the space between the tiles.
gp	list of gpar objects used for the drawing the tiles.
split_vertical	vector of logicals indicating, for each dimension of the table, the split direction.

**Author(s)**

David Meyer <David.Meyer@R-project.org>

## References

- Cohen, A. (1980), On the graphical display of the significant components in a two-way contingency table. *Communications in Statistics—Theory and Methods*, **A9**, 1025–1041.
- Friendly, M. (1992), Graphical methods for categorical data. *SAS User Group International Conference Proceedings*, **17**, 190–200. <http://datavis.ca/sugi/sugi17.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. URL <http://www.jstatsoft.org/v17/i03/> and available as vignette("strucplot").

## See Also

[assoc](#), [strucplot](#), [structable](#)

## Examples

```
## UCB Admissions
data("UCBAdmissions")
ucb <- aperm(UCBAdmissions)

## association plot for conditional independence
strucplot(ucb, expected = ~ Dept * (Admit + Gender),
  core = struc_assoc(ylim = c(-4, 4)), labeling_args = list(abbreviate = c(Admit = 3)))
```

---

struc\_mosaic

*Core-generating Function for Mosaic Plots*

---

## Description

Core-generating function for strucplot returning a function producing mosaic plots.

## Usage

```
struc_mosaic(zero_size = 0.5, zero_split = FALSE, zero_shade = TRUE,
  zero_gp = gpar(col = 0), panel = NULL)
```

## Arguments

- |            |  |
|------------|--|
| zero_size  | size of the bullets used for zero-entries in the contingency table (if 0, no bullets are drawn).   |
| zero_split | logical controlling whether zero cells should be further split. If FALSE and zero_shade is FALSE, only one bullet is drawn (centered) for unsplit zero cells. If FALSE and zero_shade is TRUE, a bullet for each zero cell is drawn to allow, e.g., residual-based shadings to be effective also for zero cells. |
| zero_shade | logical controlling whether zero bullets should be shaded.   |
| zero_gp    | object of class "gpar" used for zero bullets in case they are <i>not</i> shaded.   |

`panel` Optional function with arguments: `residuals`, `observed`, `expected`, `index`, `gp`, and `name` called by the `struc_mosaic` workhorse for each tile that is drawn in the mosaic. `index` is an integer vector with the tile's coordinates in the contingency table, `gp` a `gpar` object for the tile, and `name` a label to be assigned to the drawn grid object.

### Details

This function is usually called by `strucplot` (typically when called by `mosaic`) and returns a function used by `strucplot` to produce mosaic plots.

### Value

A function with arguments:

`residuals` table of residuals.  
`observed` table of observed values.  
`expected` not used by `struc_mosaic`.  
`spacing` object of class "unit" specifying the space between the tiles.  
`gp` list of `gpar` objects used for the drawing the tiles.  
`split_vertical` vector of logicals indicating, for each dimension of the table, the split direction.

### Author(s)

David Meyer <David.Meyer@R-project.org>

### References

Cohen, A. (1980), On the graphical display of the significant components in a two-way contingency table. *Communications in Statistics—Theory and Methods*, **A9**, 1025–1041.

Friendly, M. (1992), Graphical methods for categorical data. *SAS User Group International Conference Proceedings*, **17**, 190–200. <http://datavis.ca/sugi/sugi17.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. URL <http://www.jstatsoft.org/v17/i03/> and available as `vignette("strucplot")`.

### See Also

[mosaic](#), [strucplot](#), [structable](#)

### Examples

```
## Titanic data
data("Titanic")
## mosaic plot with large zeros
strucplot(Titanic, core = struc_mosaic(zero_size = 1))
```

---

struc\_sieve

*Core-generating Function for Sieve Plots*


---

**Description**

Core-generating function for strucplot returning a function producing sieve plots.

**Usage**

```
struc_sieve(sievetype = c("observed", "expected"), gp_tile = gpar())
```

**Arguments**

sievetype	logical indicating whether rectangles should be filled according to observed or expected frequencies.
gp_tile	object of class "gpar", controlling the appearance of all <i>static</i> elements of the cells (e.g., border and fill color).

**Details**

This function is usually called by `strucplot` (typically when called by `sieve`) and returns a function used by `strucplot` to produce sieve plots.

**Value**

A function with arguments:

residuals	table of residuals.
observed	table of observed values.
expected	not used by struc_sieve.
spacing	object of class "unit" specifying the space between the tiles.
gp	list of gpar objects used for the drawing the tiles.
split_vertical	vector of logicals indicating, for each dimension of the table, the split direction.

**Author(s)**

David Meyer <David.Meyer@R-project.org>

**References**

Riedwyl, H., and Schüpbach, M. (1994), Parquet diagram to plot contingency tables. In F. Faulbaum (ed.), *Softstat '93: Advances in Statistical Software*, 293–299. Gustav Fischer, New York.

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

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. URL <http://www.jstatsoft.org/v17/i03/> and available as `vignette("strucplot")`.

**See Also**

[sieve](#), [strucplot](#), [structable](#)

**Examples**

```
## Titanic data
data("Titanic")
strucplot(Titanic, core = struc_sieve)
```

---

Suicide

*Suicide Rates in Germany*

---

**Description**

Data from Heuer (1979) on suicide rates in West Germany classified by age, sex, and method of suicide.

**Usage**

```
data("Suicide")
```

**Format**

A data frame with 306 observations and 6 variables.

**Freq** frequency of suicides.

**sex** factor indicating sex (male, female).

**method** factor indicating method used.

**age** age (rounded).

**age.group** factor. Age classified into 5 groups.

**method2** factor indicating method used (same as method but some levels are merged).

**Source**

Michael Friendly (2000), Visualizing Categorical Data: <http://euclid.psych.yorku.ca/ftp/sas/vcd/catdata/suicide.sas>

**References**

J. Heuer (1979), *Selbstmord bei Kindern und Jugendlichen*. Ernst Klett Verlag, Stuttgart.

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

**Examples**

```
data("Suicide")
structable(~ sex + method2 + age.group, data = Suicide)
```

---

table2d_summary	<i>Summary of a 2-way Table</i>
-----------------	---------------------------------

---

**Description**

Prints a 2-way contingency table along with percentages, marginal, and conditional distributions.

**Usage**

```
table2d_summary(object, margins = TRUE, percentages = FALSE,  
  conditionals = c("none", "row", "column"), ...)
```

**Arguments**

object	a $r \times c$ -contingency table
margins	if TRUE, marginal distributions are computed.
percentages	if TRUE, relative frequencies are computed.
conditionals	if not "none", the conditional distributions, given the row/column factor, are computed.
...	currently not used.

**Value**

Returns invisibly a  $r \times c \times k$  table,  $k$  depending on the amount of choices (at most 3).

**Author(s)**

David Meyer <David.Meyer@R-project.org>

**See Also**

[mar\\_table](#), [prop.table](#), [independence\\_table](#)

**Examples**

```
data("UCBAdmissions")  
table2d_summary(margin.table(UCBAdmissions, 1:2))
```

---

ternaryplot

*Ternary Diagram*


---

### Description

Visualizes compositional, 3-dimensional data in an equilateral triangle.

### Usage

```
ternaryplot(x, scale = 1, dimnames = NULL,
  dimnames_position = c("corner", "edge", "none"),
  dimnames_color = "black", id = NULL, id_color = "black",
  id_just = c("center", "center"),
  coordinates = FALSE, grid = TRUE, grid_color = "gray",
  labels = c("inside", "outside", "none"),
  labels_color = "darkgray", border = "black", bg = "white",
  pch = 19, cex = 1, prop_size = FALSE, col = "red",
  main = "ternary plot", newpage = TRUE, pop = TRUE, ...)
```

### Arguments

x	a matrix with three columns.
scale	row sums scale to be used.
dimnames	dimension labels (defaults to the column names of x).
dimnames_position, dimnames_color	position and color of dimension labels.
id	optional labels to be plotted below the plot symbols. coordinates and id are mutual exclusive.
id_color	color of these labels.
id_just	character vector of length 1 or 2 indicating the justification of these labels.
coordinates	if TRUE, the coordinates of the points are plotted below them. coordinates and id are mutual exclusive.
grid	if TRUE, a grid is plotted. May optionally be a string indicating the line type (default: "dotted").
grid_color	grid color.
labels, labels_color	position and color of the grid labels.
border	color of the triangle border.
bg	triangle background.
pch	plotting character. Defaults to filled dots.
cex	a numerical value giving the amount by which plotting text and symbols should be scaled relative to the default. Ignored for the symbol size if prop_size is not FALSE.

prop_size	if TRUE, the symbol size is plotted proportional to the row sum of the three variables, i.e., represents the weight of the observation.
col	plotting color.
main	main title.
newpage	if TRUE, the plot will appear on a new graphics page.
pop	logical; if TRUE, all newly generated viewports are popped after plotting.
...	additional graphics parameters (see par)

### Details

A points' coordinates are found by computing the gravity center of mass points using the data entries as weights. Thus, the coordinates of a point  $P(a, b, c)$ ,  $a + b + c = 1$ , are:  $P(b + c/2, c\sqrt{3}/2)$ .

### Author(s)

David Meyer <David.Meyer@R-project.org>

### References

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

### Examples

```
data("Arthritis")

## Build table by crossing Treatment and Sex
tab <- as.table(xtabs(~ I(Sex:Treatment) + Improved, data = Arthritis))

## Mark groups
col <- c("red", "red", "blue", "blue")
pch <- c(1, 19, 1, 19)

## plot
ternaryplot(
  tab,
  col = col,
  pch = pch,
  prop_size = TRUE,
  bg = "lightgray",
  grid_color = "white",
  labels_color = "white",
  main = "Arthritis Treatment Data"
)

## legend
grid_legend(0.8, 0.7, pch, col, rownames(tab), title = "GROUP")

## Titanic
data("Lifeboats")
attach(Lifeboats)
ternaryplot(
```

```

Lifeboats[,4:6],
pch = ifelse(side == "Port", 1, 19),
col = ifelse(side == "Port", "red", "blue"),
id = ifelse(men / total > 0.1, as.character(boat), NA),
prop_size = 2,
dimnames_position = "edge",
main = "Lifeboats on Titanic"
)
grid_legend(0.8, 0.9, c(1, 19),
  c("red", "blue"), c("Port", "Starboard"),
  title = "SIDE")

## Hitters
data("Hitters")
attach(Hitters)
colors <- c("black", "red", "green", "blue", "red", "black", "blue")
pch <- substr(levels(Positions), 1, 1)
ternaryplot(
  Hitters[,2:4],
  pch = as.character(Positions),
  col = colors[as.numeric(Positions)],
  main = "Baseball Hitters Data"
)
grid_legend(0.8, 0.9, pch, colors, levels(Positions),
  title = "POSITION(S)")

```

---

tile

*Tile Plot*


---

## Description

Plots a tile display.

## Usage

```

## Default S3 method:
tile(x,
  tile_type = c("squaredarea", "area", "height", "width"),
  halign = c("left", "center", "right"),
  valign = c("bottom", "center", "top"),
  split_vertical = NULL,
  shade = FALSE,
  spacing = spacing_equal(unit(1, "lines")),
  set_labels = NULL,
  margins = unit(3, "lines"),
  keep_aspect_ratio = FALSE,
  legend_width = NULL,

```

```

        squared_tiles = TRUE,
        main = NULL, sub = NULL, ...)
## S3 method for class 'formula'
tile(formula, data,
      ..., main = NULL, sub = NULL, subset = NULL, na.action = NULL)

```

### Arguments

<code>x</code>	a contingency table, or an object coercible to one.
<code>formula</code>	a formula specifying the variables used to create a contingency table from data.
<code>data</code>	either a data frame, or an object of class "table" or "ftable".
<code>subset</code>	an optional vector specifying a subset of observations to be used.
<code>na.action</code>	a function which indicates what should happen when the data contain NAs. Ignored if data is a contingency table.
<code>tile_type</code>	character string indicating how the tiles should reflect the table frequencies (see details).
<code>halign, valign</code>	character string specifying the horizontal and vertical alignment of the tiles.
<code>split_vertical</code>	vector of logicals of length $k$ , where $k$ is the number of margins of <code>x</code> (values are recycled as needed). A TRUE component indicates that the tile(s) of the corresponding dimension should be split vertically, FALSE means horizontal splits. Default is FALSE.
<code>spacing</code>	spacing object, spacing function, or corresponding generating function (see <a href="#">strucplot</a> for more information).
<code>set_labels</code>	An optional character vector with named components replacing the so-specified variable names. The component names must exactly match the variable names to be replaced.
<code>shade</code>	logical specifying whether shading should be enabled or not (see <a href="#">strucplot</a> ).
<code>margins</code>	either an object of class "unit" of length 4, or a numeric vector of length 4. The elements are recycled as needed. The four components specify the top, right, bottom, and left margin of the plot, respectively. When a numeric vector is supplied, the numbers are interpreted as "lines" units. In addition, the unit or numeric vector may have named arguments ('top', 'right', 'bottom', and 'left'), in which case the non-named arguments specify the default values (recycled as needed), overloaded by the named arguments.
<code>legend_width</code>	An object of class "unit" of length 1 specifying the width of the legend (if any). Default: 5 lines.
<code>keep_aspect_ratio</code>	logical indicating whether the aspect ratio should be fixed or not. The default is FALSE to enable the creation of squared tiles.
<code>squared_tiles</code>	logical indicating whether white space should be added as needed to rows or columns to obtain squared tiles in case of an unequal number of row and column labels.
<code>main, sub</code>	either a logical, or a character string used for plotting the main (sub) title. If logical and TRUE, the name of the data object is used.
<code>...</code>	Other arguments passed to <a href="#">strucplot</a>

## Details

A tile plot is a matrix of tiles. For each tile, either the "width", "height", "area", or squared area is proportional to the corresponding entry. The first three options allow column-wise, row-wise and overall comparisons, respectively. The last variant allows to compare the tiles both column-wise and row-wise, considering either the width or the height, respectively.

In contrast to other high-level strucplot functions, `tile` also accepts a table with duplicated levels (see examples). In this case, artificial dimnames will be created, and the actual ones are drawn using `set_labels`.

Note that multiway-tables are first "flattened" using `structable`.

## Value

The "structable" visualized is returned invisibly.

## Author(s)

David Meyer <David.Meyer@R-project.org>

## See Also

[assoc](#), [strucplot](#), [mosaic](#), [structable](#),

## Examples

```
data("Titanic")

## default plot
tile(Titanic)
tile(Titanic, type = "expected")
tile(Titanic, shade = TRUE)

## some variations
tile(Titanic, tile_type = "width", squared_tiles = FALSE)
tile(Titanic, tile_type = "height", squared_tiles = FALSE)
tile(Titanic, tile_type = "area", halign = "center", valign = "center")

## repeat levels
tile(Titanic[, , c(1,2,1,2)])
```

---

Trucks

*Truck Accidents Data*

---

## Description

Data from a study in England in two periods from November 1969 to October 1971 and November 1971 to October 1973. A new compulsory safety measure for trucks was introduced in October 1971. Therefore, the question is whether the safety measure had an effect on the number of accidents and on the point of collision on the truck.

**Usage**

```
data("Trucks")
```

**Format**

A data frame with 24 observations on 5 variables.

**Freq** frequency of accidents involving trucks.

**period** factor indicating time period (before, after) 1971-11-01.

**collision** factor indicating whether the collision was in the back or forward (including the front and the sides) of the truck (back, forward).

**parked** factor indicating whether the truck was parked (yes, no).

**light** factor indicating light conditions: day light (daylight), night on an illuminated road (night, illuminate), night on a dark road (night, dark).

**Source**

E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*, Table 6.8.

**References**

E. B. Andersen (1991), *The Statistical Analysis of Categorical Data*. 2nd edition. Springer-Verlag, Berlin.

**Examples**

```
data("Trucks")
tab <- xtabs(Freq ~ period + collision + light + parked, data = Trucks)
loglm(~ (collision + period) * parked * light, data = tab)
doubledecker(collision ~ parked + light + period, data = tab)
cotabplot(tab, panel = cotab_coinddep)
```

---

UKSoccer

*UK Soccer Scores*


---

**Description**

Data from Lee (1997), on the goals scored by Home and Away teams in the Premier Football League, 1995/6 season.

**Usage**

```
data("UKSoccer")
```

**Format**

A 2-dimensional array resulting from cross-tabulating the number of goals scored in 380 games. The variables and their levels are as follows:

No	Name	Levels
1	Home	0, 1, ..., 4
2	Away	0, 1, ..., 4

**Source**

M. Friendly (2000), *Visualizing Categorical Data*, page 27.

**References**

A. J. Lee (1997), Modelling scores in the Premier League: Is Manchester United really the best?, *Chance*, **10**(1), 15–19.

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

**See Also**

[Bundesliga](#)

**Examples**

```
data("UKSoccer")
mosaic(UKSoccer, gp = shading_max, main = "UK Soccer Scores")
```

---

VisualAcuity

*Visual Acuity in Left and Right Eyes*

---

**Description**

Data from Kendall & Stuart (1961) on unaided vision among 3,242 men and 7,477 women, all aged 30-39 and employed in the U.K. Royal Ordnance factories 1943-1946.

**Usage**

```
data("VisualAcuity")
```

**Format**

A data frame with 32 observations and 4 variables.

**Freq** frequency of visual acuity measurements.

**right** visual acuity on right eye.

**left** visual acuity on left eye.

**gender** factor indicating gender of patient.

**Source**

M. Friendly (2000), *Visualizing Categorical Data*: <http://euclid.psych.yorku.ca/ftp/sas/vcd/catdata/vision.sas>

## References

- M. G. Kendall & A. Stuart (1961), *The Advanced Theory of Statistics*, Vol. 2. Griffin, London.  
M. Friendly (2000), *Visualizing Categorical Data*. SAS Institute, Cary, NC.

## Examples

```
data("VisualAcuity")
structable(~ gender + left + right, data = VisualAcuity)
```

---

VonBort

*Von Bortkiewicz Horse Kicks Data*

---

## Description

Data from von Bortkiewicz (1898), given by Andrews & Herzberg (1985), on number of deaths by horse or mule kicks in 14 corps of the Prussian army.

## Usage

```
data("VonBort")
```

## Format

A data frame with 280 observations and 4 variables.

**deaths** number of deaths.

**year** year of the deaths.

**corps** factor indicating the corps.

**fisher** factor indicating whether the corresponding corps was considered by Fisher (1925) or not.

## Source

Michael Friendly (2000), *Visualizing Categorical Data*: <http://euclid.psych.yorku.ca/ftp/sas/vcd/catdata/vonbort.sas>

## References

- D. F. Andrews & A. M. Herzberg (1985), *Data: A Collection of Problems from Many Fields for the Student and Research Worker*. Springer-Verlag, New York, NY.  
R. A. Fisher (1925), *Statistical Methods for Research Workers*. Oliver & Boyd, London.  
L. von Bortkiewicz (1898), *Das Gesetz der kleinen Zahlen*. Teubner, Leipzig.  
M. Friendly (2000), *Visualizing Categorical Data*. SAS Institute, Cary, NC.

## See Also

[HorseKicks](#) for a popular subsample.

**Examples**

```
data("VonBort")
## HorseKicks data
xtabs(~ deaths, data = VonBort, subset = fisher == "yes")
```

---

WeldonDice

*Weldon's Dice Data*


---

**Description**

Data from Pearson (1900) about the frequency of 5s and 6s in throws of 12 dice. Weldon tossed the dice 26,306 times and reported his results in a letter to Francis Galton on 1894-02-02.

**Usage**

```
data("WeldonDice")
```

**Format**

A 1-way table giving the frequency of a 5 or a 6 in 26,306 throws of 12 dice where 10 indicates '10 or more' 5s or 6s. The variable and its levels are

No	Name	Levels
1	n56	0, 1, ..., 10

**Source**

M. Friendly (2000), *Visualizing Categorical Data*, pages 20–21.

**References**

K. Pearson (1900), On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen by random sampling, *Philosophical Magazine*, **50** (5th series), 157–175.

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

**Examples**

```
data("WeldonDice")
gf <- goodfit(WeldonDice, type = "binomial")
summary(gf)
plot(gf)
```

---

 WomenQueue

 Women in Queues
 

---

**Description**

Data from Jinkinson \& Slater (1981) and Hoaglin \& Tukey (1985) reporting the frequency distribution of females in 100 queues of length 10 in a London Underground station.

**Usage**

```
data("WomenQueue")
```

**Format**

A 1-way table giving the number of women in 100 queues of length 10. The variable and its levels are

No	Name	Levels
1	nWomen	0, 1, ..., 10

**Source**

M. Friendly (2000), *Visualizing Categorical Data*, pages 19–20.

**References**

D. C. Hoaglin \& J. W. Tukey (1985), Checking the shape of discrete distributions. In D. C. Hoaglin, F. Mosteller, J. W. Tukey (eds.), *Exploring Data Tables, Trends and Shapes*, chapter 9. John Wiley \& Sons, New York.

R. A. Jinkinson \& M. Slater (1981), Critical discussion of a graphical method for identifying discrete distributions, *The Statistician*, **30**, 239–248.

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

**Examples**

```
data("WomenQueue")
gf <- goodfit(WomenQueue, type = "binomial")
summary(gf)
plot(gf)
```

---

woolf_test	<i>Woolf Test</i>
------------	-------------------

---

**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).

**Usage**

```
woolf_test(x)
```

**Arguments**

x                    A  $2 \times 2 \times k$  table.

**Value**

A list of class "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(s) of the data.
observed	the observed counts.
expected	the expected counts under the null hypothesis.

**References**

Woolf, B. 1955. On estimating the relation between blood group and disease. *Ann. Human Genet.* (London) **19**, 251-253.

**See Also**

[mantelhaen.test](#)

**Examples**

```
data("CoalMiners")
woolf_test(CoalMiners)
```

# Index

- \*Topic **array**
  - co\_table, 25
  - independence\_table, 44
- \*Topic **category**
  - agreementplot, 3
  - assocstats, 9
  - distplot, 27
  - goodfit, 35
  - independence\_table, 44
  - Kappa, 46
  - mar\_table, 57
  - oddsratio, 63
  - Ord\_plot, 65
  - table2d\_summary, 106
- \*Topic **datasets**
  - Arthritis, 5
  - Baseball, 10
  - BrokenMarriage, 12
  - Bundesliga, 12
  - Bundestag2005, 14
  - Butterfly, 15
  - CoalMiners, 18
  - DanishWelfare, 26
  - Employment, 30
  - Federalist, 31
  - Hitters, 40
  - HorseKicks, 42
  - Hospital, 43
  - JobSatisfaction, 44
  - JointSports, 45
  - Lifeboats, 56
  - MSPatients, 61
  - NonResponse, 62
  - OvaryCancer, 67
  - PreSex, 75
  - Punishment, 77
  - RepVict, 78
  - Rochdale, 79
  - Saxony, 81
  - SexualFun, 82
  - SpaceShuttle, 89
  - Suicide, 105
  - Trucks, 111
  - UKSoccer, 112
  - VisualAcuity, 113
  - VonBort, 114
  - WeldonDice, 115
  - WomenQueue, 116
- \*Topic **hplot**
  - agreementplot, 3
  - assoc, 6
  - cd\_plot, 16
  - cotab\_panel, 23
  - cotabplot, 21
  - doubledecker, 28
  - fourfold, 33
  - grid\_barplot, 38
  - grid\_legend, 39
  - hls, 41
  - labeling\_border, 48
  - labeling\_cells\_list, 52
  - legends, 54
  - mosaic, 58
  - Pairs plot panel functions for diagonal cells, 68
  - Pairs plot panel functions for off-diagonal cells, 70
  - pairs.table, 72
  - plot.loglm, 74
  - rootogram, 80
  - shadings, 83
  - sieve, 87
  - spacings, 91
  - spine, 92
  - struc\_assoc, 100
  - struc\_mosaic, 102
  - struc\_sieve, 104
  - strucplot, 94

- structable, 98
- ternaryplot, 107
- tile, 109
- \*Topic **htest**
  - coindep\_test, 19
  - woolf\_test, 117
- [, 25
- agreementplot, 3, 47
- aperm, 99
- aperm.structable (structable), 98
- Arthritis, 5
- as.matrix.structable (structable), 98
- as.table.structable (structable), 98
- as.vector.structable (structable), 98
- assoc, 6, 24, 35, 60, 71, 74, 75, 86, 89, 97, 98, 102, 111
- assoc.loglm (plot.loglm), 74
- assocplot, 8
- assocstats, 9
- barplot, 38
- Baseball, 10, 40
- BrokenMarriage, 12
- Bundesliga, 12, 113
- Bundestag2005, 14
- Butterfly, 15
- cbind, 99
- cbind.structable (structable), 98
- cd\_plot, 16, 94
- chisq.test, 20
- co\_table, 23, 24, 25
- CoalMiners, 18
- coindep\_test, 19, 23, 24, 85
- confint, 47, 64
- confint.Kappa (Kappa), 46
- confint.oddsratio (oddsratio), 63
- coplot, 22
- cotab\_assoc (cotab\_panel), 23
- cotab\_coindep, 23
- cotab\_coindep (cotab\_panel), 23
- cotab\_fourfold (cotab\_panel), 23
- cotab\_mosaic, 22, 23
- cotab\_mosaic (cotab\_panel), 23
- cotab\_panel, 23
- cotab\_sieve (cotab\_panel), 23
- cotabplot, 21, 24
- DanishWelfare, 26
- density, 17
- dim, 99
- dim.structable (structable), 98
- dimnames.structable (structable), 98
- distplot, 27
- diverge\_hcl, 85, 86
- doubledecker, 28, 60, 89, 91, 92, 98
- Employment, 30
- Extract.structable (structable), 98
- Federalist, 31
- fisher.test, 20
- fitted.coindep\_test (coindep\_test), 19
- fitted.goodfit (goodfit), 35
- fourfold, 33
- ftable, 99
- goodfit, 35
- gray.colors, 17, 93
- grid.newpage, 17, 27, 38, 66, 80, 93
- grid.points, 27, 66
- grid.text, 51, 54
- grid\_barplot, 38, 80, 81
- grid\_legend, 39
- hcl2hex, 41, 85
- hcl2hex (shadings), 83
- hex, 85, 86
- hist, 93, 94
- Hitters, 11, 40
- hls, 41
- HorseKicks, 42, 114
- Hospital, 43
- hsv, 41, 85, 86
- independence\_table, 44, 106
- is.na, 99
- is.na.structable (structable), 98
- is.structable (structable), 98
- JobSatisfaction, 44
- JointSports, 45
- Kappa, 46
- labeling\_border, 48, 53, 54
- labeling\_cboxed (labeling\_border), 48
- labeling\_cells, 50, 51
- labeling\_cells (labeling\_cells\_list), 52

- labeling\_cells\_list, [52](#)
- labeling\_conditional (labeling\_border), [48](#)
- labeling\_doubledecker (labeling\_border), [48](#)
- labeling\_lboxed (labeling\_border), [48](#)
- labeling\_left (labeling\_border), [48](#)
- labeling\_left2 (labeling\_border), [48](#)
- labeling\_list, [50](#), [51](#)
- labeling\_list (labeling\_cells\_list), [52](#)
- labeling\_residuals (labeling\_border), [48](#)
- labeling\_values (labeling\_border), [48](#)
- labelings, [95](#), [98](#)
- labelings (labeling\_border), [48](#)
- legend, [39](#)
- legend\_fixed (legends), [54](#)
- legend\_resbased, [88](#), [96](#)
- legend\_resbased (legends), [54](#)
- legends, [54](#), [88](#), [96](#), [98](#)
- length, [99](#)
- length.structable (structable), [98](#)
- Lifeboats, [56](#)
- loglin, [59](#), [95](#)
- loglm, [59](#), [74](#), [75](#), [95](#)
  
- mantelhaen.test, [117](#)
- mar\_table, [57](#), [106](#)
- mosaic, [9](#), [24](#), [30](#), [35](#), [58](#), [71](#), [74](#), [75](#), [86](#), [89](#), [94](#), [97–99](#), [103](#), [111](#)
- mosaic.loglm (plot.loglm), [74](#)
- mosaicplot, [55](#), [59](#), [60](#)
- MSPatients, [61](#)
  
- NonResponse, [62](#)
  
- oddsratio, [63](#)
- Ord\_estimate (Ord\_plot), [65](#)
- Ord\_plot, [65](#)
- OvaryCancer, [67](#)
  
- pairs, [73](#)
- Pairs plot panel functions for diagonal cells, [68](#)
- Pairs plot panel functions for off-diagonal cells, [70](#)
- pairs.structable (pairs.table), [72](#)
- pairs.table, [68](#), [70](#), [71](#), [72](#)
- pairs\_assoc, [70](#), [71](#), [74](#)
- pairs\_assoc (Pairs plot panel functions for off-diagonal cells), [70](#)
- pairs\_barplot, [70](#), [71](#), [74](#)
- pairs\_barplot (Pairs plot panel functions for diagonal cells), [68](#)
- pairs\_diagonal\_mosaic (Pairs plot panel functions for diagonal cells), [68](#)
- pairs\_diagonal\_text, [74](#)
- pairs\_diagonal\_text (Pairs plot panel functions for diagonal cells), [68](#)
- pairs\_mosaic, [70](#), [71](#), [74](#)
- pairs\_mosaic (Pairs plot panel functions for off-diagonal cells), [70](#)
- pairs\_sieve, [71](#), [74](#)
- pairs\_sieve (Pairs plot panel functions for off-diagonal cells), [70](#)
- pairs\_strucplot (Pairs plot panel functions for off-diagonal cells), [70](#)
- pairs\_text, [70](#), [71](#), [74](#)
- pairs\_text (Pairs plot panel functions for diagonal cells), [68](#)
- par, [4](#), [64](#)
- plot.goodfit (goodfit), [35](#)
- plot.loglm, [74](#)
- plot.oddsratio (oddsratio), [63](#)
- plotViewport, [17](#), [93](#)
- polarLUV, [41](#), [85](#), [86](#)
- POSIXt, [56](#)
- predict.goodfit (goodfit), [35](#)
- PreSex, [75](#)
- print.assocstats (assocstats), [9](#)
- print.goodfit (goodfit), [35](#)
- print.Kappa (Kappa), [46](#)
- print.oddsratio (oddsratio), [63](#)
- print.summary.assocstats (assocstats), [9](#)
- print.summary.Kappa (Kappa), [46](#)
- print.summary.oddsratio (oddsratio), [63](#)
- print.table2d\_summary (table2d\_summary), [106](#)
- prop.table, [106](#)
- Punishment, [77](#)
  
- r2dtable, [20](#)
- rbind, [99](#)
- rbind.structable (structable), [98](#)
- RepVict, [78](#)

- Rochdale, [79](#)
- rootogram, [36](#), [80](#)
  
- Saxony, [81](#)
- SexualFun, [82](#)
- shading\_binary (shadings), [83](#)
- shading\_Friendly, [87](#)
- shading\_Friendly (shadings), [83](#)
- shading\_hcl, [24](#)
- shading\_hcl (shadings), [83](#)
- shading\_hsv (shadings), [83](#)
- shading\_max (shadings), [83](#)
- shading\_sieve (shadings), [83](#)
- shadings, [24](#), [55](#), [59](#), [74](#), [83](#), [87](#), [95](#), [98](#)
- sieve, [24](#), [71](#), [74](#), [87](#), [98](#), [104](#), [105](#)
- SpaceShuttle, [89](#)
- spacing\_conditional, [7](#)
- spacing\_conditional (spacings), [91](#)
- spacing\_dimequal (spacings), [91](#)
- spacing\_equal (spacings), [91](#)
- spacing\_highlighting (spacings), [91](#)
- spacing\_increase (spacings), [91](#)
- spacings, [91](#), [95](#), [98](#)
- spine, [17](#), [92](#)
- struc\_assoc, [96](#), [98](#), [100](#)
- struc\_mosaic, [96](#), [98](#), [102](#)
- struc\_sieve, [96](#), [98](#), [104](#)
- strucplot, [7–9](#), [29](#), [30](#), [50](#), [53](#), [59](#), [60](#), [71](#), [75](#),  
[86–89](#), [91](#), [92](#), [94](#), [99](#), [102–105](#), [110](#),  
[111](#)
- structable, [9](#), [51](#), [54](#), [55](#), [60](#), [89](#), [98](#), [98](#), [102](#),  
[103](#), [105](#), [111](#)
- Suicide, [105](#)
- summary.assocstats (assocstats), [9](#)
- summary.goodfit (goodfit), [35](#)
- summary.Kappa (Kappa), [46](#)
- summary.oddsratio (oddsratio), [63](#)
  
- t.structable (structable), [98](#)
- table2d\_summary, [106](#)
- ternaryplot, [107](#)
- tile, [109](#)
- Trucks, [111](#)
  
- UKSoccer, [13](#), [112](#)
  
- VisualAcuity, [113](#)
- VonBort, [42](#), [114](#)
  
- WeldonDice, [115](#)
  
- WomenQueue, [116](#)
- woolf\_test, [117](#)
  
- xtabs, [4](#)