

Package ‘extracat’

May 15, 2012

Type Package

Title Graphic Extensions for Categorical Data

Version 1.5-0

Date 2012-05-14

Author Alexander Pilhoefer <alexander.pilhoefer@math.uni-augsburg.de>

Maintainer Alexander Pilhoefer <alexander.pilhoefer@math.uni-augsburg.de>

Description graphic extensions for categorical data

Depends R (>= 2.14.0), grid, MASS, colorspace, hexbin, scales,ggplot2, reshape, plyr

Suggests JGR, iplots, vcd, ca, amap, rgl

LazyLoad yes

LazyData yes

License GPL (>= 2)

Repository CRAN

Date/Publication 2012-05-15 10:09:09

R topics documented:

arsim	2
boxplot2g	3
carcustomers	4
cfuctile	6
classcrit	8
cpcp	10
CPScluster	12
dcor	13
dmc	14
extracat	15

fluctile	16
fluctile3d	18
hexpie	19
innerval	21
kendalls	23
listen	23
olives	24
optile	25
plants	31
quickfechner	34
regmax	35
resort	36
rmb	37
rmbmat	42
spread	44
subtable	45
subtree	46
tfluctile	47
untableSet	49

Index **50**

arsim *block-structured arrays*

Description

Generates an array or matrix that includes k fully separated block-clusters.

Usage

arsim(n, dim, k, noise = 0, shuffle = TRUE, v = 0.1, minc = 1, exp.prop = NULL, min.prop = 1/dim/4)

Arguments

n	The number of observations in the array.
dim	The dimension of the array.
k	The number of clusters. 1 for no clusters.
noise	The proportion of the n observations which is white noise.
shuffle	Whether or not to disrupt the block structure by randomly mixing up the category orders.
v	The variability with which the observations are assigned to the clusters.
minc	The minimum number of categories each cluster must have in each variable. E.g. minc = 2 means, that each cluster is at least 2 categories thick in all dimensions.
exp.prop	expected proportions of observations in the clusters.
min.prop	minimum proportion of observations in each cluster.

Details

Not a very sophisticated way of generating random arrays but it serves for tests and illustrations of the other functions.

Value

A simulated data array.

Examples

```
A <- arsim(1000, c(12,12), 3, shuffle = FALSE)
fluctile(A)

## Not run:
A2<- arsim(1000, c(12,12,12), 3, shuffle = FALSE)
fluctile3d(A2, shape ="oct")

## End(Not run)
```

 boxplot2g

2D projection boxplots

Description

Contours derived from the boxplot marks of data projections.

Usage

```
boxplot2g(x, y = NULL, groups = NULL, smooth = loess,
smooth.args = list(span = 0.1), colv = NULL, alpha = 1, n = 360, ...)
```

Arguments

x	Numeric x variable.
y	Numeric y variable.
groups	A grouping variable or NULL.
smooth	A smoothing function such as loess or FALSE.
smooth.args	Additional arguments for the smoother, such as span.
colv	A color vector with one color for each group. Not implemented.
alpha	The alpha-blending value for the points of the scatterplot.
n	The number of irection vectors. Defaults to 360 which means 1-degree steps.
...	Further args.

Details

The data is first centered and standardized using the inverse of the covariance matrix. Then projections of the data onto different direction vectors (0 to 360 degree) are computed. The boxplot marks (median, box, whisker) are computed for each such projection and finally retransformed again using the covariance matrix. Finally the projected marks are possibly smoothed, for instance using loess.

Value

The ggplot object.

Note

Please note that I do not claim that the resulting contours have any statistically useful characteristics! Usage at one's own risk!

Author(s)

Alexander Pilhoefer

See Also

[contour](#)

Examples

```
data(olives)
boxplot2g(olives$palmitoleic,olives$oleic,olives$Region)

boxplot2g(olives$palmitoleic,olives$oleic,olives$Area,
alpha=0.5,smooth.args=list(span=0.3))
```

carcustomers

The car customers dataset from 1983

Description

This dataset is taken from the website of the Department of Statistics, University of Munich. *The data are based upon a poll from a german car-company. In 1983 questionnaires were sent to 2000 customers, who had purchased a new car approximately three months earlier. The point of interest was the degree of satisfaction, reasons for the particular choice, consumer profile, etc. Participation was of course voluntary. Only 1182 persons answered the questions and after removing forms with "missing values" only 793 questionnaires remained. Each form contained 46 questions, which resulted in a dataset of 46 covariates with 793 observations each. Due to the abundance of ordinal and categorical covariates the dataset is particularly suited for generalized linear models.*

Usage

```
data(carcustomers)
```

Format

A data frame with 774 observations on the following 47 variables.

model a factor with levels A B C D

gear a factor with levels 4-gear 5-gear (overdrive) 5-gear (sport) Automatic

lease a factor with levels bought leased

usage a factor with levels business private private and business

premod a factor with levels Audi BMW 3er BMW 5er BMW 7er Ford Mercedes Benz Opel other origin
Volkswagen

other a factor with levels No Yes, both Yes, other manufact Yes, same manufact.

testdrv influence on buying decision: testdrive

promotion influence on buying decision: promotion

exp influence on buying decision: experience

recom influence on buying decision: recommendation

clear influence on buying decision: clearness

eco influence on buying decision: economical aspects

drvchar influence on buying decision: driving character

service influence on buying decision: service

interior influence on buying decision: interior

quality influence on buying decision: overall quality

tech influence on buying decision: technical aspects

evo influence on buying decision: evolution

comfort influence on buying decision: comfort

reliab influence on buying decision: reliability

handling influence on buying decision: handling

prestige influence on buying decision: prestige

concept influence on buying decision: overall concept

char influence on buying decision: character

power influence on buying decision: engine power

valdecr influence on buying decision:value decrease

styling influence on buying decision: styling

safety influence on buying decision:safety

sport influence on buying decision: sportive

fcons influence on buying decision: fuel consumption

space influence on buying decision: available space

sat overall satisfaction with the car: 1(very satisfied) to 5(not satisfied)

adv1 satisfaction with concept and styling: a factor with levels does not suit neither nor
suits

adv2 satisfaction with body/bare essentials: a factor with levels does not suit neither nor suits

adv3 satisfaction with chassis/drive/gearshift: a factor with levels does not suit neither nor suits

adv4 satisfaction with engine/power: a factor with levels does not suit neither nor suits

adv5 satisfaction with electronics: a factor with levels does not suit neither nor suits

adv6 satisfaction with financial aspects: a factor with levels does not suit neither nor suits

adv7 asatisfaction with equipment: a factor with levels does not suit neither nor suits

spoco balance variables: a factor with levels comfort could be better handling could be better well balanced

faver usual driving style: a factor with levels economical extreme normal powerful

sspeed usual speed (Autobahn): a factor with levels >110 mph 60-80 mph 81-9g mph 96-110 mph

sfcons satisfaction with fuel consumption: a factor with levels Appropriate Definitely too high Just okay Pleasingly low

sex customer's gender: a factor with levels Female Male

prof customer's profession: a factor with levels Employee/Workman Free lanced Self employed

family customers's family type: a factor with levels >3 persons 1-2 persons

Freq the weighting variable

Source

http://www.stat.uni-muenchen.de/service/datenarchiv/auto/auto_e.html

Examples

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

cfluctile

cluster fluctuation for 2-way tables after optile

Description

Plots a fluctuation diagram of a 2-way table and adds red rectangles for clusters. Uses Kendalls tau for the cluster identification.

Usage

```
cfluctile(x, tau0 = NULL, method = "Kendall", col = "red", lwd =
  2, lty = 1, gap.prop = 0.2, floor = 0, rev.y = FALSE,
  add = FALSE, shape = "r", just = "c", dir = "b", ...)
```

Arguments

x	A 2-way table
tau0	The minimum value of Kendalls tau or Cohens Kappa as a stop criterion for each split.
method	Either "Kendall" for Kendalls Tau or "Cohen" for Cohens Kappa.
col	color of the rectangles which highlight the clusters
lwd	The line width.
lty	The line type.
gap.prop	proportion of the gaps between rows/columns.
floor	floor censored zooming: all cases will be plotted but only those with a frequency of at least floor will be considered for the clustering.
rev.y	revert the y axis.
add	Whether to make a new plot or to add to an existing one.
shape	The shape of the objects. See fluctile .
just	See fluctile .
dir	See fluctile .
...	dots

Details

This function calls [fluctile](#) to create a 2-way fluctuation diagram and then adds cluster rectangles to it. The cluster rectangles are computed in the following way:

First Kendalls tau is computed for the whole table if no tau0 value is defined.

Initially all columns and all rows are seen as one single cluster (1x1 table).

Then the columns as well as the rows are split into 2 groups each such that Kendalls tau is maximal for the resulting table (2x2).

This process is then repeated within the resulting diagonal blocks as long as the tau values are better than tau0.

Value

invisible(TRUE)

Note

This was part of the Google Summer of Code 2011.

Author(s)

Alexander Pilhoefer
Department for Computer Oriented Statistics and Data Analysis
University of Augsburg
Germany

See Also[mosaic](#)**Examples**

```
M <- arsim(1000,c(10,10),3)
M2 <- optile(M)
cfluctile(M2)

M2 <- optile(M, iter = 100)
cfluctile(M2)

M2 <- optile(M, iter = 100, presort = TRUE)
cfluctile(M2)

## Not run:
M2 <- optile(M, fun = "hamm", iter = 20)
cfluctile(M2)

## End(Not run)

M2 <- optile(M, fun = "casort")
cfluctile(M2)

M2 <- optile(M, fun = "csvd")
cfluctile(M2)

M2 <- optile(M, fun = "preclass", foreign = ".Call")
cfluctile(M2)

M <- arsim(1000,c(10,10),3)
MX <- optile(M, iter = 100)
MX <- ceiling(M/3) + MX
cfluctile(MX)
```

classcrit

The classification criterion

Description

These functions compute the classification criterion, the weighted classification criterion and their scaled versions.

Usage

```
classcrit(x, concordant = TRUE)
hammcrit(x)
ihcrit(x)
```

```
iccrit(x)
shcrit(x)
sccrit(x, concordant = FALSE)
```

Arguments

x A data matrix, table or array. The hamming criterion is currently only implemented for two-dimensional tables and arrays.

concordant Whether to return the number of concordant or discordant observation pairs.

Details

`classcrit` and `hammcrit` compute the criteria itself. `iccrit` and `ihcrit` compute the worst possible value after optimization for the given marginal distributions. `sccrit` and `shcrit` are equivalent to `classcrit(x, concordant = FALSE)/iccrit(x)` and `hammcrit(x)/ihcrit(x)` respectively.

Value

The criterion value.

Author(s)

Alexander Pilhofer

See Also

[kendalls](#)

Examples

```
M <- arsim(1000, c(12,12), 3)
classcrit(M)
hammcrit(M)
shcrit(M)
sccrit(M)

M2 <- optile(M, iter = 100)

classcrit(M2)
hammcrit(M2)
shcrit(M2)
sccrit(M2)
```

Description

This function provides a parallel coordinates plot for categorical as well as continuous data based on the `ipcp` function in the `iplots` package. It applies sorted numeric point sequences to the categories which indicate the relative frequencies and allow a sensible interactive highlighting. There are options to change the rule for the gaps between these sequences and to apply an additional ordering algorithm.

Usage

```
cpcp(V,ord = NULL, freqvar = NULL,numerics = NULL, gap.type = "equal.tot",
na.rule = "omit", spread=0.3,gap.space = 0.2, sort.individual=FALSE,
jitter = FALSE, plot = TRUE, return.df = !plot,...)
```

Arguments

<code>V</code>	data frame containing the variables for the plot. A special choice and order can be made by setting the second argument <code>ord</code> . If <code>V</code> is a frequency table (see e.g. fable) the frequency variable has to be named "Freq".
<code>freqvar</code>	Optional name of a frequency variable in <code>V</code> . If <code>V</code> contains a variable called "Freq" (see fable) it will be defined as frequency variable if <code>freqvar</code> is unspecified.
<code>numerics</code>	An integer vector specifying the indices of numeric variables. Variables of class "numeric" are indicated automatically but integer variables are not.
<code>ord</code>	An integer vector specifying the indices and order of the variables.
<code>gap.type</code>	The rule for the gaps between different categories: "equal.gaps" will lead to equal gap sizes in the whole plot (and thus depends on the maximum number of categories). "equal.tot" will force the gaps within each variable to sum up to <code>gap.space</code> . "spread" will lead to equal distances between the centerpoints of each category. In the latter case the spread value denotes the maximal spread width around each centerpoint.
<code>na.rule</code>	"omit" will omit all incomplete cases from <code>V</code> . Otherwise all NAs will be transformed to categories called "n/a".
<code>spread</code>	The maximum width of a single category if <code>gap.type</code> is set to "spread". Should lie between 0 and 1.
<code>gap.space</code>	The space parameter for each gap. Should lie in $[0, 1)$.
<code>sort.individual</code>	If TRUE all polygons with the same categories in two neighboring variables will be drawn together in the right variable.
<code>jitter</code>	logical: whether or not to jitter integer variables which are treated as a numeric variable.

plot	Whether or not to draw the plot. If FALSE only the iset is computed.
return.df	A logical indicating whether or not to return the iset.
...	Further arguments.

Value

invisible(TRUE) or, if return.df == TRUE, the iset which was computed.

Note

This function is based on the **iplots** package which will be installed automatically on the first run. For unix systems it is also necessary to run it from the **JGR** console. To receive **JGR** please visit <http://www.rosuda.org/software> and download the JGR launcher which will start the console automatically and install all other required packages on its first run.

Author(s)

Alexander Pilhoefer
Department for Computer Oriented Statistics and Data Analysis
University of Augsburg
Germany

References

Alexander Pilhoefer *New approaches in visualization of categorical data: R-package extracat*
Journal of Statistical Software, submitted March 2010

See Also

[ipcp listen resort](#)

Examples

```
## Not run:  
cpcp(V = housing)  
  
A <- arsim(1000,c(4,8,12,16),4, noise = 0.05)  
cpcp(A, sort.individual = TRUE)  
  
A2 <- optile(A)  
cpcp(A2, sort.individual = TRUE)  
  
## End(Not run)
```

 CPScluster

CPS subset with clusterings

Description

Different hierarchical clusterings and k-means clusterings as well as a model-based clustering have been applied to several financial variables for a random sample of ten thousand observations.

Usage

```
data(CPScluster)
```

Format

A data frame with 10000 observations on the following 39 variables.

Age a numeric vector

Sex a factor with levels female male

Race a factor with levels Black White

Ethnic a factor with levels Central and South American Chicano Cuban Mexican American
Mexicano Other Spanish Others Puerto Rican don t know

Marital.Status a factor with levels Divorced Married Never married Separated Widowed

Kind.of.Family a factor with levels Husband-Wife family Other female head family Other male head family

Classical a factor with levels All other Classical Husband-Wife family

Family.Type a factor with levels Nonfamily householder Primary family Related subfamily
Unrelated individual Unrelated subfamily

Number.of.Persons.in.Family a numeric vector

Number.of.Kids a numeric vector

Education.of.Head a factor with levels 00. Children 01. Less than 1st grade 02. 1st, 2nd, 3rd or 4th grade
03. 5th or 6th grade 04. 7th or 8th grade 05. 9th grade 06. 10th grade
07. 11th grade 08. 12th grade, no diploma 09. High school graduate, high school diploma or equivalent
10. Some college but no degree 11. Ass. college deg.:occup./voc. program
12. Ass. college deg.:academic program 13. Bachelors degree 14. Masters degree
15. Professional school degree 16. Doctoral degree

Labor.Status a factor with levels Armed forces Children Disabled Employed Employed, not at work
Not in labor force, other Retired Unemployed, looking for work Unemployed, on layoff

Class.of.Worker a factor with levels Federal government Local government Not in labor force
Private sector Self-empl.: incorporated Self-empl.: unincorporated State government
Unemployed, no previous experience Without pay

Working.Hours a numeric vector

Income.of.Head a numeric vector

Family.Income a numeric vector

Taxable.Income a numeric vector
 Federal.tax a numeric vector
 Family.sequence.number a numeric vector
 State a factor with levels Alabama Alaska Arizona Arkansas California Colorado Connecticut
 Delaware District of Columbia Florida Georgia Hawaii Idaho Illinois Indiana
 Iowa Kansas Kentucky Louisiana Maine Maryland Massachusetts Michigan Minnesota
 Mississippi Missouri Montana Nebraska Nevada New Hampshire New Jersey New Mexico
 New York North Carolina North Dakota Ohio Oklahoma Oregon Pennsylvania Rhode Island
 South Carolina South Dakota Tennessee Texas Utah Vermont Virginia Washington
 West Virginia Wisconsin Wyoming
 Division a factor with levels East North Central East South Central Middle Atlantic
 Mountain New England Pacific South Atlantic West North Central West South Central
 Region a factor with levels Midwest North East South West
 hc4 a numeric vector
 hc6 a numeric vector
 hc8 a numeric vector
 hc12 a numeric vector
 hcs4 a numeric vector
 hcs6 a numeric vector
 hcs8 a numeric vector
 hcs12 a numeric vector
 hcw4 a numeric vector
 hcw6 a numeric vector
 hcw8 a numeric vector
 hcw12 a numeric vector
 km4 a numeric vector
 km6 a numeric vector
 km8 a numeric vector
 km12 a numeric vector
 mc12 a numeric vector

 dcor

Distance correlation for tables

Description

This computes the distance correlation for two categorical variables.

Usage

dcor(x)

Arguments

x A data crosstable.

Value

The correlation value which is between 0 and 1.

Author(s)

Alexander Pilhoefer

References

Szekely, G. J. Rizzo, M. L. and Bakirov, N. K. (2007). "Measuring and testing independence by correlation of distances", *Annals of Statistics*, 35/6, 2769-2794

Examples

```
require(MASS)
p1 <- (p1<-runif(8, 0.5,1))/sum(p1)
p2 <- (p2<-runif(12,0.5,1))/sum(p2)

rmat <- matrix(rmultinom(1,1000,prob = as.vector(outer(p1,p2))), ncol=12)

dcor(rmat)
rmat2 <- optile(rmat)
dcor(rmat2 <- optile(rmat, fun = "distcor"))

fluctile(rmat)

bmat <- arsim(1000, c(12,12), 3)

dcor(bmat)
dcor(bmat2 <- optile(bmat, fun = "distcor"))

fluctile(bmat)
cfluctile(bmat2)
```

dmc

dmc 2009 insurance variables

Description

five insurance variables from the dmc 2009 dataset, which have a ordinal structure which has been lost somehow. Can we find it again?

Usage

```
data(dmc)
```

Format

A data frame with 693 observations on the following 6 variables.

eiw_scr a factor with levels 5 6 4 3 2 1
eih_scr a factor with levels 6 3 5 1 4 2
ifi_scr a factor with levels 4 3 5 2 1 6
tec_scr a factor with levels 5 1 3 2 4 6
klv_scr a factor with levels 2 5 6 1 3 4
Freq a numeric vector

Details

The Data Mining Cup (dmc) is a competition for students.

Examples

```
data(dmc)
```

extracat

Graphical extensions for catagorical data

Description

Provides interactive parallel coordinates plots based on package `iplots` for categorical and continuous data as well as a grid-based special version of multiple barcharts, `rmb` which shows the relative frequencies of a target variable an the corresponding weights in a MB-like graphical layout. Functions related to the Google Summer of Code 2011 are also included: `optile` resorts the categories of variables in order to sort of diagonalize the tables and graphical representations, which makes them easier to interpret. `fluctile` provides fluctuation diagrams for the visualisation.

Details

Package: extracat
Type: Package
Version: 1.5-0
Date: 2012-04-25
License: -
LazyLoad: yes

Author(s)

Alexander Pilhoefer
 Department for Computer Oriented Statistics and Data Analysis
 University of Augsburg
 Germany

Maintainer: Alexander Pilhoefer <alexander.pilhoefer@math.uni-augsburg.de>

References

Alexander Pilhoefer *Analysen von kategoriellen Daten*
 Diploma Thesis 2009

Alexander Pilhoefer *New approaches in visualization of categorical data: R-package extracat*
 Journal of Statistical Software, accepted 2011

 fluctile

fluctuation diagrams

Description

Create a fluctuation diagram from a multidimensional table.

Usage

```
fluctile(tab, dir = "b", just = "c", hsplit = FALSE, shape = "r", gap.prop = 0.1, border = NULL, label = T
```

Arguments

tab	The table which is to be plotted.
dir	The bar/rectangle direction: "v" and "h" stand for vertical or horizontal bars. "b" stands for "both" and leads to standard fluctuation diagrams with quadratic rectangles. Use "n" for a same-binsize-plot
just	A shortcut version of the argument used in grid for the anchorpoint of the rectangles: "rb" is equivalent to c("right", "bottom"), "t" is equivalent to "ct" or c("centre", "top") and so on. See examples.
hsplit	A logical for alternating columns and rows or a vector of logicals with TRUE for each variable on the x-axis.
shape	Instead of rectangles ("r") it is possible to use circles ("c"), diamonds ("d") or octagons ("o"). The arguments dir and just work for rectangular shapes only.
gap.prop	proportion of the gaps between the rows/columns within each block.
abbrev	abbreviate the labels to a given number of characters.
border	The proportion of the space used for the labels.
label	Whether or not to plot labels.
lab.opt	A list with options for the labels. Currently lab.cex and abbrev work.

add	Whether to create a new plot or add it to an existing one.
tile.col	The color of the tiles.
bg.col	The background color in each cell.
...	dots

Value

The viewport tree behind the graphic.

Note

This was part of the Google Summer of Code 2011.

Author(s)

Alexander Pilhofer
Department for Computer Oriented Statistics and Data Analysis
University of Augsburg
Germany

See Also

[mosaic](#)

Examples

```
M <- arsim(1000, c(12,12), 3)
fluctile(M)

M2 <- optile(M)

# the standard fluctuation diagram with centralized rectangles
fluctile(M2)

# the standard fluctuation diagram with centralized octagons
fluctile(M2, shape = "o")

#another option such as it is used in iplots or MONDRIAN
# is to plot the rectangles in the bottom left corner
fluctile(M2, just = "lb")

# a multiple barchart
fluctile(M2, just = "b", dir = "h")

# or with vertical bars
fluctile(M2, just = "l", dir = "v")

# a same-binsize-plot
fluctile(M2, dir = "n")

fluctile(xtabs(Freq~Type+Infl+Cont+Sat,data=housing),dir="h",just="b",
```

```
lab.opt=list(lab.cex=1))  
  
A <- arsim(2000, c(6,6,4,4), 3, shuffle = FALSE, noise = 0.05)  
fluctile(A)
```

fluctile3d *3d fluctuation diagram*

Description

Uses rgl to create a 3-dimensional fluctuation diagram.

Usage

```
fluctile3d(x, shape = "cube", col = "darkgrey", alpha = 0.8, ...)
```

Arguments

x	A 3-dimensional data table or array.
shape	Either "cube" for cubes or "oct" for octahedrons.
col	The color of the cubes.
alpha	The alpha-value of the cubes.
...	dots

Value

TRUE

Author(s)

Alexander Pilhoefer

See Also

[fluctile](#)

Examples

```
## Not run:  
A <- arsim(1000,c(5,7,9),3)  
fluctile3d(A)  
  
## End(Not run)
```

Description

This function bins two continuous variables into a hexagonal grid and represents a third variable (which is usually a factor) via piecharts or nested hexagons within the bins. The main idea is to avoid overplotting and unfortunate effects that emerge from mixing up colors, e.g. with alpha-blending.

Usage

```
hexpie(x, y = NULL, z = NULL, n = 24, shape = "hex", p.rule = "radial",
      decr.by.rank = NULL, freq.trans = I, alpha.freq = FALSE, col = "hcl", col.opt = list(),
      show.hex = TRUE, random = NULL, xlim = range(x),
      ylim = range(y), label.opt = list(), vp = NULL)
```

Arguments

x	The variable for the horizontal axis. Should be integer or numeric.
y	The variable for the vertical axis. Should be integer or numeric.
z	The target variable for the colors. Is handled as a factor via <code>as.integer(as.factor(z))</code> .
n	The number of bins into which x is divided. See hexbin .
shape	There are two possibilities: "hex", "hexagonal", and "h" lead to hexagonal representations and "pie", "piechart", "circular" and "c" lead to circular representations.
p.rule	This controls the rules for the representation of the relative frequencies of the target categories. For shape = "hex" this should be one of "rad", "radius", "radial" meaning that the probabilities are represented by the radii. For shape = "circular" it is also possible to create piecharts via "angular", "angles" or "ang".
decr.by.rank	Whether or not to sort the categories within each hexagon individually by their frequencies in decreasing order. Defaults to NULL for no reordering but may be either TRUE (decreasing order) or FALSE (increasing order).
freq.trans	A function which is used to rescale the total counts of the cells. <code>sqrt</code> is a common choice.
alpha.freq	The frequencies may additionally be reflected in terms of the alpha values of the colors.
col	The choice of a color palette. See rmb for further explanations.
col.opt	Additional color options to replace the defaults. See rmb for further explanations.
show.hex	Whether or not to draw the hexagons. Setting <code>col.opt = list(line.col.hex = NA)</code> leaves the lines out and draws the background only.

random	If this is not NULL in each bin a random sample of $n = \text{random}$ observations will be drawn (with replacement) from the corresponding data points. The resulting frequencies are then used to draw the piechart or hexagon. The main idea is to use $\text{random} = 1$ with larger numbers of bins such as $n = 120$ and $\text{show.hex} = \text{FALSE}$.
xlim	A vector of length 2 defining the x-limits e.g. computed via innerval .
ylim	A vector of length 2 defining the y-limits e.g. computed via innerval .
label.opt	Additional labeling options to replace the defaults. Not yet implemented.
vp	A viewport to plot in, e.g. for conditional plots.

Value

invisible(TRUE)

Author(s)

Alexander Pilhoefer

See Also

[stat_binhex](#), [hexbin](#)

Examples

```

data(olives)
x <- olives$oleic
y <- olives$linoleic
z <- olives$Region

# the default
hexpie(x,y,z)

## Not run:
# zooming in (transformation of the total number of obs in each bin)
hexpie(x,y,z, freq.trans=sqrt)

# circular shapes
hexpie(x,y,z, freq.trans=sqrt, shape="pie")

# classical piecharts
hexpie(x,y,z, freq.trans=sqrt, shape="pie", p.rule="angles")

# the total numbers of obs are reflected via alpha-blending,
# the grid is not shown and RGB colors are used
hexpie(x,y,z, freq.trans=sqrt, shape="hex", p.rule="radial",
  alpha.freq=TRUE, col="rgb", show.hex=F)

hexpie(x,y,z, freq.trans=NULL, shape="hex", p.rule="radial",
  alpha.freq=TRUE, col="rgb", show.hex=T)

```

```

require(ggplot2)
data(diamonds)
x2 <- diamonds$carat
y2 <- diamonds$price
z2 <- diamonds$color

# a standard plot with colors via ggplot2
qplot(x2,y2,colour=z2)

# the hexpie version
hexpie(x2,y2,z2,n=36)

# due to the few bins with the majority of observations
# it is sensible to zoom in
hexpie(x2,y2,z2,n=36,freq.trans=function(s) log(1+s))

# the same, but this time the central color is the most frequent one
hexpie(x2,y2,z2,n=36,freq.trans=function(s) log(1+s), decr.by.rank = TRUE)

# this way the difference is more obvious
# (although the color palette is better suited for ordinal target variables)

mat.layout <- grid.layout(nrow = 1 , ncol = 2 , widths = c(1/2,1/2), heights=1)
grid.newpage()
vp.mat <- viewport(layout = mat.layout)
pushViewport(vp.mat)

vp1 <- viewport(layout.pos.row = 1, layout.pos.col = 1)
pushViewport(vp1)

hexpie(x2,y2,z2,n=18,freq.trans=NULL,
decr.by.rank=NULL,col="div", vp = vp1)

vp2 <- viewport(layout.pos.row = 1, layout.pos.col = 2)
pushViewport(vp2)

hexpie(x2,y2,z2,n=18,freq.trans=NULL,
decr.by.rank=T,col="div", vp = vp1)
popViewport()

# random samples from the data (within bins) with many bins
# (takes some time)
grid.newpage()
hexpie(x2,y2,z2, freq.trans=function(s) log(1+s),random=1,
n=120, show.hex=FALSE, col.opt=list(bg="black"))

## End(Not run)

```

Description

This function computes the boundaries of an interval which is symmetric around the median and includes a given percentage of the data. If that's impossible due to ties the interval is chosen to minimize the squared difference between the desired percentage and the actual percentage of the observations included.

Usage

```
innerval(x, p = 0.95, data.points = TRUE)
```

Arguments

<code>x</code>	A data vector.
<code>p</code>	The percentage of observations inside the interval.
<code>data.points</code>	Whether to return the most extreme data points within the interval or the interval boundaries.

Value

A vector with the lower and upper boundaries of the interval.

Author(s)

Alexander Pilhoefer

See Also

[quantile](#)

Examples

```
x1 <- rnorm(200)
innerval(x1)
quantile(x1, c(0.025, 0.975))
```

```
x2 <- rexp(200)
innerval(x2, data.points = FALSE)
innerval(x2)
quantile(x2, c(0.025, 0.975))
```

kendalls	<i>Kendalls Tau for a matrix</i>
----------	----------------------------------

Description

Computes Kendalls Tau for a two-way table or matrix.

Usage

```
kendalls(x)
```

Arguments

x A two-way table or matrix.

Details

Kendalls tau is a rank-correlation coefficient.

Value

numeric between -1 and +1.

Author(s)

Alexander Pilhoefer

Examples

```
M <- arsim(300,c(8,8),3)
kendalls(M)
kendalls(optile(M))
```

listen	<i>auto-update interactivity for cpcp plots</i>
--------	---

Description

Makes the cpcp plot listen to selection changes and calls [resort](#) to update the display.

Usage

```
listen(init = TRUE)
```

Arguments

`init` logical. If FALSE no message will be printed. It is usually not necessary for the user to change this parameter.

Details

Choose BREAK in the File menu in order to return back to the console.

Value

No return value.

Author(s)

Alexander Pilhoefer
Department for Computer Oriented Statistics and Data Analysis
University of Augsburg
Germany

References

Alexander Pilhoefer *New approaches in visualization of categorical data: R-package extracat*
Journal of Statistical Software, submitted March 2010

See Also

[ievent.wait cpcp resort](#)

Examples

```
## Not run:  
cpcp(housing,ord = c(4,3,2,1),gap.space=0.15)  
listen()  
  
## End(Not run)  
# now make your selections and see what happens.
```

olives

olive oil fatty acids

Description

Various fatty acid measurements.

Usage

```
data(olives)
```

Format

A data frame with 572 observations on the following 11 variables.

Area a factor with levels Calabria Coast-Sardinia East-Liguria Inland-Sardinia North-Apulia Sicily South-Apulia Umbria West-Liguria

Region a factor with levels North Sardinia South

palmitic a numeric vector

palmitoleic a numeric vector

stearic a numeric vector

oleic a numeric vector

linoleic a numeric vector

linolenic a numeric vector

arachidic a numeric vector

eicosenoic a numeric vector

Test.Training a factor with levels Test Training

Examples

```
data(olives)
```

optile	<i>optimization of category orders for nominal categorical datasets and tables</i>
--------	--

Description

This function will take a categorical data object (data.frame, table, ftable, matrix, array) and optimize its category orders graphical representations. It offers an interface which will by default return the same type of object that has been passed to the function such that it is possible to write `myplot(optile(x))` for an optimized version of `myplot(x)`. The interface is extendible using ones own functions should they fulfill the requirements (see details).

Usage

```
optile(x, fun = "class", presort = FALSE, foreign = NULL,
  args = list(), perm.cat = TRUE, method = NULL, iter = 1,
  past = NULL, scale = FALSE, freqvar = NULL,
  return.data = TRUE, return.type = "data.frame", vs = 0, tree = NULL,...)
```

```
## S3 method for class 'list'
```

```
optile(x, fun = "class", presort = FALSE, foreign = NULL,
  args = list(), perm.cat = TRUE, method = NULL, iter = 1,
  past = NULL, scale = FALSE, freqvar = NULL,
  return.data = TRUE, return.type = "table", vs = 0, tree = NULL,
  k = NULL, h = NULL, ...)
```

Arguments

x	The categorical data of one of the following classes: <code>data.frame</code> , <code>table</code> , <code>fTable</code> , <code>matrix</code> , <code>array</code>
fun	The optimization function. Currently available are: <code>class</code> , <code>hamming</code> , <code>casort</code> , <code>csvd</code> , <code>rmca</code> and <code>preclass</code> . <code>Class</code> , <code>hamming</code> and <code>preclass</code> require <code>foreign = ".Call"</code> See Details.
presort	Whether or not to use <code>preclass</code> for an initial pre-ordering
foreign	Where to find the optimization function <code>fun</code> . NULL if it is an R function.
args	further arguments which will be passed to <code>fun</code> .
perm.cat	All variables with a fixed category order (e.g. ordinal variables) should have a FALSE here. Has no effect if <code>fun = "casort"</code>
method	Either NULL, "joint" or "stepwise". NULL will pass the dataframe to <code>fun</code> , "joint" will pass the Burt matrix (with all pairwise tables) and "stepwise" will repeatedly call <code>fun</code> for 2, 3, 4, and so on variables. See Details.
iter	Some optimizations depend on the initial category orders (e.g. "class" and "preclass"). This will compute the optimization for <code>iter</code> random starting points and choose the best.
past	Not implemented yet.
scale	Not implemented yet. (Function to scale resulting criteria to make them comparable)
freqvar	The name of the frequency variable, if any.
return.data	Whether to return the data or just the new orders.
return.type	The class of the object which will be returned. Defaults to the input type.
vs	An optional version number. "hamming" is currently equivalent to "class" and <code>vs = 1</code>
tree	A list whose entries are either a tree object like they are obtained from <code>hclust</code> or the string "hc". If the <code>i</code> th entry is a tree object, the <code>i</code> th variable should to be the result from cutting the tree into <code>dim(x)[i]</code> clusters via <code>subtree</code> . "hc" will compute a hierarchical clustering for the rows and columns with arguments specified in <code>args</code> .
k	A vector of integers specifying the numbers of clusters into which the tree objects shall be cut. See <code>hclust</code> .
h	Instead of a number of clusters <code>k</code> the height at which the dendrogram shall be cut can be specified. See <code>hclust</code> .
...	dots

Details

How to add a custom optimization function:

It is possible to use custom functions for the `optile` interface as long as they meet the following requirements:

The function should have the form

```
fun( data, dims, perm.cat, ... ) or
```

```
foreign( "fun", data, dims, perm.cat, ... )
where fun is the name of the function and foreign is ".Call", ".C", ...
```

The function returns a vector of the new category orders and the resulting criterion, e.g. `c(1, 3, 5, 2, 4, 5, 4, 3, 2, 1, 6, 7, 78)` (CURRENTLY: 0..k-1 instead of 1..k)

The function works with integer input weights/frequencies.

`dims` is a vector with the number of categories for each variable and `perm.cat` is a 0/1 vector which indicates whether or not to change the category order of a variable.

There are three possible types for the data argument of `fun` which can be set via method:

The argument `method` can be one of `NULL`, `"stepwise"` or `"joint"`. The default `method = NULL` indicates that `fun` accepts a multidimensional table as for instance can be produced via `xtabs`.

If `method = "joint"` a Burt matrix is computed and passed to `fun`. This matrix contains all 2-way tables and is an easy way to pass the joint 2-way-interaction data to a function. For instance `"fun=casort"` uses this data representation.

`method = "stepwise"` or `method = "sw"` passes `fun`, `data`, `foreign` as well as any args to a function called `steptile` which initially builds a 2-way table of the first pair of variables, passes it to `fun` and stores the computed category orders. Afterwards the other variables are added one by one. i.e. in a step for the k -th variable the function passes a k -way table to `fun` and a new category order for this variable is computed given the (already fixed) category orders of the variables 1 to $k-1$. E.g. for the `"class"` and `"hamming"` criteria this is a lot faster than the multivariate (`method = NULL`) version.

CURRENTLY AVAILABLE SORTING FUNCTIONS:

`"class"` and `"hamming"`: both maximize the number of pairs of observations which are consistent with a pseudo diagonal, i.e. for such a pair all factor levels of one observation should be higher than the corresponding ones of the other observation. The `"hamming"` version additionally uses weights according to the hamming distance of the observations.

`"casort"`: computes a correspondence analysis (SVD) and sorts by the first coordinate vector of each dimension. For more than two dimensions Multiple CA based on the Burt matrix is used. It is also useful to compute initial category orders.

`"rmca"`: Adopts the idea of CA for $k > 2$ dimensions without dropping information: For each dimension $d = 1..k$ with categories $d_1...d_r$ compute the scaled average $k-1$ dimensional profile sdd and perform an SVD of $(sd_1...sdr)-sdd$. Like in correspondence analysis the first coordinate vector is used for the reordering.

`"csvd"`: For each variable d in $1..k$ (iteratively) compute the cumulative sums over the multidimensional table for each variable except d . Transform this multidimensional table to an $r \times s$ matrix with r being the number of categories of variable d and s being the product of these numbers for all other variables. Resort the categories of variable d by the first coordinate vector of an SVD of that matrix. Repeat this procedure for all variables in turn until a stopping criterion is met. Idea: for any variable $h \neq d$ we have $h_1 < h_2 < \dots < h_x$ due to the cumulative sums. Hence the current order of the categories will (tend to) be the same as in the coordinates of the `svd` which means that the `svd` computes coordinates for variable d with respect to the current category orders of the other variables. The algorithm uses `casort` for an initial solution to start from.

`"distcor"`: Two-way tables or matrices can also be optimized by means of the distance correlation. See [dcor](#).

"preclass": This function is used for the presorting and is written in C (foreign = ".Call"). It iteratively sorts the categories of one variable at a time. Therefore it computes the average over the remaining dimensions and scales the profiles of each category as well as the average profile. It then computes the classification criterion between each category profile and the average profile which results in one value per category. The categories are then sorted by this criterion. The procedure is very quick and yields good results in most cases. It strongly depends on the initial category orders as do the classification or hamming algorithms. It is possible to choose this for the main sorting function by setting fun = "preclass".

Value

The function returns the data. The return type is by default the same as the input type but can be redefined via return.type.

Note

This was part of the Google Summer of Code 2011.

Author(s)

Alexander Pilhofer
 Department for Computer Oriented Statistics and Data Analysis
 University of Augsburg
 Germany

Examples

```
## Not run:
##### ----- EXAMPLE I ----- #####
# ----- Cluster results from the Current Population Survey ----- #
data(CPScluster)
cpsX = subtable(CPScluster,c(5, 12, 26, 34, 38, 39), allfactor=TRUE)

# joint and stepwise classification criterion. both use a very quick algorithm.
ss <- optile(cpsX,presort=TRUE, return.data=TRUE, method="joint")
ss2 <- optile(cpsX,presort=TRUE, return.data=TRUE, method="sw")

# original cpcp plot
cpcp(cpsX)

# cpcp for joint algorithm
cpcp(ss)

# cpcp and fluctuation for the stepwise algorithm
# (should be good for pcpl plots and hierarchical plots)
fluctile(xtabs(Freq~.,data=ss2[,-4]))
cpcp(ss2)

# The multivariate algorithm
ss3 <- optile(cpsX,presort=TRUE, return.data=TRUE, method=NULL)
cpcp(ss3)
```

```

# cpcp for casort algorithm
ssca <- optile(cpsX,presort=FALSE, fun = "casort", return.data=TRUE, method="joint")
cpcp(ssca)

# cpcp for rmca algorithm results. works better for the dmc data
ssR <- optile(cpsX,presort=FALSE, fun = "rmca", return.data=TRUE, method=NULL)
cpcp(ssR)

# cpcp for csvd algorithm
ssC <- optile(cpsX,presort=FALSE, fun = "csvd", return.data=TRUE, method=NULL)
fluctile(xtabs(Freq~,data=ssC[,-4]))
cpcp(ssC)

# cpcp for presort algorithm with 20 iterations
ssP <- optile(cpsX,presort=FALSE, fun = "preclass",
return.data=TRUE, method=NULL, foreign = ".Call",iter=20)
cpcp(ssP)

##### ----- EXAMPLE II ----- #####
# ----- Italian wines ----- #
library(MMST)
data(wine)

swine <- scale(wine[,1:13])
kmd <- data.frame(wine$class, replicate(9, kmeans(swine, centers = 6)$cluster) )
kmd <- subtable(kmd, 1:10, allfactor = TRUE)

cpcp(kmd)

# there is a good joint order and hence the joint result is better than the stepwise
kmd2 <- optile(kmd, method = "sw")
kmd3 <- optile(kmd, method = "joint")

cpcp(kmd2)
cpcp(kmd3)

##### ----- EXAMPLE III ----- #####
# ----- The BicatYeast microarray dataset ----- #

# ----- with different clusterings for the genes ----- #
library(biclust)
data(BicatYeast)

Dby <- dist(BicatYeast)

hc1 <- hclust(Dby, method = "ward")
hc2 <- hclust(Dby, method = "average")
hc3 <- hclust(Dby, method = "complete")

```

```

hcc1 <- cutree(hc1, k = 6)
hcc2 <- cutree(hc2, k = 6)
hcc3 <- cutree(hc3, k = 6)

km1 <- kmeans(BicatYeast, centers = 6, nstart = 100, iter.max = 30)$cluster

library(mclust)
mc1 <- Mclust(BicatYeast, G = 6)$class

clusterings <- data.frame(hcc1,hcc2,hcc3,km1,mc1)
clusterings <- subtable(clusterings, 1:5, allfactor = TRUE)

clusterings2 <- optile(clusterings, method = "joint")
clusterings3 <- optile(clusterings, fun = "casort")

cpcp(clusterings2)

# a fluctuation diagram of all but the avg. clustering
fluctile(xtabs(Freq~.,data=clusterings2[,-2]))

# compute agreement via Fleiss kappa in irr:
require(irr)
rawdata <- untableSet(clusterings2)
for(i in 1:5) levels(rawdata[,i]) <- 1:6
(kappam.fleiss(rawdata))
(kappam.fleiss(rawdata[,-2]))

##### ----- EXAMPLE IV ----- #####
# ----- The Eisen Yeast data ----- #
library(biclust)
data(EisenYeast)
SEY <- scale(EisenYeast)

Dby2 <- dist(SEY)

hc1 <- hclust(Dby2, method = "ward")
hc2 <- hclust(Dby2, method = "complete")

hcc1 <- cutree(hc1, k = 16)
km1 <- kmeans(scale(EisenYeast), centers = 16, nstart = 20, iter.max = 30)$cluster
optile( table(hcc1, km1) )

##### ----- EXAMPLE V ----- #####
# ----- The Bicat Yeast data ----- #

# how many clusters are a good choice for kmeans?
# one possible way to find out:
# compute kmeans for 100 random initial settings, sort the results (clusters)
# and compute their agreement

```

```

# e.g. via Fleiss' Kappa (available in package irr)

require(biclust)
data(BicatYeast)
require(irr)

st <- Sys.time()
fk <- NULL
for(k in 3:8){
  test <- subtable(replicate(100,kmeans(BicatYeast, centers = k)$cluster),1:100)
  test <- optile(test, fun = "casort")
  test <- optile(test, method="joint")
  test <- untableSet(test)
  for(i in 1:100) levels(test[,i]) <- 1:k
  fk <- c(fk,kappam.fleiss(test)$value)
}
Sys.time()-st
plot(x = 3:8, y = fk, type="l", lwd=2)

##### ----- EXAMPLE VI ----- #####
# ----- hierarchical clustering ----- #

# A list with hierarchical clustering objects:
require(amac)

hc1 <- hcluster(t(plants[,-1]), method="manhattan", link = "ward")
hc2 <- hcluster(t(plants[,-1]), method="manhattan", link = "complete")

hclist <- list(hc1, hc2)
tfluctile( optile(hclist, k= c(8,8) ) )

# or a table with corresponding tree objects:

tt <- table( subtree(hc1, 12)$data, subtree(hc2, 8)$data )

tfluctile(optile(tt, tree = list(hc1, hc2)))

# only one tree object, the other variable is free:

tt <- table( subtree(hc1, 8)$data, kk <- kmeans(t(plants[,-1]),centers=8)$cluster )

tfluctile(optile(tt, tree = list(hc1, NA)))

## End(Not run)

```

Description

Binary state variables indicating which of more than 30000 plants grow in that state.

Usage

```
data(plants)
```

Format

A data frame with 34781 observations on the following 70 variables.

V1 a factor with the names of more than 30000 plants, e.g. *abelia*

ab a numeric vector

ak a numeric vector

ar a numeric vector

az a numeric vector

ca a numeric vector

co a numeric vector

ct a numeric vector

de a numeric vector

dc a numeric vector

fl a numeric vector

ga a numeric vector

hi a numeric vector

id a numeric vector

il a numeric vector

in. a numeric vector

ia a numeric vector

ks a numeric vector

ky a numeric vector

la a numeric vector

me a numeric vector

md a numeric vector

ma a numeric vector

mi a numeric vector

mn a numeric vector

ms a numeric vector

mo a numeric vector

mt a numeric vector

ne a numeric vector

nv a numeric vector
nh a numeric vector
nj a numeric vector
nm a numeric vector
ny a numeric vector
nc a numeric vector
nd a numeric vector
oh a numeric vector
ok a numeric vector
or a numeric vector
pa a numeric vector
pr a numeric vector
ri a numeric vector
sc a numeric vector
sd a numeric vector
tn a numeric vector
tx a numeric vector
ut a numeric vector
vt a numeric vector
va a numeric vector
vi a numeric vector
wa a numeric vector
wv a numeric vector
wi a numeric vector
wy a numeric vector
al a numeric vector
bc a numeric vector
mb a numeric vector
nb a numeric vector
lb a numeric vector
nf a numeric vector
nt a numeric vector
ns a numeric vector
nu a numeric vector
on a numeric vector
pe a numeric vector
qc a numeric vector
sk a numeric vector
yt a numeric vector
dengl a numeric vector
fraspm a numeric vector

Source

<http://archive.ics.uci.edu/ml/datasets/Plants>

quickfechner *fechnerian scaling*

Description

This function computes a fechnerian distance matrix from either a similarity matrix or a dissimilarity matrix.

Usage

```
quickfechner(x, x.type = "diss", scale = "-", path.op = "+", sym.op = "+", rescale = FALSE)
```

Arguments

x	A similarity or dissimilarity matrix.
x.type	The type of the matrix ('sim' or 'diss').
scale	Either divide the similarities by the diagonal entries ('div', '/', '*', 'exp', 'expected', 'mult', 'multiplicative') or subtract the diagonal entries in the dissimilarity matrix ('-', '+', 'add', 'additive')
path.op	Whether to use the similarities to find multiplicative paths ('*', 'exp', 'expected', 'mult', 'multiplicative') or to use the dissimilarities and find additive paths ('+', 'add', 'additive', 'max', 'maximum').
sym.op	This sets the function which is used to ensure symmetry. "min" uses the minimum value, "+", "sum" or "mean" use the sum. "none", NA or FALSE stand for no operation and hence the resulting matrix will not necessarily be symmetric.
rescale	Whether or not the original diagonal will be used for a correction of the results.

Details

The algorithm first computes a dissimilarity matrix with a zero-diagonal. Then it iteratively tries to find shorter paths between the items.

Value

The Fechnerian distance matrix.

Author(s)

Alexander Pilhofer

Examples

```
data(olives)
#not a distance matrix, but a similarity matrix in some sense
cx <- 1-abs(cor(olives[-c(1,2,11)]))

cx2 <- quickfechner(cx)

rownames(cx2) <- names(olives)[-c(1,2,11)]
plot(hclust(as.dist(cx2)))
```

regmax

Regular maximality

Description

Checks whether or not a matrix fulfills the regular maximality or minimality.

Usage

```
regmax(x)
regmin(x)
```

Arguments

x A symmetric data matrix.

Value

boolean

Author(s)

Alexander Pilhofer

Examples

```
x <- replicate(20,rnorm(20))
cx <- abs(cor(x))
regmax(x)
regmin(x)

diag(cx) = runif(20)
regmax(x)
regmin(x)
```

resort	<i>Rearrangement of cpcp plots</i>
--------	------------------------------------

Description

Rearranging of the cpcp plot after a new selection.

Usage

```
resort(V = iset(which = iset.cur()),  
       sel = iset.selected(), listen = FALSE)
```

Arguments

V	An iset created by the cpcp function for which the sorting is to be computed. Usually it is not necessary to choose V individually.
sel	The indices of the selected cases.
listen	If TRUE the function listen will be (re-)called after the updating process is done.

Details

If called individually after making a new selection the function will simply rearrange the selected cases at the top of each category. Can also be called indirectly by [listen](#).

Value

No return value.

Author(s)

Alexander Pilhofer
Department for Computer Oriented Statistics and Data Analysis
University of Augsburg
Germany

References

Alexander Pilhofer *New approaches in visualization of categorical data: R-package extracat*
Journal of Statistical Software, submitted March 2010

See Also

[listen cpcp](#)

Examples

```
## Not run:
  cpcp(V = housing, ord = c(4,3,1,2))
  # selection via console
  s = iset()
  iset.select(what = which(ivar.data(s$Type) == "Terrace") )
  resort()

## End(Not run)
```

 rmb

Multiple Barchart for relative frequencies and generalized Spineplots

Description

The `rmb` function basically produces a Multiple Barchart for the relative frequencies of some target categories within each combination of the explanatory variables. The weights of those combinations (that is the absolute frequencies) are represented in the total with of the corresponding barchart. The result is a graphic which allows to read the conditional target distributions exactly from the graphic without losing the information about the importance (in the sense of the number of observations) of the different combinations.

Additionally the `rmb` function allows to draw spineplots instead of the barcharts within each explanatory combination. On that score it can be seen as a generalization of Spineplots.

Usage

```
## S3 method for class 'formula'
rmb(formula, data, col.vars = NULL, spine = FALSE, circular = FALSE,
     eqwidth = FALSE, cat.ord = NULL, cut = NULL, innerval = 1, freq.trans = NULL,
     num.mode = FALSE, max.scale = 1, use.na = FALSE, expected = NULL, residuals = NULL,
     model.opt = list(), gap.prop = 0.2, gap.mult = 1.5, col = "hcl", col.opt = list(), label = TRUE,
     label.opt = list(),...)
## S3 method for class 'ftable'
rmb(x, col.vars = NULL, spine = FALSE, circular = FALSE, eqwidth = FALSE,
     cat.ord = NULL, freq.trans = NULL, max.scale = 1, use.na = FALSE, expected = NULL,
     residuals = NULL, model.opt = list(), gap.prop = 0.2, gap.mult = 1.5, col = "hcl", col.opt = list(),
     label = TRUE, label.opt = list(),...)
```

Arguments

`x` Either a table or a model of class `"glm"` and family `"poisson"` or `"binomial"`. A table must be either of class `table` or of class `ftable`. The latter also implicitly defines the the order in which the variables will be added to the plot. The arguments `formula` and `data` will be omitted. Please note that the model based version is still beta and will be improved in a future release.

formula	The <code>formula</code> specifying the variables in their given order with the last variable being the target variable. The left hand is either empty or denotes the frequency variable.
data	The dataset in form of a frequency table (see <code>fTable</code> or <code>subtable</code> for more information) with a column named "Freq" or in raw format (that is the rows represent the cases and the columns represent the variables).
col.vars	Logical vector with split directions where TRUE stands for horizontal splitting. The last (target) variable is always arranged on the x-axis.
spine	If TRUE a spineplot will be drawn instead of each barchart. This is recommended for binary target variables.
circular	If TRUE a piechart will be drawn instead of each barchart. <code>spine</code> is set to FALSE.
eqwidth	If TRUE the bar length of the multiple barchart in the background no longer restricts the width of the barcharts/spineplots for the relative frequencies of the target variable.
cat.ord	A vector specifying the categories of the target variable which will be visualized in the specified order. The default is to use all categories.
cut	Numeric variables will be cut into this number of intervals. May also be a vector with specifications for each variable.
innerval	The function <code>innerval</code> is used to reduce numeric variables to an interval which is symmetric around the median contains the specified proportion of observations (or as close to this as possible).
freq.trans	This parameter allows to transform the absolute frequencies used for the underlying multiple barchart. Possible values are "log", "sqrt" or <code>list("sqrt", k)</code> . The latter stands for the k-th root transformation.
num.mode	In the numeric mode the gaps are removed and axes typical for numeric variables are drawn. Ignored for factor variables.
max.scale	The maximum value of the probability (y-axis) scale for each combination. Unsurprisingly the default is 1. The axis will be drawn if <code>yaxis</code> is TRUE.
use.na	If TRUE missing values will be changed to a level "N/A" and else (which is the default) the function <code>na.omit</code> will be called to reduce the dataset to complete cases only.
expected	There are three possibilities how to specify this parameter: <ol style="list-style-type: none"> 1. A list of integer vectors denoting the interaction terms in the poisson or proportional odds model, e.g. <code>list(c(1,2,3), c(1,4))</code> for all interactions between variables 1,2 and 3 as well as between 1 and 4. 2. A logical indicating whether or not to use a model (logit independence model). 3. A vector with expected values, e.g. from a model. If residuals remains undefined the response residuals will be plotted.
residuals	If undefined or set to FALSE only the observed values will be plotted. If <code>expected</code> is a vector with expected values it is also possible to specify residuals. This is used internally by <code>rmb.glm</code> .

<code>model.opt</code>	A list with optional parameters for model specifications. Possible parameters are:
<code>use.expected.values</code>	A logical specifying whether or not to use the frequencies predicted by the model instead of the observed values.
<code>mod.type</code>	Either "poisson" or "polr". See glm and polr .
<code>resid.type</code>	"pearson", "deviance", "working", "partial" or "response". For polr models only the latter is available.
<code>resid.display</code>	One of "values", "color" or "both". "values" will result in bars or wedges for both expected and observed values.
<code>max.rat</code>	If a model is specified and <code>resid.display = "both"</code> the x-scales will not be reduced to less than 1.
<code>gap.prop</code>	The maximum proportion of the total plot width which is used for the gaps.
<code>gap.mult</code>	The incremental multiplier for the gaps of different dimensions. The gaps corresponding to any one variable are <code>gap.mult</code> times larger than those corresponding to the next variable on the same axis.
<code>col</code>	Either a vector defining the colors of the bars or a name specifying a palette: "hsv" and "rgb" for hsv-based rainbow colors, "hcl" for hcl-based rainbow colors (default), "div" or "diverge" for hcl-based diverging colors and finally "seq" or "sequential" for hcl-based sequential colors. Additional arguments can be specified via the <code>col.opt</code> argument according to the underlying functions in the <code>colorspace</code> package, e.g. rainbow_hcl . For the hsv-based colors see rainbow . Specifying a color or palette has no effect if an expected model is defined.
<code>col.opt</code>	Further options for the color palettes. See e.g. rainbow_hcl or rainbow . Other parameters are: <ul style="list-style-type: none"> <code>col2</code> for the color of the background/weight bars, <code>line.col</code> for the color of all lines (bars, rectangles), <code>bg</code> for the background color of the whole graphic, <code>bgs</code> for the background color of each tile
<code>label</code>	Either a logical specifying whether or not to draw labels or a numeric vector defining which variables shall be labelled.
<code>label.opt</code>	A list with optional parameters for label specifications. Possible parameters are: <ul style="list-style-type: none"> <code>yaxis</code> If TRUE a vertical axis will be drawn at both sides of the plot. This is recommended when changing the <code>max.scale</code>. <code>boxes</code> Should the labels be surrounded by boxes? <code>lab.tv</code> Should the target variable be included in the labeling? <code>var.names</code> Should the variable names be shown as labels? <code>abbrev</code> An integer value specifying the number of characters to which the labels will automatically be abbreviated. The label will be truncated if it is longer than <code>abbrev</code> characters. <code>lab.cex</code> The fontsize multiplier.
<code>...</code>	further arguments. Usually not necessary.

Details

A similar way to regard the graphic is the following: A Multiple Barchart of the explanatory variables is drawn with bars in horizontal direction. Then within each of the resulting bars a barchart of the conditional distribution of the target variable is drawn with bars in vertical direction.

Value

```
invisible(TRUE)
```

Author(s)

Alexander Pilhoefer
 Department for Computer Oriented Statistics and Data Analysis
 University of Augsburg
 Germany

References

Alexander Pilhoefer *New approaches in visualization of categorical data: R-package extracat*
 Journal of Statistical Software, submitted March 2010

See Also

[mosaic](#) for an implementation of mosaicplots.

Examples

```
data(housing)
# simple example
rmb(formula = ~Type+Infl+Cont+Sat, data = housing, gap.mult = 2,
     col.vars = c(FALSE,TRUE,TRUE,FALSE), label.opt = list(abbrev = 3))

# with sqrt-transformation and horizontal splits only
rmb(formula = ~Type+Infl+Cont+Sat, data = housing, gap.mult = 2,
     col.vars = c(TRUE,TRUE,TRUE,TRUE), freq.trans = "sqrt",
     label.opt = list(abbrev = 3) )

# a generalized spineplot with the first category highlighted
rmb(formula = ~Type+Infl+Cont+Sat, data = housing, spine = TRUE,
     cat.ord = 1, mult = 2, col.vars = c(1,3,4),
     freq.trans = list("sqrt",3), label.opt = list(abbrev = 2))

# a generalized spineplot with all categories highlighted
# in a changed order
rmb(formula = ~Type+Infl+Cont+Sat, data = housing, spine = TRUE,
     cat.ord = c(3,1,2), gap.mult = 2, col.vars = c(TRUE,FALSE,TRUE,TRUE),
     freq.trans = "sqrt", label.opt = list(abbrev = 3))

# the barchart version only for categories 1 and 3
rmb(formula = ~Type+Infl+Cont+Sat, data = housing,
     cat.ord = c(1,3), gap.mult = 2, col.vars = c(TRUE,FALSE,TRUE,TRUE),
```

```

freq.trans = "sqrt", label.opt = list(abbrev = 3))

# with equal widths
rmb(formula = ~Type+Infl+Cont+Sat, data = housing, eqwidth = TRUE,
    gap.mult = 2, col.vars = c(TRUE,FALSE,TRUE,TRUE),
    label.opt = list(abbrev = 2, lab.tv = TRUE))

# ----- models and residuals ----- #
# using the logistic model: Sat by Type only

# residual shadings and expected values
rmb(formula = ~Type+Infl+Cont+Sat, data = housing,
    gap.mult = 2, col.vars = c(TRUE,FALSE,TRUE,TRUE),
    label.opt = list(abbrev = 3), expected = list(c(1,2,3),c(1,4)),
    model.opt = list(use.expected.values = TRUE, resid.display = "color") )

# residual values without shadings
rmb(formula = ~Type+Infl+Cont+Sat, data = housing,
    gap.mult = 2, col.vars = c(TRUE,FALSE,TRUE,TRUE),
    label.opt = list(abbrev = 3), expected = list(c(1,2,3),c(1,4)),
    model.opt = list( resid.display = "values") )

# residual shadings and expected values
rmb(formula = ~Type+Infl+Cont+Sat, data = housing,
    gap.mult = 2, col.vars = c(TRUE,FALSE,TRUE,TRUE),
    label.opt = list(abbrev = 3), expected = list(c(1,2,3),c(1,4)),
    model.opt = list(use.expected.values = TRUE, resid.display = "color") )

# barcharts with residual shadings and values
rmb(formula = ~Type+Infl+Cont+Sat, data = housing,
    gap.mult = 2, col.vars = c(TRUE,FALSE,TRUE,TRUE),
    label.opt = list(abbrev = 3), expected = list(c(1,2,3),c(1,4)) )

# spineplots with residual shadings and values
rmb(formula = ~Type+Infl+Cont+Sat, data = housing, spine = TRUE,
    gap.mult = 2, col.vars = c(TRUE,FALSE,TRUE,TRUE),
    label.opt = list(abbrev = 3), expected = list(c(1,2,3),c(1,4)) )

# piecharts with residual shadings and values
rmb(formula = ~Type+Infl+Cont+Sat, data = housing, circular = TRUE,
    gap.mult = 2, col.vars = c(TRUE,FALSE,TRUE,TRUE),
    label.opt = list(abbrev = 3), expected = list(c(1,2,3),c(1,4)) )

# ----- using an ftable to create the plot ----- #
tt = xtabs(Freq~Type+Cont+Infl+Sat, data = housing)
ft = ftable(tt, col.vars= c(1,4))
rmb(tt)
rmb(ft)

# ----- using a glm model ----- #
mod1 <- glm(Freq ~ Type*Infl*Cont + Type*Sat, data = housing, family = poisson)
rmb(mod1, circular = TRUE,

```

```

gap.mult = 2, col.vars = c(TRUE,FALSE,TRUE,TRUE),
label.opt = list(abbrev = 3), model.opt = list(use.expected.values = TRUE) )

# ----- the numeric mode and cuts ----- #
data(olives)
# only three cuts to show how it works
rmb(~palmitoleic+stearic+Region, data = olives, cut = 3)

require(ggplot2)
data(diamonds)
diamonds$lprice <- log(diamonds$price)
# a minority of extreme observations mess the display up:
rmb(~depth+table+lprice, data = diamonds, eqwidth = TRUE, spine = TRUE,
cut = c(36,36,5), col = "seq", num.mode = TRUE)

# we can zoom in via innerval:
rmb(~depth+table+lprice, data = diamonds, circular = TRUE,
cut = c(36,36,5), col = "div", innerval = 0.95,
num.mode = TRUE, freq.trans="log")

# price, carat and color
diamonds$lprice <- log(diamonds$price)
diamonds$lcarat <- log(diamonds$carat)
rmb(~lcarat+lprice+color, data = diamonds,
cut = c(24,24,0), col = "rgb", num.mode = TRUE,
freq.trans="sqrt", eqwidth=TRUE, max.scale=0.5)

```

rmbmat

Pairwise RMB-Plots

Description

This function generates a matrix with RMB-plots of all pairs of variables with a specified target variable. Both categorical and numerical variables are accepted and the latter will be binned. This makes the graphic useful for a mixture of variable types and the binning avoids overplotting and color mash as it occurs in (colored) scatterplots of large datasets.

Usage

```
rmbmat(x, tv, cut = 20, freqvar = NULL, plot.tv = FALSE, num.mode = TRUE, mode = "circular", eqwidth = FALSE)
```

Arguments

x	Anything that can be converted to a data.frame via as.data.frame.
tv	The index of the target variable. The target variable will not be plotted unless plot.tv is TRUE.
cut	The number of intervals into which numeric variables will be cut.
freqvar	An optional frequency variable. "Freq" is handled automatically.

<code>plot.tv</code>	Whether or not to include the target variable(s) in the plot.
<code>num.mode</code>	Whether or not to use the numeric mode (no gaps and a numeric axis) for numeric variables.
<code>mode</code>	One of "circular", "pie", "piechart", "p" or "c" for piecharts, "spine" or "s" for spineplots, "bars", "bar" or "b" for barcharts. NOT YET IMPLEMENTED: "rect" or "r" for nested rectangles. "nested.circles" are abbreviated by "nc" or "ncircles".
<code>eqwidth</code>	See rmb .
<code>freq.trans</code>	See rmb .
<code>innerval</code>	See rmb .
<code>allocation</code>	The widths and heights for the plots are proportional to <code>allocation(nlevels(x))</code> .
<code>max.scale</code>	See rmb .
<code>use.na</code>	See rmb .
<code>expected</code>	See rmb .
<code>model.opt</code>	See rmb .
<code>gap.prop</code>	See rmb .
<code>gap.mult</code>	See rmb .
<code>col</code>	See rmb .
<code>col.opt</code>	See rmb .
<code>label</code>	See rmb .
<code>label.opt</code>	See rmb and details.
<code>diag.opt</code>	A list with rmb parameters. These overwrite the general parameters for all plots on the diagonal.
<code>lower.opt</code>	The same as <code>diag.opt</code> but for the lower triangular matrix. Additionally it is possible to define a second target variable, e.g. <code>lower.opt = list(tv2 = 3, ...)</code> .
<code>upper.opt</code>	The same as <code>diag.opt</code> but for the upper triangular matrix. Additionally it is possible to define a second target variable, e.g. <code>upper.opt = list(tv2 = 3, ...)</code> .
<code>rc.opt</code>	A list with which it is possible to define parameters for single matrix cells (plots), columns or rows. This will overwrite all other parameters for the specified plots. It works like this: <pre>rc.opt = list(r2c12 = list(spine = FALSE), r1 = list(col="rgb"), c4 = list(col="seq"))</pre> where the plot in row 2 and column 14 is a spineplot, the first row uses RGB colors and the fourth column a sequential color palette. Later arguments overwrite the preceding ones. For instance in the example the plot in row 1 and column 4 will use the sequential color palette.
<code>factor.opt</code>	The same as <code>diag.opt</code> , <code>lower.opt</code> , <code>upper.opt</code> but for all pairs of two categorical variables. This overwrites the other option lists.
<code>...</code>	Further parameters.

Details

Creates a matrix of all pairwise rmb-plots using all possible rmb parameters except `cat.ord`, `expected = list()` and `residuals`. The parameters are applied to all plots and afterwards possibly overwritten by one of the parameter lists.

Value

An environment with the parameter lists and matrices. This can be used to update (parts of) the plot without a complete new construction. The `update.rmbmat` function is under development.

Author(s)

Alexander Pilhoefer

See Also

[rmb](#), [pairs](#)

Examples

```
data(olives)

rmbmat(olives, tv=2)

## Not run:
rmbmat(olives[,1:5], tv=2, col = "div", plot.tv = TRUE, lower.opt = list(tv2 = 1, col = "rgb"))

rmbmat(olives[,c(1:5,11)], tv=2, upper.opt=list(mode="s", eqwidth = TRUE),
rc.opt = list( c5 = list(eqwidth=FALSE,mode="s"), r5 = list(eqwidth=TRUE, mode="s")),allocation=NULL)

## End(Not run)
```

spread

Matrix Expansion

Description

Replaces each element of a matrix with a submatrix (containing this element) with a prespecified number of rows and columns.

Usage

```
spread(M, ncol = 1, nrow = 1)
```

Arguments

M	The matrix to be expanded.
ncol	The number of column repetitions.
nrow	The number of row repetitions.

Value

A matrix.

Note

This function will be advanced in the future to support more than two dimensions.

Author(s)

Alexander Pilhoefer
Department for Computer Oriented Statistics and Data Analysis
University of Augsburg
Germany

References

Alexander Pilhoefer *New approaches in visualization of categorical data: R-package extracat*
Journal of Statistical Software, submitted March 2010

Examples

```
M = matrix(1:12, ncol=4)
spread(M, ncol = 5, nrow = 7)
```

subtable	<i>data.frame reduction</i>
----------	-----------------------------

Description

Reduces a dataframe into a frequency table with prespecified entries. Uses a modified version of the `count` function which also accepts weights. Zero-entries can be included or excluded and the variables can be coerced into factors if necessary.

Usage

```
subtable(data, cols, freqvar = NULL,
keep.zero = FALSE, allfactor = FALSE)
```

Arguments

data	The data.frame to reduce.
cols	An ordered integer vector containing the indices of the columns to keep.
keep.zero	A logical indicating whether to include zero-cases in the output.
allfactor	A logical indicating whether to convert all variables into factor variables. Integer variables will be applied a fitting (non-lexicographic) level order.
freqvar	Optional name of a frequency variable in V. If dset contains a variable called "Freq" (see ftable) it will be defined as frequency variable if freqvar is unspecified.

Details

This function uses a modified version of [count](#) in order to provide the arguments `keep.zero` and `allfactor` and to be able to handle both raw datasets and datasets with a frequency variable.

Value

A data.frame including a "Freq" variable.

Author(s)

Alexander Pilhoefer
 Department for Computer Oriented Statistics and Data Analysis
 University of Augsburg
 Germany

References

Alexander Pilhoefer *New approaches in visualization of categorical data: R-package extracat*
 Journal of Statistical Software, submitted March 2010

Examples

```
hs2 = subtable(housing,c(3,1))
summary(hs2)
```

subtree	<i>subtrees</i>
---------	-----------------

Description

Takes a subtree of a dendrogram object such as generated by [hclust](#) according to a prespecified number of clusters or a prespecified height.

Usage

```
subtree(tree, k = NULL, h = NULL)
```

Arguments

tree	The tree object which contains the attributes merge and height in the same way as an hclust object.
k	The number of clusters at which to cut.
h	The height at which to cut.

Details

Does the same as [cutree](#) with two differences: Firstly it gives back an entire tree object, i.e. an object with attributes merge, height, labels and order, as well as data, which contains the cluster ids. Secondly the cluster ids are chosen by the heights at which the clusters were built.

Value

An [hclust](#) object.

Author(s)

Alexander Pilhoefer

See Also

[cutree](#)

Examples

```
hc <- hclust(dist(USArrests), "ave")

hcs <- subtree(hc, k = 7)
hcs2 <- subtree(hc, h= 30)

attributes(hcs)
fluctile(table(hcs$data, cutree(hc, k=7)))

par(mfrow=c(1,3))
plot(hc)
plot(hcs)
plot(hcs2)
```

tfluctile

Fluctuation diagram with additional dendrograms

Description

Plots a fluctuation diagram via [fluctile](#) and adds dendrograms for the rows and columns to it.

Usage

```
tfluctile(x, tree = NULL, dims = c(1, 2), tw = 0.2, border = NULL, tile.col = hsv(0.1, 0.1, 0.1, alpha = 0
```

Arguments

x	The two-way table or matrix with the data.
tree	A list with tree objects. This may be NULL or will be disregarded if x has an attribute <code>attr(x, "tree")</code> which should also be a list. The latter way is the standard for objects returned by <code>optile.list</code> or <code>optile</code> .
dims	If x has more than two dimensions this vector of length 2 indicates which variables to plot.
tw	The proportion of the total space to the left and at the top which is used for the dendrogram.
border	How much space is left white around the dendrogram.
tile.col	The tile color.
bg.col	A background color for the cells.
vp	A viewport to which the plot should be added or NULL.
...	further args

Value

```
invisible(TRUE)
```

Author(s)

Alexander Pilhoefer

See Also

[fluctile](#), [cfluctile](#)

Examples

```
library(ama)
hc1 <- hcluster(t(plants[,-1]), method="manhattan", link = "ward")
hc2 <- hcluster(t(plants[,-1]), method="manhattan", link = "complete")

hclist <- list(hc1, hc2)
tfluctile( tt<-optile(hclist, k= c(8,8) ) )

s1 <- subtree(hc1, k = 12)
s2 <- subtree(hc2, k = 10)

tfluctile( table(s1$data, s2$data), tree = list(s1,s2))
```

untableSet	<i>data.frame conversion</i>
------------	------------------------------

Description

Converts a frequency table into a raw data.frame.

Usage

```
untableSet(data, freqvar = NULL)
```

Arguments

data	The data.frame including a frequency variable "Freq".
freqvar	Optional name of a frequency variable in V. If dset contains a variable called "Freq" (see fable) it will be defined as frequency variable if freqvar is unspecified.

Value

A data.frame.

Author(s)

Alexander Pilhofer
Department for Computer Oriented Statistics and Data Analysis
University of Augsburg
Germany

References

Alexander Pilhofer *New approaches in visualization of categorical data: R-package extracat*
Journal of Statistical Software, submitted March 2010

Examples

```
hs2 = untableSet(housing)  
summary(hs2)
```

Index

- *Topic **categorical**
 - cpcp, 10
 - extracat, 15
- *Topic **datasets**
 - carcustomers, 4
 - CPScluster, 12
 - dmc, 14
 - olives, 24
 - plants, 32
- *Topic **event**
 - listen, 23
- *Topic **frequency table**
 - untableSet, 49
- *Topic **interactive**
 - cpcp, 10
 - extracat, 15
- *Topic **listen**
 - listen, 23
- *Topic **mosaicplots**
 - extracat, 15
- *Topic **multiple barcharts**
 - extracat, 15
- *Topic **parallel coordinates**
 - cpcp, 10
 - extracat, 15
- arsim, 2
- boxplot2g, 3
- carcustomers, 4
- cfluctile, 6, 48
- class, 10
- classcrit, 8
- contour, 4
- count, 45, 46
- cpcp, 10, 24, 36
- CPScluster, 12
- cutree, 47
- dcor, 13, 27
- dmc, 14
- extracat, 15
- fluctile, 7, 15, 16, 18, 47, 48
- fluctile3d, 18
- formula, 38
- ftable, 10, 38, 46, 49
- glm, 39
- hamcrit (classcrit), 8
- hclust, 26, 46, 47
- hexbin, 19, 20
- hexpie, 19
- iccrit (classcrit), 8
- ievent.wait, 24
- ihcrit (classcrit), 8
- innerval, 20, 21, 38
- ipcp, 10, 11
- kendalls, 9, 23
- listen, 11, 23, 36
- mosaic, 8, 17, 40
- na.omit, 38
- olives, 24
- optile, 15, 25
- pairs, 44
- plants, 31
- polr, 39
- quantile, 22
- quickfechner, 34
- rainbow, 39
- rainbow_hcl, 39

regmax, [35](#)
regmin (regmax), [35](#)
resort, [11](#), [23](#), [24](#), [36](#)
rmb, [15](#), [19](#), [37](#), [43](#), [44](#)
rmbmat, [42](#)

sccrit (classcrit), [8](#)
shcrit (classcrit), [8](#)
spread, [44](#)
stat_binhex, [20](#)
subtable, [38](#), [45](#)
subtree, [26](#), [46](#)

tfluctile, [47](#)

untableSet, [49](#)