Package 'cabootcrs'

March 3, 2022

Type Package

Title Bootstrap Confidence Regions for Simple and Multiple Correspondence Analysis

Version 2.1.0

Author Trevor Ringrose

Maintainer Trevor Ringrose <t.j.ringrose@cranfield.ac.uk>

Description Performs simple correspondence analysis on a two-way contingency table, or multiple correspondence analysis (homogeneity analysis) on data with p categorical variables, and produces bootstrap-based elliptical confidence regions around the projected coordinates for the category points. Includes routines to plot the results in a variety of styles. Also reports the standard numerical output for correspondence analysis.

Depends R (>= 3.5.0), lpSolve, colorspace

Imports methods

Suggests ellipse

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2022-03-02 23:00:02 UTC

R topics documented:

`cabootcrs-package` *Bootstrap Confidence Regions for Simple and Multiple Correspondence Analysis*

Description

Performs simple (classical) correspondence analysis on a two-way contingency table and produces bootstrap-based confidence regions around the projected coordinates for the category points. Includes additional routines for summarising the output and for plotting the results in a variety of ways, including both french and biplot styles.

Performs multiple correspondence analysis (homogeneity analysis) of a Burt matrix (a matrix of two-way contingency tables for p variables) and produces bootstrap-based confidence regions around the projected coordinates for the variable category points. This includes a new method to correct the confidence regions for the well-known distortion of the results caused by the diagonal of the Burt matrix. Also contains a highly experimental method to produce confidence regions when analysing an indicator matrix.

Details

Package:	cabootcrs
Type:	Package
Version:	2.1.0
Date:	2022-02-03
License:	GPL-3
Depends:	lpSolve, colorspace

Simple Correspondence Analysis:

Correspondence Analysis plots usually only show the coordinates for each of the row and column category points projected onto the new axes, with no indication of the degree of sampling variation. This package produces bootstrap-based confidence ellipses for each of the row and column points with respect to the axes shown.

These confidence regions are based on the sampling variation of the difference between sample and population points when both are projected onto the sample axes, allowing for variation in both the points and the axes and the correlation between them. Hence the coverage percentage is the chance of drawing a sample such that the confidence ellipse contains the population point when it is projected onto the samples axes as a supplementary point. See the reference below for further details.

There are options for different ways of generating the bootstrap resamples, notably based on either the Poisson or the multinomial distribution, with the latter allowing the option of fixed row or column sums.

Correspondence analysis results can be plotted in two main ways. The default option here is to produce a biplot where the row category points are plotted in principal coordinates (i.e. coordinates which allow for the different inertias of the axes) and the confidence ellipses are shown for these row category points. The column category points are shown in standard coordinates on this plot and drawn as directions in the common biplot style. A second biplot is also produced where the roles of the rows and columns are reversed.

The other main plotting option is to produce a "french-style" plot where both row categories and column categories are plotted as points in principal coordinates. However, again two plots are produced, one with confidence ellipses for the row category points and one with confidence ellipses for the column category points. This is a deliberate restriction, partly to reduce plot clutter but mostly to emphasize that the row and column points are in different spaces and that their relative positions should not be over-interpreted.

There are several options for different ways of plotting the data, with simple choices to vary the colour schemes, or to suppress point labels, or to show only a few of the ellipses, intended particularly to reduce the clutter in pictures with large data matrices. There is also an option for fuller control over the graphics, by supplying either files or data frames to define groups of points which can be plotted in common colours and symbols, or to suppress their point labels and ellipses.

The package can also be used just to perform Correspondence Analysis as usual, but with the above plotting options available.

The package can also be used just to produce the covariance matrices for each of the category points, which for example can then be used with ellipse() to add confidence ellipses to results from the ca() and mjca() (below) routines in the ca package.

Multiple Correspondence Analysis:

The same principles as above carry over to the Multiple Correspondence Analysis case with p categorical variables.

The recommended approach is to bootstrap the indicator matrix and then apply MCA to the derived Burt matrix. A new method is provided in the variance calculations to correct for the well-known problems induced by the diagonal of the Burt matrix.

Standard options for correcting inertias and coordinates are also given. In particular options exists such that MCA with p=2 gives the same results as SCA, so that MCA can be regarded as a proper generalisation of SCA (not all would agree with this, however).

The default plotting of the results is to produce p plots, with each plot showing confidence ellipses for all of the categories of just one variable. This is again done to reduce plot clutter, albeit at the expense of more plots. Again, numerous options for controlling the plots are provided.

Bootstrapping can also be applied to the analysis of an indicator matrix or a doubled matrix, but the procedure is highly experimental and very slow.

The data set can be input in numerous formats, and routines are provided to convert between them. However note that the data cannot be inputted as a Burt matrix because this loses some of the information in the data.

Changes from Version 1.0:

Multiple correspondence analysis routines added.

Hungarian algorithm added for axis rearranging, to replace the embarrassingly poor method used previously.

Bootstrap critical values added, and are now the default.

Numerous fairly minor changes made to existing simple correspondence analysis routines.

New routines to convert data between different formats.

There may be a little backwards incompatibility as some options to the routines have changed a bit, but standard use with few options will still work. Any changes should produce easily fixable failures to run.

Changes from Version 2.0:

Debugging, including for plotca to call dev.new() when in R but not Rstudio, and option to call it from Rstudio.

New routines to reflect axes and reorder categories, e.g. for when using results with other packages.

New routine to add supplementary points to plots.

More options and examples for plotting.

Final Notes:

The package does not use any routines from any of the other Correspondence Analysis packages, only base R routines. This was deliberate, in order to maintain control over the precise details. The only external routines used are lp.assign, from the lpSolve package, for the Hungarian algorithm, and some hcl functions from the colo(u)rspace package for plotting.

The results can be used as input to other packages, such as using ellipse() to draw the ellipses on a plot from ca(), see `covmat` and `plotca` for this.

Author(s)

T.J. Ringrose <t.j.ringrose@cranfield.ac.uk>

References

Ringrose, T.J. (2012). Bootstrap confidence regions for correspondence analysis.
Journal of Statistical Computation and Simulation. Vol 83, No. 10, October 2012, 1397-1413.

A paper on the application to MCA is in preparation.

See Also

cabootcrs , plotca , summaryca , printca , convert , covmat , allvarscovs , reflectaxes , reordercategories , addsupplementary , sca , rearrange , cabootcrsresults , cabasicresults

Examples

```
# Data frame of a contingency table, with row and column labels

data(DreamData)

# Perform (simple) correspondence analysis, calculate variances and show confidence ellipses.
# Use all defaults: 999 bootstrap replicates, Poisson resampling, calculate variances
# only for first two axes, but give usual output for up to the first 4 axes.
# Show one biplot with confidence ellipses for row points in principal coordinates,
# another biplot with confidence ellipses for column points in principal coordinates.
# In each case the other set of points are in standard coordinates, but note that the
# lines are cropped to fit the plot by default, as it is the directions that matter most.

bd <- cabootcrs(DreamData)

# Plot in "french" style where both rows and columns are in principal coordinates,
# not as a biplot, but still produce two plots, with row ellipses in one plot
# and column ellipses in the other.

plotca(bd, plottype="french")

## Not run:

# See the stored results, an object of type cabootcrsresults

bd

# Prettier printed output, no plots.

printca(bd)

# Brief summary output, similar style to ca package, no plots

summaryca(bd,datasetname="Dreams")

# Extract the covariance matrix of:
# row 4 for axes 1 and 2;
# column 1 for axes 1 and 2.
```

```
vmr4 <- covmat(bd,4,"row",1,2)
vmc1 <- covmat(bd,1,"column",1,2)

# Display all variances and covariances for each row and column, axes 1-2

allvarscovs(bd, "rows")
allvarscovs(bd, "columns")

# Convert the data set into a 223 individuals by 2 variables
# matrix of category membership values

ddnbyp <- convert(DreamData,input="CT",output="nbyp")$result

# Perform multiple correspondence analysis with all defaults:
# non-parametric resampling, analyse Burt matrix, correct for
# the Burt diagonal in the inertias, coordinates and bootstrapping.
# Note that the coordinates, inertias etc are identical to those
# from simple CA above, while the standard deviations and hence
# the ellipses are very similar, but not quite the same.

bdmca <- cabootcrs(ddnbyp, catype="mca")


## End(Not run)
```

`addsupplementary` *Calculate coordinates for supplementary points, with option to add to the currently selected plot.*

Description

`addsupplementary` calculates principal coordinate values for supplementary rows or columns in SCA, or supplementary variables in MCA, then plots them on the current selected graph, which it assumes to be appropriate.

Usage

```
addsupplementary(
  x,
  supp,
  thing = "columns",
  suppsymbol = "*",
  suppcolour = "blue",
  plotsupp = TRUE,
  varandcat = TRUE
)
```

Arguments

x	An object of class cabootcrsresults
supp	A data frame in the format: SCA supplementary rows: each row is a supplementary row category, the columns are the same as in the original data, each cell value is the cross-classified count. SCA supplementary columns: each row is a supplementary column category, the columns are the same as the rows in the original data, each cell value is the cross-classified count. MCA supplementary variables: each row is the same individual as in the original data, each column is a supplementary variable, each cell value is the category which that individual belongs to.
thing	Whether to calculate the supplementary principal coordinates for **"rows"** rows, or **"columns"** columns Note that "rows" is only needed for supplementary rows in SCA.
suppsymbol	The plot symbol used for the supplementary points
suppcolour	The colour of the supplementary points and their labels
plotsupp	TRUE if you want the points plotted on the currently active graph, FALSE otherwise
varandcat	Flag for how to construct column names for supplementary variables in MCA: **TRUE** if many variables have the same categories, e.g. Likert, column names will be varname:catname **FALSE** when variables have distinct categories, column names will just be category names

Details

To add supplementary rows in SCA, define the parameter supp as a data frame with one named row for each supplementary row and with the columns the same as in the original data.
If plotting the points, ensure that the plot for rows is selected.

To add supplementary columns in SCA define the parameter supp as a data frame with one named row for each supplementary column and with the columns the same as the rows in the original data.
If plotting the points, ensure that the plot for columns is selected.

To add supplementary variables in MCA define the parameter supp as a data frame with one column for each supplementary variable in individuals by variables format. Each row represents the same individual as the same row in the original data, and each entry is the category value for that supplementary variable. Hence the new columns should just look exactly like new columns in the 'nbyp' format.
If plotting the points, any of the standard plots is suitable.

Value

A matrix containing the principal coordinates of the supplementary points

See Also

cabootcrs-package, cabootcrs, plotca, cabootcrsresults

Examples

```
results <- cabootcrs(DreamData)
# SCA case two supplementary columns, make sure that the Columns plot is active
suppcols <- data.frame(rbind(c(5,3,6,8,12),c(1,7,3,1,5)))
suppcolpc <- addsupplementary(results, suppcols)
suppcolpc

## Not run:
# SCA case one supplementary row, make sure that the Rows plot is active
supprow <- data.frame(cbind(12,4,8,3),row.names="supprow")
supprowpc <- addsupplementary(results, supprow, thing="rows")
supprowpc

# MCA case, one or two supplementary variables, plots the same on any of the usual plots
results3 <- cabootcrs(DreamData223by3, catype="mca", varandcat=FALSE,
                      datasetname="Dream data with extra random column")
newsupcol <- c(rep(c(rep("s1",10),rep("s2",10),rep("s3",10)),8))[1:223]
newsupcol2 <- c(rep(c(rep("t1",5),rep("t2",15),rep("t3",25),rep("t4",35)),5))[1:223]
newsupcols <- cbind(newsupcol,newsupcol2)
suppvarpc <- addsupplementary(results3, newsupcol, varandcat=FALSE)
supp2varpc <- addsupplementary(results3, newsupcols, varandcat=FALSE)

## End(Not run)
```

`allvarscovs` *Extract all variances and covariances in readable form as a data frame*

Description

`allvarscovs` extracts all variances and covariances for either rows or columns and puts them in a data frame

Usage

```
allvarscovs(x, thing = "columns")
```

Arguments

`x`	An object of class `cabootcrsresults`
`thing`	Whether to extract the variances for **"rows"** rows, or **"columns"** columns Note that default is "columns" as this is more convenient for MCA

Value

A data frame with one row for each row or column category

See Also

cabootcrs-package, cabootcrs, covmat, cabootcrsresults

Examples

```
results <- cabootcrs(DreamData, showresults=FALSE)
rowvars <- allvarscovs(results,"rows")
colvars <- allvarscovs(results,"columns")

## Not run:

resultsmca <- cabootcrs(DreamData223by3, catype="mca", showresults=FALSE)
allvars <- allvarscovs(resultsmca)


## End(Not run)
```

AsbestosData *Asbestos data*

Description

Cases and severity of asbestosis, classified by years working with asbestos

Usage

```
AsbestosData
```

Format

A contingency table with 5 rows and 4 columns

rows Years working with asbestos: 0-9, 10-19, 20-29, 30-39, 40+

columns Has asbestosis or not: No (N), severity graded as 1-3 (G1-G3)

Source

Still looking for it

AttachmentData *van Ijzendoorn's attachment data*

Description

Classification of mother's attachment to her child and child's reaction

Usage

```
AttachmentData
```

Format

A contingency table with 4 rows and 4 columns

rows Infant response: Avoidant, Secure, Resistant, Disorganised

columns Mother's Classification: Dismissing, Autonomous, Preoccupied, Unresolved

Source

E.J. Beh, *Elliptical confidence regions for simple correspondence analysis*, Journal of Statistical Planning and Inference 140 (2010), pp. 2582–2588.

cabasicresults-class *A class containing the basic results from CA*

Description

This is intended for internal use within `cabootcrs` and only contains the data structures required for each bootstrap replicate

Slots

`Rprofile` Row profile matrix, class `"matrix"`

`Cprofile` Column profile matrix, class `"matrix"`

`Rweights` Matrix of weights for row points: square roots of inverse column sums, class `"matrix"`

`Cweights` Matrix of weights for column points: square roots of inverse row sums, class `"matrix"`

`Raxes` Matrix of axes for row points: right singular vectors of weighted, centred data matrix, class `"matrix"`

`Caxes` Matrix of axes for column points: left singular vectors of weighted, centred data matrix, class `"matrix"`

`r` Rank of weighted, centred data matrix, class `"numeric"`

`realr` In multiple CA, the number of singular values (Burt matrix) or squared singular values (indicator matrix) exceeding $1/p$ where p is the number of variables, class `"numeric"`

`mu` Singular values of weighted, centred data matrix, class `"numeric"`

See Also

cabootcrsresults

---

cabootcrs *Calculate category point variances using bootstrapping*

Description

cabootcrs performs simple or multiple correspondence analysis and uses bootstrap resampling to construct confidence ellipses for each appropriate category point, printing and plotting the results; for help on the package see cabootcrs-package.

Usage

```
cabootcrs(
  xobject = NULL,
  datafile = NULL,
  datasetname = NULL,
  nboots = 999,
  resampledistn = "Poisson",
  multinomialtype = "whole",
  printdims = 4,
  lastaxis = 4,
  maxrearrange = 6,
  rearrangemethod = "lpassign",
  usebootcrits = TRUE,
  groupings = NULL,
  grouplabels = NULL,
  varnames = NULL,
  plotsymbolscolours = c(19, "inferno", 18, "inferno"),
  othersmonochrome = "grey",
  crpercent = 95,
  catype = "sca",
  scainput = "CT",
  mcainput = "nbyp",
  mcatype = "Burt",
  mcavariant = "mca",
  mcasupplementary = "offdiag",
  mcaadjustinertias = TRUE,
  mcauseadjustinertiasum = FALSE,
  mcaadjustcoords = TRUE,
  mcaadjustmassctr = FALSE,
  mcaoneploteach = TRUE,
  mcashowindividuals = FALSE,
  mcavariablecolours = FALSE,
  mcacategorycolours = FALSE,
```

```
  Jk = NULL,
  varandcat = TRUE,
  likertarrows = FALSE,
  mcastoreindicator = TRUE,
  mcaindividualboot = FALSE,
  mcalikertnoise = 0.1,
  poissonzeronewmean = 0,
  newzeroreset = 0,
  bootstdcoords = FALSE,
  reflectonly = FALSE,
  showresults = TRUE,
  eps = 1e-15
)
```

Arguments

`xobject`	Name of data object (data frame or similar class that can be coerced to data frame). For simple CA (SCA) the default is contingency table format, recommended that rows >= columns. For multiple CA (MCA) the default is an n individuals by p variables matrix of category values (numbers or text).
`datafile`	Name of a text file (in " ") containing the data, same defaults as xobject, ignored if xobject is non-null
`datasetname`	A string to use as the name of the data set in the plots, defaults to name of xobject or datafile
`nboots`	Number of boostrap replicate matrices used, default and recommended minimum is 999, but 9999 is recommended if machine and data set size allows; the calculated variances will sometimes differ around the third decimal place, but the pictures should look the same. If nboots=0 then correspondence analysis is performed as usual with no variances calculated
`resampledistn`	Poisson resampling is the default for SCA, nonparametric is the default for MCA (Poisson and multinomial will both default to nonparametric) **"Poisson"** resampled matrices constructed using Poisson resampling on each cell separately (only SCA) **"multinomial"** resampled matrices constructed using multinomial resampling, treating the cells as defining one or more multinomial distributions (only SCA) **"nonparametric"** non-parametric resampling of the rows of the n individuals by p variables matrix, equivalent to multinomial (only MCA) **"balanced"** multinomial resampling balanced so that each data point occurs equally often over the resamples (only MCA) **"myresample"** resampling algorithm is contained in a file called myresample.R

`multinomialtype`
: Only relevant for multinomial sampling in SCA, otherwise ignored:

  "whole" all cells define a single multinomial distribution

  "rowsfixed" row sums fixed, each row defines a separate multinomial distribution

  "columnsfixed" column sums fixed, each column defines a separate multinomial distribution

`printdims`
: Print full correspondence analysis coordinates, contributions, correlations etc for all output dimensions up to and including this one

`lastaxis`
: Calculate variances and covariances for all output axes (dimensions) up to this one (or the number of dimensions in the solution if smaller).
  Recommended maximum is maxrearrange-1 as variances for those above this axis may be inaccurate

`maxrearrange`
: The maximum number of axes to consider when rearranging

`rearrangemethod`
: The method used to rearrange the axes:

  "lpassign" The Hungarian algorithm in the lpSolve package

  anything else The embarrassingly slow direct comparison method used in version 1.0 - don't use it

  Option is only included in case something weird goes wrong with lpSolve.

`usebootcrits`
: To be passed to the plot routine, see `plotca` for details

`groupings`
: To be passed to the plot routine, see `plotca` for details

`grouplabels`
: To be passed to the plot routine, see `plotca` for details

`varnames`
: Character p-vector naming the variables, defaults to c("Rows","Columns") in sca

`plotsymbolscolours`
: To be passed to the plot routine, see `plotca` for details

`othersmonochrome`
: To be passed to the plot routine, see `plotca` for details

`crpercent`
: To be passed to the plot routine, see `plotca` for details

`catype`
: Type of correspondence analysis:

  "sca" Simple (classical) correspondence analysis of a contingency table

  "mca" Multiple correspondence analysis of a Burt, indicator or doubled matrix

`scainput`
: Format of input data, only applies for SCA:

  "CT" Contingency table of counts, preferably with rows >= columns

  "nbyp" An n individuals/objects/data points by p=2 categorical variables matrix, where each row is a different data point and each column contains the category for that data point on that variable, where these categories can be numbers, strings or factors

  "nbypcounts" Similar to the above, but each row represents all of the data points taking the same combination of categories, and the first column contains the count for this combination (hence the name used here is a bit of a misnomer, but it emphasises the similarities to an n by p=2)

`mcainput` Format of input data, only applies for MCA:

"nbyp" An n individuals/objects/data points by p categorical variables matrix, where each row is a different data point and each column contains the category for that data point on that variable, where these categories can be numbers, strings or factors

"nbypcounts" Similar to the above, but each row represents all of the data points taking the same combination of categories, and the now first column contains the count for this combination, so that the input matrix is n by p+1 (hence the name used here is a bit of a misnomer, but it emphasises the similarities to an n by p)

"indicator" An n by sum-of-distinct-variable-categories (i.e. sum of elements of Jk) indicator matrix

`mcatype` Format of data matrix analysed, only applies for MCA:

"Burt" Analyse the Burt matrix. Output will be given for the column (variable category) points but not for the (identical) row points.
NOTE: it is highly recommended that this version is used

"indicator" Analyse the indicator matrix. Output will be given for the column (variable category) points but not usually for the row (individual) points.
NOTE: the bootstrap method used in this case is highly experimental and very slow, see Details section part (2)

"doubled" Analyse the doubled matrix. Output will be given for the column points (two points, high and low, for each variable) but not usually for the row (individual) points.
NOTE: the bootstrap method used in this case is highly experimental and very slow, see Details section part (2)

`mcavariant` Currently must be "mca", placeholder for future updates

`mcasupplementary`
How the sample points are projected as supplementary points onto the bootstrap axes when calculating the variances in MCA of a Burt matrix, see Details section for full explanation

"offdiag" Only the off-diagonal parts of the Burt matrix are used in the projection

"all" Projection is calculated in the usual way

If "offdiag" then when p=2 the variances will be very similar to those from SCA.

If "all" then the fact that the diagonal elements of the Burt matrix are by definition the same for both the sample and bootstrap matrices means that the projected differences between sample and population points, and hence the variances, will be artificially small.

NOTE: if mcaadjustinertias is FALSE then mcasupplementary will be set to "all", because adjusting the coordinates in the calculation of the variances but not adjusting the inertias makes no sense

`mcaadjustinertias`
Whether to adjust inertias to allow for the meaningless inertia terms induced by the diagonal of the Burt matrix in MCA:

TRUE Analysing Burt matrix: subtract 1/p from all singular values, then positive singular values are multiplied by p/(p-1) while negative ones are ignored
Analysing indicator matrix: same applies but to the squared singular values

FALSE No adjustment

If TRUE then when p=2 the inertias will agree with those from SCA.

NOTE: inertias are the square (Burt) or fourth power (indicator) of the (adjusted) singular values.

NOTE: if mcaadjustinertias is FALSE then mcaadjustcoords will also be set to FALSE and mcasupplementary set to "all", as adjusting the coordinates but not the inertias makes no sense

`mcauseadjustinertiasum`

How to define the total inertia in MCA, whether to just use the sum of the adjusted inertias:

TRUE The inertias are expressed as a percentage of the sum of the adjusted inertias (Benzecri)

FALSE The inertias are expressed as a percentage of the average of off-diagonal inertias (Greenacre), note that this will be incorrect if Jk includes categories that are not observed in the data

If TRUE then when p=2 the inertias will agree with those from SCA
If TRUE then the percentage inertias will sum to 100%
If FALSE then the percentage inertias will usually sum to less than 100%

`mcaadjustcoords`

Whether to adjust the principal coordinates in MCA using the adjusted inertias above, as in Greenacre and Blasius, p68:

TRUE Adjust coordinates, but only for column points

FALSE No adjustment

If TRUE then when p=2 the coordinates will agree with those from SCA.

NOTE: if mcaadjustinertias is FALSE then mcaadjustcoords will also be set to FALSE, as adjusting the coordinates but not the inertias makes no sense

`mcaadjustmassctr`

Whether to adjust the point masses and column contributions in MCA so that the masses and contributions are with respect to each variable (as in SCA) rather than with respect to all variables together:

TRUE Multiply point masses and contributions by p so that they sum to p over all variables

FALSE No adjustment

If TRUE then when p=2 the CTR will agree with those from SCA, though when p>2 the contributions can be >1

`mcaoneploteach` Parameter passed to `plotca` for MCA.

A flag saying whether to produce one plot for each variable, where confidence ellipses are shown for that variable but not others:

TRUE p plots are produced, each showing confidence regions for the category points for just one variable

FALSE only one plot produced, with confidence regions shown for each category point of each variable, which could be very "busy"

`mcashowindividuals`

Parameter passed to `plotca` for MCA.

For MCA on an indicator matrix only, a flag saying whether to plot the individuals:

TRUE plot the individuals, which could be very "busy".
NOTE: if mcaindividualboot=TRUE this also plots the CRs constructed using the experimental method, see Details section part (2)

FALSE don't plot the individuals

`mcavariablecolours`

Parameter passed to `plotca` for MCA:

TRUE In MCA each variable has its own colour, and all category points and ellipses for that variable have the same colour

FALSE Colours chosen in the default way

`mcacategorycolours`

Parameter passed to `plotca` for MCA:

TRUE In MCA each category number has its own colour, and all points and ellipses for that category number have the same colour, for all variables (intended for Likert type data so that all category 1 points are the same colour etc)

FALSE Colours chosen in the default way

`Jk` The number of classes for each variable in MCA, as a list or vector, which only needs specifying when inputting an indicator matrix, as in other cases it can be derived from the input matrix

`varandcat` Flag for how to construct variable category names:

TRUE names are varname:catname, to be used if many variables have the same categories, e.g. Likert

FALSE names are just category names, to be used if variables all have distinct categories

`likertarrows` Parameter passed to `plotca` for MCA.

If TRUE then, for likert-type ordered categorical data, draw arrows connecting the category points for each variable, with the arrow drawn from a category point to the next higher category point

`mcastoreindicator`

If TRUE then store the indicator matrix created for MCA

`mcaindividualboot`

If TRUE then use the experimental method to bootstrap an indicator or doubled matrix, see Details section part (2) for full explanation

`mcalikertnoise` The "noise" value to use in the experimental method (above) to bootstrap an indicator or doubled matrix, see Details section part (2) for full explanation

`poissonzeronewmean`
: Experimental method for SCA to deal with contingency tables where zero cells could have been non-zero, i.e. they are not structural zeros.
Only relevant for Poisson sampling in SCA, otherwise ignored:
0 : no effect, method as described in paper
1 : cells which are zero in the data are instead resampled from a Poisson distribution with this mean, which should be very small (say 0.1); this is for situations where rare cases did not occur but could have done, so that it might be appropriate for zero cells in the sample to be occasionally non-zero in resamples

`newzeroreset`
: Experimental method for SCA to deal with sparse contingency tables.
Only relevant for SCA, otherwise ignored:
0 : no effect, method as described in paper
1 : if a cell value is non-zero in the sample but zero in the resample then it is reset to 1 in the resample, so that the sparsity structure of the sample is maintained in the resample. This can be useful with sparse data sets and, in effect, conditions on the sample sparsity structure

`bootstdcoords`
: If TRUE then produce bootstrap variances for points in standard coordinates instead of principal coordinates
Note: intended only for experiments with the methodology

`reflectonly`
: If TRUE then just allow for axis reflections and not axis reorderings
Note: intended only for experiments with the methodology

`showresults`
: If TRUE then output the results using `summaryca` and `plotca`, otherwise output suppressed

`eps`
: Any value less than this is treated as zero in some calculations and comparisons

Details

This routine performs all of the usual Correspondence Analysis calculations while also using bootstrapping to estimate the variance of the difference between the sample and population point when both are projected onto the sample axes in principal coordinates. This is done for each row and column category on each dimension of the solution, allowing for sampling variation in both the points and the axes.

It hence constructs confidence ellipses for each category point, plots the results by a call to `plotca` and prints the usual Correspondence Analysis summary output and the calculated standard deviations through a call to `summaryca`. Use `printca` for more detailed numerical results.

For further examples and help on the package as a whole see `cabootcrs-package`.

(1) Corrections for Burt diagonal

It is well-known that in multiple CA (MCA) the results are distorted by the diagonal elements of the Burt matrix. As well as the standard methods to correct for this, here we propose and implement a new method to correct for this when bootstrapping. If bootstrapping is applied in a naive way then, even when the standard corrections are used, the estimated variances will be much too small because diagonal elements of the standardised Burt matrix are the same in every bootstrap replicate, thus underestimating the true variation in the data.

All bootstrapping is performed on the indicator matrix (or equivalently the n by p matrix) and the resampled Burt matrix is then constructed from the resampled indicator matrix in the usual way.

Included here are the usual corrections to the inertias (mcaadjustinertias=TRUE, the default) and the coordinates (mcaadjustcoords=TRUE, the default). In addition you can choose to use, as the total inertia, either the sum of these adjusted inertias (mcauseadjustinertiasum=TRUE) as proposed by Benzecri or the average of the off-diagonal inertias (mcauseadjustinertiasum=FALSE, the default) as proposed by Greenacre. You can adjust (multiply by p) the Contribution figures in MCA so that they sum to p over all variables, i.e. an average of 1 for each variable as in SCA (mcaadjustmassctr=TRUE), rather than a total of 1 over all variables, as usually in MCA (mcaadjustmassctr=FALSE, the default). Note that when p=2 this will be the same as in SCA, but when p>3 you can get contributions greater than 1, so use with caution. This also adjusts (multiplies by p) the point masses so that they sum to 1 for each variable, rather than over all variables, same caveat applies.

The fundamental problem with MCA is that in a Burt matrix each diagonal element is the value of a variable category cross-classified with itself, so it is always equal to the number of times that category appears. Hence if a category appears k times then the row (or column) in the Burt matrix consists of p blocks each of which sum to k, so its row (and column) sum is kp.

Therefore when the row (or column) profile matrix is calculated the diagonal elements of the Burt matrix are always all 1/p while the elements for the categories in the offdiagonal blocks sum to 1/p in each block. Hence when the projected difference between the sample and resample row (or column) profile matrices is calculated this is artificially small because the diagonals of the two matrices are always the same, no matter how different the off-diagonal elements are.

The new method to correct for the diagonal elements of the Burt matrix when calculating variances therefore works by re-expressing the coordinates purely in terms of the interesting and variable off-diagonal elements, excluding the uninteresting and constant diagonal elements.

First calculate the Burt principal coordinates (PC), but with the diagonal elements of the profile matrix ignored (or set to zero). It is easy to verify that the usual Burt principal coordinates can be re-expressed, using the singular values (SV), as

Burt PC = (Burt SV / (Burt SV - (1/p))) x Burt PC without diagonal element

Hence in the bootstrapping the sample and resample points are re-expressed in this way and their differences when projected onto the bootstrap axes are calculated as usual.

Similarly the adjusted principal coordinates are calculated as

adj Burt PC = (p/(p-1)) x ((Burt SV - (1/p)) / Burt SV) x Burt PC

So, when using the usual adjusted coordinates (and adjusted inertias), both of the above will be used, hence ending up with just a correction of (p/(p-1)). The unadjusted coordinates can be used, but this is not recommended.

One consequence of this correction is that when mcaadjustinertias=TRUE, mcaadjustcoords=TRUE, mcauseadjustinertiasum=TRUE, mcaadjustmassctr=TRUE and mcasupplementary="offdiag" then when p=2 the bootstrap variances for MCA are almost the same as those for SCA, while all other results for MCA are the same as those for SCA. The package author regards this as a good thing, making MCA more of a proper generalisation of MCA, but recognises that some people regard MCA as a fundamentally different method to SCA, linked only by the common algebra.

Note that if this adjustment is not made then in the p=2 case it is easy to see that the projected differences are half those in the SCA case and the standard deviations are a quarter the size.

The new method will be written up for publication once this update to the package is finished.

(2) Experimental method for bootstrapping indicator matrices

A highly experimental method is included for bootstrapping with an indicator or doubled matrix in the case of ordered categorical (e.g. Likert scale) data. This has not been studied or optimised extensively, and is currently very slow, so use is very much at the user's discretion and at user's risk. To try it, choose:

mcatype="indicator" or mcatype="doubled" with nboots>0

If CRs are required for the individual points then also choose:

mcaindividualboot=TRUE

The bootstrap methodology used here relies on the comparison of bootstrap to sample points when both are projected onto bootstrap axes. In SCA this is fine because when looking at column points the axes are given by the rows and vice versa, and row i and column j each represent the same category in all bootstrap replicates. Similarly in MCA with a Burt matrix when looking at column points the axes are given by the rows, which again each represent the same category in all bootstrap replicates.

However, with an indicator or doubled matrix, row i of the matrix does not represent the same individual in each replicate, it just represents the i-th individual drawn in the resampling for that particular replicate. In order for this type of bootstrapping to work with an indicator or doubled matrix we would need a resampling method whereby the i-th row represents the i-th individual in all bootstrap replicates. The resampled row would need to represent the answers of the same individual "on another day". This might make sense with questionnaire data where an individual's answers have uncertainty attached, in that if asked the same questions on multiple occasions they would give different answers due to random variation rather than temporal change. If a believable model for the sampling, and hence the resampling, could be derived (CUB models perhaps) then this could lead to CRs for an individual point, representing the uncertainty in what that person actually thinks.

The method here uses the same idea of treating the sample row as representing an individual, and bootstrap replicates of that row as representing the variability in how that individual might have answered the questionnaire (or similar). However, instead of an explicit model to represent this variability, it is generated by the data themselves. The bootstrap replicate indicator (or doubled) matrix is generated as usual, by either non-parametric (multinomial) or balanced resampling, and then its rows are "matched" to the rows of the sample indicator (or doubled) matrix. The rows of the resampled matrix are reordered so that the rows of the resampled matrix are, overall, as similar as possible to the same rows of the sample matrix. Hence the resampled rows can reasonably be viewed as representing the same individual in each bootstrap replicate, and hence variances for the column points (categories) and row points (individuals) can be produced.

The matching again uses the Hungarian algorithm from lpSolve. This only makes sense if all variables are ordered categorical. Applying this to (ordered) categorical data results in a very large number of ties, so the mcalikertnoise parameter defines the standard deviation of white noise added to each of the sample and resample category numbers for the purpose of the matching (only).

Note that this is very slow for all but small data sets, and if all of the CRs for individuals are shown then the plot is impossibly busy. Hence it is recommended that this experimental method only be used for fairly small data sets where CRs are wanted for only a few example individuals. A better approach might be to average the CRs of all of the individuals who give the same results in the sample, but this is not implemented yet.

Note that supplementary row points in indicator matrix MCA are usually regarded as different

people answering the same questions, whereas in this case for CRs to make any sense we need to regard them as the same people answering the same questions on different days.

(3) Critical values

Bootstrap critical values are calculated by re-using the bootstrap replicates used to calculate the variances, with a critical value calculated for each ellipse. The projected differences between bootstrap and sample points are ordered and the appropriate percentile value picked. These are usually slightly larger than the χ^2 critical values. Only 90%, 95% (default) and 99% critical values are calculated.

Alternatively use χ^2 critical values, usually with df=2, but with df=1 if only 2 non-zero cells in row/col.

The experimental method to construct ellipses for individuals in MCA always uses bootstrap critical values

Value

An object of class cabootcrsresults

See Also

cabootcrs-package, cabootcrsresults, plotca, printca, summaryca, covmat, allvarscovs

Examples

```
# Simple CA (SCA) of a 5 by 4 contingency table, using all SCA defaults:
# 999 bootstraps, Poisson resampling, variances for up to first four axes,
# usual output for up to the first 4 axes,
# one biplot with CRs for rows in principal coordinates and another with
# CRs for columns in principal coordinates

bd <- cabootcrs(DreamData)

## Not run:

# Same data set with a completely random three-category third variable added,
# analysed with MCA but with standardisations which mimic SCA as much as possible

bd3 <- cabootcrs(DreamData223by3, catype="mca")

Explicitly stating what the rows and columns represent, often needed for a contingency table

bd <- cabootcrs(DreamData, datasetname="Maxwell's dream data",
                varnames=c("What the rows are","What the columns are"))

# Multiple CA (MCA) of 3 categorical variables with all defaults:
# non-parametric resampling, Burt matrix analysed,
# each variable has one plot with it in colour with CRs shown, other variables in monochrome.
# Same data set but now as 223 by 3 matrix, with random 3rd column (with 3 categories) added.

bd3 <- cabootcrs(DreamData223by3, catype="mca")
```

```
# Comparison of SCA to MCA with p=2, by converting contingency table to 223 by 2 matrix.
# Note that the coordinates and inertias etc are the same while the standard deviations
# and hence the ellipses are very similar but not identical.

bd <- cabootcrs(DreamData)
DreamData223by2 <- convert(DreamData,input="CT",output="nbyp")$result
bdmca <- cabootcrs(DreamData223by2, catype="mca", varandcat=FALSE)

# Not adjusting inertias, which means that coordinates will also not be adjusted and
# the bootstrapping will use the Burt diagonal.
# Note how the coordinates are larger but the inertias and ellipses are smaller.

bdmcaunadj <- cabootcrs(DreamData223by2, catype="mca", varandcat=FALSE, mcaadjustinertias=FALSE)

# Applying the standard adjustments to inertias and coordinates, but with
# the bootstrapping still using the Burt diagonal.
# Note how inertias and coordinates are now the same as SCA, but ellipses are smaller.

bdmcaadjbutall <- cabootcrs(DreamData223by2, catype="mca", varandcat=FALSE, mcasupplementary="all")


# Effect of sample size in SCA:

bdx4 <- cabootcrs(4*DreamData)
bdx9 <- cabootcrs(9*DreamData)

ba <- cabootcrs(AttachmentData)

bs <- cabootcrs(SuicideData)

bas <- cabootcrs(AsbestosData)


# Options for SCA:

# SCA with multinomial resampling, with the matrix treated as a single multinomial distribution

bdm <- cabootcrs(DreamData, resampledistn="multinomial")

# Fix the row sums, i.e. keep sum of age group constant

bdmrf <- cabootcrs(DreamData, resampledistn="multinomial", multinomialtype="rowsfixed")

# Use chi-squared critical values for the CRs

bdchisq <- cabootcrs(DreamData, usebootcrits=FALSE)

# Just perform correspondence analysis, without bootstrapping

bdnb0 <- cabootcrs(DreamData, nboots=0)


# Effect of sample size in MCA:
```

```
bn <- cabootcrs(NishData, catype="mca")


# Options for MCA

# Using default settings the SCA and MCA standard results are the same when p=2,
# bootstrap standard deviations (multinomial/nonparametric) are similar but not identical

bdsca <- cabootcrs(DreamData,resampledistn="multinomial")
bdmca <- cabootcrs(convert(DreamData,input="CT",output="nbyp")$result, catype="mca")

# Row A can be labelled A rather than R:A
# because the three variables have all different category names

bd3l <- cabootcrs(DreamData223by3, catype="mca", varandcat=FALSE)

# Balanced resampling, each of the 223 rows occurs 999 times in the 999 resamples

bd3b <- cabootcrs(DreamData223by3, catype="mca", resampledistn="balanced")

# Do not adjust inertias, coordinates or contributions
# (if inertias are not adjusted then coordinates are also not adjusted)

bd3unadj <- cabootcrs(DreamData223by3, catype="mca",mcaadjustinertias=FALSE)

## Comparisons to ellipses from FactoMineR

# Generate some completely random uniform categorical data, construct ellipses.
# The cabootcrs ellipses are very large and overlap extensively, as you would expect
# from completely random data.
# The FactoMineR ellipses are much smaller, often with minimal overlaps,
# giving a completely false impression of genuine differences between categories.

library(FactoMineR)
p <- 4
maxcat <- 5
n <- 100
Xnpr <- apply( as.data.frame( matrix( round(runif(n*p,0.5,maxcat+0.5)), n, p)), 2, factor )
fr <- MCA(Xnpr, method="Burt")
plotellipses(fr)
br <- cabootcrs(Xnpr, catype="mca", showresults=FALSE)
plotca(br, mcacategorycolours = TRUE, showcolumnlabels=FALSE)

## Comparisons to results in ca and FactoMineR

Summary: If using unadjusted inertias, coordinates the packages produce identical results,
apart from differences in presentation (rounding off, the naming of rep/cor/cos2).

Summary: When using adjusted inertias and coordinates (not an option in FactoMineR::MCA)
the correlations in ca::mjca no longer sum to 1 over all dimensions, in cabootcrs they do.
Ratios are the same for each dimension, but not each point, they are standardised differently.
```

```
# Example comparisons with random data
library(FactoMineR)
library(ca)
p <- 4
maxcat <- 5
n <- 100
Xnpdf <- as.data.frame( matrix( round(runif(n*p,0.5,maxcat+0.5)), n, p))
Xnpr <- apply( Xnpdf, 2, factor )

# Note that ca::mjca only accepts the data as numerical,
# FactoMineR::MCA only acccepts the data as characters
rbun <- cabootcrs(Xnpr, catype="mca", nboots=0, mcaadjustinertias = FALSE)
rcun <-  mjca(Xnpdf,lambda="Burt")
summary(rcun)
rfm <- MCA(Xnpr,method="Burt", graph=FALSE)
summary(rfm)
rb <- cabootcrs(Xnpr, catype="mca", nboots=0)
rc <- mjca(Xnpdf)
summary(rc)
realr <- rb@br@realr
rb@ColREP[,1:realr]
rc$colcor[,1:realr]
apply(rb@ColREP[,1:realr],1,"sum")
apply(rc$colcor[,1:realr],1,"sum")


## End(Not run)
```

`cabootcrsresults-class` *A class containing the results from CA with bootstrapping*

Description

This contains all of the usual output from simple or multiple CA, plus the results of the bootstrap analysis and the various settings used for this.

Details

The meanings and possible values for the settings are described in `cabootcrs`

Slots

`br` The basic results from CA, class `cabasicresults`

`datasetname` Name of the data set for printing, class `"character"`

`DataMatrix` The sample data matrix, class `"matrix"`

`rows` Number of rows, class `"numeric"`

`columns` Number of columns, class `"numeric"`

`rowlabels` Row category labels, class `"character"`

`collabels` Column category labels, class `"character"`

`varnames` Names of the variables, class `"character"`

`Rowprinccoord` Principal coordinates for row points, class `"matrix"`

`Colprinccoord` Principal coordinates for column points, class `"matrix"`

`Rowstdcoord` Standard coordinates for row points, class `"matrix"`

`Colstdcoord` Standard coordinates for column points, class `"matrix"`

`RowCTR` Contributions for row points, class `"matrix"`

`RowREP` Representations for row points, class `"matrix"`

`ColCTR` Contributions for column points, class `"matrix"`

`ColREP` Representations for column points, class `"matrix"`

`RowVar` Variances for row points, class `"matrix"`

`RowCov` Covariances for row points, class `"array"`

`ColVar` Variances for column points, class `"matrix"`

`ColCov` Covariances for column points, class `"array"`

`inertiasum` Total inertia, class `"numeric"`

`inertias` Axis inertias, class `"matrix"`

`rowmasses` Masses of row points, class `"numeric"`

`colmasses` Masses of column points, class `"numeric"`

`nboots` Number of bootstrap replicates used to calculate the (co)variances, class `"numeric"`. If nboots=0 then standard CA or MCA is performed with no confidence regions produced.

`resampledistn` Distribution used for resampling, class `"character"`

`multinomialtype` Form of multinomial resampling used, class `"character"`

`sameaxisorder` Number of resamples with no reordering in first six bootstrap axes, class `"numeric"`

`poissonzeronewmean` Mean used for resampling zero cells, class `"numeric"`

`newzeroreset` Option to reset resample zero cells, class `"numeric"`

`printdims` Number of dimensions to print, though note that all are stored, class `"numeric"`

`axisvariances` Number of axes for which variances were calculated and are stored, class `"numeric"`

`bootcritR` Bootstrap critical values for row points, class `"array"`

`bootcritC` Bootstrap critical values for column points, class `"array"`

`usebootcrits` Whether to use bootstrap critical values for confidence ellipses, class `"logical"`

`catype` Type of correspondence analysis performed, class `"character"`

`mcatype` Type of multiple correspondence analysis performed, class `"character"`

`mcaindividualboot` Whether the experimental method to bootstrap an indicator or doubled matrix was used, class `"logical"`

`IndicatorMatrix` The indicator matrix derived from the data matrix, class `"matrix"`

`Jk` The number of classes for each variable, class `"numeric"`

`p` The number of variables, class `"numeric"`

`mcalikertnoise` The noise value used in the experimental method to bootstrap an indicator or doubled matrix, class `"numeric"`

`mcaadjustinertias` Whether MCA inertias were adjusted, class `"logical"`

`mcauseadjustinertiasum` Whether the adjusted MCA inertia sum was used, class `"logical"`

`mcaadjustcoords` Whether the MCA coordinates were adjusted, class `"logical"`

`mcaadjustmassctr` Whether the MCA masses and contributions were adjusted, class `"logical"`

`mcasupplementary` How supplementary points were calculated when bootstrapping a Burt matrix, class `"character"`

See Also

`cabasicresults`

`convert` *Converting a data matrix from one format into another*

Description

`convert` recodes a data matrix from one format, used by versions of correspondence analysis, into another (n objects by p variables, counts for distinct combinations of p variables, indicator matrix, contingency table).

Usage

```
convert(
  Xinput,
  input = "nbyp",
  output = "indicator",
  Jk = NULL,
  maxcat = NULL,
  varandcat = TRUE
)
```

Arguments

`Xinput` A data matrix, in the form of a data frame or similar

`input` The format of the input matrix:

"nbyp" An n individuals/objects/data points by p categorical variables matrix, where each row is a different data point and each column contains the category for that data point on that variable, where these categories can be numbers, strings or factors

"nbypcounts" Similar to the above, but each row represents all of the data points taking the same combination of categories, and the first column contains the count for this combination (hence the name used here is a bit of a misnomer, but it emphasises the similarities to an n by p)

"indicator" An indicator matrix, similar to the n by p matrix except that a variable with J_k categories is represented by J_k columns and a data point taking the i-th category has 1 in the i-th of these columns and a zero in the others

"CT" A contingency table of counts

`output` The format of the output matrix:

"nbyp" As above

"nbypcounts" As above

"indicator" As above

"doubled" Similar to indicator but each variable is now represented by 2 columns, and a data point taking the i-th category for a variable with J_k categories is given the values J_k-i in the first (low) column and i-1 in the second (high) column

`Jk` A list containing the number of distinct categories for each variable.
Either Jk or maxcat must be specified if input is "indicator"

`maxcat` The maximum category value, for use when all variables are Likert on a scale of 1 to maxcat.
Either Jk or maxcat must be specified if input is "indicator"

`varandcat` Flag for how to construct column names in an indicator matrix:

TRUE if many variables have the same categories, e.g. Likert, column names will be varname:catname

FALSE when variables have distinct categories, column names will just be category names

Value

A list containing:

result the output data matrix formatted according to the output argument

varnames a list of length p containing the names of each variable

catnames a list/array (of length p) containing the lists (of length Jk[i]) of category names for each variable

Jk a list of length p containing the number of distinct categories for each variable

p the number of variables

See Also

`getBurt` to obtain a Burt matrix or a subset of an existing one
`getCT` to obtain a contingency table (only if p=2)
`getindicator` to obtain an indicator matrix
`getdoubled` to obtain a doubled matrix if all variables are ordered categorical with numbered categories

Other conversion functions: `getBurt()`, `getCT()`, `getdoubled()`, `getindicator()`

Examples

```
dreamdataCT <- DreamData
dreamdatanbyplist <- convert(dreamdataCT,input="CT",output="nbyp")
dreamdatanbyp <- dreamdatanbyplist$result

## Not run:

dreamdataCTb <- table(dreamdatanbyp)
dreamdatanbypcounts <- convert(dreamdatanbyp,input="nbyp",output="nbypcounts")$result
dreamdataindicatorlist <- convert(dreamdatanbypcounts,input="nbypcounts",output="indicator")
dreamdatanbypb <- convert(dreamdataindicatorlist$result,input="indicator",
                          output="nbyp",Jk=dreamdataindicatorlist$Jk)$result

nishdatanbyp <- NishData
nishdataindicator <- convert(nishdatanbyp)$result
nishdataBurt <- t(nishdataindicator)%*%nishdataindicator


## End(Not run)
```

---

covmat *Extract a single 2 by 2 covariance matrix*

Description

covmat extracts a 2 by 2 covariance matrix for one data point on two dimensions, allowing the confidence ellipse to be plotted

Usage

```
covmat(x, i, thing = "column", axis1 = 1, axis2 = 2, show = TRUE)
```

Arguments

x	An object of class cabootcrsresults
i	The number of the row or column, note that in MCA this will be the number of the variable category (e.g. for p=3 variables with 5 categories each, column 8 is the 3rd category of the 2nd variable)
thing	Whether to extract the covariance matrix for the i-th **"row"** row, or **"column"** column Note that default is "column" as this is more convenient for MCA
axis1	First axis for which (co)variances are required
axis2	Second axis for which (co)variances are required
show	If TRUE then print the extracted covariance matrix

Details

This can be used with the ellipse() package to add the confidence ellipse to a picture from another package

Example: confidence ellipse for row or column i on axes 1,2 from cabootcrs() output Results is:

lines(ellipse(x=covmat(Results,i,"row",1,2,FALSE), centre=Results@Rowprinccoord[i,cbind(1,2)], npoints=1000), cex=1, pch=".", col="blue")
lines(ellipse(x=covmat(Results,i,"column",1,2,FALSE), centre=Results@Colprinccoord[i,cbind(1,2)], npoints=1000), cex=1, pch=".", col="blue")

Note that reflectaxes will be needed if cabootcrs() and ca() axes are reflected with respect to each other

Value

An object of class "matrix" (square symmetric, 2 by 2)

See Also

cabootcrs-package, cabootcrs, allvarscovs, cabootcrsresults

Examples

```
results <- cabootcrs(DreamData, showresults=FALSE)
row2covmataxes12 <- covmat(results,2,"row")
col3covmataxes23 <- covmat(results,3,"column",2,3)

## Not run:

# There are now 3 variables with 5,4,3 categories, hence 12 columns
resultsmca <- cabootcrs(DreamData223by3, catype="mca", showresults=FALSE)
row2covmataxes12mca <- covmat(resultsmca,2,"column")
col3covmataxes23mca <- covmat(resultsmca,8,"column",2,3)
newvarcat2covmataxes12mca <- covmat(resultsmca,11,"column")


# Use ellipse() to put confidence regions around row points on a plot produced by ca().
# Note that reflectaxes() will be needed if cabootcrs() and ca() axes
# are reflected with respect to each other

library(ca)
library(ellipse)
TheData <- DreamData
Results <- cabootcrs(TheData, showresults=FALSE)
caResults <- ca(TheData)
plot(caResults)
for (i in 1:dim(TheData)[1]) {
  lines( ellipse(x=covmat(Results,i,"row",1,2,FALSE),
                 centre=Results@Rowprinccoord[i,cbind(1,2)], npoints=1000),
        cex=1, pch=".", col="blue")
```

```
}

## End(Not run)
```

`DreamData` *Maxwell's dream data set, with simplified labels*

Description

Reported disturbance of dreams among boys, using labels A, a etc

Usage

```
DreamData
```

Format

A contingency table with 5 rows and 4 columns

rows Age group: 5-7 (A), 8-9 (B), 10-11 (C), 12-13 (D), 13-14 (E)

columns Severity of disturbance of dream: lowest (a) to highest (d)

Source

G. Iliopoulos, M. Kateri and I. Ntzoufras, *Bayesian estimation of unrestricted and order-restricted association models for a two-way contingency table*, Computational Statistics and Data Analysis 51 (2007), pp. 4643–4655.

`DreamData223by3` *Maxwell's dream data set with added totally random column*

Description

Reported disturbance of dreams among boys, plus a random column

Usage

```
DreamData223by3
```

Format

A matrix of 223 individuals by 3 variables

Age group - R 5-7 (A), 8-9 (B), 10-11 (C), 12-13 (D), 13-14 (E)

Severity of disturbance of dream - C Severity of disturbance of dream: lowest (a) to highest (d)

Random fake variable - V 1 to 3

Source

Adapted from G. Iliopoulos, M. Kateri and I. Ntzoufras, *Bayesian estimation of unrestricted and order-restricted association models for a two-way contingency table*, Computational Statistics and Data Analysis 51 (2007), pp. 4643–4655.

`DreamDataNames` *Maxwell's dream data set, using full original labels*

Description

Reported disturbance of dreams among boys, using ages and original code for severities

Usage

```
DreamDataNames
```

Format

A contingency table with 5 rows and 4 columns

rows Age group: 5-7 (A), 8-9 (B), 10-11 (C), 12-13 (D), 13-14 (E)

columns Severity of disturbance of dream: lowest (1) to highest (4)

Source

G. Iliopoulos, M. Kateri and I. Ntzoufras, *Bayesian estimation of unrestricted and order-restricted association models for a two-way contingency table*, Computational Statistics and Data Analysis 51 (2007), pp. 4643–4655.

`getBurt` *Converting a data matrix into a Burt matrix*

Description

`getBurt` recodes a data matrix from one format (n objects by p variables, counts for distinct combinations of p variables, contingency table) into a Burt matrix, or extracts a subset of a Burt matrix for selected variables

Usage

```
getBurt(
  Xinput,
  input = "nbyp",
  Jk = NULL,
  maxcat = NULL,
  varandcat = TRUE,
  vars = NULL
)
```

Arguments

Xinput	A data matrix, in the form of a data frame or similar
input	The format of the input matrix: **"nbyp"** An n individuals/objects/data points by p categorical variables matrix, where each row is a different data point and each column contains the category for that data point on that variable, where these categories can be numbers, strings or factors **"nbypcounts"** Similar to the above, but each row represents all of the data points taking the same combination of categories, and the first column contains the count for this combination (hence the name used here is a bit of a misnomer, but it emphasises the similarities to an n by p) **"indicator"** An indicator matrix, similar to the n by p matrix except that a variable with J_k categories is represented by J_k columns and a data point taking the i-th category has 1 in the i-th of these columns and a zero in the others **"Burt"** A Burt matrix, symmetrical and block-diagonal with each block being a contingency table for a pair of variables
Jk	See convert
maxcat	See convert
varandcat	See convert
vars	A list of the variable numbers to be used in the Burt matrix, if only a subset is wanted. If not all variables are wanted and the input is not n by p then Jk must be specified.

Value

A Burt matrix, symmetrical and block-diagonal with each block being a contingency table for a pair of variables

See Also

Other conversion functions: convert(), getCT(), getdoubled(), getindicator()

Examples

```
nishburt <- getBurt(NishData)
nishburtvars1to3 <- getBurt(NishData,vars=1:3)
nishburtvars2and4 <- getBurt(nishburt,input="Burt",Jk=rep(3,4),vars=c(2,4))
```

getCT *Converting a data matrix into a contingency table*

Description

getCT recodes a data matrix from one format (n objects by p variables, counts for distinct combinations of p variables, indicator matrix or Burt matrix) into a contingency table, for cases where table() doesn't work

Usage

```
getCT(
  Xinput,
  input = "nbyp",
  Jk = NULL,
  maxcat = NULL,
  varandcat = TRUE,
  vars = NULL
)
```

Arguments

Xinput	A data matrix, in the form of a data frame or similar
input	See getBurt
Jk	See convert
maxcat	See convert
varandcat	See convert
vars	A list of the variable numbers to be used in the contingency table when there are more than 2. If not all variables are wanted and the input is not n by p then Jk must be specified.

Value

A contingency table, giving counts for the two cross-classified variables

See Also

Other conversion functions: convert(), getBurt(), getdoubled(), getindicator()

Examples

```
nishCTvars23 <- getCT(NishData,Jk=rep(3,4),vars=2:3)
```

getdoubled *Converting a data matrix into a doubled matrix*

Description

getdoubled recodes a data matrix from one format (n objects by p variables, counts for distinct combinations of p variables, contingency table) into a doubled matrix

Usage

```
getdoubled(Xinput, input = "nbyp", Jk = NULL, maxcat = NULL)
```

Arguments

Xinput	A data matrix, in the form of a data frame or similar, all variables must be ordered categorical with numerical categories
input	See convert
Jk	See convert
maxcat	See convert

Value

A doubled matrix, where each variable is represented by 2 columns, and a data point taking the i-th category for a variable with J_k categories is given the values J_k-i in the first (low) column and i-1 in the second (high) column

See Also

Other conversion functions: convert(), getBurt(), getCT(), getindicator()

Examples

```
nishdoubled <- getdoubled(NishData)
```

getindicator *Converting a data matrix into an indicator matrix*

Description

getindicator recodes a data matrix from one format (n objects by p variables, counts for distinct combinations of p variables, contingency table) into an indicator matrix

Usage

```
getindicator(
  Xinput,
  input = "nbyp",
  Jk = NULL,
  maxcat = NULL,
  varandcat = TRUE
)
```

Arguments

Xinput	A data matrix, in the form of a data frame or similar
input	See convert
Jk	See convert
maxcat	See convert
varandcat	See convert

Value

An indicator matrix, where a variable with J_k categories is represented by J_k columns and a data point taking the i-th category has 1 in the i-th of these columns and a zero in the others

See Also

Other conversion functions: convert(), getBurt(), getCT(), getdoubled()

Examples

```
nishindicator <- getindicator(NishData)
```

---

myresamplefn *Example of a user-generated resampling routine.*

Description

myresamplefn in this case assumes that each pair of cells represents 50 people answering yes or no to a question, with undecideds not recorded

Usage

```
myresamplefn(X)
```

Arguments

X	a data matrix, to be resampled from

Details

This is only intended as an example of a user-generated resampling routine, users should replace it with their own function of this name

In this example we assume that rows groups of 50 people have each been asked columns/2 questions, with possible answers yes/no/undecided. It uses binomial bootstrapping for pairs of columns, assuming that successive columns are "yes" and "no" answers, with others undecided, with sums of pairs of columns having a maximum, in this case 50.

Value

a resampled version of the input data matrix

See Also

cabootcrs-package, cabootcrs

Examples

```
# Five groups of people answer two yes/no/undecided questions
# Note: this is just an example, and does not intend to claim that this
# is the correct analysis for such a data set

x <- as.matrix( rbind( c(22,25,18,22), c(12,23,21,27),
                       c(31,12,28,22), c(29,14,35,11), c(7,31,12,21)))
xresampled <- myresamplefn(x)
bmr <- cabootcrs(x,"myresample",nboots=199)
```

NishData *Nishisato's Singapore data*

Description

Questionnaire data collected at a Dual Scaling workshop in Singapore

Usage

```
NishData
```

Format

A matrix of 13 individuals by 4 variables each with 3 possible answers

Age - a 20-29 (1), 30-39 (2), 40+ (3)

Children are less disciplined now - b Agree (1), Disagree (2), Can't tell (3)

Children are less fortunate now - c Agree (1), Disagree (2), Can't tell (3)

Religions should not be taught in school - d Agree (1), Disagree (2), Indifferent (3)

Source

Nishisato, S. (1994). *Elements of Dual Scaling: An Introduction to Practical Data Analysis*. Lawrence Erlbaum Associates, New Jersey. (p153)

`OsteoData` *Osteoarchaeological data with categories given as numbers*

Description

Animal bones classified by 11 different variables

Usage

```
OsteoData
```

Format

A matrix of 6027 bones by 11 variables, each with 2-12 categories

Species/Taxon - Taxa 1-10

Anatomical element - Elem 1-12

Recent breakage - Rec P/A

Cut marks - Cut P/A

Gnawing - Gnaw P/A

Weathering stage - Weat 1-4 or A

Thermal stage - Therm 1-5 or A

Trampling - Tram P/A

Root etching - Root P/A

Post-depositional - Post P/A

Context - Context 1-4

Source

Macheridis, S. *The Use of Multiple Correspondence Analysis (MCA) in Taphonomy: The Case of Middle Helladic Asine, Greece* International Journal of Osteoarchaeology 27 (2017), pp. 477—487.

OsteoDataNames *Osteoarchaeological data with named categories*

Description

Animal bones classified by 11 different variables

Usage

```
OsteoDataNames
```

Format

A matrix of 6027 bones by 11 variables, each with 2-12 categories

Species/Taxon - Taxa Sheep/goat,Pig,Cattle,Deer,Dog,Equid,Large,Medium,Small,Size:indet

Anatomical element - Elem Horn/antler,Head,Neck,Axial,Upper_front,Lower_front,Upper_hind,Lower_hind,Feet,Metapo

Recent breakage - Rec P/A

Cut marks - Cut P/A

Gnawing - Gnaw P/A

Weathering stage - Weat stage 1-4 or A

Thermal stage - Therm stage 1-5 or A

Trampling - Tram P/A

Root etching - Root P/A

Post-depositional - Post P/A

Context - Context Secondary,Room_fills,Primary,Floors

Source

Macheridis, S. *The Use of Multiple Correspondence Analysis (MCA) in Taphonomy: The Case of Middle Helladic Asine, Greece* International Journal of Osteoarchaeology 27 (2017), pp. 477—487.

plotca *Plotting results with confidence regions*

Description

`plotca` produces one or more scatterplots of the results of simple or multiple correspondence analysis, with elliptical confidence regions around chosen points.

Usage

```
plotca(
  x,
  datasetname = NULL,
  mytitles = NULL,
  showrowlabels = TRUE,
  showcolumnlabels = TRUE,
  plotsymbolscolours = c(19, "inferno", 18, "inferno"),
  othersmonochrome = "grey",
  crpercent = 95,
  usebootcrits = NULL,
  plottype = "biplot",
  showrowcrs = TRUE,
  showcolumncrs = TRUE,
  likertarrows = FALSE,
  firstaxis = 1,
  lastaxis = 2,
  plotallpairs = "successive",
  picsize = NULL,
  mcaoneploteach = TRUE,
  mcashowindividuals = FALSE,
  mcavariablecolours = FALSE,
  mcacategorycolours = FALSE,
  groupings = NULL,
  grouplabels = NULL,
  eps = 1e-15,
  plotwithdevnew = FALSE
)
```

Arguments

x	An object of class cabootcrsresults
datasetname	A string to use as the name of the data set in the plots, defaults to that in cabootcrs object
mytitles	A list of text strings, to be used instead of the default titles of the plots, where the list must be at least as long as the number of plots to be produced
showrowlabels	If TRUE then label row points as usual, otherwise suppress labels of row points. Note: when analysing a Burt matrix the columns points are plotted
showcolumnlabels	If TRUE then label column points as usual, otherwise suppress labels of column points. Note: when analysing a Burt matrix the columns points are plotted
plotsymbolscolours	A vector/list of length 1, 2, 4 or equal to the number of category points to be plotted. (a) If longer than length 4 then it contains the colours for all the points, their ellipses and labels.

Taken to be a vector or list of valid R colours, length at least equal to the number of category points to be plotted, in the order rows followed by columns for sca or variable categories in order for mca. The colours can be named (e.g. "azure2") or RGB hexadecimal (e.g. "#1173B3"), and can be subsetted from colours() or from the colo(u)rspace library's palettes.

(b) If length 4 then it takes the form:

c(row symbol, "row colour", column symbol, "column colour")

giving plot symbols and colours for row and column points and ellipses when they are the primary points.

First element: The plot symbol for row points - either an R number code (usually 0-25), or a character (in " ")

Second element: The colour for row points, their ellipses and labels - either a valid R colour name (in " ") or
"differentreds" - each row is plotted with a different shade of red to green, using the grDevices rainbow palette
"differentblues" - each row is plotted with a different shade of green to blue, using the grDevices rainbow palette
"alldifferent" - each row is plotted with a different colour, using the grDevices rainbow palette from reds to blues
"viridis" - each row is plotted with a different colour, using the grDevices viridis palette
"inferno" - each row is plotted with a different colour, using the grDevices inferno palette, though not the extremes of it

Third element: As first element but for column points

Fourth element: As second element but for column points

(c1) If length 2 and the first element is a valid colour choice as above then it takes the form:

c("row colour", "column colour")

giving colours for row and column points and ellipses when they are the primary points.

First element: The colour for row points, choices as second element above

Second element: The colour for column points, choices as second element above

(c2) If length 2 and the first element is *not* a valid colour choice as above then it takes the form:

c(column symbol, "column colour")

giving plot symbols and colours for column points and ellipses when they are the primary points.

First element: The plot symbol for column points, choices as first element above

Second element: The colour for column points, choices as second element above

(d) If length 1 then it takes the form:

"column colour"

giving colours for column points and ellipses when they are the primary points, choices as second element above.

Note: colour can also be specified as colours()[i] which picks out the i-th colour.

Note: when analysing a Burt matrix the columns points are plotted.

The idea behind "alldifferent" etc is that the colours change gradually, so that if the order of the rows/columns is meaningful then the colour change tracks this.

Note: groupings and grouplabels below override all this.

`othersmonochrome`

Either:

NULL secondary points plotted using same colours as when they are primary points

a valid R colour (in " ") all secondary points are plotted in this colour

`crpercent` The nominal coverage percentage of the confidence ellipses (90, 95 or 99 only if using bootstrap critical values)

`usebootcrits` Whether to use bootstrap critical values for the ellipses:

TRUE use bootstrap critical values

FALSE use χ^2 critical values

NULL inherit this choice from the chosen object of class `cabootcrsresults`

Note: only 90%, 95% and 99% bootstrap critical values are available

`plottype` This is only relevant for simple CA, in MCA the choice is automatic.

"biplot" One plot with confidence regions for rows in principal coordinates while columns are shown as directions in standard coordinates, another plot with confidence regions for columns in principal coordinates while rows are shown as directions in standard coordinates

"french" Two plots each with both rows and columns in principal coordinates, one plot shows confidence regions for the rows and the other shows confidence regions for the columns

`showrowcrs` Whether to plot confidence ellipses for row points:

TRUE plot all confidence ellipses for row points as usual

A row number or vector of row numbers plot confidence ellipses only for these row points, which can be specified in any standard R format such as 7 or c(1,4,9) or 5:11

FALSE suppress plotting of all confidence ellipses for row points

Note: when analysing a Burt matrix the columns points are plotted

`showcolumncrs` Whether to plot confidence ellipses for column points:

TRUE plot all confidence ellipses for column points as usual

A column number or vector of column numbers plot confidence ellipses only for these column points, which can be specified in any standard R format such as 7 or c(1,4,9) or 5:11

FALSE suppress plotting of all confidence ellipses for column points

Note: when analysing a Burt or indicator matrix the column points are plotted, the columns being all of the variable categories, ordered by variable number, e.g. for p=3 variables each with 5 categories then columns 6:10 are variable 2.

Note: in MCA this is overridden by mcaoneploteach=TRUE, in which case there is one plot per variable, automatically giving ellipses for each of its categories.

`likertarrows` If TRUE then, for MCA on likert-type ordered categorical data, draw arrows connecting the category points for each variable, with the arrows drawn from a category point to the next higher category point.
Note: will only be drawn from category points with ellipses shown.

`firstaxis` Number of the first (i.e. highest inertia) axis to be plotted

`lastaxis` Number of the last (i.e. lowest inertia) axis to be plotted, which must be <= axisvariances value for x.

`plotallpairs` Whether to plot all pairs of axes against each other:

"onlythese" plot firstaxis v lastaxis only

"successive" plot all successive pairs of axes between firstaxis and lastaxis

"all" plot all possible pairs of axes between firstaxis and lastaxis

`picsize` A 2-vector/list or 4-vector/list specifying the plot size:

2-vector minimum and maximum of both x and y axes on each plot

4-vector min and max of x axis followed by min and max of y axis, where the difference between max and min must be the same in both cases

All plots have an aspect ratio of 1.

The same scales are used for all plots, so in the biplot case it might occasionally be preferred to run plotca twice with different picsize values, one being better for rows in principal coordinates and the other better for columns in principal coordinates.

If picsize is used to focus in on a particular area of the plot then biplot labels might not appear properly.

If using Rstudio then it may override this somewhat, especially if you resize the plot window after plotting.

If Rstudio has too much of a mind of its own when plotting then try plotwithdevnew=TRUE to put each plot in a new window, as in standard R.

`mcaoneploteach` For MCA only, a flag or list of column numbers saying whether to produce one plot for each variable, where confidence ellipses are shown for that variable but not others:

TRUE p plots are produced, each showing confidence regions for all of the category points for just one variable, overriding showcolumncrs

number or vector of numbers produce one plot for each of the specified variable numbers, as above

FALSE only one plot is produced, with confidence regions shown for each category point of each variable, although this can be controlled by showcolumncrs

`mcashowindividuals`

For MCA on an indicator matrix only, a flag saying whether to plot the individuals on the plot(s):

TRUE plot the individuals, which could be very "busy"

FALSE don't plot the individuals

`mcavariablecolours`

TRUE In MCA each variable has its own colour, and all category points and ellipses for that variable have the same colour

FALSE Colours chosen in the default way

If TRUE then the only valid colour options for plotsymbolscolours are "viridis", "inferno", "alldifferent", "differentreds" or "differentblues" as above.
Hence if TRUE then colours vary from variable 1 to variable p with each variable having all its categories plotted with the same colour, while if FALSE then colours vary from variable 1 category 1 to variable p category J_p.

Note: if plotsymbolscolours has length > 4, and so is specifying a colour for each category, then it overrides this.

`mcacategorycolours`

TRUE In MCA each category number has its own colour, and all points and ellipses for that category number have the same colour, for all variables (intended for Likert type data so that all category 1 points are the same colour etc)

FALSE Colours chosen in the default way

If TRUE then the only valid colour options for plotsymbolscolours are "viridis", "inferno", "alldifferent", "differentreds" or "differentblues" as above.
Hence if TRUE then colours vary from category 1 to category J_k with category i being the same colour for each variable, while if FALSE then colours vary from variable 1 category 1 to variable p category J_p.

Note: if plotsymbolscolours has length > 4, and so is specifying a colour for each category, then it overrides this.

`groupings` The name of a file (in " ") or data frame containing group structure of row and column points:

the n rows are divided into m groups and the p columns divided into k groups

the file or data frame is n+p by 2, where:

first column is 1..n 1..p (to make the file easier to read)
second column contains the number of the group-of-rows (1..m) or group-of-columns (1..k) that the row or column belongs to.

Hence the file or data frame is:

1 <the number of the group-of-rows to which row 1 belongs>
...
n <the number of the group-of-rows to which row n belongs>
1 <the number of the group-of-columns to which column 1 belongs>
...
p <the number of the group-of-columns to which column p belongs>

`grouplabels` The name of the file (in " ") or data frame containing the colours and labels to be used, in association with the groupings option above, in a m+k by 5 array:

1 <legend> <plot symbol> <plot colour> <draw ellipse?>
...
m <legend> <plot symbol> <plot colour> <draw ellipse?>
1 <legend> <plot symbol> <plot colour> <draw ellipse?>
...
k <legend> <plot symbol> <plot colour> <draw ellipse?>

The first column contains the number of the group-of-rows or group-of-columns, the others are:

legend (in " " if in data frame but not in file) for this group of rows/columns, to be shown on plot

symbol for all rows/columns in this group, either an R number code for a symbol or a character (the latter in " " if in a data frame but not if in a file) Note: both rows and columns need to be either all numbers or all characters, or the legend will not come out right

colour (in " ") for symbols and ellipses for this group

draw? T or F to draw or suppress ellipses and label points for this group (both in " " in data frame but not in file)

See Details section and examples below to make more sense of this.

This can also be used for multiple CA, remembering that only column points are shown (usually) and that columns are ordered by variable and then by category, so that three variables each with 5 categories will be columns 1:5, 6:10 and 11:15 respectively. It should not be used when mcaoneploteach=TRUE, however, as that already takes care of this sort of grouping.

These options are particularly intended for large data sets, to allow attention to be drawn to some points above others, to emphasize any group structure within the data, or to show only the most important ellipses in order to make the picture less cluttered.

`eps` Any value less than this is treated as zero in some calculations and comparisons

`plotwithdevnew` When using Rstudio, a flag saying whether to put each plot in a new device (plot window) or just use the default Rstudio plot pane:

TRUE Each plot gets its own new plot window, as in standard R outside Rstudio

FALSE All plots appear squashed into the usual Rstudio plot pane

Details

In the following, the categories for which confidence regions are being shown are referred to as the primary points, the others as the secondary points. The primary points are always plotted in principal coordinates while the secondary points can be in standard (biplot style in simple CA) or principal (french style in simple CA, always in multiple CA) coordinates.

The default colour scheme is for the primary points and their confidence ellipses to be plotted each in a different colour, as this makes it easier to see which ellipse goes with which point, while the secondary points are all plotted in monochrome to make it easier to distinguish between the two sets of points. This can all be controlled by the user. Note that a point will still be treated as a primary point and plotted with its own colour even when the plotting of its ellipse is suppressed with the showrowcrs or showcolumncrs options.

Note that the plots will look better if saved as .eps or .pdf rather than as .jpg or .png.

(1) Simple CA

Two plots are produced, in each plot one set of points (rows or columns) is regarded as the primary set and is plotted in principal coordinates with confidence regions shown:
- one plot shows confidence regions for rows in principal coordinates
- one plot shows confidence regions for columns in principal coordinates

The other set of points (columns or rows) is regarded as the secondary set and the plotting depends on the choice of biplot or french-style plot:

biplot - secondary points shown as directions in standard coordinates
french - secondary points shown in principal coordinates

(2) Multiple CA

All points are plotted in principal coordinates ("french")

Burt matrix (mcatype="Burt"):

a) only plot the columns of the Burt matrix (the rows are the same)
b) plot all variable categories, i.e. columns
c) if mcaoneploteach=TRUE then produce p plots, each with CRs for all categories of one of the variables,
otherwise produce one plot showing CRs for all variables (busy)
d) Columns are ordered by variable then category (e.g. for p=3 with 5 categories each, columns 6:10 are variable 2)

Indicator matrix (mcatype="indicator"):

a) if mcashowindividuals=TRUE then plot individual (row) points, without CRs
b) as Burt for variables

Indicator matrix (mcatype="indicator") with experimental likert resampling (cabootcrs had mcaindividualboot=TRUE):

a) if mcashowindividuals=TRUE then plot individual (row) points, with CRs (busy)
b) as Burt for variables

(3) Critical values

Critical values for the ellipses default to those specified in cabootcrs, which default to bootstrap critical values

(4) Choosing colours and which ellipses to show

The showrowcrs, showcolumncrs, showrowlabels, showcolumnlabels and othersmonochrome options are available as ways of reducing plot clutter in large data sets, for example by showing the column points unlabelled and monochrome as a way of drawing the eye to the multicoloured row points and ellipses.

The default is for each primary point to be in a different colour, with secondary points in the colour defined by othersmonochrome (default grey). If othersmonochrome=NULL then secondary points are also plotted with different colours.

The plotsymbolscolours option can be used to specify quickly the set of symbols and colours used, with the options described in (5) below giving far more control at the cost of extra work.

The default colour scheme is grDevices:inferno, but with k+2 colours picked and then 2:k+1 used, because the end colour is a bit too yellow and hard to see.

Note that french-style plots in simple CA are often less cluttered because they omit the biplot lines, while they also show the two sets of points on similar scales so that it is easier to fit all the points on one picture without cropping or excessive empty space.

(5) Specifying colours for (groups of) points and ellipses

For large matrices the plots from exploratory multivariate methods are often so busy that the whole point of the method, to clarify the structure of the data, is nullified. This is even more of a problem when confidence regions are shown on the plots.

Hence points can be defined in groups as below, so you can divide them into groups in one or more ways, e.g. rows 1-3 in red and rows 4-8 in blue, or rows 1-5 in green and rows 6-8 in orange etc.

The groupings and grouplabels options are chosen via separate text files or data frames to define the groups of points. If groupings is left null then plotsymbolscolours is used instead.

There are two ways of defining groupings and group labels. The first of these is by defining a pair of data frames within R and supplying them as parameters either to cabootcrs initially or to plotca. This method works in R CMD check and hence is the one used in the examples, but as you can see is rather hard to follow.

To plot with colours defined using groups-of-points:

bd <- cabootcrs(DreamData)

Then define the groups using data frames in R or text files:

(5a) Using data frames

These data frames define the same groupings and colours as the files below, see the files for a clearer explanation:

```
bd <- cabootcrs(DreamData)
groupingsframe <- cbind(c(1:5,1:4),c(1,1,2,2,3,1,1,2,2))
grouplabframe <- cbind( c(1,2,3,1,2), c("AB","CD","E","ab","cd"), c(19,20,21,"+","*"), c("green","blu
plotca(bd, groupings=groupingsframe, grouplabels=grouplabframe)
```

(5b) Using text files

A version which produces identical results, but does not work in R CMD check, is usually much easier for the user as they can be edited outside R. The groupings and group labels are defined in files, present in the directory specified in setwd(). To obtain identical results to the above, create two text files as below:

DreamGroupings.txt contains

```
1 1
2 1
3 2
4 2
5 3
1 1
2 1
3 2
4 2
```

e.g. the first two lines show that rows 1,2 belong to group-of-rows 1, while the last two lines show that columns 3,4 belong to group-of-columns 2.

DreamGroupLabels.txt contains

```
1 AB 19  "green"  T
2 CD 20  "blue"   T
3  E 21  "yellow" T
1 ab  +  "red"    T
2 cd  *  "orange"  T
```

e.g. group-of-rows 1 will be shown in green and plotted with symbol 19, with the legend AB.

```
bd <- cabootcrs(DreamData, showresults=FALSE)
plotca(bd, groupings="DreamGroupings.txt", grouplabels="DreamGroupLabels.txt")
```

(5c) General use

Even without groupings this can be used to specify all colours, simply by specifying each point as its own group, in this case rows 1-5 and columns 1-4 define row groups 1-5 and column groups 1-4, no legend is required so repeat "", choose 9 plot symbols and 9 colours.

Hence to plot each point with its own specified colour and symbol:

```
bd <- cabootcrs(DreamData, showresults=FALSE)
groupingsframe <- cbind(c(1:5,1:4),c(1:5,1:4))
grouplabframe <- cbind( c(1:5,1:4), rep("",9), 11:19, c("green","blue","yellow","red","orange","grey1
plotca(bd, groupings=groupingsframe, grouplabels=grouplabframe)
```

Note: plotsymbolscolours can be used to plot with a different colour for each category point, but with default symbols.

(5d) MCA use

As before, but need to specify for both row and column categories even though only column categories will be plotted, so just duplicate the data frames (yes I know it's a bodge).
Hence to plot each point with your own choice of colour and symbol:

```
bd3 <- cabootcrs(DreamData223by3, catype="mca", varandcat=FALSE)
groupingsframe <- cbind(1:12,1:12)
groupingsframe <- rbind(groupingsframe,groupingsframe)
grouplabframe <- cbind( 1:12, rep("",12), 11:22, c("green","blue","yellow","red","orange","grey1","gr
grouplabframe <- rbind(grouplabframe,grouplabframe)
plotca(bd3, groupings=groupingsframe, grouplabels=grouplabframe)
```

This can also be used to specify colours and symbols for each variable in MCA, but note that you are still specifying for each column point, so that you need to know how many categories each variable has - a simpler way to do this will be added to a later update.

```
ost <- cabootcrs(OsteoDataNames, catype="mca", varandcat=FALSE)
totcolumns <- ost@columns
totvars <- ost@p
numcats <- ost@Jk
cats <- NULL
for (i in 1:totvars) { cats <- c(cats,rep(i,numcats[i])) }
groupingsframe <- cbind(1:totcolumns,cats)
groupingsframe <- rbind(groupingsframe,groupingsframe)
grouplabframe <- cbind( 1:totvars, rep("",totvars), rep(19,totvars), c("blue","red","green","darkgree
grouplabframe <- rbind(grouplabframe,grouplabframe)
plotca(ost, groupings=groupingsframe, grouplabels=grouplabframe)
```

This can also be used to plot the individual points with colours to denote groups, for example in the below the first 100 individual points are plotted with one colour, the rest with another, while all columns get their own colour.

```
bd3indnoboot <- cabootcrs(DreamData223by3,catype="mca",mcatype="indicator",varandcat=FALSE,nboots=0)
rowgroups <- cbind( 1:223, c(rep(1,100),rep(2,123)) )
colgroups <- cbind(1:12,1:12)
groupingsframe <- rbind( rowgroups, colgroups )
rowlabs <- cbind( 1:2, c("1-100","101-223"), c("+","+"), c("black","grey"), "T")
collabs <- cbind( 1:12, rep("",12), "*", c("green","cyan","yellow","red","orange","blue","blue1","blu
grouplabframe <- rbind(rowlabs,collabs)
plotca(bd3indnoboot, groupings=groupingsframe, grouplabels=grouplabframe, mcashowindividuals=TRUE, mc
```

(6) Plotting results from cabootcrs() using ellipse() or ca()

This can be used with the ellipse() package to add the confidence ellipse to a picture from another package

Example: confidence ellipse for row or column i on axes 1,2 from cabootcrs output Results is:

```
lines(ellipse(x=covmat(Results,i,"row",1,2,FALSE), centre=Results@Rowprinccoord[i,cbind(1,2)], npoin
lines(ellipse(x=covmat(Results,i,"column",1,2,FALSE), centre=Results@Colprinccoord[i,cbind(1,2)], np
```

Example: to add row CRs to a plot from the ca() package to data set TheData

```
Results <- cabootcrs(TheData, showresults=FALSE)
caResults <- ca(TheData)
plot(caResults)
for (i in 1:dim(TheData)[1]) {
 lines(ellipse(x=covmat(Results,i,"row",1,2,FALSE), centre=Results@Rowprinccoord[i,cbind(1,2)], npoi
}
```

However note that reflectaxes() may also be needed.

(7) Note

Note that plotca, summaryca and printca are all defined as new functions, rather than as overloaded versions of plot, summary and print, simply in order to avoid complication and unintended consequences within R

Value

One or more plots are produced but no output object is created

See Also

cabootcrs-package, cabootcrs, printca, summaryca, cabootcrsresults

Examples

```
# the main function also calls plotca with the default options

bd <- cabootcrs(DreamDataNames, datasetname="Maxwell's dream data",
                varnames=c("Age groups","Severity of disturbance"),showresults=FALSE)
plotca(bd)

## Not run:

### Plot options for SCA:

# Note that Rstudio changes plots depending on the size of your plot window,
# so the picsize parameter (used for xlim, ylim in the plot command) is partially
# overridden, so warnings that a point is outside the plot limits may not be correct

# Plot with specified size to fit the whole of the arrows in without cropping

plotca(bd, picsize=c(-2.5,2.5))

# or smaller, note the warning

plotca(bd, picsize=c(-0.5,0.5))

# Replacing the plot titles with your own

plotca(bd, mytitles=c("Plot 1 Title line 1\nline 2\nline 3","Plot 2 Title line 1\nline 2\nline 3" ))

# All points in colour

plotca(bd,othersmonochrome=NULL)

# 90% regions in reds and blue

plotca(bd, plotsymbolscolours=c(3,"differentreds","*","blue"), crpercent=90)

# Many different colour schemes and ways of specifying colours and symbols

# Specify colour/colour scheme and symbols
plotca(bd, plotsymbolscolours=c(3,"differentreds","*","blue") )
plotca(bd, plotsymbolscolours=c(3,"viridis") )
plotca(bd, plotsymbolscolours="inferno" )
plotca(bd, plotsymbolscolours=colours()[641] )

# Just give a list of colours, one for each category point
plotca(bd,
plotsymbolscolours=c("green","blue","yellow","red","orange","red","blue","tan1","orchid4") )
plotca(bd, plotsymbolscolours=colours()[161:170] )
plotca(bd, plotsymbolscolours=colours()[c(111:115,561:564)] )

# This time using colo(u)rspace package colour palettes
library(colorspace)
plotca(bd, plotsymbolscolours=hcl.colors(9,palette="Peach") )
```

```
plotca(bd, plotsymbolscolours=hcl.colors(50,palette="Mint")[c(11:15,31:34)] )
plotca(bd,
plotsymbolscolours=c(sequential_hcl(bd@rows,"Blues 3"),sequential_hcl(bd@columns,"Reds 3")) )

# suppress labels for column points, to de-clutter row points picture,
# this is mostly useful for larger data sets than this one

plotca(bd, showcolumnlabels=FALSE)

# only show ellipses for rows 1, 1-2 and 1-3 respectively

plotca(bd, showrowcrs=1)
plotca(bd, showrowcrs=c(1,2))
plotca(bd, showrowcrs=1:3)

# plot axes 1 v 2, 1 v 3 and 2 v 3

plotca(bd, firstaxis=1, lastaxis=3, plotallpairs="all")

# If the cell values were all 10 times larger

bdx10 <- cabootcrs(10*DreamData)
plotca(bdx10,plottype = "french",picsize=c(-0.4,0.4))

# Various plots for a larger data set, note that the default colour scheme picks out
# males, females and ages because of the ordering of the rows

bs <- cabootcrs(SuicideData)
plotca(bs, picsize=c(-0.7,0.8))
plotca(bs, plottype="french", picsize=c(-0.7,0.8))
plotca(bs, plottype="french", picsize=c(-0.7,0.8),
           plotsymbolscolours=c(".","inferno","+","black"))

# Note that the ellipses follow the horseshoe

bas <- cabootcrs(AsbestosData)

# more complicated plotting, define group structure in data frames

groupingsframe <- cbind( c(1:5,1:4), c(1,1,2,2,3,1,1,2,2) )
grouplabframe <- cbind( c(1,2,3,1,2), c("AB","CD","E","ab","cd"), c(19,20,21,"+","*"),
                        c("green","blue","yellow","red","orange"), "T" )
plotca(bd, groupings=groupingsframe, grouplabels=grouplabframe)

plotca(bd, groupings=groupingsframe, grouplabels=grouplabframe, plottype="french")

# This can also be used for custom colour schemes other than "differentreds" etc as
# defined in the plotsymbolscolours option, though note that R colours are not ordered in the
# way you might expect, so the colour scheme below is purely illustrative and not very sensible

customframe <- cbind( c(1:5,1:4), c(1:5,1:4) )
customlabframe <- cbind( c(1:5,1:4), rep("",9), c(rep(18,5),rep(19,4)),
                         colours()[c(seq(10,130,30),seq(440,590,50))], "T" )
```

```
plotca(bd, groupings=customframe, grouplabels=customlabframe)


### Plot options for MCA:

# Use one of the below, labelling row A as R:A or just A (etc) as preferred

bd3 <- cabootcrs(DreamData223by3, catype="mca",
                 datasetname="Dream data with extra random column")
bd3 <- cabootcrs(DreamData223by3, catype="mca", varandcat=FALSE,
                 datasetname="Dream data with extra random column")

# just variable 2

plotca(bd3,mcaoneploteach=2)

# just variables 1 and 3

plotca(bd3,mcaoneploteach=c(1,3))

# one plot showing CRs for all variable categories (busy)

plotca(bd3,mcaoneploteach=FALSE)

# each variable has its own colour

plotca(bd3,mcavariablecolours=TRUE)

# each category number has its own colour

plotca(bd3,mcacategorycolours=TRUE)

# draw arrows between successive ordered categories

plotca(bd3,likertarrows=TRUE)

# secondary points black rather than grey

plotca(bd3,othersmonochrome="black")

# 99% CRs

plotca(bd3,crpercent=99)

# Plot together CRs for the first category of each variable

plotca(bd3,showcolumncrs=c(1,6,10),mcaoneploteach=FALSE)

# Plot together CRs for the second category of each variable

plotca(bd3,showcolumncrs=c(2,7,11),mcaoneploteach=FALSE)
```

```
# One plot with CRs only for variable 3

plotca(bd3,showcolumncrs=10:12,mcaoneploteach=FALSE)

# Three plots, various colour schemes

plotca(bd3,othersmonochrome="black")
plotca(bd3,othersmonochrome="black",mcacategorycolours=TRUE)
plotca(bd3,mcavariablecolours=TRUE,likertarrows=TRUE)

# All on one plot, various colour schemes, very busy

plotca(bd3,mcaoneploteach=FALSE,showcolumncrs=1:5,othersmonochrome="black")
plotca(bd3,mcaoneploteach=FALSE,showcolumncrs=1:5,likertarrows=TRUE)
plotca(bd3,mcaoneploteach=FALSE,likertarrows=TRUE,mcacategorycolours=TRUE)
plotca(bd3,mcaoneploteach=FALSE,likertarrows=TRUE,mcavariablecolours=TRUE)

# Plots with more complicated colour and grouping structure, as above but now in MCA case.
# Note the need to duplicate both data frames as groupings must be specified for both rows
# and columns, though only columns are used.
# Note also that symbol types need to be defined either all as numbers or all as symbols

groupingsframe <- cbind(1:12,c(1,1,2,2,3,4,4,5,5,6,7,7))
groupingsframe <- rbind(groupingsframe,groupingsframe)
grouplabframe <- cbind( 1:7, c("AB","CD","E","ab","cd","v1","v23"), 19:25,
                    c("cyan","deepskyblue","blue","red","tomato","chartreuse","green"),
                       "T" )
grouplabframe <- rbind(grouplabframe,grouplabframe)
plotca(bd3, groupings=groupingsframe, grouplabels=grouplabframe,
       mcaoneploteach=FALSE)


### Adding confidence ellipses to plots from ca package ca() and mjca() functions

## Simple CA

# Adding confidence ellipses for row points to plots from ca() using ellipse()
# Note: reflectaxes() is needed if cabootcrs() and ca() axes are reflected wrt each other

library(ca)
library(ellipse)
cad <- ca(DreamData)
plot(cad)
for (i in 1:dim(DreamData)[1]) {
  lines( ellipse(x=covmat(bd,i,"row",1,2,FALSE), centre=bd@Rowprinccoord[i,cbind(1,2)],
                 npoints=1000),
         cex=1, pch=".", col="blue")
}

# These plots can also be produced almost identically here

bd <- cabootcrs(DreamData, showresults=FALSE)
```

```
# both plots almost the same as the default plot from ca()

plotca(bd, plottype="french", showrowcrs=FALSE, showcolumncrs=FALSE, othersmonochrome=NULL,
       plotsymbolscolours=c(19,"blue",17,"red"), picsize=c(-0.5,0.6) )

# plot almost the same as the ca() plot, but with ellipses added

plotca(bd, plottype="french", othersmonochrome=NULL, plotsymbolscolours=c(19,"blue",17,"red"),
       picsize=c(-0.5,0.6))


## Multiple CA

# Adding confidence ellipses for category points to plots from mjca() using ellipse()
# Note that ca also uses standardised inertias and coordinates by default

library(ca)
library(ellipse)
cad3 <- mjca(DreamData223by3)

# Obtain covariance matrices, using same scalings and standardisations

bd3 <- cabootcrs(DreamData223by3, catype="mca", showresults=FALSE)

# Reflect axis 1 for consistency (may differ on other machines)

bd3 <- reflectaxes(bd3,1)

# Plot and then add ellipses for categories of variable 1 only

plot(cad3)
for (i in 1:bd3@Jk[1]) {
 lines( ellipse(x=covmat(bd3,i,"column",1,2,FALSE), centre=bd3@Colprinccoord[i,cbind(1,2)],
                npoints=1000),
        cex=1, pch=".", col="red")
}

# These plots can also be produced almost identically here

plotca(bd3,picsize=c(-0.35,0.35), mcaoneploteach=FALSE, mcavariablecolours=TRUE,
           showcolumncrs=FALSE)
plotca(bd3,picsize=c(-0.35,0.35), mcaoneploteach=FALSE, plotsymbolscolours=c(17,"red"),
           showcolumncrs=FALSE)
plotca(bd3,picsize=c(-0.35,0.35), mcaoneploteach=FALSE, mcacategorycolours=TRUE,
           showcolumncrs=1:5 )

# Three separate plots with ellipses for one variable on each

for (j in 1:length(bd3@Jk)) {
  plot(cad3)
  if (j==1) { firstcol <- 1 } else { firstcol <- cumsum(bd3@Jk)[j-1]+1 }
  for (i in firstcol:cumsum(bd3@Jk)[j]) {
   lines( ellipse(x=covmat(bd3,i,"column",1,2,FALSE), centre=bd3@Colprinccoord[i,cbind(1,2)],
```

```
                    npoints=1000),
            cex=1, pch=".", col="red")
  }
}

# For comparison, default plot with one plot each showing the ellipses for each variable

plotca(bd3,picsize=c(-0.35,0.35), mcacategorycolours=TRUE )


## End(Not run)
```

---

printca *Prints reasonably full results, including variances*

Description

printca prints full correspondence analysis results, including inertias, coordinates, representations, contributions, variances, covariances and critical values

Usage

```
printca(x, datasetname = NULL)
```

Arguments

x	An object of class cabootcrsresults
datasetname	The name (in "") of the data set, to be used in the output, defaults to name in cabootcrs object

Value

Printed results, no plots or objects produced

See Also

cabootcrs-package, cabootcrs, summaryca, plotca, cabootcrsresults

Examples

```
results <- cabootcrs(DreamData, showresults=FALSE)
printca(results, datasetname="Dreams")
```

`rearrange` *Rearranges bootstrap axes by comparing to sample axes*

Description

`rearrange` compares one set of axes for row points and column points (from the bootstrap data matrix) to another (from the sample data matrix) by looking at all possible reorderings and reflections (only) of the bootstrap axes and picking the one which best matches the sample axes.

Usage

```
rearrange(
  RS,
  RB,
  CS,
  CB,
  r,
  reflectonly = FALSE,
  catype = "sca",
  mcatype = "Burt",
  mcaindividualboot = FALSE,
  maxrearrange = 6
)
```

Arguments

`RS`	Sample axes for row points (as columns)
`RB`	Bootstrap axes for row points (as columns)
`CS`	Sample axes for column points (as columns)
`CB`	Bootstrap axes for column points (as columns)
`r`	Rank of the bootstrap matrix
`reflectonly`	TRUE to reflect the axes only, no reordering
`catype`	Can be "sca" for simple or "mca" for multiple CA. If "sca" then the rearranging will use both row axes and column axes, if "mca" then this depends on the mcatype parameter
`mcatype`	"Burt" if using Burt matrix rows and columns are the same, so only use column axes "indicator" if using indicator matrix, only use row axes
`mcaindividualboot`	TRUE to use highly experimental method
`maxrearrange`	Maximum number of axes to rearrange

Details

This is only intended for internal use by the cabootcrs function.

Finds the rearrangement of columns of RB and CB to maximise match = tr(abs(RS'*RB + CS'*CB))

Uses the Hungarian algorithm via lp.assign in lpSolve up to a maximum of maxrearrange vectors.

Algorithm assigns columns (B) to rows (S), hence transpose matrix so postmultiplication moves B to coincide with S.

In effect this is Procrustes rotation of bootstrap axes to best match sample axes, except that there is no rotation, only reflection and reordering of axes (aka rearranging).

Note that this seeks the best fit to all axes, not best fit just to the ones whose variances are being calculated, and does not weight the reordering by eigenvalues or restrict how far a vector can be reordered by. Hence a fairly low maxrearrange may be preferable.

Faster than full comparison when rank >= 4, for maxrearrange=6, but can take much longer if rearrange all axes

Rearranging more axes means higher chance of finding a matching axis, so std dev can be decreased by average of 1-2% if all axes are rearranged.

Limited testing suggests that rearranging all axes tends to over-reorder and hence underestimate variances, due to ignoring eigenvalues, hence seems best to rearrange 6 as before, unless very large numbers of close eigenvalues.

When mca bootstrap replicate has fewer "real" singular values (i.e. > 1/p) than the sample matrix then only the first B@realr axes will be compared, so that the last sample axis will get nothing from this replicate and the "real" bootstrap axes will be matched only with the same first few sample ones.

r = rank of bootstrap matrix, so if < sample rank will ignore last sample axis

Value

list containing:
T = matrix to rearrange xB so it is equivalent to xS, i.e. xS <- xB * T
numre = number of axes checked for rearranging = min(r,maxrearrange)
match = assign$objval from the Hungarian algorithm
same = flag for whether there was no reordering of axes (but may have been reflection)

See Also

cabootcrs-package, cabootcrs

Examples

```
# Not intended for direct call by users
```

`rearrange_old` *Old and rubbish algorithm to rearrange bootstrap axes by comparing to sample axes*

Description

`rearrange_old` compares one set of axes for row points and column points (from the bootstrap data matrix) to another (from the sample data matrix) by looking at all possible reorderings and reflections (only) of the bootstrap axes and picking the one which best matches the sample axes.

Usage

```
rearrange_old(RS, RB, CS, CB, r)
```

Arguments

`RS`	Sample axes for row points (as columns)
`RB`	Bootstrap axes for row points (as columns)
`CS`	Sample axes for column points (as columns)
`CB`	Bootstrap axes for column points (as columns)
`r`	Rank of the bootstrap matrix

Details

This is only intended for internal use by the `cabootcrs` function, and only for simple CA if for some reason the lpSolve package is unavailable.

It has not been used with MCA, and so will almost certainly not work properly in that case.

Finds the rearrangement of columns of RB and CB to maximise match = tr(abs(RS'*RB + CS'*CB))

Goes through all possible orderings and so is painfully slow.

Value

list containing:
T = matrix to rearrange xB so it is equivalent to xS, i.e. xS <- xB * T
numre = number of axes checked for rearranging = min(r,maxrearrange)
match = assign$objval from the Hungarian algorithm
same = flag for whether there was no reordering of axes (but may have been reflection)

See Also

`cabootcrs-package`, `cabootcrs`, `rearrange`

Examples

```
# Not intended for direct call by users
```

`reflectaxes` *Reflect coordinates for chosen axes*

Description

`reflectaxes` reflects the principal and standard coordinates of the axes chosen, and the appropriate covariances where needed

Usage

```
reflectaxes(x, axes = c(1, 2))
```

Arguments

`x`	An object of class `cabootcrsresults`
`axes`	A list or vector containing the numbers of the axes to be reflected

Details

This may be useful when comparing results between different data sets, or from different packages

Value

An object of class `cabootcrsresults`

See Also

`cabootcrs-package`, `cabootcrs`, `reordercategories`, `cabootcrsresults`

Examples

```
results <- cabootcrs(DreamData)
resultsreflectfirstaxis <- reflectaxes(results, 1)
summaryca(resultsreflectfirstaxis)
plotca(resultsreflectfirstaxis)

## Not run:

# Often needed when comparing results between different packages,
# or same package on different machines,
# or to allow ellipses from this package to be added to plots from other packages

library(ca)
cad3 <- mjca(DreamData223by3)
bd3 <- cabootcrs(DreamData223by3, catype="mca")
summary(cad3)
bd3reflect1 <- reflectaxes(bd3,1)
summaryca(bd3reflect1)
```

```
## End(Not run)
```

`reordercategories` *Reorder categories for chosen variable in MCA case only*

Description

`reordercategories` reorders the principal and standard coordinates, CTR, REP, variances and covariances of the categories for a single MCA variable

Usage

```
reordercategories(x, varno, newcats)
```

Arguments

`x`	An object of class `cabootcrsresults`
`varno`	The number of the variable to be reordered
`newcats`	A vector of length equal to the number of categories for this variable, giving the new order for the categories (e.g. c(4,1,2,3) means that the original 4th category is moved to first)

Details

This may be useful when comparing results between different data sets or from different packages

Note: does not reorder anything in the `cabasicresults` part of the `cabootcrsresults` object

Value

An object of class `cabootcrsresults`

See Also

`cabootcrs-package`, `cabootcrs`, `reflectaxes`, `cabootcrsresults`

Examples

```
bd3 <- cabootcrs(DreamData223by3, catype="mca", nboots=0, showresults=FALSE)
bd3reorderedvar2 <- reordercategories(bd3, 2, c(3,2,4,1))
summaryca(bd3)
summaryca(bd3reorderedvar2)

## Not run:

# Can be used when comparing results in different packages,
# or when adding ellipses from this package to output from others
```

```
library(FactoMineR)
library(ca)
data(tea)
# remove duplicated age variable
teamod <- tea[,c(1:18,20:36)]

# ca package uses standardised coordinates and inertias by default
catea <- mjca(teamod)
btea <- cabootcrs(teamod, catype="mca", showresults=FALSE, nboots=0, varandcat=FALSE)

# FactoMineR package uses unstandardised coordinates and inertias by default
fmtea <- MCA(teamod, method="Burt", graph=FALSE)
bteaunstd <- cabootcrs(teamod, catype="mca", showresults=FALSE, nboots=0,
                    mcaadjustinertias = FALSE, mcaadjustcoords = FALSE, varandcat=FALSE)

summary(fmtea)
summaryca(bteaunstd)
summary(catea)
summaryca(btea)

# slight difference due to different orderings of categories for these two
fmtea$var$coord / bteaunstd@Colprinccoord[,1:5]
catea$colpcoord[,1:5] / btea@Colprinccoord[,1:5]
fmtea$var$coord / catea$colpcoord[,1:5]

# Variables 22 and 23, in columns 57-65, are the problem
# The coordinates agree (apart from reflection) but the categories are in a different order
fmtea$var$coord[57:65,1:3]
bteaunstd@Colprinccoord[57:65,1:3]

catea$colpcoord[57:65,1:3]
btea@Colprinccoord[57:65,1:3]

# Coordinates agree when categories reordered and axes reflected
bteaunstdreord <- reordercategories(bteaunstd,22,c(2:5,1))
bteaunstdreord <- reordercategories(bteaunstdreord,23,c(3,2,1,4))
bteaunstdreordreflect <- reflectaxes(bteaunstdreord,c(1,4))
fmtea$var$coord / bteaunstdreordreflect@Colprinccoord[,1:5]

bteareord <- reordercategories(btea,22,c(2:5,1))
bteareord <- reordercategories(bteareord,23,c(3,2,1,4))
bteareordreflect <- reflectaxes(bteareord,c(2,5))
catea$colpcoord[,1:5] / bteareordreflect@Colprinccoord[,1:5]


## End(Not run)
```

---

sca *Performs standard Correspondence Analysis calculations*

Description

sca returns all the basic results from a CA of a matrix with rows >= cols, in an object of class cabasicresults

Usage

```
sca(X, catype = "sca", mcatype = NULL, p = 2, needtrans = FALSE)
```

Arguments

X	A data matrix with rows >= cols
catype	Can be "sca" for simple CA or "mca" for multiple CA
mcatype	If catype="mca" then this can be "Burt", "Indicator" or "doubled" depending on the analysis required. This affects the number of meaningful singular values, as does p below
p	Number of variables, only needed if catype="mca"
needtrans	TRUE if rows < columns so need to transpose in the routine

Details

This is only intended for internal use by the cabootcrs function.

Value

An object of class cabasicresults

See Also

cabootcrs-package, cabootcrs, cabasicresults

Examples

```
results <- sca(as.matrix(DreamData))
```

---

settingsinertias *Internal function to be used by printca and summaryca*

Description

settingsinertias prints the settings and the inertias

Usage

```
settingsinertias(x)
```

Arguments

x	An object of class cabootcrsresults

Value

printed output only

See Also

summaryca, printca

Examples

```
# Purely internal, not intended for use by users
```

SuicideData *Suicide data*

Description

Methods of suicide in Germany, 1974-1977

Usage

```
SuicideData
```

Format

A contingency table with 34 rows and 9 columns

rows Gender and age of individual: Females aged 10-15 (F10) to males aged 90+ (M90)

columns Method: drugs/poison (Mat), gas at home (Gas.h), gas-others (Gas.o), hanging (Hang), drowning (Drown), gunshot (Gun), stabbing (Stab), jumping (Jump), Other

Source

Nishisato, S. (1994). *Elements of Dual Scaling: An Introduction to Practical Data Analysis*. Lawrence Erlbaum Associates, New Jersey. (p12)

summaryca *Prints brief 2-d results, with standard deviations*

Description

summaryca prints correspondence analysis results for the first two dimensions, giving inertias, co-ordinates, representations, contributions and standard deviations

Usage

```
summaryca(x, datasetname = NULL, mcaprintindividuals = FALSE)
```

Arguments

x	An object of class cabootcrsresults
datasetname	The name (in "") of the data set, to be used in the output, defaults to that in cabootcrs object
mcaprintindividuals	If TRUE then print individual (row) point results in multiple correspondence analysis when using indicator or doubled matrix

Value

Printed results, no plots or objects produced

See Also

cabootcrs-package, cabootcrs, printca, plotca, cabootcrsresults

Examples

```
results <- cabootcrs(DreamData, showresults=FALSE)
summaryca(results, datasetname="Dreams")
```

Index