Package ‘ExPosition’

April 13, 2025

Type Package

Title Exploratory Analysis with the Singular Value Decomposition

Version 2.11.0

Date 2025-03-30

Description A variety of descriptive multivariate analyses with the singular value decomposition, such as principal components analysis, correspondence analysis, and multidimensional scaling. See An ExPosition of the Singular Value Decomposition in R (Beaton et al 2014) <doi:10.1016/j.csda.2013.11.006>.

License GPL-2

Encoding UTF-8

Depends prettyGraphs (>= 2.2.0)

BugReports `https://github.com/derekbeaton/ExPosition1/issues`

RoxygenNote 7.3.2

NeedsCompilation no

Author Derek Beaton [aut, cre],
Cherise R. Chin Fatt [aut],
Herve Abdi [aut]

Maintainer Derek Beaton <`exposition.software@gmail.com`>

Repository CRAN

Date/Publication 2025-04-13 16:00:22 UTC

Contents

`ExPosition-package` *ExPosition:* E*xploratory Analysis with the Singular Value Decom*Position

Description

Exposition is defined as *a comprehensive explanation of an idea*. With ExPosition for R, a comprehensive explanation of your data will be provided with minimal effort.

The core of ExPosition is the singular value decomposition (SVD; see: `svd`). The point of ExPosition is simple: to provide the user with an overview of their data that only the SVD can provide. ExPosition includes several techniques that depend on the SVD (see below for examples and functions).

Author(s)

Questions, comments, compliments, and complaints go to Derek Beaton <`exposition.software@gmail.com`>.

The following people are authors or contributors to ExPosition code, data, or examples:
Derek Beaton, Hervé Abdi, Cherise Chin-Fatt, Joseph Dunlop, Jenny Rieck, Rachel Williams, Anjali Krishnan, and Francesca M. Filbey.

References

Abdi, H., and Williams, L.J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2, 433-459.
Abdi, H. and Williams, L.J. (2010). Correspondence analysis. In N.J. Salkind, D.M., Dougherty, & B. Frey (Eds.): *Encyclopedia of Research Design*. Thousand Oaks (CA): Sage. pp. 267-278.
Abdi, H. (2007). Singular Value Decomposition (SVD) and Generalized Singular Value Decomposition (GSVD). In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics*.Thousand Oaks (CA): Sage. pp. 907-912.
Abdi, H. (2007). Metric multidimensional scaling. In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics.* Thousand Oaks (CA): Sage. pp. 598-605.
Greenacre, M. J. (2007). Correspondence Analysis in Practice. *Chapman and Hall.*

Benzécri, J. P. (1979). Sur le calcul des taux d'inertie dans l'analyse d'un questionnaire. *Cahiers de l'Analyse des Données*, **4**, 377-378.

See Also

`epPCA`, `epMDS`, `epCA`, `epMCA`

`acknowledgements` *acknowledgements*

Description

`acknowledgements` returns a list of people who have contributed to ExPosition.

Usage

```
acknowledgements()
```

Value

A list of people who have contributed something beyond code to the ExPosition family of packages.

Author(s)

Derek Beaton

`authors` *(A truncated form of) Punctuation used by six authors (data).*

Description

How six authors use 3 different types of puncatuation throughout their writing.

Usage

```
data(authors)
```

Format

authors\$ca\$data: Six authors (rows) and the frequency of three puncutuations (columns). For use with `epCA`.
authors\$mca\$data: A Burt table reformatting of the \$ca\$data. For use with `epMCA`.

References

Brunet, E. (1989). Faut-il ponderer les donnees linguistiques. *CUMFID*, 16, 39-50.
Abdi, H., and Williams, L.J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2, 433-459.
Abdi, H., and Williams, L.J. (2010). Correspondence analysis. In N.J. Salkind, D.M., Dougherty, & B. Frey (Eds.): *Encyclopedia of Research Design*. Thousand Oaks (CA): Sage. pp. 267-278.

`bada.wine` *Twelve wines from 3 regions in France with 18 attributes.*

Description

This data should be used for discriminant analyses or analyses where the *group* information is important.

Usage

```
data(bada.wine)
```

Format

bada.wine$data: Data matrix with twelve wines (rows) from 3 regions with 18 attributes (columns).
bada.wine$design: Design matrix with twelve wines (rows) with 3 regions (columns) to indicate group relationship of the data matrix.

References

Abdi, H. and Williams, L.J. (2010). Barycentric discriminant analysis (BADIA). In N.J. Salkind, D.M., Dougherty, & B. Frey (Eds.): *Encyclopedia of Research Design*. Thousand Oaks (CA): Sage. pp. 64-75.

`beer.tasting.notes` *Some of authors' personal beer tasting notes.*

Description

Tasting notes, preferences, breweries and styles of 38 different craft beers from various breweries, across various styles.

Usage

```
data(beer.tasting.notes)
```

Format

beer.tasting.notes$data: Data matrix. Tasting notes (ratings) of 38 different beers (rows) described by 16 different flavor profiles (columns).
beer.tasting.notes$brewery.design: Design matrix. Source brewery of 38 different beers (rows) across 26 breweries (columns).
beer.tasting.notes$style.design: Design matrix. Style of 38 different beers (rows) across 20 styles (columns) (styles as listed from Beer Advocate website).
beer.tasting.notes$sup.data: Supplementary data matrix. ABV and overall preference ratings of 38 beers described by two features (ABV & overall) in original value and rounded value.

Source

Jenny Rieck and Derek Beaton laboriously “collected” these data for “experimental purposes”.

References

http://www.beeradvocate.com

`beers2007` *Ten assessors sort eight beers into groups.*

Description

Ten assessors perform a free-sorting task to sort eight beers into groups.

Usage

```
data(beers2007)
```

Format

beer2007$data: A data matrix with 8 rows (beers) described by 10 assessors (columns).

References

Abdi, H., Valentin, D., Chollet, S., & Chrea, C. (2007). Analyzing assessors and products in sorting tasks: DISTATIS, theory and applications. *Food Quality and Preference*, 627-640.

`calculateConstraints` *calculateConstraints*

Description

Calculates constraints for plotting data..

Usage

```
calculateConstraints(results,x_axis=1,y_axis=2,constraints=NULL)
```

Arguments

`results`	results from ExPosition (i.e., $ExPosition.Data)
`x_axis`	which component should be on the x axis?
`y_axis`	which component should be on the y axis?
`constraints`	if available, axis constraints for the plots (determines end points of the plots).

Value

Returns a list with the following items:

`$constraints`	axis constraints for the plots (determines end points of the plots).

Author(s)

Derek Beaton

`caNorm` *Correspondence analysis preprocessing*

Description

Performs all steps required for CA processing (row profile approach).

Usage

```
caNorm(X, X_dimensions, colTotal, rowTotal, grandTotal, weights =
NULL, masses = NULL)
```

Arguments

`X`	Data matrix
`X_dimensions`	The dimensions of X in a vector of length 2 (rows, columns). See `dim`
`colTotal`	Vector of column sums.
`rowTotal`	Vector of row sums.
`grandTotal`	Grand total of X
`weights`	Optional weights to include for the columns.
`masses`	Optional masses to include for the rows.

Value

`rowCenter`	The barycenter of X.
`masses`	Masses to be used for the GSVD.
`weights`	Weights to be used for the GSVD.
`rowProfiles`	The row profiles of X.
`deviations`	Deviations of row profiles from `rowCenter`.

Author(s)

Derek Beaton

`caSupplementalElementsPreProcessing` *Correspondence Analysis preprocessing.*

Description

CA preprocessing for data. Can be performed on rows or columns of your data. This is a row-profile normalization.

Usage

```
caSupplementalElementsPreProcessing(SUP.DATA)
```

Arguments

`SUP.DATA`	Data that will be supplemental. Row profile normalization is used. For supplemental rows use `t(SUP.DATA)`.

Value

returns a matrix that is preprocessed for supplemental projections.

Author(s)

Derek Beaton

See Also

`mdsSupplementalElementsPreProcessing`, `pcaSupplementaryColsPreProcessing`, `pcaSupplementaryRowsPreProce`
`hellingerSupplementaryColsPreProcessing`, `hellingerSupplementaryRowsPreProcessing`,
`supplementaryCols`, `supplementaryRows`, `supplementalProjection`, `rowNorms`

`chi2Dist` *Chi-square Distance computation*

Description

Performs a chi-square distance. Primarily used for `epMDS`.

Usage

```
chi2Dist(X)
```

Arguments

`X`	Compute chi-square distances between row items.

Value

`D`	Distance matrix for `epMDS` analysis.
`MW`	a list of masses and weights. Weights not used in MDS.

Author(s)

Hervé Abdi

`coffee.data` *Small data set on flavor perception and preferences for coffee.*

Description

One coffee from Oak Cliff roasters (Dallas, TX) was used in this experiment. Honduran source with a medium roast. The coffee was brewed in two ways and served in two ways (i.e., a 2x2 design). Two batches each of coffee were brewed at 180 degrees (Hot) Farenheit or at room temperature (Cold). One of each was served cold or heated back up to 180 degrees (Hot).

Usage

```
data(coffee.data)
```

Format

coffee.data$preferences: Ten participants indicated if they liked a particular serving or not.
coffee.data$ratings: Ten participants indicated on a scale of 0-2 the presence of particular flavors. In an array format.

Details

Flavor profiles measured: Salty, Spice Cabinet, Sweet, Bittery, and Nutty.

`computeMW` *computeMW*

Description

Computes masses and weights for use.

Usage

```
computeMW(DATA, masses = NULL, weights = NULL)
```

Arguments

`DATA`	original data; will be used to compute masses and weights if none are provided.
`masses`	a vector or (diagonal) matrix of masses for the row items. If NULL (default), masses are computed as 1/# of rows
`weights`	a vector or (diagonal) matrix of weights for the column items. If NULL (default), weights are computed as 1/# of columns

Value

Returns a list with the following items:

`M`	a diagonal matrix of masses (if too large, a vector is returned).
`W`	a diagonal matrix of weights (if too large, a vector is returned).

Author(s)

Derek Beaton

`coreCA` *coreCA*

Description

coreCA performs the core of correspondence analysis (CA), multiple correspondence analysis (MCA) and related techniques.

Usage

```
coreCA(DATA, masses = NULL, weights = NULL, hellinger = FALSE,
symmetric = TRUE, decomp.approach = 'svd', k = 0)
```

Arguments

`DATA`	original data to decompose and analyze via the singular value decomposition.
`masses`	a vector or diagonal matrix with masses for the rows (observations). If NULL, one is created or the plain SVD is used.
`weights`	a vector or diagonal matrix with weights for the columns (measures). If NULL, one is created or the plain SVD is used.
`hellinger`	a boolean. If FALSE (default), Chi-square distance will be used. If TRUE, Hellinger distance will be used.
`symmetric`	a boolean. If TRUE (default) symmetric factor scores for rows and columns are computed. If FALSE, the simplex (column-based) will be returned.
`decomp.approach`	string. A switch for different decompositions (typically for speed). See `pickSVD`.
`k`	number of components to return (this is not a rotation, just an *a priori* selection of how much data should be returned).

Details

This function should not be used directly. Please use `epCA` or `epMCA` unless you plan on writing extensions to ExPosition. Any extensions wherein CA is the primary analysis should use `coreCA`.

Value

Returns a large list of items which are also returned in `epCA` and `epMCA` (the help files for those functions will refer to this as well).
All items with a letter followed by an i are for the I rows of a DATA matrix. All items with a letter followed by an j are for the J rows of a DATA matrix.

`fi`	factor scores for the row items.
`di`	square distances of the row items.

`ci`	contributions (to the variance) of the row items.
`ri`	cosines of the row items.
`fj`	factor scores for the column items.
`dj`	square distances of the column items.
`cj`	contributions (to the variance) of the column items.
`rj`	cosines of the column items.
`t`	the percent of explained variance per component (tau).
`eigs`	the eigenvalues from the decomposition.
`pdq`	the set of left singular vectors (pdq$p) for the rows, singular values (pdq$Dv and pdq$Dd), and the set of right singular vectors (pdq$q) for the columns.
`M`	a column-vector or diagonal matrix of masses (for the rows)
`W`	a column-vector or diagonal matrix of weights (for the columns)
`c`	a centering vector (for the columns).
`X`	the final matrix that was decomposed (includes scaling, centering, masses, etc...).
`hellinger`	a boolean. TRUE if Hellinger distance was used.
`symmetric`	a boolean. FALSE if asymmetric factor scores should be computed.

Author(s)

Derek Beaton and Hervé Abdi.

References

Abdi, H., and Williams, L.J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2, 433-459.
Abdi, H., and Williams, L.J. (2010). Correspondence analysis. In N.J. Salkind, D.M., Dougherty, & B. Frey (Eds.): *Encyclopedia of Research Design*. Thousand Oaks (CA): Sage. pp. 267-278.
Abdi, H. (2007). Singular Value Decomposition (SVD) and Generalized Singular Value Decomposition (GSVD). In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics*.Thousand Oaks (CA): Sage. pp. 907-912. Greenacre, M. J. (2007). Correspondence Analysis in Practice. *Chapman and Hall*.

See Also

`epCA`, `epMCA`

`coreMDS` *coreMDS*

Description

coreMDS performs metric multidimensional scaling (MDS).

Usage

```
coreMDS(DATA, masses = NULL, decomp.approach = 'svd', k = 0)
```

Arguments

`DATA`	original data to decompose and analyze via the singular value decomposition.
`masses`	a vector or diagonal matrix with masses for the rows (observations). If NULL, one is created.
`decomp.approach`	string. A switch for different decompositions (typically for speed). See `pickSVD`.
`k`	number of components to return (this is not a rotation, just an *a priori* selection of how much data should be returned).

Details

`epMDS` should not be used directly unless you plan on writing extensions to ExPosition. See `epMDS`

Value

Returns a large list of items which are also returned in `epMDS`.
All items with a letter followed by an *i* are for the *I* rows of a DATA matrix. All items with a letter followed by an *j* are for the *J* rows of a DATA matrix.

`fi`	factor scores for the row items.
`di`	square distances of the row items.
`ci`	contributions (to the variance) of the row items.
`ri`	cosines of the row items.
`masses`	a column-vector or diagonal matrix of masses (for the rows)
`t`	the percent of explained variance per component (tau).
`eigs`	the eigenvalues from the decomposition.
`pdq`	the set of left singular vectors (pdq$p) for the rows, singular values (pdq$Dv and pdq$Dd), and the set of right singular vectors (pdq$q) for the columns.
`X`	the final matrix that was decomposed (includes scaling, centering, masses, etc...).

Author(s)

Derek Beaton and Hervé Abdi.

References

Abdi, H. (2007). Metric multidimensional scaling. In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics.* Thousand Oaks (CA): Sage. pp. 598-605.
O'Toole, A. J., Jiang, F., Abdi, H., and Haxby, J. V. (2005). Partially distributed representations of objects and faces in ventral temporal cortex. *Journal of Cognitive Neuroscience*, *17*(4), 580-590.

See Also

`epMDS`

`corePCA` *corePCA*

Description

corePCA performs the core of principal components analysis (PCA), and related techniques.

Usage

```
corePCA(DATA, M = NULL, W = NULL, decomp.approach = 'svd', k = 0)
```

Arguments

`DATA`	original data to decompose and analyze via the singular value decomposition.
`M`	a vector or diagonal matrix with masses for the rows (observations). If NULL, one is created or the plain SVD is used.
`W`	a vector or diagonal matrix with weights for the columns (measures). If NULL, one is created or the plain SVD is used.
`decomp.approach`	string. A switch for different decompositions (typically for speed). See `pickSVD`.
`k`	number of components to return (this is not a rotation, just an *a priori* selection of how much data should be returned).

Details

This function should not be used directly. Please use `epPCA` unless you plan on writing extensions to ExPosition.

Value

Returns a large list of items which are also returned in `epPCA` (the help files for those functions will refer to this as well).
All items with a letter followed by an *i* are for the *I* rows of a DATA matrix. All items with a letter followed by an *j* are for the *J* rows of a DATA matrix.

`fi`	factor scores for the row items.
`di`	square distances of the row items.
`ci`	contributions (to the variance) of the row items.
`ri`	cosines of the row items.
`fj`	factor scores for the column items.
`dj`	square distances of the column items.
`cj`	contributions (to the variance) of the column items.
`rj`	cosines of the column items.
`t`	the percent of explained variance per component (tau).
`eigs`	the eigenvalues from the decomposition.
`pdq`	the set of left singular vectors (pdq$p) for the rows, singular values (pdq$Dv and pdq$Dd), and the set of right singular vectors (pdq$q) for the columns.
`X`	the final matrix that was decomposed (includes scaling, centering, masses, etc...).

Author(s)

Derek Beaton and Hervé Abdi.

References

Abdi, H., and Williams, L.J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2, 433-459.
Abdi, H. (2007). Singular Value Decomposition (SVD) and Generalized Singular Value Decomposition (GSVD). In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics*.Thousand Oaks (CA): Sage. pp. 907-912.

See Also

`epPCA`

`createDefaultDesign` *createDefaultDesign*

Description

Creates a default design matrix, wherein all observations (i.e., row items) are in the same group.

Usage

```
createDefaultDesign(DATA)
```

Arguments

`DATA`	original data that requires a design matrix

Value

`DESIGN`	a column-vector matrix to indicate that all observations are in the same group.

Author(s)

Derek Beaton

`designCheck` *designCheck*

Description

Checks and/or creates a dummy-coded design matrix.

Usage

```
designCheck(DATA, DESIGN = NULL, make_design_nominal = TRUE)
```

Arguments

`DATA`	original data that should be matched to a design matrix
`DESIGN`	a column vector with levels for observations or a dummy-coded matrix
`make_design_nominal`	a boolean. Will make DESIGN nominal if TRUE (default).

Details

Returns a properly formatted, dummy-coded (or disjunctive coding) design matrix.

Value

`DESIGN`	dummy-coded design matrix

Author(s)

Derek Beaton

Examples

```
data <- iris[,c(1:4)]
design <- as.matrix(iris[,c('Species')])
iris.design <- designCheck(data,DESIGN=design,make_design_nominal=TRUE)
```

`dica.ad` *Alzheimer's Patient-Spouse Dyads.*

Description

Conversational data from Alzheimer's Patient-Spouse Dyads.

Usage

```
data(dica.ad)
```

Format

dica.ad$data: Seventeen dyads described by 58 variables.
dica.ad$design: Seventeen dyads that belong to three groups.

References

Williams, L.J., Abdi, H., French, R., & Orange, J.B. (2010). A tutorial on Multi-Block Discriminant Correspondence Analysis (MUDICA): A new method for analyzing discourse data from clinical populations. *Journal of Speech Language and Hearing Research*, **53**, 1372-1393.

`dica.wine` *Twelve wines from 3 regions in France with 16 attributes.*

Description

This data should be used for discriminant analyses or analyses where the *group* information is important.

Usage

```
data(dica.wine)
```

Format

dica.wine$data: Data matrix with twelve wines (rows) from 3 regions with 16 attributes (columns) *in disjunctive (0/1) coding*.
dica.wine$design: Design matrix with twelve wines (rows) with 3 regions (columns) to indicate group relationship of the data matrix.

References

Abdi, H. (2007). Discriminant correspondence analysis. In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics*. Thousand Oaks (CA): Sage. pp. 270-275.

`ep.iris` *Fisher's iris Set (for ExPosition)*

Description

The world famous Fisher's iris set: 150 flowers from 3 species with 4 attributes.

Usage

```
data(ep.iris)
```

Format

ep.iris$data: Data matrix with 150 flowers (rows) from 3 species with 4 attributes (columns) describing sepal and petal features.
ep.iris$design: Design matrix with 150 flowers (rows) with 3 species (columns) indicating which flower belongs to which species.

Source

http://en.wikipedia.org/wiki/Iris_flower_data_set

`epCA` *epCA: Correspondence Analysis (CA) via ExPosition.*

Description

Correspondence Analysis (CA) via ExPosition.

Usage

```
epCA(DATA, DESIGN = NULL, make_design_nominal = TRUE, masses = NULL,
weights = NULL, hellinger = FALSE, symmetric = TRUE, graphs = TRUE, k = 0)
```

Arguments

`DATA`	original data to perform a CA on.
`DESIGN`	a design matrix to indicate if rows belong to groups.
`make_design_nominal`	a boolean. If TRUE (default), DESIGN is a vector that indicates groups (and will be dummy-coded). If FALSE, DESIGN is a dummy-coded matrix.
`masses`	a diagonal matrix or column-vector of masses for the row items.
`weights`	a diagonal matrix or column-vector of weights for the column it
`hellinger`	a boolean. If FALSE (default), Chi-square distance will be used. If TRUE, Hellinger distance will be used.
`symmetric`	a boolean. If TRUE (default) symmetric factor scores for rows and columns are computed. If FALSE, the simplex (column-based) will be returned.
`graphs`	a boolean. If TRUE (default), graphs and plots are provided (via `epGraphs`)
`k`	number of components to return.

Details

`epCA` performs correspondence analysis. Essentially, a PCA for qualitative data (frequencies, proportions). If you decide to use Hellinger distance, it is best to set `symmetric` to FALSE.

Value

See `coreCA` for details on what is returned.

Author(s)

Derek Beaton

References

Abdi, H., and Williams, L.J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2, 433-459.
Abdi, H., and Williams, L.J. (2010). Correspondence analysis. In N.J. Salkind, D.M., Dougherty, & B. Frey (Eds.): *Encyclopedia of Research Design*. Thousand Oaks (CA): Sage. pp. 267-278.
Abdi, H. (2007). Singular Value Decomposition (SVD) and Generalized Singular Value Decomposition (GSVD). In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics*.Thousand Oaks (CA): Sage. pp. 907-912.
Greenacre, M. J. (2007). Correspondence Analysis in Practice. *Chapman and Hall*.

See Also

`coreCA`, `epMCA`

Examples

```
data(authors)
ca.authors.res <- epCA(authors$ca$data)
```

`epGraphs` *epGraphs: ExPosition plotting function*

Description

ExPosition plotting function which is an interface to `prettyGraphs`.

Usage

```
epGraphs(res, x_axis = 1, y_axis = 2, epPlotInfo = NULL, DESIGN=NULL,
fi.col = NULL, fi.pch = NULL, fj.col = NULL, fj.pch = NULL, col.offset =
NULL, constraints = NULL, xlab = NULL, ylab = NULL, main = NULL,
contributionPlots = TRUE, correlationPlotter = TRUE, graphs = TRUE)
```

Arguments

`res`	results from ExPosition
`x_axis`	which component should be on the x axis?
`y_axis`	which component should be on the y axis?
`epPlotInfo`	A list (`$Plotting.Data`) from `epGraphs` or `ExPosition`.
`DESIGN`	A design matrix to apply colors (by pallete selection) to row items
`fi.col`	A matrix of colors for the row items. If NULL, colors will be selected.
`fi.pch`	A matrix of pch values for the row items. If NULL, pch values are all 21.
`fj.col`	A matrix of colors for the column items. If NULL, colors will be selected.
`fj.pch`	A matrix of pch values for the column items. If NULL, pch values are all 21.

`col.offset`	A numeric offset value. Is passed to `createColorVectorsByDesign`.
`constraints`	Plot constraints as returned from `prettyPlot`. If NULL, constraints are selected.
`xlab`	x axis label
`ylab`	y axis label
`main`	main label for the graph window
`contributionPlots`	a boolean. If TRUE (default), contribution bar plots will be created.
`correlationPlotter`	a boolean. If TRUE (default), a correlation circle plot will be created. Applies to PCA family of methods (CA is excluded for now).
`graphs`	a boolean. If TRUE, graphs are created. If FALSE, only data associated to plotting (e.g., constraints, colors) are returned.

Details

epGraphs is an interface between `ExPosition` and `prettyGraphs`.

Value

The following items are bundled inside of $Plotting.Data:

`$fi.col`	the colors that are associated to the row items ($fi).
`$fi.pch`	the pch values associated to the row items ($fi).
`$fj.col`	the colors that are associated to the column items ($fj).
`$fj.pch`	the pch values associated to the column items ($fj).
`$constraints`	axis constraints for the plots (determines end points of the plots).

Author(s)

Derek Beaton

See Also

`prettyGraphs`

Examples

```
#this is for ExPosition's iris data
data(ep.iris)
pca.iris.res <- epPCA(ep.iris$data)
#this will put plotting data into a new variable.
epGraphs.2.and.3 <- epGraphs(pca.iris.res,x_axis=2,y_axis=3)
```

`epMCA` *epMCA: Multiple Correspondence Analysis (MCA) via ExPosition.*

Description

Multiple Correspondence Analysis (MCA) via ExPosition.

Usage

```
epMCA(DATA, make_data_nominal = TRUE, DESIGN = NULL,
make_design_nominal = TRUE, masses = NULL, weights = NULL, hellinger =
FALSE, symmetric = TRUE, correction = c("b"), graphs = TRUE, k = 0)
```

Arguments

`DATA`	original data to perform a MCA on. This data can be in original formatting (qualitative levels) or in dummy-coded variables.
`make_data_nominal`	a boolean. If TRUE (default), DATA is recoded as a dummy-coded matrix. If FALSE, DATA is a dummy-coded matrix.
`DESIGN`	a design matrix to indicate if rows belong to groups.
`make_design_nominal`	a boolean. If TRUE (default), DESIGN is a vector that indicates groups (and will be dummy-coded). If FALSE, DESIGN is a dummy-coded matrix.
`masses`	a diagonal matrix or column-vector of masses for the row items.
`weights`	a diagonal matrix or column-vector of weights for the column it
`hellinger`	a boolean. If FALSE (default), Chi-square distance will be used. If TRUE, Hellinger distance will be used.
`symmetric`	a boolean. If TRUE symmetric factor scores for rows.
`correction`	which corrections should be applied? "b" = Benzécri correction, "bg" = Greenacre adjustment to Benzécri correction.
`graphs`	a boolean. If TRUE (default), graphs and plots are provided (via `epGraphs`)
`k`	number of components to return.

Details

`epMCA` performs multiple correspondence analysis. Essentially, a CA for categorical data.
It should be noted that when `hellinger` is selected as TRUE, no correction will be performed. Additionally, if you decide to use Hellinger, it is best to set `symmetric` to FALSE.

Value

See `coreCA` for details on what is returned. In addition to the values returned:

`$pdq`	this is the corrected SVD data, if a correction was selected. If no correction was selected, it is uncorrected.
`$pdq.uncor`	uncorrected SVD data.

Author(s)

Derek Beaton

References

Abdi, H., and Williams, L.J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2, 433-459.
Abdi, H., and Williams, L.J. (2010). Correspondence analysis. In N.J. Salkind, D.M., Dougherty, & B. Frey (Eds.): *Encyclopedia of Research Design*. Thousand Oaks (CA): Sage. pp. 267-278.
Abdi, H. (2007). Singular Value Decomposition (SVD) and Generalized Singular Value Decomposition (GSVD). In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics*.Thousand Oaks (CA): Sage. pp. 907-912.
Benzécri, J. P. (1979). Sur le calcul des taux d'inertie dans l'analyse d'un questionnaire. *Cahiers de l'Analyse des Données*, **4**, 377-378.
Greenacre, M. J. (2007). Correspondence Analysis in Practice. *Chapman and Hall*.

See Also

`coreCA`, `epCA`, `mca.eigen.fix`

Examples

```
data(mca.wine)
mca.wine.res <- epMCA(mca.wine$data)
```

`epMDS` *epMDS: Multidimensional Scaling (MDS) via ExPosition.*

Description

Multidimensional Scaling (MDS) via ExPosition.

Usage

```
epMDS(DATA, DATA_is_dist = TRUE, method="euclidean", DESIGN = NULL,
make_design_nominal = TRUE, masses = NULL, graphs = TRUE, k = 0)
```

Arguments

`DATA`	original data to perform a MDS on.
`DATA_is_dist`	a boolean. If TRUE (default) the DATA matrix should be a symmetric distance matrix. If FALSE, a Euclidean distance of row items will be computed and used.
`method`	which distance metric should be used. `method` matches `dist`; Two additional distances are avaialble: "correlation" and "chi2". For "chi2" see `chi2Dist`. Default is "euclidean".
`DESIGN`	a design matrix to indicate if rows belong to groups.
`make_design_nominal`	a boolean. If TRUE (default), DESIGN is a vector that indicates groups (and will be dummy-coded). If FALSE, DESIGN is a dummy-coded matrix.
`masses`	a diagonal matrix (or vector) that contains the masses (for the row items).
`graphs`	a boolean. If TRUE (default), graphs and plots are provided (via `epGraphs`)
`k`	number of components to return.

Details

`epMDS` performs metric multi-dimensional scaling. Essentially, a PCA for a symmetric distance matrix.

Value

See `coreMDS` for details on what is returned. epMDS only returns values related to row items (e.g., fi, ci); no column data is returned.

`D`	the distance matrix that was decomposed. In most cases, it is returned as a squared distance.

Note

With respect to input of `DATA`, `epMDS` differs slightly from other versions of multi-dimensional scaling.
If you provide a rectangular matrix (e.g., observations x measures), `epMDS` will compute a distance matrix and square it.
If you provide a distance (dissimilarity) matrix, `epMDS` does not square it.

Author(s)

Derek Beaton

References

Abdi, H. (2007). Metric multidimensional scaling. In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics.* Thousand Oaks (CA): Sage. pp. 598-605.
O'Toole, A. J., Jiang, F., Abdi, H., and Haxby, J. V. (2005). Partially distributed representations of objects and faces in ventral temporal cortex. *Journal of Cognitive Neuroscience*, *17*(4), 580-590.

See Also

corePCA, epPCA

Examples

```
data(jocn.2005.fmri)
#by default, components 1 and 2 will be plotted.
mds.res.images <- epMDS(jocn.2005.fmri$images$data)

##iris example
data(ep.iris)
iris.rectangular <- epMDS(ep.iris$data,DATA_is_dist=FALSE)
iris.euc.dist <- dist(ep.iris$data,upper=TRUE,diag=TRUE)
iris.sq.euc.dist <- as.matrix(iris.euc.dist^2)
iris.sq <- epMDS(iris.sq.euc.dist)
```

`epPCA` *epPCA: Principal Component Analysis (PCA) via ExPosition.*

Description

Principal Component Analysis (PCA) via ExPosition.

Usage

```
epPCA(DATA, scale = TRUE, center = TRUE, DESIGN = NULL,
make_design_nominal = TRUE, graphs = TRUE, k = 0)
```

Arguments

`DATA`	original data to perform a PCA on.
`scale`	a boolean, vector, or string. See `expo.scale` for details.
`center`	a boolean, vector, or string. See `expo.scale` for details.
`DESIGN`	a design matrix to indicate if rows belong to groups.
`make_design_nominal`	a boolean. If TRUE (default), DESIGN is a vector that indicates groups (and will be dummy-coded). If FALSE, DESIGN is a dummy-coded matrix.
`graphs`	a boolean. If TRUE (default), graphs and plots are provided (via `epGraphs`)
`k`	number of components to return.

Details

`epPCA` performs principal components analysis on a data matrix.

Value

See corePCA for details on what is returned.

Author(s)

Derek Beaton

References

Abdi, H., and Williams, L.J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2, 433-459.
Abdi, H. (2007). Singular Value Decomposition (SVD) and Generalized Singular Value Decomposition (GSVD). In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics*.Thousand Oaks (CA): Sage. pp. 907-912.

See Also

corePCA, epMDS

Examples

```
data(words)
pca.words.res <- epPCA(words$data)
```

`expo.scale` *Scaling functions for ExPosition.*

Description

`expo.scale` is a more elaborate, and complete, version of scale. Several text options are available, but more importantly, the center and scale factors are *always* returned.

Usage

```
expo.scale(DATA, center = TRUE, scale = TRUE)
```

Arguments

`DATA`	Data to center, scale, or both.
`center`	boolean, or (numeric) vector. If boolean or vector, it works just as scale.
`scale`	boolean, text, or (numeric) vector. If boolean or vector, it works just as scale. The following text options are available: 'z': z-score normalization, 'sd': standard deviation normalization, 'rms': root mean square normalization, 'ss1': sum of squares (of columns) equals 1 normalization.

Value

A data matrix that is scaled with the following `attributes` (see `scale`):

`$‘scaled:center‘`	The center of the data. If no center is provided, all 0s will be returned.
`$‘scaled:scale‘`	The scale factor of the data. If no scale is provided, all 1s will be returned.

Author(s)

Derek Beaton

`faces2005` *Faces analyzed using Four Algorithms*

Description

Four algorithms compared using a distance matrix between six faces.

Usage

```
data(faces2005)
```

Format

faces2005$data: A data structure representing a distance matrix (6X6) for four algorithms.

References

Abdi, H., & Valentin, D. (2007). DISTATIS: the analysis of multiple distance matrices. *Encyclopedia of Measurement and Statistics*. 284-290.

`french.social` *How twelve French families spend their income on groceries.*

Description

This data should be used with `epPCA`

Usage

```
data(french.social)
```

Format

french.social$data: Data matrix with twelve families (rows) with 7 attributes (columns) describing what they spend their income on.

References

Lebart, L., and Fénelon, J.P. (1975) *Statistique et informatique appliquées*. Paris: Dunod
Abdi, H., and Williams, L.J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2, 433-459.

`genPDQ` *genPDQ: the GSVD*

Description

genPDQ performs the SVD and GSVD for all methods in `ExPosition`.

Usage

```
genPDQ(datain, M = NULL, W = NULL, is.mds = FALSE, decomp.approach =
"svd", k = 0)
```

Arguments

`datain`	fully preprocessed data to be decomposed.
`M`	vector of masses (for the rows)
`W`	vector of weights (for the columns)
`is.mds`	a boolean. If the method is of MDS (e.g., `epMDS`), use TRUE. All other methods: FALSE
`decomp.approach`	a string. Allows for the user to choose which decomposition method to perform. Current options are SVD or Eigen.
`k`	number of components to return (this is not a rotation, just an *a priori* selection of how much data should be returned).

Details

This function should only be used to create new methods based on the SVD or GSVD.

Value

Data of class `epSVD` which is a list of matrices and vectors:

`P`	The left singular vectors (rows).
`Q`	The right singular vectors (columns).
`Dv`	Vector of the singular values.
`Dd`	Diagonal matrix of the singular values.
`ng`	Number of singular values/vectors
`rank`	Rank of the decomposed matrix. If it is 1, 0s are padded to the above items for plotting purposes.
`tau`	Explained variance per component

Author(s)

Derek Beaton

See Also

`pickSVD`

`great.beer.tasting.1` *A collection of beer tasting notes from untrained assessors.*

Description

A collection of beer tasting notes of 9 beers, across 16 descriptors, from 4 untrained assessors.

Usage

```
data(great.beer.tasting.1)
```

Format

great.beer.tasting.1$data: Data matrix (cube). Tasting notes (ratings) of 9 different beers (rows) described by 16 different flavor profiles (columns) by 4 untrained assessors. Thes data contain NAs and must be imputed or adjusted before an analysis is performed.
great.beer.tasting.1$brewery.design: Design matrix. Source brewery of 9 different beers (rows) across 5 breweries (columns).
great.beer.tasting.1$flavor: Design matrix. Intended prominent flavor of 9 different beers (rows) across 3 flavor profiles (columns).

Source

Rachel Williams, Jenny Rieck and Derek Beaton recoded, collected data and/or “ran the experiment”.

`great.beer.tasting.2` *A collection of beer tasting notes from untrained assessors.*

Description

A collection of beer tasting notes of 13 beers, across 15 descriptors, from 9 untrained assessors.

Usage

```
data(great.beer.tasting.2)
```

Format

great.beer.tasting.2$data: Data matrix (cube). Tasting notes (ratings) of 13 different beers (rows) described by 15 different flavor profiles (columns) by 9 untrained assessors. All original values were on an interval scale of 0-5. Any decimal values are imputed from alternate data sources or additional assessors.
great.beer.tasting.2$brewery.design: Design matrix. Source brewery of 13 different beers (rows) across 13 breweries (columns).
great.beer.tasting.2$style.design: Design matrix. Style of 13 different beers (rows) across 8 styles (columns). Some complex styles were truncated.

Source

Rachel Williams, Jenny Rieck and Derek Beaton recoded, collected data and/or "ran the experiment".

`hellingerNorm` *Hellinger version of CA preprocessing*

Description

Performs all steps required for Hellinger form of CA processing (row profile approach).

Usage

```
hellingerNorm(X, X_dimensions, colTotal, rowTotal, grandTotal,
weights = NULL, masses = NULL)
```

Arguments

`X`	Data matrix
`X_dimensions`	The dimensions of X in a vector of length 2 (rows, columns). See `dim`
`colTotal`	Vector of column sums.
`rowTotal`	Vector of row sums.
`grandTotal`	Grand total of X
`weights`	Optional weights to include for the columns.
`masses`	Optional masses to include for the rows.

Value

`rowCenter`	The barycenter of X.
`masses`	Masses to be used for the GSVD.
`weights`	Weights to be used for the GSVD.
`rowProfiles`	The row profiles of X.
`deviations`	Deviations of row profiles from `rowCenter`.

Author(s)

Derek Beaton and Hervé Abdi

`hellingerSupplementaryColsPreProcessing`

Preprocessing for supplementary columns in Hellinger analyses.

Description

Preprocessing for supplementary columns in Hellinger analyses.

Usage

```
hellingerSupplementaryColsPreProcessing(SUP.DATA, W = NULL, M = NULL)
```

Arguments

`SUP.DATA`	A supplemental matrix that has the same number of rows as an active data set.
`W`	A vector or matrix of Weights. If none are provided, a default is computed.
`M`	A vector or matrix of Masses. If none are provided, a default is computed.

Value

a matrix that has been preprocessed to project supplementary rows for Hellinger methods.

Author(s)

Derek Beaton

`hellingerSupplementaryRowsPreProcessing`
Preprocessing for supplementary rows in Hellinger analyses.

Description

Preprocessing for supplementary rows in Hellinger analyses.

Usage

```
hellingerSupplementaryRowsPreProcessing(SUP.DATA, center = NULL)
```

Arguments

`SUP.DATA`	A supplemental matrix that has the same number of rows as an active data set.
`center`	The center from the active data. NULL will center `SUP.DATA` to itself.

Value

a matrix that has been preprocessed to project supplementary columns for Hellinger methods.

Author(s)

Derek Beaton

`jlsr.2010.ad` *Data from 17 Alzheimer's Patient-Spouse dyads.*

Description

Seventeen Alzheimer's Patient-Spouse Dyads had conversations recorded and 58 attributes were recoded for this data. Each attribute is a frequency of occurence of the item.

Usage

```
data(jlsr.2010.ad)
```

Format

jlsr.2010.adcadata: Seventeen patient-spouse dyads (rows) described by 58 conversation items. For use with `epCA` and discriminant analyses.
jlsr.2010.admcadesign: A design matrix that indicates which group the dyad belongs to: control (CTRL), early stage Alzheimer's (EDAT) or middle stage Alzheimer's (MDAT).

References

Williams, L.J., Abdi, H., French, R., and Orange, J.B. (2010). A tutorial on Multi-Block Discriminant Correspondence Analysis (MUDICA): A new method for analyzing discourse data from clinical populations. *Journal of Speech Language and Hearing Research*, 53, 1372-1393.

`jocn.2005.fmri` *Data of categories of images as view in an* f*MRI experiment.*

Description

Contains 2 data sets: distance matrix of *f*MRI scans of participants viewing categories of items and distance matrix of the actual pixels from the images in each category.

Usage

```
data(jocn.2005.fmri)
```

Format

jocn.2005.fmri$images$data: A distance matrix of 6 categories of images based on a pixel analysis.
jocn.2005.fmri$scans$data: A distance matrix of 6 categories of images based on *f*MRI scans.

References

O'Toole, A. J., Jiang, F., Abdi, H., and Haxby, J. V. (2005). Partially distributed representations of objects and faces in ventral temporal cortex. *Journal of Cognitive Neuroscience*, *17*(4), 580-590.
Haxby, J. V., Gobbini, M. I., Furey, M. L., Ishai, A., Schouten, J. L., and Pietrini, P. (2001). Distributed and overlapping representation of faces and objects in ventral temporal cortex. *Science*, 293, 2425-2430.

See Also

http://openfmri.org/dataset/ds000105

`makeDistancesAndWeights` *Makes distances and weights for MDS analyses (see* `epMDS`*).*

Description

Makes distances and weights for MDS analyses (see `epMDS`).

Usage

```
makeDistancesAndWeights(DATA, method = "euclidean", masses = NULL)
```

Arguments

`DATA`	A data matrix to compute distances between row items.
`method`	which distance metric should be used. `method` matches `dist`; Two additional distances are avaialble: "correlation" and "chi2". For "chi2" see `chi2Dist`. Default is "euclidean".
`masses`	a diagonal matrix (or vector) that contains the masses (for the row items).

Value

`D`	Distance matrix for analysis
`MW`	a list item with masses and weights. Weights are not used in `epMDS`.

Author(s)

Derek Beaton

See Also

`link{computeMW}`, `link{epMDS}`, `link{coreMDS}`

`makeNominalData` *makeNominalData*

Description

Transforms each column into measure-response columns with disjunctive (0/1) coding. If NA is found somewhere in matrix, barycentric recoding is peformed for the missing value(s).

Usage

```
makeNominalData(datain)
```

Arguments

`datain`	a data matrix where the *columns* will be recoded.

Value

`dataout`	a transformed version of *datain*.

Author(s)

Derek Beaton

See Also

`epMCA`

Examples

```
data(mca.wine)
nominal.wine <- makeNominalData(mca.wine$data)
```

`makeRowProfiles` *Preprocessing for CA-based analyses*

Description

This function performs all preprocessing steps required for Correspondence Analysis-based preprocessing.

Usage

```
makeRowProfiles(X, weights = NULL, masses = NULL, hellinger = FALSE)
```

Arguments

`X`	Data matrix.
`weights`	optional. Weights to include in preprocessing.
`masses`	optional. Masses to include in preprocessing.
`hellinger`	a boolean. If TRUE, Hellinger preprocessing is used. Else, CA row profile is computed.

Value

Returns from `link{hellingerNorm}` or `caNorm`.

Author(s)

Derek Beaton

`mca.eigen.fix` *mca.eigen.fix*

Description

A function for correcting the eigenvalues and output from multiple correspondence analysis (MCA, `epMCA`)

Usage

```
mca.eigen.fix(DATA, mca.results, make_data_nominal = TRUE,
numVariables = NULL, correction = c("b"), symmetric = FALSE)
```

Arguments

`DATA`	original data (i.e., not transformed into disjunctive coding)
`mca.results`	output from `epMCA`
`make_data_nominal`	a boolean. Should *DATA* be transformed into disjunctive coding? Default is TRUE.
`numVariables`	the number of actual measures/variables in the data (typically the number of columns in *DATA*)
`correction`	which corrections should be applied? "b" = Benzécri correction, "bg" = Greenacre adjustment to Benzécri correction.
`symmetric`	a boolean. If the results from MCA are symmetric or asymmetric factor scores. Default is FALSE.

Value

`mca.results`	a modified version of mca.results. Factor scores (e.g., $fi, $fj), and $pdq are updated based on corrections chosen.

Author(s)

Derek Beaton

References

Benzécri, J. P. (1979). Sur le calcul des taux d'inertie dans l'analyse d'un questionnaire. *Cahiers de l'Analyse des Données*, **4**, 377-378.
Greenacre, M. J. (2007). Correspondence Analysis in Practice. *Chapman and Hall*.

See Also

`epMCA`

Examples

```
data(mca.wine)
#No corrections used in MCA
mca.wine.res.uncor <- epMCA(mca.wine$data,correction=NULL)
data <- mca.wine$data
expo.output <- mca.wine.res.uncor$ExPosition.Data
#mca.eigen.fix with just Benzécri correction
mca.wine.res.b <- mca.eigen.fix(data, expo.output,correction=c('b'))
#mca.eigen.fix with Benzécri + Greenacre adjustment
mca.wine.res.bg <- mca.eigen.fix(data,expo.output,correction=c('b','g'))
```

`mca.wine` *Six wines described by several assessors with qualitative attributes.*

Description

Six wines described by several assessors with qualitative attributes.

Usage

```
data(mca.wine)
```

Format

mca.wine$data: A (categorical) data matrix with 6 wines (rows) from several assessors described by 10 attributes (columns). For use with `epMCA`.

References

Abdi, H., & Valentin, D. (2007). Multiple correspondence analysis. In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics*. Thousand Oaks (CA): Sage. pp. 651-657.

`mdsSupplementalElementsPreProcessing` *MDS preprocessing*

Description

Preprocessing of supplemental data for MDS analyses.

Usage

```
mdsSupplementalElementsPreProcessing(SUP.DATA = NULL, D = NULL, M =
NULL)
```

Arguments

`SUP.DATA`	A supplementary data matrix.
`D`	The original (active) distance matrix that `SUP.DATA` is supplementary to.
`M`	masses from the original (active) analysis for `D`.

Value

a matrix that is preprocessed for supplementary projection in MDS.

Author(s)

Derek Beaton

`mdsTransform` *Transform data for MDS analysis.*

Description

Transform data for MDS analysis.

Usage

```
mdsTransform(D, masses)
```

Arguments

`D`	A distance matrix
`masses`	A vector or matrix of masses (see `computeMW`).

Value

`S`	a preprocessed matrix that can be decomposed.

Author(s)

Derek Beaton

`nominalCheck` *Checks if data are disjunctive.*

Description

Checks if data is in disjunctive (sometimes called complete binary) format. To be used with MCA (e.g., `epMCA`).

Usage

```
nominalCheck(DATA)
```

Arguments

`DATA`	A data matrix to check. This should be 0/1 disjunctive coded. `nominalCheck` just checks to make sure it is complete.

Value

If `DATA` are nominal, `DATA` is returned. If not, `stop` is called and execution halts.

Author(s)

Derek Beaton

`pause` *pause*

Description

A replication of MatLab pause function.

Usage

```
pause(x = 0)
```

Arguments

`x`	optional. If x>0 a call is made to `Sys.sleep`. Else, execution pauses until a key is entered.

Author(s)

Derek Beaton (but the pase of which is provided by Phillipe Brosjean from the R mailing list.)

References

Copied from:
https://stat.ethz.ch/pipermail/r-help/2001-November/

`pca.wine` *Six wines described by several assessors with rank attributes.*

Description

Six wines described by several assessors with rank attributes.

Usage

```
data(pca.wine)
```

Format

pca.wine$data: A data matrix with 6 wines (rows) from several assessors described by 11 attributes (columns). For use with `epPCA`.

References

Abdi, H., and Williams, L.J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2, 433-459.

See Also

mca.wine

`pcaSupplementaryColsPreProcessing` *Preprocessing for supplementary columns in PCA.*

Description

Preprocessing for supplementary columns in PCA.

Usage

```
pcaSupplementaryColsPreProcessing(SUP.DATA = NULL, center = TRUE,
scale = TRUE, M = NULL)
```

Arguments

`SUP.DATA`	A supplemental matrix that has the same number of rows as an active data set.
`center`	The center from the active data. NULL will center `SUP.DATA` to itself.
`scale`	The scale factor from the active data. NULL will scale (z-score) `SUP.DATA` to itself.
`M`	Masses from the active data.

Value

a matrix that has been preprocessed to project supplementary columns for PCA methods.

Author(s)

Derek Beaton

`pcaSupplementaryRowsPreProcessing` *Preprocessing for supplemental rows in PCA.*

Description

Preprocessing for supplemental rows in PCA.

Usage

```
pcaSupplementaryRowsPreProcessing(SUP.DATA = NULL, center = TRUE,
scale = TRUE, W = NULL)
```

Arguments

`SUP.DATA`	A supplemental matrix that has the same number of columns as an active data set.
`center`	The center from the active data. NULL will center `SUP.DATA` to itself.
`scale`	The scale factor from the active data. NULL will scale (z-score) `SUP.DATA` to itself.
`W`	Weights from the active data.

Value

a matrix that has been preprocessed to project supplementary rows for PCA methods.

Author(s)

Derek Beaton

`pickSVD` *Pick which generalized SVD (or related) decomposition to use.*

Description

This function is an interface for the user to a general SVD or related decomposition. It provides direct access to `svd` and `eigen`. Future decompositions will be available.

Usage

```
pickSVD(datain, is.mds = FALSE, decomp.approach = "svd", k = 0)
```

Arguments

`datain`	a data matrix to decompose.
`is.mds`	a boolean. TRUE for a MDS decomposition.
`decomp.approach`	a string. 'svd' for singular value decomposition, 'eigen' for an eigendecomposition. All approaches provide identical output. Some approaches are (in some cases) faster than others.
`k`	numeric. The number of components to return.

Value

A list with the following items:

`u`	Left singular vectors (rows)
`v`	Right singular vectors (columns)
`d`	Singular values
`tau`	Explained variance per component

Author(s)

Derek Beaton

`print.epCA` *Print Correspondence Analysis (CA) results*

Description

Print Correspondence Analysis (CA) results

Usage

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

Arguments

`x`	an list that contains items to make into the epCA class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

`print.epGraphs` *Print epGraphs results*

Description

Print epGraphs results

Usage

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

Arguments

`x`	an list that contains items to make into the epGraphs class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

See Also

`epGraphs`

`print.epMCA` *Print Multiple Correspondence Analysis (MCA) results*

Description

Print Multiple Correspondence Analysis (MCA) results

Usage

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

Arguments

`x`	an list that contains items to make into the epMCA class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

`print.epMDS` *Print Multidimensional Scaling (MDS) results*

Description

Print Multidimensional Scaling (MDS) results

Usage

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

Arguments

`x`	an list that contains items to make into the epMDS class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

`print.epPCA` *Print Principal Components Analysis (PCA) results*

Description

Print Principal Components Analysis (PCA) results

Usage

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

Arguments

`x`	an list that contains items to make into the epPCA class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

`print.epSVD` *Print results from the singular value decomposition (SVD) in ExPosition*

Description

Print results from the singular value decomposition (SVD) in ExPosition

Usage

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

Arguments

`x`	an list that contains items to make into the epSVD class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

`print.expoOutput` *Print results from ExPosition*

Description

Print results from ExPosition

Usage

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

Arguments

`x`	an list that contains items to make into the expoOutput class.
`...`	inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton and Cherise Chin-Fatt

See Also

`epPCA`, `epGraphs`

`rowNorms` *Normalize the rows of a matrix.*

Description

This function will normalize the rows of a matrix.

Usage

```
rowNorms(X, type = NULL, center = FALSE, scale = FALSE)
```

Arguments

`X`	Data matrix
`type`	a string. Type of normalization to perform. Options are `hellinger`, `ca`, `z`, `other`
`center`	optional. A vector to center the columns of X.
`scale`	optional. A vector to scale the values of X.

Details

rowNorms works like `link{expo.scale}`, but for rows. Hellinger row norm via `hellinger`, Correspondence analysis analysis row norm (row profiles) via `ca`, Z-score row norm via `z`. `other` passes `center` and `scale` to `expo.scale` and allows for optional centering and scaling parameters.

Value

Returns a row normalized version of X.

Author(s)

Derek Beaton

`rvCoeff` *Perform Rv coefficient computation.*

Description

Perform Rv coefficient computation.

Usage

```
rvCoeff(Smat, Tmat, type)
```

Arguments

`Smat`	A square covariance matrix
`Tmat`	A square covariance matrix
`type`	DEPRECATED. Any value here will be ignored

Value

A single value that is the Rv coefficient.

Author(s)

Derek Beaton

References

Robert, P., & Escoufier, Y. (1976). A Unifying Tool for Linear Multivariate Statistical Methods: The RV-Coefficient. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *25*(3), 257–265.

`snps.druguse` *Small data set for Partial Least Squares-Correspondence Analysis*

Description

The data come from a larger study on marijuauna dependent individuals (see Filbey et al., 2009) and are illustrated in Beaton et al., 2013.
The data contain 2 genetic markers and 3 additional drug use questions from 50 marijuauna dependent individuals.

Usage

```
data(snps.druguse)
```

Format

snps.druguse$DATA1: Fifty marijuana dependent participants indicated which, if any, other drugs they have ever used.
snps.druguse$DATA2: Fifty marijuana dependent participants were genotyped for the COMT and FAAH genes.

Details

In snps.druguse$DATA1:
e - Stands for ecstacy use. Responses are yes or no. cc - Stands for crack/cocaine use. Responses are yes or no. cm - Stands for crystal meth use. Responses are yes or no.
In snps.druguse$DATA2:
COMT - Stands for the COMT gene. Alleles are AA, AG, or GG. Some values are NA. FAAH - Stands for FAAH gene. Alleles are AA, CA, CC. Some values are NA.

References

Filbey, F. M., Schacht, J. P., Myers, U. S., Chavez, R. S., & Hutchison, K. E. (2009). Marijuana craving in the brain. Proceedings of the National Academy of Sciences, 106(31), 13016 – 13021.

Beaton D., Filbey F. M., Abdi H. (2013, in press). Integrating Partial Least Squares Correlation and Correspondence Analysis for Nominal Data. In Abdi H, Chin W, Esposito-Vinzi V, Russolillo G, Trinchera L. *Proceedings in Mathematics and Statistics (Vol. 56): New Perspectives in Partial Least Squares and Related Methods*. New York, NY: Springer-Verlag.

`sqrt_mat` *sqrt_mat*

Description

sqrt_mat performs the square root of a matrix only for square symmetric matrices This function should not be used directly.

Usage

```
sqrt_mat(X)
```

Arguments

`X`	a matrix that is square and symmetric

Author(s)

Derek Beaton

`supplementalProjection` *Supplemental projections.*

Description

Performs a supplementary projection across ExPosition (and related) techniques.

Usage

```
supplementalProjection(sup.transform = NULL, f.scores = NULL, Dv =
NULL, scale.factor = NULL, symmetric = TRUE)
```

Arguments

`sup.transform`	Data already transformed for supplementary projection. That is, the output from: `caSupplementalElementsPreProcessing`, `mdsSupplementalElementsPreProcessing`, `pcaSupplementaryColsPreProcessing`, or `pcaSupplementaryRowsPreProcessing`.
`f.scores`	Active factor scores, e.g., res$ExPosition.Data$fi
`Dv`	Active singular values, e.g., res$ExPosition.Data$pdq$Dv
`scale.factor`	allows for a scaling factor of supplementary projections. Primarily used for MCA supplemental projections to a correction (e.g., Benzecri).
`symmetric`	a boolean. Default is TRUE. If FALSE, factor scores are computed with asymmetric properties (for rows only).

Value

A list with:

`f.out`	Supplementary factor scores.
`d.out`	Supplementary square distances.
`r.out`	Supplementary cosines.

Author(s)

Derek Beaton

See Also

It is preferred for users to compute supplemental projections via `supplementaryRows` and `supplementaryCols`. These handle some of the nuances and subtleties due to the different methods.

`supplementaryCols` *Supplementary columns*

Description

Computes factor scores for supplementary measures (columns).

Usage

```
supplementaryCols(SUP.DATA, res, center = TRUE, scale = TRUE)
```

Arguments

`SUP.DATA`	a data matrix of supplementary measures (must have the same observations [rows] as active data)
`res`	ExPosition or TExPosition results
`center`	a boolean, string, or numeric. See `expo.scale`
`scale`	a boolean, string, or numeric. See `expo.scale`

Details

This function recognizes the class types of: `epPCA`, `epMDS`, `epCA`, `epMCA`, and `TExPosition` methods. Further, the function recognizes if Hellinger (as opposed to row profiles; in CA, MCA and DICA) were used.

Value

A list of values containing:

`fjj`	factor scores computed for supplemental columns
`djj`	squared distances for supplemental columns
`rjj`	cosines for supplemental columns

Author(s)

Derek Beaton

`supplementaryRows` *Supplementary rows*

Description

Computes factor scores for supplementary observations (rows).

Usage

```
supplementaryRows(SUP.DATA, res)
```

Arguments

`SUP.DATA`	a data matrix of supplementary observations (must have the same measures [columns] as active data)
`res`	ExPosition or TExPosition results

Details

This function recognizes the class types of: `epPCA`, `epMDS`, `epCA`, `epMCA` and `TExPosition` methods. Further, the function recognizes if Hellinger (as opposed to row profiles; in CA, MCA and DICA) were used.

Value

A list of values containing:

`fii`	factor scores computed for supplemental observations
`dii`	squared distances for supplemental observations
`rii`	cosines for supplemental observations

Author(s)

Derek Beaton

`wines2007` *Six wines described by 3 assessors.*

Description

How six wines are described by 3 assessors across various flavor profiles, totaling 10 columns.

Usage

```
data(wines2007)
```

Format

wines2007$data: A data set with 3 experts (studies) describing 6 wines (rows) using several variables using a scale from 1 to 7 with a total of 10 measures (columns).
wines2007$table: A data matrix which identifies the 3 experts (studies).

References

Abdi, H., & Valentin, D. (2007). STATIS. In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics*. Thousand Oaks (CA): Sage. pp. 955-962.

`wines2012` *Wines Data from 12 assessors described by 15 flavor profiles.*

Description

10 experts who describe 12 wines using four variables (cat-pee, passion fruit, green pepper, and mineral) considered as standard, and up to two additional variables if the experts chose.

Usage

```
data(wines2012)
```

Format

wines2012$data: A data set with 10 experts (studies) describing 12 wines (rows) using four to six variables using a scale from 1 to 9 with a total of 53 measures (columns).
wines2012$table: A data matrix which identifies the 10 experts (studies).
wines2012$supplementary: A data matrix with 12 wines (rows) describing 4 Chemical Properties (columns).

References

Abdi, H., Williams, L.J., Valentin, D., & Bennani-Dosse, M. (2012). STATIS and DISTATIS: Optimum multi-table principal component analysis and three way metric multidimensional scaling. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4, 124-167.

`words` *Twenty words described by 2 features.*

Description

Twenty words “randomly” selected from a dictionary and described by two features: length of word and number of definitions.

Usage

```
data(words)
```

Format

words$data: A data matrix with 20 words (rows) described by 2 attributes (columns). For use with `epPCA`.

References

Abdi, H., and Williams, L.J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2, 433-459.

Index