Package ‘syndRomics’

June 26, 2020

Type Package

Title Component loading interpretation and visualization with syndromic plots

Version 0.0.1.9000

Author Abel Torres Espin, Austin Chou

Maintainer Abel Torres Espin <atpspin@gmail.com>

Description
The syndRomics package offers functionalities that aid in the steps for syndromics analysis by means of principal component analysis (PCA). The key steps in the syndromic analysis by PCA workflow are:
1) component selection, the selection of PCs or syndromic components to keep for subsequent analysis,
2) component interpretation, the determination of the meaning of the extracted syndromic components,
3) component stability, the study of the syndromic component robustness for better generalization, and
4) component visualization.

License MIT

Encoding UTF-8

LazyData false

Imports ggrepel,
ggnewscale,
pracma,
png,
boot,
rlang,
grid

RoxygenNote 7.1.0

Depends R (>= 3.5.0), dplyr, ggplot2, stringr, tidyr

Suggests knitr, rmarkdown, Gifi

VignetteBuilder knitr

R topics documented:

`barmap_commun` *Barmap of communalities*

Description

Plot a Barmap of the communalities from a PCA solution given the first ndim PCs.

Usage

```
barmap_commun(
  pca,
  pca_data,
  ndim = 10,
  load_list = NULL,
  conf = 0.95,
  plot_original = T,
  plot_list_center = F,
  plot_title = "Communalities",
  legend_title = "communa.",
  text_values = F,
  text_size = 2,
  var_order = "increasing",
  vars = NULL
)
```

Arguments

`pca`	Object of class *prcomp*, *princals*, or *data.frame*. If object is a *prcomp* or *princals* object, *pca_data* is required, and the loadings will be extracted. If object is a data.frame object, the dataframe needs to be formatted as: first column named *Variables* and all other columns corresponding to a PC. One row per variable. The values are the loadings.
`pca_data`	Data passed to the *prcomp* or *princals* function.
`ndim`	Numeric. Number of PCs to plot

`load_list`	List. List of loading matrices used to plot percentile confidence intervals around the average. If NULL, no error bars are ploted. Default=NULL
`conf`	Numeric. Confidence level used when *load_list* is provided.
`plot_original`	Boolean. Whether to plot the communalities obtained from the values passed to *pca* and *pca_data*.
`plot_list_center`	Boolean. Whether to plot the average communalities obtained from *load_list*.
`plot_title`	String. Title of the plot.
`legend_title`	String. Title of the legend.
`text_values`	Boolean. Whether to plot the values of the communalities or not. Default=FALSE
`text_size`	Numeric. Size of the text_values.
`var_order`	Character. Specify the order of the variables in the plot by the communalitys values, starting at 12 o'clock and moving counterclockwise. Possible values: 'abs decreasing': plot by decreasing absolute value; 'abs increasing': plot by increasing absolute value; 'decreasing'; or 'increasing'.
`vars`	Character vector. Variables will be ordered as the provided variable names. Non-specified variables will be excluded from the plot. By default variables are ordered in alphabetically by ggplot.

Value

Returns a *ggplot2* object.

Author(s)

Abel Torres Espin

`barmap_loading` *Barmap of standardized loadings*

Description

Plot a Barmap of the standardized loadings from a PCA solution.

Usage

```
barmap_loading(
  pca,
  pca_data,
  ndim = 10,
  cutoff = 0.5,
  arbitrary_var = NULL,
  load_list = NULL,
  conf = 0.95,
  plot_list_original = F,
  plot_list_center = F,
  plot_title = "Standardized loadings",
  legend_title = "s. loading",
  text_values = F,
```

```
  star_values = T,
  text_size = 2,
  plot_cutoff = T,
  vars = NULL
)
```

Arguments

pca
: Object of class *prcomp*, *princals*, or *data.frame*. If object is a *prcomp* or *princals* object, *pca_data* is required, and the loadings will be extracted. If object is a data.frame object, the dataframe needs to be formatted as: first column named *Variables* and all other columns corresponding to a PC. One row per variable. The values are the loadings.

pca_data
: Data passed to the *prcomp* or *princals* function.

ndim
: Numeric. Number of PCs to plot

cutoff
: Numeric or numeric vector of length *ndim*. Value of the loadings threshold (i.e. |loadings| >= cutoff) to plot with stars. Default = 0.5

arbitrary_var
: Character or character vector with the names of the variables where the loadings should be plot as absolute values. This is the case for categorical variables in categorical PCA where variables do not have direction.

load_list
: List. List of loading matrices used to plot percentile confidence intervals around the average. If NULL, no error bars are ploted. Default=NULL

conf
: Numeric. Confidence level used when *load_list* is provided.

plot_list_original
: Boolean. Whether to plot the loadings obtained from the values passed to *pca* and *pca_data* when *load_list* is not NULL.

plot_list_center
: Boolean. Whether to plot the average loadings obtained from *load_list*.

plot_title
: String. Title of the plot. 'Standardized loadings' by default.

legend_title
: String. Title of the legend. 's. loading' by default.

text_values
: Boolean. Whether to plot the values of the loadings or not. Default=TRUE

star_values
: Boolean. Whether to plot a star in |loadings|>=cutoff. Only relevant if *text_values*=FALSE. Default=FALSE

text_size
: Numeric. Size of the text_values.

plot_cutoff
: Boolean. Whether to plot the cutoff lines or not.

vars
: Character vector. Variables will be ordered as the provided variable names. Non-specified variables will be excluded from the plot. By default variables are ordered in alphabetically by ggplot.

Value

Returns a *ggplot2* object.

Author(s)

Abel Torres Espin

`boot_pca_sample` *Bootstrapping PCA sample*

Description

This function is passed to the *statistic* argument of the *boot* function from the **boot** package. It generates a bootstrapped PCA sample, extracting the standardized loadings using linear PCA (*prcomp()*) or non-linear PCA (*Gifi::princals()*).

Usage

```
boot_pca_sample(data, indices, pca, original_loadings, ndim, pb = NULL)
```

Arguments

`data`	The data argument passed from the *boot* function
`indices`	The indices of the rows for the bootstrapped sample passed from the *boot* function
`pca`	Object of class *prcomp*, *princals* or *data.frame*. If the object is a *prcomp* or *princals* object, *pca_data* is required and the loadings will be extracted. If the object is a data.frame object, the dataframe needs to be formatted as: first column named *Variables* and all other columns corresponding to a PC. One row per variable. The values are the loadings.
`original_loadings`	The data.frame containing the standardized loadings of the original sample.
`ndim`	Numeric. Number of PCs to save (1 to ndim).
`pb`	Object of class "Progress" "R6" generated by *dplyr::progress_estimated()*. Not required.

Details

A major problem of performing bootstrapping procedures in PCA is what is known as sign reflection: the change of the sign (positive/negative) on the component loadings in a PC given slight variations in the data. In addition, component/factor translocation can occur, meaning that the position of one component can change in a particular PCA solution, especially when two components have similar VAF. Another problem on performing PCAs with variations in the data is the possibility of rotation indeterminacy when the PCA solution of a resampled data presents with a different rotation of the original PCA solution. These issues generate artificially biased bootstrapped distributions, reducing the performance of the procedure. We have implemented a step of procrustes rotation between the original loadings (target) and the bootstrapped sample, which has previously been demonstrated to be a reasonable method to deal with such issues. The procrustes rotation is obtained by the *pracma::procrustes()* function.

Value

A matrix with one sample bootstrapped standardized loadings. This will be returned to the *boot* function when *pc_stability* function is called.

Author(s)

Abel Torres Espin

References

1. Linting M, Meulman JJ, Groenen PJF, van der Kooij AJ. Stability of nonlinear principal components analysis: An empirical study using the balanced bootstrap. Psychol Methods. 2007;12(3):359–79.
2. Babamoradi H, van den Berg F, Rinnan Å. Bootstrap based confidence limits in principal component analysis — A case study. Chemom Intell Lab Syst. 2013 Jan 15;120:97–105.
3. Timmerman ME, Kiers HAL, Smilde AK. Estimating confidence intervals for principal component loadings: a comparison between the bootstrap and asymptotic results. Br J Math Stat Psychol. 2007 Nov;60(Pt 2):295–314

`component_similarity` *Extracts different component/factor similarity (matching) indices*

Description

Given a list of loadings for a set of factors or components, computes the Pearson's coefficient of determination (r), the coefficient of congruence (CC), Cattell's S-statistic, and the root mean square error (RMSE) between them.

Usage

```
component_similarity(
  load.list,
  s_cut_off = 0.4,
  ndim = 5,
  similarity_metric = "all"
)
```

Arguments

`load.list` List of factors to match. Each element of the list is a matrix *p x m* where p are the variables and m the factors or components. All matrices must have variables and components in the same order.

`s_cut_off` Numerical value for the loading cut off used to determine if a variable is silent or not in Cattell's terms.

`ndim` Numeric. Number of PCs to compute the similarity from. Default=5

`similarity_metric` Character or character vector. Possible values are "cc_index" (congruence coefficient), "r_correlation" (Pearson's r), "rmse" (root mean squared error), "s_index' (Cattell's s metric), or "all". Default="all". See below for details on calculations.

Details

This function is internally called by *pc_stability()*. Each metric is computed using an external function:

"cc_index"*(extract_cc()) function* The congurence coefficient is calculated as:

$$CC_{x,y} = sum(x_i X y_i)/sqrt(sum(x_i^2) X sum(yi)^2)$$

Where x_i and y_i are the loadings of the variable *i* on the component or factor *x* and *y* respectively. CC is equivalent to the cosine of the angle between two vectors (the cosine similarity metric) and has a numerical range from -1 to 1. The sign of a component is arbitrary and can be flipped without affecting its interpretation. Here we consider the absolute value of CC (0 to 1). The closer the CC is to 1, the more similar the two components are. (see refs 1,2)

"r_correlation"*(cor()) function* The Pearson's r between two vectors of component loadings has also been used as a similarity metric for component/factor matching(ref 3). We calculate it here using the *cor()* function.

"rmse"*(extract_rmse()) function* RMSE has been also used as a metric for factor matching (see ref 3). It is calculated as:

$$RMSE_{x,y} = sqrt(sum((x_i - y_i)^2)/n)$$

Where *n* is the number of variables in both components *x* and *y*. A RMSE of 0 corresponds to a perfect match. The smaller the RMSE is, the more equivalent two components are.

"s_index"*(extract_s()) function* The s index was first suggested by Cattell et al. It is based on the factor mandate matrix (ref 4) where loadings are either 1 if a component is considered to act on a variable, called a salient variable, or 0 if not (forming the hyperplane space). Cattell's suggested an arbitrary ±0.1 cut-off to be considered as salient variables. In practice, one might want to alter the threshold depending on the experimental conditions.

Value

Returns a list of three objects. **Index_all** contains all the comparisons between all the elements of the *load.list*. In general, similarity is calculated between two matrices of loadings, but the user can extract the all the comparisons in case length (load.list) is > 2. **index_mean** is the average of the similarity metrics between all the comparisons. It will be the same as the individual metric (index_all) when length(load.list)==2, because there is only a single comparison made in that scenario. **index_sd** is the standard deviation of the index in case length (load.list) is > 2.

Author(s)

Abel Torres Espin

References

1. Burt C. The Factorial Study of Temperamental Traits. Br J Stat Psychol. 1948;1(3):178–203.

2. Tucker, L. R. A method for synthesis of factor analysis studies. Personnel Research Section Report No.984. Washington D.C.: Department of the Army.; 1951.

3. Guadagnoli E, Velicer W. A Comparison of Pattern Matching Indices. Multivar Behav Res. 1991 Apr;26(2):323–43

4. Cattell RB, Balcar KR, Horn JL, Nesselroade JR. Factor Matching Procedures: an Improvement of the s Index; with Tables. Educ Psychol Meas. 1969 Dec;29(4):781–92

`extract_cc` *Extracts Coefficient of congruence*

Description

Given two vectors, generates the coefficient of congruence between them. This is equivalent to the cosine of the angle between both vectors.

Usage

```
extract_cc(vector1, vector2)
```

Arguments

`vector1`	First numerical vector for the calculation.
`vector2`	Second numerical vector for the calculation.

Details

The congruence coefficient is calculated as:

$$CC_{x,y} = sum(x_i X y_i)/sqrt(sum(x_i^2) X sum(yi)^2)$$

Where x_i and y_i are the loadings of the variable *i* on the component or factor *x* and *y* respectively. CC is equivalent to the cosine of the angle between two vectors (the cosine similarity metric) and has a numerical range from -1 to 1. The sign of a component is arbitrary and can be flipped without affecting its interpretation. Here we consider the absolute value of CC (0 to 1). The closer the CC is to 1, the more similar the two components are.

Value

Returns the coefficent of congruence (CC) between vector1 and vector2.

Author(s)

Abel Torres Espin

References

1. Burt C. The Factorial Study of Temperamental Traits. Br J Stat Psychol. 1948;1(3):178–203.
2. Tucker, L. R. A method for synthesis of factor analysis studies. Personnel Research Section Report No.984. Washington D.C.: Department of the Army.; 1951.

`extract_loadings` *extract loadings*

Description

This is a wrapper function for *stand_loadings()* with added functionalities such as error breakers that is used by most functions in the package.

Usage

```
extract_loadings(pca, pca_data)
```

Arguments

`pca`	Object of class *prcomp*, *princals* or *data.frame*. If the object is a *prcomp* or *princals* object, *pca_data* is required, and the loadings will be extracted. If the object is a data.frame object, the dataframe needs to be formatted as: first column named *Variables* and all other columns corresponding to a PC. One row per variable. The values are the loadings.
`pca_data`	Data passed to the *prcomp* or *princals* function.

Author(s)

Abel Torres Espin

`extract_rmse` *Extracts root mean square error (RMSE)*

Description

Given two vectors of loadings, computes the root mean square between them.

Usage

```
extract_rmse(vector1, vector2)
```

Arguments

`vector1`	First numerical vector of loadings for the calculation.
`vector2`	Second numerical vector of loadings for the calculation.

Details

RMSE has been also used as a metric for factor matching (see ref). It is calculated as:

$$RMSE_{x,y} = sqrt(sum((x_i - y_i)^2)/n)$$

Where n is the number of variables in both components x and y. A RMSE of 0 corresponds to a perfect match. The smaller the RMSE is, the more equivalent two components are.

Value

Returns the root mean square (RMS) between vector1 and vector2.

Author(s)

Abel Torres Espin

References

Guadagnoli E, Velicer W. A Comparison of Pattern Matching Indices. Multivar Behav Res. 1991 Apr;26(2):323–43

`extract_s` *Extracts Cattell's S-statistic*

Description

Given two vectors of loadings, computes the Cattell's S-statistic between them with a specified cut off.

Usage

```
extract_s(vector1, vector2, cut_off = 0.1)
```

Arguments

`vector1`	First numerical vector of loadings for the calculation.
`vector2`	Second numerical vector of loadings for the calculation.
`cut_off`	Numerical value for the loading cut off to determine if a variable is silent or not in Cattell's terms. Default = 0.1

Details

The s index was first suggested by Cattell et al. It is based on the factor mandate matrix (see ref) where loadings are either 1 if a component is considered to act on a variable, called a salient variable, or 0 if not (forming the hyperplane space). Cattell's suggested an arbitrary ±0.1 cut-off to be considered as salient variables. In practice, one might want to alter the threshold depending on the experimental conditions.

Value

Returns the Cattell's S-statistic between vector1 and vector2 at cut_off.

Author(s)

Abel Torres Espin

References

Cattell RB, Balcar KR, Horn JL, Nesselroade JR. Factor Matching Procedures: an Improvement of the s Index; with Tables. Educ Psychol Meas. 1969 Dec;29(4):781–92

`extract_syndromic_plot` *Extract the syndromic plot*

Description

Extract the syndromic plot of a given list of loadings and specified pc.

Usage

```
extract_syndromic_plot(
  load_df,
  pc,
  cutoff = 0.5,
  VAF,
  arbitrary_var = NULL,
  arrow_size_multi = 10,
  repel = T,
  plot_legend = T,
  text_size = 9,
  var_order = "abs decreasing"
)
```

Arguments

`load_df`	data.frame with the loadings. Format: one column named *Variable* and a column for each PC. It can be extracted from a *prcomp* or *princals* object using *stand_loadings()*.
`pc`	String. Name of the PC in *load_df* to plot.
`cutoff`	Numeric. Value of the loadings threshold (i.e. |loadings| > cutoff) to plot. Default = 0.5
`VAF`	String. Text for the center of the syndromic plot. The text generally corresponds to the variance accounted for (VAF) of the PC. If pca is the results of *prcomp* or *princals*, VAF is internally calculated.
`arbitrary_var`	Character or character vector with the names of the variables where loadings should be plot as absolute values. This is the case for categorical variables in categorical PCA where variables do not have direction.
`arrow_size_multi`	Numeric. Controls the size of the arrows proportional to the loading. Default=10
`repel`	Boolean. Whether to repel the text for preventing text overlap. Default=TRUE
`plot_legend`	Boolean. Whether to plot the legend or not. Default=TRUE
`text_size`	Numeric. Controls for the size of the text. Default=9
`var_order`	Character, character vector or numeric vector. Specify the order of the variables in the plot by the loading values, starting at 12 o'clock and moving counterclockwise. Possible values: 'abs decreasing': plot by decreasing absolute value; 'abs increasing': plot by increasing absolute value; 'decreasing'; or 'increasing'. A vector can be specified with a custom order of the variables to plot. If a numeric vector is provided, the variables will be ordered as specified by the numbers. If

a character vector is provided, the variables will be ordered in the same order as the variable names provided. In case of vector, non-specified variables will be excluded from the plot.

Value

A *ggplot2* object of the syndromic plot.

Author(s)

Abel Torres Espin

---

heatmap_loading *Heatmap of standardized loadings*

Description

Plot a heatmap of the standardized loadings from a PCA solution.

Usage

```
heatmap_loading(
  pca,
  pca_data,
  ndim = 10,
  cutoff = 0.5,
  arbitrary_var = NULL,
  plot_title = "Standardized loadings",
  legend_title = "s. loading",
  text_values = T,
  star_values = F,
  text_size = 2,
  vars = NULL
)
```

Arguments

`pca`	Object of class *prcomp*, *princals*, or *data.frame*. If object is a *prcomp* or *princals* object, *pca_data* is required, and the loadings will be extracted. If object is a data.frame object, the dataframe needs to be formatted as: first column named *Variables* and all other columns corresponding to a PC. One row per variable. The values are the loadings.
`pca_data`	Data passed to the *prcomp* or *princals* function.
`ndim`	Numeric. Number of PCs to plot
`cutoff`	Numeric or numeric vector of length *ndim*. Value of the loadings threshold (i.e. |loadings| >= cutoff) to plot with stars. Default = 0.5
`arbitrary_var`	Character or character vector with the names of the variables where the loadings should be plot as absolute values. This is the case for categorical variables in categorical PCA where variables do not have direction.
`plot_title`	String. Title of the plot. 'Standardized loadings' by default.

`legend_title`	String. Title of the legend. 's. loading' by default.
`text_values`	Boolean. Whether to plot the values of the loadings or not. Default=TRUE
`star_values`	Boolean. Whether to plot a star in |loadings|>=cutoff. Only relevant if *text_values*=FALSE. Default=FALSE
`text_size`	Numeric. Size of the text_values.
`vars`	Character vector. Variables will be ordered as the provided variable names. Non-specified variables will be excluded from the plot. By default variables are ordered in alphabetically by ggplot.

Value

Returns a *ggplot2* object.

Author(s)

Abel Torres Espin

`pc_stability` *PC stability by nonparametric bootstrapping*

Description

Extract nonparametric estimate of the standardized loadings and the confident region by means of bootstrapping. This function uses the *boot* function from the **boot** package.

Usage

```
pc_stability(
  pca,
  pca_data,
  ndim = 3,
  B = 1000,
  sim = "ordinary",
  communalities = T,
  test_similarity = T,
  similarity_metric = "all",
  s_cut_off = 0.1,
  ci_type = "bca",
  conf = 0.95,
  barmap_plot = T,
  ...
)
```

Arguments

`pca`	Object of class *prcomp*, *princals*, or *data.frame*. If object is a *prcomp* or *princals* object, *pca_data* is required, and the loadings will be extracted. If object is a data.frame object, the dataframe needs to be formatted as: first column named *Variables* and all other columns corresponding to a PC. One row per variable. The values are the loadings.

`pca_data`	Data passed to the *prcomp* or *princals* function.
`ndim`	Numeric. Number of PCs (1 to *ndim*) to run the analysis on. Default = 3.
`B`	Numeric. Number of bootstrapped samples passed to *boot*.
`sim`	Character. Determines the bootstrapping method for *boot*. From *boot*: A character string indicating the type of simulation required. Possible values are "ordinary" (the default), "parametric", "balanced", "permutation", or "antithetic". Default="ordinary".
`communalities`	Boolean. Whether to compute and return communalities.
`test_similarity`	Boolean. Whether to compute component similarity by the specified *similarity_metric* and the nonparametric percentile confident interval. See *?component_similarity* for more information. Default=TRUE
`similarity_metric`	character or character vector. Specify the similarity metric to use when *test_similarity*=TRUE. Possible values are "cc_index" (congruence coefficient), "r_correlation" (Pearson's r), "rmse" (root mean squared error), "s_index' (Cattell's s metric), or "all". See *?component_similarity* for more details on the available metrics. Default="all".
`s_cut_off`	Numeric. This is the loading cut off used to determine if a variable is silent or not in Cattell's terms. See *?extract_s* for more information. Default=0.1.
`ci_type`	Character. Type of confidence interval to compute. This argument is passed to the *boot.ci* function from the *boot* package. See *?boot.ci* for options. Given that the BCA method has demonstrated good performance for bootstrapping PCAs, we have set 'bca' as the default. See ref for more details.
`conf`	Numeric. Level of confidence region for the confidence interval. E.g. 0.95 generates 95CI. Default=0.95
`barmap_plot`	Boolean. Whether to generate a barmap plot of the bootstrapped loadings or not. See *?barmap_loadings* for details. Default=TRUE
`...`	Other arguments passed to the *barmap_loadings* or *barmap_commun* function when *barmap_plot*=TRUE

Details

The number of bootstrap samples is set to 1000 by default, as it has been shown to be a robust number in most conditions of data complexity and sample size. The user must be careful on setting such number too low which would reduce the performance of the approximation. However, values that are too high might unnecessarily increase computing time with little gain (REFs).

Value

Returns a list object.

boot_sample A list of length B containing the resampled loadings.

results **boot_mean** A numeric matrix containing the mean loadings of the bootstrapped sample.

- **ci_low and ci_high** A numeric matrix with the lower and upper CI respectively.

pc_similarity A list of results when *test_similarity*=TRUE.

- **similarity_mean** A numerix matrix with the mean of the chosen similarity metric.
- **similarity_ci_low and similarity_ci_high** A numeric matrix with the lower and upper CI respectively.

boot_barmap_loadings A ggplot2 object with a barmap plot of loadings of the bootstrapped solution.

boot_barmap_loadings A ggplot2 object with a barmap plot of communalities of the bootstrapped solution.

Author(s)

Abel Torres Espin

References

Efron B. Better Bootstrap Confidence Intervals. J Am Stat Assoc. 1987 Mar 1;82(397):171–85.

`permut_pca` *Permutation PCA*

Description

Generates a list of permutation samples from a PCA that can be used for nonparametric permutation test (see *permut_pc_test*).

Usage

```
permut_pca(
  pca,
  pca_data,
  P = 1000,
  ndim,
  output = "VAF",
  pb = NULL,
  perm.method = "permD"
)
```

Arguments

`pca`	Object of class *prcomp*, *princals* or *data.frame*. If the object is a *prcomp* or *princals* object, *pca_data* is required, and the loadings will be extracted. If the object is a data.frame object, the data.frame needs to be formatted as: first column named *Variables* and all other columns corresponding to a PC. One row per variable. The values are the loadings.
`pca_data`	Data passed to the *prcomp* or *princals* function.
`P`	Numeric. Number of permutations to run calling the *permuted_pca* function. Default=1000
`ndim`	Numeric. Number of PCs to save (1 to ndim).
`output`	Character. Determines the output to compute. Possible values are Variance accounted for ("VAF") or the standardized loadings ("s.loadings"). Default="VAF"
`pb`	Object of class "Progress" "R6" generated by *dplyr::progress_estimated()*. Not required.

`perm.method` Character determining the permutation method to use as in Linting et al., 2011. "permD" (Buja & Eyuboglu, 1992; Linting et al., 2011) where variables as permuted independently and concomitantly as opposite of the "permV" permutation strategy (Linting et al., 2011) where variables are permuted one at the time. If statistic is set to "VAF", *perm.method* "permD" will be used. Default="permV"

Details

This is a helper function internally called by *permut_pca_test()* to produce P permutations of the given output of the *prcomp()* or the *princals()* functions. It returns a list of the results of permuting the data, conducting a PCA and extracting either the VAF or the standardized loadings for each P.

Value

Returns a list of objects in which each element is a permutated PCA solution of the specified output.

Author(s)

Abel Torres Espin

References

1. Buja A, Eyuboglu N. Remarks on Parallel Analysis. Multivar Behav Res. 1992 Oct 1;27(4):509–40
2. Linting M, van Os BJ, Meulman JJ. Statistical Significance of the Contribution of Variables to the PCA solution: An Alternative Permutation Strategy. Psychometrika. 2011 Jul 1;76(3):440–60

`permut_pc_test` *permutation test of PCA*

Description

Compute a nonparametric permutation test for a PCA solution. Two options are possible: hypothesis testing of the total variance accounted for (VAF) for each component as in XXX ref, and hypothesis testing of the standardized loadings as in XXX ref.

Usage

```
permut_pc_test(
  pca,
  pca_data,
  P = 1000,
  ndim = 3,
  statistic = "VAF",
  conf = 0.95,
  adj.method = "BH",
  perm.method = "permV",
  plot = T,
  ...
)
```

Arguments

`pca`	Object of class *prcomp*, *princals*, or *data.frame*. If object is a *prcomp* or *princals* object, *pca_data* is required, and the loadings will be extracted. If object is a data.frame object, the dataframe needs to be formatted as: first column named *Variables* and all other columns corresponding to a PC. One row per variable. The values are the loadings.
`pca_data`	Data passed to the *prcomp* or *princals* function.
`P`	Numeric. Number of permutations to run calling the *permuted_pca* function. Default=1000
`ndim`	Numeric. Number of PCs (1 to *ndim*) to run the analysis on. D
`statistic`	Character. Determines the statistic to compute. Possible values are Variance accounted for ("VAF") or the standardized loadings ("s.loadings"). Default="VAF"
`conf`	Numeric. Level of confidence region for the confidence interval. E.g. 0.95 generates 95CI. Default=0.95
`adj.method`	Character passed to the *stats::p.adjust()* to adjust the p value for multiple comparisons. See *?p.adjust.methods*. Default="BH"
`perm.method`	Character determining the permutation method to use as in Linting et al., 2011. "permD" (Buja & Eyuboglu, 1992; Linting et al., 2011) where variables as permuted independently and concomitantly (Fig. 2A) as opposite of the "permV" permutation strategy (Linting et al., 2011) where variables are permuted one at the time. If statistic is set to "VAF", *perm.method* "permD" will be used. Default="permV".
`plot`	Boolean. Whether to retrun a ggplot2 object containing a plot of "VAF", "loadings" or "communalities" as specified in *statistic*.
`...`	Other arguments passed to the *barmap_loadings* or *barmap_commun* function when *barmap_plot*=TRUE

Details

Nonparametric permutation for hypothesis testing of the VAF of component, the loadings or communalities have been studied (see refs). The hypothesis test is defined as: H(null): PC metric (either VAF or loading) is indistinguishable from a random generation H(alternative): PC metric (either VAF or loading) is different from random The null distribution is generated by permuting the values of each variable several times (P) and re-running the PCA on each permuted sample. Confidence intervals of the permuted distribution (null distribution) are calculated using the percentile method. The p values are calculated as p = ((q+1))⁄((P+1)), where q is the number of times the chosen metric is higher in the permuted distribution than in the original PCA solution and P is the number of permutations. The user should note that the lowest p value that can be calculated is dependent on P. As an example, if P is set to a value of 10 (a relatively low value), the smallest p value that can be detected is 0.09, considering q=0. Accordingly, P should be set high enough to reach the desired floor p value. By default, we have set the number of permutations to 1000 (smallest p value approximately equal to 0.001 as a result) as this has been shown to be high enough for approximating the null distribution in most cases.

Permutation test of the loadings as in (Buja & Eyuboglu, 1992; Peres-Neto et al., 2003) that can serve to determine the loading threshold, where the variables are permuted simultaneously and concomitantly. Linting et al., designed and tested an strategy where only one variable is permuted at the time, showing great results in determining the contribution of variables using communalities (Linting et al., 2011). This method has resulted in better determination of the significant contribution of variables on the PCA solution with higher statistical power and proper type I error, and therefore has been incorporated in the package as the suggested method for loadings and communalities. Following Linting et al., terminology, user can specify the permutation strategy for the loadings as one variable at the time (*permV*, as in Linting et al., 2011) or as all the variable together (*permD*, as in Buja & Eyuboglu, 1992; Peres-Neto et al., 2003).

Value

List containing the following objects

perm.methods Character for the used permutation method

per_list List of P loading matrices for the P permuted samples

statistic Character for the used statistic

adj.method Character for the used method for adjusting p values

per_plot ggplot2 object with the plot of VAF, loadings or communalities depending on the specified statistic

results List of data.frame(s) with the results of the permutation test

Author(s)

Abel Torres Espin

References

1. Buja A, Eyuboglu N. Remarks on Parallel Analysis. Multivar Behav Res. 1992 Oct 1;27(4):509–40
2. Linting M, van Os BJ, Meulman JJ. Statistical Significance of the Contribution of Variables to the PCA solution: An Alternative Permutation Strategy. Psychometrika. 2011 Jul 1;76(3):440–60

`stand_loadings` *standardized loadings*

Description

Extract the standardized loadings from a *prcomp* object by correlating the PC scores and the original data.

Usage

```
stand_loadings(pca, pca_data)
```

Arguments

`pca`	Object of class *prcomp*, *princals*, or *data.frame*. If object is a *prcomp* or *princals* object, *pca_data* is required, and the loadings will be extracted. If object is a data.frame object, the dataframe needs to be formatted as: first column named *Variables* and all other columns corresponding to a PC. One row per variable. The values are the loadings.
`pca_data`	Data passed to the *prcomp* or *princals* function.

Details

The standardized loadings are calculated as the eigenvectors times the square roots of the respective eigenvalues and divided by the variable standard deviation (which is 1 in case of standardized PCA (from correlation matrix)). These are equivalent to the Pearson's correlation between the pca scores and the original dataset. This is the correlations of the PC with the variables and the same as the correlation of vector coefficients suggested by Jackson and Hearne in 1987.

This function extracts the standardized loadings from the output of the *prcomp()* or the *princals()* functions. In the case of the *prcomp()* solution, the standardized loadings are calculated as: s.loadings = eigenvectors x sqrt(eigenvalues) if the PCA was performed on the standardized (scaled to unit variance) data or s.loadings=(eigenvector x sqrt(eigenvalues))/S where S is the vector of the variables standard deviation. In the case of *princals()*, standardized loadings are returned directly in its output and therefore *stand_loadings()* returns those.

Value

A data.frame with the standardized loadings in the form of variables as rows and components as columns.

Author(s)

Abel Torres Espin

References

Jackson JE, Hearne FT. Relationships Among Coefficients of Vectors Used In Principal Components. Technometrics. 1973 Aug 1;15(3):601–10.

`syndromic_plot` *Syndromic plot*

Description

Extract the syndromic plots from a pca solution or from a table of loadings for all the specified PCs.

Usage

```
syndromic_plot(
  pca,
  pca_data = NULL,
  ndim = 3,
  cutoff,
  VAF,
  arbitrary_var = NULL,
  arrow_size_multi = 10,
  repel = T,
  plot_legend = T,
  text_size = 9,
  var_order = "abs decreasing",
  ...
)
```

Arguments

`pca`	Object of class *prcomp*, *princals*, or *data.frame*. If object is a *prcomp* or *princals* object, *pca_data* argument is required, and the loadings will be extracted. If object is a data.frame object, the dataframe needs to be formatted as: first column named *Variables* and all other columns corresponding to a PC. One row per variable. The values are the loadings.
`pca_data`	Data passed to the *prcomp* or *princals* function.
`ndim`	Numeric. Number of PCs to plot.
`cutoff`	Numeric or numeric vector of length *ndim*. Value of the loadings threshold to plot. If only one value is passed, the same cutoff will be used for all the PCs. If a vector passed, each value will be used for the corresponding PC.
`VAF`	If *pca* is from prcomp, VAF argument is not needed. Otherwise, VAF is a String vector with text for the centers of the syndromic plots. The text generally corresponds to the variance accounted for (VAF) for the respective PCs.
`arbitrary_var`	Character or character vector with the names of the variables where loadings should be plot as absolute values. This is the case for categorical variables in categorical PCA where variables do not have direction.
`arrow_size_multi`	Numeric. Controls the size of the arrows proportional to the loading. Default=10
`repel`	Boolean. Whether to repel the text for preventing text overlap. Default=TRUE
`plot_legend`	Boolean. Whether to plot the legend or not. Default=TRUE
`text_size`	Numeric. Controls for the size of the text. Default=9
`var_order`	Character. Specify the order of the variables in the plot by the loading values, starting at 12 o'clock and moving counterclockwise. Possible values: 'abs decreasing': plot by decreasing absolute value; 'abs increasing': plot by increasing absolute value; 'decreasing'; or 'increasing'.
`...`	Other arguments passed to *extract_syndromic_plot()*

Value

Returns a list of *ggplot2* objects with one element for each PC plot. It also renders and saves the plots as *.pdf in the working directory or in the specified *path*.

Examples

```
pca<-prcomp(mtcars, center = TRUE, scale. = TRUE)
syndromic_plot(pca, mtcars, cutoff = 0.65, ndim=1)
```

Index