Package 'PSA'

October 4, 2024

Title Principal Nested Simplices Analysis

Version 0.0.0.9000

Description What the package does (one paragraph).

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports stats,
utils,
tidyr,
dplyr,
graphics,
ggplot2 (>= 3.4.0),
GGally,
dendextend,
compositions

Depends R (>= 4.0.0)

Suggests testthat (>= 3.0.0)

Enhance ggtern

Contents

`compare_analysis` *Comparison of PSA and Benchmark Methods*

Description

A wrapper function of PSA and benchmark methods for convenience

Usage

```
compare_analysis(X)
```

Arguments

`X`	a data matrix

Value

a list of data matrix `X` and outcomes of PSA-S, PSA-O, PCA, log-ratio PCA, and power transform PCA with power 1/2.

See Also

`psa()`, `comp_pca()`, `comp_apca()`, `power_pca()`

`comp_apca` *Log-Ratio Principal Component Analysis for PSA Comparison*

Description

A wrapper function of `princomp.acomp()` for comparison to PSA. Zeros are substituted by half of the overall nonzero minimum. Manipulates result of `princomp.acomp()` in a similar format to `psa()`.

Usage

```
comp_apca(X)
```

Arguments

`X`	a data matrix.

Value

A list

- pts.approx a list of lower dimensional representations with respect to the original basis
- scores a matrix of scores.
- rss a vector of residual sums of squares.
- modes a list of modes of variation. The rth element of the list is the difference of rank r approximations to rank $r-1$ approximations.
- loadings a matrix of loading vectors.
- center mean of the data

See Also

`comp_pca()`, `psa()`

`comp_pca` *Principal Component Analysis for PSA Comparison*

Description

A wrapper function of `princomp()` for comparison to PSA. Manipulates result of `princomp()` in a similar format to `psa()`.

Usage

```
comp_pca(X)
```

Arguments

`X`	a data matrix.

Value

A list

- pts.approx a list of lower dimensional representations with respect to the original basis
- scores a matrix of scores.
- rss a vector of residual sums of squares.
- modes a list of modes of variation. The rth element of the list is the difference of rank r approximations to rank $r-1$ approximations.
- loadings a matrix of loading vectors.
- center mean of the data

See Also

`psa()`

ggbmean *Plot the backwards mean*

Description

I *think* this is equivalent to the zero-dimensional result, i.e. `result$vertices["r=1"]]`

Usage

```
ggbmean(result, plot = TRUE)
```

Arguments

`result`	Output of psoa() pssa() or similar.
`plot`	If TRUE, the makes a parallel coordinates plot using `ggplot2`. Otherwise returns the backward mean.

ggscores_pairs *Plot scores from two merges*

Description

Scatter plot of scores. Along the border of the plot the component names making up the vertices involved are printed. The top border has the components that make up the positive-direction vertex of the merge specified by `y` in `mapping`. The bottom border has the components that make up the negative-direction vertex of the merge.

For PSSA scores depend on the ratio of mass in each of the vertices in the merge, and also on the total mass in both vertices (this could potentially be removed later for).

Usage

```
ggscores_pairs(result, ..., dimensions, mapping = NULL)
```

Arguments

`result`	Output from psoa or pssa
`...`	Passed to tidy_scores
`dimensions`	Passed to tidy_scores. Note dimensions is a misnomer - it is actually the number of vertices.
`mapping`	Create using `ggplot2::aes()`, do not specify `x` and `y` aesthetics - these are created from `dimensions` (possibly hacky, expoiting the fact that aes() creates a list).

gridsearch *Find the best merge by grid search*

Description

Find the best merge by grid search

Usage

```
gridsearch(X, evaluator, testweights = seq(0, 1, length.out = 100), ...)
```

Arguments

X	matrix of data points for a simplex with vertices at the basis coordinates.
evaluator	A function that evaluates each proposed merge and return a single non-negative value for the quality of the merge. Arguments of X, v1, v2, and w at least
testweights	A vector of weights to try.
...	Passed to evaluator

Value

The best pair and weight as a single row data frame (data frame for easier row binding and because the v1 and v2 are best stored as integers).

gridsearch_pair *Find the best weight for merging two vertices by grid search*

Description

Find the best weight for merging two vertices by grid search

Usage

```
gridsearch_pair(
  X,
  v1,
  v2,
  testweights = seq(0, 1, length.out = 100),
  evaluator,
  ...
)
```

Arguments

X	matrix of data points for a simplex with vertices at the basis coordinates.
v1	an index of a vertex (a basis vector)
v2	an index of a vertex (a basis vector)
testweights	A vector of weights to try.
evaluator	A function that evaluates each proposed merge and return a single non-negative value for the quality of the merge. Arguments of X, v1, v2, and w at least
...	Passed to evaluator

---

loading_bar *Loading Bar Plots for PSA*

Description

Creating a bar plot summarizing a loading vector

Usage

```
loading_bar(v, max.k = 12)
```

Arguments

v	a loading vector
max.k	maximum number of elements to display

Value

a ggplot object

---

pairwisereplace *Iteratively replace components with scaled components, and rescale to a composition.*

Description

Replaces the mass in a component with two new components that are multiples of the original component, with the weight1 the multiple for the first new component and weight2 the multiple for the second new component The compositional data has the same number of rows as weight and begins as all 1, with one component.

Usage

```
pairwisereplace(weight1, weight2, component = rep(1, ncol(weight1)))
```

Arguments

weight1, weight2	A matrix specifying how the mass in the component is split.
component	A vector of integers specifying the component to be split each time.

pairwisesubpartition *Iteratively split components by given weight.*

Description

Splits the mass in a component into two, with the `weight` dictating how much of the mass goes to the first new component. The compositional data has the same number of rows as `weight` and begins as all 1, with one component.

Usage

```
pairwisesubpartition(weight, component = rep(1, ncol(weight)))
```

Arguments

`weight`	A matrix specifying how the mass in the component is split.
`component`	A vector of integers specifying the component to be split each time.

plotdendrogram2 *Plot a dendrogram of the merges.*

Description

Converts results to a `dendrogram` (see `stats::as.dendrogram()`) object and plots it.

Usage

```
plotdendrogram2(
  result,
  plot = TRUE,
  edge.root = TRUE,
  edgePar = list(p.lty = 0, t.cex = 1, t.adj = c(1, 1)),
  horiz = TRUE,
  nodeheight = "step",
  colors = NULL,
  ...
)
```

Arguments

`result`	The `dendrogram.input` value from `psa()`.
`plot`	logical; if TRUE, plot the dendrogram output.
`edge.root`	logical; if TRUE, draw an edge to the root node.
`edgePar`	a list of plotting parameters for edges. See `graphics::segments()`.
`horiz`	logical; if TRUE, draw the dendrogram horizontally.
`nodeheight`	a string specifying node height. "step" for heights given by the number of vertices after the merge or "rmse scores" for heights given by the RMSE of the scores of the merge.
`colors`	a vector of colors of nodes
`...`	Passed to `graphics::plot()`.

power_pca *Power Transform Principal Component Analysis for PSA Comparison*

Description

A wrapper function of `princomp()` applied to power transformed data.

Usage

```
power_pca(X, alpha = 1/2)
```

Arguments

`X`	a data matrix.
`alpha`	a scalar $alpha > 0$ for power transform.

Value

A list

- pts.approx a list of lower dimensional representations with respect to the original basis
- scores a matrix of scores.
- rss a vector of residual sums of squares.
- modes a list of modes of variation. The rth element of the list is the difference of rank r approximations to rank $r-1$ approximations.
- loadings a matrix of loading vectors.
- center mean of the data

See Also

`comp_pca()`, `psa()`

projectsuborthant *Project data to a lower suborthant*

Description

Project data to a lower suborthant

Usage

```
projectsuborthant(X, V = diag(ncol(X)), v1, v2, w)
```

Arguments

`X`	a data matrix whose coordinates are computed with respect to `V`.
`V`	matrix for vertices of the reduced simplex.
`v1, v2`	vertices to be merged.
`w`	weight for merge.

Details

The order of v1 and v2 specify the sign of the scores: points on the v2 side of the orthant have positive scores

Value

- `X` projected data points in the new coordinates
- `V` new vertices
- `scores` Signed scores of the projection
- `mergedirection` The direction of the merge, with positive pointing towards positive score points.

`projectsubsimplex` *Project to a Subsimplex*

Description

Project to a Subsimplex

Usage

```
projectsubsimplex(X, V = diag(ncol(X)), v1, v2, w)
```

Arguments

`X`	a data matrix whose coordinates are computed with respect to `V`.
`V`	matrix for vertices of the reduced simplex.
`v1, v2`	vertices to be merged.
`w`	weight for merge.

Details

Following Version 4 of the descriptions. The result of the projection are points with mass (i.e. value) in the newly created vertex equal to the sum of the mass (i.e. value) in the components specified by `v1` and `v2`. The new vertex is computed as `w * V[v1, ] + (1-w) * V[v2, ]`, which is the convex combination of two vertices and is *not* a vector with L2-norm equal to 1. Scores a signed as positive if the difference between the input point and the projected point has a positive inner-product with the merge direction `V[v2, ] - V[v1, ]`.

psa *Principal Nested Simplices Analysis*

Description

Estimate PSA-S or PSA-O of given data matrix.

Usage

```
psa(type, X, testweights = seq(0, 1, length.out = 100))
```

Arguments

type	s for PSA-S or o for PSA-O.
X	A data matrix.
testweights	A vector of weights to try.

Value

A list

- vertices a list of matrix representing vertices of the lower dimensional subsimplex. The rth element of the list corresponds to the rank $r-1$ subsimplex.
- pts a list of lower dimensional representations with respect to the reduced basis `vertices`
- pts.approx a list of lower dimensional representations with respect to the original basis
- scores a matrix of scores.
- rss a vector of residual sums of squares.
- modes a list of modes of variation. The rth element of the list is the difference of rank r approximations to rank $r-1$ approximations.
- loadings a matrix of loading vectors.
- const.info a data frame of merged vertices and merging weight at each merge.
- dendrogram.input additional information to apply `plotdendrogram2()`.

See Also

`plotdendrogram2()`

psoa *Principle Sub Orthant Analysis for the Simplex*

Description

Principle Sub Orthant Analysis for the Simplex

Usage

```
psoa(
  X,
  V = diag(ncol(X)),
  testweights = seq(0, 1, length.out = 100),
  merges = NULL,
  maxsteps = min(ncol(X), nrow(merges)) - 1
)
```

Arguments

X	matrix of data points for a simplex with vertices at the basis coordinates. Each row an observation.
V	matrix of vertices in the basis of the *original* data. Each row a vertex corresponding to the *columns* of X.
testweights	A vector of weights to try.
merges	if non-NULL, then a data frame of prespecified merges.
maxsteps	maximum number of merges to compute.

pssa *Principle Sub Orthant Analysis for the Simplex*

Description

Principle Sub Orthant Analysis for the Simplex

Usage

```
pssa(
  X,
  V = diag(ncol(X)),
  testweights = seq(0, 1, length.out = 100),
  merges = NULL,
  maxsteps = min(ncol(X), nrow(merges)) - 1
)
```

Arguments

X	matrix of data points for a simplex with vertices at the basis coordinates. Each row an observation.
V	matrix of vertices in the basis of the *original* data. Each row a vertex corresponding to the *columns* of X.
testweights	A vector of weights to try.
merges	if non-NULL, then a data frame of prespecified merges.
maxsteps	maximum number of merges to compute.

---

recurse_tolowerdimension

Recurse through dimensions

Description

Given a merge evaluator and a merge projector, recurse through dimensions keeping track of the appropriate data.

Usage

```
recurse_tolowerdimension(
  X,
  V = diag(ncol(X)),
  evaluator,
  projector,
  testweights = seq(0, 1, length.out = 100),
  merges = NULL,
  maxsteps = NULL
)
```

Arguments

X	matrix of data points for a simplex with vertices at the basis coordinates. Each row an observation.
V	matrix of vertices in the basis of the *original* data. Each row a vertex corresponding to the *columns* of X.
evaluator	A function that evaluates each proposed merge and return a single non-negative value for the quality of the merge. Arguments of X, v1, v2, and w at least
projector	A function that takes result of gridsearch() and applies a projection. Returning X, V, scores, mergedirection.
testweights	A vector of weights to try.
merges	if non-NULL, then a data frame of prespecified merges.
maxsteps	maximum number of merges to compute.

Details

- Vbirth keeps track of the merge that created each vertex. The merge is recorded acording to the destination dimension of the merge, so a value of i means the merge that created that vertex was creating the data for dimension d - (i-1), where d is the starting dimension.

tidy_modes *Prepare modes of variation for ggplotting*

Description

Prepare modes of variation for ggplotting

Usage

```
tidy_modes(result, ..., dimensions = NULL)
```

Arguments

result	a list generated by psa().
...	Data frames of covariates with rows corresponding exactly to the input data. Column names should not be the same as column names of the input.
dimensions	Supply the number of vertices after the merge corresponding to the modes of variation. By default all dimensions are included.

tolowerdimension *Finds and applies best merge*

Description

Finds and applies best merge

Usage

```
tolowerdimension(
  X,
  V = diag(1, ncol(X)),
  evaluator,
  projector,
  testweights = seq(0, 1, length.out = 100),
  merge = NULL
)
```

Arguments

X	matrix of data points for a simplex with vertices at the basis coordinates. Each row an observation.
V	matrix of vertices in the basis of the *original* data. Each row a vertex corresponding to the *columns* of X.
evaluator	A function that evaluates each proposed merge and return a single non-negative value for the quality of the merge. Arguments of X, v1, v2, and w at least
projector	A function that takes result of gridsearch() and applies a projection. Returning X, V, scores, mergedirection.
testweights	A vector of weights to try.
merge	If supplied, used instead of gridsearch(). bestmerge must have same structure as result from gridsearch()

Details

If `merge` is not supplied and the *largest* score of the found merge is *negative*, then swaps the signs of the scores and records this by swapping the returned `v1` with `v2`, changing w to 1-w, and flipping the direction of `mergedirection`. This should mean that the returned merge information can be reapplied on new data and the direction will be the same.

If `merge` is non-`NULL` then `evaluator` is ignored - no search occurs.

Value

A list

- `X` Projected data points in with the new vertices as the coordinates
- `V` New set of vertices with row names.
- `scores` Signed scores for the merge
- `mergedirection` The unit vector parallel to `v2 - v1` and in the direction of positive scores, and in the original coordinates.
- `v1` The index of `v1` used in `projector` (corresponding to a row of the old vertices matrix).
- `v2` The index of `v2` used in `projector` (corresponding to a row of the old vertices matrix).
- `w` The weight of the merge between `v1` and `v2`. As in `w * v1 + (1-w) * v2`, where `v1` and `v2` are the first and second vertices given by `mergedvertices`.

Index