Package ‘visxhclust’

October 12, 2022

Type Package

Title A Shiny App for Visual Exploration of Hierarchical Clustering

Version 1.0.0

Maintainer Rafael Henkin <`r.henkin@qmul.ac.uk`>

Description A Shiny application and functions for visual exploration of hierarchical clustering with numeric datasets. Allows users to iterative set hyperparameters, select features and evaluate results through various plots and computation of evaluation criteria.

License GPL-3

URL `https://github.com/rhenkin/visxhclust`

BugReports `https://github.com/rhenkin/visxhclust/issues`

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

biocViews

Imports shiny, shinyhelper, shinycssloaders, bsplus, DT, ggplot2, dplyr, readr, tidyr, cluster, fastcluster, clusterCrit, dunn.test, RColorBrewer, dendextend, ComplexHeatmap, circlize, patchwork, knitr, kableExtra

RoxygenNote 7.1.1

Suggests testthat (>= 3.0.0), rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Rafael Henkin [aut, cre]

Repository CRAN

Date/Publication 2021-10-19 07:20:02 UTC

R topics documented:

`annotate_clusters` *Annotate data frame with clusters*

Description

Annotate data frame with clusters

Usage

```
annotate_clusters(df, cluster_labels, long = TRUE, selected_clusters = NULL)
```

Arguments

`df`	a data frame
`cluster_labels`	list of cluster labels, automatically converted to factor.
`long`	if `TRUE`, returned data frame will be in long format. See details for spec. Default is `TRUE`.
`selected_clusters`	optional cluster labels to filter

Details

Long data frame will have columns: `Cluster`, `Measurement` and `Value`.

Value

a wide or long data frame

Examples

```
dmat <- compute_dmat(iris, "euclidean", TRUE, c("Petal.Length", "Sepal.Length"))
res <- compute_clusters(dmat, "complete")
cluster_labels <- cut_clusters(res, 2)
annotated_data <- annotate_clusters(iris[, c("Petal.Length", "Sepal.Length")], cluster_labels)
head(annotated_data)
```

`bin_df` *Simulated binary data*

Description

Simulated binary data

Usage

```
bin_df
```

Format

A data frame with 200 rows and 10 variables:

a variable a

b variable b

c variable c

d variable d

e variable e

f variable f

g variable g

h variable h

i variable i

j variable j

Source

package author

`cluster_boxplots` *Plot boxplots with clusters*

Description

This is a convenience wrapper function for `facet_boxplot()`. Combined with `annotate_clusters()`, it doesn't require specifying axes in `facet_boxplot()`.

Usage

```
cluster_boxplots(annotated_data, ...)
```

Arguments

`annotated_data`	data frame returned by `annotate_clusters()`
`...`	arguments passed to `facet_boxplot()`

Value

boxplots faceted by clusters

Examples

```
dmat <- compute_dmat(iris, "euclidean", TRUE, c("Petal.Length", "Sepal.Length"))
clusters <- compute_clusters(dmat, "complete")
cluster_labels <- cut_clusters(clusters, 2)
annotated_data <- annotate_clusters(iris[, c("Petal.Length", "Sepal.Length")], cluster_labels)
cluster_boxplots(annotated_data, boxplot_colors = visxhclust::cluster_colors)
```

`cluster_colors` *List of colors used in the Shiny app for clusters*

Description

List of colors used in the Shiny app for clusters

Usage

```
cluster_colors
```

Format

An object of class `character` of length 39.

`cluster_heatmaps` *Plot heatmap with cluster results and dendrogram*

Description

Plot heatmap with cluster results and dendrogram

Usage

```
cluster_heatmaps(
  scaled_selected_data,
  clusters,
  k,
  cluster_colors,
  scaled_unselected_data = NULL,
  annotation = NULL
)
```

Arguments

`scaled_selected_data`	scaled matrix or data frame with variables used for clustering
`clusters`	hierarchical cluster results produced by `fastcluster::hclust()`
`k`	targeted number of clusters
`cluster_colors`	list of cluster colors to match with boxplots
`scaled_unselected_data`	(optional) scaled matrix or data frame with variables not used for clustering
`annotation`	(optional) ComplexHeatmap::columnAnnotation object

Value

a ComplexHeatmap::Heatmap

Examples

```
dmat <- compute_dmat(iris, "euclidean", TRUE, c("Petal.Length", "Sepal.Length"))
clusters <- compute_clusters(dmat, "complete")
species_annotation <- create_annotations(iris, "Species")
cluster_heatmaps(scale(iris[c("Petal.Length", "Sepal.Length")]),
                 clusters,
                 3,
                 visxhclust::cluster_colors,
                 annotation = species_annotation)
```

`compute_clusters` *Compute clusters hierarchically from distance matrix*

Description

Compute clusters hierarchically from distance matrix

Usage

```
compute_clusters(dmat, linkage_method)
```

Arguments

`dmat`	a distance matrix
`linkage_method`	a linkage method supported by `fastcluster::hclust()`

Value

clusters computed by `fastcluster::hclust()`

Examples

```
dmat <- compute_dmat(iris, "euclidean", TRUE, c("Petal.Length", "Sepal.Length"))
res <- compute_clusters(dmat, "complete")
```

`compute_dmat` *Compute a distance matrix from scaled data*

Description

This function applies scaling to the columns of a data frame and computes and returns a distance matrix from a chosen distance measure.

Usage

```
compute_dmat(
  x,
  dist_method = "euclidean",
  apply_scaling = FALSE,
  subset_cols = NULL
)
```

Arguments

`x`	a numeric data frame or matrix
`dist_method`	a distance measure to apply to the scaled data. Must be those supported by `stats::dist()`, plus `"mahalanobis"` and `"cosine"`. Default is `"euclidean"`.
`apply_scaling`	use TRUE to apply `base::scale()`. By default does not scale data.
`subset_cols`	(optional) a list of columns to subset the data

Value

an object of class "dist" (see `stats::dist()`)

Examples

```
dmat <- compute_dmat(iris, "euclidean", TRUE, c("Petal.Length", "Sepal.Length"))
print(class(dmat))
```

`compute_gapstat` *Compute Gap statistic for clustered data*

Description

Compute Gap statistic for clustered data

Usage

```
compute_gapstat(df, clusters, gap_B = 50, max_k = 14)
```

Arguments

`df`	the data used to compute clusters
`clusters`	output of `compute_clusters()` or `fastcluster::hclust()`
`gap_B`	number of bootstrap samples for `cluster::clusGap()` function. Default is 50.
`max_k`	maximum number of clusters to compute the statistic. Default is 14.

Value

a data frame with the Tab component of `cluster::clusGap()` results

Examples

```
data_to_cluster <- iris[c("Petal.Length", "Sepal.Length")]
dmat <- compute_dmat(data_to_cluster, "euclidean", TRUE)
clusters <- compute_clusters(dmat, "complete")
gap_results <- compute_gapstat(scale(data_to_cluster), clusters)
head(gap_results)
```

`compute_metric` *Compute an internal evaluation metric for clustered data*

Description

Metric will be computed from 2 to max_k clusters. Note that the row number in results will be different from k.

Usage

```
compute_metric(df, clusters, metric_name, max_k = 14)
```

Arguments

`df`	data frame used to compute clusters
`clusters`	output of `compute_clusters()` or `fastcluster::hclust()`
`metric_name`	valid metric name from `clusterCrit::getCriteriaNames()` (with TRUE argument)
`max_k`	maximum number of clusters to cut using `dendextend::cutree()`. Default is 14.

Value

a data frame with columns `k` and `score`

Examples

```
data_to_cluster <- iris[c("Petal.Length", "Sepal.Length")]
dmat <- compute_dmat(data_to_cluster, "euclidean", TRUE)
clusters <- compute_clusters(dmat, "complete")
compute_metric(scale(data_to_cluster), clusters, "Dunn")
```

`correlation_heatmap` *Plot a correlation heatmap*

Description

Computes pairwise Pearson correlation; if there are fewer than 15 columns, prints the value of the correlation coefficient inside each tile.

Usage

```
correlation_heatmap(df)
```

Arguments

`df`	numeric data frame to compute correlations

Value

a ComplexHeatmap::Heatmap

`create_annotations` *Create heatmap annotations from selected variables*

Description

This function will create a ComplexHeatmap::columnAnnotation object with rows for each variable passed as argument. Character columns will be coerced into factors. For factors, the ColorBrewer palette `Set3` will be used. For non-negative numeric, the `PuBu` palette will be used, and for columns with negative values, the reversed `RdBu` will be used.

Usage

```
create_annotations(df, selected_variables)
```

Arguments

`df`	a data frame. It can be an original unscaled data, or a scaled one
`selected_variables`	list of columns in the data frame to create annotations for

Value

a ComplexHeatmap::columnAnnotation object

`cut_clusters` *Cut a hierarchical tree targeting k clusters*

Description

Cut a hierarchical tree targeting k clusters

Usage

```
cut_clusters(clusters, k)
```

Arguments

`clusters`	cluster results, produced by e.g. `fastcluster::hclust()`
`k`	target number of clusters

Value

cluster labels

Examples

```
dmat <- compute_dmat(iris, "euclidean", TRUE, c("Petal.Length", "Sepal.Length"))
clusters <- compute_clusters(dmat, "complete")
cluster_labels <- cut_clusters(clusters, 2)
head(cluster_labels)
```

`dmat_projection` *Plot a 2D MDS projection of a distance matrix*

Description

Plot a 2D MDS projection of a distance matrix

Usage

```
dmat_projection(dmat, point_colors = NULL, point_palette = NULL)
```

Arguments

`dmat`	distance matrix
`point_colors`	optional list of labels to color points (will be coerced to factor)
`point_palette`	optional palette used with `ggplot2::scale_colour_manual()`

Value

a ggplot object

Examples

```
dmat <- dist(iris[, c("Sepal.Width", "Sepal.Length")])
dmat_projection(dmat)
```

`facet_boxplot` *Faceted boxplots with points or violin plots*

Description

Faceted boxplots with points or violin plots

Usage

```
facet_boxplot(
  df,
  x,
  y,
  facet_var = NULL,
  boxplot_colors = NULL,
  shape = c("boxplot", "violin"),
  plot_points = TRUE
)
```

Arguments

`df`	a data frame containing all the variables matching the remaining arguments
`x`	categorical variable
`y`	continuous variable
`facet_var`	optional variable to facet data
`boxplot_colors`	list of colors to use as fill for boxplots
`shape`	either "boxplot" or "violin"
`plot_points`	boolean variable to overlay jittered points or not. Default is `TRUE`

Value

a ggplot2::ggplot object

Examples

```
facet_boxplot(iris, x = "Species", y = "Sepal.Length", facet_var = "Species")
```

`line_plot` *A custom line plot with optional vertical line*

Description

A custom line plot with optional vertical line

Usage

```
line_plot(df, x, y, xintercept = NULL)
```

Arguments

`df`	data source
`x`	variable for horizontal axis
`y`	variable for vertical axis
`xintercept`	optional value in horizontal axis to highlight

Value

a ggplot2::ggplot object

`logscaled_df` *Simulated logscaled data*

Description

Simulated logscaled data

Usage

```
logscaled_df
```

Format

A data frame with 200 rows and 10 variables:

a variable a

b variable b

c variable c

d variable d

e variable e

f variable f

g variable g

h variable h

i variable i

j variable j

Source

package author

`normal_annotated` *Simulated normal data with annotations*

Description

Simulated normal data with annotations

Usage

```
normal_annotated
```

Format

A data frame with 200 rows and 10 variables:

a variable a

b variable b

c variable c

d variable d

e variable e

f variable f

g variable g

h variable h

i variable i

j variable j

annot annotation column

Source

package author

`normal_df` *Simulated normal data*

Description

Simulated normal data

Usage

```
normal_df
```

Format

A data frame with 200 rows and 10 variables:

a variable a

b variable b

c variable c

d variable d

e variable e

f variable f

g variable g

h variable h

i variable i

j variable j

Source

package author

`normal_missing` *Simulated normal data with missing values*

Description

Simulated normal data with missing values

Usage

```
normal_missing
```

Format

A data frame with 200 rows and 10 variables:

a variable a

b variable b

c variable c

d variable d

e variable e

f variable f

g variable g

h variable h

i variable i

j variable with randomly missing values

Source

package author

`optimal_score` *Find minimum or maximum score in a vector*

Description

This function is meant to be used with compute_metric. For Gap statistic, use `cluster::maxSE()`.

Usage

```
optimal_score(x, method = c("firstmax", "globalmax", "firstmin", "globalmin"))
```

Arguments

`x`	a numeric vector
`method`	one of "firstmax", "globalmax", "firstmin" or "globalmin"

Value

the index (not k) of the identified maximum or minimum score

Examples

```
data_to_cluster <- iris[c("Petal.Length", "Sepal.Length")]
dmat <- compute_dmat(data_to_cluster, "euclidean", TRUE)
clusters <- compute_clusters(dmat, "complete")
res <- compute_metric(scale(data_to_cluster), clusters, "Dunn")
optimal_score(res$score, method = "firstmax")
```

`plot_annotation_dist` *Plot distribution of annotation data across clusters*

Description

Plot distribution of annotation data across clusters

Usage

```
plot_annotation_dist(annotations_df, cluster_labels, selected_clusters = NULL)
```

Arguments

`annotations_df`	data frame with variables not used in clustering
`cluster_labels`	output from `cut_clusters()`
`selected_clusters`	optional vector of cluster labels to include in plots

Value

a `patchwork` object

Examples

```
dmat <- compute_dmat(iris, "euclidean", TRUE, c("Petal.Length", "Sepal.Length"))
clusters <- compute_clusters(dmat, "complete")
cluster_labels <- cut_clusters(clusters, 2)
plot_annotation_dist(iris["Species"], cluster_labels)
```

`run_app` *Runs the Shiny app*

Description

Runs the Shiny app

Usage

```
run_app()
```

Value

No return value, runs the app by passing it to print

Examples

```
## Only run this example in interactive R sessions
if (interactive()) {
library(visxhclust)
run_app()
}
```

Index