

# Package: visxhclust (via r-universe)

August 31, 2024

**Type** Package

**Title** A Shiny App for Visual Exploration of Hierarchical Clustering

**Version** 1.1.0.9000

**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)

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

**RoxygenNote** 7.2.3

**Roxygen** list(markdown = TRUE)

**Suggests** testthat (>= 3.0.0), rmarkdown

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Repository** <https://rhenkin.r-universe.dev>

**RemoteUrl** <https://github.com/rhenkin/visxhclust>

**RemoteRef** HEAD

**RemoteSha** cd5be1cbaae51c0089f8c4b82f59c54818d26748

## Contents

annotate_clusters . . . . .	2
bin_df . . . . .	3
cluster_boxplots . . . . .	4
cluster_colors . . . . .	4
cluster_heatmaps . . . . .	5
compute_clusters . . . . .	6
compute_dmat . . . . .	6
compute_gapstat . . . . .	7
compute_metric . . . . .	8
correlation_heatmap . . . . .	8
create_annotations . . . . .	9
cut_clusters . . . . .	9
dmat_projection . . . . .	10
facet_boxplot . . . . .	10
line_plot . . . . .	11
logscaled_df . . . . .	12
normal_annotated . . . . .	13
normal_df . . . . .	13
normal_missing . . . . .	14
optimal_score . . . . .	15
plot_annotation_dist . . . . .	15
run_app . . . . .	16
<b>Index</b>	<b>17</b>

---

annotate_clusters	<i>Annotate data frame with clusters</i>
-------------------	--

---

### 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	<i>Plot heatmap with cluster results and dendrogram</i>
------------------	---

---

## 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	<i>Compute clusters hierarchically from distance matrix</i>
------------------	---

---

### 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	<i>Compute a distance matrix from scaled data</i>
--------------	---

---

### 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 <code>stats::dist()</code> , plus "mahalanobis" and "cosine". Default is "euclidean".
apply_scaling	use TRUE to apply <code>base::scale()</code> . 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	<i>Compute Gap statistic for clustered data</i>
-----------------	---

---

**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 <code>compute_clusters()</code> or <code>fastcluster::hclust()</code>
gap_B	number of bootstrap samples for <code>cluster::clusGap()</code> 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(dmat, clusters, metric_name, max_k = 14)
```

### Arguments

<code>dmat</code>	distance matrix output of <code>compute_dmat()</code> or <code>stats::dist()</code>
<code>clusters</code>	output of <code>compute_clusters()</code> or <code>fastcluster::hclust()</code>
<code>metric_name</code>	"silhouette" or "dunn"
<code>max_k</code>	maximum number of clusters to cut using <code>dendextend::cutree()</code> . 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(dmat, 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

<code>df</code>	numeric data frame to compute correlations
-----------------	--



**Value**

a [ComplexHeatmap::Heatmap](#)

---

create\_annotatations      *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_annotatations(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	<i>Plot a 2D MDS projection of a distance matrix</i>
-----------------	--

---

### 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 <code>ggplot2::scale_colour_manual()</code>

### Value

a ggplot object

### Examples

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

---

facet_boxplot	<i>Faceted boxplots with points or violin plots</i>
---------------	---

---

### 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	<i>A custom line plot with optional vertical line</i>
-----------	---

---

**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	<i>Simulated logscaled data</i>
--------------	---------------------------------

---

**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	<i>Simulated normal data with annotations</i>
------------------	---

---

**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	<i>Simulated normal data</i>
-----------	------------------------------

---

**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	<i>Find minimum or maximum score in a vector</i>
---------------	--

---

**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(dmat, clusters, "dunn")
optimal_score(res$score, method = "firstmax")
```

---

plot_annotation_dist	<i>Plot distribution of annotation data across clusters</i>
----------------------	---

---

**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

- \* **datasets**
  - bin\_df, 3
  - cluster\_colors, 4
  - logscaled\_df, 12
  - normal\_annotated, 13
  - normal\_df, 13
  - normal\_missing, 14
- annotate\_clusters, 2
- base::scale(), 7
- bin\_df, 3
- cluster::clusGap(), 7
- cluster::maxSE(), 15
- cluster\_boxplots, 4
- cluster\_colors, 4
- cluster\_heatmaps, 5
- ComplexHeatmap::columnAnnotation, 5, 9
- ComplexHeatmap::Heatmap, 5, 9
- compute\_clusters, 6
- compute\_clusters(), 7, 8
- compute\_dmat, 6
- compute\_dmat(), 8
- compute\_gapstat, 7
- compute\_metric, 8
- correlation\_heatmap, 8
- create\_annotations, 9
- cut\_clusters, 9
- cut\_clusters(), 16
- dendextend::cutree(), 8
- dmat\_projection, 10
- facet\_boxplot, 10
- fastcluster::hclust(), 5–9
- ggplot2::ggplot, 11, 12
- ggplot2::scale\_colour\_manual(), 10
- line\_plot, 11
- logscaled\_df, 12
- normal\_annotated, 13
- normal\_df, 13
- normal\_missing, 14
- optimal\_score, 15
- plot\_annotation\_dist, 15
- run\_app, 16
- stats::dist(), 7, 8