

# Package ‘RclusTool’

February 4, 2020

**Type** Package

**Title** Graphical Toolbox for Clustering and Classification of Data Frames

**Version** 0.91.3

**Date** 2020-01-29

**Author** Guillaume Wacquet [aut],  
Pierre-Alexandre Hebert [aut, cre],  
Emilie Poisson [aut],  
Pierre Talon [aut]

**Maintainer** Pierre-Alexandre Hebert <hebert@univ-littoral.fr>

**Description** Graphical toolbox for clustering and classification of data frames.

It proposes a graphical interface to process clustering and classification methods on features data-frames, and to view initial data as well as resulted cluster or classes. According to the level of available labels, different approaches are proposed: unsupervised clustering, semi-supervised clustering and supervised classification.

To assess the processed clusters or classes, the toolbox can import and show some supplementary data formats: either profile/time series, or images.

These added information can help the expert to label clusters (clustering), or to constrain data frame rows (semi-supervised clustering), using Constrained spectral embedding algorithm by Wacquet et al. (2013) <doi:10.1016/j.patrec.2013.02.003> and the methodology provided by Wacquet et al. (2013) <doi:10.1007/978-3-642-35638-4\_21>.

**License** GPL (>= 2)

**RoxygenNote** 7.0.2

**LazyData** TRUE

**Depends** R (>= 3.0.0), tcltk, tcltk2, tkrplot

**Imports** class, cluster, conclust, corrplot, e1071, factoextra,  
FactoMineR, ggplot2, grid, jpeg, MASS, mclust, mda, mmand,  
nnet, png, randomForest, reshape, sp, stringi, stringr, tools

**SystemRequirements** XQuartz (on OSX)

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2020-02-04 16:20:05 UTC

URL <http://mawenzi.univ-littoral.fr/RclusTool/>

## R topics documented:

|                                     |           |
|-------------------------------------|-----------|
| addOperation . . . . .              | 2         |
| applyPreprocessing . . . . .        | 3         |
| clusterSummary . . . . .            | 4         |
| computeSemiSupervised . . . . .     | 5         |
| computeSupervised . . . . .         | 7         |
| computeUnSupervised . . . . .       | 9         |
| extractProtos . . . . .             | 10        |
| formatLabelSample . . . . .         | 11        |
| imgClassif . . . . .                | 12        |
| importSample . . . . .              | 13        |
| loadPreprocessFile . . . . .        | 15        |
| purgeSample . . . . .               | 16        |
| RclusToolGUI . . . . .              | 17        |
| readTrainSet . . . . .              | 18        |
| saveCalcul . . . . .                | 19        |
| saveClustering . . . . .            | 20        |
| saveCounts . . . . .                | 21        |
| saveManualProtos . . . . .          | 22        |
| savePreprocess . . . . .            | 23        |
| saveSummary . . . . .               | 23        |
| sigClassif . . . . .                | 24        |
| visualizeSampleClustering . . . . . | 25        |
| <b>Index</b>                        | <b>28</b> |

---

|              |                      |
|--------------|----------------------|
| addOperation | <i>Add operation</i> |
|--------------|----------------------|

---

### Description

addOperation create configuration object for the datasample

### Usage

```
addOperation(parameterList, featureOperations)
```

### Arguments

parameterList,  
list of Preprocessing instructions for an operation.

featureOperations,  
matrix where to list Operations on features.

**Value**

The configuration object created by the list of preprocessing instructions parameterList in featureOperations.

**Examples**

```
featOp <- matrix(ncol=4,nrow=0)
#Adding two differents variables
featOp <- addOperation(list("+","x","y"), featOp)
#Select a variable
featOp <- addOperation(list("select","x"), featOp)
#Change a profile color
featOp <- addOperation(list("signalColor","x","grey"), featOp)
#Make a PCA projection (with the number of dimensions)
featOp <- addOperation(list("projection","pca","0"), featOp)
#Make a spectral projection
featOp <- addOperation(list("projection","spectral"), featOp)
#Scale the data
featOp <- addOperation(list("scaling","on"), featOp)
#Sample the data (with a sampling size)
featOp <- addOperation(list("sampling","150"), featOp)
#Make a log transformation of a variable
featOp <- addOperation(list("log","x"), featOp)
```

---

applyPreprocessing      *Preprocessing application*

---

**Description**

Apply a new preprocess to a data.sample object.

**Usage**

```
applyPreprocessing(
  data.sample,
  operations = NULL,
  RclusTool.env = initParameters(),
  reset = TRUE,
  preprocessed.only = FALSE
)
```

**Arguments**

|               |  |
|---------------|--|
| data.sample   | sample object.   |
| operations    | list of data.frames describing all preprocessing operations.                 |
| RclusTool.env | environment in which all global parameters, raw data and results are stored. |
| reset         | boolean : if TRUE (default) the configuration is reset.                      |

preprocessed.only

boolean : if TRUE (default) processing are restricted to the "preprocessed" features.

### Details

applyPreprocessing applies a new preprocess to a data.sample object

### Value

The data.sample sample object on which was applied the operations or NULL if preprocessing operations fail.

### See Also

[loadPreprocessFile](#)

### Examples

```
dat <- rbind(matrix(rnorm(150, mean = 2, sd = 0.3), ncol = 3),
             matrix(rnorm(150, mean = 4, sd = 0.3), ncol = 3),
             matrix(rnorm(150, mean = 6, sd = 0.3), ncol = 3))
colnames(dat) <- c("x", "y", "z")
tf1 <- tempfile()
write.table(dat, tf1, sep=";", dec=",")
x <- importSample(file.features=tf1, sepFeat=";", decFeat=",", dir.save=tempdir())

instr <- rbind(c("select", "x", "log", ""), c("select", "y", "log", ""))
tf2 <- tempfile()
write.table(instr, tf2, sep=",", col.names = FALSE, row.names = FALSE)

operations <- loadPreprocessFile(tf2)
x <- applyPreprocessing(x, operations)
```

---

clusterSummary

*Clusters summaries computation*

---

### Description

Save clusters summaries results in a csv file.

### Usage

```
clusterSummary(
  data.sample,
  label,
  features.to.keep = colnames(data.sample$features[["preprocessed"]])$x,
```

```
summary.functions = c(Min = "min", Max = "max", Sum = "sum", Average = "mean", SD =
  "sd")
)
```

### Arguments

`data.sample` list containing features, profiles and clustering results.

`label` vector of labels.

`features.to.keep` vector of features names on which the summaries are computed.

`summary.functions` vector of functions names for the summaries computation. Could be 'Min', 'Max', 'Sum', 'Average', 'sd'.

### Details

`clusterSummary` computes the clusters summaries (min, max, sum, average, sd) from a clustering result.

### Value

out data.frame containing the clusters summaries.

### Examples

```
dat <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
tf1 <- tempfile()
write.table(dat, tf1, sep=",", dec=".")

x <- importSample(file.features=tf1, dir.save=tempdir())
res <- KmeansQuick(x$features$initial$x, K=3)
labels <- formatLabelSample(res$cluster, x)
cluster.summary <- clusterSummary(x, labels)
```

---

computeSemiSupervised *Semi-supervised clustering*

---

### Description

Perform semi-supervised clustering based on pairwise constraints, dealing with the number of clusters  $K$ , automatically or not.

**Usage**

```
computeSemiSupervised(
  data.sample,
  ML,
  CNL,
  K = 0,
  kmax = 20,
  method.name = "Constrained_KM",
  maxIter = 2,
  pca = F,
  pca.nb.dims = 0,
  spec = F,
  use.sampling = FALSE,
  sampling.size.max = 0,
  scaling = F,
  RclusTool.env = initParameters(),
  echo = T
)
```

**Arguments**

|                                |   |
|--------------------------------|---|
| <code>data.sample</code>       | list containing features, profiles and clustering results.  |
| <code>ML</code>                | list of ML (must-link) constrained pairs (as row.names of features).  |
| <code>CNL</code>               | list of CNL (cannot-link) constrained pairs (as row.names of features).   |
| <code>K</code>                 | number of clusters. If <code>K=0</code> (default), this number is automatically computed thanks to the Elbow method.                                    |
| <code>kmax</code>              | maximum number of clusters.   |
| <code>method.name</code>       | character vector specifying the constrained algorithm to use. Must be 'Constrained_KM' (default) or 'Constrained_SC' (Constrained Spectral Clustering). |
| <code>maxIter</code>           | number of iterations for SemiSupervised algorithm   |
| <code>pca</code>               | boolean: if TRUE, Principal Components Analysis is applied to reduce the data space.  |
| <code>pca.nb.dims</code>       | number of principal components kept. If <code>pca.nb.dims=0</code> , this number is computed automatically.   |
| <code>spec</code>              | boolean: if TRUE, spectral embedding is applied to reduce the data space.   |
| <code>use.sampling</code>      | boolean: if FALSE (default), data sampling is not used.   |
| <code>sampling.size.max</code> | numeric: maximal size of the sampling set.  |
| <code>scaling</code>           | boolean: if TRUE, scaling is applied.   |
| <code>RclusTool.env</code>     | environment in which data and intermediate results are stored.  |
| <code>echo</code>              | boolean: if FALSE (default), no description printed in the console.   |

**Details**

`computeSemiSupervised` performs semi-supervised clustering based on pairwise constraints, dealing with the number of clusters `K`, automatically or not

**Value**

The function returns a list containing:

|         |  |
|---------|--|
| label   | vector of labels.  |
| summary | data.frame containing clusters summaries (min, max, sum, average, sd). |
| nbItems | number of observations.  |

**See Also**

[computeCKmeans](#), [computeCSC](#), [KwaySSSC](#)

**Examples**

```
dat <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),
              matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
              matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
tf <- tempfile()
write.table(dat, tf, sep=",", dec=".")
x <- importSample(file.features=tf, dir.save=tempdir())

pairs.abs <- visualizeSampleClustering(x, selection.mode = "pairs",
                                     profile.mode="whole sample", wait.close=TRUE)

res.ckm <- computeSemiSupervised(x, ML=pairs.abs$ML, CNL=pairs.abs$CNL, K=0)
plot(dat[,1], dat[,2], type = "p", xlab = "x", ylab = "y",
      col = res.ckm$label, main = "Constrained K-means clustering")
```

---

computeSupervised      *Supervised classification*

---

**Description**

Perform supervised classification based on the use of a training set.

**Usage**

```
computeSupervised(
  data.sample,
  prototypes,
  method.name = "K-NN",
  model = NULL,
  RclusTool.env = initParameters()
)
```

**Arguments**

|                            |   |
|----------------------------|---|
| <code>data.sample</code>   | list containing features, profiles and clustering results.  |
| <code>prototypes</code>    | data.frame containing the features of each prototype associated to a class.   |
| <code>method.name</code>   | character vector specifying the supervised algorithm to use. Must be 'K-NN' (K-Nearest Neighbor by default), 'MLP' (MultiLayer Perceptron), 'SVM' (Support Vector Machine) or 'RF' (Random Forest). |
| <code>model</code>         | option to predict directly from model   |
| <code>RclusTool.env</code> | environment in which all global parameters, raw data and results are stored.  |

**Details**

`computeSupervised` performs supervised classification based on the use of a training set

**Value**

The function returns a list containing:

|                         |   |
|-------------------------|---|
| <code>label</code>      | vector of labels.   |
| <code>summary</code>    | data.frame containing classes summaries (min, max, sum, average, sd).       |
| <code>nbItems</code>    | number of observations.   |
| <code>prototypes</code> | data.frame containing the features of each prototype associated to a class. |

**See Also**

[readTrainSet](#)

**Examples**

```
rep <- system.file("extdata", package="RclusTool")
featuresFile <- file.path(rep, "sample_example_features.csv")
features <- read.csv(featuresFile, header = TRUE)
features$ID <- NULL
traindir <- file.path(rep, "train_example")
tf <- tempfile()
write.table(features, tf, sep=",", dec=".")

x <- importSample(file.features=tf, dir.save=tempdir())

train <- readTrainSet(traindir)

res <- computeSupervised(x, prototypes=train)

plot(features[,3], features[,4], type = "p", xlab = "x", ylab = "y",
      col = res$label, main = "K-Nearest-Neighbor classification")
```



---

computeUnSupervised     *Unsupervised clustering*

---

### Description

Perform unsupervised clustering, dealing with the number of clusters  $K$ , automatically or not.

### Usage

```
computeUnSupervised(
  data.sample,
  K = 0,
  method.name = "K-means",
  pca = F,
  pca.nb.dims = 0,
  spec = F,
  use.sampling = FALSE,
  sampling.size.max = 0,
  scaling = F,
  RclusTool.env = initParameters(),
  echo = FALSE
)
```

### Arguments

|                   |  |
|-------------------|--|
| data.sample       | list containing features, profiles and clustering results.   |
| K                 | number of clusters. If $K=0$ (default), this number is automatically computed thanks to the Elbow method.  |
| method.name       | character vector specifying the constrained algorithm to use. Must be 'K-means' (default), 'EM' (Expectation-Maximization), 'Spectral', 'HC' (Hierarchical Clustering) or 'PAM' (Partitioning Around Medoids). |
| pca               | boolean: if TRUE, Principal Components Analysis is applied to reduce the data space.   |
| pca.nb.dims       | number of principal components kept. If $pca.nb.dims=0$ , this number is computed automatically.   |
| spec              | boolean: if TRUE, spectral embedding is applied to reduce the data space.  |
| use.sampling      | boolean: if FALSE (default), data sampling is not used.  |
| sampling.size.max | numeric: maximal size of the sampling set.   |
| scaling           | boolean: if TRUE, scaling is applied.  |
| RclusTool.env     | environment in which all global parameters, raw data and results are stored.   |
| echo              | boolean: if FALSE (default), no description printed in the console.  |

**Details**

computeUnSupervised performs unsupervised clustering, dealing with the number of clusters  $K$ , automatically or not

**Value**

data.sample list containing features, profiles and updated clustering results (with vector of labels and clusters summaries).

**See Also**

[computeKmeans](#), [computeEM](#), [spectralClustering](#), [computePcaSample](#), [computeSpectralEmbeddingSample](#)

**Examples**

```
dat <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
tf <- tempfile()
write.table(dat, tf, sep=",", dec=".")
x <- importSample(file.features=tf, dir.save=tempdir())

x <- computeUnSupervised(x, K=0, pca=TRUE, echo=TRUE)
label <- x$clustering[["K-means_pca"]]$label
plot(dat[,1], dat[,2], type = "p", xlab = "x", ylab = "y",
      col = label, main = "K-means clustering")
```

---

extractProtos

*Prototypes extraction*

---

**Description**

Extract prototypes of each cluster automatically, according to a clustering result, and save them in different directories. In order to catch the whole variability, each cluster is divided into several sub-clusters, and medoids of each sub-cluster are considered as prototypes.

**Usage**

```
extractProtos(
  data.sample,
  method,
  K.max = 20,
  kmeans.variance.min = 0.95,
  user.name = ""
)
```

**Arguments**

`data.sample` list containing features, profiles and clustering results.  
`method` character vector specifying the clustering method (already performed) to use.  
`K.max` maximal number of clusters (K.max=20 by default).  
`kmeans.variance.min` elbow method cumulative explained variance > criteria to stop K-search.  
`user.name` character vector specifying the user name.

**Details**

`extractProtos` extracts prototypes automatically according to a clustering result, and save them in different directories

**Value**

csv file containing the prototypes

**Examples**

```
dat <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
tf1 <- tempfile()
write.table(dat, tf1, sep="," , dec=".")

x <- importSample(file.features=tf1, dir.save=tempdir())
x <- computeUnSupervised(x, K=3, method.name="K-means")

extractProtos(x, method = "K-means_preprocessed")
```

---

formatLabelSample      *Labels formatting*

---

**Description**

Format labels for unsupervised classification and add cleaned observations as 'Noise'.

**Usage**

```
formatLabelSample(  
  label,  
  data.sample,  
  new.labels = TRUE,  
  use.sampling = FALSE,  
  noise.cluster = "Noise"  
)
```

**Arguments**

|               |   |
|---------------|---|
| label         | vector of labels.   |
| data.sample   | sample object.  |
| new.labels    | boolean: if TRUE (default), new names are given for each cluster (beginning by 'Cluster').                            |
| use.sampling  | boolean: if TRUE (not default), data.sample\$sampling is used to generalize label from sampling set to the whole set. |
| noise.cluster | character name of the cluster "noise".  |

**Details**

formatLabelSample formats labels for unsupervised classification and adds cleaned observations as 'Noise'

**Value**

new.labels formatted labels.

**Examples**

```
dat <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
tf <- tempfile()
write.table(dat, tf, sep=",", dec=".")

x <- importSample(file.features=tf, dir.save=tempdir())
res <- KmeansQuick(x$features$initial$x, K=3)

new.labels <- formatLabelSample(res$cluster, x)
```

---

imgClassif

*Images clustering*


---

**Description**

Sort images (if available) in different directories according to a clustering result.

**Usage**

```
imgClassif(data.sample, imgdir, method, user.name = "")
```

**Arguments**

|                          |   |
|--------------------------|---|
| <code>data.sample</code> | list containing features, profiles and clustering results.                    |
| <code>imgdir</code>      | character vector specifying the path of the images directory.                 |
| <code>method</code>      | character vector specifying the clustering method (already performed) to use. |
| <code>user.name</code>   | character vector specifying the user name.                                    |

**Details**

`imgClassif` sorts images (if available) in different directories according to a clustering result

**Value**

images files in the different directories, csv file containing the detail.

**See Also**

[sigClassif](#)

**Examples**

```
dat <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
tf1 <- tempfile()
write.table(dat, tf1, sep=",", dec=".")

rep <- system.file("extdata", package="RclusTool")
imgdir <- file.path(rep, "img_example")

x <- importSample(file.features=tf1, dir.images=imgdir, dir.save=tempdir())
x <- computeUnSupervised(x, K=3, method.name="K-means")

imgClassif(x, imgdir, method = "K-means_preprocessed")
```

---

importSample

*Sample importation*

---

**Description**

Import the required and the optional files, and build a dataset.

**Usage**

```

importSample(
  file.features = "",
  file.meta = "",
  file.profiles = "",
  file.RDS = "",
  file.config = "",
  dir.images = "",
  dir.save = ".",
  sepFeat = ",",
  decFeat = ".",
  naFeat = c("", "NA"),
  sepSig = ",",
  decSig = ".",
  naSig = c("", "NA"),
  headerCSV = TRUE,
  RclusTool.env = new.env(),
  save.to.disk = TRUE,
  ...
)

```

**Arguments**

|                            |   |
|----------------------------|---|
| <code>file.features</code> | character vector specifying the csv file containing features data.  |
| <code>file.meta</code>     | character vector specifying the txt file containing metadata.   |
| <code>file.profiles</code> | character vector specifying the csv file containing profiles data.  |
| <code>file.RDS</code>      | character vector for a RDS file containing a <code>data.sample</code> object. This file is automatically saved when importing a (csv-)file-features. When both a csv-file-features and a RDS file are given, the last one is ignored. |
| <code>file.config</code>   | character vector for the name of the configuration file.  |
| <code>dir.images</code>    | character vector containing the path of images directory.   |
| <code>dir.save</code>      | character vector specifying path of the working directory.  |
| <code>sepFeat</code>       | character specifying the field separator for the csv file containing features data.   |
| <code>decFeat</code>       | character specifying the decimal points for the csv file containing features data.  |
| <code>naFeat</code>        | vector containing missing values for the csv file containing features data.   |
| <code>sepSig</code>        | character specifying the field separator for the csv file containing profiles data.   |
| <code>decSig</code>        | character specifying the decimal point for the csv file containing profiles data.   |
| <code>naSig</code>         | vector containing missing values for the csv file containing profiles data.   |
| <code>headerCSV</code>     | boolean if TRUE (default) the file contains the names of the variables as its first line.   |
| <code>RclusTool.env</code> | environment in which data and intermediate results are stored.  |
| <code>save.to.disk</code>  | boolean if TRUE (default) some data files and folders are saved to disk.  |
| <code>...</code>           | parameters addressed to <code>read.csv</code> functions.  |

**Details**

function to import sample from CSV files; sample is preprocessed

**Value**

data.sample loaded data.sample.

**See Also**

[loadSample](#)

**Examples**

```
dat <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
tf1 <- tempfile()
write.table(dat, tf1, sep=",", dec=".")

metadat <- rbind("First metadata: ...", "Second metadata: ...")
tf2 <- tempfile()
writeLines(metadat, tf2)

x <- importSample(file.features=tf1, file.meta=tf2, dir.save=tempdir())
```

---

loadPreprocessFile      *Preprocessing loading*

---

**Description**

Load a csv file configuration with instruction to remove bad observations and builds object config that describes all preprocessings to apply.

**Usage**

```
loadPreprocessFile(file.config, ...)
```

**Arguments**

|             |   |
|-------------|---|
| file.config | character vector specifying the name of a csv file with preprocessing instructions. |
| ...         | parameters addressed to read.csv functions.   |

**Details**

loadPreprocessFile reads a csv file configuration with instruction to remove bad particles and builds object config that describes all preprocessings done

**Value**

operations character matrix describing all preprocessing operations.

**See Also**

[applyPreprocessing](#)

**Examples**

```
instr <- rbind(c("select","x","log",""), c("select","y","log",""))
tf <- tempfile()
write.table(instr, tf, sep=",", col.names = FALSE, row.names = FALSE)

operations <- loadPreprocessFile(tf)
```

---

purgeSample

*Sample purging*

---

**Description**

Purge sample from its temporary computing results.

**Usage**

```
purgeSample(
  data.sample,
  purge.preprocessing = TRUE,
  purge.clustering = TRUE,
  user.expert = FALSE
)
```

**Arguments**

```
data.sample    sample object
purge.preprocessing    boolean: if TRUE (default), the configuration is reset.
purge.clustering    boolean: if TRUE (default), the clusterings are reset.
user.expert      boolean : if FALSE (default), initial classification feature space is PCA.
```

**Details**

Function to purgeSample from its temporary computing results

**Value**

data.sample purged data.sample.



**Examples**

```
dat <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
tf <- tempfile()
write.table(dat, tf, sep="," , dec=".")

x <- importSample(file.features=tf, dir.save=tempdir())
x <- computeUnSupervised(x, K=3, method.name="K-means")
x <- purgeSample(x, purge.clustering=TRUE)
```

---

RclusToolGUI

*Username and user type selection*

---

**Description**

Generate a first window to enter the username and to select the user type ('standard' or 'expert').

**Usage**

```
RclusToolGUI(RclusTool.env = new.env(), debug = F)
```

**Arguments**

`RclusTool.env` environment in which data and results will be stored. If NULL, a local environment will be created.

`debug` boolean: if TRUE, the debug mode is activated.

**Details**

function to display the first window of the RclusTool interface (username and user type selection)

**Value**

Nothing, just open the graphical user interface.

**Examples**

```
RclusToolGUI()
```

---

`readTrainSet`*Training set reading*

---

**Description**

Read a training set built from prototypes, to train a classifier for supervised classification.

**Usage**

```
readTrainSet(  
  traindir,  
  keep_ = FALSE,  
  operations = NULL,  
  RclusTool.env = initParameters()  
)
```

**Arguments**

|                            |   |
|----------------------------|---|
| <code>traindir</code>      | character vector specifying the path of the training set.                             |
| <code>keep_</code>         | boolean: if FALSE (default), the '_' directory is not considered in the training set. |
| <code>operations</code>    | list of data.frames describing all preprocessing operations.                          |
| <code>RclusTool.env</code> | environment in which all global parameters, raw data and results are stored.          |

**Details**

`readTrainSet` reads a training set built from prototypes, to train a classifier for supervised classification

**Value**

prototypes data.frame containing the features of each prototype associated to a class.

**See Also**

[dropTrainSetVars](#)

**Examples**

```
rep <- system.file("extdata", package="RclusTool")  
traindir <- file.path(rep, "train_example")  
train <- readTrainSet(traindir)
```

---

|            |                      |
|------------|----------------------|
| saveCalcul | <i>Object saving</i> |
|------------|----------------------|

---

### Description

Save object created after calculation in a csv file.

### Usage

```
saveCalcul(filename.rdata, dat, dir)
```

### Arguments

filename.rdata character vector specifying the path and the name of the rdata file.  
dat object to save.  
dir character vector specifying the directory where to save the rdata file.

### Details

saveCalcul saves object created after calculation in a csv file

### Value

RDS file containing calculation.

### Examples

```
dat <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
tf1 <- tempfile()
write.table(dat, tf1, sep=",", dec=".")

x <- importSample(file.features=tf1, dir.save=tempdir())
res.pca <- computePcaSample(x)

tf2 <- tempfile()
saveCalcul(basename(tf2), res.pca$pca, tempdir())
```

---

|                |                          |
|----------------|--------------------------|
| saveClustering | <i>Clustering saving</i> |
|----------------|--------------------------|

---

## Description

Save a clustering result in a csv file.

## Usage

```
saveClustering(filename.csv, label, dir)
```

## Arguments

|              |   |
|--------------|---|
| filename.csv | character vector specifying the path and the name of the csv file.    |
| label        | vector of labels.   |
| dir          | character vector specifying the directory where to save the csv file. |

## Details

saveClustering saves a clustering result in a csv file

## Value

csv file containing clustering result.

## See Also

[loadClusteringSample](#)

## Examples

```
dat <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
tf1 <- tempfile()
write.table(dat, tf1, sep=",", dec=".")

x <- importSample(file.features=tf1, dir.save=tempdir())
res <- KmeansQuick(x$features$initial$x, K=3)

tf2 <- tempfile()
saveClustering(basename(tf2), res$cluster, tempdir())
```

---

|            |                     |
|------------|---------------------|
| saveCounts | <i>Count saving</i> |
|------------|---------------------|

---

## Description

Save a count result in a csv file.

## Usage

```
saveCounts(filename.csv, counts, dir)
```

## Arguments

filename.csv    character vector specifying the path and the name of the csv file.  
counts            vector of counts.  
dir                character vector specifying the directory where to save the csv file.

## Details

saveCounts saves a count result in a csv file

## Value

csv file containing count result.

## Examples

```
dat <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),  
             matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),  
             matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))  
tf1 <- tempfile()  
write.table(dat, tf1, sep=",", dec=".")  
  
x <- importSample(file.features=tf1, dir.save=tempdir())  
res <- KmeansQuick(x$features$initial$x, K=3)  
  
tf2 <- tempfile()  
saveCounts(basename(tf2), table(res$cluster), tempdir())
```

---

saveManualProtos      *Manual prototypes saving*

---

## Description

Save the profiles and images of prototypes selected manually by user in a scatterplot.

## Usage

```
saveManualProtos(data.sample, protos)
```

## Arguments

`data.sample`      list containing features, profiles and clustering results.  
`protos`            list of selected prototypes (with index and name).

## Details

`saveManualProtos` saves the profiles and images of prototypes selected manually by user in a scatterplot

## Value

profiles and images of prototypes selected, csv file with detail.

## Examples

```
dat <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
tf1 <- tempfile()
write.table(dat, tf1, sep=",", dec=".")

x <- importSample(file.features=tf1, dir.save=tempdir())

new.protos <- visualizeSampleClustering(x, selection.mode = "prototypes",
  profile.mode="whole sample", wait.close=FALSE)
saveManualProtos(x, new.protos)
```

---

|                |                                  |
|----------------|----------------------------------|
| savePreprocess | <i>Preprocessing exportation</i> |
|----------------|----------------------------------|

---

**Description**

Export all preprocessing operations in a csv file.

**Usage**

```
savePreprocess(filename.csv, config, dir)
```

**Arguments**

|              |   |
|--------------|---|
| filename.csv | character vector specifying the name of the csv file.               |
| config       | 4-columns character matrix describing all preprocessing operations. |
| dir          | character vector specifying the directory of the csv file.          |

**Details**

savePreprocess exports all preprocessing operations in a csv file

**Value**

csv file containing preprocessing.

**Examples**

```
test.file <- "test.savePreprocess.csv"  
config <- matrix(c("select","x",NA,NA,"select","y",NA,NA), byrow=TRUE, ncol=4)  
savePreprocess(test.file, config, tempdir())
```

---

|             |                                  |
|-------------|----------------------------------|
| saveSummary | <i>Clusters summaries saving</i> |
|-------------|----------------------------------|

---

**Description**

Save clusters summaries results in a csv file.

**Usage**

```
saveSummary(filename.csv, cluster.summary, dir, info = NULL)
```

**Arguments**

filename.csv    character vector specifying the path and the name of the csv file.  
 cluster.summary    data.frame containing the clusters summaries results.  
 dir    character vector specifying the directory where to save the csv file.  
 info    character vector about sample or clustering.

**Details**

saveSummary saves clusters summaries results in a csv file

**Value**

csv file containing clusters summaries results.

**See Also**

[loadSummary](#)

**Examples**

```
dat <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
colnames(dat) <- c("x", "y")
tf1 <- tempfile()
write.table(dat, tf1, sep=",", dec=".")

x <- importSample(file.features=tf1, dir.save=tempdir())
res <- KmeansQuick(x$features$initial$x, K=3)
labels <- formatLabelSample(res$cluster, x)
cluster.summary <- clusterSummary(x, labels)

tf2 <- tempfile()
saveSummary(basename(tf2), cluster.summary, tempdir())
```

---

sigClassif

*Signals clustering*

---

**Description**

Sort signals (if available) in different directories according to a clustering result.

**Usage**

```
sigClassif(data.sample, method, user.name = "")
```



**Arguments**

`data.sample` list containing features, profiles and clustering results.  
`method` character vector specifying the clustering method (already performed) to use.  
`user.name` character vector specifying the user name.

**Details**

`sigClassif` sorts signals (if available) in different directories according to a clustering result

**Value**

`signals` plots images in the different directories.

**See Also**

[imgClassif](#)

**Examples**

```
dat <- rbind(matrix(rnorm(100, mean = 0, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2))
tf1 <- tempfile()
write.table(dat, tf1, sep=",", dec=".")

sig <- data.frame(ID=rep(1:150, each=30), SIGNAL=rep(dnorm(seq(-2,2,length=30)),150))
tf2 <- tempfile()
write.table(sig, tf2, sep=",", dec=".")

x <- importSample(file.features=tf1,file.profiles = tf2, dir.save=tempdir())
x <- computeUnSupervised(x, K=3, method.name="K-means")

sigClassif(x, method = "K-means_preprocessed")
```

---

visualizeSampleClustering

*Interactive figure with 2D scatter-plot*

---

**Description**

Open an interactive figure with 2D scatter-plot of all particles with axis choice. Grey color (label=0) is for data to cleaned or to remove in classification process.

**Usage**

```

visualizeSampleClustering(
  data.sample,
  label = NULL,
  clustering.name = "proposed clustering",
  cluster.summary = NULL,
  RclusTool.env = initParameters(),
  prototypes = NULL,
  profile.mode = "none",
  selection.mode = "none",
  compare.mode = "off",
  pairs = NULL,
  features.mode = "initial",
  wait.close = FALSE,
  fontsize = 9
)

```

**Arguments**

|                              |  |
|------------------------------|--|
| <code>data.sample</code>     | list containing features, profiles and clustering results.   |
| <code>label</code>           | vector of labels.  |
| <code>clustering.name</code> | character vector specifying the clustering method used to get labels.  |
| <code>cluster.summary</code> | data.frame containing the clusters summaries (as returned by 'clusterSummary').  |
| <code>RclusTool.env</code>   | environment in which all global parameters, raw data and results are stored.   |
| <code>prototypes</code>      | list containing vectors of prototypes indices.   |
| <code>profile.mode</code>    | character vector specifying the plot mode of profiles. Must be 'none' (default), 'whole sample', 'cluster i' or 'constrained pairs'.   |
| <code>selection.mode</code>  | character vector specifying the selection mode of profiles. Must be 'none' (default), 'prototypes' or 'pairs'.   |
| <code>compare.mode</code>    | character vector specifying the mode of comparison between two clusterings results. Must be 'off' (default) or 'on'.   |
| <code>pairs</code>           | list of constrained pairs (must-link and cannot-link).   |
| <code>features.mode</code>   | character vector specifying the plot mode of features (projection in a specific space). Must be 'initial' (default), 'preprocessed', 'pca', 'pca_full' or 'spectral', or prefixed versions ('sampled', 'scaled') of those space names. |
| <code>wait.close</code>      | boolean: if FALSE (default), the following steps of the analysis calculations are computed even if the window is not closed.   |
| <code>fontsize</code>        | size of font (default is 9)  |

**Details**

visualizeSampleClustering opens an interactive figure with 2D scatter-plot of all particles with axis choice

**Value**

prototypes in selection.mode = "prototypes" mode, pairs in selection.mode = "pairs" mode.

**See Also**

[plotProfile](#), [plotSampleFeatures](#)

**Examples**

```
dat <- rbind(matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 4, sd = 0.3), ncol = 2),
             matrix(rnorm(100, mean = 6, sd = 0.3), ncol = 2))
colnames(dat) <- c("x", "y")
tf1 <- tempfile()
write.table(dat, tf1, sep=",", dec=".")

sig <- data.frame(ID=rep(1:150, each=30), SIGNAL=rep(dnorm(seq(-2,2,length=30)),150))
tf2 <- tempfile()
write.table(sig, tf2, sep=",", dec=".")

x <- importSample(file.features=tf1, file.profiles=tf2, dir.save=tempdir())

res <- KmeansQuick(x$features$initial$x, K=3)
new.labels <- formatLabelSample(res$cluster, x)

visualizeSampleClustering(x, label = new.labels, clustering.name="K-means",
                          profile.mode="whole sample")
```

# Index

addOperation, [2](#)  
applyPreprocessing, [3](#), [16](#)  
  
clusterSummary, [4](#)  
computeCKmeans, [7](#)  
computeCSC, [7](#)  
computeEM, [10](#)  
computeKmeans, [10](#)  
computePcaSample, [10](#)  
computeSemiSupervised, [5](#)  
computeSpectralEmbeddingSample, [10](#)  
computeSupervised, [7](#)  
computeUnSupervised, [9](#)  
  
dropTrainSetVars, [18](#)  
  
extractProtos, [10](#)  
  
formatLabelSample, [11](#)  
  
imgClassif, [12](#), [25](#)  
importSample, [13](#)  
  
KwaySSSC, [7](#)  
  
loadClusteringSample, [20](#)  
loadPreprocessFile, [4](#), [15](#)  
loadSample, [15](#)  
loadSummary, [24](#)  
  
plotProfile, [27](#)  
plotSampleFeatures, [27](#)  
purgeSample, [16](#)  
  
RclusToolGUI, [17](#)  
readTrainSet, [8](#), [18](#)  
  
saveCalcul, [19](#)  
saveClustering, [20](#)  
saveCounts, [21](#)  
saveManualProtos, [22](#)  
  
savePreprocess, [23](#)  
saveSummary, [23](#)  
sigClassif, [13](#), [24](#)  
spectralClustering, [10](#)  
  
visualizeSampleClustering, [25](#)