# Package 'optCluster'

April 16, 2022

**Title** Determine Optimal Clustering Algorithm and Number of Clusters

**Version** 1.3.2

**Date** 2022-04-16

**Author** Michael Sekula <michael.sekula@louisville.edu>,
Somnath Datta <somnath.datta@louisville.edu>,
and Susmita Datta <susmita.datta@louisville.edu>

**Maintainer** Michael Sekula <michael.sekula@louisville.edu>

**Description** Cluster analysis using statistical and biological validation measures for both continuous and count data.

**Depends** R (>= 3.0)

**biocViews**

**Imports** grDevices, graphics, stats, utils, RankAggreg, mclust, methods, cluster, gplots, kohonen, MBCluster.Seq,

**Suggests** Biobase, annotate, GO.db, org.At.tair.db

**LazyLoad** true

**Collate** orphaned-Code.R optCluster-Classes.R optCluster-Methods.R optCluster-Internal.R optCluster-Functions.R

**License** GPL (>= 2)

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-04-16 16:10:02 UTC

## R topics documented:

---

aggregPlot *Rank Aggregation Plot*

---

## Description

aggregPlot displays a figure representing the results from rank aggregation.

## Usage

```
aggregPlot(x, show.average = TRUE, show.legend = TRUE, colR = "red", ...)
```

## Arguments

| | |
|---|---|
| x | An object of class "optCluster". |
| show.average | If TRUE, the average aggregation will be plotted. |
| show.legend | If TRUE, the legend will be displayed. |
| colR | A character string specifying the line color for the optimal ranked list. |
| ... | Additional plotting parameters. |

## Details

This function will produce a visual representation of rank aggregation results in three plots combined into a single figure. The top left plot displays the minimum values of the objective function as the number of iterations increases. The top right plot is a histogram of the objective function scores at the last iteration. The bottom plot shows the individual ranks of the clustering algorithms from the data (in grey), the final "optimal" rankings (in red), and the average ranking of each clustering algorithm (in black).

## References

Pihur, V., Datta, S. and Datta, S. (2009). RankAggreg, an R package for weighted rank aggregation. BMC Bioinformatics, 10:62, https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-10-62.

## See Also

RankAggreg, optCluster-class

## Examples

```
## This example may take a few minutes to compute

## Obtain Dataset
data(arabid)
```

```
## Normalize Data with Respect to Library Size
obj <- t(t(arabid)/colSums(arabid))

## Analysis of Normalized Data using Internal and Stability Validation Measures
norm1 <- optCluster(obj, 2:4, clMethods = "all")
aggregPlot(norm1)
```

---

arabid *Subset of Arabidopsis RNA-Seq Data*

---

## Description

A subset of RNA-Seq data studying the defense response of Arabidopsis thaliana to a bacterial infection. Two treatment groups were analyzed in the study: a mock inoculation ("mock") and a bacteria ("hrcc"). Three independent samples were used for each treatment. The 200 genes in this subset were randomly selected without replacement from the 26,222 genes in the original dataset. The rows in this matrix correspond to genes and the columns correspond to samples. Each individual cell of the matrix contains the counts of the RNA-Seq reads that are mapped to a reference database of known genes.

## Usage

```
data(arabid)
```

## Format

A 200 by 6 matrix of RNA-Seq read frequencies.

## Source

Di, Y., Schafer, D. W., Cumbie, J. S., & Chang, J. H. (2011). The NBP negative binomial model for assessing differential gene expression from RNA-Seq. Statistical Applications in Genetics and Molecular Biology, 10, 1-28.

## Examples

```
## These examples may each take a few minutes to compute

## Obtain Dataset
data(arabid)

## Analysis of Count Data using Internal and Stability Validation Measures
count1 <- optCluster(arabid, 2:4, clMethods = "all", countData = TRUE)
topMethod(count1)

## Analysis of Normalized Data using Internal and Stability Validation Measures
```

```
obj <- t(t(arabid)/colSums(arabid)) ## Normalized with Respect to Library Size
norm1 <- optCluster(obj, 2:4, clMethods = "all")
topMethod(norm1)
```

---

clValid-class *Class "clValid"*

---

## Description

The class "clValid" is taken from the ORPHANED clValid package (Brock et al., 2008). The description below is taken verbatum from the reference manual of clValid.

## Objects from the Class

Objects can be created using the function clValid.

## Slots

clusterObjs: Object of class "list". A list containing the results from the clustering methods.

measures: Object of class "array". A 3-dimensional array which contains the validation measures for the clustering results. The first dimension indicates the validation measures, the second the number of clusters, and the third the clustering methods.

measNames: Object of class "character". The names of the validation measures.

clMethods: Object of class "character". A character vector giving the clustering methods.

labels: Object of class "character". A character vector giving the item (gene) labels.

nClust: Object of class "numeric". A numeric vector giving the numbers of clusters which were evaluated.

validation: Object of class "character". A character vector giving the type of validation measures used, consisting of some combination of "internal", "stability", or "biological".

metric: Object of class "character". The metric used to determine the distance matrix.

method: Object of class "character". For hierarchical clustering, the agglomeration method used.

neighbSize: Object of class "numeric". For internal validation, the neighborhood size used for the connectivity measure.

annotation: Object of class "character or array or list". Either a character string naming the Bioconductor annotation package for mapping genes to GO categories, or a list with the names of the functional classes and the observations belonging to each class.

GOcategory: Object of class "character". For biological validation, gives which GO categories to use for biological validation. Can be one of "BP", "MF", "CC", or "all"

goTermFreq: Object of class "numeric". For the BSI, what threshold frequency of GO terms to use for functional annotation.

call: Object of class "call". Gives the call to clValid used to create the clValid object.

### Methods

**clusterMethods** `signature(object = "clValid")`: Returns the names of the clustering methods.

**clusters** `signature(object = "clValid")`: Returns the results from the clustering methods.

Additional arguments:

`method = clMethods(object)` The clustering method(s) to extract.

**measNames** `signature(object = "clValid")`: Returns the names of the validation measures.

**measures** `signature(object = "clValid")`: Returns the validation measures.

Additional arguments:

`measures = measNames(object)` The validation measure(s) to extract.

**nClusters** `signature(object = "clValid")`: Returns the numbers of clusters evaluated.

**plot** `signature(x = "clValid",y = "missing")`: Plots the validation measures.

Additional arguments:

`measures=measNames(x)` The validation measures to plot.

`legend=TRUE` If TRUE provides a legend.

`legendLoc="topright"` The location of the legend.

`main=NULL` Title of graph.

`pch=NULL` Plotting characters to use.

`type="b"` Type of plot.

`ask=prod(par("mfcol")) < length(measures) && dev.interactive()` Logical. If `TRUE` the user is prompted before each plot.

**print** `signature(x = "clValid")`: Print method for class `clValid`.

**show** `signature(object = "clValid")`: Same as print.

**summary** `signature(object = "clValid")`: Summary method for class `clValid`.

Additional arguments:

`digits = max(3,getOption("digits")-3)` The number of significant digits to use.

---

`optCluster` *Determine Optimal Clustering Algorithm and Number of Clusters*

---

### Description

`optCluster` performs statistical and/or biological validation of clustering results and determines the optimal clustering algorithm and number of clusters through rank aggreation. The function returns an object of class `"optCluster"`.

### Usage

```
optCluster(obj, nClust, clMethods = c("clara", "diana", "hierarchical",
"kmeans", "model", "pam", "som", "sota"), countData = FALSE,
validation = c("internal", "stability"), hierMethod = "average",
annotation = NULL, clVerbose = FALSE, rankMethod = "CE",
distance = "Spearman", importance = NULL, rankVerbose = FALSE, ...)
```

## Arguments

| | |
|---|---|
| obj | The dataset to be evaluated as either a data frame, a numeric matrix, or an ExpressionSet object. Items to be clustered must be the rows of the data. In the case of data frames, all columns must be numeric. |
| nClust | A numeric vector providing the range of clusters to be evaluated (e.g. to evaluate the number of clusters ranging from 2 to 4, input 2:4). A single number can also be provided. |
| clMethods | A character vector providing the names of the clustering algorithms to be used. The available algorithms are: "agnes", "clara", "diana", "fanny", "hierarchical", "kmeans", "model", "pam", "som", "sota", "em.nbinom", "da.nbinom", "sa.nbinom", "em.poisson", "da.poisson", "sa.poisson". Any number of selected methods is allowed. The option "all" may also be used but with some caution, see **Clustering Algorithms** in the 'Details' section for more information. |
| countData | A logical argument, indicating whether the data is count based or not. Can also be used in conjuction with the "all" option for the 'clMethods' argument. If TRUE and 'clMethods' = "all", all of the clustering algorithms for count data are selected: "em.nbinom", "da.nbinom", "sa.nbinom", "em.poisson", "da.poisson", "sa.poisson". If FALSE and 'clMethods' = "all", all of the relevant clustering algorithms used with continuous data are selected: "agnes", "clara", "diana", "fanny", "hierarchical", "kmeans", "model", "pam", "som", "sota". |
| validation | A character vector providing the names of the types of validation measures to be used. The options of "internal", "stability", "biological", and "all" are available. Any number or combination of choices is allowed. |
| hierMethod | A character string, providing the agglomeration method to be used by the hierarchical clustering options (hclust and agnes). Available choices are "average", "complete", "single", and "ward". |
| annotation | Used in biological validation. Either a character string providing the name of the Bioconductor annotation package for mapping genes to GO categories, or the names of each functional class and the observations that belong to them in either a list or logical matrix format. |
| clVerbose | If TRUE, the progress of cluster validation will be produced as output. |
| rankMethod | A character string providing the method to be used for rank aggregation. The two options are the cross-entropy Monte Carlo algorithm ("CE") or Genetic algorithm ("GA"). Selection of only one method is allowed. |
| distance | A character string providing the type of distance to be used for measuring the similarity between ordered lists in rank aggregation. The two available methods are the weighted Spearman footrule distance ("Spearman") or the weighted Kendall's tau distance ("Kendall"). Selection of only one distance is allowed. |
| importance | Vector of weights indicating the importance of each validation measure list. Default of NULL represents equal weights to each validation measure. See **Weighted Rank Aggregation** in the 'Details' section for more information. |
| rankVerbose | If TRUE, current rank aggregation results are displayed at each iteration. |

`...` Additional arguments that can be passed to internal functions of `clValid` or `RankAggreg`:
Additional `clValid` arguments:

- `metric` - Metric used to determine distance matrix in validation measures. Possible choices are: "euclean" (default), "correlation", and "manhattan".
- `neighbSize` - Integer giving neighborhood size used in "connectivity" validation measure.
- `GOcategory` - For biological valdation, a character string providing which GO category to use. Options include: "BP", "MF", "CC", or "all" (default).
- `goTermFreq` - For BSI validation, the threshold frequency of GO terms to used for functional annotation.
- `dropEvidence` - For biological validation, either NULL or a character vector of GO evidence codes to omit.

Additional `RankAggreg` arguments:

- `maxIter` - The maximum number of iterations allowed. Default = 1000
- `k` - Size of top-k list in aggregation.
- `convIN` - Stopping criteria for CE and GA algorithms. The algorithm converges once the "best" solution does not change after convIN iterations. Default: 7 for CE and 30 for GA.
- `N` - Number of samples generated by MCMC in the CE algorithm. Default = 10*k^2
- `rho` - For CE algorithm, (rho*N) is the qunatile of candidate list sorted by function values.
- `weight` - For CE algorithm, the learning factor used in the probability update feature. Default = 0.25
- `popSize` - For GA algorithm population size in each generation. Default = 100
- `CP` - For GA algorithm, the crossover probability. Default = 0.4
- `MP` - For GA algorithm, the mutation probability. Default = 0.01

## Details

This function has been created as an extension of the `clValid` function. In addition to the validation measures and clustering algorithms available in the `clValid` function, six clustering algorithms for count data are included in the `optCluster` function. This function also determines a unique solution for the optimal clustering algorithm and number of clusters through rank aggregation of validation measure lists. A brief description of the available clustering algorithms, validation measures, and rank aggregation algorithms is provided below. For more details, please refer to the references.

**Clustering Algorithms:** A total of sixteen clustering algorithms are available for cluster analysis.

- Ten clustering algorithms for continuous data are available through the internal function `clValid`: "agnes", "clara", "diana", "fanny", "hierarchical", "kmeans", "model", "pam", "som", and "sota". ***NOTE:*** *Some algorithms (especially Fanny) may have difficulty finding certain numbers of clusters. If warnings or errors are produced, the offending algorithm(s) should be removed from the* `clMethods` *argument.*

- Six clustering algorithms for count data are available through the **MBCluster.Seq** package: "em.nbinom", "da.nbinom", "sa.nbinom", "em.poisson", "da.poisson", and "sa.poisson". The expectation maximization (EM) algorithm, and two of its variations, the deterministic annealing (DA) algorithm and the simulated annealing (SA) algorithm, have been proposed for model-based clustering of RNA-Seq count data. These three methods can be based on a mixture of either Poisson distributions or negative binomial distributions. The clustering options for count data reflect both the algorithm and the distribution being used. For example, "da.nbinom" represents the deterministic annealing algorithm based on the negative binomial distribution.
  The **MBCluster.Seq** package uses an adjustment by a normalization factor for these algorithms, with the default being log(Q3) where Q3 is 75th percentile. A different normalization factor can be passed through the `optCluster` function by using the argument 'Normalizer'.

**Stability Validation Measures:** Four stability validation are provided: average proportion of non-overlap (APN), average distance (AD), average distance between means (ADM), and figure of merit (FOM). These measures compare the clustering partitions established with the full data to the clustering partitions established while removing each column, one at a time. For each measure, an average is taken over all of the removed columns, which should be minimized.

- The APN determines the average proportion of observations placed in different clusters for both cases. The APN measure can range from 0 to 1.
- The AD calculates the average distance between the observations assigned to the same cluster for both cases. The AD measure can range between zero and infinity.
- The ADM computes the average distance between the centers of clusters for observations put into the same cluster for both cases. ADM values can range between zero and infinity.
- The FOM measures the average intra-cluster variance for the observations in the removed column, using clustering partitions from the remaining columns. The FOM values can range between zero and infinity.

**Internal Validation Measures:** The three internal validation measures included are: connectivity, Dunn index, and silhouette width.

- Connectivity measures the extent at which neighboring observations are clustered together. With a value ranging between zero and infinity, this validation measure should be minimized.
- The Dunn index is the ratio of the minimum distance between observations in different clusters to the maximum cluster diameter. With a value between zero and infinity, this measurement should be maximized.
- Silhouette width is defined as the average of each observation's silhouette value. The silhouette value is a measurement of the degree of confidence in an observation's clustering assignment. Values near 1 mean that the observation is clustered well, while values near -1 mean the observation is poorly clustered.

**Biological Validation Measures:** The biological homogeneity index (BHI) and the biological stability index (BSI) are the two biological validation measures. They were originally proposed to provide guidance in choosing a clustering technique for microarray data, but can also be used for any other molecular expression data as well. Both measures have a range of [0,1] and should be maximized.

- The BHI evaluates how biologically similar defined clusters are by calculating the average proportion of paired genes that are statistically clustered together and have the same functional class.
- The BSI examines the consistency of clustering similar biologically functioning genes together. Observations are removed from the dataset one column at a time and the statistical cluster assignments of genes with the same functional class are compared to the cluster assignments based on the full dataset.

**Rank Aggregation Algorithms:** The cross-entropy Monte Carlo algorithm and the Genetic algorithm are the two approaches available for rank aggregation and come from the **RankAggreg** package. Both rank aggregation algorithms can use either the weighted Spearman footrule distance or the weighted Kendall's tau to measure the "distance" between any two ordered lists.

**Weighted Rank Aggregation:** A list of weights for each validation measure list can be included using the `importance` argument. The default value of equal weights (NULL) is represented by rep(1, length(x)), where x is the character vector of validation measure names. This means each validation measure list has a weight of 1/length(x). To manually change the weights, the order of the validation measures selected needs to be known. The order of validation measures used in `optCluster` is provided below:

- When selected, stability measures will ALWAYS be listed first and in the following order: "APN", "AD", "ADM", "FOM".
- When selected, internal measures will only precede biological measures. The order of these measures is: "Connectivity", "Dunn", "Silhouette".
- When selected, biological measures will always be listed last and in the following order: "BHI", "BSI".

## Value

`optCluster` returns an object of class `"optCluster"`. The class description is provided in the help file.

## Note

Prespecifying a list or a logical matrix of genes corresponding to functional classes for biological validation does not require any additional packages. If entering a list, each item in the list must be a vector providing the genes belonging to a specific biological class. If entering a logical matrix, each column should be a logical vector indicating the genes that belong to a biological class.

If neither a list nor a logical matrix is provided, biological validation will require the **Biobase**, **annotate**, and **GO.db** packages from Bioconductor in addition to an annotation package for your particular data type. See http://www.bioconductor.org for instructions on installing these.

## References

Sekula, M., Datta, S., and Datta, S. (2017). optCluster: An R package for determining the optimal clustering algorithm. Bioinformation, 13(3), 101. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5450252

Brock, G., Pihur, V., Datta, S. and Datta, S. (2008). clValid: An R Package for Cluster Validation. Journal of Statistical Software 25(4), https://www.jstatsoft.org/v25/i04.

Datta, S. and Datta, S. (2003). Comparisons and validation of statistical clustering techniques for microarray gene expression data. Bioinformatics 19(4): 459-466.

Pihur, V., Datta, S. and Datta, S. (2007). Weighted rank aggregation of cluster validation measures: A Mounte Carlo cross-entropy approach. Bioinformatics 23(13): 1607-1615.

Pihur, V., Datta, S. and Datta, S. (2009). RankAggreg, an R package for weighted rank aggregation. BMC Bioinformatics, 10:62, https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-10-62.

Si, Y., Liu, P., Li, P., & Brutnell, T. (2014). Model-based clustering for RNA-seq data. Bioinformatics 30(2): 197-205.

## See Also

For a description of the `clValid` function, including all available arguments that can be passed to it, see `clValid` in the **clValid** package.

For a description of the `RankAggreg` function, including all available arguments that can be passed to it, see `RankAggreg` in the **RankAggreg** package.

For details on the clustering algorithm functions for continuous data see `agnes`, `clara`, `diana`, `fanny`, and `pam` in package **cluster**, `hclust` and `kmeans` in package **stats**, `som` in package **kohonen**, `Mclust` in package **mclust**, and `sota` in package **clValid**.

For details the on the clustering algorithm functions for count data see `Cluster.RNASeq` in package **MBCluster.Seq**.

For details on the validation measure functions see `BHI`, `BSI`, `stability`, `connectivity` and `dunn` in package **clValid** and `silhouette` in package **cluster**.

## Examples

```
## These examples may each take a few minutes to compute

## Obtain Dataset
data(arabid)

## Analysis of Count Data using Internal and Stability Validation Measures
count1 <- optCluster(arabid, 2:4, clMethods = "all", countData = TRUE)
summary(count1)
# Obtain optimal clustering assignment
optAssign(count1)



## Normalize Data with Respect to Library Size
obj <- t(t(arabid)/colSums(arabid))

## Analysis of Normalized Data using Internal and Stability Validation Measures
norm1 <- optCluster(obj, 2:4, clMethods = "all")
summary(norm1)
```

```
# Obtain optimal clustering assignment
optAssign(norm1)
#Obtain clustering assignment for diana with 2 clusters
clusterResults(norm1, "diana", k = 2)$cluster

## Analysis with Only UPGMA using Internal and Stability Validation Measures
hier1 <- optCluster(obj, 2:10, clMethods = "hierarchical")
summary(hier1)

 ## Analysis of Normalized Data using All Validation Measures
 ## Note: These lines of code require the following Bioconductor
 ## packages for the biological validation measures:
 ## "Biobase", "annotate", "GO.db", and "org.At.tair.db".
 ## If all of these packages are installed, then set
 ## allBioconductorPackagesInstalled = TRUE

 allBioconductorPackagesInstalled = FALSE
 if(allBioconductorPackagesInstalled){
 require("Biobase")
 require("annotate")
 require("GO.db")
 require("org.At.tair.db")
 norm2 <- optCluster(obj, 2:4, clMethods = "all", validation = "all",
annotation = "org.At.tair.db")
 summary(norm2)
 }
```

---

optCluster-class *Class "optCluster"*

---

## Description

The class "optCluster" contains the dataset, clustering results, validation measures, ranked lists of clustering algorithms, ordered lists of validation scores, and final rank aggregation results from the function optCluster.

## Objects from the Class

The function optCluster creates objects of the class "optCluster".

## Slots

**inputData:** Object of class "matrix" containing the original dataset.

**clVal:** Object of class "clValid" containing the clustering results and validation measures from the internal clValid function.

**ranksWeights:** Object of class "list" containing the ordered ranks of clustering algorithms and the ordered validation scores for each measure.

`rankAgg`: Object of class `"raggr"` containing the rank aggregation results from the internal `RankAggreg` function.

## Methods

**optAssign** `signature(object = "optCluster")`: Returns the cluster assignment corresponding to the optimal clustering algorithm and number of clusters.

**getDataset** `signature(object = "optCluster")`: Returns the original dataset as an object of class `"matrix"`.

**getClValid** `signature(object = "optCluster")`: Returns an object of class `"clValid"`.

**methodRanks** `signature(object = "optCluster")`: Returns the ranked lists of clustering algorithms for each validation measure.

**scoreRanks** `signature(object = "optCluster")`: Returns the ordered lists of scores for each validation measure.

**getRankAggreg** `signature(object = "optCluster")`: Returns an object of class `"raggr"`.

**topMethod** `signature(object = "optCluster")`: Returns the name of the optimal clustering algorithm and number of clusters.

**measureNames** `signature(object = "optCluster")`: Returns the names of the validation measures used.

**methodNames** `signature(object = "optCluster")`: Returns the names of the clustering algorithms used.

**clusterResults** `signature(object = "optCluster")`: Returns an object of the class corresponding to the selected clustering method for each number of cluster in the analysis. If provided `k`, the object and clustering assignment for the corresponding method and number of clusters is returned.

Additional arguments:

`method = methodNames(object)` The clustering algorithm to extract. The selection of only one algorithm is allowed.

`k = NULL` The number of clusters to extract. The selection of only one number of clusters is allowed.

**valScores** `signature(object = "optCluster")`: Returns the scores from the selected validation measure(s).

Additional arguments:

`measures = measureNames(object)` The validation measure(s) to extract.

**optimalScores** `signature(object = "optCluster")`: Returns the optimal score for each validation measure as well as the corresponding clustering algorithm and number of clusters.

**print** `signature(x = "optCluster")`: Print method for class `"optCluster"`.

**show** `signature(object = "optCluster")`: Same as print.

**summary** `signature(object = "optCluster")`: Summary method for class `"optCluster"`.

## References

Sekula, M., Datta, S., and Datta, S. (2017). optCluster: An R package for determining the optimal clustering algorithm. Bioinformation, 13(3), 101. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5450252

Brock, G., Pihur, V., Datta, S. and Datta, S. (2008). clValid: An R Package for Cluster Validation. Journal of Statistical Software 25(4), https://www.jstatsoft.org/v25/i04.

Datta, S. and Datta, S. (2003). Comparisons and validation of statistical clustering techniques for microarray gene expression data. Bioinformatics 19(4): 459-466.

Pihur, V., Datta, S. and Datta, S. (2007). Weighted rank aggregation of cluster validation measures: A Mounte Carlo cross-entropy approach. Bioinformatics 23(13): 1607-1615.

Pihur, V., Datta, S. and Datta, S. (2009). RankAggreg, an R package for weighted rank aggregation. BMC Bioinformatics, 10:62, https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-10-62.

## See Also

For details on the function `optCluster` see `optCluster`.

For a description of the `clValid` function, including all available arguments that can be passed to it, see `clValid` in the **clValid** package. For a desciption of the class `"clValid"` including all available methods see `clValid-class`.

For a description of the `RankAggreg` function, including all available arguments that can be passed to it, see `RankAggreg` in the **RankAggreg** package.

## Examples

```
## This example may take a few minutes to compute

## Obtain Dataset
data(arabid)

## Normalize Data with Respect to Library Size
obj <- t(t(arabid)/colSums(arabid))

## Analysis of Normalized Data using Internal and Stability Measures
norm1 <- optCluster(obj, 2:4, clMethods = "all")

## View results
norm1
topMethod(norm1)
summary(norm1)
optimalScores(norm1)
optAssign(norm1)

## Extract cluster results for kmeans and all numbers of clusters
clusterResults(norm1, method = "kmeans")
## Extract cluster results for kmeans and 3 clusters only
clusterResults(norm1, method = "kmeans", k = 3)
```

```
## Extract all validation scores
valScores(norm1)
## Extract validations scores for APN and ADM only
valScores(norm1, measures = c("APN", "ADM"))

## Extract additional information from slots
methodNames(norm1)
measureNames(norm1)
methodRanks(norm1)
scoreRanks(norm1)
```

---

optHeatmap *Optimal Hierarchical Clustering Heat Map*

---

## Description

optHeatmap creates a heat map from an object of class "optCluster" if the optimal clustering algorithm is one of the available hierarchical algorithms: "hierarchical", "agnes", or "diana".

## Usage

```
optHeatmap(x, dendroClusters = TRUE, barClusters = FALSE,
clusterColors = "rainbow",
mapColors = colorRampPalette(c("green", "black", "red"))(256),
Colv = FALSE, dendrogram = "row", density.info = "none", ...)
```

## Arguments

| | |
|---|---|
| x | An object of class "optCluster". |
| dendroClusters | If TRUE, the row dendrogram is colored to represent the optimal k clusters. |
| barClusters | If TRUE, a colored sidebar is produced down the rows to represent the optimal k clusters. |
| clusterColors | Character vector listing the k colors for the clustering paritions. Default is rainbow(k) where k is the optimal number of clusters. |
| mapColors | Specifies colors to use for heat map image. |
| Colv | Determines if and how the column dendrogram should be reordered. If TRUE, a dendrogram is computed and the columns are reordered by means. A dendrogram object or vector of integers can also be used to specify the reordering. |
| dendrogram | Character string specifying which dendrogram(s) to display. Options include: "none", "row", "column", "both". |
| density.info | Character string specifying the type of plot to superimpose on the color-key. Options include: "none", "histogram", "density". |
| ... | Additional plotting parameters from the heatmap.2 function. |

## Details

This function utilizes the heatmap.2 function to produce a heat map based on the clustering results from the optimal clustering algorithm and number of clusters (as determined by the optCluster function). The rows in the heat map are ordered based on the optimal hierarchical clustering algorithm, with the corresponding dendrogram displayed to the left side of the map. The clustering partition of the rows into the optimal number of clusters can be visualized using the 'dendroClusters' argument and/or the 'barClusters' argument.

Because the optCluster function performs a cluster analysis on the rows of the dataset, only the rows are reordered as default in the optHeatmap function. However, columns can be reordered and displayed with a dendrogram using the 'Colv' and 'dendrogram' arguments, respectively. The heat map can also be further customized by passing additional agruments to the heatmap.2 function through the optHeatmap function.

## See Also

heatmap.2, optCluster-class

## Examples

```
## Obtain Dataset
data(arabid)

## Normalize Data with Respect to Library Size
obj <- t(t(arabid)/colSums(arabid))

## Analysis with Only UPGMA using Internal and Stability Validation Measures
hier1 <- optCluster(obj, 2:10, clMethods = "hierarchical")
topMethod(hier1)

## Create Default Heat Map Reordering Rows Only
optHeatmap(hier1)

## Create a Heat Map Reordering Both Rows and Columns
optHeatmap(hier1, Colv = TRUE, dendrogram = "both")

## Customized Heat Map Using Several heatmap.2 Arguments
optHeatmap(hier1, Colv = TRUE, dendrogram = "both",
labRow = "", cexCol = 1.0, keysize = 1)
```

---

# repRankAggreg *Repeat Rank Aggregation*

---

## Description

repRankAggreg repeats rank aggregation of ordered validation measure lists obtained from an object of class "optCluster". The function returns an object of class "optCluster".

## Usage

```
repRankAggreg(optObj, rankMethod = "same", distance = "same",
importance = NULL, rankVerbose = FALSE, ... )
```

## Arguments

| | |
|---|---|
| optObj | An object of class "optCluster". |
| rankMethod | A character string providing the method to be used for rank aggregation. As default, the "same" method as the input "optCluster" object is used. The cross-entropy Monte Carlo algorithm ("CE") or Genetic algorithm ("GA") can also be directly specified. Selection of only one method is allowed. |
| distance | A character string providing the type of distance to be used for measuring the similarity between ordered lists in rank aggregation. As default, the "same" distance as the input "optCluster" object is used. The weighted Spearman footrule distance ("Spearman") or the weighted Kendall's tau distance ("Kendall") can also be directly specified. Selection of only one distance is allowed. |
| importance | Vector of weights indicating the importance of each validation measure list. Default of NULL represents equal weights to each validation measure. See **Weighted Rank Aggregation** in the 'Details' section for more information. |
| rankVerbose | If TRUE, current rank aggregation results are displayed at each iteration. |
| ... | Additional arguments that can be passed to the internal function RankAggreg: |

- maxIter - The maximum number of iterations allowed. Default = 1000
- k - Size of top-k list in aggregation.
- convIN - Stopping criteria for CE and GA algorithms. The algorithm converges once the "best" solution does not change after convIN iterations. Default: 7 for CE and 30 for GA.
- N - Number of samples generated by MCMC in the CE algorithm. Default = 10*k^2
- rho - For CE algorithm, (rho*N) is the qunatile of candidate list sorted by function values.
- weight - For CE algorithm, the learning factor used in the probability update feature. Default = 0.25
- popSize - For GA algorithm population size in each generation. Default = 100
- CP - For GA algorithm, the crossover probability. Default = 0.4
- MP - For GA algorithm, the mutation probability. Default = 0.01

## Details

This function tests the consistency of the rank aggregation results by repeating rank aggregation with the same rank aggregation method, distance measure, clustering algorithm lists, and validation score lists used to create the input object of class "optCluster". A different rank aggregation algorithm or type of distance measure can also be evaluated using this function, but doing so may affect the final results.

**Weighted Rank Aggregation:** A list of weights for each validation measure list can be included using the `importance` argument. The default value of equal weights (NULL) is represented by rep(1, length(x)), where x is the character vector of validation measure names. This means each validation measure list has a weight of 1/length(x). To manually change the weights, the order of the validation measures selected needs to be known. The order of validation measures used in `optCluster` is provided below:

- When selected, stability measures will ALWAYS be listed first and in the following order: "APN", "AD", "ADM", "FOM".
- When selected, internal measures will only precede biological measures. The order of these measures is: "Connectivity", "Dunn", "Silhouette".
- When selected, biological measures will always be listed last and in the following order: "BHI", "BSI".

## Value

`repRankAggreg` returns an object of class `"optCluster"`. The class description is provided in the help file.

## References

Sekula, M., Datta, S., and Datta, S. (2017). optCluster: An R package for determining the optimal clustering algorithm. Bioinformation, 13(3), 101. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5450252

Pihur, V., Datta, S. and Datta, S. (2007). Weighted rank aggregation of cluster validation measures: A Mounte Carlo cross-entropy approach. Bioinformatics 23(13): 1607-1615.

Pihur, V., Datta, S. and Datta, S. (2009). RankAggreg, an R package for weighted rank aggregation. BMC Bioinformatics, 10:62, https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-10-62.

## See Also

For a description of the `RankAggreg` function, including all available arguments that can be passed to it, see `RankAggreg` in the **RankAggreg** package.

## Examples

```
## These examples may take a few minutes to compute

## Obtain Dataset
data(arabid)

## Normalize Data with Respect to Library Size
obj <- t(t(arabid)/colSums(arabid))

## Analysis of Normalized Data using Internal and Stability Validation Measures
norm1 <- optCluster(obj, 2:4, clMethods = "all")
print(norm1)
```

```
repCE <- repRankAggreg(norm1)
print(repCE)
repGA <- repRankAggreg(norm1, rankMethod = "GA")
print(repGA)
```

---

valPlot *Validation Measure Plots*

---

## Description

valPlot displays a plot of the scores for each selected validation measure.

## Usage

```
valPlot(x, measures = measureNames(x), legend = TRUE,
legendLoc = "topright", main = NULL, pch = NULL, type = "b",
ask = prod(par("mfcol")) < length(measures) && dev.interactive(), ...)
```

## Arguments

| | |
|---|---|
| x | An object of class "optCluster". |
| measures | Character vector of the names of the validation measures to plot. Any number of choices is allowed. |
| legend | If TRUE, provides a legend. |
| legendLoc | Character string specifying the location of the legend. |
| main | Character string specifying the title of graph. |
| pch | Specifies the plotting characters to use. |
| type | A character string specifying the type of plot. |
| ask | If TRUE, the user is prompted before each plot. |
| ... | Additional plotting parameters. |

## Details

The the biological homogeneity index (BHI), biological stability index (BSI), Dunn index, and silhouette width measures should all be maximized.

The average proportion of non-overlap (APN), average distance (AD), average distance between means (ADM), figure of merit (FOM), and connectivity measures should all be minimized.

## See Also

clValid-class, optCluster-class

## Examples

```
## This example may take a few minutes to compute

## Obtain Dataset
data(arabid)

## Normalize Data with Respect to Library Size
obj <- t(t(arabid)/colSums(arabid))

## Analysis of Normalized Data using Internal and Stability Validation Measures
norm1 <- optCluster(obj, 2:4, clMethods = "all")

## Plots of Internal and Stability Validation Measures
par(mfrow = c(4,2))
valPlot(norm1)

## Plots of Internal Validation Measures in a Single Figure
par(mfrow = c(2,2))
valPlot(norm1, measure = c("Dunn", "Silhouette", "Connectivity"), legend = FALSE)
plot(0, type="n", axes=FALSE, xlab = "", ylab = "")
legend("center", methodNames(norm1), col=1:9, lty=1:9, pch=paste(c(1:9)),
cex=0.8)
```

# Index