

# Package ‘gcExplorer’

May 6, 2009

**Version** 0.9-1

**Date** 2009-04-13

**Author** Theresa Scharl, Ingo Voglhuber, Friedrich Leisch

**Maintainer** Theresa Scharl <Theresa.Scharl@ci.tuwien.ac.at>

**Title** Graphical Cluster Explorer

**Description** Visualize cluster results and investigate additional properties of clusters using interactive neighbourhood graphs. By clicking on the node representing the cluster, information about the cluster is provided using additional graphics or summary statistics. For microarray data, tables with links to genetic databases like gene ontolgy can be created for each cluster.

**Depends** graphics, methods, flexclust (>= 1.0-0), Rgraphviz

**Imports** flexclust, modeltools, Rgraphviz

**Suggests** splines, lattice

**License** GPL-2

**Repository** CRAN

**Date/Publication** 2009-05-06 17:38:35

## R topics documented:

clusterPlot . . . . .	2
comp_test . . . . .	3
edgeTest . . . . .	4
fitsod . . . . .	5
gcExplorer . . . . .	7
gcModify . . . . .	9
gcOffline . . . . .	11
gcProfile . . . . .	13
gcSim . . . . .	14
gcTable . . . . .	15

go.details . . . . .	16
Group2Cluster . . . . .	18
jkdist . . . . .	19
node.tight . . . . .	20
oxygen . . . . .	22
pattern . . . . .	23
ps19 . . . . .	23
sigma . . . . .	24
write.htmltable . . . . .	25

<b>Index</b>	<b>27</b>
--------------	-----------

---

clusterPlot	<i>Cluster solution plot</i>
-------------	------------------------------

---

## Description

Plot the expression profiles of the smallest clusters of an object of class "kccasimple".

## Usage

```
## S4 method for signature 'kccasimple':
clusterPlot(object, method = c("size", "tight"), layout = c(3, 4), xlabels = NULL,
```

## Arguments

object	An object of class "kccasimple".
method	Which clusters should be plotted: either small clusters or tight clusters.
layout	A vector of the form <code>c(nr, nc)</code> . Only a subset of <code>nr x nc</code> clusters will be drawn. The arrangement of <code>nr</code> rows and <code>nc</code> columns is passed to the <code>layout</code> argument of <code>lattice</code> function <code>xyplot</code> .
xlabels	Either a numeric vector of time points giving the positions on the x-axis or a character vector with names of the positions on the x-axis.
xlab	Character string or expression giving label for the x-axis
...	Further arguments can be passed to function <code>xyplot</code> .

## Author(s)

Theresa Scharl

## Examples

```
data("hsod")
c11 <- qtclust(hsod, radius = 2, save.data = TRUE)

clusterPlot(c11, method = "tight", layout = c(3, 2))
```

---

`comp_test`*Compare Cluster Results*

---

**Description**

Cluster validation by testing the validity of a cluster solution under different experimental conditions.

**Usage**

```
comp_test(data, cll, N = 500, ...)
```

**Arguments**

<code>data</code>	Dataset with the same number of rows as the clustered dataset.
<code>cll</code>	Vector of cluster memberships of the clustered dataset.
<code>N</code>	Number of permutations.
<code>...</code>	Further arguments can be passed to the subfunctions.

**Value**

A matrix giving for each cluster the size of the cluster, the observed average within cluster distance to the computed cluster center in the new dataset, the proportion of permutations where the observed within cluster distance is lower than the permuted and the proportion of permutations where the observed within cluster distance is larger than the permuted.

**Author(s)**

Theresa Scharl

**Examples**

```
data(comp19)
set.seed(1111)
cl3 <- qtclust(comp19, radius=1.5, family=kccaFamily(dist=distEuclidean,
  cent=colMeans), save.data=TRUE, control=list(min.size=5))
cl3

ct1 <- comp_test(comp17, clusters(cl3), N=1000)
ct1
```

---

`edgeTest`*Functional Relevance Test*

---

**Description**

Perform a functional relevance test on the edges of a neighborhood graph

**Usage**

```
edgeTest(object, min.size = 1, group, N = 500, filt = 0.1, useNH = TRUE)
```

**Arguments**

<code>object</code>	An object of class "kccasimple".
<code>min.size</code>	Minimum number of grouped genes in a cluster to be considered for testing
<code>group</code>	Vector of cluster memberships of functionally grouped genes (from function <code>Group2Cluster</code> ).
<code>N</code>	Number of permutations.
<code>filt</code>	Threshold for edges in the neighborhood graph to be considered for testing.
<code>useNH</code>	Use the neighborhood structure or test all combination of nodes?

**Value**

A matrix giving the cluster sizes, the difference in proportions and the corresponding p-value for each edge considered.

**Author(s)**

Theresa Scharl

**See Also**

`Group2Cluster`

**Examples**

```
data("hsod")
data("gobp")
set.seed(1111)
c11 <- qtclust(hsod, radius = 2, save.data = TRUE)

g1 <- Group2Cluster(c11, gonr = "GO:0009061",
  source.group = gobp[,3], source.id=gobp[,1],
  id = bn_hsod)
test1 = edgeTest(c11, group=g1, min.size=2, useNH=TRUE, filt=0.1, N=1000)
```

**Description**

E. coli Fermentation Fit Data Object containing M-values, P-values, GeneNames and Links zu NCBI. Output of limma function `write.fit` with links to NCBI added for each gene.

**Usage**

```
data(fitsod)
data(hsod)
data(gobp)
```

**Format**

A data frame with 4368 observations on the following variables.

**A** a numeric vector giving the mean A-values

**Coef.stress3A** a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

**Coef.stress3B** a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

**Coef.stress3C** a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

**Coef.stress3F** a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

**Coef.stress4** a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

**Coef.stress4A** a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

**Coef.stress5A** a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

**Coef.stress6** a numeric vector for each coefficient giving the estimated coefficient for a particular gene for the contrast to the reference

**t.stress3A** a numeric vector giving the t-statistic to the coefficient estimate

**t.stress3B** a numeric vector giving the t-statistic to the coefficient estimate

**t.stress3C** a numeric vector giving the t-statistic to the coefficient estimate

**t.stress3F** a numeric vector giving the t-statistic to the coefficient estimate

**t.stress4** a numeric vector giving the t-statistic to the coefficient estimate

**t.stress4A** a numeric vector giving the t-statistic to the coefficient estimate

**t.stress5A** a numeric vector giving the t-statistic to the coefficient estimate

**t.stress6** a numeric vector giving the t-statistic to the coefficient estimate  
**p.value.stress3A** a numeric vector giving the corresponding p-value  
**p.value.stress3B** a numeric vector giving the corresponding p-value  
**p.value.stress3C** a numeric vector giving the corresponding p-value  
**p.value.stress3F** a numeric vector giving the corresponding p-value  
**p.value.stress4** a numeric vector giving the corresponding p-value  
**p.value.stress4A** a numeric vector giving the corresponding p-value  
**p.value.stress5A** a numeric vector giving the corresponding p-value  
**p.value.stress6** a numeric vector giving the corresponding p-value  
**F** a numeric vector giving the overall F-statistik  
**F.p.value** a numeric vector giving the corresponding F-p-value  
**Genes.Block** first block position of the gene  
**Genes.Row** first row position of the gene  
**Genes.Column** first column position of the gene  
**Genes.GeneName** Short genename  
**Genes.ID** Gene ID  
**Genes.AccessionReference** Blattner numbers  
**Genes.Status** status of the gene: always gene  
**links** link to NCBI

## Details

The data set `hsod` is a filtered subset of the original data. It contains 527 differentially expressed genes at the 8 time points. The vector `bn_hsod` contains the corresponding identifiers which can be used to search for functional groups in the data set `gobp`. `links_hsod` contains the corresponding links to the NCBI database.

`gobp` is a data set listing functional groups to gene identifiers. The data set consists of 8726 observations, the first column gives the gene identifier, the second column gives the gene name and the third column gives the functional group.

## References

Duerrschmid, K., Reischer, H., Schmidt-Heck, W., Hrebicek, T., Guthke, R., Rizzi, A., Bayer, K. (2008). Monitoring of transcriptome and proteome profiles to investigate the cellular response of *E. coli* towards recombinant protein expression under defined chemostat conditions. *Journal of Biotechnology* 135, 34–44.

## Examples

```
data(fitsod)
```

**Description**

Plot a neighborhood graph for "kccasimple" cluster solutions.

**Usage**

```
## S4 method for signature 'kccasimple':
gcExplorer(object, layout = c("dot", "neato", "twopi", "circo", "fdp"),
  theme = "grey", edge.method=c("orig", "mean", "min", "max"),
  node.function = NULL, node.args = NULL, doViewPort = FALSE,
  filt = 0.1, interactive = !is.null(panel.function), dev=c("one", "many"),
  panel.function = NULL, panel.args = NULL, bgdata = NULL,
  colscale = NULL, mfrow = c(1,1), legend.pos = "none")
```

**Arguments**

object	Object of class "kccasimple".
layout	Layout method used: One of "dot", "neato", "twopi", "circo", and "fdp".
theme	Color theme used.
edge.method	Several methods are available to draw edges: "orig", "mean", "min", and "max", see details below.
node.function	Optional. Additional information about the clusters can be included in the representation of nodes. Either a function calculating node colors or a grid-based function (see doViewPort).
node.args	List of arguments which should be passed to node.function.
doViewPort	Currently not used in release version of the package. Call a grid-based function specified by argument node.function and use it for node representation?
filt	Cutoff value for similarities between clusters, edges above the threshold will be displayed.
interactive	Should the plot be interactive?
dev	Only used if interactive=TRUE. Display each cluster plot (specified by panel.function) in one device or open new devices for each cluster when clicking on a node.
panel.function	Only used if interactive=TRUE. The panel function which should be used to display the corresponding cluster
panel.args	List of arguments which should be passed to panel.function.
bgdata	Background data to be plotted by panel.function or node.function.

<code>colscale</code>	A vector of length 2 specifying the color range for edges and nodes, e.g. <code>c(0,0.5)</code> .
<code>mfrow</code>	Only used if <code>interactive=TRUE</code> . The panel layout in which the panel plots should be displayed.
<code>legend.pos</code>	Position of the legend.

## Details

A neighborhood graph is the default plot method for cluster objects of package `flexclust`. For large and highdimensional data sets like microarray data linear projection of the data into two dimensions may not scale well in the number of clusters. In this case non-linear arrangement of the nodes using layout algorithms from Graphviz can be helpful. An interface to Graphviz is provided in Bioconductor package `Rgraphviz`. One of the implemented layout algorithms can be selected using `layout`.

In a neighborhood graph each node corresponds to a cluster centroid. Two nodes are connected by an edge if there exist data points that have these two centroids as closest and second closest. The edge weights are taken from `clusterSim(object)`. The similarity between two clusters is bounded between 0 and 1 where well-separated clusters have values close to 0. The larger the similarity between clusters the stronger the edge will be drawn in the graph. The cutoff value for drawing the edge between two centroids can be chosen by argument `filt`. The larger the `filt` value the fewer edges will be drawn.

Originally the neighborhood graph is a directed graph. An edge will be drawn from centroid 1 to centroid 2 if there exists at least one data point that has centroid 1 as closest and centroid 2 as second closest. But there need not necessarily be a data point that has centroid 2 as closest and centroid 1 as second closest centroid. For this reason there are several methods for plotting the edges between nodes. The default `edge.method` is 'orig' where each edge is drawn separately with its corresponding weight. This method will result in a directed graph. All other edge methods yield undirected graphs where the mean, minimum or maximum of the similarities between two clusters is used.

Additional information about the clusters can be included in the graph using `node.function` and `panel.function`. `node.function` is used for the node representation. If no `node.function` is given all nodes will be drawn in one color. The `node.function` can be used to calculate different colors for the nodes like cluster size or cluster tightness. Additionally `node.function` can be a grid-based function displaying the data in the underlying cluster, e.g. a scatterplot or a boxplot.

`gcExplorer` is implemented interactively. If `interactive=TRUE` `panel.function` is used to plot a cluster when clicking on the corresponding node. An example of a `panel.function` is given by function `gcProfile`.

Function `calcHCL` is used to calculate a HCL-based color.

## Value

Object of class "graphdata" with the following slots: an object of class "Ragraph" (see package `Rgraphviz`), `object`, `bgdata`, `node.function`, `edge.method`, `theme` and `colscale`.

**Author(s)**

Theresa Scharl and Ingo Voglhuber

**References**

Theresa Scharl and Friedrich Leisch. gcExplorer: Interactive Exploration of Gene Clusters. *Bioinformatics*, 25(8): 1089-1090, 2009.

**See Also**

[node.tight](#)

**Examples**

```
data("hsod")
cl1 <- qtclust(hsod, radius = 2, save.data = TRUE)

gcExplorer(cl1, theme = "blue", node.function = node.size)
```

---

gcModify

*Modify Ragraph Objects and Replot them*

---

**Description**

gcModify is a function to modify and plot an object of class "graphdata":

- remove edges/nodes
- zoom
- draw custom node plots

**Usage**

```
## S4 method for signature 'graphdata':
gcModify(graphdata, clsim = NULL, rmNodes = NULL,
         kpNodes = NULL, edgeDep = TRUE, nodeDep = FALSE,
         zoom = c("none", "manual", "auto"),
         keepAspectRatio = TRUE, node.function = NULL,
         doViewport = TRUE, bgdata = NULL)
```

**Arguments**

graphdata	list, containing object of class "Ragraph", object of class "kcca" and other parameters of graph created by gcExplorer).
clsim	matrix, new clsim to define removal or modification of edges.
rmNodes	character vector, names of nodes to remove. (can not be used in combination with kpNodes)



```

## create Ragraph object from kcca object with gcExplorer
graph <- gcExplorer(c11, theme = "blue", node.function = node.size)

## extract and modify clsim
clsim <- clusterSim(c11)
clsim[clsim < 0.5] <- 0

## use modified clsim on Ragraph object to remove edges (<0.5)
gcModify(graph, clsim)

## use nodeDep=TRUE to delete nodes without edges
gcModify(graph, clsim, nodeDep = TRUE, zoom = "none")

## use zoom="auto" to center and maximize subgraph
gcModify(graph, clsim, nodeDep = TRUE, zoom = "auto")

## Not run:
## R package symbols is available from Rforge:
## http://r-forge.r-project.org/projects/symbols/

require("symbols")

## create a grid based plotting function: plot cluster data and centers in matplot.
gmatplot <- function (object, cluster, bgdata) {
  grid.rect()
  data <- object@data@get("designMatrix")
  ylimits <- c(min(data, na.rm = TRUE), max(data, na.rm = TRUE))
  index <- (object@cluster == cluster)
  nodedata <- data[index,]
  symb.matplot(1:ncol(nodedata), t(nodedata), type = "l",
              col = "gray", ylim = ylimits, pch = 1)
  center <- object@centers[cluster,]
  symb.matplot(1:ncol(object@centers), center, type = "l",
              col = "red", ylim = ylimits, pch = 1)
}
## use grid based node function to draw nodes
gcModify(graph, clsim, nodeDep = TRUE, zoom = "auto",
         node.function = gmatplot)
## End(Not run)

```

---

gcOffline

*Offline gcExplorer*


---

## Description

Save gcExplorer plots or tables to a file.

**Usage**

```
## S4 method for signature 'kccasimple':
gcOffline(object, panel.function, panel.args=NULL,
          type=pdf, file="gcOffline", which=NULL, html=FALSE, ...)
```

**Arguments**

<code>object</code>	Object of class "kccasimple".
<code>panel.function</code>	Only used if <code>interactive=TRUE</code> . The panel function which should be used to display the corresponding cluster
<code>panel.args</code>	List of arguments which should be passed to <code>panel.function</code> .
<code>type</code>	Create graphics of type <code>type</code> , e.g., pdf, postscript, jpeg, png.
<code>file</code>	File name prefix used for graphics files. Of the form <code>file-which.type</code> and <code>file-graph.type</code> .
<code>which</code>	A vector specifying if all cluster plots (default) or only a subset should be created.
<code>html</code>	Logical. Does the <code>panel.function</code> produce HTML tables.
<code>...</code>	Further arguments can be passed to <code>gcExplorer</code> .

**Author(s)**

Theresa Scharl

**See Also**

[gcTable](#), [gcProfile](#)

**Examples**

```
data("hsod")
set.seed(1111)
cl1 <- qtclust(hsod, radius=2, save.data=TRUE)

# create three files: hsod-003.pdf, hsod-005.pdf, hsod-graph.pdf
gcOffline(cl1, panel.function=gcProfile, file="hsod", which=c(3,5))

# create two files: hsod-003.html, hsod-005.html
gcOffline(cl1, panel.function = gcTable, html = TRUE,
          panel.args = list(links = links_hsod),
          file = "hsod", which=c(3,5))

# tidy up
unlink(list.files()[grep("hsod-", list.files())])
```

gcProfile

*Plot for cluster results***Description**

Plot a single cluster of a 'kccasimple' object.

**Usage**

```
## S4 method for signature 'kccasimple':
gcProfile(object, which, data = NULL, cex1 = 0.8, xlab = "",
  ylab = "M", ylim=c(-6,6), cex.axis=1, xlabels=NULL,
  opar = par(las=1, mar=c(5, 4, 2, 0.5) + 0.1),
  data.type=c("time", "other"), legend=TRUE, ...)
```

**Arguments**

object	an object of class "kccasimple"
data	Plot either the data stored in object or external data.
which	Number of the cluster.
cex1	Point size of the legend.
xlab	Label for the x-axis.
ylab	Label for the y-axis.
ylim	Range of the y-axis.
cex.axis	Point size of x-axis.
xlabels	Positions on the x-axis. Default is 1:ncol(data).
opar	Graphical parameters.
data.type	If the data come from arbitrary source (default) colnames of the data are used as xlabels if not stated otherwise using xlabels. If the data comes from a time course experiment x-values start at 0 and different time intervals are supported.
legend	Logical. Should a legend be drawn?.
...	Further arguments can be passed to matplot.

**Author(s)**

Theresa Scharl

**Examples**

```
data("hsod")
cl1 <- qtclust(hsod, radius=2, save.data=TRUE)

gcProfile(cl1, which=5)
gcProfile(cl1, which=5, xlabels=c(0,8,15,22,45,68,90,150,180),
  xlab="time after induction [min]",data.type="time")
```

gcSim

*Create artificial cluster data***Description**

Functionality to create artificial time course gene cluster data.

**Usage**

```
gcSim(sim=c("arima", "norm", "pattern", "noise", "outlier"), time=10,
      sd=0.1, sd.ri=0, size=50, n=10, ar=NULL, o=NULL, cent)

gcData(...)
```

**Arguments**

sim	simulation method used
time	number of time points
sd	standard deviation of the expression profiles
sd.ri	standard deviation of the random intercept or gene specific shift
size	cluster size, either one value for all clusters or a vector of cluster sizes of length n
n	number of clusters
ar	any value between -1 and 1
o	the degree of differencing
cent	a data matrix giving expression profiles in rows, only used if sim="pattern" or sim="outlier"
...	Several "gcSim" objects can be combined using function gcData.

**Details**

gcSim is a unifying function to call different data simulators.

arima generates expression patterns that come from an integrated AR-process with AR order 1 that can be controlled via ar and the degree of differencing o. sim="norm" and sim="noise" generate normally distributed expression patterns where sim="noise" is used to form a noise set of genes.

sim="pattern" and sim="outlier" can be used to generate clusters based on a set of cluster centers which are passed to the functions using the argument cent. sim="outlier" can be used to test Jackknife distance measures.

gcData can be used to combine different artificial data generators.

**Value**

a data matrix

**Author(s)**

Theresa Scharl

**See Also**[pattern](#)**Examples**

```
## generate 10 clusters with normally distributed expression patterns:
data <- gcSim(sim="norm", time=16, sd=0.1, sd.ri=0.5,
             size=50, n=10)
matplot(t(data), type="l", pch=1)

## combine expression patterns that follow an ARIMA process and a null cluster:

data <- gcData(gcSim(sim="arima", time=16, sd=0.1, sd.ri=0.5,
                    size=c(20,50,100,100), n=4),
              gcSim(sim="noise", time=16, size=100, sd=0))
matplot(t(data), type="l")
```

gcTable

*HTML table for cluster results***Description**

Create HTML table for a single cluster of a "kccasimple" object.

**Usage**

```
## S4 method for signature 'kccasimple':
gcTable(object, which, links, file="gcTable", ...)
```

**Arguments**

object	an object of class "kccasimple"
which	Number of the cluster.
links	Vector of the same length as rows in the data with links to a database.
file	File name prefix used for HTML tables. Of the form file-which.html.
...	Further arguments can be passed to <code>write.htmltable</code> .

**Author(s)**

Theresa Scharl

**See Also**[write.htmltable](#)

**Examples**

```

data("hsod")
cl1 <- qtclust(hsod, radius=2, save.data=TRUE)

gcTable(cl1, which=5, links = links_hsod, file = "hsod")
## Not run:
gcExplorer(cl1, theme = "blue", panel.function = gcTable,
           panel.args = list(links = links_hsod, file="hsod"),
           node.function = node.size)
## End(Not run)

```

go.details

*Functional Group Methods***Description**

Plot or extract size, members or data of a functional group

**Usage**

```

## S4 method for signature 'data.frame':
go.details(object, mvalues, gn, id, stats, links, gonr,
           source.id, source.group, details = c("size", "names", "id", "data"),
           table = TRUE, file = "go.details", plot = TRUE, cex1 = 0.8,
           xlab = "", xlabel = NULL, ylab = "M", ylim = c(-6,6), cex.axis = 1, ...)

```

**Arguments**

object	An object of class "data.frame".
mvalues	Vector giving the columns in object which correspond to the gene expression values.
gn	Column of object which corresponds to the gene names used for representation.
id	Column of object which corresponds to the unique IDs of the same type as given in source.id.
links	Column of object which corresponds to links to database.
stats	Column(s) of object which correspond to statistics.
gonr	Unique identifier from source.group giving the group of genes to be extracted.
source.id	Vector of gene IDs assigned to functional groups given in source.group.
source.group	Vector of the same length as source.id.
details	The type of details to be extracted.
table	Logical. Should an html table be created.

file	The file where the output of 'gotable' will be written.
plot	Logical. Should the genes be plotted.
cexl	Point size of the legend.
xlab	Label for the x-axis.
xlabels	Either a numeric vector of time points giving the positions on the x-axis or a character vector with names of the positions on the x-axis.
ylab	Label for the y-axis.
ylim	Range of the y-axis.
cex.axis	Point size of the axis.
...	Further arguments can be passed to <code>matplot</code> or <code>write.htmltable</code> .

**Author(s)**

Theresa Scharl

**See Also**[fitsod](#)**Examples**

```

data(fitsod)
data(gobp)

## Plot the functional group
go.details(fitsod, mvalues = 2:9, gn = 31, id = 33, links = 35, stats = 26,
           gonr = "flagellar", source.group = gobp[,3], source.id = gobp[,1],
           plot = TRUE)

## A file named "go.details.html" will be created in the current
## working directory.
go.details(fitsod, mvalues = 2:9, gn = 31, id = 33, links = 35, stats = 26,
           gonr = "flagellar", source.group = gobp[,3], source.id = gobp[,1],
           table = TRUE)

## Names of the genes in functional group "flagellar"
go.details(fitsod, mvalues = 2:9, gn = 31, id = 33, links = 35, stats = 26,
           gonr = "flagellar", source.group = gobp[,3], source.id = gobp[,1],
           details = "names")

## Gene expression values of the functional group
d1 <- go.details(fitsod, mvalues = 2:9, gn = 31, id = 33, links = 35, stats = 26,
                gonr = "flagellar", source.group = gobp[,3], source.id = gobp[,1],
                details = "data")
dim(d1)

```

---

Group2Cluster      *Find clusters to a group*

---

### Description

Find the cluster memberships for a group and create the vector of all cluster memberships where the grouped elements are assigned to.

### Usage

```
Group2Cluster(object, gonr, source.group, source.id, id)
Random2Cluster(object, perc)
DefinedCluster(object, filt=0, numEdges=6, perc=1, noise=0)
```

### Arguments

object	An object of class <code>kccasimple</code>
gonr	Unique identifier from <code>source.group</code> giving the group of genes to be extracted
source.group	Vector of functional groups where <code>source.id</code> are assigned to
source.id	Corresponding vector of identifiers to <code>source.group</code>
id	Vector of identifiers of the same length as rows in the clustered data of the same type as given in <code>source.id</code>
perc	For artificial assignment: the percentage of elements in a cluster that should be assigned to the group
filt	Edges above this threshold are taken into account
numEdges	Number of edges chosen where clusters are assigned similar amount of affected elements
noise	The percentage of noise that should be added (i.e., further assigned elements in different clusters)

### Value

A vector of cluster memberships.

### Author(s)

Theresa Scharl

### See Also

`edge.test`

**Examples**

```

data("hsod")
data("gobp")
set.seed(1111)
cl1 <- qtclust(hsod, radius = 2, save.data = TRUE)

g1 <- Group2Cluster(cl1, gonr = "GO:0009061",
  source.group = gobp[,3], source.id=gobp[,1],
  id = bn_hsod)
table(g1)

```

jkdist

*Further Distance and Centroid Computations***Description**

Helper functions to create 'kccaFamily' objects.

**Usage**

```

distJackCor(x, centers)
distJackEuc(x, centers)
distJackMan(x, centers)
distJackMax(x, centers)

centSpline(d)

```

**Arguments**

x	A data matrix
d	A data matrix
centers	A matrix of centroids

**Details**

A possible problem using classical distance measures for clustering time-course gene expression data is that single outlier variables can completely change the expression pattern of certain genes. Outliers at special time points are very common in microarray experiments as technical problems like dust or a scratch on the slide can easily distort the data. In such a case these outlier variables can lead to unwanted correlations between genes and to incorrect assignment to clusters. There is a need for distance measures which are robust against outlier variables. The idea of Jackknife (Efron, 1982) distance measures is not to exclude the whole observation for such a gene but rather one or several variables. We want to introduce so-called "Jackknife" distance measures which can handle one outlier time point. The so-called Jackknife correlation was first used by Heyer et al. (1999) to cluster gene expression data. It is defined as

$$d_{xy} = 1 - \min(\rho_{xy}^{(1)}, \rho_{xy}^{(2)}, \dots, \rho_{xy}^{(T)})$$

where  $\rho_{xy}^{(t)}$  is the correlation of pair x,y computed with the t-th time point deleted.

This concept can be extended for the three geometric distance measures Euclidean, Manhattan and Maximum distance. Jackknife Euclidean distance is defined as

$$d_{xy} = \min(d_{xy}^{(1)}, d_{xy}^{(2)}, \dots, d_{xy}^{(T)})$$

where  $d_{xy}^{(t)}$  is the Euclidean distance of pair x,y computed with the t-th time point deleted. Jackknife Manhattan distance and Jackknife Maximum distance can be defined in the same way.

### Author(s)

Theresa Scharl

### References

Theresa Scharl and Friedrich Leisch: Jackknife distances for clustering time-course gene expression data, in JSM Proceedings 2006

---

node.tight

*Node Methods for Neighborhood Graphs*

---

### Description

Several methods how to color nodes of a neighborhood graph.

### Usage

```
## S4 method for signature 'kccasimple':
node.tight(object, theme, colscale)
## S4 method for signature 'kccasimple':
node.size(object, theme, colscale)
## S4 method for signature 'kccasimple':
node.go(object, theme, colscale, gonr, source.group, source.id, id)
## S4 method for signature 'kccasimple':
node.group(object, theme, colscale, group)
## S4 method for signature 'kccasimple':
legend.size(object, theme, colscale=NULL, pos="bottomleft")
## S4 method for signature 'kccasimple':
legend.tight(object, theme, colscale=NULL, pos="bottomleft")
```

### Arguments

object	An object of class "kccasimple"
theme	A color theme, eg. theme="blue".
colscale	Range of luminescence lum of hcl colors, default is min to max.
gonr	Unique identifier from source.group giving the group of genes to be extracted.

<code>source.id</code>	Vector of gene IDs assigned to functional groups given in <code>source.group</code> .
<code>source.group</code>	Vector of the same length as <code>source.id</code> .
<code>id</code>	Vector of identifiers of the same length as rows in the clustered data of the same type as given in <code>source.id</code> .
<code>group</code>	Vector of integers giving the cluster membership of grouped genes.
<code>pos</code>	Position where the legend should be placed.

### Details

Function `node.size` is used to highlight large clusters where the largest cluster will be assigned the darkest color.

Function `node.tight` is used to highlight tight clusters where the tightest cluster will be assigned the darkest color.

Function `node.go` is used to highlight clusters with accumulation of the functional group given by `gonr` where the highest proportion will be assigned the darkest color.

Function `node.group` is used to highlight clusters with accumulation of a functional group where the class membership are passed by argument `group`. Again the highest proportion will be assigned the darkest color.

### Author(s)

Theresa Scharl and Ingo Voglhuber

### See Also

[gcExplorer](#)

### Examples

```
data("hsod")
set.seed(1111)
c11 <- qtclust(hsod, radius = 2, save.data = TRUE)

gcExplorer(c11, theme = "blue", node.function = node.size,
           legend.pos= "topleft")

gcExplorer(c11, theme = "red", node.function = node.tight,
           legend.pos= "topleft")

data("gobp")
gcExplorer(c11, theme = "green", node.function = node.go,
           node.args = list(gonr = "transport", source.group = gobp[,3],
                           source.id = gobp[,1], id = bn_hsod),
           legend.pos= "topleft")
```

---

`oxygen`*Preprocessed microarray oxygen deprivation data*

---

**Description**

Normalized gene expression microarray data from Escherichia coli (E. coli).

**Usage**

```
data(oxygen)
```

**Format**

`oxygen` is a data matrix containing  $n=43$  experiments of various mutants under oxygen deprivation (Covert et al., 2004). The mutants were designed to monitor the response from E. coli during an oxygen shift in order to target the a priori most relevant part of the transcriptional network by using six strains with knockouts of five key transcriptional regulators in the oxygen response (*arcA*, *appY*, *fnr*, *oxyR* and *soxS*). The data was obtained by downloading the corresponding CEL files from the Gene Expression Omnibus (<http://www.ncbi.nlm.nih.gov/geo>) under accession GDS680 and then normalized using the `rma()` function from the `affy` package. Following the steps described in (Castelo and Roverato, 2008) probesets were mapped to Entrez Gene Identifiers and filtered such that the `ExpressionSet` in the `qpgraph` package names `EcoliOxygen` contains a total of  $p=4205$  genes. Here a subset of the `EcoliOxygen` data was used containing all genes where Blattner numbers were available.

**Note**

This dataset was taken from Bioconductor package `qpgraph` and modified

**Source**

Covert, M.W., Knight, E.M., Reed, J.L., Herrgard, M.J., and Palsson, B.O. Integrating high-throughput and computational data elucidates bacterial networks. *Nature*, 429(6987):92-96, 2004.

Gama-Castro, S., Jimenez-Jacinto, V., Peralta-Gil, M., Santos-Zavaleta, A., Penaloza-Spinola, M.I., Contreras-Moreira, B., Segura-Salazar, J., Muniz-Rascado, L., Martinez-Flores, I., Salgado, H., Bonavides-Martinez, C., Abreu-Goodger, C., Rodriguez-Penagos, C., Miranda-Rios, J., Morett, E., Merino, E., Huerta, A.M., Trevino-Quintanilla, L., and Collado-Vides, J. RegulonDB (version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation. *Nucleic Acids Res.*, 36(Database issue):D120-124, 2008.

Castelo, R. and Roverato, A. Reverse engineering molecular regulatory networks from microarray data with qp-graphs. *J. Comp. Biol.*, *accepted*, 2008.

**Examples**

```
data(oxygen)
```

---

pattern	<i>Expression pattern</i>
---------	---------------------------

---

**Description**

Expression patterns that can be used to generate artificial gene expression data.

**Usage**

```
pattern(time = 8, v = 5)
```

**Arguments**

time	number of time points
v	absolute value of maximum gene expression

**Value**

a data matrix

**Author(s)**

Theresa Scharl

**See Also**

[gcSim](#)

**Examples**

```
cent <- pattern(time=15)
data <- gcSim(sim="pattern", cent=cent)
matplot(t(data), type="l")
```

---

ps19	<i>E. coli Fermentation Data</i>
------	----------------------------------

---

**Description**

E. coli Fermentation Data - Transcription profiling of E. coli HMS174(DE3)(pET30aNproGFPmut3.1)  
- cellular response to limited induction with IPTG

**Usage**

```
data(ps19)
```

**Format**

A data frame with 918 observations on the following 10 variables.

- 10 Estimated coefficient for a particular gene for the contrast of the sample 10 hours past induction to the sample before induction
- 12 Estimated coefficient for a particular gene for the contrast of the sample 12 hours past induction to the sample before induction
- 14 Estimated coefficient for a particular gene for the contrast of the sample 14 hours past induction to the sample before induction
- 16 Estimated coefficient for a particular gene for the contrast of the sample 16 hours past induction to the sample before induction
- 18 Estimated coefficient for a particular gene for the contrast of the sample 18 hours past induction to the sample before induction
- 20 Estimated coefficient for a particular gene for the contrast of the sample 20 hours past induction to the sample before induction
- 22 Estimated coefficient for a particular gene for the contrast of the sample 22 hours past induction to the sample before induction
- 24 Estimated coefficient for a particular gene for the contrast of the sample 24 hours past induction to the sample before induction
- 26 Estimated coefficient for a particular gene for the contrast of the sample 26 hours past induction to the sample before induction
- 28 Estimated coefficient for a particular gene for the contrast of the sample 28 hours past induction to the sample before induction

**Source**

Array Express Number tbd

**References**

MCF Publication tbd

**Examples**

```
data(ps19)
```

---

sigma

*E. coli Sigma Factors and Global Regulators*

---

**Description**

The *E. coli* sigma factors and the genes they regulate.

**Usage**

```
data(sigma)
```

**Format**

A data frame with 1851 observations on the following 6 variables.

**SigmaFactor** a factor with levels Sigma19 Sigma24 Sigma28 Sigma32 Sigma38 Sigma54 Sigma70

**SigmaGene** a factor with levels fecI fliA rpoD rpoDS rpoE rpoH rpoN rpoS

**RegulatedGeneName** The genename of the regulated genes.

**RegulatedGenebnumber** The Blattner numbers of the regulated genes.

**function** a factor with levels +

**GeneType** a factor with levels Phantom Gene Pseudo Gene

**Source**

<http://regulondb.ccg.unam.mx/LicenseRegulonDBd.jsp>

**References**

Salgado H, Gama-Castro S, Peralta-Gil M, Diaz-Peredo E, Sanchez-Solano F, Santos-Zavaleta A, Martinez-Flores I, Jimenez-Jacinto V, Bonavides-Martinez C, Segura-Salazar J, Martinez-Antonio A, Collado-Vides J. RegulonDB (version 5.0): Escherichia coli K-12 transcriptional regulatory network, operon organization, and growth conditions Nucleic Acids Res. 2006 Jan 1;34(Database issue):D394-7

**Examples**

```
data(sigma)
data(reg)
```

---

```
write.htmltable      Write a data frame into an html table within a html page
```

---

**Description**

Write a "data.frame" into an html table within a html page

**Usage**

```
write.htmltable(x, filename, title="", sortby=NULL, decreasing=TRUE,
               open="wt", formatNumeric=function(x) paste(signif(x, 3)))
```

**Arguments**

<code>x</code>	<code>data.frame</code> .
<code>filename</code>	character. File name.
<code>title</code>	character. Title of html page.
<code>sortby</code>	character. Name of column by which to sort the table rows.
<code>decreasing</code>	logical. Should the sort order be increasing or decreasing?
<code>open</code>	character. This argument is passed on to <code>file</code> .
<code>formatNumeric</code>	function that takes a numeric and returns a character. This function is called for all numeric values in the table.

**Details**

This function is taken from package `arrayMagic`.

**Value**

The function is called for its side effect: writing a file.

**Author(s)**

Wolfgang Huber <http://www.dkfz.de/mga/whuber>

**Examples**

```
out = tempfile()

n = 10
ex = data.frame(genename=paste("Gene", 1:n, sep=""), score=
  signif(16*runif(n)), database=paste("http://super.data.base/?id",
  round(1e9*runif(n)), sep=""))

write.htmltable(ex, out, "Hi there", sortby="score")

cat("Now have a look at ", out, ".html\n", sep="")
```

# Index

- \*Topic **IO**
  - write.htmltable, 25
- \*Topic **cluster**
  - gcSim, 14
  - jkdist, 19
  - pattern, 23
- \*Topic **datasets**
  - fitsod, 5
  - oxygen, 22
  - ps19, 23
  - sigma, 24
- \*Topic **hplot**
  - clusterPlot, 2
  - gcExplorer, 7
  - gcModify, 9
  - gcOffline, 11
  - gcProfile, 13
  - go.details, 16
- \*Topic **htest**
  - comp\_test, 3
  - edgeTest, 4
- \*Topic **methods**
  - clusterPlot, 2
  - gcExplorer, 7
  - gcModify, 9
  - gcOffline, 11
  - gcProfile, 13
  - gcTable, 15
  - go.details, 16
  - Group2Cluster, 18
  - node.tight, 20
  
- bn\_hsod (*fitsod*), 5
- bn\_oxy (*oxygen*), 22
- bn\_ps19 (*ps19*), 23
  
- calcHCL (*gcExplorer*), 7
- centSpline (*jkdist*), 19
- clusterPlot, 2
  
- clusterPlot, kccasimple-method  
    (*clusterPlot*), 2
- comp17 (*ps19*), 23
- comp19 (*ps19*), 23
- comp\_diff (*edgeTest*), 4
- comp\_dist (*comp\_test*), 3
- comp\_perm (*comp\_test*), 3
- comp\_test, 3
  
- DefinedCluster (*Group2Cluster*), 18
- distJackCor (*jkdist*), 19
- distJackEuc (*jkdist*), 19
- distJackMan (*jkdist*), 19
- distJackMax (*jkdist*), 19
  
- edgeTest, 4
  
- f (*ps19*), 23
- file, 26
- fitsod, 5, 17
  
- gcData (*gcSim*), 14
- gcExplorer, 7, 10, 21
- gcExplorer, kccasimple-method  
    (*gcExplorer*), 7
- gcModify, 9
- gcModify, graphdata-method  
    (*gcModify*), 9
- gcOffline, 11
- gcOffline, kccasimple-method  
    (*gcOffline*), 11
- gcProfile, 12, 13
- gcProfile, kccasimple-method  
    (*gcProfile*), 13
- gcSim, 14, 23
- gcTable, 12, 15
- gcTable, kccasimple-method  
    (*gcTable*), 15
- go.details, 16
- go.details, data.frame-method  
    (*go.details*), 16

`gobp(fitsod)`, 5  
`graphdata-class(gcExplorer)`, 7  
`Group2Cluster`, 18

`hsod(fitsod)`, 5

`jkdist`, 19

`legend.size(node.tight)`, 20  
`legend.size,kccasimple-method`  
    `(node.tight)`, 20  
`legend.tight(node.tight)`, 20  
`legend.tight,kccasimple-method`  
    `(node.tight)`, 20  
`links_hsood(fitsod)`, 5  
`links_ps19(ps19)`, 23

`newclsim(edgeTest)`, 4  
`node.go(node.tight)`, 20  
`node.go,kccasimple-method`  
    `(node.tight)`, 20  
`node.group(node.tight)`, 20  
`node.group,kccasimple-method`  
    `(node.tight)`, 20  
`node.size(node.tight)`, 20  
`node.size,kccasimple-method`  
    `(node.tight)`, 20  
`node.tight`, 9, 20  
`node.tight,kccasimple-method`  
    `(node.tight)`, 20

`oxygen`, 22

`pattern`, 15, 23  
`ps19`, 23  
`pvalFromPermMat(comp_test)`, 3

`Random2Cluster(Group2Cluster)`, 18  
`reg(sigma)`, 24

`sigma`, 24

`write.htmltable`, 15, 25