Package 'CSFA'

August 14, 2018

Type Package

Title Connectivity Scores with Factor Analysis

Version 1.2.0

Date 2018-08-13

Author Ewoud De Troyer

Maintainer Ewoud De Troyer <ewoud.detroyer@uhasselt.be>

Description Applies factor analysis methodology to microarray data in order to derive connectivity scores between compounds. The package also contains an implementation of the connectivity score algorithm by Zhang and Gant (2008) <doi:10.1186/1471-2105-9-258>.

License GPL-3

Imports methods, stats, graphics, grDevices, fabia, pls, FactoMineR, elasticnet, randomcoloR, parallel, snowFT

VignetteBuilder knitr

Suggests knitr, gplots, viridis

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-08-13 22:40:02 UTC

R topics documented:

`CSanalysis` *Connectivity Score Analysis.*

Description

Doing a CS analysis, interactively generating graphs. See specific type for additional parameteres. Types:

- `Zhang and Gant`
- `MFA`
- `PCA`
- `Sparse MFA`
- `FABIA`

Usage

```
CSanalysis(querMat, refMat, type, ...)
```

Arguments

`querMat`	Query matrix (Rows = genes and columns = compounds)
`refMat`	Reference matrix
`type`	Type of Factor Analysis or Zhang & Gant (`"CSfabia"`, `"CSmfa"`, `"CSpca"`, `"CSsmfa"` or `"CSzhang"`)
`...`	Additional parameters for analysis

Value

An object of the S4 Class `CSresult-class`.

Examples

```
data("dataSIM",package="CSFA")
Mat1 <- dataSIM[,c(1:6)]
Mat2 <- dataSIM[,-c(1:6)]

MFA_analysis <- CSanalysis(Mat1,Mat2,"CSmfa")
FABIA_analysis <- CSanalysis(Mat1,Mat2,"CSfabia")
ZHANG_analysis <- CSanalysis(Mat1,Mat2,"CSzhang")
```

CSanalysis,matrix,matrix,character-method

Connectivity Score Analysis.

Description

Doing a CS analysis, interactively generating graphs. See specific type for additional parameteres. Types:

- Zhang and Gant
- MFA
- PCA
- Sparse MFA
- FABIA

Usage

```
## S4 method for signature 'matrix,matrix,character'
CSanalysis(querMat, refMat, type, ...)
```

Arguments

querMat	Query matrix (Rows = genes and columns = compounds)
refMat	Reference matrix
type	Type of Factor Analysis or Zhang & Gant ("CSfabia", "CSmfa", "CSpca", "CSsmfa" or "CSzhang")
...	Additional parameters for analysis

Value

An object of the S4 Class CSresult-class.

Examples

```
data("dataSIM",package="CSFA")
Mat1 <- dataSIM[,c(1:6)]
Mat2 <- dataSIM[,-c(1:6)]

MFA_analysis <- CSanalysis(Mat1,Mat2,"CSmfa")
FABIA_analysis <- CSanalysis(Mat1,Mat2,"CSfabia")
ZHANG_analysis <- CSanalysis(Mat1,Mat2,"CSzhang")
```

CSanalysis,matrix,matrix,CSfabia-method

"CSfabia"

Description

Doing interactive CS analysis with FABIA (Factor Analysis for Bicluster Acquisition). One or multiple query compounds are possible in this analysis.

Usage

```
## S4 method for signature 'matrix,matrix,CSfabia'
CSanalysis(querMat, refMat,
  type = "CSfabia", p = 13, alpha = 0.01, cyc = 500, spl = 0,
  spz = 0.5, non_negative = 0, random = 1, center = 2, norm = 1,
  scale = 0, lap = 1, nL = 0, lL = 0, bL = 0, which = c(2, 3, 4, 5),
  component.plot = NULL, CSrank.queryplot = FALSE, column.interest = NULL,
  row.interest = NULL, profile.type = "gene", color.columns = NULL,
  gene.highlight = NULL, gene.thresP = 1, gene.thresN = -1,
  thresP.col = "blue", thresN.col = "red", grouploadings.labels = NULL,
  grouploadings.cutoff = NULL, legend.names = NULL, legend.cols = NULL,
  legend.pos = "topright", labels = TRUE, result.available = NULL,
  result.available.update = FALSE, plot.type = "device",
  basefilename = NULL)
```

Arguments

querMat	Query matrix (Rows = genes and columns = compounds)
refMat	Reference matrix
type	"CSfabia"
p	*Fabia Parameter:* number of hidden factors = number of biclusters; default = 13
alpha	*Fabia Parameter:* sparseness loadings (0 - 1.0); default = 0.01
cyc	*Fabia Parameter:* number of iterations; default = 500
spl	*Fabia Parameter:* sparseness prior loadings (0 - 2.0); default = 0 (Laplace)

`spz` *Fabia Parameter:* sparseness factors (0.5 - 2.0); default = 0.5 (Laplace)

`non_negative` *Fabia Parameter:* Non-negative factors and loadings if non_negative > 0; default = 0

`random` *Fabia Parameter:* <=0: by SVD, >0: random initialization of loadings in [-random,random]; default = 1.0

`center` *Fabia Parameter:* data centering: 1 (mean), 2 (median), > 2 (mode), 0 (no); default = 2

`norm` *Fabia Parameter:* data normalization: 1 (0.75-0.25 quantile), >1 (var=1), 0 (no); default = 1

`scale` *Fabia Parameter:* loading vectors are scaled in each iteration to the given variance. 0.0 indicates non scaling; default = 0.0

`lap` *Fabia Parameter:* minimal value of the variational parameter; default = 1.0

`nL` *Fabia Parameter:* maximal number of biclusters at which a row element can participate; default = 0 (no limit)

`lL` *Fabia Parameter:* maximal number of row elements per bicluster; default = 0 (no limit)

`bL` *Fabia Parameter:* cycle at which the nL or lL maximum starts; default = 0 (start at the beginning)

`which` Choose one or more plots to draw:

1. Information Content for Bicluster (Only available for "CSfabia")
2. Loadings for query compounds
3. Loadings for Component (Factor/Bicluster) `component.plot`
4. Gene Scores for Component (Factor/Bicluster) `component.Plot`
5. Connectivity Ranking Scores for Component `component.plot`
6. Component `component.plot` VS Other Component : Loadings & Genes
7. Profile plot (see `profile.type`)
8. Group Loadings Plots for all components (see `grouploadings.labels`).

`component.plot` Which components (Factor/Bicluster) should be investigated? Can be a vector of multiple (e.g. `c(1,3,5)`). If `NULL`, you can choose components of interest interactively from query loadings plot.

`CSrank.queryplot`
Logical value deciding if the CS Rank Scores (`which=5`) should also be plotted per query (instead of only the weighted mean).

`column.interest`
Numeric vector of indices of reference columns which should be in the profiles plots (`which=7`). If `NULL`, you can interactively select genes on the Compound Loadings plot (`which=3`).

`row.interest` Numeric vector of gene indices to be plotted in gene profiles plot (`which=7`, `profile.type="gene"`). If `NULL`, you can interactively select them in the gene scores plot (`which=4`).

`profile.type` Type of `which=7` plot:

- "gene": Gene profiles plot of selected genes in row.interest with the query compounds and those selected in column.interest ordered first on the x axis. The other compounds are ordered in decreasing CScore.
- "cmpd": Compound profiles plot of query and selected compounds (column.interest) and only those genes on the x-axis which beat the thresholds (gene.thresP, gene.thresN)

color.columns Vector of colors for the query and reference columns (compounds). If NULL, blue will be used for query and black for reference. Use this option to highlight query columns and reference columns of interest.

gene.highlight Single numeric vector or list of maximum 5 numeric vectors. This highlights gene of interest in gene scores plot (which=4) up to 5 different colors. (e.g. You can use this to highlight genes you know to be differentially expressed)

gene.thresP Threshold for genes with a high score (which=4).

gene.thresN Threshold for genes with a low score (which=4).

thresP.col Color of genes above gene.thresP.

thresN.col Color of genes below gene.thresN.

grouploadings.labels

This parameter used for the Group Loadings Plots (which=8). In general this plot will contain the loadings of all factors, grouped and colored by the labels given in this parameter. Two types of plot can be created:

1. If grouploadings.labels!=NULL:
   Provide a vector for all samples (query + ref) containing labels on which the plot will be based on.
2. If grouploadings.labels=NULL:
   If no labels are provided when choosing which=8, automatic labels ("Top Samples of Component 1, 2....") will be created. These labels are given to the top grouploadings.cutoff number of samples based on the absolute values of the loadings.

Plot which=8 can be used to check 2 different situations. The first plot checks if your provided labels coincide with the discovered structures in the analysis. The second plot aims to find new interesting structures (of samples) which strongly appear in one or multiple components. A subsequent step could be to take some strong samples/compounds of these compounds and use them as a new query set in a new CS analysis to check its validity or to find newly connected compounds.

Please note that even when group.loadings.labels!=NULL, that the labels based on the absolute loadings of all the factors (the top grouploadings.cutoff) will always be generated and saved in samplefactorlabels in the extra slot of the CSresult object. This can then later be used for the CSlabelscompare function to compare them with your true labels.

grouploadings.cutoff

Parameter used in plot which=8. An integer for the number of cut-offs. See grouploadings.labels=NULL for more information. If this parameter is not provided, it will be automatically set to 10% of the total number of loadings.

legend.names Option to draw a legend of for example colored columns in Compound Loadings plot (which=3). If NULL, only "References" will be in the legend.

legend.cols	Colors to be used in legends. If NULL, only blue for "Queries is used".
legend.pos	Position of the legend in all requested plots, can be "topright", "topleft", "bottomleft", "bottomright", "bottom", "top", "left", "right", "center".
labels	Boolean value (default=TRUE) to use row and/or column text labels in the score plots (which=c(3,4,5,6)).
result.available	You can a previously returned object by CSanalysis in order to only draw graphs, not recompute the scores.
result.available.update	Logical value. If TRUE, the CS and GS will be overwritten depending on the new component.plot choice. This would also delete the p-values if permutation.object was available.
plot.type	How should the plots be outputted? "pdf" to save them in pdf files, device to draw them in a graphics device (default), sweave to use them in a sweave or knitr file.
basefilename	Directory including filename of the graphs if saved in pdf files

Value

An object of the S4 Class CSresult-class.

CSanalysis,matrix,matrix,CSmfa-method
"CSmfa"

Description

Doing interactive CS analysis with MFA (Multiple Factor Analysis). Should use multiple queries for this analysis. Uses the MFA function.

Usage

```
## S4 method for signature 'matrix,matrix,CSmfa'
CSanalysis(querMat, refMat, type = "CSmfa",
  ncp = 5, weight.col.mfa = NULL, row.w = NULL, mfa.type = "s",
  which = c(2, 3, 4, 5), component.plot = NULL, CSrank.queryplot = FALSE,
  column.interest = NULL, row.interest = NULL, profile.type = "gene",
  color.columns = NULL, gene.highlight = NULL, gene.thresP = 1,
  gene.thresN = -1, thresP.col = "blue", thresN.col = "red",
  grouploadings.labels = NULL, grouploadings.cutoff = NULL,
  legend.names = NULL, legend.cols = NULL, legend.pos = "topright",
  labels = TRUE, result.available = NULL, result.available.update = FALSE,
  plot.type = "device", basefilename = NULL)
```

Arguments

querMat	Query matrix (Rows = genes and columns = compounds)
refMat	Reference matrix
type	"CSmfa"
ncp	*MFA Parameter:* Number of dimensions kept in the results (by default 5).
weight.col.mfa	*MFA Parameter:* Vector of weights, useful for HMFA method (by default, NULL and an MFA is performed).
row.w	*MFA Parameter:* An optional row weights (by default, a vector of 1 for uniform row weights).
mfa.type	*MFA Parameter:* The type of column variables (compounds) in both the Query and Reference matrix. "c" or "s" (= default) for quantitative variables (the difference is that for "s" variables are scaled to unit variance), "n" for categorical variables and "f" for frequencies (from a contingency tables)
which	Choose one or more plots to draw: 1. Information Content for Bicluster (Only available for "CSfabia") 2. Loadings for query compounds 3. Loadings for Component (Factor/Bicluster) component.plot 4. Gene Scores for Component (Factor/Bicluster) component.Plot 5. Connectivity Ranking Scores for Component component.plot 6. Component component.plot VS Other Component : Loadings & Genes 7. Profile plot (see profile.type) 8. Group Loadings Plots for all components (see grouploadings.labels).
component.plot	Which components (Factor/Bicluster) should be investigated? Can be a vector of multiple (e.g. c(1,3,5)). If NULL, you can choose components of interest interactively from query loadings plot.
CSrank.queryplot	Logical value deciding if the CS Rank Scores (which=5) should also be plotted per query (instead of only the weighted mean).
column.interest	Numeric vector of indices of reference columns which should be in the profiles plots (which=7). If NULL, you can interactively select genes on the Compound Loadings plot (which=3).
row.interest	Numeric vector of gene indices to be plotted in gene profiles plot (which=7, profile.type="gene"). If NULL, you can interactively select them in the gene scores plot (which=4).
profile.type	Type of which=7 plot: • "gene": Gene profiles plot of selected genes in row.interest with the query compounds and those selected in column.interest ordered first on the x axis. The other compounds are ordered in decreasing CScore. • "cmpd": Compound profiles plot of query and selected compounds (column.interest) and only those genes on the x-axis which beat the thresholds (gene.thresP, gene.thresN)

`color.columns` Vector of colors for the query and reference columns (compounds). If `NULL`, blue will be used for query and black for reference. Use this option to highlight query columns and reference columns of interest.

`gene.highlight` Single numeric vector or list of maximum 5 numeric vectors. This highlights gene of interest in gene scores plot (`which=4`) up to 5 different colors. (e.g. You can use this to highlight genes you know to be differentially expressed)

`gene.thresP` Threshold for genes with a high score (`which=4`).

`gene.thresN` Threshold for genes with a low score (`which=4`).

`thresP.col` Color of genes above `gene.thresP`.

`thresN.col` Color of genes below `gene.thresN`.

`grouploadings.labels`

This parameter used for the Group Loadings Plots (`which=8`). In general this plot will contain the loadings of all factors, grouped and colored by the labels given in this parameter. Two types of plot can be created:

1. If `grouploadings.labels!=NULL`:
   Provide a vector for all samples (query + ref) containing labels on which the plot will be based on.
2. If `grouploadings.labels=NULL`:
   If no labels are provided when choosing `which=8`, automatic labels ("Top Samples of Component 1, 2....") will be created. These labels are given to the top `grouploadings.cutoff` number of samples based on the absolute values of the loadings.

Plot `which=8` can be used to check 2 different situations. The first plot checks if your provided labels coincide with the discovered structures in the analysis. The second plot aims to find new interesting structures (of samples) which strongly appear in one or multiple components. A subsequent step could be to take some strong samples/compounds of these compounds and use them as a new query set in a new CS analysis to check its validity or to find newly connected compounds.

Please note that even when `group.loadings.labels!=NULL`, that the labels based on the absolute loadings of all the factors (the top `grouploadings.cutoff`) will always be generated and saved in `samplefactorlabels` in the `extra` slot of the `CSresult` object. This can then later be used for the `CSlabelscompare` function to compare them with your true labels.

`grouploadings.cutoff`

Parameter used in plot `which=8`. An integer for the number of cut-offs. See `grouploadings.labels=NULL` for more information. If this parameter is not provided, it will be automatically set to 10% of the total number of loadings.

`legend.names` Option to draw a legend of for example colored columns in Compound Loadings plot (`which=3`). If `NULL`, only "References" will be in the legend.

`legend.cols` Colors to be used in legends. If `NULL`, only blue for "Queries is used".

`legend.pos` Position of the legend in all requested plots, can be `"topright"`, `"topleft"`, `"bottomleft"`, `"bottomright"`, `"bottom"`, `"top"`, `"left"`, `"right"`, `"center"`.

`labels` Boolean value (default=TRUE) to use row and/or column text labels in the score plots (`which=c(3,4,5,6)`).

`result.available`
: You can a previously returned object by `CSanalysis` in order to only draw graphs, not recompute the scores.

`result.available.update`
: Logical value. If `TRUE`, the CS and GS will be overwritten depending on the new `component.plot` choice. This would also delete the p-values if `permutation.object` was available.

`plot.type`
: How should the plots be outputted? `"pdf"` to save them in pdf files, `device` to draw them in a graphics device (default), `sweave` to use them in a sweave or knitr file.

`basefilename`
: Directory including filename of the graphs if saved in pdf files

Value

An object of the S4 Class `CSresult-class`.

`CSanalysis,matrix,matrix,CSpca-method`
"CSpca"

Description

Doing interactive CS analysis with PCA (Principal Component Analysis). This analysis is meant for 1 query signature. Uses the `PCA` function.

Usage

```
## S4 method for signature 'matrix,matrix,CSpca'
CSanalysis(querMat, refMat, type = "CSpca",
  ncp = 5, scale.unit = TRUE, row.w = NULL, col.w = NULL, which = c(2,
  3, 4, 5), component.plot = NULL, CSrank.queryplot = FALSE,
  column.interest = NULL, row.interest = NULL, profile.type = "gene",
  color.columns = NULL, gene.highlight = NULL, gene.thresP = 1,
  gene.thresN = -1, thresP.col = "blue", thresN.col = "red",
  grouploadings.labels = NULL, grouploadings.cutoff = NULL,
  legend.names = NULL, legend.cols = NULL, legend.pos = "topright",
  labels = TRUE, result.available = NULL, result.available.update = FALSE,
  plot.type = "device", basefilename = NULL)
```

Arguments

`querMat`
: Query matrix (Rows = genes and columns = compounds)

`refMat`
: Reference matrix

`type`
: `"CSpca"`

`ncp`
: *PCA Parameter:* Number of dimensions kept in the results (by default 5).

`scale.unit` *PCA Parameter:* A boolean, if TRUE (value set by default) then data are scaled to unit variance.

`row.w` *PCA Parameter:* An optional row weights (by default, a vector of 1 for uniform row weights).

`col.w` *PCA Parameter:* An optional column weights (by default, uniform column weights).

`which` Choose one or more plots to draw:

1. Information Content for Bicluster (Only available for "CSfabia")
2. Loadings for query compounds
3. Loadings for Component (Factor/Bicluster) `component.plot`
4. Gene Scores for Component (Factor/Bicluster) `component.Plot`
5. Connectivity Ranking Scores for Component `component.plot`
6. Component `component.plot` VS Other Component : Loadings & Genes
7. Profile plot (see `profile.type`)
8. Group Loadings Plots for all components (see `grouploadings.labels`).

`component.plot` Which components (Factor/Bicluster) should be investigated? Can be a vector of multiple (e.g. `c(1,3,5)`). If `NULL`, you can choose components of interest interactively from query loadings plot.

`CSrank.queryplot`
Logical value deciding if the CS Rank Scores (`which=5`) should also be plotted per query (instead of only the weighted mean).

`column.interest`
Numeric vector of indices of reference columns which should be in the profiles plots (`which=7`). If `NULL`, you can interactively select genes on the Compound Loadings plot (`which=3`).

`row.interest` Numeric vector of gene indices to be plotted in gene profiles plot (`which=7`, `profile.type="gene"`). If `NULL`, you can interactively select them in the gene scores plot (`which=4`).

`profile.type` Type of `which=7` plot:

- `"gene"`: Gene profiles plot of selected genes in `row.interest` with the query compounds and those selected in `column.interest` ordered first on the x axis. The other compounds are ordered in decreasing CScore.
- `"cmpd"`: Compound profiles plot of query and selected compounds (`column.interest`) and only those genes on the x-axis which beat the thresholds (`gene.thresP`, `gene.thresN`)

`color.columns` Vector of colors for the query and reference columns (compounds). If `NULL`, blue will be used for query and black for reference. Use this option to highlight query columns and reference columns of interest.

`gene.highlight` Single numeric vector or list of maximum 5 numeric vectors. This highlights gene of interest in gene scores plot (`which=4`) up to 5 different colors. (e.g. You can use this to highlight genes you know to be differentially expressed)

`gene.thresP` Threshold for genes with a high score (`which=4`).

`gene.thresN` Threshold for genes with a low score (`which=4`).

`thresP.col` Color of genes above `gene.thresP`.

`thresN.col` Color of genes below `gene.thresN`.

`grouploadings.labels`
This parameter used for the Group Loadings Plots (`which=8`). In general this plot will contain the loadings of all factors, grouped and colored by the labels given in this parameter. Two types of plot can be created:

1. If `grouploadings.labels!=NULL`:
   Provide a vector for all samples (query + ref) containing labels on which the plot will be based on.
2. If `grouploadings.labels=NULL`:
   If no labels are provided when choosing `which=8`, automatic labels ("Top Samples of Component 1, 2....") will be created. These labels are given to the top `grouploadings.cutoff` number of samples based on the absolute values of the loadings.

Plot `which=8` can be used to check 2 different situations. The first plot checks if your provided labels coincide with the discovered structures in the analysis. The second plot aims to find new interesting structures (of samples) which strongly appear in one or multiple components. A subsequent step could be to take some strong samples/compounds of these compounds and use them as a new query set in a new CS analysis to check its validity or to find newly connected compounds.

Please note that even when `group.loadings.labels!=NULL`, that the labels based on the absolute loadings of all the factors (the top `grouploadings.cutoff`) will always be generated and saved in `samplefactorlabels` in the `extra` slot of the `CSresult` object. This can then later be used for the `CSlabelscompare` function to compare them with your true labels.

`grouploadings.cutoff`
Parameter used in plot `which=8`. An integer for the number of cut-offs. See `grouploadings.labels=NULL` for more information. If this parameter is not provided, it will be automatically set to 10% of the total number of loadings.

`legend.names` Option to draw a legend of for example colored columns in Compound Loadings plot (`which=3`). If `NULL`, only "References" will be in the legend.

`legend.cols` Colors to be used in legends. If `NULL`, only blue for "Queries is used".

`legend.pos` Position of the legend in all requested plots, can be `"topright"`, `"topleft"`, `"bottomleft"`, `"bottomright"`, `"bottom"`, `"top"`, `"left"`, `"right"`, `"center"`.

`labels` Boolean value (default=TRUE) to use row and/or column text labels in the score plots (`which=c(3,4,5,6)`).

`result.available`
You can a previously returned object by `CSanalysis` in order to only draw graphs, not recompute the scores.

`result.available.update`
Logical value. If `TRUE`, the CS and GS will be overwritten depending on the new `component.plot` choice. This would also delete the p-values if `permutation.object` was available.

`plot.type` How should the plots be outputted? `"pdf"` to save them in pdf files, `device` to draw them in a graphics device (default), `sweave` to use them in a sweave or knitr file.

basefilename Directory including filename of the graphs if saved in pdf files

Value

An object of the S4 Class CSresult-class.

CSanalysis,matrix,matrix,CSsmfa-method
"CSsmfa"

Description

Doing interactive CS analysis with sMFA (Sparse Multiple Factor Analysis). Should use multiple queries for this analysis. Either spca or arrayspc is used.

Usage

```
## S4 method for signature 'matrix,matrix,CSsmfa'
CSanalysis(querMat, refMat, type = "Csmfa",
  K = 15, para, lambda = 1e-06, sparse.dim = 2, sparse = "penalty",
  max.iter = 200, eps.conv = 0.001, which = c(2, 3, 4, 5),
  component.plot = NULL, CSrank.queryplot = FALSE, column.interest = NULL,
  row.interest = NULL, profile.type = "gene", color.columns = NULL,
  gene.highlight = NULL, gene.thresP = 1, gene.thresN = -1,
  thresP.col = "blue", thresN.col = "red", grouploadings.labels = NULL,
  grouploadings.cutoff = NULL, legend.names = NULL, legend.cols = NULL,
  legend.pos = "topright", labels = TRUE, result.available = NULL,
  result.available.update = FALSE, plot.type = "device",
  basefilename = NULL)
```

Arguments

querMat	Query matrix (Rows = genes and columns = compounds)
refMat	Reference matrix
type	"CSsmfa"
K	*sMFA Parameters:* Number of components.
para	*sMFA Parameters:* A vector of length K. All elements should be positive. If sparse="varnum", the elements integers.
lambda	*sMFA Parameters:* Quadratic penalty parameter. Default value is 1e-6. If the target dimension of the sparsness is higher than the other dimension (p > n), it is advised to put lambda to Inf which uses the arrayspc algorithm optimized for this case. For the other case, p < n, a zero or positive lambda is sufficient and will utilize the normal spca algorithm.
sparse.dim	*sMFA Parameters:* Which dimension should be sparse? 1: Rows, 2: Columns (default) (Note: For Connectivity Scores it is advised to apply sparsity on the compounds/columns)

`sparse` *sMFA Parameters (*`lambda < Inf` *only):* If `sparse="penalty"`, `para` is a vector of 1-norm penalty parameters. If `sparse="varnum"`, `para` defines the number of sparse loadings to be obtained.

`max.iter` *sMFA Parameters:* Maximum number of iterations.

`eps.conv` *sMFA Parameters:* Convergence criterion.

`which` Choose one or more plots to draw:

1. Information Content for Bicluster (Only available for "CSfabia")
2. Loadings for query compounds
3. Loadings for Component (Factor/Bicluster) `component.plot`
4. Gene Scores for Component (Factor/Bicluster) `component.Plot`
5. Connectivity Ranking Scores for Component `component.plot`
6. Component `component.plot` VS Other Component : Loadings & Genes
7. Profile plot (see `profile.type`)
8. Group Loadings Plots for all components (see `grouploadings.labels`).

`component.plot` Which components (Factor/Bicluster) should be investigated? Can be a vector of multiple (e.g. `c(1,3,5)`). If `NULL`, you can choose components of interest interactively from query loadings plot.

`CSrank.queryplot`
Logical value deciding if the CS Rank Scores (`which=5`) should also be plotted per query (instead of only the weighted mean).

`column.interest`
Numeric vector of indices of reference columns which should be in the profiles plots (`which=7`). If `NULL`, you can interactively select genes on the Compound Loadings plot (`which=3`).

`row.interest` Numeric vector of gene indices to be plotted in gene profiles plot (`which=7`, `profile.type="gene"`). If `NULL`, you can interactively select them in the gene scores plot (`which=4`).

`profile.type` Type of `which=7` plot:

- `"gene"`: Gene profiles plot of selected genes in `row.interest` with the query compounds and those selected in `column.interest` ordered first on the x axis. The other compounds are ordered in decreasing CScore.
- `"cmpd"`: Compound profiles plot of query and selected compounds (`column.interest`) and only those genes on the x-axis which beat the thresholds (`gene.thresP`, `gene.thresN`)

`color.columns` Vector of colors for the query and reference columns (compounds). If `NULL`, blue will be used for query and black for reference. Use this option to highlight query columns and reference columns of interest.

`gene.highlight` Single numeric vector or list of maximum 5 numeric vectors. This highlights gene of interest in gene scores plot (`which=4`) up to 5 different colors. (e.g. You can use this to highlight genes you know to be differentially expressed)

`gene.thresP` Threshold for genes with a high score (`which=4`).

`gene.thresN` Threshold for genes with a low score (`which=4`).

`thresP.col` Color of genes above `gene.thresP`.

`thresN.col` Color of genes below `gene.thresN`.

`grouploadings.labels`

This parameter used for the Group Loadings Plots (`which=8`). In general this plot will contain the loadings of all factors, grouped and colored by the labels given in this parameter. Two types of plot can be created:

1. If `grouploadings.labels!=NULL`:
   Provide a vector for all samples (query + ref) containing labels on which the plot will be based on.
2. If `grouploadings.labels=NULL`:
   If no labels are provided when choosing `which=8`, automatic labels ("Top Samples of Component 1, 2....") will be created. These labels are given to the top `grouploadings.cutoff` number of samples based on the absolute values of the loadings.

Plot `which=8` can be used to check 2 different situations. The first plot checks if your provided labels coincide with the discovered structures in the analysis. The second plot aims to find new interesting structures (of samples) which strongly appear in one or multiple components. A subsequent step could be to take some strong samples/compounds of these compounds and use them as a new query set in a new CS analysis to check its validity or to find newly connected compounds.

Please note that even when `group.loadings.labels!=NULL`, that the labels based on the absolute loadings of all the factors (the top `grouploadings.cutoff`) will always be generated and saved in `samplefactorlabels` in the `extra` slot of the `CSresult` object. This can then later be used for the `CSlabelscompare` function to compare them with your true labels.

`grouploadings.cutoff`

Parameter used in plot `which=8`. An integer for the number of cut-offs. See `grouploadings.labels=NULL` for more information. If this parameter is not provided, it will be automatically set to 10% of the total number of loadings.

`legend.names` Option to draw a legend of for example colored columns in Compound Loadings plot (`which=3`). If `NULL`, only "References" will be in the legend.

`legend.cols` Colors to be used in legends. If `NULL`, only blue for "Queries is used".

`legend.pos` Position of the legend in all requested plots, can be `"topright"`, `"topleft"`, `"bottomleft"`, `"bottomright"`, `"bottom"`, `"top"`, `"left"`, `"right"`, `"center"`.

`labels` Boolean value (default=TRUE) to use row and/or column text labels in the score plots (`which=c(3,4,5,6)`).

`result.available`

You can a previously returned object by `CSanalysis` in order to only draw graphs, not recompute the scores.

`result.available.update`

Logical value. If `TRUE`, the CS and GS will be overwritten depending on the new `component.plot` choice. This would also delete the p-values if `permutation.object` was available.

`plot.type` How should the plots be outputted? `"pdf"` to save them in pdf files, `device` to draw them in a graphics device (default), `sweave` to use them in a sweave or knitr file.

`basefilename` Directory including filename of the graphs if saved in pdf files

Value

An object of the S4 Class CSresult-class.

CSanalysis,matrix,matrix,CSzhang-method
"CSzhang"

Description

Compute the Connectivity Scores by Zhang and Gant (2008). One or multiple query compounds are possible in this analysis. In the case of multiple query compounds, the average of each pairwise score is taken.

Usage

```
## S4 method for signature 'matrix,matrix,CSzhang'
CSanalysis(querMat, refMat,
  type = "CSzhang", nquery = NULL, nref = NULL, ntop.scores = 20,
  which = c(1), color.ref = NULL, legend.names = NULL,
  legend.cols = NULL, legend.pos = "topright", labels = TRUE,
  result.available = NULL, result.available.update = FALSE,
  plot.type = "device", basefilename = NULL)
```

Arguments

querMat	Query matrix (Rows = genes and columns = compounds)
refMat	Reference matrix
type	"CSzhang"
nquery	*Zhang Parameter:* Number of top up- and downregulated genes in query signature. If NULL, all rows (genes) are used.
nref	*Zhang Parameter:* Number of top up- and downregulated genes in reference signature. If NULL, all rows (genes) are used. (Note that $nquery >= nref$)
ntop.scores	*Zhang Parameter:* Number of top positive and negative CS to be reported first.
which	Choose plot to draw. 1. Zhang and Gant Scores Plot
color.ref	Vector of colors for the reference columns. You can use this option to highlight columns(compounds) of interest in the CS plot. (This does not include the query columns since they are not included in the CS plot.)
legend.names	Option to draw a legend (about the highlights in color.ref) in the CS plot. If NULL, no legend will be drawn.
legend.cols	Colors to be used for the legend.names.
legend.pos	Position of the legend in all requested plots, can be "topright", "topleft", "bottomleft", "bottomright", "bottom", "top", "left", "right", "center".

`labels` Boolean value (default=TRUE) to use column labels inside the ZG plot.

`result.available`
You can a previously returned object by `CSanalysis` in order to only draw graphs, not recompute the scores. If this object also contains the permutation object, in the score plot the values with a (adjusted) pvalue smaller than 0.05 will be colored purple.

`result.available.update`
Logical value. If `TRUE`, the CS and GS will be overwritten depending on the new `component.plot` choice. This would also delete the p-values if `permutation.object` was available.

`plot.type` How should the plots be outputted? `"pdf"` to save them in pdf files, `device` to draw them in a graphics device (default), `sweave` to use them in a sweave or knitr file.

`basefilename` Directory including filename of the graphs if saved in pdf files

Value

An object of the S4 Class `CSresult-class`. The CS slot will also contain the top positive and negative scores as well as the top p-values. The GS slot will be empty for Zhang and Gant.

`CScluster` *CScluster*

Description

Apply the Connectivity Scores to a *K* clustering result. More information can be found in the Details section below.

Usage

```
CScluster(data, clusterlabels, type = "CSmfa", WithinABS = TRUE,
  BetweenABS = TRUE, FactorABS = FALSE, verbose = FALSE, Within = NULL,
  Between = NULL, WithinSave = FALSE, BetweenSave = TRUE, ...)
```

Arguments

`data` A gene expression matrix with the compounds in the columns.

`clusterlabels` A vector of integers that represents the cluster grouping of the columns (compounds) in `data`. The labels should be integers starting from 1 to the total number of clusters. (e.g. the output of `cutree`)

`type` Type of CS anaylsis (default=`"CSmfa"`):

- `"CSmfa"` (MFA or PCA)
- `"CSsmfa"` (Sparse MFA or Sparse PCA)
- `"CSfabia"` (Fabia)
- `"CSzhang"` (Zhang and Gant)

In the first two options, either MFA or PCA is used depending on the cluster size. If the query set only contains a single compound, the latter is used. Also note that if a cluster only contains a single compound, no *Within-CS* can be computed.

`WithinABS` Boolean value to take the mean of the absolute values in the final step of the *Within-Cluster CS* (default=`TRUE`).

`BetweenABS` Boolean value to take the mean of the absolute values in the final step of the *Between-Cluster CS* (default=`TRUE`).

`FactorABS` Boolean value to take the absolute value of the query loadings when determining the best factor (= factor with highest query loadings) in a `CSanalysis` application (default=`FALSE`). This option might be helpful if the 'best factor' contains large positive and negative query loading which would average to zero.

`verbose` Boolean value to output warnings and information about which factor is chosen in a CS analysis (if applicable).

`Within` A vector for which cluster numbers the *Within-Cluster CS* should be computed. By default (=`NULL`) all within-cluster scores are computed, but this might not be feasible for larger data in which a single `CSanalysis` run might already take a sufficient amount of computation time.

`Between` A vector fir which cluster numbers the *Beween-Cluster CS* (with the cluster as a query set) should be computed. By default (=`NULL`) all between-cluster scores are computed, but this might not be feasible for larger data in which a single `CSanalysis` run might already take a sufficient amount of computation time.

`WithinSave` Boolean value to save the `Within` object in the `Save` slot of the returned list (default=`FALSE`).

`BetweenSave` Boolean value to save the `Between` object in the `Save` slot of the returned list (default=`TRUE`).

`...` Additional parameters given to `CSanalysis` specific to a certain `type` of CS analysis.

Details

After applying cluster analysis on the additional data matrix, *K* clusters are obtained. Each cluster will be seen as a potential query set (for `CSanalysis`) for which 2 connectivity score metrics can be computed, the *Within-Cluster CS* and the *Between-Cluster CS*.

Within-Cluster CS
This metric will answer the question if the *k*th cluster is connected on a gene expression level (in addition to the samples being similar based on the other data source). The Within-Cluster CS for a cluster is computed as following:

1. Repeatedly for the *i*th sample in the *k*th cluster, apply CSMFA with:
   - *Query Set*: All cluster samples **excluding** the *i*th sample.
   - *Reference*: All samples including the *i*th sample of the *k*th cluster.
   - Retrieve the CS of the *i*th sample in the cluster.
2. The Within-Cluster CS for cluster *k* is now defined as the average of all retrieved CS.

The concept of this metric is to investigate the connectivity for each compound with the cluster. The average of the 'leave-one-out' connectivity scores, the Within-Cluster CS, gives an indication of the gene expression connectivity of this cluster. A high Within-Cluster CS implies that the cluster is both similar on the external data source and on the gene expression level. A low score indicates that the cluster does not share a similar latent gene profile structure.

Between-Cluster CS
In this stage of the analysis, we focus on the *l*th cluster and use all compounds in this cluster as the query set. A CSMFA is performed in which all other clusters are the reference set. Next, the connectivity scores are calculated for all reference compounds and averaged over the clusters (=the between connectivity score). A high Between-Cluster CS between the *l*th and *j*th clusters implies that, while the two clusters are not similar based on the other data source, they do share a latent structure when considering the gene expression data.

Value

A list object with components:

- `CSmatrix`: A $K \times K$ matrix containing the Within scores on the diagonal and the Between scores elsewhere with the rows being the query set clusters (e.g. m_{13} = Between CS between cluster 1 (as query set) and cluster 3).
- `CSRankmatrix`: The same as `CSmatrix`, but with connectivity ranking scores (if applicable).
- `clusterlabels`: The provided `clusterlabels`
- `Save`: A list with components:
  - `Within`: A list with a component for each cluster *k* that contains:
    - `LeaveOneOutCS`: Each leave-one-out connectivity score for cluster *k*.
    - `LeaveOneOutCSRank`: Each leave-one-out connectivity ranking score for cluster *k* (if applicable).
    - `factorselect`: A vector containing which factors/BCs were selected in each leave-one-out CS analysis (if applicable).
    - `CS`: A (columns (compounds) $\times$ size of cluster *k*) matrix that contains all the connectivity scores in a leave-one-out CS analysis for each left out compound.
    - `CSRank`: The same as `CS`, but with connectivity ranking scores (if applicable).
  - `Between`: List:
    - `DataBetweenCS`: A (columns (compounds) $\times$ clusters) matrix containing all compound connectivity scores for each query cluster set.
    - `DataBetweenCSRank`: The same as `DataBetweenCS`, but with connectivity ranking scores (if applicable).
    - `queryindex`: The column indices for each query set in all CS analyses.
    - `factorselect`: A vector containing which factors/BCs were selected in each CS analysis (if applicable).

Author(s)

Ewoud De Troyer

Examples

```
# Example Data Set
data("dataSIM",package="CSFA")
# Remove some no-connectivity compounds
nosignal <- sapply(colnames(dataSIM),FUN=function(x){grepl("c-",x)})
data <- dataSIM[,-which(nosignal)[1:250]]

# Toy example with random cluster assignment:
# Note: clusterlabels can be acquired through cutree(hclust(...))
clusterlabels <- sample(1:10,size=ncol(data),replace=TRUE)

result1 <- CScluster(data,clusterlabels,type="CSmfa")
result2 <- CScluster(data,clusterlabels,type="CSzhang")

result1$CSmatrix
result1$CSRankmatrix

result2$CSmatrix
```

`CScompare` *Compare CS Results.*

Description

After applying different CSanalysis on the same data, you can compare 2 different results of connectivity loadings, connectivity ranking scores and gene scores Unless the result came from a Zhang and Gant analysis, you choose from which component (factor, PC, bicluster) the scores should be derived. Further, for Zhang and Gant analysis, the "CRanking Scores" and "CLoadings" will be the same as the ZG Score as well as the p-values.

Usage

```
CScompare(CSresult1, CSresult2, component1.plot, component2.plot,
  threshold.pvalues = 0.05, which = c(1, 2, 3), color.columns = NULL,
  gene.thresP = NULL, gene.thresN = NULL, thresP.col = c("blue",
  "light blue"), thresN.col = c("red", "pink"), legend.names = NULL,
  legend.cols = NULL, legend.pos = "topright", labels = TRUE,
  plot.type = "device", basefilename = NULL)
```

Arguments

`CSresult1`	First result.
`CSresult2`	Second result.

component1.plot
: If you are using a non-Zhang&Gant result, specify the bicluster, factor or principal component which should be used to derive connectivity scores from for the *first* result.

component2.plot
: If you are using a non-Zhang&Gant result, specify the bicluster, factor or principal component which should be used to derive connectivity scores from for the *second* result.

threshold.pvalues
: If both CSresult1 and CSresult contain pvalues (and adjusted pvalues), this threshold will be used to compare the number of overlapping significant results.

which
: Choose one or both plots which should be created.
  1. CS Comparison Plot
  2. GS Comparison Plot
  3. CSRankScores (Normal CS for CSzhang) Comparison Plot
  4. CS p-values comparison plot (Raw & Adjusted).
  5. CRankScores p-values comparison plot (Raw & Adjusted).

color.columns
: Vector of colors for the query and reference columns (compounds). If NULL, blue will be used for query and black for reference. Use this option to highlight query columns and reference columns of interest.

gene.thresP
: Vector of length 2 containing the positive gene thresholds for CSresult1 and CSresult2. Genes above the threshold will be colored. (e.g. c(1,2))

gene.thresN
: Vector of length 2 containing the negative gene thresholds for CSresult1 and CSresult2. Genes below the threshold will be colored. (e.g. c(-1,-2))

thresP.col
: Vector of length 2 containing the colors for the high gene scores for CSresult1 and CSresult2 (e.g. c("blue","light blue")).

thresN.col
: Vector of length 2 containing the colors for the low gene scores for CSresult1 and CSresult2 (e.g. c("red","pink")).

legend.names
: Option to draw a legend (about the highlights in color.columns) in the CS plot. If NULL, only queries are in the legend.

legend.cols
: Colors to be used for the legend.names.

legend.pos
: The location of the legend: "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center".

labels
: Boolean value (default=TRUE) to use row and/or column text labels in the comparison score plots.

plot.type
: How should the plots be outputted? "pdf" to save them in pdf files, device to draw them in a graphics device (default), sweave to use them in a sweave or knitr file.

basefilename
: Directory including filename of the graphs if saved in pdf files

Value

A list object with 2 slots. In the first slot, Pearson and Spearman correlation between the results (CLoadings, Gene Scores, CRanking Scores, (adjusted) p-values) can be found. The second slot, if permutation was applied, contains a small comparison between the significant results based on threshold.pvalues.

Examples

```
data("dataSIM",package="CSFA")
Mat1 <- dataSIM[,c(1:6)]
Mat2 <- dataSIM[,-c(1:6)]

MFA_analysis <- CSanalysis(Mat1,Mat2,"CSmfa",component.plot=1)
ZHANG_analysis <- CSanalysis(Mat1,Mat2,"CSzhang")

CScompare(MFA_analysis,ZHANG_analysis,1)
```

CSFA *Computing connectivity scores with Factor Analysis methodology.*

Description

CSFA is a wrapper of multiple packages containing a factor analysis method. These methods are used to derive the the connectivity scores of reference gene signatures with one or multiple query signatures. CSFA will apply them, output the scores and immediately produce a number of meaningful plots interactively. The included methods are PCA and MFA from the `FactoMineR` package, FABIA from the `fabia` package and Sparse PCA/MFA from the `elasticnet` package. Further, CSFA also contains an implementation of the Zhang and Gant score.

References

Abdi, H. et al. (2013), "Multiple factor analysis: principal component analysis for multitable and multiblock data sets," *WIREs Comput Stat*, 1-31.

Hochreiter, S. et al., "FABIA: Factor Analysis for Bicluster acquisition," *Bioinformatics*, 26, 1520-1527.

Lamb, J. et al. (2006), "The Connectivity Map: Using Gene-Expression Signatures to Connect Small Molecules, Genes, and Disease," *Science*, 313, 1929-1934.

Zhang, S.-D. and Gant, T.W. (2008), "A simple and robust method for connecting small-molecule drugs using gene-expression signatures," *BMC Bioinformatics*, 9, 10.

Working Paper: De Troyer E., Shkedzy Z., Kasim A. and Perualila-Tan N.-J. (2018), Connectivity Mapping Using Multiple Factor Analysis

---

CSlabelscompare *Compare Automatic Factor Labels with Manual Provided Labels.*

Description

With this function you can compare the automatic created labels based of the absolute loadings in `CSanalysis` (`which=8`) with your own provided labels to investigate if there is relation between them.
See the `type` parameter which two plots can be created.
Note that the automatic created factor labels in `CSanalysis` denote which factors this loading has a high/low value and these can be regenerated (with different a different cut-off) by simply running `CSanalysis` again. Providing `result.available` will skip the analysis computation step and only regenerate the labels.

Usage

```
CSlabelscompare(CSresult, labels, type = "factors",
  basefilename = "CSanalysis", plot.type = "device")
```

Arguments

`CSresult`	Object of CSresult S4 Class.
`labels`	Provide a vector with labels. (Length should be the number of queries and references together)
`type`	• `type="factorlabels"`: A K number of plots will be created (K = number of components in the analysis). Each plot will have the loadings on the y-axis and the original automatic generated factor labels on the x-axis. The loadings are plotted for these factor labels (with jitter) and are colored according to the manual provided labels (`labels`) which is shown in the legend. The coloring also shows which loadings were in the query set. • `type="factors"`: A K number of plots will be created (K = number of components in the analysis). Each plot will have the loadings on the y-axis and factor labels on the x-axis. These factor labels are not exactly the generated labels, but simply "Factor 1", "Factor 2",..., "None" or "BC 1", "BC 2",..., "None". This means that should a loading be high/low in multiple factors, it will appear multiple times on this plot, namely for each corresponding factor. The loadings are plotted for these factor labels (with jitter) and are colored according to the manual provided labels (`labels`) which is shown in the legend. The coloring also shows which loadings were in the query set. **Note** that if none of the loadings is high/low in multiple factors, the two types of plots should be identical.
`basefilename`	Base of the filename when saving the graph as a pdf (`plot.type="pdf"`)
`plot.type`	How should the plots be outputted? `"pdf"` to save them in pdf files, `device` to draw them in a graphics device (default), `sweave` to use them in a sweave or knitr file.

Examples

```
data("dataSIM",package="CSFA")
```

```
Mat1 <- dataSIM[,c(1:6)]
Mat2 <- dataSIM[,-c(1:6)]

MFA_out <- CSanalysis(Mat1,Mat2,"CSmfa",component.plot=1,which=c())

labels <- rep("Noise",ncol(dataSIM))
labels[c(1:31,332:341)] <- "Signal"

CSlabelscompare(CSresult=MFA_out,labels=labels,type="factors")
CSlabelscompare(CSresult=MFA_out,labels=labels,type="factorlabels")
```

`CSpermute` *Permute CS results*

Description

Apply permutation on MFA or Zhang results to obtain p-values of 1 of the components. The function asks for a CSresult object which is returned by CSanalysis. The CSpermute function will return the same CSresult object with added information such as p-values. If asked, the CSpermute function will also draw a volcanoplot and/or histograms of the p-values. If you simply want to redraw these plots, simply use the returned CSresult object by CSpermute again in the CSpermute function. If the number of permutations was not changed, this will prevent the entire permutation analysis from being redone.

Usage

```
CSpermute(querMat, refMat, CSresult, B = 500, mfa.factor = NULL,
  method.adjust = "none", verbose = TRUE, which = c(1, 3),
  cmpd.hist = NULL, color.columns = NULL, labels = TRUE,
  plot.type = "device", basefilename = NULL, MultiCores = FALSE,
  MultiCores.number = detectCores(logical = FALSE), MultiCores.seed = NULL,
  save.permutation = TRUE)
```

Arguments

`querMat`	Query matrix (Rows = genes and columns = compounds).
`refMat`	Reference matrix
`CSresult`	A CSresult class object.
`B`	Number of permutations.
`mfa.factor`	If permuting a CSmfa result, mfa.factor will decide of which factor the p-values should be computed. If `NULL`, the factor chosen in CSanalysis will be chosen (the factor chosen in the CS slot of the CSresult). NOTE: If the mfa.factor is different from the factor in the CS slot, the CS slot will be overwritten with this new factor.
`method.adjust`	Correction method of multiplicity adjusted p-values: "none", "holm", "hochberg", "hommel", "bonferroni", "BH", "BY" or "fdr". (Raw p-values are also always provided)

`verbose` If TRUE, progression dots of the permutation analysis will be printed.

`which` Choose which plot to draw:

1. A volcano plot of the -log(p-values) versus the observed connection scores.
2. A histogram of the permuted connection scores under the null hypothesis for a specific compound. A vertical line(s) is added for the observed CS and its p-value. The `cmpd.hist` parameter determines which compounds are drawn like this.
3. Analog to `which=1`, but for CSRankScores.
4. Analog to `which=2`, but for CSRankScores.

`cmpd.hist` Reference index vector which decides which reference compounds are plotted for the histogram distribution under null hypothesis (`which=2`). If `NULL`, you can select which compounds you want interactively on the volcano plot.

`color.columns` Option to color the compounds on the volcano plot (`which=1`). Should be a vector of colors with the length of number of references.

`labels` Boolean value (default=TRUE) to use row and/or column text labels in the volcano plots (`which=c(1,3)`).

`plot.type` How should the plots be outputted? `"pdf"` to save them in pdf files, `device` to draw them in a graphics device (default), `sweave` to use them in a sweave or knitr file.

`basefilename` Directory including filename of the graphs if saved in pdf files

`MultiCores` Logical value parallelisation should be used for permutation. `FALSE` by default. (This option uses `clusterApplyFT` in order to provide load balancing and reproducible results with `MultiCores.seed`)

`MultiCores.number`
Number of cores to be used for `MultiCores=TRUE`. By default total number of physical cores.

`MultiCores.seed`
Seed to be used for `MultiCores=TRUE` using see (`clusterSetupRNG.FT`)

`save.permutation`
Logical value if the scores (CLoadings, CRankingScores, ZG Scores) of each permuted data set should be saved (default=`TRUE`). This information is necessary to recalculate the p-values for different components as well as for producing the histograms. However for larger data, disabling this option will reduce the size of the resulting `CSresult-class` object.

Details

IMPORTANT! For MFA, `CSpermute` should *only* be used to compute the p-values of the Component in which the structure (loadings) of the queries is the strongest. This because in each permutation the factor with the highest average query loadings will be chosen. The ability to compute p-values of other factors (in which the query set also increased loadings) will be added in a later release.

Value

Returns the same CSresult-class object with added p-values to the CS slot and added information to the permutation.object slot. This CSresult can be reused in CSpermute to redraw the plots without calculation.

Examples

```
data("dataSIM",package="CSFA")
Mat1 <- dataSIM[,c(1:6)]
Mat2 <- dataSIM[,-c(1:6)]

MFA_analysis <- CSanalysis(Mat1,Mat2,"CSmfa")
MFA_analysis <- CSpermute(Mat1,Mat2,MFA_analysis,B=200)
```

CSresult-class *An S4 class in which the results of the Connectivity Scores by Factor Analysis are stored.*

Description

An S4 class in which the results of the Connectivity Scores by Factor Analysis are stored.

Slots

`type` A character string containing the analysis type.

`CS` List of any number of lists (depending on how many components were selected) which contain the connectivity loadings and ranking scores for the reference (and query loadings). If permutation was applied, will also contain p-values.

`GS` Dataframe containing the gene scores.

`extra` List which contains `CSRank_Full` (contains all intermediate values while calculating the CS Ranking Score), `Object` (contains the complete original FA or Zhang result) and `samplefactorlabels` (contains thresholded labels based on the factor loadings, see plot `which=8`).

`permutation.object` Contains CS for permuted data (matrix) and a dataframe with the p-values (only for MFA and Zhang).

`call` List object containing the original call of `CSanalysis` as well as the parameters for the chosen method.

dataSIM *Simulated Microarray Data*

Description

A matrix containing some simulated example microarray data. The first 6 columns of this matrix make up the query matrix part.

Format

A matrix with 1000 rows and 341 columns.

Index