Package 'stringgaussnet'

July 22, 2015

Type Package

Title PPI and Gaussian Network Construction from Transcriptomic Analysis Results Integrating a Multilevel Factor

Version 1.1

Date 2015-07-22

Author Emmanuel Chaplais, Henri-Jean Garchon

Maintainer Emmanuel Chaplais <emmanuel.chaplais@inserm.fr>

Description

A toolbox for a construction of protein-protein interaction networks through the 'STRING' application programming interface, and an inference of Gaussian networks through 'SIMoNe' and 'WGCNA' approach, from DE genes analysis results and expression data. Additional functions are provided to import automatically networks into an active 'Cytoscape' session.

Imports

AnnotationDbi,GO.db,VennDiagram,simone,biomaRt,limma,pspearman,igraph,httr,RJSONIO,RCurl,org.Hs.eg.db,graphic

License GPL-3

Suggests knitr

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2015-07-22 12:18:19

R topics documented:

stringgaussnet-package *stringgaussnet*

Description

Genome-wide transcriptomic arrays are, from some years, a standard method to identify differentially expressed (DE) genes affected by an observed phenotype. Several statistical analysis methods are now well defined to generate those DE gene lists. The graph theory can be very useful to prioritize key DE genes, and consists in linking genes (nodes) by different interactions (edges). Gene network analyses have already given very interesting results in the literature. There are mainly two kinds of gene networks: semantic networks are based on already known interactions from literature, and gaussian networks are constructed by existing correlations in expression between genes. We propose stringgaussnet, an R package that allows to construct, easily and with much flexibility, those two kinds of networks after DE genes analysis.

Author(s)

- Emmanuel Chaplais <emmanuel.chaplais@inserm.fr>
- Henri-Jean Garchon

Examples

```
## Please note that for constructing STRINGNet objects, an internet connexion is necessary.
## Some lines are commented out for less computation time or Cytoscape dependency. But all are
##executable if all required conditions are filled (see package dependencies and suggests).

#data(SpADataExpression) # Import example expression data
#data(SpADEGenes) # Import example DE genes analysis results
#data(SpASamples) # Import example sample description
#SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes) # Create an object of class DEGeneExpr with
##expression data and DE genes analysis results. This object class is the basis for all functions
##in the package.

#StatusFactor<-SpASamples$status # We create a factor based on the status
#TimeFactor<-as.character(SpASamples$LPStime) # Create a factor based on LPS stimulation time
#names(StatusFactor)=names(TimeFactor)=SpASamples$chipnum # Attribute sample names to factors

#STRINGSpAData<-DEGeneExpr(t(SpADataExpression[30:60,]),SpADEGenes[30:60,]) # We subset the
##DEGeneExpr object for faster computation in the example
#SpASTRINGNet<-getSTRINGNet(STRINGSpAData) # Construct a STRING network through the API
## If you wish to add gene annotations (can take a while):
##SpASTRINGNet<-getSTRINGNet(STRINGSpAData,AddAnnotations=TRUE)
#print(SpASTRINGNet,5) # We print the STRINGNet object
#summary(SpASTRINGNet) # We summarize the STRINGNet object
## If you wish to export the STRINGNet object:
##export(SpASTRINGNet,"SpASTRINGNet",T)
#PPISpASTRINGNet<-selectInteractionTypes(SpASTRINGNet,c("coexpression","experimental","knowledge")
# ,0.9) # We select specific interaction sources, by filtering on confidence scores
```

```
#print(PPISpASTRINGNet,5)# We can see that the number of interactions decreases
#summary(PPISpASTRINGNet) # We can see that the minimum score is 0.9
## If you wish to export the PPI STRINGNet object:
##export(PPISpASTRINGNet,"PPISpASTRINGNet",T)

#shortPathSpANet<-getShortestPaths(PPISpASTRINGNet) # We compute shortest paths between initial
##nodes
#shortPathSpANet<-FilterEdges(shortPathSpANet,5) # We filter edges on the distance
#print(shortPathSpANet,5) # We print the ShortPathSTRINGNetobject
#summary(shortPathSpANet) # We summarize the ShortPathSTRINGNetobject
## If you wish to export the ShortPathSTRINGNetobject:
##export(shortPathSpANet,"shortPathSpANet",T)

#NodesForSIMoNe<-rownames(SpADEGenes)[1:17] # We select a reasonable number of genes for SIMoNe
##network inference
#GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])
## We create the DEGeneExpr object for the gaussian networks inference

## If you wish to have help for choosing parameters in SIMoNe network inference:
##pickSIMoNeParam(GaussianSpAData)
#GlobalSIMoNeNetNF<-getSIMoNeNet(GaussianSpAData) # We infer SIMoNe network with default parameters
## If you wish to add gene annotations (can take a while):
##GlobalSIMoNeNetNF<-getSIMoNeNet(GaussianSpAData,AddAnnotations=TRUE)
#GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNetNF,0.4) # We filter edges on absolute values of
##spearman's rho
#print(GlobalSIMoNeNet,5) # We print the SIMoNeNet object
#summary(GlobalSIMoNeNet) # We summarize the SIMoNeNet object
#plot(GlobalSIMoNeNet) # We plot the SIMoNeNet object
##dev.off() # We close the used device for plot
## We can plot the highest positive correlation between two DE genes:
##plot(GaussianSpAData$DataExpression[,"NUDT3"]~
## GaussianSpAData$DataExpression[,"P2RX1"],xlab="P2RX1",ylab="NUDT3")
## If you wish to export the SIMoNeNet object:
##export(GlobalSIMoNeNet,"GlobalSIMoNeNet",T)

## If you wish to have help for choosing parameters in WGCNAnetwork inference:
##pickWGCNAParam(GaussianSpAData)
#GlobalWGCNANet<-getWGCNANet(GaussianSpAData) # We infer WGCNA network with default parameters
## If you wish to add gene annotations (can take a while):
##GlobalWGCNANet<-getWGCNANet(GaussianSpAData,AddAnnotations=TRUE)
#print(GlobalWGCNANet,5) # We print the WGCNANet object
#summary(GlobalWGCNANet) # We summarize the WGCNANet object
#plot(GlobalWGCNANet) # We plot the WGCNANet object
##dev.off() # We close the used device for plot
## If you wish to export the WGCNANet object:
##export(GlobalWGCNANet,"GlobalWGCNANet",T)

## If you wish to compare SIMoNe and WGCNA results:
##compareGaussNetworks(GlobalSIMoNeNetNF,GlobalWGCNANet,c("SIMoNe","WGCNA"))

#StatusFactorSIMoNeNet<-FactorNetworks(GaussianSpAData,StatusFactor,"SIMoNe") # We infer different
##SIMoNe networks on different groups of samples (patients and controls)
## If you wish to add gene annotations (can take a while):
```

```
##StatusFactorSIMoNeNet<-FactorNetworks(GaussianSpADataStatusFactor,"SIMoNe",
## list(AddAnnotations=TRUE))
#StatusFactorSIMoNeNet<-FilterEdges(StatusFactorSIMoNeNet,0.4) # We filter edges on absolute
##values of spearman's rho
#plot(StatusFactorSIMoNeNet$Patient$Network) # We plot the network in patients
##dev.off() # We close the used device for plot
#plot(StatusFactorSIMoNeNet$Control$Network) # We plot the network in controls
##dev.off() # We close the used device for plot
## You can also use directly:
## par(mfrow=c(2,1))
## plot(StatusFactorSIMoNeNet,interactiveMode=F)
## If you wish to compare results between different level of factors infered by SIMoNe:
##compareFactorNetworks(StatusFactorSIMoNeNet)

## If you wish to infer different SIMoNe networks at different LPS stimulation times:
##TimeFactorSIMoNeNet<-FactorNetworks(GaussianSpAData,TimeFactor,"SIMoNe")
##TimeFactorSIMoNeNet<-FilterEdges(TimeFactorSIMoNeNet,0.4)
##plot(TimeFactorSIMoNeNet$H0$Network)
##plot(TimeFactorSIMoNeNet$H6$Network)
##plot(TimeFactorSIMoNeNet$H24$Network)

#MultiSpAData<-MultiDEGeneExpr(GaussianSpAData,DEGeneExpr(t(SpADataExpression[18:34,]),
# SpADEGenes[18:34,]),DEGeneExpr(t(SpADataExpression[35:51,]),SpADEGenes[35:51,])) # We
# #gather multiple lists of DEGeneExpr objects. Should come from the same experiment, but
# #not compulsory.
#MultiSpANetworks<-MultiNetworks(MultiSpAData,
# SelectInteractionsSTRING=c("coexpression","experimental","knowledge"),STRINGThreshold=0.9,
# FilterSIMoNeOptions=list(Threshold=0.4),Factors=StatusFactor) # We infer all kinds of
# #networks in the MultiDEGeneExpr object
## If you wish to add gene annotations (can take a while):
##MultiSpANetworks<-MultiNetworks(MultiSpAData,
## SelectInteractionsSTRING=c("coexpression","experimental","knowledge"),STRINGThreshold=0.9,
## FilterSIMoNeOptions=list(Threshold=0.4),Factors=StatusFactor,
## STRINGOptions=list(AddAnnotations=TRUE),
## SIMoNeOptions=list(AddAnnotations=TRUE),
## WGCNAOptions=list(AddAnnotations=TRUE))

## The following section is commented out due to the dependence on Cytoscape and cyREST
##installation.
## Before using this part of code, please be sure to have installed the last version of Cytoscape
##here: http://www.cytoscape.org/download.php
## And the cyREST plugin: http://apps.cytoscape.org/apps/cyrest
## If you can't use Cytoscape or cyREST, you can still construct networks and export these in
##a correct file formats with the previous sections.

##resetCytoscapeSession() # We reset the Cytoscape session
##addNetworkStyle("STRINGNet.noannot",class(SpASTRINGNet),Annotations=FALSE,
## points.size.map="P.Value",points.fill.map="logFC") # We add the style in Cytoscape for
## displaying STRINGNet objects
##NetId<-addGraphToCytoscape(SpASTRINGNet,StyleName="STRINGNet.noannot") # We add the global
## STRINGNet object. This network won't show up, due to its large size (numer of edges
## above 10). If you absolutely want to visualize the network, right click on this and
## create view.
```

```
##NetId<-addGraphToCytoscape(PPISpASTRINGNet,StyleName="STRINGNet.noannot") # We add the PPI
## STRINGNet object
##addNetworkStyle("ShortPathSTRINGNet.noannot",class(shortPathSpANet),Annotations=FALSE,
## points.size.map="P.Value",points.fill.map="logFC") # We add the style in Cytoscape for
## displaying ShortPathSTRINGNet objects
##NetId<-addGraphToCytoscape(shortPathSpANet,StyleName="ShortPathSTRINGNet.noannot") # We add the
## ShortPathSTRINGNet object
##addNetworkStyle("SIMoNeNet.noannot",class(GlobalSIMoNeNet),Annotations=FALSE,
## points.size.map="P.Value",points.fill.map="logFC") # We add the style in Cytoscape for
## displaying SIMoNeNet objects
##NetId<-addGraphToCytoscape(GlobalSIMoNeNet,StyleName="SIMoNeNet.noannot") # We add the
## SIMoNeNet object
##addNetworkStyle("WGCNANet.noannot",class(GlobalWGCNANet),Annotations=FALSE,
## points.size.map="P.Value",points.fill.map="logFC") # We add the style in Cytoscape for
## displaying WGCNANet objects
##NetId<-addGraphToCytoscape(GlobalWGCNANet,StyleName="WGCNANet.noannot") # We add the
## WGCNANet object
##addFactorGraphsToCytoscape(StatusFactorSIMoNeNet,
## StyleNames=rep("SIMoNeNet.noannot",length(StatusFactorSIMoNeNet))) # We add the
## FactorNetworks object
##saveCytoscapeSession("SingleNetworks",overwrite=T) # We save the Cytoscape session

##resetCytoscapeSession() # We reset the Cytoscape session
##addMultiGraphToCytoscape(MultiSpANetworks,points.size.map="P.Value",points.fill.map="logFC")
## We add the MultiNetworks object
##saveCytoscapeSession("MultiNetworks",overwrite=T) # We save the Cytoscape session
```

---

addFactorGraphsToCytoscape

Add FactorNetworks object to Cytoscape

Description

This function allows to import an object of class FactorNetworks into a Cytoscape session.

Usage

```
addFactorGraphsToCytoscape(FactorNets, Name=deparse(substitute(FactorNets)),
LayoutNames=rep("force-directed",length(FactorNets)),
StyleNames=sapply(FactorNets,function(x) class(x[["Network"]])), port.number=1234)
```

Arguments

FactorNets	An object of class FactorNetworks
Name	The name of the added network in each collection
LayoutNames	The layout name to display the network in Cytoscape. By default it is "force-directed".
StyleNames	The style name to display the network in Cytoscape. We advice you to use addNetworkStyle() before this function. By default it is the network object class.

`port.number` The local port number used by cyREST plugin to communicate with Cytoscape. By default it uses 1234.

Details

This function creates a collection for each level of factor, and adds the corresponding network in this collection. Cytoscape must be running during the use of this function.

See Also

checkCytoscapeRunning, addNetworkStyle, FactorNetworks.default

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# data(SpASamples)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# StatusFactor<-paste(SpASamples$status,SpASamples$b27,sep=".")
# names(StatusFactor)=SpASamples$chipnum

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# GlobalSIMoNeNetNF<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNetNF,0.4)

# StatusFactorSIMoNeNet<-FactorNetworks(GaussianSpAData,StatusFactor,"SIMoNe")
# StatusFactorSIMoNeNet<-FilterEdges(StatusFactorSIMoNeNet,0.4)

# resetCytoscapeSession()
# addNetworkStyle("SIMoNeNet",class(GlobalSIMoNeNet),points.size.map="P.Value",
# points.fill.map="logFC")
# addFactorGraphsToCytoscape(StatusFactorSIMoNeNet)
```

---

addGraphToCytoscape *Add network to Cytoscape*

Description

This function allows to import a network created from the stringgaussnet package into Cytoscape, with the use of the cyREST plugin.

Usage

```
addGraphToCytoscape(Network, Collection = class(Network),
Name = deparse(substitute(Network)), LayoutName = "force-directed",
StyleName = Collection, port.number = 1234)
```

Arguments

Network	A network object from the stringgaussnet package. Can be of class STRINGNet, ShortPathSTRINGNet, SIMoNeNet or WGCNANet.
Collection	The collection name used in Cytoscape. By default it is the network object class.
Name	The network name used in Cytoscape
LayoutName	The layout name to display the network in Cytoscape. By default it is "force-directed".
StyleName	The style name to display the network in Cytoscape. We advice you to use addNetworkStyle() before this function. By default it is the collection name.
port.number	The local port number used by cyREST plugin to communicate with Cytoscape. By default it uses 1234.

Details

Cytoscape must be running during the use of this function, with the activation of the cyREST plugin. Please see checkCytoscapeRunning() for more details.

Value

The network ID in the Cytoscape session.

See Also

addNetworkStyle, checkCytoscapeRunning

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# data(SpASamples)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# StatusFactor<-paste(SpASamples$status,SpASamples$b27,sep=".")
# names(StatusFactor)=SpASamples$chipnum

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# GlobalSIMoNeNetNF<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNetNF,0.4)

# resetCytoscapeSession()
# addNetworkStyle("SIMoNeNet",class(GlobalSIMoNeNet),points.size.map="P.Value",
# points.fill.map="logFC")
# NetId<-addGraphToCytoscape(GlobalSIMoNeNet)
```

addMultiGraphToCytoscape *Add MultiNetworks to Cytoscape*

Description

This function allows to import an object of class MultiNetworks into a Cytoscape session. This function automatically adds network styles for each class.

Usage

```
addMultiGraphToCytoscape(MultiNets, points.size.map = "PValue", min.points.value = 0.05,
max.points.value = 0, points.fill.map = "FC", min.points.fill = -2,
max.points.fill = 2, LayoutName = "force-directed", port.number = 1234)
```

Arguments

MultiNets	An object of class MultiNetworks
points.size.map	Node attribute for which the node size mapping is done. By default it is "PValue", which is the p-value from DE genes analysis results.
min.points.value	Maximum value of node attribute for which the size is minimal. By default it is 0.05.
max.points.value	Minimum value of node attribute for which the size is maximal. By default it is 0.
points.fill.map	Node attribute for which the node color mapping is done. By default it is the fold change.
min.points.fill	Minimum value for which the color mapping is done. By default it is -2.
max.points.fill	Maximum value for which the color mapping is done. By default it is 2.
LayoutName	The layout name used to display the network in Cytoscape. By default it is "force-directed".
port.number	The local port number used by cyREST plugin to communicate with Cytoscape. By default it uses 1234.

Details

Cytoscape must be running during the use of this function, with the activation of the cyREST plugin. Please see checkCytoscapeRunning() for more details. This function adds network for each item in the network list, and a collection is attributed for each network class and factor level if used. This also adds automatically pre-defined styles for each network class.

See Also

addNetworkStyle, addGraphToCytoscape, MultiNetworks.default, checkCytoscapeRunning

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# data(SpASamples)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# StatusFactor<-paste(SpASamples$status,SpASamples$b27,sep=".")
# names(StatusFactor)=SpASamples$chipnum

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# MultiSpAData<-MultiDEGeneExpr(GaussianSpAData,DEGeneExpr(t(SpADataExpression[18:34,]),
# SpADEGenes[18:34,]),DEGeneExpr(t(SpADataExpression[35:51,]),SpADEGenes[35:51,]))
# MultiSpANetworks<-MultiNetworks(MultiSpAData,
# SelectInteractionsSTRING=c("coexpression","experimental","knowledge"),STRINGThreshold=0.9,
# FilterSIMoNeOptions=list(Threshold=0.4),Factors=StatusFactor,
# STRINGOptions=list(AddAnnotations=F),SIMoNeOptions=list(AddAnnotations=F),
# WGCNAOptions=list(AddAnnotations=F))

# resetCytoscapeSession()
# addMultiGraphToCytoscape(MultiSpANetworks,points.size.map="P.Value",points.fill.map="logFC")
```

---

addNetworkStyle *Add network style to Cytoscape*

Description

This function allows to add pre-defined styles to properly display networks from package stringgaussnet in Cytoscape.

Usage

```
addNetworkStyle(style.name, style.class, Annotations = F, points.size.map = "PValue",
min.points.value = 0.05, max.points.value = 0, points.fill.map = "FC",
min.points.fill = -2, max.points.fill = 2, port.number = 1234)
```

Arguments

style.name	The name you want to give to the style
style.class	The class of network used for edge mapping. Can be either "STRINGNet", "ShortPathSTRINGNet", "SIMoNeNet" or "WGCNANet". It depends on the class of network you wish to import.

`Annotations` Does the style must include gene annotations? It depends if you wanted to add annotations during your network construction.

`points.size.map`
Node attribute for which the node size mapping is done. By default it is "PValue", which is the p-value from DE genes analysis results.

`min.points.value`
Maximum value of node attribute for which the size is minimal. By default it is 0.05.

`max.points.value`
Minimum value of node attribute for which the size is maximal. By default it is 0.

`points.fill.map`
Node attribute for which the node color mapping is done. By default it is the fold change.

`min.points.fill`
Minimum value for which the color mapping is done. By default it is -2.

`max.points.fill`
Maximum value for which the color mapping is done. By default it is 2.

`port.number` The local port number used by cyREST plugin to communicate with Cytoscape. By default it uses 1234.

Details

This function only adds a pre-defined style to the Cytoscape session, and does not import a network object. Cytoscape must be running with the plugin cyREST installed. This is much advised to use this function before importing your networks into the Cytoscape session, excepted for addMultiGraphToCytoscape(). The node size mapping is inversely proportional to the selected attribute. This is in the aim to preferably see genes with lowest p-values. The node color is blue for under-expressed genes, and red for over-expressed genes, depending on the fold change in DE genes analysis results. The node mapping is common for all network classes in the stringgaussnet package, whereas the edge mapping is more specific.

See Also

addGraphToCytoscape, checkCytoscapeRunning, addSTRINGNetMappings, addShortPathSTRINGNetMappings, addSIMoNeNetMappings, addWGCNANetMappings

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# data(SpASamples)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# StatusFactor<-paste(SpASamples$status,SpASamples$b27,sep=".")
# names(StatusFactor)=SpASamples$chipnum

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])
```

```
# GlobalSIMoNeNetNF<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNetNF,0.4)

# resetCytoscapeSession()
# addNetworkStyle("SIMoNeNet",class(GlobalSIMoNeNet),points.size.map="P.Value",
# points.fill.map="logFC")
# NetId<-addGraphToCytoscape(GlobalSIMoNeNet)
```

`addShortPathSTRINGNetMappings` *Add Cytoscape mapping for ShortPathSTRINGNet*

Description

This function allows to add edge mapping in a pre-defined Cytoscape style. This function is called in addNetworkStyle().

Usage

```
addShortPathSTRINGNetMappings(mappings)
```

Arguments

`mappings`	A list where will be added the mapping attributes

Details

The edge transparency depends on the distance computed by Dijkstra's algorithm. Direct interactions are displayed by black solid lines, whereas indirect interactions are displayed by blue dashed lines.

Value

The mappings list updated, containing added JSON-like lists (converted to JSON objects by the addNetworkStyle() function).

See Also

`addNetworkStyle`

addSIMoNeNetMappings *Add Cytoscape mapping for SIMoNeNet*

Description

This function allows to add edge mapping in a pre-defined Cytoscape style. This function is called in addNetworkStyle().

Usage

```
addSIMoNeNetMappings(mappings)
```

Arguments

mappings	A list where will be added the mapping attributes

Details

The edge colors depend on spearman's rho values. This is red for positive correlations, and blue for negative correlations. The edge width is inversely proportionnal to the spearman's test p-value.

Value

The mappings list updated, containing added JSON-like lists (converted to JSON objects by the addNetworkStyle function).

See Also

addNetworkStyle

addSkeletonDefaults *Add default values for Cytoscape styles*

Description

This function is called by addNetworkStyle() to add common default values for styles of all network classes in the stringaussnet package.

Usage

```
addSkeletonDefaults(defaults, Annotations)
```

Arguments

defaults	A list where will be added the default attributes
Annotations	A boolean variable indicating whether gene annotations were added to the network and then to account this in the node style

Details

If annotations were added, then the node border is a little wider.

Value

The defaults list updated, containing added JSON-like lists (converted to JSON objects by the addNetworkStyle function).

See Also

addSkeletonMappings, addNetworkStyle

---

addSkeletonMappings *Add default mappings for Cytoscape styles*

Description

This function adds common node mappings during the call of addNetworkStyle() for all network classes of the stringgaussnet package.

Usage

```
addSkeletonMappings(mappings, Annotations, points.size.map, min.points.value,
max.points.value, points.fill.map, min.points.fill, max.points.fill)
```

Arguments

mappings	A list where will be added the mapping attributes
Annotations	A boolean variable indicating whether gene annotations were added to the network and then to account this in the node style
points.size.map	Node attribute for which the node size mapping is done
min.points.value	Maximum value of node attribute for which the size is minimal
max.points.value	Minimum value of node attribute for which the size is maximal
points.fill.map	Node attribute for which the node color mapping is done
min.points.fill	Minimum value for which the color mapping is done
max.points.fill	Maximum value for which the color mapping is done

Details

If annotations were added, then the node border color depends on the cellular localization of the gene product.

Value

The mappings list updated, containing added JSON-like lists (converted to JSON objects by the addNetworkStyle function).

See Also

addSkeletonDefaults, addNetworkStyle

---

addSTRINGNetMappings *Add Cytoscape mapping for STRINGNet*

Description

This function allows to add edge mapping in a pre-defined Cytoscape style. This function is called in addNetworkStyle().

Usage

```
addSTRINGNetMappings(mappings)
```

Arguments

mappings	A list where will be added the mapping attributes

Details

The edge colors depend on the interaction source given by STRING, black being attributed for combined scores. The edge transparency depends on the confidence score given by STRING.

Value

The mappings list updated, containing added JSON-like lists (converted to JSON objects by the addNetworkStyle function).

See Also

addNetworkStyle

addWGCNANetMappings *Add Cytoscape mapping for WGCNANet*

Description

This function allows to add edge mapping in a pre-defined Cytoscape style. This function is called in addNetworkStyle().

Usage

```
addWGCNANetMappings(mappings)
```

Arguments

mappings	A list where will be added the mapping attributes

Details

The edge colors depend on spearman's rho values. This is red for positive correlations, and blue for negative correlations. The edge width is inversely proportionnal to the spearman's test p-value.

Value

The mappings list updated, containing added JSON-like lists (converted to JSON objects by the addNetworkStyle function).

See Also

addNetworkStyle

applyLayout *Apply layout to a network in Cytoscape*

Description

This functions allows to apply a given layout to a particular network in Cytoscape, with the use of the cyREST plugin.

Usage

```
applyLayout(network.suid, layout.name, port.number = 1234)
```

Arguments

network.suid	The network ID in the cytoscape session
layout.name	The layout name to display the network in Cytoscape
port.number	The local port number used by cyREST plugin to communicate with Cytoscape

Details

Layouts from yFiles must be set manually in Cytoscape, because it can not be used by cyREST for license use reasons.

See Also

addGraphToCytoscape

---

applyStyle *Apply style to a network in Cytoscape*

Description

This function helps to apply an existing style to particular network in a Cytoscape session, with the use of the cyREST plugin.

Usage

```
applyStyle(style.name, network.suid, port.number = 1234)
```

Arguments

style.name	The name of the existing style
network.suid	The network ID in the cytoscape session
port.number	The local port number used by cyREST plugin to communicate with Cytoscape

Details

The style must exist in the Cytoscape session. This function is already included in addGraphToCytoscape().

See Also

addGraphToCytoscape, addNetworkStyle

as.igraph.stringgaussnet *Convert stringgaussnet network into igraph*

Description

This function converts any network object from the package stringgaussnet into an igraph object (package igraph).

Usage

```
## S3 method for class 'stringgaussnet'
as.igraph(x, ...)
```

Arguments

x	A network object from stringgaussnet package. Can be of class STRINGNet, ShortPathSTRINGNet, SIMoNeNet or WGCNANet.
...	Additional parameters. Not used here

Details

This function is used in this package to convert a network into an igraph object, and then into a json object in order to import the network into Cytoscape. But you can use this function anywhere in your R script if you wish to manipulate your network with the igraph package.

Value

An igraph object with non-directed edges (can be multiple).

See Also

addGraphToCytoscape

Examples

```
data(SpADataExpression)
data(SpADEGenes)
SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)
NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])
GlobalSIMoNeNet<-getSIMoNeNet(GaussianSpAData,AddAnnotation=FALSE)
iGraphSpAObj<-as.igraph.stringgaussnet(GlobalSIMoNeNet)
```

checkCytoscapeRunning *Check Cytoscape running*

Description

This function checks if Cytoscape is running in your os. It is used in any function to import network into Cytoscape. This communication with Cytoscape is done with the plugin cyREST.

Usage

```
checkCytoscapeRunning(port.number = 1234)
```

Arguments

port.number	The local port number used by cyREST plugin to communicate with Cytoscape. By default it uses 1234.

Details

If you wish to download Cytoscape, please go here: http://www.cytoscape.org/download.php If you wish to install the Cytoscape plugin cyREST, please go here: http://apps.cytoscape.org/apps/cyrest cyREST works as a local API to communicate with Cytoscape through the use of URIs.

Value

Returns TRUE in case of success, or an error message in case of failure.

See Also

addGraphToCytoscape

Examples

```
# checkCytoscapeRunning()
```

compareFactorNetworks *Compare levels of FactorNetworks*

Description

This function draws a series of plots to compare different levels of a factor used to infer multiple gaussian networks, with an object of class FactorNetworks.

Usage

```
compareFactorNetworks(Networks, Colors = rainbow(length(Networks)), interactiveMode = T)
```

Arguments

Networks	An object of class FactorNetworks
Colors	Colors to plot for each level of factor
interactiveMode	Boolean variable indicating whether the plots are in interactive mode. If false, it is useful for automatically saving plots in a single pdf file.

Details

The first plot shows the absolute values of spearman's rho in the different level groups. The second one shows p-values in those different groups. The third displays the respective numbers of edges. The last one shows node connectivities.

See Also

FactorNetworks.default

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# data(SpASamples)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# StatusFactor<-paste(SpASamples$status,SpASamples$b27,sep=".")
# names(StatusFactor)=SpASamples$chipnum

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# GlobalSIMoNeNetNF<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNetNF,0.4)

# StatusFactorSIMoNeNet<-FactorNetworks(GaussianSpAData,StatusFactor,"SIMoNe")
# StatusFactorSIMoNeNet<-FilterEdges(StatusFactorSIMoNeNet,0.4)
# compareFactorNetworks(StatusFactorSIMoNeNet)
```

---

compareGaussNetworks *Compare gaussian networks*

Description

A function to compare gaussian networks. It was originally created to compare between SIMoNe and WGCNA networks, but you can compare any network with nodes in common.

Usage

```
compareGaussNetworks(Network1, Network2, Names = c("Network1", "Network2"),
Colors = c("yellow", "blue", "green"), interactiveMode = T, RhoThreshold = 0.4,
PValueThreshold = 0.05)
```

Arguments

Network1	First gaussian network to compare
Network2	Second gaussian network to compare
Names	Names attributed to networks
Colors	Colors attributed to first, second and common networks
interactiveMode	Boolean variable indicating whether the plots are in interactive mode. If false, it is useful for automatically saving plots in a single pdf file.
RhoThreshold	Threshold to display vertical dashed line in the last plot
PValueThreshold	Threshold to display horizontal dashed line in the last plot

Details

Firstly, the function plots a venn diagram to compare network connectivities. Then we can see a series of boxplots displaying absolute values of rhos and spearman's p-values in the first network, the second network and the common network. Afterwards, we see mean node connectivities in each network, and finally a plot of spearman's p-values as a function of rhos.

See Also

compareFactorNetworks

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# pickWGCNAParam(GaussianSpAData)
# GlobalWGCNANet<-getWGCNANet(GaussianSpAData)
# print(GlobalWGCNANet,5)
# summary(GlobalWGCNANet)
# plot(GlobalWGCNANet)
# export(GlobalWGCNANet,"GlobalWGCNANet",T)

# compareGaussNetworks(GlobalSIMoNeNet,GlobalWGCNANet,c("SIMoNe","WGCNA"))
```

computeCombinedScores *Compute STRING combined scores*

Description

A function to compute STRING combined scores used in selectInteractionTypes()

Usage

```
computeCombinedScores(ScoresSTRING, hscores)
```

Arguments

ScoresSTRING	Confidence scores computed by STRING
hscores	Homology scores computed by STRING

Details

This method of STRING computation was used in STRING version 8.

Value

A vector of combined scores.

References

http://string-stitch.blogspot.fr/2010/03/combining-scores-right-way.html https://bitbucket.org/mkuhn/stringtools/src/dcc109

See Also

selectInteractionTypes

computeSimilarities *Compute similarities for WGCNA*

Description

This function computes spearman's rhos from an expression data matrix and transforms it into similarity score. This function is used in getWGCNANet().

Usage

```
computeSimilarities(DEGeneExpr)
```

Arguments

DEGeneExpr	Object of class DEGeneExpr

Details

The similarity score S = (1+rho) / 2. In this way, S is always positive while still keeping the correlation sign.

Value

A matrix of similarity scores.

See Also

getWGCNANet

---

convertToDistGraph *Convert to distance graph from STRINGNet*

Description

This function converts confidence scores into distance from a STRINGNet object, in order to compute the shortest paths. This function is called in getShortestPaths().

Usage

```
convertToDistGraph(Network)
```

Arguments

Network	An object of class STRINGNet

Details

Combined scores S are converted to distances D for each interaction i with Di=max(S)+1-Si.

Value

A distance matrix converted from a STRINGNet object

See Also

getShortestPaths

DEGeneExpr.default *Creation of DEGenesExpr object.*

Description

This function allows to create an object of class DEGeneExpr from expression data and DE genes analysis results.

Usage

```
## Default S3 method:
DEGeneExpr(x, y, Identifier = 0, ...)
```

Arguments

x	A numeric matrix of expression data with samples as rows and genes as columns.
y	Results from DE genes analysis (for example LIMMA). Rows are genes, and columns are gene attributes. This is suggested to have at least fold changes and p-values.
Identifier	Which column identifies genes in DEGenesResults? If equals to 0, row names are picked. Identifiers must be identical to column names in DataExpression.
...	Other parameters from the generic function. Not used here.

Value

An object of class DEGenesExpr, which is a list containing DataExpression and DEGenesResults. This object is the basis for using all other functions in the package stringgaussnet.

See Also

print.DEGeneExpr, DEGeneExpr

Examples

```
data(SpADataExpression)
data(SpADEGenes)
SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)
print(SpAData)
print(SpAData,10) # Prints only 10 first lines of each matrix.
```

`export.ShortPathSTRINGNet` *Export ShortPathSTRINGNet*

Description

Function to export a ShortPathSTRINGNet object to a directory in standard table file formats. Those files can be imported for example into Cytoscape.

Usage

```
## S3 method for class 'ShortPathSTRINGNet'
export(x, dirname, overwrite = F, ...)
```

Arguments

`x`	Object of class ShortPathSTRINGNet
`dirname`	Directory path where will be saved network files
`overwrite`	Boolean variable indicating whether the function deletes and recreates an existing directory with the same path
`...`	Additional parameters. Not used here

Details

This function creates two kinds of table files: edge and node attributes. All files are writen with column names at first line and with tabulations as field separator. Primary and secondary node attributes are exported in two distinct files.

Note

Please notice that this functions does not create any style for cytoscape and only network structure with attributes will be saved. To import directly your network into a Cytoscape session and to save this, please report to attributed functions of the package.

See Also

`export`, `export.STRINGNet`, `export.SIMoNeNet`, `export.WGCNANet`

Examples

```
data(SpADataExpression)
data(SpADEGenes)
SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# SpASTRINGNet<-getSTRINGNet(SpAData)
# Can be longer.
```

```
# SpASTRINGNet<-getSTRINGNet(SpAData,AddAnnotations=FALSE)
# print(SpASTRINGNet,5)
# summary(SpASTRINGNet)
# PPISpASTRINGNet<-selectInteractionTypes(SpASTRINGNet,
# c("coexpression","experimental","knowledge"),0.9)
# export(PPISpASTRINGNet,"PPISpASTRINGNet",T)

# shortPathSpANet<-getShortestPaths(PPISpASTRINGNet)
# shortPathSpANet<-FilterEdges(shortPathSpANet,2.2)
# print(shortPathSpANet,5)
# summary(shortPathSpANet)
# export(shortPathSpANet,"shortPathSpANet",T)
```

`export.SIMoNeNet` *Export SIMoNeNet*

Description

Function to export a SIMoNeNet object to a directory in standard table file formats. Those files can be imported for example into Cytoscape.

Usage

```
## S3 method for class 'SIMoNeNet'
export(x, dirname, overwrite = F, ...)
```

Arguments

`x`	Object of class SIMoNeNet
`dirname`	Directory path where will be saved network files
`overwrite`	Boolean variable indicating whether the function deletes and recreates an existing directory with the same path
`...`	Additional parameters. Not used here.

Details

This function creates two kinds of table files: edge and node attributes. All files are writen with column names at first line and with tabulations as field separator. Primary and secondary node attributes are exported in two distinct files.

Note

Please notice that this functions does not create any style for cytoscape and only network structure with attributes will be saved. To import directly your network into a Cytoscape session and to save this, please report to attributed functions of the package.

See Also

export, export.STRINGNet, export.ShortPathSTRINGNet, export.WGCNANet

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# pickSIMoNeParam(GaussianSpAData)

# GlobalSIMoNeNet<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNet,0.4)
# print(GlobalSIMoNeNet,5)
# summary(GlobalSIMoNeNet)
# plot(GlobalSIMoNeNet)

# export(GlobalSIMoNeNet,"GlobalSIMoNeNet",T)
```

---

export.STRINGNet *Export STRINGNet*

Description

Function to export a STRINGNet object to a directory in standard table file formats. Those files can be imported for example into Cytoscape.

Usage

```
## S3 method for class 'STRINGNet'
export(x, dirname, overwrite = F, ...)
```

Arguments

x	Object of class STRINGNet
dirname	Directory path where will be saved network files
overwrite	Boolean variable indicating whether the function deletes and recreates an existing directory with the same path
...	Additional parameters. Not used here.

Details

This function creates two kinds of table files: edge and node attributes. All files are writen with column names at first line and with tabulations as field separator. Primary and secondary node attributes are exported in two distinct files.

Note

Please notice that this functions does not create any style for cytoscape and only network structure with attributes will be saved. To import directly your network into a Cytoscape session and to save this, please report to attributed functions of the package.

See Also

export, export.ShortPathSTRINGNet, export.SIMoNeNet, export.WGCNANet

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# SpASTRINGNet<-getSTRINGNet(SpAData)
# Can be longer.

# SpASTRINGNet<-getSTRINGNet(SpAData,AddAnnotations=FALSE)
# print(SpASTRINGNet,5)
# summary(SpASTRINGNet)
# PPISpASTRINGNet<-selectInteractionTypes(SpASTRINGNet,
# c("coexpression","experimental","knowledge"),0.9)
# export(PPISpASTRINGNet,"PPISpASTRINGNet",T)

# shortPathSpANet<-getShortestPaths(PPISpASTRINGNet)
# shortPathSpANet<-FilterEdges(shortPathSpANet,2.2)
# print(shortPathSpANet,5)
# summary(shortPathSpANet)
# export(shortPathSpANet,"shortPathSpANet",T)
```

---

export.WGCNANet *Export WGCNANet*

Description

Function to export a WGCNANet object to a directory in standard table file formats. Those files can be imported for example into Cytoscape.

Usage

```
## S3 method for class 'WGCNANet'
export(x, dirname, overwrite = F, ...)
```

Arguments

x	Object of class WGCNANet
dirname	Directory path where will be saved network files
overwrite	Boolean variable indicating whether the function deletes and recreates an existing directory with the same path
...	Additional parameters. Not used here.

Details

This function creates two kinds of table files: edge and node attributes. All files are writen with column names at first line and with tabulations as field separator. Primary and secondary node attributes are exported in two distinct files.

Note

Please notice that this functions does not create any style for cytoscape and only network structure with attributes will be saved. To import directly your network into a Cytoscape session and to save this, please report to attributed functions of the package.

See Also

export, export.STRINGNet, export.ShortPathSTRINGNet, export.SIMoNeNet

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# pickWGCNAParam(GaussianSpAData)
# GlobalWGCNANet<-getWGCNANet(GaussianSpAData)
# print(GlobalWGCNANet,5)
# summary(GlobalWGCNANet)
# plot(GlobalWGCNANet)
# export(GlobalWGCNANet,"GlobalWGCNANet",T)

# compareGaussNetworks(GlobalSIMoNeNet,GlobalWGCNANet,c("SIMoNe","WGCNA"))
```

FactorNetworks.default

Function to create an object of class FactorNetworks

Description

This function allows to infer multiple gaussian networks from a single DEGeneExpr object with a factor attributed to samples.

Usage

```
## Default S3 method:
FactorNetworks(x, Factor, method = "SIMoNe", options = NULL, ...)
```

Arguments

x	An object of class DEGeneExpr
Factor	A factor attributed to samples. Names must fit with sample names. If it is a character vector, it is automatically converted to a factor.
method	Which method for gaussian network inference to use. Can be either "SIMoNe" or "WGCNA".
options	A list giving options for the gaussian network inference method. Each name corresponds to the parameters of the function getSIMoNeNet() or getWGCNANet().
...	Additional parameters. Not used here.

Value

An object of class FactorNetworks, which is a list of gaussian networks for each level of the given factor.

See Also

FactorNetworks, print.FactorNetworks, FilterEdges.FactorNetworks, addFactorGraphsToCytoscape

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# data(SpASamples)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# StatusFactor<-paste(SpASamples$status,SpASamples$b27,sep=".")
# names(StatusFactor)=SpASamples$chipnum

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])
```

```
# GlobalSIMoNeNetNF<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNetNF,0.4)

# StatusFactorSIMoNeNet<-FactorNetworks(GaussianSpAData,StatusFactor,"SIMoNe")
# StatusFactorSIMoNeNet<-FilterEdges(StatusFactorSIMoNeNet,0.4)

# resetCytoscapeSession()
# addNetworkStyle("SIMoNeNet",class(GlobalSIMoNeNet),points.size.map="P.Value",
# points.fill.map="logFC")
# addFactorGraphsToCytoscape(StatusFactorSIMoNeNet)
```

`FilterEdges.FactorNetworks`

Filter edges in FactorNetworks

Description

Function to filter on edge attribute in an FactorNetworks object.

Usage

```
## S3 method for class 'FactorNetworks'
FilterEdges(x, Threshold, Superior = T, AttributeFilter = "Rho",
Absolute = T, ...)
```

Arguments

`x`	An object of class FactorNetworks
`Threshold`	Threshold used to filter on edge attribute
`Superior`	Boolean variable indicating whether values must be superior or inferior to the threshold
`AttributeFilter`	Character indicating on which edge attribute to filter
`Absolute`	Boolean indicating whether the attribute must transformed into absolute values before filtering
`...`	Additional parameters. Not used here

Value

Object of class FactorNetworks with filtered edges

See Also

`FactorNetworks`, `FactorNetworks.default`, `print.FactorNetworks`

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# data(SpASamples)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# StatusFactor<-paste(SpASamples$status,SpASamples$b27,sep=".")
# names(StatusFactor)=SpASamples$chipnum

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# GlobalSIMoNeNetNF<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNetNF,0.4)

# StatusFactorSIMoNeNet<-FactorNetworks(GaussianSpAData,StatusFactor,"SIMoNe")
# StatusFactorSIMoNeNet<-FilterEdges(StatusFactorSIMoNeNet,0.4)
```

FilterEdges.ShortPathSTRINGNet

Filter edges in ShortPathSTRINGNet

Description

Function to filter on edge distance or number of intermediates in a ShortPathSTRINGNet object.

Usage

```
## S3 method for class 'ShortPathSTRINGNet'
FilterEdges(x, Threshold, AttributeFilter = "Distance", ...)
```

Arguments

x	Object of class ShortPathSTRINGNet
Threshold	Maximum threshold used to filter on edge attributes
AttributeFilter	Character indicating on which edge attribute to filter. Can be "Distance" or "NIntermediates".
...	Additional parameters. Not used here

Value

Object of class ShortPathSTRINGNet with filtered edges.

See Also

FilterEdges, FilterEdges.SIMoNeNet, FilterEdges.FactorNetworks

Examples

```
data(SpADataExpression)
data(SpADEGenes)
SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# SpASTRINGNet<-getSTRINGNet(SpAData)
# Can be longer.

# SpASTRINGNet<-getSTRINGNet(SpAData,AddAnnotations=FALSE)
# print(SpASTRINGNet,5)
# summary(SpASTRINGNet)
# PPISpASTRINGNet<-selectInteractionTypes(SpASTRINGNet,
# c("coexpression","experimental","knowledge"),0.9)

# shortPathSpANet<-getShortestPaths(PPISpASTRINGNet)
# shortPathSpANet<-FilterEdges(shortPathSpANet,2.2)
# print(shortPathSpANet,5)
# summary(shortPathSpANet)
```

FilterEdges.SIMoNeNet *Filter edges in SIMoNeNet*

Description

Function to filter on theta score or spearman's statistics in a SIMoNeNet object.

Usage

```
## S3 method for class 'SIMoNeNet'
FilterEdges(x, Threshold, Superior = T, AttributeFilter = "Rho",
Absolute = T, ...)
```

Arguments

x	Object of class SIMoNeNet
Threshold	Threshold used to filter on edge attribute
Superior	Boolean variable indicating whether values must be superior or inferior to the threshold
AttributeFilter	Character indicating on which edge attribute to filter. Can be "Rho", "P.Value" or "Theta".
Absolute	Boolean indicating whether the attribute must be transformed into absolute values before filtering
...	Additional parameters. Not used here

Value

Object of class SIMoNeNet with filtered edges

See Also

SIMoNeNet, SIMoNeNet.default, getSIMoNeNet, print.SIMoNeNet, summary.SIMoNeNet, export.SIMoNeNet, pickSIMoNeParam

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# pickSIMoNeParam(GaussianSpAData)

# GlobalSIMoNeNet<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNet,0.4)
# print(GlobalSIMoNeNet,5)
# summary(GlobalSIMoNeNet)
# plot(GlobalSIMoNeNet)

# export(GlobalSIMoNeNet,"GlobalSIMoNeNet",T)
```

---

getGenesInformations *Get gene annotations*

Description

This function uses biomaRt and AnnotationDbi to add gene annotations as node attributes in a network from stringgaussnet package.

Usage

```
getGenesInformations(Identifiers, ensembl)
```

Arguments

Identifiers	Can be Ensembl IDs or HGNC symbols. Ensembl IDs are recommended.
ensembl	A mart dataset object

Details

Firstly, this function adds cellular localizations of gene products. A prioritization is performed to rank gene products localizations from nuclear, the most relevant, and then extracellular, plasma membrane and cytoplasm. Secondly, those annotations are added from biomaRt: chromosome name, band, strand, start and end positions, and gene descriptions.

Value

A matrix of node attributes with annotations.

See Also

getSTRINGNet, getSIMoNeNet, getWGCNANet

`getMartDatasets` *Gert maRt datasets*

Description

This function returns the list of datasets in biomaRt from Ensembl.

Usage

```
getMartDatasets()
```

See Also

getGenesInformations

`getShortestPaths` *Get shortest paths between given nodes in STRING network.*

Description

This function is dedicated to compute shortest paths and to shrink a STRING network between genes selected by the user.

Usage

```
getShortestPaths(Network, SelectedGenes = 0)
```

Arguments

`Network`	Object of class STRINGNet
`SelectedGenes`	Genes to keep after computation of shortest paths. If equals to 0, initial nodes from DE genes analysis results are selected.

Details

Shortest paths are computed with the Dijkstra's algorithm from the package igraph.

Value

An object of class ShortPathSTRINGNet.

See Also

ShortPathSTRINGNet, ShortPathSTRINGNet.default, print.ShortPathSTRINGNet, summary.ShortPathSTRINGNet, export.ShortPathSTRINGNet, FilterEdges.ShortPathSTRINGNet

Examples

```
data(SpADataExpression)
data(SpADEGenes)
SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# SpASTRINGNet<-getSTRINGNet(SpAData)
# Can be longer.

# SpASTRINGNet<-getSTRINGNet(SpAData,AddAnnotations=FALSE)
# print(SpASTRINGNet,5)
# summary(SpASTRINGNet)
# PPISpASTRINGNet<-selectInteractionTypes(SpASTRINGNet,
# c("coexpression","experimental","knowledge"),0.9)

# shortPathSpANet<-getShortestPaths(PPISpASTRINGNet)
# shortPathSpANet<-FilterEdges(shortPathSpANet,2.2)
# print(shortPathSpANet,5)
# summary(shortPathSpANet)
```

---

getSIMoNeNet *Infer SIMoNe network from expression data*

Description

This function infers a SIMoNe network from expression data. This gives a non-supervised gaussian network with partial correlation computations.

Usage

```
getSIMoNeNet(DEGeneExpr, NEdges = NA, ClusterMethod = "both", AddAnnotations = F,
MartDataset = "hsapiens_gene_ensembl")
```

Arguments

DEGeneExpr	Object of class DEGeneExpr
NEdges	Criter selection of SIMoNe model. Can be the number of edges, 'BIC' or 'AIC'. If it is set to NA, the function chooses the number of edges by computing the mean between those with maximal AIC and BIC scores.

ClusterMethod	Can be TRUE, FALSE, or 'both'. If it is set to 'both', the function computes networks with and without clustering constraints, and pick common edges between the both.
AddAnnotations	Boolean variable indicating whether gene annotations must be added through biomaRt
MartDataset	Which mart dataset to use for querying gene annotations through biomaRt. See getMartDatasets() for some help.

Value

An object of class SIMoNeNet. See SIMoNeNet.default() for more details.

Note

A precaution must be taken by choosing the parameters, and the expression data matrix dimensions. You can use pickSIMoNeParam() to help in the choice of parameters.

References

Chiquet, J. et al. SIMoNe Statistical Inference for MOdular NEtworks. Bioinforma. Oxf. Engl. 25, 417 (2009).

See Also

SIMoNeNet, SIMoNeNet.default, print.SIMoNeNet, summary.SIMoNeNet, export.SIMoNeNet, FilterEdges.SIMoNeNet, pickSIMoNeParam

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# pickSIMoNeParam(GaussianSpAData)

# GlobalSIMoNeNet<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNet,0.4)
# print(GlobalSIMoNeNet,5)
# summary(GlobalSIMoNeNet)
# plot(GlobalSIMoNeNet)

# export(GlobalSIMoNeNet,"GlobalSIMoNeNet",T)
```

`getSTRINGNet` *Get STRING network from gene identifiers*

Description

This function gets PPIs interactions between given genes through the STRING API. This functions uses an URI to query STRING, then an internet connection is required.

Usage

```
getSTRINGNet(DEGeneExpr, Identifier = 0, NAdditionalNodes = NA, Species = 9606,
ConvertAliases = T, AddAnnotations = F, MartDataset = "hsapiens_gene_ensembl")
```

Arguments

`DEGeneExpr`	Object of class DEGeneExpr. See DEGeneExpr.default() for more details.
`Identifier`	Which column in DE genes analysis results DEGeneExpr object is used as identifier for STRING. By default row names are taken when it equals to 0.
`NAdditionalNodes`	Number of additional nodes inserted by STRING
`Species`	From which species come gene identifiers. By default it is homo sapiens (9606).
`ConvertAliases`	Boolean variable indicating whether gene symbol aliases must be converted to HGNC symbols.
`AddAnnotations`	Boolean variable indicating whether gene annotations must be added through biomaRt
`MartDataset`	Which mart dataset to use for querying gene annotations through biomaRt

Details

Gene identifiers can be Ensembl IDs or HGNC symbols. STRING gives the number of additional nodes + 10 added nodes by default. If you don't want any additional nodes at all, you can set NAdditionalNodes = NULL. By default, when NAdditionalNodes is NA, twice the number of initial nodes + 10 are added. Species are entered with taxon identifiers. To see correspondance, please have a look here: http://www.uniprot.org/taxonomy Aliases are converted with the package limma. No internet connection is needed for this step. 2 kinds of annotations are added. First, stringgaussnet uses the R package biomart to get mainly genomic localization and gene description. Secondly, it adds cellular component terms with the package AnnotationDbi. A prioritization is performed to rank gene products localizations from nuclear, the most relevant, and then extracellular, plasma membrane and cytoplasm. To know which mart dataset to use for given species, please use getMartDatasets().

Value

An object of class STRINGNet. See STRINGNet.default() for more details.

See Also

print.STRINGNet, summary.STRINGNet, export.STRINGNet, getShortestPaths, getMartDatasets, selectInteractionTypes

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# SpASTRINGNet<-getSTRINGNet(SpAData)
# Can be longer.

# SpASTRINGNet<-getSTRINGNet(SpAData,AddAnnotations=FALSE)
# print(SpASTRINGNet,5)
# summary(SpASTRINGNet)
# PPISpASTRINGNet<-selectInteractionTypes(SpASTRINGNet,
# c("coexpression","experimental","knowledge"),0.9)

# shortPathSpANet<-getShortestPaths(PPISpASTRINGNet)
# shortPathSpANet<-FilterEdges(shortPathSpANet,2.2)
# print(shortPathSpANet,5)
# summary(shortPathSpANet)
```

---

getWGCNANet *Infer WGCNA network from expression data*

Description

This function infers a WGCNA network from expression data. This gives a gaussian network simply by filtering on correlations between expressions of each pair of genes. Dissimilarities and modules computations are not implemented, because the main purpose is to compare with SIMoNe results.

Usage

```
getWGCNANet(DEGeneExpr, SoftThreshold = 8, AThreshold = 0.85, AddAnnotations = F,
MartDataset = "hsapiens_gene_ensembl")
```

Arguments

DEGeneExpr	Object of class DEGeneExpr. See DEGeneExpr.default() for more details.
SoftThreshold	Soft threshold parameter (alpha) used for adjacency computation by sigmoid function. See pickWGCNAParam() for some help.
AThreshold	Threshold on adjacency score for edges inference. Generally it is 0.85.
AddAnnotations	Boolean variable indicating whether gene annotations must be added through biomaRt
MartDataset	Which mart dataset to use for querying gene annotations through biomaRt. See getMartDatasets() for some help.

Value

An object of class WGCNANet. See WGCNANet.default() for more details.

See Also

WGCNANet, WGCNANet.default, print.WGCNANet, summary.WGCNANet, export.WGCNANet, pickWGCNAParam, compareGaussNetworks

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# pickWGCNAParam(GaussianSpAData)
# GlobalWGCNANet<-getWGCNANet(GaussianSpAData)
# print(GlobalWGCNANet,5)
# summary(GlobalWGCNANet)
# plot(GlobalWGCNANet)
# export(GlobalWGCNANet,"GlobalWGCNANet",T)

# compareGaussNetworks(GlobalSIMoNeNet,GlobalWGCNANet,c("SIMoNe","WGCNA"))
```

MultiDEGeneExpr.default

Function to create an object of class MultiDEGeneExpr

Description

This function allows to create an object of class MultiDEGeneExpr from multiple DEGeneExpr objects.

Usage

```
## Default S3 method:
MultiDEGeneExpr(...)
```

Arguments

...	Objects of class DEGeneExpr to gather in the new object of class MultiDEGeneExpr

Value

An object of class MultiDEGeneExpr, which is a list of DEGeneExpr objects

See Also

MultiDEGeneExpr, print.MultiDEGeneExpr, MultiNetworks.default

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# data(SpASamples)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# StatusFactor<-paste(SpASamples$status,SpASamples$b27,sep=".")
# names(StatusFactor)=SpASamples$chipnum

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# MultiSpAData<-MultiDEGeneExpr(GaussianSpAData,DEGeneExpr(t(SpADataExpression[18:34,]),
# SpADEGenes[18:34,]),DEGeneExpr(t(SpADataExpression[35:51,]),SpADEGenes[35:51,]))
# MultiSpANetworks<-MultiNetworks(MultiSpAData,
# SelectInteractionsSTRING=c("coexpression","experimental","knowledge"),STRINGThreshold=0.9,
# FilterSIMoNeOptions=list(Threshold=0.4),Factors=StatusFactor,
# STRINGOptions=list(AddAnnotations=FALSE),SIMoNeOptions=list(AddAnnotations=FALSE),
# WGCNAOptions=list(AddAnnotations=FALSE))
```

MultiNetworks.default *Function to create an object of class MultiNetworks*

Description

This function allows to create an object of class MultiNetworks from an object of class MultiDEGeneExpr. This is a wrapper of all methods available in the stringgaussnet package.

Usage

```
## Default S3 method:
MultiNetworks(x, Methods = c("STRING", "SIMoNe", "WGCNA"), STRINGOptions = NULL,
SIMoNeOptions = NULL, WGCNAOptions = NULL, SelectInteractionsSTRING = NULL,
STRINGThreshold = 0, FilterShortPathOptions = NULL, FilterSIMoNeOptions = NULL,
Factors = NULL, ...)
```

Arguments

x	An object of class MultiDEGeneExpr
Methods	A character vector indicating which network construction methods to use, among "STRING", "SIMoNe" and "WGCNA"
STRINGOptions	List with parameters available in the function getSTRINGNet()
SIMoNeOptions	List with parameters available in the function getSIMoNeNet()

`WGCNAOptions` List with parameters available in the function getWGCNANet()

`SelectInteractionsSTRING`
A character vector indicating which interaction sources to select in STRINGNet. Please see selectInteractionTypes() for more details.

`STRINGThreshold`
Confidence score threshold for edge filtering in STRINGNet

`FilterShortPathOptions`
List with parameters available in the function FilterEdges.ShortPathSTRINGNet()

`FilterSIMoNeOptions`
List with parameters available in the function FIlterEdges.SIMoNeNet()

`Factors` A vector of factors attributed to samples. Must gather all samples present in x.

`...` Additional parameters. Not used here.

Value

An object of class MultiNetworks, which is a list of different network objects. If STRING method is used, shortest paths between initial nodes are computed.

See Also

`MultiNetworks`, `print.MultiNetworks`, `MultiDEGeneExpr.default`

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# data(SpASamples)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# StatusFactor<-paste(SpASamples$status,SpASamples$b27,sep=".")
# names(StatusFactor)=SpASamples$chipnum

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# MultiSpAData<-MultiDEGeneExpr(GaussianSpAData,DEGeneExpr(t(SpADataExpression[18:34,]),
# SpADEGenes[18:34,]),DEGeneExpr(t(SpADataExpression[35:51,]),SpADEGenes[35:51,]))
# MultiSpANetworks<-MultiNetworks(MultiSpAData,
# SelectInteractionsSTRING=c("coexpression","experimental","knowledge"),STRINGThreshold=0.9,
# FilterSIMoNeOptions=list(Threshold=0.4),Factors=StatusFactor,
# STRINGOptions=list(AddAnnotations=FALSE),SIMoNeOptions=list(AddAnnotations=FALSE),
# WGCNAOptions=list(AddAnnotations=FALSE))
```

pickSIMoNeParam *Pick SIMoNe parameters*

Description

A function to help in choosing the SIMoNe parameter, and most particularly which model criterion, with a series of plot.

Usage

```
pickSIMoNeParam(DEGeneExpr, ClusterMethod = F, NEdges = NA)
```

Arguments

DEGeneExpr	Object of class DEGeneExpr
ClusterMethod	Boolean variable indicating whether using clustering constraint or not
NEdges	If clustering constraint is used, on which number of edges to do it. If it is set to NA, the function chooses the number of edges by computing the mean between those with maximal AIC and BIC scores from network without clustering constraint.

Details

The series of plots are directly taken from the function plot() of simone package.

Note

A precaution must be taken by choosing the parameters, and the expression data matrix dimensions.

References

Chiquet, J. et al. SIMoNe Statistical Inference for MOdular NEtworks. Bioinforma. Oxf. Engl. 25, 417 (2009).

See Also

SIMoNeNet, SIMoNeNet.default, getSIMoNeNet, print.SIMoNeNet, summary.SIMoNeNet, export.SIMoNeNet, FilterEdges.SIMoNeNet

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])
```

```
# pickSIMoNeParam(GaussianSpADataData)

# GlobalSIMoNeNet<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNet,0.4)
# print(GlobalSIMoNeNet,5)
# summary(GlobalSIMoNeNet)
# plot(GlobalSIMoNeNet)

# export(GlobalSIMoNeNet,"GlobalSIMoNeNet",T)
```

`pickWGCNAParam` *Pick WGCNA parameters*

Description

A function to help in choosing the WGCNA soft threshold parameter parameter, with a series of plot.

Usage

```
pickWGCNAParam(DEGeneExpr, Alphas = c(c(1:10), seq(from = 12, to = 20, by = 2)),
AThreshold = 0.85, interactiveMode = T)
```

Arguments

`DEGeneExpr`	Object of class DEGeneExpr
`Alphas`	The series of soft threshold parameters to test
`AThreshold`	Adjacency threshold to be displayed on plots
`interactiveMode`	Boolean variable indicating whether the plots are in interactive mode. If false, it is useful for automatically saving plots in a single pdf file.

Details

Firstly, this function plots adjacency scores as a function of similarity scores, with different soft threshold parameters. Secondly, it displays network connectivity with different alpha values (with horizontal dashed lines at 0.05, 0.1 and 0.25 of total number of possible edges). In the same way, the next plots display maximal p-values, mean node connectivities, minimal spearman's rhos as a function of alpha values. Next plots show distributions of spearman's rhos (absolute values) and of p-values (logarithm scale). Finally, we can see a plot displaying spearman's p-values as a function of absolute values of rhos, with an horizontal dash line representing a p-value of 0.05 and a vertical one showing the rho for which we are the nearest to a p-value of 0.05.

See Also

`WGCNANet`, `WGCNANet.default`, `getWGCNANet`, `print.WGCNANet`, `summary.WGCNANet`, `export.WGCNANet`, `compareGaussNetworks`

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# pickWGCNAParam(GaussianSpAData)
# GlobalWGCNANet<-getWGCNANet(GaussianSpAData)
# print(GlobalWGCNANet,5)
# summary(GlobalWGCNANet)
# plot(GlobalWGCNANet)
# export(GlobalWGCNANet,"GlobalWGCNANet",T)

# compareGaussNetworks(GlobalSIMoNeNet,GlobalWGCNANet,c("SIMoNe","WGCNA"))
```

---

plot.FactorNetworks *Plot FactorNetworks*

Description

This function plots an object of class FactorNetworks for each level. The same function as in simone package is used.

Usage

```
## S3 method for class 'FactorNetworks'
plot(x, interactiveMode = T, ...)
```

Arguments

x	An object of class FactorNetworks
interactiveMode	Boolean variable indicating whether the plots are in interactive mode. If false, it is useful for automatically saving plots in a single pdf file.
...	Additional parameters from the generic plot function. Not used here.

See Also

FactorNetworks.default

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# data(SpASamples)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# StatusFactor<-paste(SpASamples$status,SpASamples$b27,sep=".")
# names(StatusFactor)=SpASamples$chipnum

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# GlobalSIMoNeNetNF<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNetNF,0.4)

# StatusFactorSIMoNeNet<-FactorNetworks(GaussianSpAData,StatusFactor,"SIMoNe")
# StatusFactorSIMoNeNet<-FilterEdges(StatusFactorSIMoNeNet,0.4)
# plot(StatusFactorSIMoNeNet)
```

`plot.SIMoNeNet` *Plot SIMoNeNet*

Description

This function plots an object of class SIMoNeNet. The same function as in simone package is used.

Usage

```
## S3 method for class 'SIMoNeNet'
plot(x, name = x[["name"]], ...)
```

Arguments

`x`	An object of class SIMoNeNet
`name`	The name to be displayed as title in the plot
`...`	Additional parameters from the generic plot function. Not used here.

See Also

`getSIMoNeNet`

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
```

```
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# pickSIMoNeParam(GaussianSpAData)

# GlobalSIMoNeNet<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNet,0.4)
# print(GlobalSIMoNeNet,5)
# summary(GlobalSIMoNeNet)
# plot(GlobalSIMoNeNet)

# export(GlobalSIMoNeNet,"GlobalSIMoNeNet",T)
```

`plot.WGCNANet` *Plot WGCNANet*

Description

This function plots an object of class WGCNANet. The same function as in simone package is used.

Usage

```
## S3 method for class 'WGCNANet'
plot(x, ...)
```

Arguments

x	An object of class WGCNANet
...	Additional parameters from the generic plot function. Not used here.

See Also

getWGCNANet

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# pickWGCNAParam(GaussianSpAData)
# GlobalWGCNANet<-getWGCNANet(GaussianSpAData)
# print(GlobalWGCNANet,5)
# summary(GlobalWGCNANet)
# plot(GlobalWGCNANet)
# export(GlobalWGCNANet,"GlobalWGCNANet",T)

# compareGaussNetworks(GlobalSIMoNeNet,GlobalWGCNANet,c("SIMoNe","WGCNA"))
```

`resetCytoscapeSession` *Reset Cytoscape session*

Description

This function is useful to reset a Cytoscape session and to be sure that all networks and styles are removed before importing new ones.

Usage

```
resetCytoscapeSession(port.number = 1234)
```

Arguments

`port.number`	The local port number used by cyREST plugin to communicate with Cytoscape. By default it uses 1234.

See Also

`addGraphToCytoscape` `saveCytoscapeSession`

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# data(SpASamples)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# StatusFactor<-paste(SpASamples$status,SpASamples$b27,sep=".")
# names(StatusFactor)=SpASamples$chipnum

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# GlobalSIMoNeNetNF<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNetNF,0.4)

# resetCytoscapeSession()
# addNetworkStyle("SIMoNeNet",class(GlobalSIMoNeNet),points.size.map="P.Value",
# points.fill.map="logFC")
# NetId<-addGraphToCytoscape(GlobalSIMoNeNet)
```

saveCytoscapeSession *Save Cytoscape session*

Description

This function allows to save all networks and styles from a Cytoscape session in a file.

Usage

```
saveCytoscapeSession(filepath = "stringgaussnet_networks", overwrite = F, absolute = F,
port.number = 1234)
```

Arguments

filepath	Where will be saved the Cytoscape session
overwrite	A boolean variable indicating whether the file must be overwritten
absolute	A boolean variable indicating whether filepath is an absolute path. If not, the R work directory is added before filepath.
port.number	The local port number used by cyREST plugin to communicate with Cytoscape. By default it uses 1234.

Details

The file extension .cys is automatically added in the file path. The variable absolute is important because the work directories for cyREST and R are not the same.

See Also

addGraphToCytoscape

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# data(SpASamples)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# StatusFactor<-paste(SpASamples$status,SpASamples$b27,sep=".")
# names(StatusFactor)=SpASamples$chipnum

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# GlobalSIMoNeNetNF<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNetNF,0.4)

# resetCytoscapeSession()
# addNetworkStyle("SIMoNeNet",class(GlobalSIMoNeNet),points.size.map="P.Value",
```

```
# points.fill.map="logFC")
# NetId<-addGraphToCytoscape(GlobalSIMoNeNet)
# saveCytoscapeSession("SIMoNeNet")
```

`selectInteractionTypes` *Select interaction sources in STRINGNet object*

Description

This function allows to select specific interaction sources in an object of class STRINGNet, for example coexpression or experimental. This is also possible to filter on confidence score with this function.

Usage

```
selectInteractionTypes(Network, InteractionTypes = "All", Threshold = 0)
```

Arguments

`Network`	Object of class STRINGNet
`InteractionTypes`	Character vector indicating which interaction sources you are looking for. See details for possible values. If "All", no selection is made and only filtering on Threshold is processed.
`Threshold`	Numeric. Minimum threshold of confident score for selecting edges.

Details

Interaction sources can be coexpression, cooccurence, experimental, knowledge, neighborhood or textmining. Search for STRING DB help page to know what mean those interaction sources.

Value

A new object of class STRINGNet after edge filtering.

See Also

`STRINGNet`, `STRINGNet.default`, `getSTRINGNet`, `print.STRINGNet`, `summary.STRINGNet`, `export.STRINGNet`

Examples

```
data(SpADataExpression)
data(SpADEGenes)
SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# SpASTRINGNet<-getSTRINGNet(SpAData)
# Can be longer.
```

```
# SpASTRINGNet<-getSTRINGNet(SpAData,AddAnnotations=FALSE)
# print(SpASTRINGNet,5)
# summary(SpASTRINGNet)
# PPISpASTRINGNet<-selectInteractionTypes(SpASTRINGNet,
# c("coexpression","experimental","knowledge"),0.9)

# shortPathSpANet<-getShortestPaths(PPISpASTRINGNet)
# shortPathSpANet<-FilterEdges(shortPathSpANet,2.2)
# print(shortPathSpANet,5)
# summary(shortPathSpANet)
```

`ShortPathSTRINGNet.default`

Function to create object of class ShortPathSTRINGNet

Description

This function is used by getShorthestPaths() to convert results from the computation of shortest paths from STRING network.

Usage

```
## Default S3 method:
ShortPathSTRINGNet(x, DEGenes, GenesAnnotations = NULL, ...)
```

Arguments

`x`	The non-formatted shortest paths STRING network obtained by getShortestPaths.
`DEGenes`	DE genes analysis results, which are used for primary node attributes.
`GenesAnnotations`	Gene annotations got by biomaRt if it was requested by getSTRINGNetwork(). Those will be used as secondary node attributes.
`...`	Additional parameters. Not used here.

Value

A list with at least two data frames: - Edge attributes, with distances, intermediate nodes separated by commas, and number of intermediate nodes. - Node attributes given by DE genes analysis results. A third data frame giving gene annotations can be added if it is not null when calling the function.

See Also

`ShortPathSTRINGNet`, `getShortestPaths`, `print.ShortPathSTRINGNet`, `summary.ShortPathSTRINGNet`, `export.ShortPathSTRINGNet`

Examples

```
data(SpADataExpression)
data(SpADEGenes)
SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# SpASTRINGNet<-getSTRINGNet(SpAData)
# Can be longer.

# SpASTRINGNet<-getSTRINGNet(SpAData,AddAnnotations=FALSE)
# print(SpASTRINGNet,5)
# summary(SpASTRINGNet)
# PPISpASTRINGNet<-selectInteractionTypes(SpASTRINGNet,
# c("coexpression","experimental","knowledge"),0.9)

# shortPathSpANet<-getShortestPaths(PPISpASTRINGNet)
# shortPathSpANet<-FilterEdges(shortPathSpANet,2.2)
# print(shortPathSpANet,5)
# summary(shortPathSpANet)
```

SIMoNeNet.default *Function to create object of class SIMoneNet*

Description

This function is used by getSIMoNeNet() to convert results from the SIMoNe inference.

Usage

```
## Default S3 method:
SIMoNeNet(x, DEGeneExpr, GenesAnnotations = NULL, ...)
```

Arguments

x	The non-formatted SIMoNe network obtained by getSIMoNeNetwork
DEGeneExpr	DE genes analysis results contained in an object of class DEGeneExpr. Those will be used as primary node attributes.
GenesAnnotations	Gene annotations got by biomaRt if it was requested by getSIMoNeNet(). Those will be used as secondary node attributes.
...	Additional parameters. Not used here.

Value

A list with at least two data frames: - Edge attributes, with theta scores, spearman's rhos and p-values. - Node attributes given by DE genes analysis results. A third data frame giving gene annotations can be added if it is not null when calling the function.

See Also

SIMoNeNet, getSIMoNeNet, print.SIMoNeNet, summary.SIMoNeNet, export.SIMoNeNet, FilterEdges.SIMoNeNet, pickSIMoNeParam

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# pickSIMoNeParam(GaussianSpAData)

# GlobalSIMoNeNet<-getSIMoNeNet(GaussianSpAData)
# GlobalSIMoNeNet<-FilterEdges(GlobalSIMoNeNet,0.4)
# print(GlobalSIMoNeNet,5)
# summary(GlobalSIMoNeNet)
# plot(GlobalSIMoNeNet)

# export(GlobalSIMoNeNet,"GlobalSIMoNeNet",T)
```

STRINGNet.default *Function to create an object of class STRINGNet*

Description

This function is used by getSTRINGNet() to convert results from the STRING API into object of class STRINGNet.

Usage

```
## Default S3 method:
STRINGNet(x, DEGeneExpr, GenesAnnotations = NULL, ...)
```

Arguments

x	The non-formatted STRING network obtained by getSTRINGNet.
DEGeneExpr	DE genes analysis results contained in an object of class DEGeneExpr. Those will be used as primary node attributes.
GenesAnnotations	Gene annotations got by biomaRt if it was requested by getSTRINGNet(). Those will be used as secondary node attributes.
...	Additional parameters. Not used here.

Value

A list with at least two data frames: - Edge attributes, with confidence scores given by STRING and multiple edges. - Node attributes given by DE genes analysis results. A third data frame giving gene annotations can be added if it is not null when calling the function.

See Also

STRINGNet, getSTRINGNet, print.STRINGNet, summary.STRINGNet, export.STRINGNet

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# SpASTRINGNet<-getSTRINGNet(SpAData)
# Can be longer.

# SpASTRINGNet<-getSTRINGNet(SpAData,AddAnnotations=FALSE)
# print(SpASTRINGNet,5)
# summary(SpASTRINGNet)
# PPISpASTRINGNet<-selectInteractionTypes(SpASTRINGNet,
# c("coexpression","experimental","knowledge"),0.9)

# shortPathSpANet<-getShortestPaths(PPISpASTRINGNet)
# shortPathSpANet<-FilterEdges(shortPathSpANet,2.2)
# print(shortPathSpANet,5)
# summary(shortPathSpANet)
```

WGCNANet.default *Function to create an object of class WGCNANet*

Description

This function is used by getWGCNANet() to convert results from the WGCNA inference.

Usage

```
## Default S3 method:
WGCNANet(x, SoftThreshold, AThreshold, Correlations, PValues, DEGeneExpr,
GenesAnnotations = NULL, ...)
```

Arguments

x	Computed adjacency matrix by getWGCNANet()
SoftThreshold	Soft threshold parameter (alpha) used for adjacency computation by sigmoid function
AThreshold	Threshold on adjacency score for edges inference

Correlations	Correlations (spearman's rho) matrix between all pairs of genes
PValues	Spearman's p-value computed between all pairs of genes
DEGeneExpr	DE genes analysis results contained in an object of class DEGeneExpr. Those will be used as primary node attributes.
GenesAnnotations	Gene annotations got by biomaRt if it was requested by getSIMoNeNet(). Those will be used as secondary node attributes.
...	Additional parameters. Not used here.

Value

A list with at least two data frames: - Edge attributes, with spearman's rhos and p-values. - Node attributes given by DE genes analysis results. A third data frame giving gene annotations can be added if it is not null when calling the function.

See Also

WGCNANet, getWGCNANet, print.WGCNANet, summary.WGCNANet, export.WGCNANet, pickWGCNAParam, compareGaussNetworks

Examples

```
# data(SpADataExpression)
# data(SpADEGenes)
# SpAData<-DEGeneExpr(t(SpADataExpression),SpADEGenes)

# NodesForSIMoNe<-rownames(SpADEGenes)[1:17]
# GaussianSpAData<-DEGeneExpr(t(SpADataExpression[NodesForSIMoNe,]),SpADEGenes[NodesForSIMoNe,])

# pickWGCNAParam(GaussianSpAData)
# GlobalWGCNANet<-getWGCNANet(GaussianSpAData)
# print(GlobalWGCNANet,5)
# summary(GlobalWGCNANet)
# plot(GlobalWGCNANet)
# export(GlobalWGCNANet,"GlobalWGCNANet",T)

# compareGaussNetworks(GlobalSIMoNeNet,GlobalWGCNANet,c("SIMoNe","WGCNA"))
```

Index