Package ‘CoNI’

January 20, 2025

Type Package

Title Correlation Guided Network Integration (CoNI)

Version 0.1.0

Date 2021-09-28

Description Integrates two numerical omics data sets from the same samples using partial correlations. The output can be represented as a network, bipartite graph or a hypergraph structure. The method used in the package refers to Klaus et al (2021) <doi:10.1016/j.molmet.2021.101295>.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 4.0)

RoxygenNote 7.1.1

Imports igraph (>= 1.2.6), doParallel (>= 1.0.16), cocor (>= 1.1.3), ggplot2 (>= 3.3.3), forcats (>= 0.5.1), dplyr (>= 1.0.5), data.table (>= 1.13.7), tibble (>= 3.1.0), foreach (>= 1.5.1), genefilter (>= 1.72.1), ggrepel (>= 0.9.1), gplots (>= 3.1.1), gridExtra (>= 2.3), plyr (>= 1.8.6), ppcor (>= 1.1), tidyr (>= 1.1.3), Hmisc (>= 4.4.2), methods (>= 4.0.3), rlang (>= 0.4.10), tidyselect (>= 1.1.0)

Suggests kableExtra (>= 1.3.2), knitr (>= 1.31), rmarkdown (>= 2.6)

VignetteBuilder knitr

SystemRequirements python3

NeedsCompilation no

Author José Manuel Monroy Kuhn [aut, cre],
Dominik Lutter [ths],
Valentina Klaus [ctb]

Maintainer José Manuel Monroy Kuhn <nolozz@gmail.com>

Repository CRAN

Date/Publication 2021-09-30 09:10:02 UTC

Contents

assign_colorsAnnotation

Assing Colors to Class

Description

This function assigns two color columns (color name and rgb) to an annotation data frame according to a column named 'Class' or 'class'

Usage

```
assign_colorsAnnotation(AnnotationDf, col = "Class")
```

Arguments

`AnnotationDf`	Annotation data frame that contains a factor variable to use to assign colors
`col`	Column with factor variable that will be used to assign colors

Value

The input data.frame with two extra columns specifying the colors for all vertexes according to their respective vertex-class

`Chow_GeneExpData` *Chow gene expression data*

Description

Chow gene expression data

Author(s)

Jose Manuel Monroy <`nolozz@gmail.com`>

References

doi: 10.1016/j.molmet.2021.101295

`Chow_MetaboliteData` *Chow metabolite data*

Description

Chow metabolite data

Author(s)

Jose Manuel Monroy <`nolozz@gmail.com`>

References

doi: 10.1016/j.molmet.2021.101295

`Compare_Triplets` *Compare triplets*

Description

Compare vertexFeature-vertexFeature-edgeFeature between two treatments, that is, find the shared triplets between two different CoNI runs.

Usage

```
Compare_Triplets(
  Treat1_path,
  Treat2_path,
  OutputName = "Shared_Genes_and_Edges_Treat1vsTreat2.csv"
)
```

Arguments

`Treat1_path`	TableForNetwork_file1 (file generated by CoNI) with path for Treatment 1
`Treat2_path`	TableForNetwork_file2 (file generated by CoNI) with path for Treatment 2
`OutputName`	Output file name with path

Value

A data.frame with the shared triplets (vertex1 vertex2 edge_feature) between two CoNI runs

Examples

```
#For an example see the vignette
```

`Compare_VertexClasses_sharedEdgeFeatures` *Table VertexClass pairs of shared Edge Features*

Description

Compare VertexClass pairs of the shared Edge Features of two treatments (e.g., lipid-class-pairs per shared gene)

Usage

```
Compare_VertexClasses_sharedEdgeFeatures(
  Treat1_path,
  Treat2_path,
  OutputName = "Shared_Genes_and_Edges_Treat1vsTreat2.csv",
  Treat1Name = "Treat1",
  Treat2Name = "Treat2"
)
```

Arguments

`Treat1_path`	TableForNetwork_file (file generated by CoNI) with path of Treatment 1
`Treat2_path`	TableForNetwork_file (file generated by CoNI) with path of Treatment 2
`OutputName`	Output file name with path
`Treat1Name`	Name of treatment one, default Treat1
`Treat2Name`	Name of treatment one, default Treat2

Value

A data.frame with all possible vertex-class pairs and their numbers per edge-feature and treatment.

Examples

```
#For an example see the vignette
```

`CoNI` *Correlation guided Network Integration*

Description

CoNI is the main function of Correlation guided Network Integration (CoNI). Input data should come from two sources (e.g., gene expression and metabolite expression), and it should come from the same samples. It calculates all pairwise correlations of the second data input elements and the partial correlations of these pairwise elements with respect to the elements of the first data input. Both data inputs can be prefiltered to include only those elements that significantly correlate. The first data input can be prefiltered to keep just low variance elements (var<0.5). A Steiger test is used to identify significant changes between the correlation and partial correlation values. Results can be visually represented in a Network.

Usage

```
CoNI(
  edgeD,
  vertexD,
  outputDir = "./CoNIOutput/",
  saveRaw = TRUE,
  outputNameRaw = "CoNIOutput",
  onlySgRes = FALSE,
  multipleTAdj = TRUE,
  padjustvertexD = TRUE,
  correlateDFs = TRUE,
  filter_highVarianceEdge = TRUE,
  splitedgeD = TRUE,
  split_number = 2,
  delPrevious = FALSE,
  delIntermediaryFiles = TRUE,
  iteration_start = 1,
  numCores = NULL,
  verbose = TRUE,
  more_coef = FALSE,
  edgeDname = "edgeD",
  vertexDname = "vertexD",
  saveFiles = TRUE
)
```

Arguments

`edgeD` Object to use as first data input (e.g., protein expression)

`vertexD` Object to use as second data input (e.g., metabolite expression)

`outputDir` Output Directory where results are stored

`saveRaw` logical. If TRUE the raw output of CoNI is saved in the output directory (outputDir)

`outputNameRaw` Name for the raw output file if saved

`onlySgRes` logical. If TRUE CoNI output is filtered and only significant results are kept

`multipleTAdj` logical. If TRUE it will filter results after adjustment of multiple testing

`padjustvertexD` logical. If TRUE vertexD is filtered according to the significant adjusted p-value of its pairwise correlations

`correlateDFs` logical. If TRUE the elements that significantly correlate of vertexD are correlated with the elements of edgeD. Only the elements that significantly correlate are kept

`filter_highVarianceEdge` logical. If TRUE features of edgeD with high variance are filtered out

`splitedgeD` logical. If TRUE edgeD will be split in n subsets for the computation (some instances n+1). Keep as TRUE unless the data input is small

`split_number` Number of parts to split the elements of edgeD

`delPrevious` logical. If TRUE previous files of a previous run are deleted

`delIntermediaryFiles` logical. If TRUE the output file of every iteration is deleted and only a single file with all results is kept

`iteration_start` Iteration start for CoNI. Useful if run is interrupted as one can restart from the last iteration

`numCores` Cores assigned for parallelization

`verbose` logical. If TRUE output in the console is more verbose

`more_coef` logical. If TRUE it will include the partial correlation of edge and vertex Features

`edgeDname` File name extension for the edge features that significantly correlate with at least one vertex feature. This file will be read if the function is called again with the same input and with delPrevious=FALSE `vertexDname` File name extension for the vertex features that are involved in at least one significant correlation. This file will be read if the function is called again with the same input and with delPrevious=FALSE

`saveFiles` logical. If FALSE CoNI function will not save any file to disk

Value

CoNI returns a data.frame with the correlation coefficients of the vertex-pairs, the partial correlation coefficients for every triplet, and the pvalue of the Steiger tests

Examples

```
#Run CoNI

#Load gene expression - Toy dataset of two treatments
data(GeneExpToy)
#Samples in rows and genes in columns
GeneExp <- as.data.frame(t(GeneExpToy))
hfd_gene <- GeneExp[1:8,] #high fat diet
chow_gene<- GeneExp[9:nrow(GeneExp),] #chow diet
#Load metabolite expression - Toy dataset of two treatments
data(MetaboExpToy)
MetaboExp <- MetaboExpToy
hfd_metabo <- MetaboExp[11:18,] #high fat diet
chow_metabo <- MetaboExp[1:10,] #chow diet
#Match row names both data sets
rownames(hfd_metabo)<-rownames(hfd_gene)
rownames(chow_metabo)<-rownames(chow_gene)

#Run CoNI with tiny example and no significance testing
#High fat diet
#For big datasets it is recommended to set splitedgeD to TRUE
#and split_number should be adjusted accordingly
#See vignette for an example
#Running CoNI with only a tiny dataset

 CoNIResultsHFD <- CoNI(hfd_gene,hfd_metabo,
                        numCores = 2,
                        onlySgRes = FALSE,
                        filter_highVarianceEdge=FALSE,
                        padjustvertexD = FALSE,
                        correlateDFs = FALSE,
                        edgeDname=”HFD_genes”,
                        vertexDname = ”HFD_metabolites”,
                        saveFiles = FALSE,
                        splitedgeD = FALSE,
                        outputDir = ”./”)
```

`CoNIResultsHFDToy` *Toy data HFD results*

Description

Toy data HFD results

Author(s)

Jose Manuel Monroy `<nolozz@gmail.com>`

References

doi: 10.1016/j.molmet.2021.101295

`CoNIResults_Chow` *CoNI Results Chow*

Description

CoNI Results Chow

Author(s)

Jose Manuel Monroy `<nolozz@gmail.com>`

References

doi: 10.1016/j.molmet.2021.101295

`CoNIResults_HFD` *CoNI Results HFD*

Description

CoNI Results HFD

Author(s)

Jose Manuel Monroy `<nolozz@gmail.com>`

References

doi: 10.1016/j.molmet.2021.101295

`createBipartiteGraph` *Bipartite Network*

Description

This function creates a simple bipartite graph, it shows the driver and linker features as nodes.

Usage

```
createBipartiteGraph(TableNetwork, colorVertexTable, incidenceMatrix = FALSE)
```

Arguments

`TableNetwork` TableForNetwork_file (file generated by CoNI) with path

`colorVertexTable`
Table specifying the colors for the vertex features. The first column should contain the names matching the features of the vertex Data and another column should specify the colors (column name: Colors).

`incidenceMatrix`
logical. If TRUE it returns a hypergraph incidence matrix instead of a bipartite graph

Value

An igraph object for a bipartite graph or a hypergraph incidence matrix to represent ResultsCoNI. Basic network statistics are included in the bipartite graph. See generate_network function for details or consult the igraph package

Examples

```
#See vignette for an example
```

`create_edgeFBarplot` *Vertex-class pairs profile of one shared edge feature*

Description

This function will create a barplot from the output of Compare_VertexClasses_sharedEdgeFeatures for a specific shared Edge Feature (e.g., a shared gene).

Usage

```
create_edgeFBarplot(
  CompTreatTable,
  edgeF,
  treat1 = "Treatment1",
  treat2 = "Treatment2",
  factorOrder = NULL,
  col1 = "red",
  col2 = "blue",
  EdgeFeatureType = "Edge Feature",
  xlb = "Vertex-Class Pairs",
  ylb = "Number of pairs",
  szaxisTxt = 12,
  szaxisTitle = 12
)
```

Arguments

`CompTreatTable` Output of Compare_VertexClasses_sharedEdgeFeatures

`edgeF` Edge feature present in output of Compare_VertexClasses_sharedEdgeFeatures

`treat1` Name of treatment one, default Treatment1. It should match the column names of the output of Compare_VertexClasses_sharedEdgeFeatures

`treat2` Name of treatment one, default Treatment2. It should match the column names of the output of Compare_VertexClasses_sharedEdgeFeatures

`factorOrder` A list specifying the order of the treatments.

`col1` Color for Treatment 1

`col2` Color for Treatment 2

`EdgeFeatureType` Type of Edge Feature (e.g., Gene)

`xlb` Name for x-axis

`ylb` Name for the y-axis

`szaxisTxt` Size axis text

`szaxisTitle` Size axis titles

Value

A ggplot object for a barplot. The barplot shows the vertex-class pairs profile of a single shared edge feature between two treatments

Examples

```
data(VertexClassesSharedGenes_HFDvsChow)
create_edgeFBarplot(CompTreatTable = VertexClassesSharedGenes_HFDvsChow,
                    edgeF = "Lilr4b",
                    treat1 = "HFD",
                    treat2 = "Chow",
                    factorOrder = c("HFD","Chow"),
                    EdgeFeatureType = "Gene")
```

create_GlobalBarplot *Vertex-class pairs profile of shared features*

Description

This function will create a barplot from the output of Compare_VertexClasses_sharedEdgeFeatures using all shared Edge Features (e.g., genes).

Usage

```
create_GlobalBarplot(
  CompTreatTable,
  treat1 = "Treatment1",
  treat2 = "Treatment2",
  factorOrder = NULL,
  col1 = "red",
  col2 = "blue",
  maxpairs = 1,
  xlb = "Vertex-Class Pairs",
  ylb = "Number of pairs",
  szggrepel = 3.5,
  nudgey = 0.5,
  nudgex = 0.5,
  szaxisTxt = 12,
  szaxisTitle = 12
)
```

Arguments

`CompTreatTable`	Output of Compare_VertexClasses_sharedEdgeFeatures
`treat1`	Name of treatment one, default Treatment1. It should match the column names of the output of Compare_VertexClasses_sharedEdgeFeatures
`treat2`	Name of treatment one, default Treatment2. It should match the column names of the output of Compare_VertexClasses_sharedEdgeFeatures
`factorOrder`	A list specifying the order of the treatments.
`col1`	Color for Treatment 1
`col2`	Color for Treatment 2
`maxpairs`	If number of class-vertex-pairs > maxpairs, display number pairs on top of bar
`xlb`	Name for x-axis
`ylb`	Name for the y-axis
`szggrepel`	Size ggrepel labels

nudgey	Nudge y ggrepel
nudgex	Nudge x ggrepel
szaxisTxt	Size axis text
szaxisTitle	Size axis title

Value

A ggplot object for a barplot. The barplot shows the vertex-class pairs profile of all shared edge features between treatments

Examples

```
data(VertexClassesSharedGenes_HFDvsChow)
create_GlobalBarplot(CompTreatTable = VertexClassesSharedGenes_HFDvsChow,
                     treat1 = "HFD",
                     treat2 = "Chow",
                     factorOrder = c("HFD","Chow"),
                     col1="red",
                     col2 ="blue",
                     maxpairs = 1,
                     szggrepel = 6,
                     szaxisTxt = 15,
                     szaxisTitle = 15,
                     xlb = "Metabolite-pair classes")
```

create_stackedGlobalBarplot_perTreatment

Stacked Global Barplot (One treatment)

Description

This function will create a stacked barplot from the output of Compare_VertexClasses_sharedEdgeFeatures using all shared Edge Features (e.g., genes) between two treatments.

Usage

```
create_stackedGlobalBarplot_perTreatment(
  CompTreatTable,
  treat = NULL,
  xlb = "Vertex-Class Pairs",
  ylb = "Number of pairs",
  max_pairsLegend = 2,
  mx.overlaps = Inf,
  szggrepel = 6,
  force = 0.1,
  szTitle = 12,
  szaxisTxt = 12,
  szaxisTitle = 12,
  ylim = NULL
)
```

Arguments

`CompTreatTable`	Output of Compare_VertexClasses_sharedEdgeFeatures
`treat`	Name of treatment to display. It should match the column name in the output of Compare_VertexClasses_sharedEdgeFeatures
`xlb`	Name for x-axis
`ylb`	Name for y-axis
`max_pairsLegend`	If number of Edge Features >= max_pairsLegend, display number of Edge Features as label with ggrepel
`mx.overlaps`	Max number of overlaps ggrepel
`szggrepel`	Size ggrepel labels
`force`	Repelling force for ggrepel labels
`szTitle`	Size Title
`szaxisTxt`	Size axis text
`szaxisTitle`	Size axis titles
`ylim`	Optional y-limits of the plot

Value

A ggplot object to create a stacked barplot. The stacked barplot shows the vertex-class pairs profile of all shared edge features but restricted to a single treatment. Every bar consists of multiple edge features (stacked) that are represented with different colors

Examples

```
data(VertexClassesSharedGenes_HFDvsChow)
create_stackedGlobalBarplot_perTreatment(CompTreatTable = VertexClassesSharedGenes_HFDvsChow,
                                         treat = "HFD",
                                         max_pairsLegend = 9,
                                         xlb = "Metabolite-class-pairs")
```

---

find_localControllingFeatures

Find local controlling features

Description

This function applies for a selected subnetwork a binomial test using the frequency of appearance of an edge feature and the total number of edge features. The probability corresponds to 1/n_df, where n_df corresponds to the total number of edge features in the network. The selected subnetwork corresponds to the second level neighborhood of a specific node. The test is applied to all possible second level neighborhoods in the network.

Usage

```
find_localControllingFeatures(ResultsCoNI, network, padjust = TRUE)
```

Arguments

`ResultsCoNI`	The output of CoNI (after p-adjustment)
`network`	Network created with the function generate_network
`padjust`	logical. Filter output based on adjusted p values

Value

Returns a data.frame with the results of the binomial tests. Significant results correspond to local controlling features

Examples

```
#Load color nodes table
data(MetColorTable)

#Assign colors according to "Class" column
MetColorTable<-assign_colorsAnnotation(MetColorTable)

#Load CoNI results
data(CoNIResultsHFDToy)

#Generate Network
#Note: Colors not visible when ploting in Igraph
HFDNetwork<-generate_network(ResultsCoNI = CoNIResultsHFDToy,
                             colorVertexNetwork = TRUE,
                             colorVertexTable = MetColorTable,
                             Class = MetColorTable,
                             outputDir = "./",
                             outputFileName = "HFD",
                             saveFiles = FALSE)

#Note: For this tiny example nothing is significant
LCG_BinomialTestTableHFD<- find_localControllingFeatures(ResultsCoNI = CoNIResultsHFDToy,
                                                         network = HFDNetwork )
LCGenes_HFD<-as.character(unique(LCG_BinomialTestTableHFD$edgeFeatures))
```

GeneExpToy *Toy data gene expression*

Description

Toy data gene expression

Author(s)

Jose Manuel Monroy <nolozz@gmail.com>

References

doi: 10.1016/j.molmet.2021.101295

generate_network *Create network*

Description

This function creates a network using as input the output of CoNI and a table specifying the colors for the nodes.

Usage

```
generate_network(
  ResultsCoNI,
  colorVertexNetwork = TRUE,
  colorVertexTable,
  outputDir = "./",
  outputFileName = "ResultsCoNI",
  Class = NULL,
  saveFiles = TRUE
)
```

Arguments

`ResultsCoNI` The input of the function are the results of CoNI.

`colorVertexNetwork` logical. If TRUE, the table colorVertexTable has to be provided to specify vertex colors

`colorVertexTable` Table specifying the colors for the nodes (vertex features). The first column should contain the names matching the features of the vertex Data and the colors or other data can be specified in the rest of the columns

`outputDir` Output directory where the network is saved as well as the file that was used to generate the network.

`outputFileName` The name of the file used to create the network.

`Class` Optional data frame with at least two columns, first column contains all vertex features and another column the vertex feature class (column named "Class"). Necessary for treatment comparisons based on class

`saveFiles` logical. If FALSE TableForNetwork_‘outputFileName‘.csv and Network_‘outputFileName‘.graphml are not saved to disk

Value

Returns an igraph object (network) constructed from ResultsCoNI. The network includes the following network statistics

- "degree"The number of the vertex adjacent edges
- "hub_score"The principal eigenvector of A*t(A), where A is the adjacency matrix of the graph
- "transitivity"Probability that the adjacent vertices of a vertex are connected
- "closeness"Steps required to access every other vertex from a given vertex
- "betweenness"(roughly) The number of geodesics (shortest paths) going through a vertex or an edge
- "eigen_centrality"Takes a graph (graph) and returns the eigenvector centralities of positions v within it
- "centralized_betweenness"The vertice-level centrality score according to the betweenness of vertices
- "centralized_closeness"The vertice-level centrality score according to the closeness of vertices
- "centralized_degree"The vertice-level centrality score according to the degrees of vertices

For more details see igraph package

Examples

```
#Generate Network

#Load color nodes table
data(MetColorTable)
#Assign colors according to "Class" column
MetColorTable<-assign_colorsAnnotation(MetColorTable)
#Load CoNI results
data(CoNIResultsHFDToy)

#Generate Network
HFDNetwork<-generate_network(ResultsCoNI = CoNIResultsHFDToy,
                             colorVertexNetwork = TRUE,
                             colorVertexTable = MetColorTable,
                             outputDir = "./",
                             outputFileName = "HFD",
                             saveFiles = FALSE)
```

---

getstackedGlobalBarplot_and_Grid

Stacked Global Barplot Side-by-side (two treatments)

Description

This function will create a stacked barplot from the output of Compare_VertexClasses_sharedEdgeFeatures using all shared Edge Features (e.g., genes) between two treatments. Results of both Treatments are side by side for better comparison.

Usage

```
getstackedGlobalBarplot_and_Grid(
  CompTreatTable,
  Treat1,
  Treat2,
  ggrep = TRUE,
  max_pairsLegend = 1,
  force = 0.1,
  mx.overlaps = Inf,
  szggrepel = 6,
  xlb = "Vertex-Class Pairs",
  ...
)
```

Arguments

`CompTreatTable`	Output of Compare_VertexClasses_sharedEdgeFeatures
`Treat1`	Name treatment 1 as in table CompTreatTable
`Treat2`	Name treatment 2 as in table CompTreatTable
`ggrep`	logical. If TRUE includes ggrepel labels for every bar
`max_pairsLegend`	If number of Edge Features >= max_pairsLegend, display number of Edge Features as ggrepel label
`force`	Repelling force for ggrepel labels
`mx.overlaps`	Max number of overlaps ggrepel
`szggrepel`	Size ggrepel labels
`xlb`	Name for x-axis
`...`	Other parameters for inner functions, mainly ggplot2 visual parameters

Value

A gtable containing stacked barplots. The barplots show the vertex-class pairs profile of all shared edge features between two treatments (one bar plot per treatment). Every bar consists of multiple edge features that are depicted with different colors

Examples

```
data(VertexClassesSharedGenes_HFDvsChow)
VCSGs<-VertexClassesSharedGenes_HFDvsChow
HFD_vs_Chow_stackedBarplot<-getstackedGlobalBarplot_and_Grid(VCSGs,
                                                             Treat1 = "HFD",
                                                             Treat2 = "Chow",
                                                         xlb = "Metabolite-class-pairs",
                                                             max_pairsLegend=9)
plot(HFD_vs_Chow_stackedBarplot)
```

`getVertexsPerEdgeFeature`
Vertex Class profile per edge feature (one treatment)

Description

This function creates a barplot or barplots showing the number of vertex features per class for every shared edge feature between two treatments

Usage

```
getVertexsPerEdgeFeature(
  CompTreatTable,
  Annotation,
  chunks = 5,
  treat = NULL,
  small = FALSE,
  ggrep = TRUE,
  xlb = "Gene",
  onlyTable = FALSE,
  szTitle = 12,
  szaxisTxt = 12,
  szaxisTitle = 12,
  ...
)
```

Arguments

`CompTreatTable`	Output of Compare_VertexClasses_sharedEdgeFeatures
`Annotation`	Data frame that includes the rgb colors for every class. The column 'class' (or 'Class') has to be present and also the column 'ColorRgb'
`chunks`	To avoid a non readable dense plot the results can be spitted in multiple plots
`treat`	Specify the treatment for which the plot will be created. It should be one of the two treatments in the output of Compare_VertexClasses_sharedEdgeFeatures
`small`	logical. If only a few edge features are in the input set as TRUE. A single plot will be created
`ggrep`	logical. If TRUE includes ggrepel labels for every bar
`xlb`	x-axis label
`onlyTable`	logical. If TRUE a table is returned instead of a plot
`szTitle`	Size title
`szaxisTxt`	Size axis text
`szaxisTitle`	Size axis title
`...`	Other parameters for inner functions, mainly ggplot2 visual parameters

Value

A list of ggplot objects to create different barplots. The barplots show the number of vertex features per class for every shared edge feature between two treatments. The barplots restrict to one of the compared treatments. An alternative output is a data.frame with the number of vertex features per class and edge feature (onlyTable=TRUE)

Examples

```
data(VertexClassesSharedGenes_HFDvsChow)
data(MetColorTable)
#Note: No differences in example as all the Output of CoNI was kept
getVertexsPerEdgeFeature(CompTreatTable = VertexClassesSharedGenes_HFDvsChow,
                         Annotation = MetColorTable,
                         chunks = 2,
                         treat = "HFD")
```

getVertexsPerEdgeFeature_and_Grid

Vertex-Class profile per edge feature Side-by-Side (two treatments)

Description

This function creates a grid of barplots. The barplot of one side depicts the number of class vertex features per edge feature for treatment 1 and the other side the same barplot for treatment 2. Results of both Treatments are side by side for better comparison.

Usage

```
getVertexsPerEdgeFeature_and_Grid(
  CompTreatTable,
  Treat1,
  Treat2,
  Annotation,
  chunks = 3,
  ggrep = TRUE,
  xlb = "Edge Feature",
  onlyT = FALSE,
  small = FALSE,
  ...
)
```

Arguments

`CompTreatTable`	Output of Compare_VertexClasses_sharedEdgeFeatures
`Treat1`	Name treatment 1 as in table CompTreatTable
`Treat2`	Name treatment 2 as in table CompTreatTable

`Annotation` Data frame that includes the rgb colors for every class. The column 'class' (or 'Class') has to be present and also the column 'ColorRgb'

`chunks` To avoid a non readable dense plot the results can be spitted in multiple plots

`ggrep` logical. If TRUE includes ggrepel labels for every bar

`xlb` Change the x-axis label

`onlyT` logical. If TRUE a table is returned instead of a grid of plots

`small` logical. If only a few edge features are in the input set as TRUE. A single plot will be created

`...` Other parameters for inner functions, mainly ggplot2 visual parameters

Value

A gtable containing side-by-side barplots, one for each treatment, showing the number of vertex features per class for every shared edge feature

Examples

```
data(VertexClassesSharedGenes_HFDvsChow)
VCSGs<-VertexClassesSharedGenes_HFDvsChow
data(MetColorTable)
HFD_vs_Chow_LCP_Gene<-getVertexsPerEdgeFeature_and_Grid(VCSGs,
                                                        "HFD","Chow",
                                                        Annotation=MetColorTable,
                                                        ggrep=FALSE,
                                                        small = FALSE,
                                                        chunks = 3,
                                                        szLegendKey=0.2)
plot(HFD_vs_Chow_LCP_Gene)
```

`HFD_GeneExpData` *HFD gene expression data*

Description

HFD gene expression data

Author(s)

Jose Manuel Monroy <`nolozz@gmail.com`>

References

doi: 10.1016/j.molmet.2021.101295

`HFD_MetaboliteData` *HFD metabolite data*

Description

HFD metabolite data

Author(s)

Jose Manuel Monroy <`nolozz@gmail.com`>

References

doi: 10.1016/j.molmet.2021.101295

`MetaboExpToy` *Toy data metabolite expression*

Description

Toy data metabolite expression

Author(s)

Jose Manuel Monroy <`nolozz@gmail.com`>

References

doi: 10.1016/j.molmet.2021.101295

`MetaboliteAnnotation` *Metabolite Annotation*

Description

Metabolite Annotation

Author(s)

Jose Manuel Monroy <`nolozz@gmail.com`>

References

doi: 10.1016/j.molmet.2021.101295

MetColorTable *Toy data annotation*

Description

Toy data annotation

Author(s)

Jose Manuel Monroy <nolozz@gmail.com>

References

doi: 10.1016/j.molmet.2021.101295

NetStats *Network Statistics*

Description

This function calculates simple network statistics and returns them as a dataframe

Usage

```
NetStats(Network)
```

Arguments

Network	An Igraph network

Value

Returns a data.frame with nine rows with the following network statistics:

- "net_avg_pathL"Shortest paths between vertices
- "net_edge_density"Graph density, ratio of the number of edges and the number of possible edges
- "net_transitivity"Probability that the adjacent vertices of a vertex are connected
- "net_diameter"Length of the longest geodesic
- "net_nodes_first_path_diameter"The nodes along the first found path with the length of diameter
- "net_eigenvalue"The eigenvalue corresponding to the centrality scores.
- "net_centralized_betweenessIdx"The graph level centrality index after centralizing the graph according to the betweenness of vertices

- "net_centralized_closenessIdx"The graph level centrality index after centralizing the graph according to the closeness of vertices
- "net_centralized_degreeIdx"The graph level centrality index after centralizing the graph according to the degrees of vertices

For more information on the statistics consult the igraph package.

Examples

```
#Load color nodes table
data(MetColorTable)
#Assign colors according to "Class" column
MetColorTable<-assign_colorsAnnotation(MetColorTable)
#Load CoNI results
data(CoNIResultsHFDToy)

#Generate Network
HFDNetwork<-generate_network(ResultsCoNI = CoNIResultsHFDToy,
                             colorVertexNetwork = TRUE,
                             colorVertexTable = MetColorTable,
                             outputDir = "./",
                             outputFileName = "HFD",
                             saveFiles = FALSE)
NetStats(HFDNetwork)
```

plotPcorvsCor *Correlation vs Partial correlation*

Description

This function fits two linear models on standardize data and plots the results. It generates a scatter plot with two regression lines, where the slopes correspond to the correlation and partial correlation coefficients (blue for cor and red for pcor)

Usage

```
plotPcorvsCor(
  ResultsCoNI,
  edgeFeature,
  vertexD,
  edgeD,
  vertexFeatures = NULL,
  outputDir = "./",
  fname,
  label_edgeFeature = "Edge Feature",
  plot_to_screen = TRUE,
  height = 10,
  width = 8,
  saveFiles = FALSE
)
```

Arguments

`ResultsCoNI`	The significant results generated by CoNI
`edgeFeature`	The edge feature to explore e.g. Fabp2 (for a gene)
`vertexD`	Vertex data that was given as input to CoNI
`edgeD`	Edge data that was given as input to CoNI
`vertexFeatures`	The vertex features to include as a list. If not specified all metabolites available in combination with the edgeFeature will be used
`outputDir`	Output directory with path
`fname`	File name to save the plots
`label_edgeFeature`	Name for plot title e.g. Gene or Protein
`plot_to_screen`	logical. If TRUE plots will be outputted to the plotting screen
`height`	height of the plotting area for the saved file
`width`	width of the plotting are for the saved file
`saveFiles`	logical. If FALSE plot is not saved to disk

Value

Returns a ggplot object for a scatter plot with two regression lines. The blue line is the regression of the vertex features, and the red line is the regression of the resulting residuals after regressing each vertex feature with the edge feature. The slope of the blue line corresponds to the pearson correlation coefficient and the slope of the red line to the partial correlation coefficient

Examples

```
#Load gene expression - Toy dataset of two treatments
data(GeneExpToy)
#Samples in rows and genes in columns
GeneExp <- as.data.frame(t(GeneExpToy))
hfd_gene <- GeneExp[1:8,] #high fat diet
chow_gene<- GeneExp[9:nrow(GeneExp),] #chow diet

#Load metabolite expression - Toy dataset of two treatments
data(MetaboExpToy)
MetaboExp <- MetaboExpToy
hfd_metabo <- MetaboExp[11:18,] #high fat diet
chow_metabo <- MetaboExp[1:10,] #chow diet

#Match row names both data sets
rownames(hfd_metabo)<-rownames(hfd_gene)
rownames(chow_metabo)<-rownames(chow_gene)

#Load CoNI results
data(CoNIResultsHFDToy)

plotPcorvsCor(ResultsCoNI = CoNIResultsHFDToy,
              edgeFeature = "Arfrp1",
              vertexFeatures = c("PC.ae.C40.2", "SM..OH..C22.1"),
              vertexD = hfd_metabo,
              edgeD = hfd_gene,
              label_edgeFeature = "Gene",
              plot_to_screen = TRUE,
              height = 10,
              saveFiles = FALSE)
```

tableLCFs_VFs *Table local controlling edge features and vertex pairs*

Description

This function creates a table of the local controlling edge features

Usage

```
tableLCFs_VFs(CoNIResults, LCFs)
```

Arguments

CoNIResults	The output of CoNI (after p-adjustment)
LCFs	Local controlling edge features as a vector

Value

A data.frane of local controlling edge features and their respective vertex pairs, and unique vertexes.

Examples

```
#Load CoNI results
data(CoNIResultsHFDToy)
#Note: arbitrary list of genes, not Local controlling features
tableLCFs_VFs(CoNIResultsHFDToy, c("Lilr4b","Rps12"))
```

top_n_LF_byMagnitude *Linker Features by magnitude of effect*

Description

This function outputs the linker features with the strongest effect on the correlation of the vertex features

Usage

```
top_n_LF_byMagnitude(ResultsCoNI, topn = 10)
```

Arguments

ResultsCoNI	The output of CoNI
topn	Top n number of features to output

Value

Returns a data.frame, a filtered version of ResultsCoNI, showing the top n features with the strongest effect, that is, the highest difference between the partial correlation and correlation coefficient.

Examples

```
data(CoNIResultsHFDToy)
Top10HFD<-top_n_LF_byMagnitude(CoNIResults_HFD,topn = 10)
```

VertexClassesSharedGenes_HFDvsChow

Toy data comparison treatments

Description

Toy data comparison Vertex classes shared edge featuresHFD vs Chow

Author(s)

Jose Manuel Monroy <nolozz@gmail.com>

References

doi: 10.1016/j.molmet.2021.101295

Index