

Package ‘EGAD’

April 14, 2017

Type Package

Title Extending guilt by association by degree

Version 1.2.0

Date 2016-04-20

Description The package implements a series of highly efficient tools to calculate functional properties of networks based on guilt by association methods.

License GPL-2

Depends R(>= 3.3)

Imports gplots, Biobase, GEOquery, limma, arrayQualityMetrics, impute, RColorBrewer, zoo, igraph, plyr, Matrix, MASS, RCurl, affy

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

RoxygenNote 5.0.1

LazyData true

biocViews Software, FunctionalGenomics, SystemsBiology, GenePrediction, FunctionalPrediction, NetworkEnrichment, GraphAndNetwork, Network

Author Sara Ballouz [aut, cre], Melanie Weber [aut, ctb], Paul Pavlidis [aut], Jesse Gillis [aut, ctb]

Maintainer Sara Ballouz <sballouz@cshl.edu>

NeedsCompilation no

R topics documented:

| | |
|-------------------------------------|---|
| assortativity | 3 |
| attr.human | 3 |
| attr.mouse | 4 |
| auc_multifunc | 4 |
| auprc | 5 |
| auroc_analytic | 5 |
| biogrid | 6 |
| build_binary_network | 6 |
| build_coexp_expressionSet | 7 |

| | |
|---|-----------|
| build_coexp_GEOID | 8 |
| build_coexp_network | 8 |
| build_semantic_similarity_network | 9 |
| build_weighted_network | 9 |
| calculate_multifunc | 10 |
| conv_smoother | 11 |
| example_annotations | 11 |
| example_binary_network | 11 |
| example_coexpression | 12 |
| example_neighbor_voting | 12 |
| extend_network | 12 |
| filter_network | 13 |
| filter_network_cols | 13 |
| filter_network_rows | 14 |
| filter_orthologs | 15 |
| fmeasure | 16 |
| genes | 16 |
| get_auc | 17 |
| get_biogrid | 17 |
| get_counts | 18 |
| get_density | 18 |
| get_expression_data_gemma | 19 |
| get_expression_matrix_from_GEO | 19 |
| get_phenocarta | 20 |
| get_prc | 20 |
| get_roc | 21 |
| GO.human | 21 |
| GO.mouse | 22 |
| GO.voc | 22 |
| make_annotations | 23 |
| make_genelist | 23 |
| make_gene_network | 24 |
| make_transparent | 25 |
| neighbor_voting | 25 |
| node_degree | 26 |
| ortho | 27 |
| pheno | 27 |
| plot_densities | 28 |
| plot_density_compare | 28 |
| plot_distribution | 29 |
| plot_network_heatmap | 30 |
| plot_prc | 30 |
| plot_roc | 31 |
| plot_roc_overlay | 32 |
| plot_value_compare | 32 |
| predictions | 33 |
| repmat | 34 |
| run_GBA | 34 |
| Index | 36 |

| | |
|---------------|--|
| assortativity | <i>Calculating network assortativity</i> |
|---------------|--|

Description

The function calculates the assortativity of a network, that measures the preference of interactions between similar nodes. As in most literature, 'similarity' is here defined in terms of node degrees.

Usage

```
assortativity(network)
```

Arguments

network matrix indicating network structure (symmetric)

Value

Numeric value

Examples

```
network <- matrix( sample(c(0,1),36, replace=TRUE), nrow=6,byrow=TRUE)
assort_value <- assortativity(network)
```

| | |
|------------|--|
| attr.human | <i>Human GENCODE annotations (v22)</i> |
|------------|--|

Description

A dataset containing identifiers for gene transcripts

Format

A data frame with 60483 rows and 10 variables:

chr chromosome

start chromosomal start position, in base pairs

end chromosomal end position, in base pairs

strand chromosomal strand, + or -

un unknown

ensemblID ENSEMBL identifier

type type of transcript

stat status of transcript

name HUGO identifier

entrezID Entrez identifier

@source ftp://ftp.sanger.ac.uk/pub/genencode/Gencode_human/release_22/

attr.mouse

Mouse GENCODE annotations (M7)

Description

A dataset containing identifiers for gene transcripts

Format

A data frame with 46517 rows and 10 variables:

chr chromosome

start chromosomal start position, in base pairs

end chromosomal end position, in base pairs

strand chromosomal strand, + or -

un unknown

ensemblID ENSEMBL identifier

type type of transcript

stat status of transcript

name HUGO identifier

entrezID Entrez identifier

@source ftp://ftp.sanger.ac.uk/pub/gencode/Gencode_mouse/release_M7/

auc_multifunc

Calculating AUC for functional groups from ranked lists

Description

The function calculates the AUC for a functional group analytically using an optimal ranked list of genes that indicates association between genes and groups.

Usage

```
auc_multifunc(annotations, optimallist)
```

Arguments

annotations binary matrix indicating which list elements are in which functional groups.

optimallist Ranked list (multifunctionality analysis, see [calculate_multifunc](#)).

Value

aucs array of aucs for each group in annotations

Examples

```
annotations <- c(rep(0,10))
annotations[c(1,3,5)] <- 1
optimallist <- 10:1
aurocs_mf <- auc_multifunc(annotations, optimallist)
```

| | |
|-------|--|
| auprc | <i>Area under the precision recall curve</i> |
|-------|--|

Description

The function calculates the area under the precision-recall curve

Usage

```
auprc(scores, labels)
```

Arguments

| | |
|--------|---------------|
| scores | numeric array |
| labels | binary array |

Value

auprc Numeric value

Examples

```
labels <- c(rep(0,10))
labels[c(1,3,5)] <- 1
scores <- 10:1
auprc <- auprc(scores, labels)
```

| | |
|----------------|---|
| auroc_analytic | <i>Area under the receiver operating characteristic curve</i> |
|----------------|---|

Description

The function calculates the area under the receiver operating characteristic (ROC) curve analytically

Usage

```
auroc_analytic(scores, labels)
```

Arguments

| | |
|--------|---------------|
| scores | numeric array |
| labels | binary array |

Value

auroc Numeric value

Examples

```
labels <- c(rep(0,10))
labels[c(1,3,5)] <- 1
scores <- 10:1
auroc <- auroc_analytic(scores, labels)
```

biogrid

BIOGRID v3.4.126

Description

A data frame containing protein-protein interactions

Format

A data frame with 211506 rows and 2 variables:

entrezID_A List of Entrez identifiers, interactor A

entrezID_B List of Entrez identifiers, interactor B

@source <http://thebiogrid.org/>

build_binary_network *Builds a binary network*

Description

The function creates a gene-by-gene matrix with binary entries indicating interaction (1) or no interaction (0) between the genes.

Usage

```
build_binary_network(data, list)
```

Arguments

data 2-column matrix, each row a pair indicating a relationship or interaction

list string array of genes/labels/ids

Value

net matrix binary characterizing interactions

Examples

```
data <- cbind(edgeA=c('gene1', 'gene2'), edgeB=c('gene3', 'gene3'))
list <- c('gene1', 'gene2', 'gene3')
network <- build_binary_network(data, list)
```

build_coexp_expressionSet

Builds a coexpression network from an expressionSet

Description

The function generates a dense coexpression network from expression data stored in the expressionSet data type. Correlation coefficients are used as to weight the edges of the nodes (genes). Calls [build_coexp_network](#).

Usage

```
build_coexp_expressionSet(exprsSet, gene.list, method = "spearman",
  flag = "rank")
```

Arguments

| | |
|-----------|--|
| exprsSet | data class ExpressionSet |
| gene.list | array of gene labels |
| method | correlation method to use, default Spearman's rho |
| flag | string to indicate if the network should be ranked |

Value

net Matrix symmetric

Examples

```
exprs <- matrix( rnorm(1000), ncol=10, byrow=TRUE)
gene.list <- paste('gene', 1:100, sep='')
sample.list <- paste('sample', 1:10, sep='')
rownames(exprs) <- gene.list
colnames(exprs) <- sample.list
network <- build_coexp_expressionSet(exprs, gene.list, method='pearson')
```

build_coexp_GEOID *Builds a coexpression network given a GEO ID*

Description

The function generates a dense coexpression network from expression data stored in GEO. The expression data is downloaded from GEO. Correlation coefficients are used as to weight the edges of the nodes (genes). Calls [get_expression_matrix_from_GEO](#) and [build_coexp_network](#).

Usage

```
build_coexp_GEOID(gseid, gene.list, method = "spearman", flag = "rank")
```

Arguments

| | |
|-----------|--|
| gseid | string GEO ID of expression experiment |
| gene.list | array of gene labels |
| method | correlation method to use, default Spearman's rho |
| flag | string to indicate if the network should be ranked |

Value

net Matrix symmetric

build_coexp_network *Builds a coexpression network from an expressionSet*

Description

The function generates a dense coexpression network from expression data stored as a matrix, with the genes as row labels, and samples as column labels. Correlation coefficients are used as to weight the edges of the nodes (genes). Calls [cor](#).

Usage

```
build_coexp_network(exprs, gene.list, method = "spearman", flag = "rank")
```

Arguments

| | |
|-----------|--|
| exprs | matrix of expression data |
| gene.list | array of gene labels |
| method | correlation method to use, default Spearman's rho |
| flag | string to indicate if the network should be ranked |

Value

net Matrix symmetric

Examples

```
exprs <- matrix( rnorm(1000), ncol=10,byrow=TRUE)
gene.list <- paste('gene',1:100, sep='')
sample.list <- paste('sample',1:10, sep='')
rownames(exprs) <- gene.list
colnames(exprs) <- sample.list
network <- build_coexp_network(exprs, gene.list)
```

build_semantic_similarity_network

Builds a semantic similarity network

Description

The function builds a semantic similarity network given a data and labels

Usage

```
build_semantic_similarity_network(genes.labels, genes)
```

Arguments

| | |
|--------------|---|
| genes.labels | matrix with rows as genes and columns as a function/label |
| genes | array of gene IDs |

Value

net Numeric value

Examples

```
genes.labels <- matrix( sample(c(0,1), 100, replace=TRUE),ncol=10,nrow=10)
rownames(genes.labels) <- 1:10
genes <- 1:10
net <- build_semantic_similarity_network(genes.labels, genes)
```

build_weighted_network

Builds a weighted network

Description

The function creates a gene-by-gene matrix with binary entries indicating interaction (1) or no interaction (0) between the genes.

Usage

```
build_weighted_network(data, list)
```

Arguments

`data` 3-column matrix, each row a pair indicating a relationship or interaction, and the last column the weight

`list` string array of genes/labels/ids

Value

net matrix characterizing interactions

Examples

```
data <- cbind(edgeA=c('gene1','gene2'),edgeB=c('gene3','gene3'), weight=c(0.5, 0.9))
list <- c('gene1','gene2','gene3')
network <- build_weighted_network(data,list)
```

calculate_multifunc *Performing multifunctionality analysis*

Description

The function performs multifunctionality analysis ([1]) for a set of annotated genes and creates a rank based optimallist. For annotations use an ontology that is large enough to serve as a prior (e.g. GO, Phenocarta).

Usage

```
calculate_multifunc(genes.labels)
```

Arguments

`genes.labels` Annotation matrix

Value

`gene.mfs` Returns matrix with evaluation of gene function prediction by given labels:

Examples

```
genes.labels <- matrix( sample(c(0,1), 100, replace=TRUE),ncol=10,nrow=10)
rownames(genes.labels) = paste('gene', 1:10, sep='')
colnames(genes.labels) = paste('label', 1:10, sep='')
mf <- calculate_multifunc(genes.labels)
```

| | |
|---------------|----------------------------|
| conv_smoother | <i>Plot smoothed curve</i> |
|---------------|----------------------------|

Description

The function plots a smoothed curve using the `convolve` function.

Usage

```
conv_smoother(X, Y, window, xlab = "", ylab = "", raw = FALSE)
```

Arguments

| | |
|--------|--|
| X | numeric array |
| Y | numeric array |
| window | numeric value indicating size of window to use |
| xlab | string of x-axis label |
| ylab | string of y-axis label |
| raw | boolean |

Value

smoothed X,Y and std Y matrix

Examples

```
x <- 1:1000
y <- rnorm(1000)
conv <- conv_smoother(x,y,10)
```

| | |
|-----------------------|-------------------------------|
| example_annotatations | <i>Example of annotations</i> |
|-----------------------|-------------------------------|

Description

This dataset includes

| | |
|------------------------|----------------------------------|
| example_binary_network | <i>Example of binary network</i> |
|------------------------|----------------------------------|

Description

This dataset includes

Format

Matrices and vectors

example_coexpression *Example of binary network*

Description

This dataset includes

Format

Matrices and vectors

example_neighbor_voting
Example of binary network

Description

This dataset includes

Format

entrezID chromosomal start position, in base pairs
name HUGO gene identifier
species species
disease disease

extend_network *Builds an extended network from a binary network*

Description

The function extends a binary network by using the inverse of the path length between nodes as a weighted edge

Usage

```
extend_network(net, max = 6)
```

Arguments

net matrix binary and symmetric
max numeric maximum number of jumps

Value

ext_net matrix dense and symmetric

Examples

```
net <- matrix( sample(c(0,1),36, replace=TRUE), nrow=6,byrow=TRUE)
ext_net <- extend_network(net)
```

filter_network *Filter on matrix*

Description

The function filters out the rows or columns of a matrix such that the size of the group is exclusively between given min and max values

Usage

```
filter_network(network, flag = 1, min = 0, max = 1, ids = NA)
```

Arguments

| | |
|---------|---|
| network | numeric matrix |
| flag | numeric 1 for row filtering, 2 for column filtering |
| min | numeric value |
| max | numeric value |
| ids | array to filter on |

Value

network numeric matrix

Examples

```
net <- matrix( rnorm(10000), nrow=100)
filt_net <- filter_network(net,1,10,100)
```

filter_network_cols *Filter on columns*

Description

The function filters out the columns of a matrix such that the size of the group is exclusively between given min and max values

Usage

```
filter_network_cols(network, min = 0, max = 1, ids = NA)
```

Arguments

| | |
|---------|----------------|
| network | numeric matrix |
| min | numeric value |
| max | numeric value |
| ids | array |

Value

network numeric matrix

Examples

```
genes.labels <- matrix( sample( c(0,1), 10000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:100, sep='')
genes.labels <- filter_network_cols(genes.labels,50,200)
```

```
genes.labels <- matrix( sample( c(0,1), 10000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:100, sep='')
genes.labels <- filter_network_cols(genes.labels,ids = paste('function', 1:20, sep=''))
```

filter_network_rows *Filter on rows*

Description

The function filters out the rows of a matrix such that the size of the group is exclusively between given min and max values

Usage

```
filter_network_rows(network, min = 0, max = 1, ids = NA)
```

Arguments

| | |
|---------|--------------------|
| network | numeric matrix |
| min | numeric value |
| max | numeric value |
| ids | array to filter on |

Value

network numeric matrix

Examples

```
genes.labels <- matrix( sample( c(0,1), 10000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:100, sep='')
genes.labels <- filter_network_rows(genes.labels,50,200)

genes.labels <- matrix( sample( c(0,1), 10000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:100, sep='')
genes.labels <- filter_network_rows(genes.labels,ids = paste('gene', 1:20, sep=''))
```

| | |
|------------------|----------------------------|
| filter_orthologs | <i>Filter on orthologs</i> |
|------------------|----------------------------|

Description

The function filters away the labels for the genes that are not in the orthologs list

Usage

```
filter_orthologs(annotations, genelists, orthologs)
```

Arguments

| | |
|-------------|--------------------|
| annotations | binary matrix |
| genelists | array of gene ids |
| orthologs | array to filter on |

Value

annotations_filtered binary matrix

Examples

```
genes.labels <- matrix( sample( c(0,1), 1000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:10, sep='')
gene.list <- paste('gene', 1:100, sep='')
orthologs <- paste('gene', (1:50)*2, sep='')
genes.labels.filt <- filter_orthologs(genes.labels, gene.list, orthologs)
```

| | |
|----------|-------------------------------------|
| fmeasure | <i>Fmeasure of precision-recall</i> |
|----------|-------------------------------------|

Description

The function calculates fmeasure for a given beta of a precision-recall curve

Usage

```
fmeasure(recall, precis, beta = 1)
```

Arguments

| | |
|--------|-----------------------------|
| recall | numeric array |
| precis | numeric array |
| beta | numeric value, default is 1 |

Value

fmeasure Numeric value

Examples

```
labels <- c(rep(0,10))
labels[c(1,3,5)] <- 1
scores <- 10:1
prc <- get_prc(scores, labels)
fm <- fmeasure(prc[,1], prc[,2])
```

| | |
|-------|------------------------------------|
| genes | <i>Genes from BIOGRID v3.4.126</i> |
|-------|------------------------------------|

Description

An array containing identifiers for genes in biogrid

Format

Array

genes List of Entrez identifiers

@source <http://thebiogrid.org/>

| | |
|---------|--|
| get_auc | <i>Calculates the area under a curve</i> |
|---------|--|

Description

The function calculates the area under the curve defined by x and y

Usage

```
get_auc(x, y)
```

Arguments

| | |
|---|---------------|
| x | numeric array |
| y | numeric array |

Value

auc numeric value

Examples

```
x <- 1:100  
y <- 1:100  
auc <- get_auc(x,y)
```

| | |
|-------------|--|
| get_biogrid | <i>Downloading and filtering BIOGRID</i> |
|-------------|--|

Description

The function downloads the specified version of biogrid for a particular taxon

Usage

```
get_biogrid(species = "9606", version = "3.4.131",  
            interactions = "physical")
```

Arguments

| | |
|--------------|--|
| species | numeric taxon of species |
| version | string of biogrid version |
| interactions | string stating either physical or genetic interactions |

Value

biogrid data.frame with interactions

`get_counts`*Get counts*

Description

The function formats the count distribution from the histogram function

Usage

```
get_counts(hist)
```

Arguments

hist histogram

Value

x,y

Examples

```
x <- runif(1000)
counts <- get_counts( hist(x, plot=FALSE))
```

`get_density`*Get density*

Description

The function formats the density distribution from the histogram function

Usage

```
get_density(hist)
```

Arguments

hist histogram

Value

array

Examples

```
x <- runif(1000)
density <- get_density( hist(x, plot=FALSE))
```

`get_expression_data_gemma`

Obtain expression matrix from the GEMMA database

Description

The function downloads and parses the expression matrix from the GEMMA database, specified by the GEO ID

Usage

```
get_expression_data_gemma(gseid, filtered = "true")
```

Arguments

| | |
|-----------------------|--|
| <code>gseid</code> | GEO ID of the expression experiment |
| <code>filtered</code> | flag to indicate whether or not the data is QC |

Value

list of genes and the expression matrix

`get_expression_matrix_from_GEO`

Obtain expression matrix from GEO database

Description

The function downloads and parses the expression matrix from the GEO file specified by the GEO ID

Usage

```
get_expression_matrix_from_GEO(gseid)
```

Arguments

| | |
|--------------------|-------------------------------------|
| <code>gseid</code> | GEO ID of the expression experiment |
|--------------------|-------------------------------------|

Value

list of genes and the expression matrix

get_phenocarta *Downloading and filtering Phenocarta*

Description

The function downloads the latest version of phenocarta

Usage

```
get_phenocarta(species = "human", type = "all")
```

Arguments

| | |
|---------|--------|
| species | string |
| type | string |

Value

data data.frame with phenocarta data

get_prc *Build precision-recall curve*

Description

The function calculates the recall and precision

Usage

```
get_prc(scores, labels)
```

Arguments

| | |
|--------|---------------|
| scores | numeric array |
| labels | binary array |

Value

recall,precision numeric arrays

Examples

```
labels <- c(rep(0,10))
labels[c(1,3,5)] <- 1
scores <- 10:1
prc <- get_prc(scores, labels)
```

| | |
|---------|--|
| get_roc | <i>Build receiver operating characteristic curve</i> |
|---------|--|

Description

The function calculates the FPR and TRPR for the receiver operating characteristic (ROC)

Usage

```
get_roc(scores, labels)
```

Arguments

| | |
|--------|---------------|
| scores | numeric array |
| labels | binary array |

Value

FPR,TPR numeric arrays

Examples

```
labels <- c(rep(0,10))
labels[c(1,3,5)] <- 1
scores <- 10:1
roc <- get_roc(scores, labels)
```

| | |
|----------|-------------------|
| GO.human | <i>GO - human</i> |
|----------|-------------------|

Description

A dataset of the gene GO associations

Format

A data frame with 2511938 rows and 4 variables:

name gene symbol

entrezID entrez identifier

GO gene ontology term ID

evidence evidence code

@source <http://geneontology.org/>

GO.mouse

GO - mouse

Description

A dataset of the gene GO associations

Format

A data frame with 2086086 rows and 4 variables:

name gene symbol

entrezID entrez identifier

GO gene ontology term ID

evidence evidence code

@source <http://geneontology.org/>

GO.voc

Gene ontology vocabulary

Description

A dataset of the gene ontology vocabulary

Format

A data frame with 42266 rows and 3 variables:

GOID GO identifier

term GO description

domain GO domain

@source <http://geneontology.org/>

| | |
|-----------------|----------------------------------|
| make_annotatons | <i>Creating gene annotations</i> |
|-----------------|----------------------------------|

Description

The function annotates a list of genes according to a given ontology. It creates a binary matrix associating genes (rows) with labels (columns).

Usage

```
make_annotatons(data, listA, listB)
```

Arguments

| | |
|-------|---|
| data | 2-column matrix, each row a pair indicating a relationship or interaction |
| listA | string array of genes |
| listB | string array of labels/functions |

Value

net matrix binary

Examples

```
gene.list <- paste('gene', 1:100, sep='')
labels.list <- paste('labels', 1:10, sep='')
data <- matrix(0,nrow=100, ncol=2)
data[,1] <- sample(gene.list, 100, replace=TRUE)
data[,2] <- sample(labels.list, 100, replace=TRUE)
net <- make_annotatons(data, gene.list, labels.list)
```

| | |
|---------------|--|
| make_genelist | <i>Creating list of all genes in the data set.</i> |
|---------------|--|

Description

The function extracts the list of all genes in the data set

Usage

```
make_genelist(gene_data_interacting)
```

Arguments

| | |
|-----------------------|---|
| gene_data_interacting | 2-column matrix, each row a pair indicating a relationship or interaction |
|-----------------------|---|

Value

list array of data labels

Examples

```
gene.list <- paste('gene', 1:100, sep='')
data <- matrix(0,nrow=100, ncol=2)
data[,1] <- sample(gene.list, 50, replace=TRUE)
data[,2] <- sample(gene.list, 50, replace=TRUE)
genes <- make_genelist(data)
```

make_gene_network *Creating gene-by-gene network*

Description

The function creates a gene-by-gene matrix with binary entries indicating interaction (1) or no interaction (0) between the genes.

Usage

```
make_gene_network(data, list)
```

Arguments

| | |
|------|---|
| data | 2-column matrix, each row a pair indicating a relationship or interaction |
| list | string array of genes |

Value

net matrix binary characterizing interactions

Examples

```
gene.list <- paste('gene', 1:100, sep='')
data <- matrix(0,nrow=100, ncol=2)
data[,1] <- sample(gene.list, 100)
data[,2] <- sample(gene.list, 100)
net <- make_gene_network(data, gene.list)
```

| | |
|------------------|---|
| make_transparent | <i>Make a color transparent (Taken from an answer on StackOverflow by Nick Sabbe)</i> |
|------------------|---|

Description

Make a color transparent (Taken from an answer on StackOverflow by Nick Sabbe)

Usage

```
make_transparent(someColor, alpha = 100)
```

Arguments

| | |
|-----------|--|
| someColor | color number, string or hexadecimal code |
| alpha | numeric transparency |

Value

someColor rgb

| | |
|-----------------|--|
| neighbor_voting | <i>Evaluating Gene Function Prediction</i> |
|-----------------|--|

Description

The function performs gene function prediction based on 'guilt by association' using cross validation ([1]). Performance and significance are evaluated by calculating the AUROC or AUPRC of each functional group.

Usage

```
neighbor_voting(genes.labels, network, nFold = 3, output = "AUROC",
  FLAG_DRAW = FALSE)
```

Arguments

| | |
|--------------|--|
| genes.labels | numeric array |
| network | numeric array symmetric, gene-by-gene matrix |
| nFold | numeric value, default is 3 |
| output | string, default is AUROC |
| FLAG_DRAW | binary flag to draw roc plot |

Value

scores numeric matrix

Examples

```
genes.labels <- matrix( sample( c(0,1), 1000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:10, sep='')
net <- cor( matrix( rnorm(10000), ncol=100), method='spearman')
rownames(net) <- paste('gene', 1:100, sep='')
colnames(net) <- paste('gene', 1:100, sep='')

aurocs <- neighbor_voting(genes.labels, net, output = 'AUROC')

auprcs <- neighbor_voting(genes.labels, net, output = 'PR')
```

| | |
|-------------|------------------------------|
| node_degree | <i>Calculate node degree</i> |
|-------------|------------------------------|

Description

The function calculates the node degree of a network

Usage

```
node_degree(net)
```

Arguments

net numeric matrix

Value

node_degree numeric array

Examples

```
net <- cor( matrix(rnorm(1000), ncol=10))
n <- 10
net <- matrix(rank(net, na.last = 'keep', ties.method = 'average'), nrow = n, ncol = n)
net <- net/max(net, na.rm=TRUE)
nd <- node_degree(net)
```

| | |
|-------|-----------------------|
| ortho | <i>Gene orthologs</i> |
|-------|-----------------------|

Description

A list containing identifiers for the subsets of gene orthologs

Format

List orthologs for 5 species

dros List of Entrez identifiers, Drosophila

celeg List of Entrez identifiers, C. elegans

yeast List of Entrez identifiers, Yeast

mouse List of Entrez identifiers, Mouse

zf List of Entrez identifiers, Zebrafish

@source <http://useast.ensembl.org/index.html/>

| | |
|-------|-------------------|
| pheno | <i>Phenocarta</i> |
|-------|-------------------|

Description

A dataset of gene disease associations

Format

A data frame with 142272 rows and 4 variables:

entrezID chromosomal start position, in base pairs

name HUGO gene identifier

species species

disease disease

@source <http://www.chibi.ubc.ca/Gemma/phenotypes.html>

plot_densities *Plot densities*

Description

The function plots multiple density curves and compares their modes

Usage

```
plot_densities(hists, id, col = c("lightgrey"), xlab = "",
  ylab = "Density", mode = "hist")
```

Arguments

| | |
|-------|--|
| hists | list of histogram objects or density objects |
| id | string |
| col | color for shading |
| xlab | string x-axis label |
| ylab | string y-axis label |
| mode | flag indicating histogram or density |

Value

null

Examples

```
aurocsA <- density((runif(1000)+runif(1000)+runif(1000)+runif(1000))/4)
aurocsB <- density((runif(1000)+runif(1000)+runif(1000))/3)
aurocsC <- density(runif(1000))
hists <- list(aurocsA, aurocsB, aurocsC)
temp <- plot_densities(hists, '', mode='density')
```

plot_density_compare *Plot density comparisons*

Description

The function plots two density curves and compares their modes

Usage

```
plot_density_compare(aucA, aucB, col = "lightgrey",
  xlab = "AUROC (neighbor voting)", ylab = "Density", mode = TRUE)
```

Arguments

| | |
|------|------------------------------|
| aucA | numeric array of aurops |
| aucB | numeric array of aurops |
| col | color of lines |
| xlab | string label |
| ylab | string label |
| mode | boolean to plot mode or mean |

Value

null

Examples

```

auropsA <- (runif(1000)+runif(1000)+runif(1000)+runif(1000))/4
auropsB <- runif(1000)
plot_density_compare(auropsA, auropsB)

```

plot_distribution *Plot distribution histogram*

Description

The function plots a the distribution of AUROCs

Usage

```

plot_distribution(auc, b = 20, col = "lightgrey", xlab = "",
  ylab = "Density", xlim = c(0.4, 1), ylim = c(0, 5), med = TRUE,
  avg = TRUE, density = TRUE, bars = FALSE)

```

Arguments

| | |
|---------|-----------------------------|
| auc | numeric aucs |
| b | array of breaks |
| col | color of line |
| xlab | string label |
| ylab | string label |
| xlim | range of values for xaxis |
| ylim | range of values for yaxis |
| med | boolean to plot median auc |
| avg | boolean to plot average auc |
| density | boolean |
| bars | boolean for barplot |

Value

auc list and quartiles

Examples

```
aurops <- (runif(1000)+runif(1000)+runif(1000)+runif(1000))/4  
d <- plot_distribution(aurops)
```

plot_network_heatmap *Plot network heatmap*

Description

The function draws a heatmap to visualize a network

Usage

```
plot_network_heatmap(net, colrs)
```

Arguments

| | |
|-------|---------------------------------------|
| net | a numeric matrix of edge weights |
| colrs | a range of colors to plot the network |

Value

null

Examples

```
network <- cor(matrix( rnorm(10000), nrow=100))  
plot_network_heatmap(network)
```

plot_prc *Plot precision recall curve*

Description

The function calculates the precision and recall and plots the curve

Usage

```
plot_prc(scores, labels)
```

Arguments

| | |
|--------|---------------|
| scores | numeric array |
| labels | binary array |

Value

prc numeric arrays

Examples

```
labels <- c(rep(0,10))
labels[c(1,3,5)] <- 1
scores <- 10:1
roc <- plot_prc(scores, labels)
```

plot_roc

Plot receiver operating characteristic curve

Description

The function calculates the FPR and TRPR for the receiver operating characteristic (ROC) and plots the curve

Usage

```
plot_roc(scores, labels)
```

Arguments

| | |
|--------|---------------|
| scores | numeric array |
| labels | binary array |

Value

FPR,TPR numeric arrays

Examples

```
labels <- c(rep(0,10))
labels[c(1,3,5)] <- 1
scores <- 10:1
roc <- plot_roc(scores, labels)
```

plot_roc_overlay *Plot ROC overlay*

Description

The function plots a density overlay of ROCs given the scores and labels

Usage

```
plot_roc_overlay(scores.mat, labels.mat, nbins = 100)
```

Arguments

| | |
|------------|---------------|
| scores.mat | numeric array |
| labels.mat | numeric array |
| nbins | numeric value |

Value

list of Z(matrix) and roc_sum (average ROC curve)

Examples

```
genes.labels <- matrix( c(rep(1, 1000), rep(0,9000)), nrow=1000, byrow=TRUE)
rownames(genes.labels) = paste('gene', 1:1000, sep='')
colnames(genes.labels) = paste('function', 1:10, sep='')

scores <- matrix( rnorm(10000), nrow=1000)
scores <- apply(scores, 2, rank)
rownames(scores) = paste('gene', 1:1000, sep='')
colnames(scores) = paste('function', 1:10, sep='')

z <- plot_roc_overlay(scores, genes.labels)
```

plot_value_compare *Plot value comparisons*

Description

The function plots a scatter

Usage

```
plot_value_compare(aucA, aucB, xlab = "AUROC", ylab = "AUROC", xlim = c(0,
1), ylim = c(0, 1))
```


Arguments

| | |
|------|---------------------------|
| aucA | numeric array of aucs |
| aucB | numeric array of aucs |
| xlab | string label |
| ylab | string label |
| xlim | range of values for xaxis |
| ylim | range of values for yaxis |

Value

null

predictions *Performing Gene Function Prediction*

Description

The function performs gene function prediction on the whole data set using the 'guilt by association' principle ([1]).

Usage

```
predictions(genes.labels, network)
```

Arguments

| | |
|--------------|--|
| genes.labels | numeric array |
| network | numeric array symmetric, gene-by-gene matrix |

Value

scores numeric matrix

Examples

```
genes.labels <- matrix( sample( c(0,1), 1000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:10, sep='')
net <- cor( matrix( rnorm(10000), ncol=100), method='spearman')
rownames(net) <- paste('gene', 1:100, sep='')
colnames(net) <- paste('gene', 1:100, sep='')

preds <- predictions(genes.labels, net)
```

| | |
|--------|----------------------------------|
| repmat | <i>Rep function for matrices</i> |
|--------|----------------------------------|

Description

The function generates a matrix by binding the columns and rows

Usage

```
repmat(X, m, n)
```

Arguments

| | |
|---|---------------------------------------|
| X | numeric matrix |
| m | numeric value, repeat rows m times |
| n | numeric value, repeat columns n times |

Value

list of genes and the expression matrix

Examples

```
genes.labels <- matrix( sample( c(0,1), 1000, replace=TRUE), nrow=100)
expand <- repmat( genes.labels, 1,2)
```

| | |
|---------|---|
| run_GBA | <i>Performing 'Guilt by Association' Analysis</i> |
|---------|---|

Description

The function runs and evaluates gene function prediction based on the 'guilt by association'-principle using neighbor voting ([neighbor_voting](#)) [1]. As a measure of performance and significance of results, AUCs of all evaluated functional groups are calculated.

Usage

```
run_GBA(network, labels, min = 20, max = 1000, nfold = 3)
```

Arguments

| | |
|---------|--|
| network | numeric array symmetric, gene-by-gene matrix |
| labels | numeric array |
| min | numeric value to limit gene function size |
| max | numeric value to limit gene function size |
| nfold | numeric value, default is 3 |

Value

list roc.sub, genes, auroc

Examples

```
genes.labels <- matrix( sample( c(0,1), 1000, replace=TRUE), nrow=100)
rownames(genes.labels) = paste('gene', 1:100, sep='')
colnames(genes.labels) = paste('function', 1:10, sep='')
net <- cor( matrix( rnorm(10000), ncol=100), method='spearman')
rownames(net) <- paste('gene', 1:100, sep='')
colnames(net) <- paste('gene', 1:100, sep='')

gba <- run_GBA(net, genes.labels, min=10)
```

Index

- *Topic **AUC**
 - auc_multifunc, 4
- *Topic **AUROC**
 - plot_distribution, 29
- *Topic **ExpressionSet**
 - build_coexp_expressionSet, 7
 - build_coexp_network, 8
- *Topic **FPR**
 - get_roc, 21
 - plot_roc, 31
- *Topic **GEMMA**
 - get_expression_data_gemma, 19
- *Topic **GEO**
 - build_coexp_GEOID, 8
 - get_expression_data_gemma, 19
 - get_expression_matrix_from_GEO, 19
- *Topic **GSE**
 - build_coexp_GEOID, 8
 - get_expression_data_gemma, 19
 - get_expression_matrix_from_GEO, 19
- *Topic **PRC**
 - plot_prc, 30
- *Topic **ROC**
 - auroc_analytic, 5
 - get_roc, 21
 - plot_distribution, 29
 - plot_roc, 31
 - plot_roc_overlay, 32
- *Topic **TPR**
 - get_roc, 21
 - plot_roc, 31
- *Topic **analytic**
 - auroc_analytic, 5
- *Topic **annotations**
 - filter_orthologs, 15
 - make_annotations, 23
- *Topic **area**
 - auprc, 5
 - auroc_analytic, 5
 - get_auc, 17
- *Topic **association**
 - neighbor_voting, 25
 - predictions, 33
 - run_GBA, 34
- *Topic **assortativity**
 - assortativity, 3
- *Topic **a**
 - get_auc, 17
- *Topic **biogrid**
 - get_biogrid, 17
- *Topic **by**
 - neighbor_voting, 25
 - predictions, 33
 - run_GBA, 34
- *Topic **characteristic**
 - auroc_analytic, 5
 - get_roc, 21
 - plot_roc, 31
- *Topic **coexpression**
 - build_coexp_expressionSet, 7
 - build_coexp_GEOID, 8
 - build_coexp_network, 8
- *Topic **cross**
 - neighbor_voting, 25
 - run_GBA, 34
- *Topic **curve**
 - get_auc, 17
- *Topic **degree**
 - node_degree, 26
- *Topic **dense**
 - build_coexp_expressionSet, 7
 - build_coexp_GEOID, 8
 - build_coexp_network, 8
- *Topic **density**
 - plot_densities, 28
- *Topic **distribution**
 - plot_distribution, 29
- *Topic **download**
 - get_biogrid, 17
 - get_phenocarta, 20
- *Topic **evaluation**
 - calculate_multifunc, 10
 - neighbor_voting, 25
 - run_GBA, 34
- *Topic **evalutation**
 - auc_multifunc, 4

- *Topic **experiment**
 - get_expression_data_gemma, 19
 - get_expression_matrix_from_GEO, 19
- *Topic **expressionSet**
 - build_coexp_expressionSet, 7
 - build_coexp_network, 8
- *Topic **expression**
 - get_expression_data_gemma, 19
 - get_expression_matrix_from_GEO, 19
- *Topic **extended**
 - extend_network, 12
- *Topic **extract**
 - make_genelist, 23
- *Topic **filter**
 - filter_network, 13
 - filter_network_cols, 13
 - filter_network_rows, 14
 - filter_orthologs, 15
- *Topic **fmeasure**
 - fmeasure, 16
- *Topic **function**
 - calculate_multifunc, 10
 - make_annotations, 23
 - neighbor_voting, 25
 - predictions, 33
 - run_GBA, 34
- *Topic **gene-by-gene**
 - build_binary_network, 6
 - build_weighted_network, 9
 - make_gene_network, 24
- *Topic **gene**
 - calculate_multifunc, 10
 - make_annotations, 23
 - make_genelist, 23
 - neighbor_voting, 25
 - predictions, 33
 - run_GBA, 34
- *Topic **guilt**
 - neighbor_voting, 25
 - predictions, 33
 - run_GBA, 34
- *Topic **heatmap**
 - plot_network_heatmap, 30
- *Topic **histogram**
 - get_counts, 18
 - get_density, 18
- *Topic **igraph**
 - extend_network, 12
- *Topic **image**
 - plot_network_heatmap, 30
- *Topic **interaction**
 - build_binary_network, 6
 - build_weighted_network, 9
 - make_gene_network, 24
- *Topic **jaccard**
 - build_semantic_similarity_network, 9
- *Topic **labels**
 - make_annotations, 23
- *Topic **length**
 - extend_network, 12
- *Topic **list**
 - make_genelist, 23
- *Topic **matrix**
 - repmat, 34
- *Topic **mean**
 - get_auc, 17
- *Topic **metric**
 - auprc, 5
 - auroc_analytic, 5
 - get_roc, 21
 - node_degree, 26
 - plot_prc, 30
 - plot_roc, 31
- *Topic **multifunctionality**
 - auc_multifunc, 4
- *Topic **neighbor**
 - neighbor_voting, 25
 - predictions, 33
 - run_GBA, 34
- *Topic **network**
 - assortativity, 3
 - build_binary_network, 6
 - build_coexp_expressionSet, 7
 - build_coexp_GEOID, 8
 - build_coexp_network, 8
 - build_semantic_similarity_network, 9
 - build_weighted_network, 9
 - extend_network, 12
 - filter_network, 13
 - filter_network_cols, 13
 - filter_network_rows, 14
 - make_gene_network, 24
 - node_degree, 26
 - plot_network_heatmap, 30
- *Topic **node**
 - node_degree, 26
- *Topic **ontology**
 - calculate_multifunc, 10
 - make_annotations, 23
- *Topic **operating**
 - auroc_analytic, 5
 - get_roc, 21

- plot_roc, 31
- *Topic **orthologs**
 - filter_orthologs, 15
- *Topic **overlay**
 - plot_roc_overlay, 32
- *Topic **path**
 - extend_network, 12
- *Topic **phenocarta**
 - get_phenocarta, 20
- *Topic **plot**
 - conv_smoother, 11
 - get_density, 18
 - plot_densities, 28
 - plot_density_compare, 28
 - plot_distribution, 29
 - plot_network_heatmap, 30
 - plot_prc, 30
 - plot_roc_overlay, 32
 - plot_value_compare, 32
- *Topic **precision-recall**
 - auprc, 5
 - fmeasure, 16
- *Topic **precision**
 - get_prc, 20
 - plot_prc, 30
- *Topic **precision-recall**
 - get_prc, 20
- *Topic **prediction**
 - calculate_multifunc, 10
 - neighbor_voting, 25
 - predictions, 33
 - run_GBA, 34
- *Topic **properties**
 - assortativity, 3
- *Topic **recall**
 - get_prc, 20
 - plot_prc, 30
- *Topic **receiver**
 - auroc_analytic, 5
 - get_roc, 21
 - plot_roc, 31
- *Topic **repeat**
 - repmat, 34
- *Topic **repmat**
 - repmat, 34
- *Topic **rolling**
 - get_auc, 17
- *Topic **rows**
 - filter_network, 13
 - filter_network_cols, 13
 - filter_network_rows, 14
- *Topic **semantic**
 - build_semantic_similarity_network, 9
- *Topic **shortest**
 - extend_network, 12
- *Topic **similarity**
 - build_semantic_similarity_network, 9
- *Topic **smooth**
 - conv_smoother, 11
- *Topic **topology**
 - assortativity, 3
 - node_degree, 26
- *Topic **to**
 - make_annotations, 23
- *Topic **under**
 - get_auc, 17
- *Topic **validation**
 - neighbor_voting, 25
 - run_GBA, 34
- *Topic **voting**
 - neighbor_voting, 25
 - predictions, 33
 - run_GBA, 34
- assortativity, 3
- attr.human, 3
- attr.mouse, 4
- auc_multifunc, 4
- auprc, 5
- auroc_analytic, 5
- biogrid, 6
- build_binary_network, 6
- build_coexp_expressionSet, 7
- build_coexp_GEOID, 8
- build_coexp_network, 7, 8, 8
- build_semantic_similarity_network, 9
- build_weighted_network, 9
- calculate_multifunc, 4, 10
- conv_smoother, 11
- convolve, 11
- cor, 8
- example_annotations, 11
- example_binary_network, 11
- example_coexpression, 12
- example_neighbor_voting, 12
- extend_network, 12
- filter_network, 13
- filter_network_cols, 13
- filter_network_rows, 14

filter_orthologs, 15
fmeasure, 16

genes, 16
get_auc, 17
get_biogrid, 17
get_counts, 18
get_density, 18
get_expression_data_gemma, 19
get_expression_matrix_from_GEO, 8, 19
get_phenocarta, 20
get_prc, 20
get_roc, 21
GO.human, 21
GO.mouse, 22
GO.voc, 22

make_annotations, 23
make_gene_network, 24
make_genelist, 23
make_transparent, 25

neighbor_voting, 25, 34
node_degree, 26

ortho, 27

pheno, 27
plot_densities, 28
plot_density_compare, 28
plot_distribution, 29
plot_network_heatmap, 30
plot_prc, 30
plot_roc, 31
plot_roc_overlay, 32
plot_value_compare, 32
predictions, 33

repmat, 34
run_GBA, 34