Package 'PPInfer'

April 12, 2018

Type Package

Title Inferring functionally related proteins using protein interaction networks

Description Interactions between proteins occur in many, if not most, biological processes. Most proteins perform their functions in networks associated with other proteins and other biomolecules. This fact has motivated the development of a variety of experimental methods for the identification of protein interactions. This variety has in turn ushered in the development of numerous different computational approaches for modeling and predicting protein interactions. Sometimes an experiment is aimed at identifying proteins closely related to some interesting proteins. A network based statistical learning method is used to infer the putative functions of proteins from the known functions of its neighboring proteins on a PPI network. This package identifies such proteins often involved in the same or similar biological functions.

Version 1.5.1

Date 2017-02-10

Author Dongmin Jung, Xijin Ge

Maintainer Dongmin Jung <dmdmjung@gmail.com>

Depends biomaRt, fgsea, kernlab, ggplot2, igraph, STRINGdb, yeastExpData

Suggests

License Artistic-2.0

biocViews Software, StatisticalMethod, Network, GraphAndNetwork, GeneSetEnrichment, NetworkEnrichment, Pathways

NeedsCompilation no

R topics documented:

PPInfer-package *Inferring functionally related proteins using protein interaction networks*

Description

Interactions between proteins occur in many, if not most, biological processes. Most proteins perform their functions in networks associated with other proteins and other biomolecules. This fact has motivated the development of a variety of experimental methods for the identification of protein interactions. This variety has in turn ushered in the development of numerous different computational approaches for modeling and predicting protein interactions. Sometimes an experiment is aimed at identifying proteins closely related to some interesting proteins. A network based statistical learning method is used to infer the putative functions of proteins from the known functions of its neighboring proteins on a PPI network. This package identifies such proteins often involved in the same or similar biological functions.

Details

The DESCRIPTION file:

Package:	PPInfer
Type:	Package
Title:	Inferring functionally related proteins using protein interaction networks
Description:	Interactions between proteins occur in many, if not most, biological processes. Most proteins perfor
Version:	1.5.1
Date:	2017-02-10
Author:	Dongmin Jung, Xijin Ge
Maintainer:	Dongmin Jung <dmdmjung@gmail.com>
Depends:	biomaRt, fgsea, kernlab, ggplot2, igraph, STRINGdb, yeastExpData
Suggests:	
License:	Artistic-2.0
biocViews:	Software, StatisticalMethod, Network, GraphAndNetwork, GeneSetEnrichment, NetworkEnrichme
NeedsCompilation:	no

Index of help topics:

```
GSEA.barplot            Visualize the gene set enrichment analysis
ORA                     Over-representation Analysis
ORA.barplot             Visualize the over-representation analysis
PPInfer-package         Inferring functionally related proteins using
                        protein interaction networks
enrich.net              Visualize network for the functional enrichment
                        analysis
net.infer               Inferring functionally related proteins using
                        networks
net.infer.ST            Inferring functionally related proteins with
```

	self training
net.kernel	Kernel matrix for a graph
ppi.infer.human	Inferring functionally related proteins using protein networks for human
ppi.infer.mouse	Inferring functionally related proteins using protein networks for mouse
self.train.kernel	Self training for a kernel matrix

Further information is available in the following vignettes:

PPInfer	User manual (source, pdf)

Author(s)

Dongmin Jung, Xijin Ge

Maintainer: Dongmin Jung <dmdmjung@gmail.com>

---

enrich.net *Visualize network for the functional enrichment analysis*

Description

The connection between nodes depends on the proportion of overlapping genes between two categories.

Usage

```
enrich.net(x, gene.set, node.id, node.name = node.id, pvalue,
           n = 50, numChar = NULL, pvalue.cutoff = 0.05,
           edge.cutoff = 0.05, degree.cutoff = 0,
           edge.width = function(x) {10*x^2},
           node.size = function(x) {2.5*log10(x)},
           group = FALSE, group.color = c('red', 'green'),
           group.shape = c('circle', 'square'),
           legend.parameter = list('topright'),
           show.legend = TRUE, ...)
```

Arguments

x	a result with category and p-value of gene sets
gene.set	gene sets which is already used for functional enrichment
node.id	name of gene sets
node.name	label of nodes in the network (default: node.id)
pvalue	pvalues for categories
n	number of top categories (default: 50)
numChar	the maximal number of characters of the label of gene sets

`pvalue.cutoff` nodes with p-values which are greater than pvalue.cutoff are removed (default: 0.05)

`edge.cutoff` edges with the proportion which is less than edge.cutoff are removed (default: 0.05)

`degree.cutoff` nodes with the degrees which are less than degree.cutoff are removed (default: 0)

`edge.width` width of edges

`node.size` size of nodes

`group` variable for group

`group.color` color for group (default: red and green for 2 groups)

`group.shape` shape for group (default: circle and square for 2 groups)

`legend.parameter` list of parametres for the legend

`show.legend` show the legend (default: TRUE)

`...` additional parameters for the igraph

Value

plot for the network. The size of nodes is proportional to the size of gene sets. The more significant categories are, the less transparent their nodes are.

Author(s)

Dongmin Jung, Xijin Ge

References

Yu G, Wang L, Yan G and He Q (2015). "DOSE: an R/Bioconductor package for Disease Ontology Semantic and Enrichment analysis." Bioinformatics, 31(4), pp. 608-609.

See Also

igraph

Examples

```
data(examplePathways)
data(exampleRanks)
set.seed(1)
result.GSEA <- fgsea(examplePathways, exampleRanks, nperm = 1000)
enrich.net(result.GSEA, examplePathways, node.id = 'pathway',
           pvalue = 'pval', edge.cutoff = 0.6, degree.cutoff = 1,
           n = 50, vertex.label.cex = 0.75, show.legend = FALSE,
           edge.width = function(x) {5*sqrt(x)},
           layout = igraph::layout.kamada.kawai)
```

`GSEA.barplot` *Visualize the gene set enrichment analysis*

Description

For the functional enrichment analysis, we can visualize the result from the gene set enrichment analysis.

Usage

```
GSEA.barplot(object, category, score, pvalue, top = 10,
             sort = NULL, decreasing = FALSE, numChar = NULL,
             title = NULL, transparency = 0.5, plot = TRUE)
```

Arguments

`object`	a table with category, enrichment score and p-value of gene sets
`category`	name of gene sets
`score`	enrichment score
`pvalue`	p-value of gene sets
`top`	the number of top categories (default: 10)
`sort`	a variable used for sorting data
`decreasing`	logical indicating whether ascending or descending order (default: FALSE)
`numChar`	the maximal number of characters of the name of gene sets
`title`	title for the plot
`transparency`	transparency (default: 0.5)
`plot`	return plot when plot is true, otherwise return table (default: TRUE)

Value

GSEA barplot

Author(s)

Dongmin Jung, Xijin Ge

References

Yu G, Wang L, Yan G and He Q (2015). "DOSE: an R/Bioconductor package for Disease Ontology Semantic and Enrichment analysis." Bioinformatics, 31(4), pp. 608-609.

See Also

ggplot2

Examples

```
data(examplePathways)
data(exampleRanks)
set.seed(1)
result.GSEA <- fgsea(examplePathways, exampleRanks, nperm = 1000)
GSEA.barplot(result.GSEA, category = 'pathway', score = 'NES',
             pvalue = 'pval', sort = 'NES', decreasing = TRUE)
```

---

net.infer *Inferring functionally related proteins using networks*

Description

Proteins can be classified by using networks to identify functionally closely related proteins.

Usage

```
net.infer(target, kernel, top = NULL, cross = 0,
          C = 1, nu = 0.2, epsilon = 0.1, cache1 = 40,
          tol1 = 0.001, shrinking1 = TRUE, cache2 = 40,
          tol2 = 0.001, shrinking2 = TRUE)
```

Arguments

target	set of interesting proteins or target class
kernel	the regularized Laplacian matrix for a graph
top	number of top proteins most closely related to target class (default: all proteins except for target and pseudo-absence class)
cross	if a integer value k>0 is specified, a k-fold cross validation on the training data is performed to assess the quality of the model
C	cost of constraints violation for SVM (default: 1)
nu	The nu parameter for OCSVM (default: 0.2)
epsilon	epsilon in the insensitive-loss function for OCSVM (default: 0.1)
cache1	cache memory in MB for OCSVM (default: 40)
tol1	tolerance of termination criterion for OCSVM (default: 0.001)
shrinking1	option whether to use the shrinking-heuristics for OCSVM (default: TRUE)
cache2	cache memory in MB for SVM (default: 40)
tol2	tolerance of termination criterion for SVM (default: 0.001)
shrinking2	option whether to use the shrinking-heuristics for SVM (default: TRUE)

Value

list	list of a target class used in the model
error	training error
CVerror	cross validation error, (when cross > 0)
top	top proteins
score	decision values for top proteins

Author(s)

Dongmin Jung, Xijin Ge

References

Senay, S. D. et al. (2013). Novel three-step pseudo-absence selection technique for improved species distribution modelling. PLOS ONE. 8(8), e71218.

See Also

ksvm

Examples

```
# example 1
## Not run:
string.db.9606 <- STRINGdb$new(version = '10',species = 9606,
                               score_threshold = 999)
string.db.9606.graph <- string.db.9606$get_graph()
K.9606 <- net.kernel(string.db.9606.graph)
rownames(K.9606) <- substring(rownames(K.9606), 6)
colnames(K.9606) <- substring(colnames(K.9606), 6)
target <- colnames(K.9606)[1:100]
infer <- net.infer(target, K.9606, 10)

## End(Not run)

# example 2
data(litG)
litG <- igraph.from.graphNEL(litG)
sg <- decompose(litG, min.vertices = 50)
sg <- sg[[1]]
K <- net.kernel(sg)
litG.infer <- net.infer(names(V(sg))[1:10], K, top=20)
```

net.infer.ST *Inferring functionally related proteins with self training*

Description

This function is the self-training version of net.infer. The function net.infer is the special case of net.infer.ST where a single iteration is conducted.

Usage

```
net.infer.ST(target, kernel, top = NULL, C = 1, nu = 0.2,
             epsilon = 0.1, cache1 = 40, tol1 = 0.001, shrinking1 = TRUE,
             cache2 = 40, tol2 = 0.001, shrinking2 = TRUE, thrConf = 0.9,
             maxIts = 10, percFull = 1, verbose = FALSE)
```

Arguments

`target` set of interesting proteins or target class

`kernel` the regularized Laplacian matrix for a graph

`top` number of top proteins most closely related to target class (default: all proteins except for target and pseudo-absence class)

`C` cost of constraints violation for SVM (default: 1)

`nu` The nu parameter for OCSVM (default: 0.2)

`epsilon` epsilon in the insensitive-loss function for OCSVM (default: 0.1)

`cache1` cache memory in MB for OCSVM (default: 40)

`tol1` tolerance of termination criterion for OCSVM (default: 0.001)

`shrinking1` option whether to use the shrinking-heuristics for OCSVM (default: TRUE)

`cache2` cache memory in MB for SVM (default: 40)

`tol2` tolerance of termination criterion for SVM (default: 0.001)

`shrinking2` option whether to use the shrinking-heuristics for SVM (default: TRUE)

`thrConf` A number between 0 and 1, indicating the required classification confidence for an unlabelled case to be added to the labelled data set with the label predicted predicted by the classification algorithm (default: 0.9)

`maxIts` The maximum number of iterations of the self-training process (default: 10)

`percFull` A number between 0 and 1. If the percentage of labelled cases reaches this value the self-training process is stoped (default: 1)

`verbose` A boolean indicating the verbosity level of the function. (default: FALSE)

Value

`list` list of a target class used in the model

`error` training error

`top` top proteins

`score` decision values for top proteins

Author(s)

Dongmin Jung, Xijin Ge

See Also

self.train

Examples

```
data(litG)
litG <- igraph.from.graphNEL(litG)
sg <- decompose(litG, min.vertices = 50)
sg <- sg[[1]]
K <- net.kernel(sg)
litG.infer.ST <- net.infer.ST(names(V(sg))[1:10], K, top=20)
```

net.kernel *Kernel matrix for a graph*

Description

This function gives the regularized Laplacian matrix for a graph.

Usage

```
net.kernel(g, decay = 0.5)
```

Arguments

g	graph
decay	decaying constant (default: 0.5)

Value

the regularized Laplacian matrix

Author(s)

Dongmin Jung, Xijin Ge

See Also

laplacian_matrix

Examples

```
# example 1
## Not run:
string.db.9606 <- STRINGdb$new(version = '10', species = 9606,
                               score_threshold = 999)
string.db.9606.graph <- string.db.9606$get_graph()
K.9606 <- net.kernel(string.db.9606.graph)

## End(Not run)

# example 2
data(litG)
litG <- igraph.from.graphNEL(litG)
sg <- decompose(litG, min.vertices=50)
sg <- sg[[1]]
K <- net.kernel(sg)
```

ORA *Over-representation Analysis*

Description

the result from the over-representation analysis

Usage

```
ORA(pathways, gene.id, minSize = 1, maxSize = Inf,
    p.adjust.methods = NULL)
```

Arguments

`pathways`	list of gene sets
`gene.id`	set of genes
`minSize`	Minimal size of a gene set
`maxSize`	Maximal size of a gene set
`p.adjust.methods`	a correction method

Value

ORA result

Author(s)

Dongmin Jung, Xijin Ge

See Also

fisher.test

Examples

```
data(examplePathways)
data(exampleRanks)
geneNames <- names(exampleRanks)
set.seed(1)
gene.id <- sample(geneNames, 100)
ORA(examplePathways, gene.id)
```

ORA.barplot *Visualize the over-representation analysis*

Description

For the functional enrichment analysis, we can visualize the result from the over-representation analysis.

Usage

```
ORA.barplot(object, category, size, count, pvalue, top = 10,
            sort = NULL, decreasing = FALSE, p.adjust.methods = NULL,
            numChar = NULL, title = NULL, transparency = 0.5,
            plot = TRUE)
```

Arguments

`object`	a table with category, size, count and p-value of gene sets
`category`	name of gene sets
`size`	size of gene sets
`count`	count of gene sets
`pvalue`	p-value of gene sets
`top`	the number of top categories (default: 10)
`sort`	a variable used for sorting data
`decreasing`	logical indicating whether ascending or descending order (default: FALSE)
`p.adjust.methods`	a correction method
`numChar`	the maximal number of characters of the name of gene sets
`title`	title for the plot
`transparency`	transparency (default: 0.5)
`plot`	return plot when plot is true, otherwise return table (default: TRUE)

Value

ORA barplot

Author(s)

Dongmin Jung, Xijin Ge

References

Yu G, Wang L, Yan G and He Q (2015). "DOSE: an R/Bioconductor package for Disease Ontology Semantic and Enrichment analysis." Bioinformatics, 31(4), pp. 608-609.

See Also

p.adjust, ggplot2

Examples

```
data(examplePathways)
data(exampleRanks)
geneNames <- names(exampleRanks)
set.seed(1)
gene.id <- sample(geneNames, 100)
result.ORA <- ORA(examplePathways, gene.id)
ORA.barplot(result.ORA, category = "Category", size = "Size",
            count = "Count", pvalue = "pvalue", sort = "pvalue")
```

ppi.infer.human *Inferring functionally related proteins using protein networks for human*

Description

This function is designed for human protein-protein interaction from STRING database. Default format is 'hgnc'. The number of proteins is 10 in default. Note that the number of proteins used as a target may be different from the number of proteins in the input since mapping between formats is not always one-to-one in getBM.

Usage

```
ppi.infer.human(target, kernel, top = 10, classifier = net.infer,
                input = "hgnc_symbol", output = "hgnc_symbol", ...)
```

Arguments

`target`	set of interesting proteins or target class
`kernel`	the regularized Laplacian matrix for a graph
`top`	number of top proteins most closely related to target class (default: 10)
`classifier`	net.infer or net.infer.ST (default: net.infer)
`input`	input format
`output`	output format
`...`	additional parameters for the chosen classifier

Value

`list`	list of a target class used in the model
`error`	training error
`CVerror`	cross validation error, (when cross > 0 in net.infer)
`top`	top proteins
`score`	decision values for top proteins

Author(s)

Dongmin Jung, Xijin Ge

See Also

net.infer, net.infer.ST, getBM

Examples

```
# example 1
string.db.9606 <- STRINGdb$new(version = '10', species = 9606,
                               score_threshold = 999)
string.db.9606.graph <- string.db.9606$get_graph()
K.9606 <- net.kernel(string.db.9606.graph)
rownames(K.9606) <- substring(rownames(K.9606), 6)
colnames(K.9606) <- substring(colnames(K.9606), 6)
target <- colnames(K.9606)[1:100]
infer.human <- ppi.infer.human(target, K.9606, input = "ensembl_peptide_id")

## Not run:
# example 2
library(graph)
data(apopGraph)
target <- nodes(apopGraph)
apoptosis.infer <- ppi.infer.human(target, K.9606, 100)

# example 3
library(KEGGgraph)
library(KEGG.db)
pName <- "p53 signaling pathway"
pId <- mget(pName, KEGGPATHNAME2ID)[[1]]
getKGMLurl(pId, organism = "hsa")
p53 <- system.file("extdata/hsa04115.xml", package="KEGGgraph")
p53graph <- parseKGML2Graph(p53,expandGenes=TRUE)

entrez <- translateKEGGID2GeneID(nodes(p53graph))
ensembl <- useMart("ensembl")
human.ensembl <- useDataset("hsapiens_gene_ensembl",mart=ensembl)
target <- getBM(attributes=c('entrezgene', 'hgnc_symbol'),
                filter = 'entrezgene', values = entrez,
                mart = human.ensembl)[,2]
p53.infer <- ppi.infer.human(target, K.9606, 100)

## End(Not run)
```

ppi.infer.mouse *Inferring functionally related proteins using protein networks for mouse*

Description

This function is designed for mouse protein-protein interaction from STRING database. Default format is 'mgi'. The number of proteins is 10 in default. Note that the number of proteins used as a target may be different from the number of proteins in the input since mapping between formats is not always one-to-one in getBM.

Usage

```
ppi.infer.mouse(target, kernel, top = 10, classifier = net.infer,
                input = "mgi_symbol", output = "mgi_symbol", ...)
```

Arguments

`target`	set of interesting proteins or target class
`kernel`	the regularized Laplacian matrix for a graph
`top`	number of top proteins most closely related to target class (default: 10)
`classifier`	net.infer or net.infer.ST (default: net.infer)
`input`	input format
`output`	output format
`...`	additional parameters for the chosen classifier

Value

`list`	list of a target class used in the model
`error`	training error
`CVerror`	cross validation error, (when cross > 0 in net.infer)
`top`	top proteins
`score`	decision values for top proteins

Author(s)

Dongmin Jung, Xijin Ge

See Also

net.infer, net.infer.ST, getBM

Examples

```
string.db.10090 <- STRINGdb$new(version = '10', species = 10090,
                                score_threshold = 999)
string.db.10090.graph <- string.db.10090$get_graph()
K.10090 <- net.kernel(string.db.10090.graph)
rownames(K.10090) <- substring(rownames(K.10090), 7)
colnames(K.10090) <- substring(colnames(K.10090), 7)
target <- colnames(K.10090)[1:100]
infer.mouse <- ppi.infer.mouse(target, K.10090, input="ensembl_peptide_id")
```

self.train.kernel *Self training for a kernel matrix*

Description

This function can be used for classification of semi-supervised data by using the kernel support vector machine.

Usage

```
self.train.kernel(K, y, type = 'response', C = 1, cache = 40,
                  tol = 0.001, shrinking = TRUE, thrConf = 0.9,
                  maxIts = 10, percFull = 1, verbose = FALSE)
```

Arguments

`K`	kernel matrix
`y`	lable vector
`type`	one of response, probabilities ,votes, decision indicating the type of output (default: response)
`C`	cost of constraints violation for SVM (default: 1)
`cache`	cache memory in MB for SVM (default: 40)
`tol`	tolerance of termination criterion for SVM (default: 0.001)
`shrinking`	option whether to use the shrinking-heuristics for OCSVM (default: TRUE)
`thrConf`	A number between 0 and 1, indicating the required classification confidence for an unlabelled case to be added to the labelled data set with the label predicted predicted by the classification algorithm (default: 0.9)
`maxIts`	The maximum number of iterations of the self-training process (default: 10)
`percFull`	A number between 0 and 1. If the percentage of labelled cases reaches this value the self-training process is stoped (default: 1)
`verbose`	A boolean indicating the verbosity level of the function (default: FALSE)

Value

prediction from the SVM

Author(s)

Dongmin Jung, Xijin Ge

References

Torgo, L. (2016) Data Mining using R: learning with case studies, second edition, Chapman & Hall/CRC.

Examples

```
data(litG)
litG <- igraph.from.graphNEL(litG)
sg <- decompose(litG, min.vertices = 50)
sg <- sg[[1]]
K <- net.kernel(sg)
y <- rep(NA, length(V(sg)))
y[1:10] <- 1
y[11:20] <- 0
y <- factor(y)
self.train.kernel(K, y)
```

Index