Package 'pandaR'

October 7, 2024

Title PANDA Algorithm

Version 1.36.0

Author Dan Schlauch, Joseph N. Paulson, Albert Young, John Quackenbush, Kimberly Glass

Maintainer
Joseph N. Paulson <paulson.joseph@gene.com>, Dan Schlauch <dschlauch@genospace.com>

Description Runs PANDA, an algorithm for discovering novel network structure by combining information from multiple complementary data sources.

Depends R (>= 3.0.0), methods, Biobase, BiocGenerics,

Imports matrixStats, igraph, ggplot2, grid, reshape, plyr, RUnit, hexbin

Suggests knitr, rmarkdown

biocViews StatisticalMethod, GraphAndNetwork, Microarray, GeneRegulation, NetworkInference, GeneExpression, Transcription, Network

VignetteBuilder knitr

License GPL-2

LazyData true

RoxygenNote 6.1.1

git_url https://git.bioconductor.org/packages/pandaR

git_branch RELEASE_3_19

git_last_commit e103db2

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-10-06

Contents

`calcDegree` *Calculate regulatory network degree*

Description

Calculates the transcription factor out-degree or gene in-degree for the estimated panda regulatory network.

Usage

```
calcDegree(x, type = c("tf", "gene"), filter = FALSE, trim = FALSE,
  ...)
```

Arguments

`x`	An object of class "panda" or matrix
`type`	Character string - 'tf' or 'gene'
`filter`	Boolean to force negative degrees to zero
`trim`	Boolean to trim using topedges or not at a cutoff (weights become binary 1,0)
`...`	Options to be passed to topedges function

Examples

```
data(pandaToyData)
pandaRes <- panda(pandaToyData$motif,
          pandaToyData$expression,pandaToyData$ppi,hamming=.001,progress=TRUE)
calcDegree(pandaRes)
calcDegree(pandaRes,trim=TRUE,cutoff=1.5)

data(pandaResult)
calcDegree(pandaResult,type="tf",trim=TRUE,1000)
calcDegree(pandaResult,type="gene",trim=TRUE,1000)
```

---

calcDegreeDifference *Calculate difference in degrees*

Description

Calculates the transcription factor out-degree or gene in-degree for two different panda regulatory networks. This is useful in comparing networks from two phenotypes.

Usage

```
calcDegreeDifference(x, y, type = c("tf", "gene"), filter = FALSE,
  trim = FALSE, ...)
```

Arguments

x	An object of class "panda" or matrix
y	A second object of class "panda" or matrix
type	Character string - 'tf' or 'gene'
filter	Boolean to force negative degrees to zero
trim	Boolean to trim using topedges or not at a cutoff (weights become binary 1,0)
...	Options to be passed to topedges function

Examples

```
data(pandaToyData)
pandaRes <- panda(pandaToyData$motif,
          pandaToyData$expression,pandaToyData$ppi,hamming=.001,progress=TRUE)
pandaRes2 <- panda(pandaToyData$motif,
          pandaToyData$expression,pandaToyData$ppi,hamming=.1,progress=TRUE)
calcDegreeDifference(pandaRes,pandaRes2)
calcDegreeDifference(pandaRes,pandaRes2,trim=TRUE,cutoff=1.5)
```

importPandaMatlab *Panda Matlab importer*

Description

Imports the files from the exportPanda.m file.

Usage

```
importPandaMatlab(dir = getwd(), celldata = "celldata.dat")
```

Arguments

dir	Working directory to search for the numeric files.
celldata	Name of the 'celldata.dat' file.

Value

Two column vector of "regulator" and "target"

Examples

```
# determine gene degree
 pandaFiles = importPandaMatlab()
 indegree <- ddply(pandaFiles[,2:ncol(pandaFiles)], .(targer), numcolwise(sum))
 row.names(indegree) <- indegree[,1]
 indegree <- indegree[,-1]
 # to export the file
 networkfiles = list.files(pattern="numeric")
 write.table(indegree,paste("indegree_",networkfiles,sep=""),
             sep="\t",quote=F,row.names=T,col.names=T)
```

multiplot *Multiple plots*

Description

Multiple plot function as described in: http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/. If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE), then plot 1 will go in the upper left, 2 will go in the upper right, and 3 will go all the way across the bottom.

Usage

```
multiplot(..., plotlist = NULL, cols = 1, layout = NULL)
```

Arguments

...	ggplot objects can be passed in or to plotlist (as a list of ggplot objects).
plotlist	NULL - if the plotlist is null, the function will figure out the panel dimensions.
cols	Number of columns in layout.
layout	A matrix specifying the layout. If present, 'cols' is ignored.

---

panda *Passing Messages between Biological Networks to Refine Predicted Interactions*

Description

This function runs the PANDA algorithm

Usage

```
panda(motif, expr = NULL, ppi = NULL, alpha = 0.1, hamming = 0.001,
  iter = NA, output = c("regulatory", "coexpression", "cooperative"),
  zScale = TRUE, progress = FALSE, randomize = c("None",
  "within.gene", "by.gene"), cor.method = "pearson",
  scale.by.present = FALSE, edgelist = FALSE,
  remove.missing.ppi = FALSE, remove.missing.motif = FALSE,
  remove.missing.genes = FALSE, mode = "union")
```

Arguments

motif	A motif dataset, a data.frame, matrix or exprSet containing 3 columns. Each row describes an motif associated with a transcription factor (column 1) a gene (column 2) and a score (column 3) for the motif.
expr	An expression dataset, as a genes (rows) by samples (columns) data.frame
ppi	A Protein-Protein interaction dataset, a data.frame containing 3 columns. Each row describes a protein-protein interaction between transcription factor 1(column 1), transcription factor 2 (column 2) and a score (column 3) for the interaction.
alpha	value to be used for update variable, alpha (default=0.1)
hamming	value at which to terminate the process based on hamming distance (default 10^-3)
iter	sets the maximum number of iterations PANDA can run before exiting.
output	a vector containing which networks to return. Options include "regulatory", "coregulatory", "cooperative".
zScale	Boolean to indicate use of z-scores in output. False will use [0,1] scale.
progress	Boolean to indicate printing of output for algorithm progress.

`randomize` method by which to randomize gene expression matrix. Default "None". Must be one of "None", "within.gene", "by.genes". "within.gene" randomization scrambles each row of the gene expression matrix, "by.gene" scrambles gene labels.

`cor.method` Correlation method, default is "pearson".

`scale.by.present` Boolean to indicate scaling of correlations by percentage of positive samples.

`edgelist` Boolean to indicate if edge lists instead of matrices should be returned.

`remove.missing.ppi` Boolean to indicate whether TFs in the PPI but not in the motif data should be removed. Only when mode=='legacy'.

`remove.missing.motif` Boolean to indicate whether genes targeted in the motif data but not the expression data should be removed. Only when mode=='legacy'.

`remove.missing.genes` Boolean to indicate whether genes in the expression data but lacking information from the motif prior should be removed. Only when mode=='legacy'.

`mode` The data alignment mode. The mode 'union' takes the union of the genes in the expression matrix and the motif and the union of TFs in the ppi and motif and fills the matrics with zeros for nonintersecting TFs and gens, 'intersection' takes the intersection of genes and TFs and removes nonintersecting sets, 'legacy' is the old behavior with version 1.19.3. #' Parameters remove.missing.ppi, remove.missingmotif, remove.missing.genes work only with mode=='legacy'.

Value

An object of class "panda" containing matrices describing networks achieved by convergence with PANDA algorithm.
"regNet" is the regulatory network
"coregNet" is the coregulatory network
"coopNet" is the cooperative network

References

Glass K, Huttenhower C, Quackenbush J, Yuan GC. Passing Messages Between Biological Networks to Refine Predicted Interactions. PLoS One. 2013 May 318(5):e64832.

Examples

```
data(pandaToyData)
pandaRes <- panda(pandaToyData$motif,
          pandaToyData$expression,pandaToyData$ppi,hamming=.1,progress=TRUE)
```

pandaR *pandaR: Passing Messages between Biological Networks to Refine Predicted Interactions.*

Description

The PANDA approach is to model the regulatory network as a bipartite network and estimate edge weights based on the evidence that information from a particular transcription factor i is successfully being passed to a particular gene j. This package provides a straightforward tool for applying this established method.

pandaResult *Analysis result from PANDA algorithm on toy data*

Description

This data panda object resulting from running the PANDA algorithm on the supplied toy dataset.
data(pandaToyData) pandaResult <- panda(pandaToyData$motif, pandaToyData$expression,pandaToyData$ppi,hamming=.

Usage

```
pandaResult
```

Format

A panda object

Value

A panda object

References

Glass K, Huttenhower C, Quackenbush J, Yuan GC. Passing Messages Between Biological Networks to Refine Predicted Interactions. PLoS One. 2013 May 31;8(5):e64832.

pandaResultPairs *Analysis result from PANDA algorithm on toy data converted into pairs*

Description

This data panda object resulting from running the PANDA algorithm on the supplied toy dataset. The data consists of a matrix of TF, Gene, initial link, and final Z. data(pandaResultPairs)

Usage

```
pandaResultPairs
```

Format

A matrix

Value

A matrix

References

Glass K, Huttenhower C, Quackenbush J, Yuan GC. Passing Messages Between Biological Networks to Refine Predicted Interactions. PLoS One. 2013 May 31;8(5):e64832.

pandaToyData *Toy gene expression, motif, and ppi data*

Description

This data is a list containing three data.frames. The motif data.frame describes a set of pairwise connections where a specific known sequence motif of a transcription factor was found upstream of the corresponding gene. The expression data.frame is a set of 1000 gene expression levels measured across 50 samples. Finally, the ppi data.frame describes a set of known pairwise protein interactions.

Usage

```
pandaToyData
```

Format

A list containing 3 data.frames

Value

A list of length 3

References

Glass K, Huttenhower C, Quackenbush J, Yuan GC. Passing Messages Between Biological Networks to Refine Predicted Interactions. PLoS One. 2013 May 31;8(5):e64832.

`plotCommunityDetection` *Community detection plot*

Description

This function performs community detection on an undirected PANDA network. The function optionally returns the graph and community.

Usage

```
plotCommunityDetection(x, scaleEdge = 5, verbose = TRUE, ...)
```

Arguments

`x`	Toy PANDA output represented as a TF, Gene, and Score.
`scaleEdge`	Visualization parameter for the edges.
`verbose`	TRUE/FALSE - Report community structure.
`...`	Options for the plot function.

Value

Optionally return a list with the graph and community.

Examples

```
# start with some toy PANDA output
mat <- cbind(rep(1:5, each=10), rep(seq(11,20),5), sample(100, 50)/100)
x =plotCommunityDetection(mat)
str(x)

#example of very different edges
set.seed(1)
subst <- sample(50,10)
mat[subst, 3] <- subst
plotCommunityDetection(mat,scaleEdge=0.5)
```

plotGraph *Plot graph*

Description

plotGraph plots a bipartite graph

Usage

```
plotGraph(x)
```

Arguments

x	an object of class "panda"

Value

An matrix describing the subsetted bipartite network.

Examples

```
data(pandaToyData)
pandaRes <- panda(pandaToyData$motif,
           pandaToyData$expression,pandaToyData$ppi,hamming=.001,progress=TRUE)
topPandaRes <- topedges(pandaRes,1000)
subnet.pandaRes <- subnetwork(topPandaRes,c("AR","ARID3A","ELK1"))
plotGraph(subnet.pandaRes)

data(pandaResult)
topPandaRes <- topedges(pandaResult, 1000)
subnet.pandaRes <- subnetwork(topPandaRes,c("AR","ARID3A","ELK1"))
plotGraph(subnet.pandaRes)
```

plotZ *Comparison of Z scores between two PANDA runs*

Description

Given two PANDA objects with the same network structure, plot the Z-score comparison. The two PANDA objects should only differ in the gene expression used for the network constructions or other parameters.

Usage

```
plotZ(x, y, hex = TRUE, bins = 200, addLine = TRUE, rank = FALSE)
```

Arguments

x	PANDA object - output of the panda function.
y	PANDA object - second PANDA object.
hex	TRUE/FALSE - If TRUE, bin data points to avoid over plotting.
bins	Number of bins to use for plotting.
addLine	TRUE/FALSE - to add y=x line.
rank	TRUE/FALSE - If TRUE, plot rank of edge weights rather than weight values.

Value

ggplot comparing the Z-scores between the two networks.

Examples

```
data(pandaResult)
data(pandaToyData)
pandaRes <- pandaRes2 <- pandaResult
plotZ(pandaRes, pandaRes2)


panda.res1 <- with(pandaToyData, panda(motif, expression, ppi, hamming=1))
panda.res2 <- with(pandaToyData, panda(motif, expression +
                    rnorm(prod(dim(expression)),sd=5), ppi, hamming=1))
plotZ(panda.res1, panda.res2,addLine=FALSE)
```

---

plotZbyTF *Plot Z by TF out-degree quantiles*

Description

Generates a Z-score scatterplot for edges according to the TF outdegree in prior. The two PANDA objects should only differ in the gene expression used for the network constructions or other parameters.

Usage

```
plotZbyTF(x, y, motif, hasPrior = TRUE, cuts = 1, cols = 2)
```

Arguments

x	PANDA object - output of the panda function.
y	PANDA object - second PANDA object.
motif	Motif used to construct the networks.

hasPrior	TRUE/FALSE, If TRUE plots the edges that are given a weight > 0 in the motif, else plot those given a weight of 0
cuts	either a numeric vector of two or more unique cut points or a single number (greater than or equal to 2) giving the number of intervals into which 'x' is to be cut.
cols	Number of columns in layout.

Value

ggplot heatmapfor each TF, get outdegree in regulatory prior

Examples

```
data(pandaResult)
data(pandaToyData)
plotZbyTF(pandaResult,pandaResult, pandaToyData$motif, hasPrior=TRUE)
plotZbyTF(pandaResult,pandaResult, pandaToyData$motif, cuts=2)
```

---

print.panda *print.panda*

Description

summarizes the results of a PANDA analysis

Usage

```
## S3 method for class 'panda'
print(x, ...)
```

Arguments

x	an object of class "panda"
...	further arguments passed to or from other methods.

Value

Summary description of panda S4 object

Examples

```
data(pandaToyData)
panda.res <- panda(pandaToyData$motif,
          pandaToyData$expression,pandaToyData$ppi,hamming=.001,progress=TRUE)
print(panda.res)

data(pandaResult)
```

subnetwork *Subnetwork*

Description

subnetwork gets a bipartite network containing only the transcription factors or genes and their respective connections

Usage

```
subnetwork(x, nodes, subTf = TRUE)
```

Arguments

x	an object of class "panda"
nodes	character vector containing the transcription factor or gene labels to subset
subTf	an optional logical indicating whether to subset by transcription factor. Default is TRUE.

Value

An matrix describing the subsetted bipartite network.

Examples

```
data(pandaToyData)
pandaRes <- panda(pandaToyData$motif,
           pandaToyData$expression,pandaToyData$ppi,hamming=.001,progress=TRUE)
topPandaRes <- topedges(pandaRes,1000)
subnet.pandaRes <- subnetwork(topPandaRes,c("AR","ARID3A","ELK1"))

data(pandaResult)
topPandaRes <- topedges(pandaResult,1000)
subnetwork(topPandaRes,c("AR","ARID3A","ELK1"))
```

summary.panda *Summary.panda*

Description

summarizes the results of a PANDA analysis

Usage

```
summary.panda(object, ...)
```

Arguments

object	an object of class "panda"
...	further arguments passed to or from other methods.

Value

Summary description of panda S4 object

Examples

```
data(pandaToyData)
panda.res <- panda(pandaToyData$motif,
           pandaToyData$expression,pandaToyData$ppi,hamming=.001,progress=TRUE)
summary(panda.res)

data(pandaResult)
```

---

targetedGenes *targetedGenes*

Description

Gets a set of genes targeted by a specified transcription factor. This function can be applied to a graph that is not complete, subsetting the edges which have non-zero edge weight. See function topEdges for dichotomizing edgeweights.

Usage

```
targetedGenes(x, tfs)
```

Arguments

x	an object of class "panda"
tfs	transcription factors to query

Value

A vector of targeted genes

Examples

```
data(pandaToyData)
pandaRes <- panda(pandaToyData$motif,
           pandaToyData$expression,pandaToyData$ppi,hamming=.001)
topPandaRes <- topedges(pandaRes,1000)
targetedGenes(topPandaRes,c("AR","ELK1"))

data(pandaResult)
topPandaRes <- topedges(pandaResult,1000)
```

testMotif *Check motif*

Description

This function adds random false positive edges to the regulatory prior and will check if they become pruned.

Usage

```
testMotif(x, motif, expr, ppi, mode = c("augment", "remove"),
  prop = 0.05, seed = 1, ...)
```

Arguments

x	Model regulatory network.
motif	Motif used to construct the model regulatory network.
expr	Expression matrix used to construct model network.
ppi	PPI used to construct model regulatory network.
mode	a character string - either "augment" to add random edges or "remove" to remove random edges.
prop	numeric specifying number of edges to augment or remove from regulatory prior, as a proportion of the number of edges in the regulatory prior.
seed	Random seed.
...	Options for the panda function.

Value

ggplot heatmap list of indices of net corresponding to each TF

Examples

```
data(pandaToyData)
data(pandaResult)
regnet = slot(pandaResult,"regNet")
with(pandaToyData, testMotif(regnet, motif, mode="augment", expression, ppi, hamming=1))
```

topedges *Top edges*

Description

topedges gets a network from a panda obj with a specified cutoff based on magnitude of edgeweight.

Usage

```
topedges(x, count = NA, cutoff = 2, networks = c("coregulation",
  "cooperation", "regulatory"))
```

Arguments

x	an object of class "panda"
count	an optional integer indicating number of top edges to be included in regulatory network.
cutoff	an optional numeric indicating the z-score edge weight cutoff to be used to identify edges. Default is 2.0. Not used if count is not NA.
networks	an optional vector specifying which networks to be included in output. May be any combination of c("coregulation","cooperation","regulatory").

Value

An object of class "panda" containing binary matrices indicating the existence of an edge between two nodes. For regulatory network the matrix indicates an edge between a transcription factor (row) and a gene (column)

Examples

```
data(pandaToyData)
pandaRes <- panda(pandaToyData$motif,
          pandaToyData$expression,pandaToyData$ppi,hamming=.001,progress=TRUE)
topPandaRes <- topedges(pandaRes,1000)

data(pandaResult)
topPandaRes <- topedges(pandaResult,1000)
```

Index