Package 'GeneNetworkBuilder'

November 12, 2026

Type Package

Version 1.54.0

Title GeneNetworkBuilder: a bioconductor package for building regulatory network using ChIP-chip/ChIP-seq data and Gene Expression Data

Author Jianhong Ou, Haibo Liu, Heidi A Tissenbaum and Lihua Julie Zhu

Maintainer Jianhong Ou <jou@morgridge.org>

Imports plyr, graph, htmlwidgets, Rgraphviz, RCy3, rjson, XML, methods, grDevices, stats, graphics

Depends R (>= 2.15.1), Rcpp (>= 0.9.13)

Suggests RUnit, BiocGenerics, RBGL, knitr, shiny, STRINGdb, BiocStyle, magick, rmarkdown, org.Hs.eg.db

LinkingTo Rcpp

Description Appliation for discovering direct or indirect targets of transcription factors using ChIP-chip or ChIP-seq, and microarray or RNA-seq gene expression data. Inputting a list of genes of potential targets of one TF from ChIP-chip or ChIP-seq, and the gene expression results, GeneNetworkBuilder generates a regulatory network of the TF.

License GPL (>= 2)

Lazyload yes

LazyData true

biocViews Sequencing, Microarray, GraphAndNetwork

VignetteBuilder knitr

RoxygenNote 7.3.2

Encoding UTF-8

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

git_branch RELEASE_3_23

git_last_commit 4be7f39

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23
Date/Publication 2026-11-11

Contents

GeneNetworkBuilder-package . . . 2
browseNetwork . . . 3
browseNetwork-shiny . . . 4
buildNetwork . . . 4
ce.IDsMap . . . 5
ce.interactionmap . . . 5
ce.mapIDs . . . 6
ce.miRNA.map . . . 7
convertID . . . 7
cy3Network . . . 8
example.data . . . 9
exportNetwork . . . 10
filterNetwork . . . 11
hs.IDsMap . . . 12
hs.interactionmap . . . 13
hs.mapIDs . . . 13
hs.miRNA.map . . . 14
networkFromGenes . . . 15
polishNetwork . . . 15
saveXGMML . . . 17
subsetNetwork . . . 17
uniqueExprsData . . . 18

Index **19**

`GeneNetworkBuilder-package`
Build Regulatory Network from ChIP-chip/ChIP-seq and Expression Data

---

Description

Appliation for discovering direct or indirect targets of transcription factors using ChIP-chip or ChIP-seq, and microarray or RNA-seq gene expression data. Inputting a list of genes of potential targets of one TF from ChIP-chip or ChIP-seq, and the gene expression results, GeneNetworkBuilder generates a regulatory network of the TF.

Author(s)

Maintainer: Jianhong Ou Developer <jou@morgridge.org>

Authors:

- Lihua Julie Zhu Developer <Julie.Zhu@umassmed.edu>

browseNetwork *browse network*

Description

plot network generated by polishNetwork

Usage

```
browseNetwork(
  gR = graphNEL(),
  layoutType = c("fdp", "dot", "neato", "twopi", "circo"),
  width = NULL,
  height = NULL,
  maxNodes = 500,
  ...
)
```

Arguments

gR	an object of graphNEL
layoutType	layout type. see GraphvizLayouts
width	width of the figure
height	height of the figure
maxNodes	max nodes number to plot. Because if there are two many nodes, the running time will be too long.
...	parameters used by GraphvizLayouts

Value

An object of class htmlwidget that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

Examples

```
data("ce.miRNA.map")
data("example.data")
data("ce.interactionmap")
data("ce.IDsMap")
sifNetwork<-buildNetwork(example.data$ce.bind, ce.interactionmap, level=2)
cifNetwork<-filterNetwork(rootgene=ce.IDsMap["DAF-16"], sifNetwork=sifNetwork,
  exprsData=uniqueExprsData(example.data$ce.exprData, "Max", condenseName='logFC'),
  mergeBy="symbols",
  miRNAlist=as.character(ce.miRNA.map[ , 1]), tolerance=1)
gR<-polishNetwork(cifNetwork)
browseNetwork(gR)
```

browseNetwork-shiny *Shiny bindings for browseNetwork*

Description

Output and render functions for using browseNetwork within Shiny applications and interactive Rmd documents.

Usage

```
browseNetworkOutput(outputId, width = "100%", height = "400px")

renderBrowseNetwork(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a browseNetwork
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

buildNetwork *construct the regulatory network*

Description

Get all the connections of interesting genes from regulatory map.

Usage

```
buildNetwork(TFbindingTable, interactionmap, level = 3)
```

Arguments

TFbindingTable	a matrix or data.frame with interesting genes. Column names must be 'from', 'to'
interactionmap	Transcription regulatory map. Column names of interactionmap must be 'from','to'
level	Depth of node path

Value

a dataframe or matrix of all the connections of interesting genes

Examples

```
data("ce.interactionmap")
data("example.data")
xx<-buildNetwork(example.data$ce.bind, ce.interactionmap, level=2)
```

`ce.IDsMap` *C.elegns gene name to wormbase identifier map*

Description

map file for converting gene name or sequence name of *Caenorhabditis elegans* to wormbase identifier

Usage

```
ce.IDsMap
```

Format

character vector

Details

character vecotr with gene name or sequence name as names and wormbase identifier as values.

Source

http://www.wormbase.org/

Examples

```
data(ce.IDsMap)
head(ce.IDsMap)
```

`ce.interactionmap` *transcript regulatory map of* Caenorhabditis elegans

Description

transcript regulatory map of *Caenorhabditis elegans*

Usage

```
ce.interactionmap
```

Format

dataframe

Details

transcript regulatory map of *Caenorhabditis elegans* is generated using databases edgedb and microCosm Targets.

Source

http://edgedb.umassmed.edu, http://www.ebi.ac.uk/enright-srv/microcosm/htdocs/targets/v5/

Examples

```
data(ce.interactionmap)
head(ce.interactionmap)
```

`ce.mapIDs` *map file for converting from wormbase identifier to* Caenorhabditis elegans *gene name*

Description

map file for converting from wormbase identifier to *Caenorhabditis elegans* gene name

Usage

```
ce.mapIDs
```

Format

character vector

Details

character vecotr with wormbase identifier as names and gene name as values.

Source

http://www.wormbase.org/

Examples

```
data(ce.mapIDs)
head(ce.mapIDs)
```

ce.miRNA.map *micro RNA of* Caenorhabditis elegans

Description

micro RNA of *Caenorhabditis elegans*

Usage

```
ce.miRNA.map
```

Format

dataframe

Details

The first column is wormbase identifier. And the second column is miRNA names.

Source

http://www.mirbase.org/

Examples

```
data(ce.miRNA.map)
head(ce.miRNA.map)
```

convertID *convert gene IDs by id map*

Description

For same gene, there are multple gene alias. In order to eliminate the possibility of missing any connections, convert the gene symbols to unique gene ids is important. This function can convert the gene symbols to unique ids and convert it back according a giving map.

Usage

```
convertID(x, IDsMap, ByName = c("from", "to"))
```

Arguments

x	a matrix or dataframe contain the columns to be converted.
IDsMap	a character vector of the identifier map
ByName	the column names to be converted

Value

a matrix or dataframe with converted gene IDs

Examples

```
data("ce.IDsMap")
bind<-cbind(from="daf-16", to=c("fkh-7", "hlh-13", "mxl-3", "nhr-3", "lfi-1"))
convertID(toupper(bind), ce.IDsMap, ByName=c("from", "to"))
```

---

cy3Network *visualize the network by Cy3*

Description

Using RCy3 to visualize the network.

Usage

```
cy3Network(
  gR = graphNEL(),
  ...,
  stringify = FALSE,
  species = "Homo sapiens",
  style = "Marquee",
  widths = c(0.25, 5)
)
```

Arguments

gR	an object of graphNEL
...	parameters will be passed to createNetworkFromGraph
stringify	Run STRINGify or not
species	if stringify is TRUE, teh species will be used to retreive the nodes and edges properties.
style	The default style when create the network
widths	The link width range.

Value

The network SUID.

Examples

```
data("ce.miRNA.map")
data("example.data")
data("ce.interactionmap")
data("ce.IDsMap")
sifNetwork<-buildNetwork(example.data$ce.bind, ce.interactionmap, level=2)
cifNetwork<-filterNetwork(rootgene=ce.IDsMap["DAF-16"], sifNetwork=sifNetwork,
  exprsData=uniqueExprsData(example.data$ce.exprData, "Max", condenseName='logFC'),
  mergeBy="symbols",
  miRNAlist=as.character(ce.miRNA.map[ , 1]), tolerance=1)
gR<-polishNetwork(cifNetwork)
if(interactive()){
  cy3Network(gR)
}
```

---

example.data *example datasets for documentation*

Description

example.data is a data list of example datasets. There is a dataset example.microarrayData, which is the example of gene expression data of a gene-chip result of *C.elegans*. Dataset example.data$ce.bind is a TF binding matrix of ChIP-chip experiment of *C.elegans*. Dataset example.data$cd.exprData is expression data of a gene-chip result of *C.elegans*. Dataset example.data$hs.bind is a TF binding matrix of ChIP-chip experiment of *H.sapiens*. Dataset example.data$hs.exprData is expression data of a combination of a gene-chip result and a RNA-SEQ result of *H.sapiens*.

Usage

```
example.data
```

Format

dataframe

Details

The dataset example.microarrayData contains columns: ID, logFC, AveExpr, t, P.Value, adj.P.Val, B, genes and symbols. The columns of ID, logFC and symbols are required by GeneNetwork-Builder. The dataset example.data$hs.bind contains columns: ID, symbols, logFC and P.Value. The dataset example.data$hs.exprData contains columns: from and to.

Examples

```
data(example.data)
names(example.data)
head(example.data$example.microarrayData)
head(example.data$ce.bind)
```

```
head(example.data$ce.exprData)
head(example.data$hs.bind)
head(example.data$hs.exprData)
```

`exportNetwork` *Save network in various formats*

Description

Save graph into HTML, json or xgmml format.

Usage

```
exportNetwork(network, file, format = c("HTML", "json", "XGMML"), ...)
```

Arguments

`network`	output of browseNetwork
`file`	Name of the file to save to.
`format`	type in which graph shall be saved. Could be one of HTML, json or XGMML.
`...`	Parameter could be used by saveWidget for HTML or writeLines for json or saveXML for XGMML.

Examples

```
data("ce.miRNA.map")
data("example.data")
data("ce.interactionmap")
data("ce.IDsMap")
sifNetwork<-buildNetwork(example.data$ce.bind, ce.interactionmap, level=2)
cifNetwork<-filterNetwork(rootgene=ce.IDsMap["DAF-16"], sifNetwork=sifNetwork,
  exprsData=uniqueExprsData(example.data$ce.exprData, "Max", condenseName='logFC'),
  mergeBy="symbols",
  miRNAlist=as.character(ce.miRNA.map[ , 1]), tolerance=1)
gR<-polishNetwork(cifNetwork)
network <- browseNetwork(gR)
exportNetwork(network, "sample.html")
```

`filterNetwork` *filter the regulatory network table by expression profile*

Description

verify every nodes in the regulatory network by expression profile

Usage

```
filterNetwork(
  rootgene,
  sifNetwork,
  exprsData,
  mergeBy = "symbols",
  miRNAlist,
  remove_miRNA = FALSE,
  tolerance = 0,
  cutoffPVal = 0.01,
  cutoffLFC = 0.5,
  minify = TRUE,
  miRNAtol = FALSE
)
```

Arguments

`rootgene`	name of root gene. It must be the ID used in xx regulatory network
`sifNetwork`	Transcription regulatory network table. Column names of xx must be 'from','to'
`exprsData`	dataset of expression comparison data, which should contain column logFC and column given by exprsDataByName
`mergeBy`	The column name contains ID information used to merge with 'to' column of sifNetwork in exprsData
`miRNAlist`	vector of microRNA ids.
`remove_miRNA`	remove miRNA from the network or not. Bool value, TRUE or FALSE
`tolerance`	maximum number of unverified nodes in each path
`cutoffPVal`	cutoff p value of valid differential expressed gene/miRNA
`cutoffLFC`	cutoff log fold change value of a valid differential expressed gene/miRNA
`minify`	Only keep the best path if multiple paths exists for single node? Bool value, TRUE or FALSE
`miRNAtol`	take miRNA expression into account for tolerance calculation. Bool value, TRUE or FALSE

Value

a dataframe of filtered regulatory network by expression profile

Examples

```
data("ce.miRNA.map")
data("example.data")
data("ce.interactionmap")
data("ce.IDsMap")
sifNetwork<-buildNetwork(example.data$ce.bind, ce.interactionmap, level=2)
cifNetwork<-filterNetwork(rootgene=ce.IDsMap["DAF-16"], sifNetwork=sifNetwork,
  exprsData=uniqueExprsData(example.data$ce.exprData, "Max", condenseName='logFC'),
  mergeBy="symbols",
  miRNAlist=as.character(ce.miRNA.map[ , 1]), tolerance=1)
```

---

hs.IDsMap *map file for converting gene name or sequence name of* Homo sapiens *to Entrez identifier*

Description

map file for converting gene name or sequence name of *Homo sapiens* to Entrez identifier

Usage

```
hs.IDsMap
```

Format

character vector

Details

character vecotr with gene name as names and Entrez identifier as values.

Examples

```
data(hs.IDsMap)
head(hs.IDsMap)
```

hs.interactionmap *transcript regulation map of* Homo sapiens

Description

transcript regulation map of *Homo sapiens*

Usage

```
hs.interactionmap
```

Format

dataframe

Details

transcript regulatory map of *Homo sapiens* is generated using databases FANTOM, mirGen and microCosm Targets.

Source

http://fantom.gsc.riken.jp/5/, http://www.ebi.ac.uk/enright-srv/microcosm/htdocs/targets/v5/, http://carolina.imis.athena-innovation.gr/diana_tools/web/index.php

Examples

```
data(hs.interactionmap)
head(hs.interactionmap)
```

hs.mapIDs *map file for converting from Entrez identifier to* Homo sapiens *gene name*

Description

map file for converting from Entrez identifier to *Homo sapiens* gene name

Usage

```
hs.mapIDs
```

Format

character vector

Details

character vecotr with Entrez identifier as names and gene name as values.

Examples

```
data(hs.mapIDs)
head(hs.mapIDs)
```

`hs.miRNA.map` *micro RNA of* Homo sapiens

Description

micro RNA of *Homo sapiens*

Usage

```
hs.miRNA.map
```

Format

dataframe

Details

The first column is entrez identifier. And the second column is miRNA names.

Source

http://www.mirbase.org/

Examples

```
data(hs.miRNA.map)
head(hs.miRNA.map)
```

networkFromGenes *Build network by a list of given genes*

Description

By providing a list of given genes, build a network for input of filterNetwork.

Usage

```
networkFromGenes(genes, interactionmap, level = 3, unrooted = FALSE)
```

Arguments

genes	A vector of character for interested genes.
interactionmap	Transcription regulatory map. Column names of interactionmap must be 'from','to'
level	Depth of node path
unrooted	Return unrooted regulatory network table or not.

Value

a list with elements: rootgene: The nodes with maximal connections. sifNetwork: Transcription regulatory network table.

Examples

```
data("ce.interactionmap")
data("example.data")
genes <- as.character(example.data$ce.bind$from)
xx<-networkFromGenes(example.data$ce.bind, ce.interactionmap, level=2)
```

polishNetwork *generate an object of grahpNEL to represent the regulation network*

Description

generate an object of grahpNEL to represent the regulation network. Each node will has three attributes: size, borderColor and fill. The size will be mapped to the length of its edges. The node fill color will be mapped to logFC.

Usage

```
polishNetwork(
  cifNetwork,
  nodeData,
  nodesDefaultSize = 48,
  nodecolor = colorRampPalette(c("green", "yellow", "red"))(5),
  nodeBg = "white",
  nodeBorderColor = list(gene = "darkgreen", miRNA = "darkblue"),
  edgeWeight = NA,
  edgelwd = 0.5,
  ...
)
```

Arguments

cifNetwork	dataframe used to draw network graph. column names of cifNetwork must contain 'from', 'to', 'logFC' and 'miRNA'
nodeData	The node data. If it is not provide, node data will be retrieved from cifNetwork for the 'to' nodes.
nodesDefaultSize	nodes default size
nodecolor	a character vector of color set. The node color will be mapped to color set by log fold change. Or the column names for the colors.
nodeBg	background of node
nodeBorderColor	a list of broder node color set. nodeBorderColor's element must be gene and miRNA
edgeWeight	the weight of edge. It can be a column name of cifNetwork.
edgelwd	the default width of edge. If edgeWeight is set, the edgelwd will be mapped to the edgeWeight.
...	any parameters can be passed to graph.par

Value

An object of graphNEL class of the network

Examples

```
data("ce.miRNA.map")
data("example.data")
data("ce.interactionmap")
data("ce.IDsMap")
sifNetwork<-buildNetwork(example.data$ce.bind, ce.interactionmap, level=2)
cifNetwork<-filterNetwork(rootgene=ce.IDsMap["DAF-16"], sifNetwork=sifNetwork,
  exprsData=uniqueExprsData(example.data$ce.exprData, "Max", condenseName='logFC'),
  mergeBy="symbols",
  miRNAlist=as.character(ce.miRNA.map[ , 1]), tolerance=1)
```

```
gR<-polishNetwork(cifNetwork)
## browseNetwork(gR)
```

---

saveXGMML *Save network as xgmml*

Description

Save graph into xgmml format.

Usage

```
saveXGMML(network, file, ...)
```

Arguments

network	output of browseNetwork
file	Name of the file to save to.
...	Parameter could be used by saveXML

---

subsetNetwork *Subset a polished network*

Description

Subset the output of polishNetwork by a list of nodes name

Usage

```
subsetNetwork(graph, genes)
```

Arguments

graph	A graphNEL object. The output of polishNetwork.
genes	A list of nodes names

Value

An object of graph.

Examples

```
library(graph)
set.seed(123)
g1 <- randomEGraph(LETTERS[seq.int(15)], edges=100)
g1 <- subsetNetwork(g1, LETTERS[seq.int(5)])
plot(g1)
```

uniqueExprsData *unique the microarray data*

Description

get unique the microarray data for each gene id.

Usage

```
uniqueExprsData(exprsData, method = "Max", condenseName = "logFC")
```

Arguments

exprsData	dataset of expression comparison data
method	method must be Max, Median or Min
condenseName	column names to be condensed

Value

a dataframe of expression data without duplicates

Examples

```
data("example.data")
example.microarrayData<-uniqueExprsData(example.data$example.microarrayData,
                                        method="Max", condenseName='logFC')
```

Index

∗ **IO**
    exportNetwork, 10
    saveXGMML, 17
∗ **convert**
    convertID, 7
∗ **data**
    ce.IDsMap, 5
    ce.interactionmap, 5
    ce.mapIDs, 6
    ce.miRNA.map, 7
    example.data, 9
    hs.IDsMap, 12
    hs.interactionmap, 13
    hs.mapIDs, 13
    hs.miRNA.map, 14
∗ **network**
    buildNetwork, 4
    filterNetwork, 11
    networkFromGenes, 15
    polishNetwork, 15
    uniqueExprsData, 18
∗ **plot**
    browseNetwork, 3

browseNetwork, 3, *10*, *17*
browseNetwork-shiny, 4
browseNetworkOutput
        *(*browseNetwork-shiny*)*, 4
buildNetwork, 4

ce.IDsMap, 5
ce.interactionmap, 5
ce.mapIDs, 6
ce.miRNA.map, 7
convertID, 7
createNetworkFromGraph, *8*
cy3Network, 8

example.data, 9
exportNetwork, 10

filterNetwork, 11

GeneNetworkBuilder
        *(*GeneNetworkBuilder-package*)*, 2
GeneNetworkBuilder-package, 2
graph.par, *16*
graphNEL, *3*, *8*
GraphvizLayouts, *3*

hs.IDsMap, 12
hs.interactionmap, 13
hs.mapIDs, 13
hs.miRNA.map, 14

networkFromGenes, 15

polishNetwork, *3*, 15

renderBrowseNetwork
        *(*browseNetwork-shiny*)*, 4

saveWidget, *10*
saveXGMML, 17
saveXML, *10*
subsetNetwork, 17

uniqueExprsData, 18