

# Package ‘GeneNetworkBuilder’

October 17, 2024

**Type** Package

**Version** 1.46.0

**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 <jianhong.ou@duke.edu>

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

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

**Suggests** RUnit, BiocGenerics, RBGL, knitr, simpIntLists, 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** 7.2.1

**Encoding** UTF-8

**git\_url** <https://git.bioconductor.org/packages/GeneNetworkBuilder>

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** 13bd7a1

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-10-16

## 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
example.data . . . . .	8
exportNetwork . . . . .	9
filterNetwork . . . . .	9
hs.IDsMap . . . . .	11
hs.interactionmap . . . . .	11
hs.mapIDs . . . . .	12
hs.miRNA.map . . . . .	13
networkFromGenes . . . . .	13
polishNetwork . . . . .	14
saveXGMMML . . . . .	15
subsetNetwork . . . . .	16
uniqueExprsData . . . . .	16
<b>Index</b>	<b>18</b>

---

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 <jianhong.ou@duke.edu>

Authors:

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

---

browseNetwork	<i>browse network</i>
---------------	-----------------------

---

## 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 <a href="#">graphNEL</a>
layoutType	layout type. see <a href="#">GraphvizLayouts</a>
width	width of the figure
height	height of the figure
maxNodes	max nodes number to plot. Because if there are too many nodes, the running time will be too long.
...	parameters used by <a href="#">GraphvizLayouts</a>

## 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	<i>C.elegns gene name to wormbase identifier map</i>
-----------	--

---

**Description**

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

**Usage**

```
ce.IDsMap
```

**Format**

character vector

**Details**

character vector 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	<i>transcript regulatory map of Caenorhabditis elegans</i>
-------------------	--

---

**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	<i>map file for converting from wormbase identifier to Caenorhabditis elegans gene name</i>
-----------	---

---

**Description**

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

**Usage**

```
ce.mapIDs
```

**Format**

character vector

**Details**

character vector 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	<i>micro RNA of Caenorhabditis elegans</i>
--------------	--

---

**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	<i>convert gene IDs by id map</i>
-----------	-----------------------------------

---

**Description**

For same gene, there are multiple 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"))
```

---

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	<i>Save network in various formats</i>
---------------	--

---

**Description**

Save graph into HTML, json or xgmml format.

**Usage**

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

**Arguments**

network	output of <a href="#">browseNetwork</a>
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 <a href="#">saveWidget</a> for HTML or <a href="#">writeLines</a> for json or <a href="#">saveXML</a> 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	<i>filter the regulatory network table by expression profile</i>
---------------	--

---

**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	<i>map file for converting gene name or sequence name of Homo sapiens to Entrez identifier</i>
-----------	--

---

### Description

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

### Usage

```
hs.IDsMap
```

### Format

character vector

### Details

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

### Examples

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

---

hs.interactionmap	<i>transcript regulation map of Homo sapiens</i>
-------------------	--

---

### 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](http://carolina.imis.athena-innovation.gr/diana_tools/web/index.php)

**Examples**

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

---

hs.mapIDs	<i>map file for converting from Entrez identifier to Homo sapiens gene name</i>
-----------	---

---

**Description**

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

**Usage**

```
hs.mapIDs
```

**Format**

character vector

**Details**

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

**Examples**

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

---

hs.miRNA.map	<i>micro RNA of Homo sapiens</i>
--------------	----------------------------------

---

**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	<i>Build network by a list of given genes</i>
------------------	---

---

**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	<i>generate an object of graphNEL to represent the regulation network</i>
---------------	---

---

**Description**

generate an object of graphNEL to represent the regulation network. Each node will has three attributes: size, borderColor and fill.

**Usage**

```
polishNetwork(
  cifNetwork,
  nodesDefaultSize = 48,
  useLogFCAsWeight = FALSE,
  nodecolor = colorRampPalette(c("green", "yellow", "red"))(5),
  nodeBg = "white",
  nodeBorderColor = list(gene = "darkgreen", miRNA = "darkblue"),
  edgelwd = 0.25,
  ...
)
```

**Arguments**

cifNetwork	dataframe used to draw network graph. column names of cifNetwork must contain 'from', 'to', 'logFC' and 'miRNA'
nodesDefaultSize	nodes default size
useLogFCAsWeight	how to determine the weights for each nodes. If TRUE, use logFC value as weight. If FALSE, use constant 1 as weight.
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 border node color set. nodeBorderColor's element must be gene and miRNA

edgewidth      the width of edge  
 ...            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	<i>Save network as xgmml</i>
-----------	------------------------------

---

### 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	<i>Subset a polished network</i>
---------------	----------------------------------

---

**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	<i>unique the microarray data</i>
-----------------	-----------------------------------

---

**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, 9
  - saveXGML, 15
- \* **convert**
  - convertID, 7
- \* **data**
  - ce.IDsMap, 5
  - ce.interactionmap, 5
  - ce.mapIDs, 6
  - ce.miRNA.map, 7
  - example.data, 8
  - hs.IDsMap, 11
  - hs.interactionmap, 11
  - hs.mapIDs, 12
  - hs.miRNA.map, 13
- \* **network**
  - buildNetwork, 4
  - filterNetwork, 9
  - networkFromGenes, 13
  - polishNetwork, 14
  - uniqueExprsData, 16
- \* **plot**
  - browseNetwork, 3
- browseNetwork, 3, 9, 15
- browseNetwork-shiny, 4
- browseNetworkOutput
  - (browseNetwork-shiny), 4
- buildNetwork, 4
- ce.IDsMap, 5
- ce.interactionmap, 5
- ce.mapIDs, 6
- ce.miRNA.map, 7
- convertID, 7
- example.data, 8
- exportNetwork, 9
- filterNetwork, 9
- GeneNetworkBuilder
  - (GeneNetworkBuilder-package), 2
- GeneNetworkBuilder-package, 2
- graph.par, 15
- graphNEL, 3
- GraphvizLayouts, 3
- hs.IDsMap, 11
- hs.interactionmap, 11
- hs.mapIDs, 12
- hs.miRNA.map, 13
- networkFromGenes, 13
- polishNetwork, 3, 14
- renderBrowseNetwork
  - (browseNetwork-shiny), 4
- saveWidget, 9
- saveXGML, 15
- saveXML, 9
- subsetNetwork, 16
- uniqueExprsData, 16