Package 'ceRNAnetsim'

November 8, 2026

Type Package

Title Regulation Simulator of Interaction between miRNA and Competing RNAs (ceRNA)

Version 1.24.0

Description This package simulates regulations of ceRNA (Competing Endogenous) expression levels after a expression level change in one or more miRNA/mRNAs. The methodolgy adopted by the package has potential to incorparate any ceRNA (circRNA, lincRNA, etc.) into miRNA:target interaction network. The package basically distributes miRNA expression over available ceRNAs where each ceRNA attracks miRNAs proportional to its amount. But, the package can utilize multiple parameters that modify miRNA effect on its target (seed type, binding energy, binding location, etc.). The functions handle the given dataset as graph object and the processes progress via edge and node variables.

License GPL (>= 3.0)

URL https://github.com/selcenari/ceRNAnetsim

BugReports https://github.com/selcenari/ceRNAnetsim/issues

Depends R (>= 4.0.0), dplyr, tidygraph

Imports furrr, rlang, tibble, ggplot2, ggraph, igraph, purrr, tidyr, future, stats

Suggests knitr, png, rmarkdown, testthat, covr

VignetteBuilder knitr

biocViews NetworkInference, SystemsBiology, Network, GraphAndNetwork, Transcriptomics

Encoding UTF-8

LazyData false

RoxygenNote 7.1.2

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

git_branch RELEASE_3_23

git_last_commit 809533e

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-08

Author Selcen Ari Yuka [aut, cre] (ORCID:
<https://orcid.org/0000-0002-0028-2453>),
Alper Yilmaz [aut] (ORCID: <https://orcid.org/0000-0002-8827-4887>)

Maintainer Selcen Ari Yuka <selcenarii@gmail.com>

Contents

calc_perturbation . . . 3
find_affected_nodes . . . 4
find_iteration . . . 5
find_node_perturbation . . . 6
find_targeting_nodes . . . 7
gene_knockdown . . . 8
huge_example . . . 9
midsamp . . . 9
midsamp_new_counts . . . 10
minsamp . . . 10
mirtarbasegene . . . 11
new_counts . . . 11
normalize . . . 12
prepare_rhs . . . 12
prepare_rhs_once . . . 13
priming_graph . . . 13
simulate . . . 14
simulate_vis . . . 15
TCGA_E9_A1N5_mirnanormal . . . 17
TCGA_E9_A1N5_mirnatumor . . . 17
TCGA_E9_A1N5_normal . . . 18
TCGA_E9_A1N5_tumor . . . 19
update_how . . . 19
update_nodes . . . 20
update_variables . . . 21
vis_graph . . . 22

Index **24**

`calc_perturbation` *Calculates average expression changes of all nodes except trigger and finds the perturbed node count for a given node.*

Description

Calculates average expression changes of all nodes except trigger and finds the perturbed node count for a given node.

Usage

```
calc_perturbation(input_graph, node_name, how = 1, cycle = 1, limit = 0)
```

Arguments

`input_graph`	the graph object that was processed with priming graph in previous step.
`node_name`	The node that is trigger for simulation.
`how`	The change of count of the given node in terms of fold change.
`cycle`	The iteration of simulation.
`limit`	The minimum fold change which can be taken into account for perturbation calculation on all nodes in terms of percentage.

Details

calc_perturbation calculates mean expression changes of elements except trigger after the change in the network in terms of percentage. It also calculates the number of nodes that have expression changes after the change occur in the network. The function determines the perturbation efficiency and number of perturbed nodes after given change with how, cycle and limit parameter.

Value

a tibble with two columns, the perturbation efficiency and number of perturbed nodes.

Examples

```
data('minsamp')

minsamp%>%
   priming_graph(competing_count = Competing_expression,
       miRNA_count = miRNA_expression)%>%
   calc_perturbation('Gene6', how= 3, cycle = 4)

 minsamp%>%
   priming_graph(competing_count = Competing_expression, miRNA_count = miRNA_expression,
       aff_factor = c(energy,seed_type), deg_factor = region)%>%
   calc_perturbation('Gene6',3, cycle = 4)
```

`find_affected_nodes` *Finds top affected nodes for perturbation from a particular node*

Description

Finds top affected nodes for perturbation from a particular node

Usage

```
find_affected_nodes(
  input_graph,
  node_name,
  how = 1,
  cycle = 1,
  limit = 0,
  top = 5
)
```

Arguments

`input_graph`	The graph object that was processed with priming_graph function.
`node_name`	The node to trigger perturbations.
`how`	The change of count (expression) of the given node in terms of fold change.
`cycle`	The iteration of simulation.
`limit`	The minimum fold change which can be taken into account for perturbation calculation on all nodes in terms of percentage.
`top`	Determines how many nodes most affected will be listed.

Details

Lists the most affected nodes after perturbation initiated from a particular node. In the background, it compares the calculated values after the simulation with their initial values.

Value

It gives a tibble form dataset that includes perturbation node, affected nodes and changes of them.

Examples

```
data('midsamp')

midsamp%>%
priming_graph(competing_count = Gene_expression,
              miRNA_count = miRNA_expression)%>%
find_affected_nodes(node_name = "Gene1",
                    how = 2,
                    cycle = 2,
```

```
                    top = 2)
```

`find_iteration` *Finds the iteration which provides maximum affected node number*

Description

searches the iteration that provides maximum affected node number. The user defines a symbolic iteration with .iter. The function calculates the number of affected nodes for each iteration and then selects the iteration that has maximum affected nodes' number.

Usage

```
find_iteration(df, limit = 0.1, plot = FALSE)
```

Arguments

`df`	A tbl graph that includes the miRNA and competing targets triggered and simulated for number of cycles.
`limit`	The minimum amount of change of any node.
`plot`	If TRUE, returns a plot.

Value

It gives an iteration number to use in simulate() function.

Examples

```
data('midsamp')

midsamp %>%
  priming_graph(Gene_expression, miRNA_expression) %>%
  update_how('Gene2',2) %>%
  simulate(10) %>%
  find_iteration(limit=0)
```

`find_node_perturbation`

Calculates average expression changes of all (or specified) nodes except trigger and finds the perturbed node count for all (or specified) nodes in system.

Description

Calculates average expression changes of all (or specified) nodes except trigger and finds the perturbed node count for all (or specified) nodes in system.

Usage

```
find_node_perturbation(input_graph, how = 2, cycle = 1, limit = 0, fast = 0)
```

Arguments

`input_graph`	The graph object that was processed with priming_graph function.
`how`	The change of count (expression) of the given node in terms of fold change.
`cycle`	The iteration of simulation.
`limit`	The minimum fold change which can be taken into account for perturbation calculation on all nodes in terms of percentage.
`fast`	specifies percentage of affected target in target expression. For example, if fast = 1, the nodes that are affected from miRNA repression activity more than one percent of their expression is determined as subgraph.

Details

find_node_perturbation calculates mean expression changes of elements after the change in the network in terms of percentage. It also calculates the number of nodes that have expression changes after the change occur in the network. The outputs of the function are the perturbation efficiency and perturbed count of nodes for each nodes.

Value

It gives a tibble form dataset that includes node names, perturbation efficiency and perturbed count of nodes.

Examples

```
data('minsamp')
data('midsamp')

 minsamp%>%
 priming_graph(competing_count = Competing_expression, miRNA_count = miRNA_expression)%>%
  find_node_perturbation()%>%
  select(name, perturbation_efficiency, perturbed_count)
```

```
minsamp%>%
 priming_graph(competing_count = Competing_expression, miRNA_count = miRNA_expression,
   aff_factor = c(energy,seed_type), deg_factor = region)%>%
 find_node_perturbation(how = 3, cycle = 4)%>%
 select(name, perturbation_efficiency, perturbed_count)

midsamp%>%
 priming_graph(competing_count = Gene_expression, miRNA_count = miRNA_expression)%>%
 find_node_perturbation(how = 2, cycle= 3, limit=1, fast = 5)%>%
 select(name, perturbation_efficiency, perturbed_count)
```

`find_targeting_nodes` *Finds potential affecting node for given particular target.*

Description

Finds potential affecting node for given particular target.

Usage

```
find_targeting_nodes(
  input_graph,
  how = 2,
  cycle = 1,
  limit = 0,
  fast = 0,
  top = 5,
  target = NULL
)
```

Arguments

`input_graph`	The graph object that was processed with priming_graph function.
`how`	The change of count (expression) of the given node in terms of fold change.
`cycle`	The iteration of simulation.
`limit`	The minimum fold change which can be taken into account for perturbation calculation on all nodes in terms of percentage.
`fast`	specifies percentage of affected target in target expression. For example, if fast = 1, the nodes that are affected from miRNA repression activity more than one percent of their expression is determined as subgraph.
`top`	Determines how many nodes most affected will be evaluated.
`target`	The target node in which is being investigated.

Details

Lists potential targeting nodes by running find_affected_nodes function for all nodes in network.

Value

It gives a tibble form dataset that includes parturbation node (source) and change in count of targeting node

Examples

```
data('midsamp')

midsamp%>%
priming_graph(competing_count = Gene_expression,
              miRNA_count = miRNA_expression)%>%
find_targeting_nodes(how = 2,
                     cycle = 2,
                     target = "Gene1",
                     top = 2)
```

---

gene_knockdown *Knocks down given node.*

Description

Knocks down given node.

Usage

```
gene_knockdown(input_graph, node_name)
```

Arguments

input_graph	The graph object that processed in previous step/s.
node_name	The name of the node whose count is to be knocked down.

Details

knocks down a given gene target.

Value

the graph object.

huge_example *huge example*

Description

A sample dataset which is utilised through integration of TCGA_E9_A1N5_normal, TCGA_E9_A1N5_mirnanormal and high-throughput experimental miRNA:gene dataset.

Format

A data frame with 7 variables and 26176 observation:

competing name of gene
miRNA name of miRNA
competing_counts Expression values of competing element (gene)
mirnaexpression_normal Expression value of miRNA elements in normal tissue
Energy Energy of miRNA:target binding
region_effect Coefficient for efficiency of location on target
seed_type_effect Coefficient for efficiency of seed sequence of miRNA:target interaction

Source

Dataset was integrated by us.

midsamp *midsamp*

Description

middle sized sample dataset

Format

A data frame with 7 variables and 26 observation of them:

Genes symbol of gene
miRNAs symol of miRNA
Gene_expression Expression values of competing gene
miRNA_expression Expression value of miRNA
seeds Coefficient for efficiency of seed type of miRNA:target interaction
targeting_region Coefficient for efficiency of location on target
Energy Energy of miRNA:target binding

Source

Dataset was created by us.

midsamp_new_counts *midsamp_new_counts*

Description

includes new expression values for middle sized sample dataset

Format

A data frame with 4 variables and 26 observation of them:

Competing symbol of gene

miRNA symol of miRNA

Competing_count Expression values of competing gene

miRNA_count Expression value of miRNA

Source

Dataset was created by us.

minsamp *minsamp*

Description

minimal sample dataset

Format

A data frame with 7 variables and 7 observation of them:

competing symbol of gene

miRNA symol of miRNA

Competing_expression Expression values of competing gene

miRNA_expression Expression value of miRNA

seed_type Coefficient for efficiency of seed sequence of miRNA:target interaction

region Coefficient for efficiency of location on target

energy Energy of miRNA:target binding

Source

Dataset was created by us.

`mirtarbasegene` *mirtarbasegene*

Description

the dataset that includes miRNA:target gene interactions downloaded from mirtarbase

Format

Classes tbl_df, tbl and data.frame with 380627 observation of 2 variables:

miRNA miRNA symbol

Target target gene symbol

Source

http://mirtarbase.mbc.nctu.edu.tw/php/index.php

`new_counts` *new_counts*

Description

includes new expression values for minimal sample dataset

Format

A data frame with 7 variables and 7 observation of them:

Competing symbol of gene

miRNA symol of miRNA

Competing_count Expression values of competing gene

miRNA_count Expression value of miRNA

Source

Dataset was created by us.

normalize *normalize*

Description

normalizes the values according to maximum values inside a group. The helper function of priming_graph.

Usage

```
normalize(x)
```

Arguments

x	The variable name that is normalized.

Value

normalized values

prepare_rhs *Carries the variables from edge to node*

Description

Carries the variables from edge to node.

Usage

```
prepare_rhs(input_graph)
```

Arguments

input_graph	Processed graph object in previous step.

Details

The function is a helper function for processing of graph object with update_nodes function.

Value

tibble object

`prepare_rhs_once` *Carries the variables from edge to node.*

Description

Carries the variables from edge to node.

Usage

```
prepare_rhs_once(input_graph)
```

Arguments

`input_graph`	Processed graph object in previous step.

Details

The function is a helper function for processing of graph object with update_nodes function.

Value

tibble object

`priming_graph` *Converts the given dataframe using first variable as competing and the second as miRNA. The function converts the given dataframe using first variable as competing and the second as miRNA. If user defines interaction factors as affinity or degradation, the factors are taken into account.*

Description

Converts the given dataframe using first variable as competing and the second as miRNA. The function converts the given dataframe using first variable as competing and the second as miRNA. If user defines interaction factors as affinity or degradation, the factors are taken into account.

Usage

```
priming_graph(
  df,
  competing_count,
  miRNA_count,
  aff_factor = dummy,
  deg_factor = dummy
)
```

Arguments

df	A data frame that includes the miRNA and competing targets.
competing_count	The counts (or expression) of competing elements of the dataset.
miRNA_count	The counts (or expression) of repressive element (miRNA) of the dataset.
aff_factor	The parameter/s of binding between miRNA and targets.
deg_factor	The parameter/s for degradation of bound miRNA:target complex.

Details

priming_graph provides grouping of competing targets and evaluation of targets within the groups taking into account miRNA:target, target:total target, interaction and degradation parameters. The target groups are determined according to miRNAs. If the factors that are important in target interactions are specified as arguments, the factors also are evaluated separately within each group. priming_graph also calculates the miRNA efficiency in steady-state conditions. It is assumed that quantity of competing targets and miRNAs are shown in the steady-state system after the miRNAs exhibit repressive efficiency. Note that the data must not include missing values such as NA or '-'.

Value

the graph object.

Examples

```
data('minsamp')

priming_graph(minsamp, Competing_expression, miRNA_expression)

priming_graph(minsamp, Competing_expression, miRNA_expression,
    aff_factor = c(seed_type,energy), deg_factor = region)
```

---

simulate *Utilizes the change in expression value/s as triggering.*

Description

simulate function uses the change in expression value/s as triggering.

Usage

```
simulate(input_graph, cycle = 1, threshold = 0, knockdown = TRUE)
```

Arguments

input_graph	The graph object that processed in previous steps.
cycle	Optimal iteration number for gaining steady-state.
threshold	absolute minimum amount of change required to be considered as up/down regulated element
knockdown	specifies gene knockdown with default TRUE

Details

The steady-state conditions of the system are disturbed after the change in the graph (with update_how or update_variables). In this case, the system tend to be steady state again. The arrangement of competetive profiles of the targets continue until all nodes are updated and steady-state nearly. Note that, If 'how' argument is specified as '0', *simulate()* and *update_how()* functions process the variables to knockdown of specified gene with default 'knockdown = TRUE' and knocked down competing RNA is kept at zero. However, if 'knockdown= FALSE' argument is applied, competing RNA which has initial expression level of zero is allowed to increase or fluctuate during calculations.

Value

The graph.

Examples

```
data('minsamp')
data('new_counts')

## new_counts, the dataset that includes the current counts of nodes.

priming_graph(minsamp, Competing_expression, miRNA_expression)%>%
  update_variables(new_counts)%>%
  simulate()

priming_graph(minsamp, Competing_expression, miRNA_expression,
       aff_factor = c(seed_type,energy), deg_factor = region)%>%
  update_variables(new_counts)%>%
  simulate(cycle = 3)
```

---

simulate_vis *Provides visualisation of the graph in addition to simulate function.*

Description

simulate_vis provides visualisation of the graph in addition to simulate function.

Usage

```
simulate_vis(
  input_graph,
  cycle = 1,
  threshold = 0,
  save = FALSE,
  Competing_color = "green",
  mirna_color = "orange",
  Upregulation = "red",
  Downregulation = "blue",
  title = "GRAPH",
  layout = "kk"
)
```

Arguments

input_graph	The graph object that processed in previous steps.
cycle	Optimal iteration number for gaining steady-state.
threshold	absolute minimum amount of change required to be considered as up/down regulated element
save	provides to save graph output
Competing_color	The color of competing elements on the graph with "green" default.
mirna_color	The color of miRNAs on the graph with "orange" default.
Upregulation	The color of Upregulated elements on the graph with "red" default.
Downregulation	The color of Downregulated elements on the graph with "blue" default.
title	Title of the given graph.
layout	The layout that will be used for visualisation of the graph.

Details

simulate_vis gives the last graph object and each iterations' image.

Value

It gives a graph and the images of states in each iteration until the end of the simulation.

Examples

```
# When does the system gain steady-state conditions again?

## new_counts, the dataset that includes the current counts of nodes.

data("minsamp")
data("new_counts")

priming_graph(minsamp, Competing_expression, miRNA_expression)%>%
```

```
  update_variables(new_counts)%>%
  simulate_vis()

priming_graph(minsamp, Competing_expression, miRNA_expression,
    aff_factor = c(seed_type,energy), deg_factor = c(region))%>%
  update_variables(new_counts)%>%
  simulate_vis(cycle = 12)
```

`TCGA_E9_A1N5_mirnanormal`

TCGA_E9_A1N5_mirnanormal

Description

The dataset contains mirna expression values for normal tissue sample of TCGA-E9-A1N5 barcoded patient

Format

Classes tbl_df, tbl and data.frame with 750 observation of 6 variables:

barcode Sample, normal tissue, barcode of patient based on TCGA

mirbase_ID mirbase id of miRNA

miRNA miRNA name

Precusor Precusor id of miRNA which is given in miRNA variable

total_read total reading count of miRNA which is produced from different gene locations

total_RPM total RPM (reading per million) of miRNA

Source

https://portal.gdc.cancer.gov/

`TCGA_E9_A1N5_mirnatumor`

TCGA_E9_A1N5_mirnatumor

Description

The dataset contains mirna expression values for tumor tissue sample of TCGA-E9-A1N5 barcoded patient

Format

Classes tbl_df, tbl and data.frame with 648 observation of 6 variables:

barcode Sample, tumor tissue, barcode of patient based on TCGA

mirbase_ID mirbase id of miRNA

miRNA miRNA name

Precusor Precusor id of miRNA which is given in miRNA variable

total_read total reading count of miRNA which is produced from different gene locations

total_RPM total RPM (reading per million) of miRNA

Source

https://portal.gdc.cancer.gov/

TCGA_E9_A1N5_normal *TCGA_E9_A1N5_normal*

Description

The dataset contains gene expression values for normal tissue sample of TCGA-E9-A1N5 barcoded patient

Format

Classes tbl_df, tbl and data.frame with 56830 observation of 7 variables:

patient Barcode of patient based on TCGA

sample Tissue sample barcode of the patient

barcode Sample barcode of the patient

definition Tissue type of sample (Solid Tissue Normal)

ensembl_gene_id Gene id

external_gene_name Gene symbol

gene_expression Gene expression value

Source

https://portal.gdc.cancer.gov/

TCGA_E9_A1N5_tumor *TCGA_E9_A1N5_tumor*

Description

The dataset contains gene expression values for cancer tissue sample of TCGA-E9-A1N5 barcoded patient

Format

Classes tbl_df, tbl and data.frame with 56830 observtion of 7 variables:

patient Barcode of patient based on TCGA

sample Tissue sample barcode of the patient

barcode Sample barcode of the patient

definition Tissue type of sample (Primary solid Tumor)

ensembl_gene_id Gene id

external_gene_name Gene symbol

gene_expression Gene expression value

Source

https://portal.gdc.cancer.gov/

update_how *Converts the count value of the given node.*

Description

this function converts the count value of the given node.

Usage

```
update_how(input_graph, node_name, how, knockdown = TRUE)
```

Arguments

input_graph	The graph object that processed in previous step/s.
node_name	The name of the node whose count is to be changed.
how	The change in terms of fold change.
knockdown	specifies gene knockdown with default TRUE

Details

update_how function calculates the current value of given mirna or gene node on the graph object. User must specify current value as fold change.

Value

the graph object.

Examples

```
data('minsamp')

priming_graph(minsamp, Competing_expression, miRNA_expression)%>%
   update_how('Gene1',3)

 priming_graph(minsamp, Competing_expression, miRNA_expression,
      aff_factor = c(seed_type,energy), deg_factor = region)%>%
   update_how('Gene1', 3)

   priming_graph(minsamp, Competing_expression, miRNA_expression,
       aff_factor = c(seed_type,energy), deg_factor = region)%>%
  update_how('Gene1', how=0, knockdown= TRUE)
```

---

update_nodes *Carries variables from edge to node.*

Description

This function carries variables from edge to node and should be used after ‘update_how‘ or ‘update_variables‘ functions

Usage

```
update_nodes(input_graph, once = FALSE, limit = 0)
```

Arguments

input_graph	Processed graph object in previous step.
once	The argument is about when the carrying process runs (internal use only)
limit	absolute minimum amount of change required to be considered as up/down regulated element

Details

If the carrying process performs after priming_graph function, the argument must be TRUE. The function helps to visualisation of processed graph object, especially that includes too many nodes.This step makes it easily to follow the processes.

Value

the graph object.

Examples

```
data('minsamp')

minsamp %>%
  priming_graph(Competing_expression, miRNA_expression) %>%
  update_how('Gene2',2)
```

`update_variables` *Replaces new values with previous values of competing or miRNA counts.*

Description

This function replaces new values with previous values of competing or miRNA counts.

Usage

```
update_variables(input_graph, current_counts)
```

Arguments

`input_graph`	The processed graph object.
`current_counts`	The additional df that provided by user.

Details

update_variables function provides updating edge variables to current values. If the microRNA or competing expression (or both) change (decreasing or increasing), this function switches the values that are found in a new dataset provided by user. But the current value dataset must be equal with initial dataset in terms of node name.

Value

the graph object.

Examples

```
data('minsamp')
data('new_counts')

 minsamp%>%
 priming_graph(Competing_expression, miRNA_expression,
     aff_factor = c(seed_type,energy), deg_factor = region)%>%
   update_variables(new_counts)
   #new_counts includes the current counts of nodes.
```

---

vis_graph *Provides visualisation of the graph.*

Description

'vis_graph' Provides visualisation of the graph.

Usage

```
vis_graph(
  input_graph,
  Competing_color = "green",
  mirna_color = "orange",
  Upregulation = "red",
  Downregulation = "blue",
  title = "GRAPH",
  layout = "kk"
)
```

Arguments

input_graph	The graph object.
Competing_color	The color of competing elements on the graph with 'green' default.
mirna_color	The color of miRNAs on the graph with 'orange' default.
Upregulation	The color of Upregulated elements on the graph with 'red' default.
Downregulation	The color of Downregulated elements on the graph with 'blue' default.
title	Title of the given graph.
layout	The layout that will be used for visualisation of the graph.

Details

vis_graph ensures the process to be followed.

Value

The graph object.

Examples

```
data('minsamp')
data('new_counts')

# Visualisation of graph in steady-state.

 priming_graph(minsamp, Competing_expression, miRNA_expression,
    aff_factor = c(seed_type,energy), deg_factor = region)%>%
   vis_graph()

# Visualisation of graph after the change.

 priming_graph(minsamp, Competing_expression, miRNA_expression,
    aff_factor = c(seed_type,energy), deg_factor = region)%>%
   update_variables(new_counts)%>%
   vis_graph()
```

Index

∗ **internal**
gene_knockdown, 8
normalize, 12
prepare_rhs, 12
prepare_rhs_once, 13

calc_perturbation, 3

find_affected_nodes, 4
find_iteration, 5
find_node_perturbation, 6
find_targeting_nodes, 7

gene_knockdown, 8

huge_example, 9

midsamp, 9
midsamp_new_counts, 10
minsamp, 10
mirtarbasegene, 11

new_counts, 11
normalize, 12

prepare_rhs, 12
prepare_rhs_once, 13
priming_graph, 13

simulate, 14
simulate_vis, 15

TCGA_E9_A1N5_mirnanormal, 17
TCGA_E9_A1N5_mirnatumor, 17
TCGA_E9_A1N5_normal, 18
TCGA_E9_A1N5_tumor, 19

update_how, 19
update_nodes, 20
update_variables, 21

vis_graph, 22