# Package 'pageRank'

November 6, 2026

**Title** Temporal and Multiplex PageRank for Gene Regulatory Network Analysis

**Version** 1.23.0

**Description** Implemented temporal PageRank analysis as defined by Rozenshtein and Gionis. Implemented multiplex PageRank as defined by Halu et al. Applied temporal and multiplex PageRank in gene regulatory network analysis.

**Depends** R (>= 4.0)

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

**Imports** GenomicRanges, igraph, motifmatchr, stats, utils, grDevices, graphics

**Suggests** bcellViper, BSgenome.Hsapiens.UCSC.hg19, JASPAR2018, TxDb.Hsapiens.UCSC.hg19.knownGene, org.Hs.eg.db, TFBSTools, GenomicFeatures, annotate

**biocViews** StatisticalMethod, GeneTarget, Network

**URL** https://github.com/hd2326/pageRank

**BugReports** https://github.com/hd2326/pageRank/issues

**RoxygenNote** 6.1.99.9001

**git_url** https://git.bioconductor.org/packages/pageRank

**git_branch** devel

**git_last_commit** 400b824

**git_last_commit_date** 2026-04-28

**Repository** Bioconductor 3.24

**Date/Publication** 2026-11-05

**Author** Hongxu Ding [aut, cre, ctb, cph]

**Maintainer** Hongxu Ding <hd2326@columbia.edu>

# Contents

accessibility_network . . . 2
adjust_graph . . . 3
aracne_network . . . 4
bubble_plot . . . 5
clean_graph . . . 6
conformation_network . . . 6
diff_graph . . . 8
gene_bin . . . 9
get_color_gradient . . . 9
multiplex_page_rank . . . 10
PX . . . 11
PXPY . . . 11
PXY . . . 12
P_dist . . . 13
P_graph . . . 13
P_null . . . 14
time_expmat . . . 15

**Index** **16**

---

`accessibility_network` *Build Network from Accessibility Peaks.*

---

## Description

Build network from accessibility, e.g. ATAC-Seq peaks.

## Usage

```
accessibility_network(table, promoter, pfm, genome, p.cutoff = 5e-05, w = 7)
```

## Arguments

| | |
|---|---|
| `table` | (data.frame) Peaks, with "Chr", "Start" and "End" in column name, and peak ID in row names. |
| `promoter` | (GRanges) Promoter regions. |
| `pfm` | (PFMatrixList) Positon Frequency Matrices (PFMs) of regulators. |
| `genome` | (BSgenome or character) Genome build in which regulator motifs will be searched. |
| `p.cutoff` | (numeric) P-value cutoff for motifs searching within peaks for TF identificaton. |
| `w` | (numeric) Window size for motifs searching within peaks for TF identificaton. |

## Value

(data.frame) Network, with "reg" and "target" in column name.

## Author(s)

DING, HONGXU (hd2326@columbia.edu)

## Examples

```
table <- data.frame(Chr=c("chr1", "chr1"), Start=c(713689, 856337),
                    End=c(714685, 862152), row.names=c("A", "B"),
                    stringsAsFactors=FALSE)
regulators=c("FOXF2", "MZF1")
#peaks and regulators to be analyzed

library(GenomicRanges)
library(GenomicFeatures)
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(org.Hs.eg.db)
library(annotate)
promoter <- promoters(genes(TxDb.Hsapiens.UCSC.hg19.knownGene))
names(promoter) <- getSYMBOL(names(promoter), data="org.Hs.eg")
promoter <- promoter[!is.na(names(promoter))]
#get promoter regions

library(JASPAR2018)
library(TFBSTools)
library(motifmatchr)
pfm <- getMatrixSet(JASPAR2018, list(species="Homo sapiens"))
pfm <- pfm[unlist(lapply(pfm, function(x) name(x))) %in% regulators]
#get regulator position frequency matrix (PFM) list

library(BSgenome.Hsapiens.UCSC.hg19)
accessibility_network(table, promoter, pfm, "BSgenome.Hsapiens.UCSC.hg19")
#generate network
```

---

`adjust_graph` *Re-calculate PageRank*

---

## Description

Re-calculate PageRank with updated damping factor, personalized vector and edge weights.

## Usage

```
adjust_graph(graph, damping = 0.85, personalized = NULL, weights = NULL)
```

## Arguments

| | |
|---|---|
| `graph` | (igraph) The graph to be adjusted. |
| `damping` | (numeric) Damping factor. |
| `personalized` | (numeric) Personalized vector. |
| `weights` | (numeric) Weight vector. |

## Value

(igraph) Network with updated "pagerank" as vertex attribute.

## Author(s)

DING, HONGXU (hd2326@columbia.edu)

## Examples

```
library(igraph)
set.seed(1)
graph <- igraph::erdos.renyi.game(100, 0.01, directed = TRUE)
igraph::V(graph)$name <- 1:100
igraph::V(graph)$pagerank <- igraph::page_rank(graph, damping=0.85)$vector
adjust_graph(graph, damping=0.1)
```

---

# aracne_network *Re-format ARACNe Network.*

---

## Description

Re-format ARACNe network in regulon object to data.frame with regulator, target and direction columns.

## Usage

```
aracne_network(regulon)
```

## Arguments

| | |
|---|---|
| regulon | (regulon) ARACNe network. |

## Value

(data.frame) Network, with "reg", "target" and "direction" in column name. For direction, 1/0 denotes positive/negative regulation.

## Author(s)

DING, HONGXU (hd2326@columbia.edu)

## Examples

```
library(bcellViper)
data(bcellViper)
aracne_network(regulon[1:10])
```

`bubble_plot` *Make Bubbleplot*

## Description

Make bubbleplot.

## Usage

```
bubble_plot(
  s_mat,
  c_mat,
  n_mat,
  col = colorRampPalette(c("Blue", "Grey", "Red"))(100),
  breaks = seq(-2, 2, length.out = 100),
  main = NULL
)
```

## Arguments

| | |
|---|---|
| `s_mat` | (matrix) Matrix denotes the size of bubbles. |
| `c_mat` | (matrix) Matrix denotes the color of bubbles. |
| `n_mat` | (matrix) Matrix denotes the name of bubbles. |
| `col` | (character) Colors. |
| `breaks` | (numeric) Breakpoints of colors. |
| `main` | (character) Title. |

## Value

(NULL) A bubbloplot.

## Author(s)

DING, HONGXU (hd2326@columbia.edu)

## Examples

```
s_mat <- c_mat <- n_mat <- matrix(1:12, 3, 4, dimnames=list(1:3, 1:4))
bubble_plot(s_mat, c_mat, n_mat, breaks=seq(1, 12, length.out=100), main="")
```

## clean_graph *Clean Graph*

### Description

Remove graph nodes by residing subgraph sizes, vertex names and PageRank values.

### Usage

```
clean_graph(graph, size = NULL, vertices = NULL, pagerank = NULL)
```

### Arguments

| | |
|---|---|
| graph | (igraph) The graph to be cleaned. |
| size | (numeric) Subgraph size cutoff. |
| vertices | (character) Vertices to be kept. |
| pagerank | (numeric) PageRank cutoff. |

### Value

(igraph) Network updated "pagerank" as vertex attribute.

### Author(s)

DING, HONGXU (hd2326@columbia.edu)

### Examples

```
library(igraph)
set.seed(1)
graph <- igraph::erdos.renyi.game(100, 0.01, directed = TRUE)
igraph::V(graph)$name <- 1:100
igraph::V(graph)$pagerank <- igraph::page_rank(graph)$vector
clean_graph(graph, size=5)
```

## conformation_network *Build Network from Conformation Peaks.*

### Description

Build network from conformation, e.g. HiChIP records.

## Usage

```
conformation_network(
  table,
  promoter,
  pfm,
  genome,
  range = 500,
  p.cutoff = 5e-05,
  w = 7
)
```

## Arguments

| | |
|---|---|
| table | (data.frame) Records, with "Chr1", "Position1", "Strand1", "Chr2", "Position2" and "Strand2" in column name, and record ID in row names. |
| promoter | (GRanges) Promoter regions. |
| pfm | (PFMatrixList) Positon Frequency Matrices (PFMs) of regulators. |
| genome | (BSgenome or character) Genome build in which regulator motifs will be searched. |
| range | (numeric) Search radius from "Position1" and "Position2" for promoters. |
| p.cutoff | (numeric) P-value cutoff for motifs searching within peaks for TF identificaton. |
| w | (numeric) Window size for motifs searching within peaks for TF identificaton. |

## Value

(data.frame) Network, with "reg" and "target" in column name.

## Author(s)

DING, HONGXU (hd2326@columbia.edu)

## Examples

```
table <- data.frame(Chr1=c("chr1", "chr1"),
                    Position1=c(569265, 713603),
                    Strand1=c("+", "+"),
                    Chr2=c("chr4", "chr1"),
                    Position2=c(206628, 715110),
                    Strand2=c("+", "-"),
                    row.names=c("A", "B"), stringsAsFactors=FALSE)
regulators=c("FOXF2", "MZF1")
#peaks and regulators to be analyzed

library(GenomicRanges)
library(GenomicFeatures)
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(org.Hs.eg.db)
library(annotate)
promoter <- promoters(genes(TxDb.Hsapiens.UCSC.hg19.knownGene))
names(promoter) <- getSYMBOL(names(promoter), data="org.Hs.eg")
promoter <- promoter[!is.na(names(promoter))]
#get promoter regions

library(JASPAR2018)
```

```
library(TFBSTools)
library(motifmatchr)
pfm <- getMatrixSet(JASPAR2018, list(species="Homo sapiens"))
pfm <- pfm[unlist(lapply(pfm, function(x) name(x))) %in% regulators]
#get regulator position frequency matrix (PFM) list

library(BSgenome.Hsapiens.UCSC.hg19)
conformation_network(table, promoter, pfm, "BSgenome.Hsapiens.UCSC.hg19")
```

---

`diff_graph` *Calculate Temporal PageRank from Two Graphs*

---

## Description

Calculate temporal PageRank by changing edges between graph1 and graph2. This is a simplified version of temporal PageRank described by Rozenshtein and Gionis, by only analyzing temporally adjacent graph pairs.

## Usage

```
diff_graph(graph1, graph2)
```

## Arguments

| | |
|---|---|
| `graph1` | (igraph) The 1st graph. |
| `graph2` | (igraph) The 2nd graph. |

## Value

(igraph) Network graph1-graph2 with "moi (mode of interaction)" and "pagerank" as edge and vertex attributes.

## Author(s)

DING, HONGXU (hd2326@columbia.edu)

## References

Rozenshtein, Polina, and Aristides Gionis. "Temporal pagerank." Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Cham, 2016.

## Examples

```
library(igraph)
set.seed(1)
graph1 <- igraph::erdos.renyi.game(100, 0.01, directed = TRUE)
igraph::V(graph1)$name <- 1:100
set.seed(2)
graph2 <- igraph::erdos.renyi.game(100, 0.01, directed = TRUE)
igraph::V(graph2)$name <- 1:100
diff_graph(graph1, graph2)
```

gene_bin *Bin Gene Expression Space*

## Description

Bin gene expression space for marginal/joint probability calculation.

## Usage

```
gene_bin(genes, expmat, sep = 5)
```

## Arguments

| | |
|---|---|
| genes | (character) Genes to be analyzed. |
| expmat | (matrix) Gene expression matrix. |
| sep | (numeric) Number of bins. |

## Value

(matrix) Border values of gene expression bins.

## Author(s)

DING, HONGXU (hd2326@columbia.edu)

get_color_gradient *Generate Color Gradient*

## Description

Generate color gradient for, e.g. gene expression.

## Usage

```
get_color_gradient(
  x,
  col = colorRampPalette(c("Blue", "Red"))(100),
  breaks = seq(-2, 2, length.out = 100)
)
```

## Arguments

| | |
|---|---|
| x | (numeric) Vector based on which color gradient is generated. |
| col | (character) Color vector. |
| breaks | (numeric) A set of breakpoints for the colors. Must be the same length of col. |

## Value

(character) Colors.

### Author(s)

DING, HONGXU (hd2326@columbia.edu)

### Examples

```
get_color_gradient(-2:2)
```

---

`multiplex_page_rank` *Calculate Multiplex PageRank*

---

### Description

Calculate multiplex PageRank following defination by Halu et al.

### Usage

```
multiplex_page_rank(graph, ..., beta = 1, gamma = 1, damping = 0.85)
```

### Arguments

| | |
|---|---|
| `graph` | (igraph) The base graph with pagerank and name as vertex attributes. |
| `...` | (igraph) Supporter graphs with pagerank and name as vertex attributes. |
| `beta` | (numeric) Parameters for adjusting supporter graph PageRank values. For the same nodes, PageRank values from different supporter graphs will first be multiplicated. The products will then be exponentiate by beta and gamma, as outgoing edge weights and personalizations of the base graph. Four special multiplex PageRank forms are defined by varying (beta, gamma), including additive (0, 1), multiplicative (1, 0), combined (1, 1) and neutral (0, 0). |
| `gamma` | (numeric) Parameters for adjusting supporter graph PageRank values. For the same nodes, PageRank values from different supporter graphs will first be multiplicated. The products will then be exponentiate by beta and gamma, as outgoing edge weights and personalizations of the base graph. Four special multiplex PageRank forms are defined by varying (beta, gamma), including additive (0, 1), multiplicative (1, 0), combined (1, 1) and neutral (0, 0). |
| `damping` | (numeric) Damping factor. |

### Value

(numeric) Multiplex PageRank values.

### Author(s)

DING, HONGXU (hd2326@columbia.edu)

### References

Halu, Arda, et al. "Multiplex pagerank." PloS one 8.10 (2013).

## Examples

```
library(igraph)
set.seed(1)
graph1 <- igraph::erdos.renyi.game(100, 0.01, directed = TRUE)
igraph::V(graph1)$name <- 1:100
igraph::V(graph1)$pagerank <- igraph::page_rank(graph1)$vector
set.seed(2)
graph2 <- igraph::erdos.renyi.game(100, 0.01, directed = TRUE)
igraph::V(graph2)$name <- 1:100
igraph::V(graph2)$pagerank <- igraph::page_rank(graph2)$vector
multiplex_page_rank(graph1, graph2)
```

---

# PX *Marginal Probability Calculation*

---

## Description

Calculate marginal probability.

## Usage

```
PX(expmat, bin)
```

## Arguments

| | |
|---|---|
| expmat | (matrix) Gene expression matrix. |
| bin | (matrix) Results of gene_bin function. |

## Value

(matrix) Marginal probability matrix.

## Author(s)

DING, HONGXU (hd2326@columbia.edu)

---

# PXPY *Product of Marginal Probability*

---

## Description

Calculate the product of marginal probability.

## Usage

```
PXPY(px, combinations)
```

### Arguments

| | |
|---|---|
| px | (matrix) Marginal probability generated by PX fuction. |
| combinations | (character) Variable combinations ("x_y") to be analyzed. |

### Value

(list) Product of marginal probability.

### Author(s)

DING, HONGXU (hd2326@columbia.edu)

---

## PXY *Joint Probability Calculation*

---

### Description

Calculate joint probability.

### Usage

```
PXY(expmat, bin, x, y)
```

### Arguments

| | |
|---|---|
| expmat | (matrix) Gene expression matrix. |
| bin | (matrix) Results of gene_bin function. |
| x | (character) The first variable. |
| y | (character) The pairing second variable. |

### Value

(list) Joint probability.

### Author(s)

DING, HONGXU (hd2326@columbia.edu)

## P_dist *Calculate Probability-Based Distance*

### Description

Calculate distance based on PXY and PXPY.

### Usage

```
P_dist(pxy, pxpy, method = c("difference", "mi"))
```

### Arguments

| | |
|---|---|
| pxy | (list) Joint probability generated by PXY fuction. |
| pxpy | (list) Product of marginal probability generated by PXPY fuction. |
| method | (character) Method for calculating distance, either PXY-PXPY ("difference") or mutual information ("mi"). |

### Value

(list) Distance values.

### Author(s)

DING, HONGXU (hd2326@columbia.edu)

## P_graph *Build Probability-Based Network*

### Description

Build probability-based regulator-target interaction network.

### Usage

```
P_graph(
  expmat,
  net,
  sep = 5,
  method = c("difference", "mi"),
  null = NULL,
  threshold = 0.001
)
```

### Arguments

| | |
|---|---|
| expmat | (matrix) Gene expression matrix. |
| net | (data.frame) Network, with "reg" and "target" in column name. |
| sep | (numeric) Number of bins for calculating marginal/joint probability. |
| method | (character) Method for calculating probability-based distance, either PXY-PXPY ("difference") or mutual information ("mi"). |
| null | (ecdf) Null distribution of probability-based distance. Either from random interactions by P_null function, or all interactions in net. |
| threshold | (numeric) P-value threshold for filtering interactions in net. |

### Value

(igraph) Network graph with "pvalue" and "direction", and "pagerank" as edge/vertex attributes.

### Author(s)

DING, HONGXU (hd2326@columbia.edu)

### Examples

```
library(bcellViper)
data(bcellViper)
dset <- exprs(dset)
net <- do.call(rbind, lapply(1:10, function(i, regulon){
  data.frame(reg=rep(names(regulon)[i], 10),
             target=names(regulon[[i]][[1]])[1:10],
             direction=rep(1, 10),
             stringsAsFactors = FALSE)}, regulon=regulon))
P_graph(dset, net, method="difference", null=NULL, threshold=0.05)
```

---

## P_null *Build Null Distribution of Probability-Based Distance*

---

### Description

Build null model for evaluating the significance of interactions by generating random regulator-target pairs.

### Usage

```
P_null(expmat, net, n = 10000, sep = 5, method = c("difference", "mi"))
```

### Arguments

| | |
|---|---|
| expmat | (matrix) Gene expression matrix. |
| net | (data.frame) Network, with "reg" and "target" in column name. |
| n | (numeric) Number of random pairs. |
| sep | (numeric) Number of bins for calculating marginal/joint probability. |
| method | (character) Method for calculating probability-based distance, either PXY-PXPY ("difference") or mutual information ("mi"). |

### Value

(ecdf) ECDF of null distribution.

### Author(s)

DING, HONGXU (hd2326@columbia.edu)

### Examples

```
library(bcellViper)
data(bcellViper)
dset <- exprs(dset)
net <- do.call(rbind, lapply(1:10, function(i, regulon){
  data.frame(reg=rep(names(regulon)[i], 10),
             target=names(regulon[[i]][[1]])[1:10],
             direction=rep(1, 10),
             stringsAsFactors = FALSE)}, regulon=regulon))
P_null(dset, net, n=100, method="difference")
```

---

## time_expmat *Generate Timewise Average Gene Expression*

---

### Description

Generate timewise average gene expression.

### Usage

```
time_expmat(time, expmat)
```

### Arguments

| | |
|---|---|
| time | (character) Time-annotation of samples. |
| expmat | (matrix) Gene expression matrix. |

### Value

(matrix) Time-wise average gene expression.

### Author(s)

DING, HONGXU (hd2326@columbia.edu)

### Examples

```
expmat <- matrix(rnorm(90), 10, 9, dimnames=list(LETTERS[1:10], 1:9))
time <- c(rep("T1", 3), rep("T2", 3), rep("T3", 3))
time_expmat(time, expmat)
```

# Index

∗ **internal**
gene_bin, 9
P_dist, 13
PX, 11
PXPY, 11
PXY, 12

accessibility_network, 2
adjust_graph, 3
aracne_network, 4

bubble_plot, 5

clean_graph, 6
conformation_network, 6

diff_graph, 8

gene_bin, 9
get_color_gradient, 9

multiplex_page_rank, 10

P_dist, 13
P_graph, 13
P_null, 14
PX, 11
PXPY, 11
PXY, 12

time_expmat, 15