

# Package ‘epiregulon.extra’

January 22, 2026

**Title** Companion package to epiregulon with additional plotting, differential and graph functions

**Version** 1.6.0

**Description** Gene regulatory networks model the underlying gene regulation hierarchies that drive gene expression and observed phenotypes. Epiregulon infers TF activity in single cells by constructing a gene regulatory network (regulons). This is achieved through integration of scATAC-seq and scRNA-seq data and incorporation of public bulk TF ChIP-seq data. Links between regulatory elements and their target genes are established by computing correlations between chromatin accessibility and gene expressions.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**Imports** scran, ComplexHeatmap, Matrix, SummarizedExperiment, checkmate, circlize, clusterProfiler, ggplot2, ggraph, igraph, patchwork, reshape2, scales, scater

**Depends** R (>= 4.4), SingleCellExperiment

**Suggests** epiregulon, knitr, rmarkdown, parallel, BiocStyle, testthat (>= 3.0.0), msigdb, GSEABase, dorothea, scMultiome, S4Vectors, scuttle, vdiff, ggrastr, ggrepel

**VignetteBuilder** knitr

**URL** <https://github.com/xiaosaiyao/epiregulon.extra/>

**biocViews** GeneRegulation, Network, GeneExpression, Transcription, ChipOnChip, DifferentialExpression, GeneTarget, Normalization, GraphAndNetwork

**Config/testthat/edition** 3

**BugReports** <https://github.com/xiaosaiyao/epiregulon.extra/issues>

**LazyData** true

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

**git\_branch** RELEASE\_3\_22

**git\_last\_commit** d60eef8

**git\_last\_commit\_date** 2025-10-29

**Repository** Bioconductor 3.22

**Date/Publication** 2026-01-22

**Author** Xiaosai Yao [aut, cre] (ORCID: <<https://orcid.org/0000-0001-9729-0726>>),  
 Tomasz Włodarczyk [aut] (ORCID:  
 <<https://orcid.org/0000-0003-1554-9699>>),  
 Timothy Keyes [aut],  
 Shang-Yang Chen [aut]

**Maintainer** Xiaosai Yao <xiaosai.yao@gmail.com>

## Contents

buildGraph . . . . .	2
calculateJaccardSimilarity . . . . .	5
enrichPlot . . . . .	6
findDifferentialActivity . . . . .	7
findPartners . . . . .	8
getSigGenes . . . . .	9
permuteGraph . . . . .	10
plotActivityDim . . . . .	10
plotActivityViolin . . . . .	12
plotBubble . . . . .	13
plotDiffNetwork . . . . .	14
plotEpiregulonNetwork . . . . .	15
plotGseaNetwork . . . . .	17
plotHeatmapActivity . . . . .	18
plotHeatmapRegulon . . . . .	19
regulon . . . . .	21
regulonEnrich . . . . .	22
<b>Index</b>	<b>23</b>

---

buildGraph	<i>Creating graphs and related operations</i>
------------	---

---

## Description

The function enable to create graph objects using as input regulon objects returned by pruneRegulon or addWeights. Both weighted and unweighted graphs can be created that can further be visualized using dedicated functions.

## Usage

```
buildGraph(
  regulon,
  mode = c("tg", "tripartite", "re", "pairs"),
  weights = "weights",
  cluster = "all",
  aggregation_function = function(x) x[which.max(abs(x))],
  na_replace = TRUE,
  keep_original_names = TRUE,
  filter_edges = NULL
```

```

)
buildDiffGraph(graph_obj_1, graph_obj_2, weighted = TRUE, abs_diff = TRUE)
addCentrality(graph)
normalizeCentrality(graph, FUN = sqrt, weighted = TRUE)
rankTfs(graph, type_attr = "type")

```

### Arguments

regulon	an object returned by the getRegulon or addWeights function.
mode	a character specifying which type of graph will be built. In 'tg' mode a bipartite graph is built by connecting transcription factors directly to the target genes and ignoring information about mediating regulatory elements; in 'pairs' mode transcription factors are connected to unique target gene-regulatory element pairs; in 'tripartite' mode the network is made up of three types of vertices (nodes): transcription factors, regulatory elements and target genes; here the path from target gene to regulatory element always contains a regulatory element; in 're' mode data in the target genes is dropped and only connections are between transcription factors and regulatory elements.
weights	a character specifying which variable should be used to assign weights to edges.
cluster	a character specifying the name of the cluster column which should be used to retrieve weight values from regulon object. Using this argument makes sense only with combination with weights parameter when it points to the regulon column that is a matrix.
aggregation_function	a function used to aggregate weights of duplicated edges, which might appear due to the many transcription factor converging at the same regulatory element; starting from this point each transcription factor is supposed to have a separate connection to the target gene, perhaps the same one across several connections. In tripartite mode this might result in many edges in the same node pair, however weights might differ since they are inherited from different tf-re-tg triplets (rows) in the regulon object. Similarly, duplicated edges are generated by one transcription factor using a regulatory element multiple times to reach different target genes. In tg mode the edges became duplicated if one transcription factor reaches the same target genes through many regulatory elements.
na_replace	a logical indicating whether NA values for weights should be replaced with zeros.
keep_original_names	A logical indicating whether gene names should be used as node names in the output graph. Note that this might lead to the duplicated node names if the same gene is present in two layers (transcription factors and target genes).
filter_edges	A numeric defining the cutoff weight used for filtering out edges which have weights equal or greater than cutoff. The isolated vertices are removed then from the graph. Defaults to NULL in which case no filtering is applied.
weighted	a logical indicating whether weighted graphs are used; in tripartite mode tf-re-tg triplet is decomposed into two edges corresponding to tf-re and re-tg pairs, and both edges inherit the same weight, which was originally assigned to the parent triplet.

abs_diff	a logical indicating whether absolute difference in the number edges or their weights will be calculated.
graph, graph_obj_1, graph_obj_2	an igraph object.
FUN	a function used for normalization. The input to this function is be the number of edges connected with each node (incident edges).
type_attr	a character corresponding to the name of the vertex attribute which indicate the type of vertex.

## Details

buildGraph function creates a directed graph based on the output of the getRegulon function. Four modes are available: (1) tg in which connections are made directly between transcription factor and target genes. Even if the same tf-tg pair is connected in the original regulon object through many regulatory elements then only one edge is created. In such a case, when weighted graph is created, weights are summarized by the aggregating function (by default the maximum absolute value with the sign of the original value). Similarly, aggregation is made in the re mode leaving only unique transcription factor-regulatory element pairs. In tripartite mode edges connect transcription factors with regulatory elements and regulatory elements with target genes. The same weights are used for both edges that correspond to the single row in the regulon data frame (tf-re and re-tg). Note that the original regulon structure is not fully preserved because each row is now represented by two edges which are independent from each other. Thus they can be coupled with different edges connected to the same regulatory element building the path from transcription factor to the target gene of another transcription factor through the shared regulatory element.

buildDiffGraph a graph difference by subtracting the edges of graph\_obj\_2 from those of the graph\_obj\_1. If weighted is set to TRUE then for each ordered pair of vertices (nodes) the difference in number of edges between graph\_obj\_1 and graph\_obj\_1 is calculated. The result is used to set the number of corresponding edges in output graph. Note that unless abs\_diff is set to TRUE any non-positive difference will translate into lack of the edges for a corresponding ordered pair of vertices in the output graph (equivalent to 0 value in the respective position in adjacency matrix). In case of weighted graphs, the weight of the output graph is calculated as a difference of the corresponding weights between input graphs.

addCentrality calculates degree centrality for each vertex using igraph::strength.

With normalizeCentrality function the normalized values of centrality are calculated from the original ones divided by FUN(total number of non-zero edges associated with each node).

rankTfs assign ranks to transcription factors according to degree centrality of their vertices

## Value

an igraph object. rankTfs returns a data.frame with transcription factors sorted according to the value of the centrality attribute.

## Examples

```
# create an artificial getRegulon output
set.seed(1234)
tf_set <- apply(expand.grid(LETTERS[1:10], LETTERS[1:10]),1, paste, collapse = '')
regulon <- DataFrame(tf = sample(tf_set, 5e3, replace = TRUE))
gene_set <- expand.grid(LETTERS[1:10], LETTERS[1:10], LETTERS[1:10])
gene_set <- apply(gene_set,1,function(x) paste0(x,collapse=''))
regulon$target <- sample(gene_set, 5e3, replace = TRUE)
regulon$idxATAC <- 1:5e3
```

```
regulon$corr <- runif(5e3)*0.5+0.5
regulon$weights <- matrix(runif(15000), nrow=5000, ncol=3)
colnames(regulon$weights) <- c('all', 'cluster1', 'cluster2')
graph_tripartite <- buildGraph(regulon, cluster='all', mode = 'tripartite')

# build bipartite graph using regulatory element-target gene pairs
graph_pairs_1 <- buildGraph(regulon, cluster = 'cluster1', mode = 'pairs')
graph_pairs_2 <- buildGraph(regulon, cluster = 'cluster2', mode = 'pairs')
graph_diff <- buildDiffGraph(graph_pairs_1, graph_pairs_2)
graph_diff <- addCentrality(graph_diff)
graph_diff <- normalizeCentrality(graph_diff)
tf_ranking <- rankTfs(graph_diff)
```

---

calculateJaccardSimilarity

*Calculate Jaccard Similarity between regulons of all transcription factors*

---

## Description

Calculate Jaccard Similarity between regulons of all transcription factors

## Usage

```
calculateJaccardSimilarity(graph)
```

## Arguments

graph            a igraph object from buildGraph or buildDiffGraph

## Value

A matrix with Jaccard similarity between all pairs of transcription factors.

## Examples

```
regulon <- data.frame(tf = sample(letters[1:4], 100, replace = TRUE), idxATAC= 1:100,
target = sample(letters[5:14], 100, replace = TRUE))
regulon$weights <- runif(100)
GRN_graph <- buildGraph(regulon)
similarity <- calculateJaccardSimilarity(GRN_graph)
```

---

enrichPlot *Plot results of regulonEnrich*

---

### Description

Plot results of regulonEnrich

### Usage

```
enrichPlot(results, top = 15, ncol = 3, title = NULL, combine = TRUE)
```

### Arguments

results	Output from regulonEnrich
top	An integer to indicate the number of pathways to plot ranked by significance. Default is 15.
ncol	An integer to indicate the number of columns in the combined plot, if combine == TRUE. Default is 3.
title	String indicating the title of the combined plot
combine	logical to indicate whether to combine and visualize the plots in one panel

### Value

A combined ggplot object or a list of ggplots if combine == FALSE

### Author(s)

Xiaosai Yao

### Examples

```
#retrieve genesets
msigdb.hs = msigdb::getMsigdb(org = 'hs', id = 'SYM', version = '7.4')

#convert genesets to be compatible with enricher
msigdb.hs <- msigdb.hs[unlist(lapply(msigdb.hs, function(x) {GSEABase::bcCategory(GSEABase::collectionType(x)
gs.list <- do.call(rbind, lapply(names(msigdb.hs), function(x) {
  data.frame(gs = x, genes = GSEABase::geneIds(msigdb.hs[x][[1]]))}))

head(gs.list)

#get regulon
library(dorothea)
data(dorothea_hs, package = 'dorothea')
regulon <- dorothea_hs
enrichment_results <- regulonEnrich(c('ESR1','AR'), regulon = regulon, weight = 'mor',
genesets = gs.list)

# plot graph
enrichPlot(results = enrichment_results )
```

---

`findDifferentialActivity`*Test for differential TF activity between pairs of single cell clusters/groups*

---

**Description**

Test for differential TF activity between pairs of single cell clusters/groups

**Usage**

```
findDifferentialActivity(  
  activity_matrix,  
  clusters,  
  test.type = c("t", "wilcox", "binom"),  
  pval.type = c("some", "any", "all"),  
  direction = c("any", "up", "down"),  
  logvalues = TRUE,  
  ...  
)
```

**Arguments**

<code>activity_matrix</code>	A matrix of TF activities inferred from <code>calculateActivity</code>
<code>clusters</code>	A character or integer vector of cluster or group labels for single cells
<code>test.type</code>	String indicating the type of statistical tests to be passed to <code>scan::findMarkers</code> , can be "t", "wilcox". or "binom"
<code>pval.type</code>	A string specifying how p-values are to be combined across pairwise comparisons for a given group/cluster. For more details see <a href="#">combineMarkers</a> .
<code>direction</code>	A string specifying direction of differential TF activity, can be "any", "up" or "down"
<code>logvalues</code>	logical indicating whether activities are computed from logged gene expression or not. If activity is computed from linear values of gene expression, setting <code>logvalues</code> to <code>FALSE</code> will return the difference. If activity is computed from logged values of gene expression, setting <code>logvalues</code> to <code>TRUE</code> will return the log changes.
<code>...</code>	Further arguments to pass to <code>scan::findMarkers</code>

**Value**

A named list of dataframes containing differential TF activity test results for each cluster/group

**Author(s)**

Xiaosai Yao, Shang-yang Chen

**Examples**

```

set.seed(1)
score.combine <- cbind(matrix(runif(2000,0,2), 20,100), matrix(runif(2000,0,10), 20,100))
rownames(score.combine) <- paste0("TF",1:20)
colnames(score.combine) <- paste0("cell",1:200)
cluster <- c(rep(1,100),rep(2,100))
markers <- findDifferentialActivity(
  activity_matrix = score.combine,
  clusters = cluster,
  pval.type = "some",
  direction = "up",
  test.type = "t")
sig.genes <- getSigGenes(markers, fdr_cutoff = 1, summary_cutoff = 0.1)

```

---

findPartners

*Find interaction partners of a transcription factor of interest*


---

**Description**

Find interaction partners of a transcription factor of interest

**Usage**

```
findPartners(graph, focal_tf)
```

**Arguments**

graph	a igraph object from buildGraph or buildDiffGraph
focal_tf	character string indicating the name of the transcription factors to find interaction partners of

**Value**

A list with elements corresponding to each transcription factor apart from the focal one. Each list element is represented as a data frame with columns containing names of all target genes shared with focal transcription factor, weights of edges connecting transcription factor with target genes, equivalent weights for focal transcription factor and the element wise product of both weight columns.

**Examples**

```

regulon <- data.frame(tf = sample(letters[1:4], 100, replace = TRUE), idxATAC= 1:100,
  target = sample(letters[5:14], 100, replace = TRUE))
regulon$weights <- runif(100)
GRN_graph <- buildGraph(regulon)
partners <- findPartners(GRN_graph, 'a')

```



---

getSigGenes	<i>Compile and summarize the output from findDifferentialActivity function</i>
-------------	--

---

## Description

Compile and summarize the output from findDifferentialActivity function

## Usage

```
getSigGenes(  
  da_list,  
  fdr_cutoff = 0.05,  
  summary_cutoff = NULL,  
  topgenes = NULL,  
  direction = c("any", "up", "down")  
)
```

## Arguments

da_list	List of dataframes from running findDifferentialActivity
fdr_cutoff	A numeric scalar to specify the cutoff for FDR value. Default is 0.05
summary_cutoff	A numeric scalar to specify the cutoff for log fold change or difference
topgenes	A integer scalar to indicate the number of top ordered genes to include in output
direction	A string specifying direction for which differential TF activity was calculated, can be "any", "up" or "down"

## Value

A compiled dataframe of TFs with differential activities across clusters/groups

## Author(s)

Xiaosai Yao, Shang-yang Chen

## Examples

```
set.seed(1)  
score.combine <- cbind(matrix(runif(2000,0,2), 20,100), matrix(runif(2000,0,10), 20,100))  
rownames(score.combine) <- paste0("TF",1:20)  
colnames(score.combine) <- paste0("cell",1:200)  
cluster <- c(rep(1,100),rep(2,100))  
markers <- findDifferentialActivity(score.combine, cluster, pval.type = "some", direction = "up",  
  test.type = "t")  
sig.genes <- getSigGenes(markers, fdr_cutoff = 1, summary_cutoff = 0.1)  
utils::head(sig.genes)
```

---

permuteGraph	<i>Calculate similarity score from permuted graphs to estimate background similarity</i>
--------------	--

---

### Description

Calculate similarity score from permuted graphs to estimate background similarity

### Usage

```
permuteGraph(graph, focal_tf, n = 100, p = 1)
```

### Arguments

graph	an igraph object from buildGraph or buildDiffGraph
focal_tf	character string indicating the name of the transcription factors to calculate similarity score
n	an integer indicating the number of permutations
p	a scalar indicating the probability of rewiring the graphs

### Value

A matrix with Jaccard similarity between the focal transcription factor and all pairs of transcription factors for n permuted graphs

### Examples

```
regulon <- data.frame(tf = sample(letters[1:4], 100, replace = TRUE), idxATAC= 1:100,
target = sample(letters[5:14], 100, replace = TRUE))
regulon$weights <- runif(100)
GRN_graph <- buildGraph(regulon)
permuted_graph <- permuteGraph(GRN_graph, focal_tf = "a")
```

---

plotActivityDim	<i>Plot cell-level reduced dimension results stored in a SingleCellExperiment object, colored by activities for a list of TFs</i>
-----------------	---

---

### Description

Plot cell-level reduced dimension results stored in a SingleCellExperiment object, colored by activities for a list of TFs

**Usage**

```
plotActivityDim(
  sce = NULL,
  activity_matrix,
  tf,
  dimtype = "UMAP",
  label = NULL,
  ncol = NULL,
  nrow = NULL,
  title = NULL,
  combine = TRUE,
  legend.label = "activity",
  colors = c("blue", "yellow"),
  limit = NULL,
  ...
)
```

**Arguments**

sce	A SingleCellExperiment object containing dimensionality reduction coordinates
activity_matrix	A matrix of TF activities inferred from calculateActivity
tf	A character vector indicating the names of the transcription factors to be plotted
dimtype	String indicating the name of dimensionality reduction matrix to be extracted from the SingleCellExperiment
label	String corresponding to the field in the colData of sce for annotation on plot
ncol	A integer to specify the number of columns in the combined plot, if combine == TRUE
nrow	A integer to specify the number of rows in the combined plot, if combine == TRUE
title	A string to specify the name of the combined plot
combine	logical to indicate whether to combine and visualize the plots in one panel
legend.label	String indicating the name of variable to be plotted on the legend
colors	A vector of 2 colors for the intensity, with the first element referring to the lower value and the second element referring to the higher value. Default is c('blue','yellow').
limit	A vector of lower and upper bounds for the color scale. The default option is NULL and will adjust to minimal and maximal values
...	Additional arguments from scater::plotReducedDim

**Value**

A combined ggplot object or a list of ggplots if combine == FALSE

**Author(s)**

Xiaosai Yao, Shang-yang Chen

**Examples**

```
# create a mock singleCellExperiment object for gene expression matrix
example_sce <- scuttle::mockSCE()
example_sce <- scuttle::logNormCounts(example_sce)
example_sce <- scater::runPCA(example_sce)
example_sce <- scater::runUMAP(example_sce)
example_sce$cluster <- sample(LETTERS[1:5], ncol(example_sce), replace = TRUE)
plotActivityDim(sce = example_sce, activity = logcounts(example_sce),
tf = c('Gene_0001', 'Gene_0002'), label = 'cluster')
```

---

plotActivityViolin	<i>Generate violin plots of inferred activities for a list of TFs grouped by cluster/group labels</i>
--------------------	---

---

**Description**

Generate violin plots of inferred activities for a list of TFs grouped by cluster/group labels

**Usage**

```
plotActivityViolin(
  activity_matrix,
  tf,
  clusters,
  ncol = NULL,
  nrow = NULL,
  combine = TRUE,
  legend.label = "activity",
  colors = NULL,
  title = NULL,
  text_size = 10,
  facet_grid_variable = NULL,
  boxplot = FALSE
)
```

**Arguments**

activity_matrix	A matrix of TF activities inferred from calculateActivity
tf	A character vector indicating the names of the transcription factors to be plotted
clusters	A vector of cluster or group labels for single cells
ncol	A integer to indicate the number of columns in the combined plot, if combine = TRUE
nrow	A integer to indicate the number of rows in the combined plot, if combine = TRUE
combine	logical to indicate whether to combine and visualize the plots in one panel
legend.label	String indicating the name of variable to be plotted on the legend
colors	A character vector representing the names of colors
title	String indicating the title of the plot if combine = TRUE

**text\_size**            Scalar indicating the font size of the title  
**facet\_grid\_variable**            A character vector of a secondary label to split the plots by facet\_grid  
**boxplot**                logical indicating whether to add boxplot on top of violin plot

**Value**

A combined ggplot object or a list of ggplots if combine = FALSE

**Author(s)**

Xiaosai Yao, Shang-yang Chen

**Examples**

```

# create a mock singleCellExperiment object for gene expression matrix
example_sce <- scuttle::mockSCE()
example_sce <- scuttle::logNormCounts(example_sce)
example_sce$cluster <- sample(LETTERS[1:5], ncol(example_sce), replace = TRUE)
plotActivityViolin(activity_matrix = logcounts(example_sce),
  tf = c('Gene_0001', 'Gene_0002'), clusters = example_sce$cluster)
  
```

---

plotBubble	<i>Generate bubble plots of relative activities across cluster/group labels for a list of TFs</i>
------------	---

---

**Description**

Generate bubble plots of relative activities across cluster/group labels for a list of TFs

**Usage**

```

plotBubble(
  activity_matrix,
  tf,
  clusters,
  bubblesize = c("log.FDR", "summary.logFC", "summary.diff"),
  color.theme = "viridis",
  legend.label = "relative_activity",
  x.label = "clusters",
  y.label = "transcription factors",
  title = "TF activity",
  ...
)
  
```

**Arguments**

activity_matrix	A matrix of TF activities inferred from calculateActivity
tf	A character vector indicating the names of the transcription factors to be plotted
clusters	A character or integer vector of cluster or group labels for single cells
bubblesize	String indicating the variable from findDifferentialActivity output to scale size of bubbles by either log.FDR, summary.logFC or summary.diff. Default is logFDR.
color.theme	String indicating the color theme used for the bubble plot and corresponding to the color options in scale_color_viridis_c
legend.label	String indicating the name of legend corresponding to the color scale
x.label	String indicating the x axis label
y.label	String indicating the y axis label
title	String indicating the title of the plot
...	Additional arguments to pass to findDifferentialActivity

**Value**

A ggplot object

**Author(s)**

Shang-yang Chen

**Examples**

```
example_sce <- scuttle::mockSCE()
example_sce <- scuttle::logNormCounts(example_sce)
example_sce$cluster <- sample(LETTERS[1:5], ncol(example_sce), replace = TRUE)
plotBubble(activity_matrix = logcounts(example_sce),
           tf = c('Gene_0001', 'Gene_0002'), clusters = example_sce$cluster)
```

---

plotDiffNetwork

*Plot graph according to grouping factor*

---

**Description**

Plot graph with separate weights for different levels of the grouping factor

**Usage**

```
plotDiffNetwork(
  regulon,
  cutoff = 0.01,
  tf = NULL,
  weight = "weight",
  clusters,
  layout = "stress"
)
```

**Arguments**

regulon	an object returned by the <code>getRegulon</code> or <code>addWeights</code> function
cutoff	a numeric used to select values of the variables passed in <code>clusters</code> parameter. Values greater than <code>cutoff</code> are retained and used as graph edge weights.
tf	a character vector storing the names of transcription factors to be included in the graph
weight	a string indicating the name of the column in the regulon to be used as the weight of the edges
clusters	a character vector indicating the clusters to be plotted
layout	a layout specification. Any values that are valid for <code>ggraph</code> or <code>create_layout</code> will work.

**Value**

a `ggraph` object

**Author(s)**

Xiaosai Yao, Tomasz Wlodarczyk

**Examples**

```
#' # create an artificial getRegulon output
set.seed(1234)
tf_set <- apply(expand.grid(LETTERS[1:10], LETTERS[1:10]),1, paste, collapse = '')
regulon <- S4Vectors::DataFrame(tf = sample(tf_set, 5e3, replace = TRUE))
gene_set <- expand.grid(LETTERS[1:10], LETTERS[1:10], LETTERS[1:10])
gene_set <- apply(gene_set,1,function(x) paste0(x,collapse=''))
regulon$target <- sample(gene_set, 5e3, replace = TRUE)
regulon$idxATAC <- 1:5e3
regulon$weight <- cbind(data.frame(C1 = runif(5e3), C2 = runif(5e3),
C3 = runif(5e3)))
plotDiffNetwork(regulon, tf = unique(tf_set)[1:3],
clusters = c('C1', 'C2', 'C3'), cutoff = 0.2)
```

---

`plotEpiRegulonNetwork` *Plot a graph build based on getRegulon output*

---

**Description**

This function takes an input an `igraph` object created by any of the following: `buildGraph`, `addCentrality`, `igraph::strength`, `normalizeCentrality`. It makes a force-directed layout plot to visualize it at a high level.

**Usage**

```
plotEpiRegulonNetwork(
  graph,
  layout = "stress",
  label_size = 3,
  tfs_to_highlight = NULL,
  edge_alpha = 0.02,
  point_size = 1,
  point_border_size = 0.5,
  label_alpha = 0.8,
  label_nudge_x = 0.2,
  label_nudge_y = 0.2,
  ...
)
```

**Arguments**

graph	an igraph object
layout	a layout specification. Any values that are valid for <a href="#">ggraph</a> or <a href="#">create_layout</a> will work. Defaults to 'stress'. Consider also trying 'mds', 'nicely', and 'fr' while you experiment.
label_size	an integer indicating how large the labels of highlighted transcription factors should be
tfs_to_highlight	a character vector specifying which TFs in the plot should be highlighted. Defaults to NULL (no labels).
edge_alpha	a numeric value between 0 and 1 indicating the level of transparency to use for the edge links in the force-directed layout. Defaults to 0.02.
point_size	a numeric value indicating the size of nodes in the force-directed layout
point_border_size	a numeric value indicating the size of point borders for nodes in the force-directed layout
label_alpha	a numeric value between 0 and 1 indicating the level of transparency to use for the labels of highlighted nodes
label_nudge_x	a numeric value indicating the shift of the labels along the x axis that should be used in the force-directed layout
label_nudge_y	A numeric value indicating the shift of the labels along the y axis that should be used in the force-directed layout.
...	optional additional arguments to pass to <a href="#">create_layout</a>

**Value**

a ggraph object

**Author(s)**

Timothy Keyes, Tomasz Włodarczyk



**Examples**

```
# create an artificial getRegulon output
set.seed(1234)
tf_set <- apply(expand.grid(LETTERS[seq_len(5)], LETTERS[seq_len(5)]),1, paste, collapse = '')
regulon <- data.frame(tf = sample(tf_set, 5e2, replace = TRUE))
gene_set <- expand.grid(LETTERS[seq_len(5)], LETTERS[seq_len(5)], LETTERS[seq_len(5)])
gene_set <- apply(gene_set,1,function(x) paste0(x,collapse=''))
regulon$target <- sample(gene_set, 5e2, replace = TRUE)
regulon$idxATAC <- seq_len(5e2)
regulon$corr <- runif(5e2)*0.5+0.5
regulon$weights <- runif(500)
#create igraph object
graph_tripartite <- buildGraph(regulon, mode = 'tripartite')
plotEpiregulonNetwork(graph_tripartite, tfs_to_highlight = sample(unique(tf_set),3),
edge_alpha = 0.2)
```

plotGseaNetwork

*Plot networks graph of significant genesets from regulonEnrich results***Description**

Plot networks graph of significant genesets from regulonEnrich results

**Usage**

```
plotGseaNetwork(
  tf,
  enrichresults,
  ntop_pathways = 10,
  p.adj_cutoff = 0.05,
  layout = "sugiyama",
  tf_label = "tf",
  gset_label = "ID",
  tf_color = "tomato",
  gset_color = "grey"
)
```

**Arguments**

tf	A vector of gene names to be plotted. They should be present in enrichresults
enrichresults	Output from regulonEnrich that computes enriched genesets from user-specified regulons of interest
ntop_pathways	An integer indicating the number of top pathways to be included in the graph
p.adj_cutoff	A scalar indicating the p.adjusted cutoff for pathways to be included in the graph. Default value is 0.05
layout	String indicating layout option from igraph
tf_label	String indicating the name of the tf label
gset_label	String indicating the name of the geneset label
tf_color	String indicating the color of the tf label
gset_color	String indicating the color of the geneset label

**Value**

an igraph plot of interconnected pathways through TFs

**Author(s)**

Phoebe Guo, Xiaosai Yao

**Examples**

```
AR <- data.frame(ID = c('ANDROGEN RESPONSE', 'PROLIFERATION', 'MAPK'),
  p.adjust = c(0.001, 0.01, 0.04))
GATA6 <- data.frame(ID = c('STK33', 'PROLIFERATION', 'MAPK'),
  p.adjust = c(0.001, 0.01, 0.04))
enrichresults <- list(AR = AR, GATA6 = GATA6)
plotGseaNetwork(tf = names(enrichresults), enrichresults = enrichresults)
```

---

plotHeatmapActivity *Plot transcription factor activity*

---

**Description**

Plot transcription factor activity

**Usage**

```
plotHeatmapActivity(
  activity_matrix,
  sce,
  tfs,
  downsample = 1000,
  scale = TRUE,
  center = TRUE,
  color_breaks = c(-2, 0, 2),
  colors = c("blue", "white", "red"),
  cell_attributes = NULL,
  col_gap = NULL,
  use_raster = TRUE,
  raster_quality = 10,
  cluster_rows = TRUE,
  cluster_columns = FALSE,
  border = TRUE,
  show_column_names = FALSE,
  ...
)
```

**Arguments**

**activity\_matrix** A matrix of values, such as TF activities inferred from calculateActivity

**sce** A SingleCellExperiment object containing information of cell attributes

**tfs** A character vector indicating the names of the transcription factors to be plotted

downsample	Integer indicating the number of cells to sample from the matrix
scale	Logical indicating whether to scale the heatmap
center	Logical indicating whether to center the heatmap
color_breaks	A vector indicating numeric breaks as input to <code>circlize::colorRamp2</code>
colors	A vector of colors corresponding to values in breaks as input to <code>circlize::colorRamp2</code>
cell_attributes	A character vector matching the column names of <code>colData(sce)</code> to be used for plotting
col_gap	String indicating the cell attribute to split the columns of the heatmap by
use_raster	Logical indicating whether to use rasterization to reduce image size
raster_quality	Integer indicating the raster quality. The higher the value, the better the resolution
cluster_rows	Logical indicating whether to cluster rows
cluster_columns	Logical indicating whether to cluster columns
border	Logical indicating whether to add border around heatmap
show_column_names	Logical indicating whether to show column names
...	other arguments for <code>ComplexHeatmap::Heatmap</code>

**Value**

A Heatmap-class object.

**Author(s)**

Xiaosai Yao

**Examples**

```
example_sce <- scuttle::mockSCE()
example_sce <- scuttle::logNormCounts(example_sce)
example_sce$cluster <- sample(LETTERS[1:5], ncol(example_sce), replace = TRUE)
activity_matrix <- matrix(rnorm(10*200), nrow=10, ncol=200)
rownames(activity_matrix) <- sample(rownames(example_sce), 10)
plotHeatmapActivity(activity_matrix=activity_matrix, sce=example_sce,
  tfs=rownames(activity_matrix), cell_attributes='cluster', col_gap='cluster')
```

---

plotHeatmapRegulon *Plot targets genes of transcription factors in regulons*

---

**Description**

Plot targets genes of transcription factors in regulons

**Usage**

```

plotHeatmapRegulon(
  sce,
  tfs,
  regulon,
  regulon_column = "weight",
  regulon_cutoff = 0.1,
  downsample = 1000,
  scale = TRUE,
  center = TRUE,
  color_breaks = c(-2, 0, 2),
  colors = c("blue", "white", "red"),
  cell_attributes,
  col_gap = NULL,
  exprs_values = "logcounts",
  use_raster = TRUE,
  raster_quality = 10,
  cluster_rows = FALSE,
  cluster_columns = FALSE,
  border = TRUE,
  show_column_names = FALSE,
  column_col = NULL,
  row_col = NULL,
  genes_label = NULL,
  ...
)

```

**Arguments**

<code>sce</code>	A <code>SingleCellExperiment</code> object containing information of cell attributes
<code>tfs</code>	A character vector indicating the names of the transcription factors to be plotted
<code>regulon</code>	A dataframe of regulons containing <code>tf</code> , <code>targets</code> and a column for filtering the regulons
<code>regulon_column</code>	String indicating the column names to be used for filtering regulons
<code>regulon_cutoff</code>	A scalar indicating the minimal value to retain the regulons for plotting
<code>downsample</code>	Integer indicating the number of cells to sample from the matrix
<code>scale</code>	Logical indicating whether to scale the heatmap
<code>center</code>	Logical indicating whether to center the heatmap
<code>color_breaks</code>	A vector indicating numeric breaks as input to <code>circlize::colorRamp2</code>
<code>colors</code>	A vector of colors corresponding to values in breaks as input to <code>circlize::colorRamp2</code>
<code>cell_attributes</code>	A character vector matching the column names of <code>colData(sce)</code> to be used for plotting
<code>col_gap</code>	String indicating the cell attribute to split the columns of the heatmap by
<code>exprs_values</code>	A string specifying which assay in <code>assays(object)</code> to obtain expression values from
<code>use_raster</code>	Logical indicating whether to use rasterization to reduce image size
<code>raster_quality</code>	Integer indicating the raster quality. The higher the value, the better the resolution

cluster_rows	Logical indicating whether to cluster rows
cluster_columns	Logical indicating whether to cluster columns
border	Logical indicating whether to add border around heatmap
show_column_names	Logical indicating whether to show column names
column_col	A list specifying the colors in the columns. See <a href="#">here</a>
row_col	A list specifying the colors in the rows. See <a href="#">here</a>
genes_label	A character vector indicating a selected list of genes to show on the rownames
...	other arguments for <code>ComplexHeatmap::Heatmap</code>

**Value**

A Heatmap-class object.

**Author(s)**

Xiaosai Yao

**Examples**

```
example_sce <- scuttle::mockSCE()
example_sce <- scuttle::logNormCounts(example_sce)
example_sce$cluster <- sample(LETTERS[1:5], ncol(example_sce), replace = TRUE)
regulon <- data.frame(tf=c(rep('Gene_0001',10),rep('Gene_0002',20)),
  target = sample(rownames(example_sce),30), weight = rnorm(30))
#plot heatmap and rotate labels
plotHeatmapRegulon(example_sce, tfs=c('Gene_0001','Gene_0002'), regulon=regulon,
  cell_attributes='cluster', col_gap = 'cluster', column_title_rot = 90)
```

---

regulon

*regulon created using epi regulon package from reprogram-seq data*

---

**Description**

regulon created using epi regulon package from reprogram-seq data

**Usage**

```
data(regulon)
```

**Format**

a DFrame.

**Value**

a DFrame.

**Examples**

```
data(regulon)
```

---

regulonEnrich	<i>Perform geneset enrichment of user-defined regulons</i>
---------------	--

---

**Description**

Perform geneset enrichment of user-defined regulons

**Usage**

```
regulonEnrich(TF, regulon, weight = "weight", weight_cutoff = 0.5, genesets)
```

**Arguments**

TF	A character vector of TF names
regulon	A matrix of weighted regulon consisting of tf, targets, corr and weight
weight	String indicating the column name that should be used to filter target genes for geneset enrichment. Default is 'weight'.
weight_cutoff	A numeric scalar to indicate the cutoff to filter on the column specified by weight. Default is 0.5.
genesets	A dataframe with the first column being the name of the geneset and the second column being the name of the genes

**Value**

A dataframe showing the significantly enriched pathways

**Author(s)**

Xiaosai Yao

**Examples**

```
#retrieve genesets
msigdb.hs = msigdb::getMsigdb(org = 'hs', id = 'SYM', version = '7.4')

#convert genesets to be compatible with enricher
msigdb.hs <- msigdb.hs[unlist(lapply(msigdb.hs, function(x) {GSEABase::bcCategory(GSEABase::collectionType(x)
gs.list <- do.call(rbind, lapply(names(msigdb.hs), function(x) {
  data.frame(gs = x, genes = GSEABase::geneIds(msigdb.hs[x][[1]]))}))

head(gs.list)

#get regulon
library(dorothea)
data(dorothea_hs, package = 'dorothea')
regulon <- dorothea_hs
enrichment_results <- regulonEnrich(c('ESR1','AR'), regulon = regulon, weight = 'mor',
genesets = gs.list)
```

# Index

- \* **datasets**
  - regulon, [21](#)
- addCentrality (buildGraph), [2](#)
- buildDiffGraph (buildGraph), [2](#)
- buildGraph, [2](#)
- calculateJaccardSimilarity, [5](#)
- combineMarkers, [7](#)
- create\_layout, [15](#), [16](#)
- enrichPlot, [6](#)
- findDifferentialActivity, [7](#)
- findPartners, [8](#)
- getSigGenes, [9](#)
- ggraph, [15](#), [16](#)
- normalizeCentrality (buildGraph), [2](#)
- permuteGraph, [10](#)
- plotActivityDim, [10](#)
- plotActivityViolin, [12](#)
- plotBubble, [13](#)
- plotDiffNetwork, [14](#)
- plotEpiregulonNetwork, [15](#)
- plotGseaNetwork, [17](#)
- plotHeatmapActivity, [18](#)
- plotHeatmapRegulon, [19](#)
- rankTfs (buildGraph), [2](#)
- regulon, [21](#)
- regulonEnrich, [22](#)