Package 'epiregulon'

November 13, 2026

Title Gene regulatory network inference from single cell epigenomic data

Version 2.3.3

Date 2026-06-14

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 8.0.0

Imports AnnotationHub, BiocParallel, ExperimentHub, Matrix, Rcpp, S4Vectors, SummarizedExperiment, checkmate, entropy, methods, stats, utils, AnnotationHub, GenomeInfoDb, GenomicRanges, BSgenome.Hsapiens.UCSC.hg19, BSgenome.Hsapiens.UCSC.hg38, BSgenome.Mmusculus.UCSC.mm10, motifmatchr, IRanges, scrapper, metapod, matrixStats

Depends R (>= 4.5.0), SingleCellExperiment

Suggests knitr, rmarkdown, parallel, BiocStyle, testthat (>= 3.0.0), coin, scMultiome, scran, scuttle

LinkingTo Rcpp

VignetteBuilder knitr

URL https://github.com/xiaosaiyao/epiregulon/

biocViews SingleCell, GeneRegulation,NetworkInference,Network, GeneExpression, Transcription, GeneTarget

Config/testthat/edition 3

BugReports https://github.com/xiaosaiyao/epiregulon/issues

Config/roxygen2/version 8.0.0

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

git_branch devel

git_last_commit 713a0f9

git_last_commit_date 2026-07-01

Repository Bioconductor 3.24

Date/Publication 2026-11-12

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

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

Contents

epiregulon-package 2
addLogFC 3
addMotifScore 4
addTFMotifInfo 5
addWeights 7
aggregateAcrossCellsFast 9
calculateActivity 11
calculateP2G 13
findMarkersSimple 16
getRegulon 17
getTFMotifInfo 18
optimizeMetacellNumber 19
pruneRegulon 21

Index **24**

`epiregulon-package` *epiregulon: Gene regulatory network inference from single cell epigenomic data*

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.

Author(s)

Maintainer: Xiaosai Yao <xiaosai.yao@gmail.com> (ORCID)

Authors:

- Xiaosai Yao <xiaosai.yao@gmail.com> (ORCID)
- Tomasz Włodarczyk <tomwlo@gmail.com> (ORCID)
- Aaron Lun <infinite.monkeys.with.keyboards@gmail.com>
- Shang-Yang Chen <sychen9584@gmail.com>

See Also

Useful links:

- https://github.com/xiaosaiyao/epiregulon/
- Report bugs at https://github.com/xiaosaiyao/epiregulon/issues

---

addLogFC *Add log fold changes of gene expression to regulons*

Description

Add log fold changes of gene expression to regulons

Usage

```
addLogFC(
  expMatrix,
  clusters,
  regulon,
  direction = c("any", "up", "down"),
  pval.type = c("any", "some", "all"),
  sig_type = c("log.FDR", "log.p.value"),
  logFC_condition = NULL,
  logFC_ref = NULL,
  min.prop = NULL,
  assay.type = "logcounts"
)
```

Arguments

expMatrix	A SingleCellExperiment object or matrix containing gene expression with genes in the rows and cells in the columns. Gene expression should be in logcounts
clusters	A character or integer vector of cluster or group labels for single cells
regulon	A data frame informing the gene regulatory relationship with the `tf` column representing transcription factors, `idxATAC` corresponding to the index in the peakMatrix and `target` column representing target genes
direction	tring specifying the direction of log-fold changes to be considered in the alternative hypothesis.
pval.type	String specifying how p-values are to be combined across pairwise comparisons for a given group/cluster.
sig_type	String specifying whether to use "FDR" or "p.value" for sig_cutoff
logFC_condition	A scalar or vector of string indicating the sample names to be compared against `logFC_ref`
logFC_ref	A scalar indicating the reference sample used to compute logFC. Default value is `rest` which is an average of all pairwise comparisons. Users can also specify a reference sample, for example, `DMSO`.
min.prop	Numeric scalar specifying the minimum proportion of significant comparisons per gene. Defaults to 0.5 when pval.type="some", otherwise defaults to zero.
assay.type	A string specifying which assay values to use, usually "logcounts".

Details

The choice of `pval.type` determines whether the highly ranked genes are those that are DE between the current group and:

- any other group (`"any"`)
- all other groups (`"all"`)
- some other groups (`"some"`)

Adapted from the deprecated scran::findMarkers function

Value

A DataFrame of regulons with additional columns of logFC and significance

Author(s)

Xiaosai Yao

Examples

```
# create a mock SingleCellExperiment object for gene expression matrix
set.seed(1000)
gene_sce <- scuttle::mockSCE()
gene_sce <- scrapper::normalizeRnaCounts.se(gene_sce)
rownames(gene_sce) <- paste0('Gene_',1:2000)

# create a mock regulon
regulon <- data.frame(tf=c(rep('Gene_1',10), rep('Gene_2',10)),
    idxATAC=sample(1:100, 20),
    target=c(paste0('Gene_', sample(3:2000,10)),
    paste0('Gene_',sample(3:2000,10))))

# filter regulon
pruned.regulon <- addLogFC(expMatrix=gene_sce,
    clusters=gene_sce$Treatment,
    regulon=regulon,
    sig_type="log.p.value")
```

`addMotifScore` *Add Motif Scores*

Description

Add Motif Scores

Usage

```
addMotifScore(
  regulon,
  field_name = "motif",
  peaks = NULL,
  pwms = NULL,
  species = c("human", "mouse"),
  genome = c("hg38", "hg19", "mm10"),
  ...
)
```

Arguments

regulon	A DataFrame consisting of tf (regulator) and target in the column names.
field_name	Character string indicating the column name of the regulon to add the motif information to
peaks	A GRanges object indicating the peaks to perform motif annotation on if ArchR project is not provided. The peak indices should match the re column in the regulon
pwms	A PWMatrixList for annotation of motifs using 'motifmatchr::matchMotifs'
species	Character string indicating species. Currently supported species is human or mouse
genome	Character string indicating the genomic build
...	Additional arguments to pass into motifmatchr::matchMotifs

Value

A DataFrame with motif matches added with 1s indicating the presence of motifs and 0s indicating the absence of motifs

Examples

```
regulon <- S4Vectors::DataFrame(tf=c('AR','AR','AR','ESR1','ESR1','NKX2-1'),
idxATAC=1:6)
peaks <- GenomicRanges::GRanges(seqnames=c('chr12','chr19','chr19','chr11','chr6','chr1'),
ranges=IRanges::IRanges(start=c(124914563,50850845, 50850844, 101034172, 151616327, 1000),
end=c(124914662,50850929, 50850929, 101034277, 151616394,2000)))
regulon <- addMotifScore(regulon, peaks=peaks)
```

---

addTFMotifInfo *Add TF binding motif occupancy information to the peak2gene object*

Description

Add TF binding motif occupancy information to the peak2gene object

Usage

```
addTFMotifInfo(p2g, grl, peakMatrix = NULL)
```

Arguments

p2g	A Peak2Gene dataframe created by ArchR or getP2Glinks() function
grl	GRangeList object containing reference TF binding information. We recommend retrieving grl from getTFMotifInfo which contains TF occupancy data derived from public and ENCODE ChIP-seq peaks. Alternatively, if the users would like to provide a GRangeList of motif annotations. This can be derived using motifmatchr::matchMotifs. See details
peakMatrix	A matrix of scATAC-seq peak regions with peak ids as rows

Details

This function annotates each regulatory element with possible transcription factors. We can either provide a GRangeList of known ChIP-seq binding sites (TF occupancy) or positions of TF motifs (TF motifs). While public ChIP-seq data may not fully align with the ground truth TF occupancy in users' data (due to technical challenges of ChIP-seq or cell type nature of TF occupancy), it does offer a few important advantages over TF motif information:

1. TF occupancy allows co-activators to be included. Co-activators are chromatin modifiers that do not directly bind to DNA but nonetheless play an important role in gene regulation
2. TF occupancy can distinguish between members of the same class that may share similar motifs but that may have drastically different binding sites

If multiple ChIP-seq are available for the same TF, we merge the ChIP-seq data to represent an universal set of possible binding sites. The predicted TF occupancy is further refined by pruneRegulon.

If the users prefer to use TF motifs instead of TF occupancy, the users can create a GRangeList of motif annotation using motifmatchr::matchMotifs. Here, we demonstrate how to annotate peaks with cisbp motif database

```
library(motifmatchr)
library(chromVARmotifs)
data("human_pwms_v1")
peaks <- GRanges(seqnames=c("chr1","chr2","chr2"),
                         ranges=IRanges(start=c(76585873,42772928, 100183786),
                                                              width=500))
eh <- AnnotationHub::query(ExperimentHub::ExperimentHub(),
pattern=c("scMultiome", "TF motifs", "human"))
pwms <- readRDS(eh[[eh$ah_id]]))
grl <- matchMotifs(pwms, peaks, genome="hg38", out="positions")
retain only TF symbols. TF symbols need to be consistent with gene names in regulon
names(grl) <- sapply(strsplit(names(grl), "_"), "[",3)
```

Value

A data frame containing overlapping ids of scATAC-seq peak regions and reference TF binding regions

Author(s)

Xiaosai Yao, Shang-yang Chen

Examples

```
set.seed(1)
# create a mock peak-to-gene matrix
p2g <- data.frame(idxATAC=c(rep(1,5), rep(2,5)), Chrom='chr1', idxRNA=1:10,
Gene=paste0('Gene_',1:10),Correlation=runif(10, 0,1))

# create mock a GRanges list of TF binding sites
grl <- GRangesList('TF1'=GRanges(seqnames='chr1',
ranges=IRanges(start=c(50,1050), width=100)),
'TF2'=GRanges(seqnames='chr1',
ranges=IRanges(start=c(1050), width=100))
)

# create a mock SingleCellExperiment object for peak matrix
peak_gr <- GRanges(seqnames='chr1',
             ranges=IRanges(start=seq(from=1, to=10000, by=1000),
             width=100))
peak_counts <- matrix(sample(x=0:4, size=100*length(peak_gr), replace=TRUE),
nrow=length(peak_gr), ncol=100)
peak_sce <- SingleCellExperiment(list(counts=peak_counts))
rowRanges(peak_sce) <- peak_gr
rownames(peak_sce) <- paste0('peak',1:10)

# create overlaps between p2g matrix, TF binding sites and peak matrix
overlap <- addTFMotifInfo(p2g, grl, peakMatrix=peak_sce)
utils::head(overlap)
```

`addWeights` *Calculate weights for the regulons by computing co-association between TF and target gene expression*

Description

Calculate weights for the regulons by computing co-association between TF and target gene expression

Usage

```
addWeights(
  regulon,
  expMatrix = NULL,
  peakMatrix = NULL,
  exp_assay = "logcounts",
  peak_assay = "PeakMatrix",
  method = c("wilcoxon", "corr", "MI"),
  clusters = NULL,
  exp_cutoff = 1,
  peak_cutoff = 0,
  block_factor = NULL,
  min_targets = 10,
  tf_re.merge = FALSE,
  aggregateCells = FALSE,
```

```
  useDim = "IterativeLSI_ATAC",
  cellNum = 10,
  BPPARAM = BiocParallel::SerialParam(progressbar = TRUE)
)
```

Arguments

regulon	A DataFrame object consisting of tf (regulator) and target in the column names.
expMatrix	A SingleCellExperiment object containing gene expression information
peakMatrix	A SingleCellExperiment object or matrix containing peak accessibility with peaks in the rows and cells in the columns
exp_assay	String specifying the name of the assay to be retrieved from the SingleCellExperiment object
peak_assay	String indicating the name of the assay in peakMatrix for chromatin accessibility
method	String specifying the method of weights calculation. Three options are available: wilcoxon, corr, and MI.
clusters	A vector corresponding to the cluster labels of the cells
exp_cutoff	A scalar indicating the minimum gene expression for transcription factor above which cell is considered as having expressed transcription factor.
peak_cutoff	A scalar indicating the minimum peak accessibility above which peak is considered open.
block_factor	String specifying the field in the colData of the SingleCellExperiment object to be used as blocking factor (such as batch)
min_targets	Integer specifying the minimum number of targets for each tf in the regulon with 10 targets as the default
tf_re.merge	A logical to indicate whether to consider both TF expression and chromatin accessibility. See details.
aggregateCells	A logical to indicate whether to aggregate cells into groups determined by cellNum. This option can be used to overcome data sparsity when using wilcoxon.
useDim	String indicating the name of the dimensionality reduction matrix in expMatrix used for cell aggregation
cellNum	A numeric specifying the number of cells per cluster for cell aggregation. Default is 10.
BPPARAM	A BiocParallelParam object specifying whether summation should be parallelized. Use BiocParallel::SerialParam() for serial evaluation and use BiocParallel::MulticoreParam() for parallel evaluation

Details

This function estimates the regulatory potential of transcription factor on its target genes, or in other words, the magnitude of gene expression changes induced by transcription factor activity, using one of the four methods:

- corr - correlation between TF and target gene expression
- MI - mutual information between the TF and target gene expression
- wilcoxon - effect size of the Wilcoxon test between target gene expression in cells jointly expressing all 3 elements vs cells that do not

Two measures (`corr` and `wilcoxon`) give both the magnitude and directionality of changes whereas `MI` always outputs positive weights. The correlation and mutual information statistics are computed on the pseudobulked gene expression or accessibility matrices, whereas the Wilcoxon method groups cells based on the joint expression of TF, RE and TG in each single cell.

When using the `corr` method, the default practice is to compute weights by correlating the pseudobulk target gene expression vs the pseudobulk TF gene expression. However, often times, an inhibitor of TF does not alter the gene expression of the TF. In rare cases, cells may even compensate by increasing the expression of the TF. In this case, the activity of the TF, if computed by TF-TG correlation, may show a spurious increase in its activity. As an alternative to gene expression, we may correlate the product of TF and RE against TG. When `tf_re.merge` is `TRUE`, we take the product of the gene expression and chromatin accessibility.

Value

A DataFrame with columns of corr and/or MI added to the regulon. TFs not found in the expression matrix and regulons not meeting the minimal number of targets were filtered out.

Author(s)

Xiaosai Yao, Shang-yang Chen, Tomasz Wlodarczyk

Examples

```
# create a mock SingleCellExperiment object for gene expression matrix
expMatrix <- scuttle::mockSCE()
expMatrix <- scrapper::normalizeRnaCounts.se(expMatrix)
expMatrix$cluster <- sample(LETTERS[1:5], ncol(expMatrix), replace=TRUE)

# create a mock SingleCellExperiment object for peak matrix
peakMatrix <- scuttle::mockSCE()
rownames(peakMatrix) <- 1:2000

# create a mock regulon
regulon <- S4Vectors::DataFrame(tf=c(rep('Gene_0001',5), rep('Gene_0002',10)),
    idxATAC=1:15, target=c(paste0('Gene_000',2:6), paste0('Gene_00',11:20)))

# add weights to regulon
regulon.w <- addWeights(regulon=regulon, expMatrix=expMatrix, exp_assay='logcounts',
    peakMatrix=peakMatrix, peak_assay='counts', clusters=expMatrix$cluster,
    min_targets=5, method='wilcox')

# add weights with cell aggregation
expMatrix <- scrapper::runPca.se(expMatrix, features=NULL)
regulon.w <- addWeights(regulon=regulon, expMatrix=expMatrix, exp_assay='logcounts',
    peakMatrix=peakMatrix, peak_assay='counts', clusters=expMatrix$cluster,
    min_targets=5, method='wilcox', aggregateCells=TRUE, cellNum=3, useDim='PCA')
```

`aggregateAcrossCellsFast`

Aggregate cells in SingleCellExperiment

Description

Aggregate expression values across cells in SingleCellExperiment based on a grouping factor. This is primarily used to create pseudo-bulk profiles for each cluster/sample combination. It is wrapped around `scrapper::aggregateAcrossCells`, which relies on the C++ code.

Usage

```
aggregateAcrossCellsFast(
  sce,
  clusters,
  assay.name = "counts",
  fun_name = c("mean", "sum"),
  num.threads = 1,
  aggregateColData = TRUE
)
```

Arguments

`sce`	A SingleCellExperiment, SummarizedExperiment or RangedSummarizedExperiment object
`clusters`	A vector used as a grouping variable. The length should be equal to the number of cells.
`assay.name`	A character indicating the name of the assay containing the values to be aggregated.
`fun_name`	A character indicating the function used to aggregate data. The selection is restricted to "mean" or "sum".
`num.threads`	Integer specifying the number of threads to be used for aggregation.
`aggregateColData`	A logical specifying if the columns in the `colData` should be included in the output object. Only those columns are selected which can be decomposed by grouping variable into the vectors whose all elements are the same.

Value

A SingleCellExperiment object containing aggregated cells.

Examples

```
# create a mock SingleCellExperiment object for gene expression matrix
set.seed(1000)
example_sce <- scuttle::mockSCE()
ids <- sample(LETTERS[1:5], ncol(example_sce), replace=TRUE)
out <- aggregateAcrossCellsFast(example_sce, ids)
```

`calculateActivity` *Calculate the per cell activity of master regulators based on a regulon*

Description

Calculate the per cell activity of master regulators based on a regulon

Usage

```
calculateActivity(
  expMatrix = NULL,
  exp_assay = "logcounts",
  regulon = NULL,
  normalize = FALSE,
  mode = "weight",
  method = NULL,
  genesets = NULL,
  clusters = NULL,
  FUN = c("mean", "sum")
)
```

Arguments

`expMatrix`	A SingleCellExperiment object containing gene expression information with rows representing genes and columns represent cells. Rownames (either gene symbols or geneID) must be consistent with the naming convention in the regulon.
`exp_assay`	String specifying the name of the assay to be retrieved from the SingleCellExperiment object. Set to 'logcounts' as the default
`regulon`	A DataFrame object consisting of tf (regulator) and target in the column names, with additional columns indicating degree of association between tf and target such as 'mor' or 'corr' obtained from `addWeights`.
`normalize`	Logical indicating whether row means should be subtracted from expression matrix. default is FALSE
`mode`	String indicating the name of column to be used as the weights
`method`	String indicating the method for calculating activity. Available methods are weightedMean or aucell. Deprecated.
`genesets`	A list of genesets. Each list element can be a dataframe with the first column indicating the genes and second column indicating the weights. Alternatively, each list element is a character vector corresponding to the genes in the geneset. A feature set collection in the form of CompressedSplitDataFrameList that contains genes in the first column and weights in the second column. See details
`clusters`	A vector indicating cluster assignment
`FUN`	function to aggregate the weights

Details

This function calculates activity score from a regulon that is a DataFrame consisting of a tf column, a target column and a weight column. Alternatively, instead of a regulon, this function also accepts weighted signature sets where each gene set or signature is a data frame or unweighted signature sets where each gene set is a character vector. The user has the option of computing signature score by weighted mean of target gene expression or the relative ranking of the target genes computed by AUCell.

Value

A matrix of inferred transcription factor (row) activities in single cells (columns)

Author(s)

Xiaosai Yao, Shang-yang Chen

Examples

```
# create a mock SingleCellExperiment object for gene expMatrixession matrix
set.seed(1000)
gene_sce <- scuttle::mockSCE()
gene_sce <- scrapper::normalizeRnaCounts.se(gene_sce)
rownames(gene_sce) <- paste0('Gene_',1:2000)

# create a mock SingleCellExperiment object for peak matrix
peak_gr <- GRanges(seqnames='chr1',
                   ranges=IRanges(start=seq(from=1, to=10000, by=100), width=100))
peak_counts <- matrix(sample(x=0:4, size=ncol(gene_sce)*length(peak_gr), replace=TRUE),
                      nrow=length(peak_gr), ncol=ncol(gene_sce))
peak_sce <- SingleCellExperiment(list(counts=peak_counts), colData=colData(gene_sce))
rownames(peak_sce) <- paste0('Peak_',1:100)

# create a mock regulon
regulon <- data.frame(tf=c(rep('Gene_1',10), rep('Gene_2',10)),
                      idxATAC=sample(1:100, 20),
                      target=c(paste0('Gene_', sample(3:2000,10)),
                                paste0('Gene_',sample(3:2000,10))))

#  prune regulon
pruned.regulon <- pruneRegulon(expMatrix=gene_sce,
                               exp_assay='logcounts',
                               peakMatrix=peak_sce,
                               peak_assay='counts',
                               regulon=regulon,
                               clusters=gene_sce$Treatment,
                               regulon_cutoff=0.5,
                               p_adj=TRUE)

regulon.w <- addWeights(regulon=regulon,
                        expMatrix=gene_sce,
                        clusters=gene_sce$Treatment,
                        exp_assay='logcounts',
                        min_targets=5,
                        method='corr')

# calculate activity
```

```
activity <- calculateActivity(expMatrix=gene_sce,
                              regulon=regulon.w,
                              exp_assay='logcounts')

# calculate cluster-specific activity if cluster-specific weights are supplied
regulon.w$weight <- matrix(runif(nrow(regulon.w)*2, -1,1), nrow(regulon.w),2)
colnames(regulon.w$weight) <- c('treat1','treat2')

activity.cluster <- calculateActivity(gene_sce,
regulon=regulon.w, clusters=gene_sce$Treatment,
exp_assay='logcounts', FUN='mean')

# compute signature scores from weighted genesets
weighted_genesets <- list(set1=data.frame(genes=c('Gene_1', 'Gene_2', 'Gene_3'),
weights=c(1,2,3)), set2=data.frame(genes=c('Gene_4', 'Gene_5', 'Gene_6'), weights=c(4,5,6)))

activity <- calculateActivity(gene_sce, genesets=weighted_genesets)

# compute signature scores from unweighted genesets
unweighted_genesets <- list(set1=c('Gene_1', 'Gene_2', 'Gene_3'),
                            set2=c('Gene_4', 'Gene_5', 'Gene_6'))
activity <- calculateActivity(gene_sce, genesets=unweighted_genesets)
```

`calculateP2G` *Establish peak to gene links based on correlations between ATAC-seq peaks and RNA-seq genes*

Description

Establish peak to gene links based on correlations between ATAC-seq peaks and RNA-seq genes

Usage

```
calculateP2G(
  peakMatrix = NULL,
  expMatrix = NULL,
  reducedDim = NULL,
  useDim,
  cutoff_stat = c("p_val", "FDR", "Correlation"),
  cutoff_sig = 0.05,
  cor_cutoff = 0.5,
  cellNum = 100,
  maxDist = 250000,
  exp_assay = "logcounts",
  peak_assay = "counts",
  gene_symbol = "name",
  clusters = NULL,
  cor_method = c("pearson", "spearman", "kendall"),
  assignment_method = c("correlation", "nearest"),
  frac_RNA = 0,
  frac_ATAC = 0,
  nRandConns = 1e+05,
```

```
  batch_size = 20000,
  BPPARAM = BiocParallel::SerialParam(progressbar = TRUE),
  verbose = TRUE
)
```

Arguments

peakMatrix	A SingleCellExperiment object containing counts of chromatin accessibility at each peak region or genomic bin from scATAC-seq. rowRanges should contain genomic positions of the peaks in the form of GRanges.
expMatrix	A SingleCellExperiment object containing gene expression counts from scRNA-seq. rowRanges should contain genomic positions of the genes in the form of GRanges. rowData should contain a column of gene symbols with column name matching the gene_symbol argument.
reducedDim	A matrix of dimension reduced values
useDim	String specifying the name of the reduced dimension matrix supplied by the user. Deprecated
cutoff_stat	A names of a statistic used to determine significant links to assign peak to gene links. Should be Correlation, p_val or FDR.
cutoff_sig	A numeric scalar to specify the p-value or FDR cutoff for the links between ATAC-seq peaks and RNA-seq genes . Default is set to 0.05.
cor_cutoff	A numeric scalar to specify the correlation cutoff between ATAC-seq peaks and RNA-seq genes to assign peak to gene links. Default correlation cutoff is 0.5. Takes effect only of cutoff_stat is set to Correlation.
cellNum	An object of the class CellNumSol returned by optimizeMetacellNumber or a numeric to specify the average number of cells per K-mean cluster.
maxDist	An integer to specify the base pair extension from transcription start start for overlap with peak regions
exp_assay	String indicating the name of the assay in expMatrix for gene expression
peak_assay	String indicating the name of the assay in peakMatrix for chromatin accessibility
gene_symbol	String indicating the column name in the rowData of expMatrix that corresponds to gene symbol
clusters	A vector corresponding to the cluster labels for calculation of correlations within each cluster. If left NULL, correlation is calculated across all clusters. See details for the use of clusters
cor_method	String indicating which correlation coefficient is to be computed. One of 'pearson' (default), 'kendall', or 'spearman'.
assignment_method	String indicating the method used to assign target genes to regulatory elements. 'Correlation' is based on correlation between ATAC and RNA above a correlation threshold set by cor_cutoff. 'Nearest' assigns the closest expressed gene to regulatory element meeting a correlation threshold set by cor_cutoff. Set cor_cutoff to 0 if wishing to assign the closest expressed gene without any correlation cutoff
frac_RNA	An integer to indicate the fraction of cells expressing a gene. It is used to filter the gene expression matrix for expressed genes

frac_ATAC	An integer to indication the fraction of cells showing chromatin accessibility. It is used to filter the peak Matrix for open regions
nRandConns	An integer specifying the number of false connections between regulatory elements and target genes which will be used to calculate empirical p-values of correlation coefficients
batch_size	An integer specifying how many peak–gene pairs are processed per batch during parallel correlation calculations.
BPPARAM	A BiocParallelParam object specifying whether summation should be parallelized. Use BiocParallel::SerialParam() for serial evaluation and use BiocParallel::MulticoreParam() for parallel evaluation
verbose	A boolean indicating whether messages should be emitted during computation

Details

Cluster information is sometimes helpful to avoid the Simpsons's paradox in which baseline differences between cell lines or cell types can create artificial or even inverse correlations between peak accessibility and gene expression. If Cluster information is provided, correlation is performed within cell aggregates of each cluster.

Value

A DataFrame of Peak to Gene correlation

Author(s)

Xiaosai Yao, Shang-yang Chen

Examples

```
# create a mock SingleCellExperiment object for gene expression matrix
set.seed(1000)
gene_sce <- scuttle::mockSCE()
gene_sce <- scrapper::normalizeRnaCounts.se(gene_sce)
gene_gr <- GenomicRanges::GRanges(seqnames=Rle(c('chr1', 'chr2', 'chr3','chr4'),
                  nrow(gene_sce)/4),
                  ranges=IRanges(start=seq(from=1, length.out=nrow(gene_sce), by=1000),
                  width=100))
rownames(gene_sce) <- rownames(gene_sce)
gene_gr$name <- rownames(gene_sce)
rowRanges(gene_sce) <- gene_gr

# create a mock SingleCellExperiment object for peak matrix
peak_gr <- GenomicRanges::GRanges(seqnames='chr1',
                  ranges=IRanges(start=seq(from=1, to=10000, by=1000), width=100))
peak_counts <- matrix(sample(x=0:4, size=ncol(gene_sce)*length(peak_gr), replace=TRUE),
                     nrow=length(peak_gr), ncol=ncol(gene_sce))
peak_sce <- SingleCellExperiment(list(counts=peak_counts), colData=colData(gene_sce))
rowRanges(peak_sce) <- peak_gr
rownames(peak_sce) <- paste0('peak',1:10)
# create a mock reducedDim matrix
reducedDim_mat <- matrix(runif(ncol(gene_sce)*50, min=0, max=1), nrow=ncol(gene_sce), 50)
rownames(reducedDim_mat) <- colnames(gene_sce)
p2g <- calculateP2G(peakMatrix=peak_sce, expMatrix=gene_sce, reducedDim=reducedDim_mat,
                   cellNum=20)
```

`findMarkersSimple` *Find Differential Genes*

Description

Functions adapted from the deprecated scran::findMarkers to identify the most differential genes through pairwise t-tests with p-values combined via various methods.

Usage

```
findMarkersSimple(
  x,
  groups,
  direction = c("any", "up", "down"),
  pval.type = c("any", "some", "all"),
  min.prop = NULL,
  assay.type = "logcounts",
  combined = TRUE
)
```

Arguments

`x`	A matrix or SummarizedExperiment object
`groups`	A vector of length equal to the number of cells, indicating the group of each cell
`direction`	A string specifying the direction of log-fold changes to be considered in the alternative hypothesis. See `scran::pairwiseTTests`
`pval.type`	A string specifying how p-values are to be combined across pairwise comparisons for a given group. See `scran::combineMarkers`
`min.prop`	Numeric scalar specifying the minimum proportion of significant comparisons per gene. Defaults to 0.5 when pval.type="some", otherwise defaults to zero. See `scran::combineMarkers`
`assay.type`	A string specifying which assay values to use. Default is `logcounts`
`combined`	logical indicating pairwise p-values should be combined

Value

If combined is FALSE, this function returns a list of 4 elements:

- `log.p.value`: A list of data.frame, of length equal to the number of groups, indicating the log p-values of each group member vs versus all other group members
- `log.FDR`: A list of data.frame, of length equal to the number of groups, indicating the log FDR of each group member vs versus all other group members
- `logFC`: A list of data.frame, of length equal to the number of groups, indicating the log fold changes of each group member vs versus all other group members
- `all.groups`: a vector corresponding to the unique group names

If combined is TRUE, output is a list of data.frame objects of length equal to the number of groups, indicating the combined log p-values, log FDR values and log fold changes.

Examples

```
set.seed(1000)
sce <- scuttle::mockSCE()
sce <- scrapper::normalizeRnaCounts.se(sce)
rownames(sce) <- paste0('Gene_',1:2000)
simple_output <- findMarkersSimple(sce, sce$Treatment, combined=FALSE,
    assay.type="logcounts", direction="any")
simple_output_combined <- findMarkersSimple(sce, sce$Treatment, combined=TRUE,
    assay.type="logcounts", direction="any")
```

---

getRegulon *Combine the TF binding info and peak to gene correlations to generate regulons*

Description

Combine the TF binding info and peak to gene correlations to generate regulons

Usage

```
getRegulon(p2g, overlap, aggregate = FALSE, FUN = "mean")
```

Arguments

p2g	A Peak2Gene data frame created by ArchR or getP2Glinks() function
overlap	A data frame storing overlaps between the regions of the peak matrix with the bulk TF ChIP-seq binding sites computed from addTFMotifInfo
aggregate	logical to specify whether regulatory elements are aggregated across the same TF-target pairs
FUN	function to aggregate TF-target sharing different regulatory elements

Value

A DataFrame consisting of tf(regulator), target and a column indicating degree of association between TF and target such as 'mor' or 'corr'.

Author(s)

Xiaosai Yao, Shang-yang Chen

Examples

```
set.seed(1)
# create a mock peak-to-gene matrix
p2g <- data.frame(idxATAC=c(rep(1,5), rep(2,5)), Chrom='chr1', idxRNA=1:10,
target=paste0('Gene_', 1:10), Correlation=runif(10, 0, 1))

# create a Granges list of TF binding sites
grl <- GRangesList('TF1'=GRanges(seqnames='chr1',
ranges=IRanges(start=c(50,1050), width=100)),
'TF2'=GRanges(seqnames='chr1',
ranges=IRanges(start=c(1050), width=100))
```

```
)

# Create a mock peak matrix
peak_gr <- GRanges(seqnames='chr1',
                   ranges=IRanges(start=seq(from=1, to=10000, by=1000), width=100))

peak_counts <- matrix(sample(x=0:4, size=100*length(peak_gr), replace=TRUE),
nrow=length(peak_gr),ncol=100)
peak_sce <- SingleCellExperiment(list(counts=peak_counts))
rowRanges(peak_sce) <- peak_gr
rownames(peak_sce) <- paste0('peak', 1:10)

# create overlaps between p2g matrix, TF binding sites and peak matrix
overlap <- addTFMotifInfo(p2g, grl, peakMatrix=peak_sce)
utils::head(overlap)

# aggregate gene expression if the gene is bound by the same TF at regulatory elements
regulon <- getRegulon(p2g, overlap, aggregate=FALSE)
```

`getTFMotifInfo` *Retrieve TF binding sites or motif positions*

Description

Combined transcription factor ChIP-seq data from ChIP-Atlas and ENCODE

Usage

```
getTFMotifInfo(
  genome = c("hg38", "hg19", "mm10"),
  source = c("atlas", "encode.sample", "atlas.sample", "atlas.tissue"),
  metadata = FALSE,
  mode = c("occupancy", "motif"),
  peaks = NULL,
  version = 2,
  peak_number = 1000
)
```

Arguments

`genome`	character string specifying the genomic build
`source`	character string specifying the ChIP-seq data source and data specificity. Source followed by dot and `sample` indicates sample-specific Chip-seq data. Adding `.tissue` to source string result in returning tissue specific data. Providing source name without suffix tells the function to return data merged from different tissues and samples.
`metadata`	logical flag specifying whether to return data or metadata only
`mode`	a string indicating whether to download a GRangelist of TF binding sites ('occupancy') or motif matches ('motif'). TF binding information is retrieved from `scMultiome::tfBinding()`. The motif information was obtained from chromVARmotifs (human_pwms_v2 and mouse_pwms_v2, version 0.2 with filtering of cisBP motifs) and is also hosted on scMultiome.

peaks	A GRanges object indicating the peaks to perform motif annotation on. The peak indices should match the idxATAC column in the regulon.
version	numeric indicating data version (for details see tfBinding)
peak_number	numeric indicating threshold to be applied to the number of peaks per transcription factor in the combined version of GenomicRanges (from all samples and tissues).

Value

A list of TF binding sites as a GrangesList object.

References

ChIP-Atlas 2021 update: a data-mining suite for exploring epigenomic landscapes by fully integrating ChIP-seq, ATAC-seq and Bisulfite-seq data. Zou Z, Ohta T, Miura F, Oki S. *Nucleic Acids Research. Oxford University Press (OUP);* 2022. doi:10.1093/nar/gkac199

ChIP-Atlas: a data-mining suite powered by full integration of public ChIP-seq data. Oki S, Ohta T, Shioi G, Hatanaka H, Ogasawara O, Okuda Y, Kawaji H, Nakaki R, Sese J, Meno C. *EMBO*; Vol. 19, EMBO reports. 2018. doi:10.15252/embr.201846255

ENCODE: https://www.encodeproject.org/

Examples

```
# retrieve TF binding info

getTFMotifInfo('mm10', 'atlas.sample')
getTFMotifInfo('hg38','atlas.tissue')
getTFMotifInfo('hg19','atlas')


# retrieve motif info
peaks <- GRanges(seqnames=c('chr12','chr19','chr19','chr11','chr6'),
ranges=IRanges(start=c(124914563,50850845, 50850844, 101034172, 151616327),
end=c(124914662,50850929, 50850929, 101034277, 151616394)))
grl <- getTFMotifInfo(genome='hg38', mode='motif', peaks=peaks)
```

---

optimizeMetacellNumber

Determine the optimal number of metacells to be used by calculateP2G *function*

Description

This function attempts to find the optimal value for the cellNum parameter, which is used by calculateP2G to define the number of metacells. The value of this parameter is critical: too many clusters may lead to insufficient signal integration to overcome the effect of data sparsity, whereas too few clusters may result in excessive averaging and loss of important biological variability.

Usage

```
optimizeMetacellNumber(
  peakMatrix,
  expMatrix,
  reducedDim,
  exp_assay,
  peak_assay,
  subsample_prop = 1,
  n_iter = 2,
  cellNumMin = NULL,
  cellNumMax = NULL,
  n_evaluation_points = 5,
  ...
)
```

Arguments

peakMatrix	A SingleCellExperiment object containing counts of chromatin accessibility at each peak region or genomic bin from scATAC-seq. rowRanges should contain genomic positions of the peaks in the form of GRanges.
expMatrix	A SingleCellExperiment object containing gene expression counts from scRNA-seq. rowRanges should contain genomic positions of the genes in the form of GRanges. rowData should contain a column of gene symbols with column name matching the gene_symbol argument.
reducedDim	A matrix of dimension reduced values
exp_assay	String indicating the name of the assay in expMatrix for gene expression
peak_assay	String indicating the name of the assay in peakMatrix for chromatin accessibility
subsample_prop	A numeric indicating the fraction of features from expMatrix and peakMatrix used to optimize kNum
n_iter	An integer indicating the number of iterations before in which the value of kNum parameter is optimized
cellNumMin	A numeric used to optimize value of cellNum parameter. Corresponds to the lower bound for the average number of cells per K-mean cluster in the first iteration of the optimization algorithm. If cellNum is not NULL this parameter is ignored.
cellNumMax	A numeric used to optimize value of cellNum parameter. Corresponds to the upper bound for the average number of cells per K-mean cluster in the first iteration of the optimization algorithm. If cellNum is not NULL this parameter is ignored.
n_evaluation_points	An integer defining how many metacells numbers are tested in the first iteration to find the optimal one. Must not be less than 3. If n_inter > 1 new evaluation points (metacell numbers) are added in the proximity of the current solution.
...	Other arguments passed to calculateP2G function

Value

An object of the class CellNumSol to be passed to calculateP2G as cellNum paramater.

pruneRegulon *Prune regulons for true transcription factor - regulatory elements - target genes relationships*

Description

Prune regulons for true transcription factor - regulatory elements - target genes relationships

Usage

```
pruneRegulon(
  regulon,
  expMatrix = NULL,
  peakMatrix = NULL,
  exp_assay = "logcounts",
  peak_assay = "PeakMatrix",
  test = c("chi.sq", "binom"),
  clusters = NULL,
  exp_cutoff = 1,
  peak_cutoff = 0,
  regulon_cutoff = 0.05,
  p_adj = TRUE,
  prune_value = "pval",
  aggregateCells = FALSE,
  useDim = "IterativeLSI_ATAC",
  cellNum = 10,
  BPPARAM = BiocParallel::SerialParam(progressbar = TRUE)
)
```

Arguments

regulon	A dataframe informing the gene regulatory relationship with the `tf` column representing transcription factors, `idxATAC` corresponding to the index in the peakMatrix and `target` column representing target genes
expMatrix	A SingleCellExperiment object or matrix containing gene expression with genes in the rows and cells in the columns
peakMatrix	A SingleCellExperiment object or matrix containing peak accessibility with peaks in the rows and cells in the columns
exp_assay	String indicating the name of the assay in expMatrix for gene expression
peak_assay	String indicating the name of the assay in peakMatrix for chromatin accessibility
test	String indicating whether `binom` or `chi.sq` test should be performed
clusters	A vector corresponding to the cluster labels of the cells if cluster-specific joint probabilities are also required. If left `NULL`, joint probabilities are calculated for all cells
exp_cutoff	A scalar indicating the minimum gene expression above which gene is considered active. Default value is 1. Applied to both transcription factors and target genes.
peak_cutoff	A scalar indicating the minimum peak accessibility above which peak is considered open. Default value is 0

regulon_cutoff	A scalar indicating the maximal value for p-value for a tf-idxATAC-target trio to be retained in the pruned regulon.
p_adj	A logical indicating whether p adjustment should be performed
prune_value	String indicating whether to filter regulon based on pval or padj.
aggregateCells	A logical to indicate whether to aggregate cells into groups determined by cellNum. Use option to overcome data sparsity if needed
useDim	String indicating the name of the dimensionality reduction matrix in expMatrix used for cell aggregation
cellNum	An integer specifying the number of cells per cluster for cell aggregation. Default is 10.
BPPARAM	A BiocParallelParam object specifying whether calculation should be parallelized. Default is set to BiocParallel::MulticoreParam()

Details

The function prunes the network by performing tests of independence on the observed number of cells jointly expressing transcription factor (TF), regulatory element (RE) and target gene (TG) vs the expected number of cells if TF/RE and TG are independently expressed.

In other words, if no regulatory relationship exists, the expected probability of cells expressing all three elements is P(TF, RE) * P(TG), that is, the product of (1) proportion of cells both expressing transcription factor and having accessible corresponding regulatory element, and (2) proportion of cells expressing target gene. The expected number of cells expressing all three elements is therefore n*P(TF, RE)*P(TG), where n is the total number of cells. However, if a TF-RE-TG relationship exists, we expect the observed number of cells jointly having all three elements (TF, RE, TG) to deviate from the expected number of cells predicted from an independent relationship.

If the user provides cluster assignment, the tests of independence are performed on a per-cluster basis in addition to providing all cells statistics. This enables pruning by cluster, and thus yields cluster-specific gene regulatory relationships.

We implement two tests, the binomial test and the chi-square test.

In the binomial test, the expected probability is P(TF, RE) * P(TG), and the number of trials is the number of cells, and the observed successes is the number of cells jointly expressing all three elements.

In the chi-square test, the expected probability for having all 3 elements active is also P(TF, RE) * P(TG) and the probability otherwise is 1- P(TF, RE) * P(TG). The observed cell count for the active category is the number of cells jointly expressing all three elements, and the cell count for the inactive category is n - n_triple.

Value

A DataFrame of pruned regulons with p-values indicating the probability of independence either for all cells or for individual clusters, z-score statistics for binomial tests or chi-square statistics for chi-square test and q-adjusted values.

Author(s)

Xiaosai Yao, Tomasz Wlodarczyk

Examples

```
# create a mock SingleCellExperiment object for gene expMatrixession matrix
set.seed(1000)
gene_sce <- scuttle::mockSCE()
gene_sce <- scrapper::normalizeRnaCounts.se(gene_sce)
rownames(gene_sce) <- paste0('Gene_',1:2000)

# create a mock SingleCellExperiment object for peak matrix
peak_gr <- GRanges(seqnames='chr1',
    ranges=IRanges(start=seq(from=1, to=10000, by=100), width=100))
peak_counts <- matrix(sample(x=0:4, size=ncol(gene_sce)*length(peak_gr), replace=TRUE),
    nrow=length(peak_gr), ncol=ncol(gene_sce))
peak_sce <- SingleCellExperiment(list(counts=peak_counts), colData=colData(gene_sce))
    rownames(peak_sce) <- paste0('Peak_',1:100)

# create a mock regulon
regulon <- data.frame(tf=c(rep('Gene_1',10), rep('Gene_2',10)),
    idxATAC=sample(1:100, 20),
    target=c(paste0('Gene_', sample(3:2000,10)),
    paste0('Gene_',sample(3:2000,10))))

# prune regulon
pruned.regulon <- pruneRegulon(expMatrix=gene_sce,
    exp_assay='logcounts', peakMatrix=peak_sce, peak_assay='counts',
    regulon=regulon, clusters=gene_sce$Treatment, regulon_cutoff=0.5)

# add weights with cell aggregation
gene_sce <- scrapper::runPca.se(gene_sce, features=NULL)
pruned.regulon <- pruneRegulon(expMatrix=gene_sce, exp_assay='logcounts',
    peakMatrix=peak_sce, peak_assay='counts', regulon=regulon,
    clusters=gene_sce$Treatment, regulon_cutoff=0.5,
    aggregateCells=TRUE, cellNum=3, useDim='PCA')
```

Index

∗ **internal**
epiregulon-package, 2

addLogFC, 3
addMotifScore, 4
addTFMotifInfo, 5
addWeights, 7
aggregateAcrossCellsFast, 9

calculateActivity, 11
calculateP2G, 13

epiregulon *(*epiregulon-package*)*, 2
epiregulon-package, 2

findMarkersSimple, 16

getRegulon, 17
getTFMotifInfo, 18

optimizeMetacellNumber, 19

pruneRegulon, *6*, 21

scMultiome::tfBinding(), *18*

tfBinding, *19*