Package 'SCORPION'

September 21, 2026

Type Package

Title Single Cell Oriented Reconstruction of PANDA Individually Optimized Networks

Version 1.3.4

Description Constructs cell-type-specific gene regulatory networks from single-cell RNA-sequencing data. The method implements the SCORPION algorithm, which first aggregates individual cells into super-cells and then applies PANDA (Passing Attributes between Networks for Data Assimilation) to infer transcription factor-target regulatory relationships. It also provides statistical methods for differential edge analysis.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports cli, methods, irlba, igraph, RANN, Matrix, pbapply, dplyr, furrr, future

Suggests RhpcBLASctl, testthat, mori, circlize, biomaRt, fgsea

URL https://github.com/kuijjerlab/SCORPION

BugReports https://github.com/kuijjerlab/SCORPION/issues

RoxygenNote 7.3.3

NeedsCompilation no

Author Daniel Osorio [aut, cre] (ORCID: <https://orcid.org/0000-0003-4424-8422>), Marieke L. Kuijjer [aut] (ORCID: <https://orcid.org/0000-0001-6280-3130>)

Maintainer Daniel Osorio <daniecos@uio.no>

Repository CRAN

Date/Publication 2026-09-21 16:40:02 UTC

Contents

`circosEdges` *Circos plot of differential network edges*

Description

Draws a circular (Circos) plot of transcription factor to target links from a `testEdges` two-sample result. Genes are placed on their genomic coordinates, links are coloured continuously by significance, flagged as known or novel against an optional a priori network, and genes belonging to supplied gene sets can be labelled around the circle.

Usage

```
circosEdges(
  edgesDF,
  species = "hsapiens_gene_ensembl",
  geneCoords = NULL,
  priorNet = NULL,
  geneSets = NULL,
  pAdjThreshold = 0.05,
  log2FCThreshold = 0,
  maxEdges = 500L,
  colorBy = "log2FoldChange",
  linkColors = c("#2166AC", "#F7F7F7", "#B2182B"),
  lwdRange = c(0.5, 4),
  nmaxTF = 20L,
  nmaxTarget = 20L,
  knownColor = "grey60",
  novelColor = "#D95F02",
  geneSetColors = NULL,
  chromosomes = NULL,
  mainChromosomesOnly = TRUE,
  ensemblMirror = "www",
  transparency = 0.5,
  hRatio = 0.6,
  fontFamily = "sans",
```

```
  legend = TRUE
)
```

Arguments

edgesDF	A data.frame produced by testEdges (two-sample or paired). Must contain the columns tf, target, log2FoldChange and pAdj (pValue is used as a fallback when pAdj is absent).
species	Ensembl dataset name passed to **biomaRt** when geneCoords is NULL, e.g. "hsapiens_gene_ensembl" for human or "mmusculus_gene_ensembl" for mouse. See biomaRt::listDatasets() for the full multi-species list.
geneCoords	Optional data.frame supplying gene coordinates from any source (overrides the **biomaRt** download). Must have columns gene, chr, start and end.
priorNet	Optional a priori TF-target network whose first two columns are the TF and target. Links present here are labelled "known", all others "novel". Accepts a data.frame or a matrix.
geneSets	Optional gene-set annotation used to label genes around the circle: either a path to a GMT file or a named list of character vectors.
pAdjThreshold	Numeric significance cutoff applied to pAdj (or pValue when pAdj is missing). Default 0.05.
log2FCThreshold	Numeric minimum absolute log2FoldChange required to draw a link. Default 0.
maxEdges	Integer cap on the number of links drawn; when exceeded, the most significant edges are kept. Default 500.
colorBy	Name of the edgesDF column mapped to the link colour ramp. Default "log2FoldChange", giving a continuous diverging colour scale centred at zero. When the default is used but log2FoldChange is absent (e.g. single-sample testEdges output), it falls back to meanEdge.
linkColors	Length-3 vector of colours for the low, mid and high ends of colorBy. Default blue-white-red; a diverging ramp is used when colorBy has negative values, otherwise a sequential low-to-high ramp.
lwdRange	Length-2 numeric giving the minimum and maximum link line width; each link's thickness is scaled linearly within this range by its -log10 adjusted p-value. Default c(0.5, 4).
nmaxTF, nmaxTarget	Integers giving how many TFs and targets to label, selected by the largest absolute out-degree and in-degree respectively. Use NULL or Inf to label all. Defaults 20.
knownColor, novelColor	Border colours distinguishing known from novel links. Defaults grey and orange.
geneSetColors	Optional named vector mapping gene-set names to colours. When NULL, colours are generated automatically.
chromosomes	Optional character vector restricting and ordering the chromosomes shown. When NULL, all chromosomes present are used.

`mainChromosomesOnly`
: Logical; when `TRUE` (the default) and `chromosomes` is `NULL`, only the main chromosomes (numbered, plus X, Y and MT) are kept and unplaced scaffolds/contigs are dropped.

`ensemblMirror`
: **biomaRt** mirror to query: one of `"www"`, `"useast"` or `"asia"`. Default `"www"`.

`transparency`
: Numeric link transparency in `[0, 1]` (0 is opaque). Default `0.5`.

`hRatio`
: Numeric in `[0, 1]` controlling how far link ribbons bend toward the circle centre; smaller values give flatter, less tangled links. Default `0.6`.

`fontFamily`
: Font family used for all plot text, e.g. `"sans"` (Helvetica/Arial, the default) for a publication look.

`legend`
: Logical; whether to draw legends for effect size, novelty and gene sets. Default `TRUE`.

Details

Requires the **circlize** package, and **biomaRt** when gene coordinates are downloaded automatically (`geneCoords = NULL`). Genes without coordinates, and links whose TF or target lacks coordinates, are dropped with a message.

Value

Invisibly, a list with `edges` (the plotted links annotated with coordinates and novelty) and `coords` (the gene coordinate table with `outDegree`, `inDegree` and total `degree` columns, each the sum of `log2FoldChange` over a gene's outgoing / incoming links). The function is called for the side effect of drawing the plot.

Author(s)

Daniel Osorio <daniecos@uio.no>

See Also

`testEdges`, `runSCORPION`

Examples

```
## Not run:
data(scorpionTest)
nets <- runSCORPION(
  gexMatrix = scorpionTest$gex,
  tfMotifs = scorpionTest$tf,
  ppiNet = scorpionTest$ppi,
  cellsMetadata = scorpionTest$metadata,
  groupBy = c("donor", "region")
)
res <- testEdges(
  networksDF = nets,
  testType = "two.sample",
  group1 = grep("--T$", colnames(nets), value = TRUE),
```

```
  group2 = grep("--N$", colnames(nets), value = TRUE)
)

# Human coordinates auto-downloaded from Ensembl, known/novel vs a prior net
circosEdges(
  edgesDF = res,
  species = "hsapiens_gene_ensembl",
  priorNet = scorpionTest$tf,
  geneSets = "hallmark.gmt"
)

## End(Not run)
```

`enrichEdges` *Gene set enrichment analysis of TF-target edges*

Description

Performs gene set enrichment analysis separately for each transcription factor (TF) using the edge-level values supplied in `numericValue`. Enrichment is performed with the multilevel implementation of **fgsea**. Calculations for individual TFs are performed in parallel.

Usage

```
enrichEdges(edgesDF, geneSets, numericValue, nCores = 3, seed = 1)
```

Arguments

`edgesDF`	A data.frame of TF-target edges, typically produced by `testEdges`. Must contain a `tf` column, a `target` column, and the numeric column named by `numericValue`.
`geneSets`	A named list of gene sets. The names of the list elements are used as gene set identifiers.
`numericValue`	Character string naming the column in `edgesDF` used as the ranking statistic for enrichment analysis.
`nCores`	Integer specifying the number of parallel workers to use. Default 3.
`seed`	Integer specifying the random seed used by the parallel enrichment calculations. Default 1.

Details

For each TF, the values in `numericValue` are used as ranked statistics for its target genes. Edges with missing targets or missing or non-finite values in the selected numeric column are excluded before enrichment analysis. If a target occurs more than once for a TF, only the observation with the largest absolute value of the selected ranking statistic is retained.

Gene set enrichment is performed using `fgsea::fgseaMultilevel`; the **fgsea** package (Bioconductor) is required. The `leadingEdge` column returned by `fgseaMultilevel` is not included in the output. P-values are adjusted across all TF-gene set enrichment tests using the Benjamini-Hochberg procedure.

Value

A data.frame of enrichment results with one row per TF-gene set pair:

- tf: Transcription factor
- geneSet: Gene set identifier
- pValue: Raw enrichment p-value
- pAdj: Benjamini-Hochberg adjusted p-value
- log2Err: Expected log2 error of the p-value estimate
- ES: Enrichment score
- NES: Normalized enrichment score
- geneSetSize: Number of genes from the set found among the targets

Author(s)

Daniel Osorio <daniecos@uio.no>

See Also

testEdges, maEdges

Examples

```
## Not run:
data(scorpionTest)
nets <- runSCORPION(
  gexMatrix = scorpionTest$gex,
  tfMotifs = scorpionTest$tf,
  ppiNet = scorpionTest$ppi,
  cellsMetadata = scorpionTest$metadata,
  groupBy = c("donor", "region")
)
res <- testEdges(
  networksDF = nets,
  testType = "two.sample",
  group1 = grep("--T$", colnames(nets), value = TRUE),
  group2 = grep("--N$", colnames(nets), value = TRUE)
)

geneSets <- list(SetA = c("ACKR1", "ACTA2"), SetB = c("ACTG2", "ADAMDEC1"))
enr <- enrichEdges(
  edgesDF = res,
  geneSets = geneSets,
  numericValue = "log2FoldChange"
)

## End(Not run)
```

maEdges *Meta-analysis of TF-target edges across studies*

Description

Performs a meta-analysis of TF-target edges across multiple studies using either a fixed-effect or DerSimonian-Laird random-effects model. Missing or non-finite effect sizes and standard errors are excluded from the corresponding study. A TF-target pair is only counted as contributing to a study when both its effect size and SE are valid.

Usage

```
maEdges(
  edgesList,
  method = c("random", "fixed"),
  minStudies = 2L,
  padjustMethod = "BH",
  moderateVariance = TRUE,
  s0 = NULL
)
```

Arguments

edgesList	A list of data.frames, one per study, typically produced by testEdges. Each data.frame must contain the columns tf, target, log2FoldChange and SE.
method	Meta-analysis model. Either "random" (DerSimonian-Laird random-effects) or "fixed" (inverse-variance fixed-effect). Default is "random".
minStudies	Minimum number of studies with valid numeric information required for a TF-target pair to be included. Default 2.
padjustMethod	Character specifying the p-value adjustment method for multiple testing correction. See p.adjust for options. Default "BH" (Benjamini-Hochberg FDR).
moderateVariance	Logical indicating whether to apply SAM-style variance moderation to the meta-analysis SE. Default TRUE.
s0	Optional variance-moderation fudge factor. If NULL and moderateVariance = TRUE, the median of all valid meta-analysis SEs is used.

Value

A data.frame containing:

- tf: Transcription factor
- target: Target gene
- k: Number of studies contributing to the meta-analysis
- log2FoldChange: Meta-analytic effect size

- SE: Meta-analysis standard error
- ciLow: Lower bound of the 95% confidence interval
- ciHigh: Upper bound of the 95% confidence interval
- zStatistic: Test statistic
- pValue: Raw p-value
- pAdj: Adjusted p-value
- Q: Cochran's Q heterogeneity statistic
- iSquared: I-squared heterogeneity (percentage)
- tauSquared: DerSimonian-Laird between-study variance

Author(s)

Daniel Osorio <daniecos@uio.no>

See Also

runSCORPION, testEdges, circosEdges

`regressEdges` *Regression analysis of edges across ordered conditions*

Description

Performs linear regression on network edges from runSCORPION output to identify edges that show significant trends across ordered conditions (e.g., disease progression: Normal -> Border -> Tumor).

Usage

```
regressEdges(networksDF, orderedGroups, padjustMethod = "BH", minMeanEdge = 0)
```

Arguments

`networksDF`	A data.frame output from runSCORPION containing TF-target pairs as rows and network identifiers as columns.
`orderedGroups`	A named list where each element is a character vector of column names in networksDF. Names represent ordered conditions (e.g., list(Normal = c("P31–N", "P32–N"), Border = c("P31–B", "P32–B"), Tumor = c("P31–T", "P32–T"))). The order of list elements defines the progression (first to last).
`padjustMethod`	Character specifying the p-value adjustment method for multiple testing correction. See p.adjust for options. Default "BH" (Benjamini-Hochberg FDR).
`minMeanEdge`	Numeric threshold for minimum mean absolute edge weight to include in testing. Edges with mean absolute weight below this threshold are excluded. Default 0 (no filtering).

Details

This function performs simple linear regression for each edge, modeling edge weight as a function of an ordered categorical variable (coded as 0, 1, 2, ... for each condition level).

The slope coefficient indicates the average change in edge weight per step along the ordered progression. Positive slopes indicate increasing edge weights, negative slopes indicate decreasing edge weights.

The function uses vectorized computations for efficiency with large datasets.

Value

A data.frame containing:

- tf: Transcription factor
- target: Target gene
- slope: Regression slope (change in edge weight per condition step)
- intercept: Regression intercept
- rSquared: R-squared value (proportion of variance explained)
- fStatistic: F-statistic for the regression
- pValue: Raw p-value for the slope
- pAdj: Adjusted p-value
- meanEdge: Overall mean edge weight across all conditions
- One column per condition showing mean edge weight in that condition

Author(s)

Daniel Osorio <daniecos@uio.no>

See Also

runSCORPION, testEdges

Examples

```
## Not run:
# Load test data and build networks by donor and region
# Note: T = Tumor, N = Normal, B = Border regions
data(scorpionTest)
nets <- runSCORPION(
  gexMatrix = scorpionTest$gex,
  tfMotifs = scorpionTest$tf,
  ppiNet = scorpionTest$ppi,
  cellsMetadata = scorpionTest$metadata,
  groupBy = c("donor", "region")
)

# Define ordered progression: Normal -> Border -> Tumor
normal_nets <- grep("--N$", colnames(nets), value = TRUE)
```

```
border_nets <- grep("--B$", colnames(nets), value = TRUE)
tumor_nets <- grep("--T$", colnames(nets), value = TRUE)

ordered_conditions <- list(
  Normal = normal_nets,
  Border = border_nets,
  Tumor = tumor_nets
)

# Perform regression analysis
results_regression <- regressEdges(
  networksDF = nets,
  orderedGroups = ordered_conditions
)

# View top edges with strongest trends
head(results_regression[order(results_regression$pAdj), ])

# Edges with positive slopes (increasing from N to T)
increasing <- results_regression[results_regression$pAdj < 0.05 &
                                 results_regression$slope > 0, ]
print(paste("Edges increasing along N->B->T:", nrow(increasing)))

# Edges with negative slopes (decreasing from N to T)
decreasing <- results_regression[results_regression$pAdj < 0.05 &
                                 results_regression$slope < 0, ]
print(paste("Edges decreasing along N->B->T:", nrow(decreasing)))

# Filter by minimum edge weight and R-squared
strong_trends <- results_regression[results_regression$pAdj < 0.05 &
                                    results_regression$rSquared > 0.7 &
                                    abs(results_regression$meanEdge) > 0.1, ]

## End(Not run)
```

`runSCORPION` *Run SCORPION across cell groups and return combined networks*

Description

Builds per-group regulatory networks by running `scorpion` on subsets of cells defined by `cellsMetadata` and combining the resulting networks into a wide-format data frame where each column corresponds to a network.

Usage

```
runSCORPION(
  gexMatrix,
  tfMotifs,
  ppiNet,
```

```
  cellsMetadata,
  groupBy,
  normalizeData = TRUE,
  removeBatchEffect = FALSE,
  batch = NULL,
  minCells = 30,
  computingEngine = "cpu",
  nCores = 1,
  gammaValue = 10,
  nPC = 25,
  assocMethod = "pearson",
  alphaValue = 0.1,
  hammingValue = 0.001,
  nIter = Inf,
  outNet = "regNet",
  zScaling = TRUE,
  showProgress = TRUE,
  randomizationMethod = "None",
  scaleByPresent = FALSE,
  filterExpr = FALSE
)
```

Arguments

`gexMatrix`	An expression dataset with genes in the rows and barcodes (cells) in the columns.
`tfMotifs`	A motif dataset, a data.frame or a matrix containing 3 columns. Each row describes a motif associated with a transcription factor (column 1) a gene (column 2) and a score (column 3).
`ppiNet`	A Protein-Protein-Interaction dataset, a data.frame or matrix containing 3 columns. Each row describes a protein-protein interaction between transcription factor 1 (column 1), transcription factor 2 (column 2) and a score (column 3).
`cellsMetadata`	A data.frame with cell-level metadata; must contain columns specified in `groupBy`.
`groupBy`	Character vector of one or more column names in `cellsMetadata` to use for grouping cells into networks.
`normalizeData`	Boolean to indicate normalization of expression data. Default TRUE performs log normalization.
`removeBatchEffect`	Boolean to indicate batch effect correction. Default FALSE.
`batch`	Factor or vector giving batch assignment for each cell; required if `removeBatchEffect = TRUE`.
`minCells`	Minimum number of cells per group required to build a network. Default is 30.
`computingEngine`	Either 'cpu' or 'gpu'. Passed to `scorpion`.
`nCores`	Number of processors to be used if BLAS or MPI is active.
`gammaValue`	Graining level of data (proportion of number of single cells to super-cells). Default 10.

nPC	Number of principal components to use for kNN network construction. Default 25.
assocMethod	Association method. Must be one of 'pearson', 'spearman' or 'pcNet'. Default 'pearson'.
alphaValue	Value to be used for update variable in PANDA. Default 0.1.
hammingValue	Value at which to terminate the process based on Hamming distance. Default 0.001.
nIter	Sets the maximum number of iterations PANDA can run before exiting. Default Inf.
outNet	Character vector specifying which network(s) to extract. Options include "regNet", "coregNet", "coopNet". Default "regNet". When more than one network is requested, an edge_type column ("tf-target", "gene-gene", "tf-tf") is added and the networks are stacked in long format.
zScaling	Boolean to indicate use of Z-Scores in output. FALSE will use [0,1] scale. Default TRUE.
showProgress	Boolean to indicate printing of output for algorithm progress. Default TRUE.
randomizationMethod	Method by which to randomize gene expression matrix. Default "None". Must be one of "None", "within.gene", "by.gene".
scaleByPresent	Boolean to indicate scaling of correlations by percentage of positive samples. Default FALSE.
filterExpr	Boolean to indicate whether or not to remove genes with 0 expression across all cells. Default FALSE.

Details

This function is a wrapper around scorpion that groups cells according to metadata columns, filters out groups with insufficient cells, runs network inference on each remaining group independently, and finally combines all resulting networks into a single wide-format data frame.

Value

A data.frame in wide format where rows represent TF-target pairs (union across all networks) and columns represent network identifiers. Cell values are edge weights from the corresponding network. When multiple network types are requested via outNet, an additional leading edge_type column identifies the network each row comes from and the network types are stacked in long format.

Author(s)

Daniel Osorio <daniecos@uio.no>

See Also

scorpion, testEdges, regressEdges

Examples

```
## Not run:
# Load test data
data(scorpionTest)

# Example 1: Group by single column (region)
nets_by_region <- runSCORPION(
  gexMatrix = scorpionTest$gex,
  tfMotifs = scorpionTest$tf,
  ppiNet = scorpionTest$ppi,
  cellsMetadata = scorpionTest$metadata,
  groupBy = "region"
)

# -- SCORPION --------------------------------------------------------------
# + Normalizing data (log scale)
# i 3 networks requested
# + 3 networks meet the minimum cell requirement (30)
# i Computing networks
# + Networks successfully constructed
# + Networks successfully combined

# head(nets_by_region)
#                            tf target           T          B           N
# 1                        AATF  ACKR1 -0.31433856 -0.3569918 -0.33734920
# 2                        ABL1  ACKR1 -0.32915008 -0.3648895 -0.34437341
# 3                       ACSS2  ACKR1 -0.31418599 -0.3557854 -0.33663144
# 4                        ADNP  ACKR1  0.04105895  0.1109288  0.09910822
# 5                       AEBP2  ACKR1 -0.18964574 -0.2202269 -0.17558140
# 6 AEBP2_EED_EZH2_RBBP4_SUZ12  ACKR1 -0.31024700 -0.3508320 -0.33054519

# Example 2: Group by single column (donor)
nets_by_donor <- runSCORPION(
  gexMatrix = scorpionTest$gex,
  tfMotifs = scorpionTest$tf,
  ppiNet = scorpionTest$ppi,
  cellsMetadata = scorpionTest$metadata,
  groupBy = "donor"
)

# -- SCORPION --------------------------------------------------------------
# + Normalizing data (log scale)
# i 3 networks requested
# + 3 networks meet the minimum cell requirement (30)
# i Computing networks
# + Networks successfully constructed
# + Networks successfully combined
# head(nets_by_donor)
#                            tf target         P31        P32         P33
# 1                        AATF  ACKR1 -0.34869366 -0.3557884 -0.35010835
# 2                        ABL1  ACKR1 -0.33724323 -0.3575331 -0.32875974
# 3                       ACSS2  ACKR1 -0.34569954 -0.3573108 -0.34980657
```

```
# 4                       ADNP  ACKR1  0.09933951  0.1045316  0.06046914
# 5                      AEBP2  ACKR1 -0.25111137 -0.2245655 -0.23157035
# 6 AEBP2_EED_EZH2_RBBP4_SUZ12  ACKR1 -0.34148264 -0.3518686 -0.34398594

# Example 3: Group by two columns (donor and region)
nets_by_donor_region <- runSCORPION(
  gexMatrix = scorpionTest$gex,
  tfMotifs = scorpionTest$tf,
  ppiNet = scorpionTest$ppi,
  cellsMetadata = scorpionTest$metadata,
  groupBy = c("donor", "region")
)

# -- SCORPION --------------------------------------------------------------
# + Normalizing data (log scale)
# i 9 networks requested
# + 9 networks meet the minimum cell requirement (30)
# i Computing networks
# + Networks successfully constructed
# + Networks successfully combined
# head(nets_by_donor_region)
#                           tf target      P31--T      P31--B     P31--N
# 1                       AATF  ACKR1 -0.32634975 -0.33717677 -0.3442886
# 2                       ABL1  ACKR1 -0.34048759 -0.33890429 -0.3509986
# 3                      ACSS2  ACKR1 -0.32570697 -0.33600811 -0.3436603
# 4                       ADNP  ACKR1  0.07975735  0.05354279  0.1048301
# 5                      AEBP2  ACKR1 -0.21472437 -0.20545660 -0.1815737
# 6 AEBP2_EED_EZH2_RBBP4_SUZ12  ACKR1 -0.31861592 -0.32809314 -0.3375652

# Example 4: Group by three columns (donor, region, and cell_type)
nets_by_donor_region_cell_type <- runSCORPION(
  gexMatrix = scorpionTest$gex,
  tfMotifs = scorpionTest$tf,
  ppiNet = scorpionTest$ppi,
  cellsMetadata = scorpionTest$metadata,
  groupBy = c("donor", "region", "cell_type")
)

# -- SCORPION --------------------------------------------------------------
# + Normalizing data (log scale)
# i 9 networks requested
# + 9 networks meet the minimum cell requirement (30)
# i Computing networks
# + Networks successfully constructed
# + Networks successfully combined
# head(nets_by_donor_region_cell_type)
#                           tf target P31--T--Epithelial P31--B--Epithelial
# 1                       AATF  ACKR1        -0.32634975        -0.33717677
# 2                       ABL1  ACKR1        -0.34048759        -0.33890429
# 3                      ACSS2  ACKR1        -0.32570697        -0.33600811
# 4                       ADNP  ACKR1         0.07975735         0.05354279
# 5                      AEBP2  ACKR1        -0.21472437        -0.20545660
# 6 AEBP2_EED_EZH2_RBBP4_SUZ12  ACKR1        -0.31861592        -0.32809314
```

```
# Example 5: Using GPU computing engine (if available)
nets_gpu <- runSCORPION(
  gexMatrix = scorpionTest$gex,
  tfMotifs = scorpionTest$tf,
  ppiNet = scorpionTest$ppi,
  cellsMetadata = scorpionTest$metadata,
  groupBy = "region",
  computingEngine = "gpu"
)

# -- SCORPION ----------------------------------------------------------------
# + Normalizing data (log scale)
# i 3 networks requested
# + 3 networks meet the minimum cell requirement (30)
# i Computing networks
# + Networks successfully constructed
# + Networks successfully combined
# head(nets_gpu)
#                           tf target           T          B           N
# 1                       AATF  ACKR1 -0.31433821 -0.3569913 -0.33734894
# 2                       ABL1  ACKR1 -0.32915005 -0.3648892 -0.34437302
# 3                      ACSS2  ACKR1 -0.31418574 -0.3557851 -0.33663106
# 4                       ADNP  ACKR1  0.04105883  0.1109285  0.09910798
# 5                      AEBP2  ACKR1 -0.18964562 -0.2202267 -0.17558131
# 6 AEBP2_EED_EZH2_RBBP4_SUZ12  ACKR1 -0.31024694 -0.3508317 -0.33054504

# Example 6: Removing batch effect using donor as batch
nets_batch_corrected <- runSCORPION(
  gexMatrix = scorpionTest$gex,
  tfMotifs = scorpionTest$tf,
  ppiNet = scorpionTest$ppi,
  cellsMetadata = scorpionTest$metadata,
  groupBy = "region",
  removeBatchEffect = TRUE,
  batch = scorpionTest$metadata$donor
)

# -- SCORPION ----------------------------------------------------------------
# + Normalizing data (log scale)
# + Correcting for batch effects
# i 3 networks requested
# + 3 networks meet the minimum cell requirement (30)
# i Computing networks
# + Networks successfully constructed
# + Networks successfully combined
# head(nets_batch_corrected)
#                           tf target          T          B           N
# 1                       AATF  ACKR1 -0.3337298 -0.3488547 -0.13011777
# 2                       ABL1  ACKR1 -0.3408020 -0.3540981 -0.17694266
# 3                      ACSS2  ACKR1 -0.3325270 -0.3511531 -0.12661518
# 4                       ADNP  ACKR1  0.1117504  0.0869148  0.01608898
# 5                      AEBP2  ACKR1 -0.2334648 -0.2211301  0.12519312
```

```
# 6 AEBP2_EED_EZH2_RBBP4_SUZ12  ACKR1 -0.3274770 -0.34475499 -0.12449908

## End(Not run)
```

`scorpion` *Build gene regulatory networks from single-cell RNA-seq data using PANDA*

Description

Constructs gene regulatory networks from single-cell/nuclei RNA-seq data by first applying coarse-graining to reduce sparsity, then running the PANDA (Passing Attributes between Networks for Data Assimilation) message-passing algorithm to integrate transcription factor motifs, protein-protein interactions, and gene expression data into unified regulatory networks.

Usage

```
scorpion(
  tfMotifs = NULL,
  gexMatrix,
  ppiNet = NULL,
  computingEngine = "cpu",
  nCores = 1,
  gammaValue = 10,
  nPC = 25,
  assocMethod = "pearson",
  alphaValue = 0.1,
  hammingValue = 0.001,
  nIter = Inf,
  outNet = c("regNet", "coregNet", "coopNet"),
  zScaling = TRUE,
  showProgress = TRUE,
  randomizationMethod = "None",
  scaleByPresent = FALSE,
  filterExpr = FALSE
)
```

Arguments

`tfMotifs`	A motif dataset (data.frame or matrix) with 3 columns: TF, target gene, and motif score. Pass NULL for co-expression analysis only.
`gexMatrix`	An expression dataset, with genes in the rows and barcodes (cells) in the columns.
`ppiNet`	A Protein-Protein-Interaction dataset (data.frame or matrix) with 3 columns: protein 1, protein 2, and interaction score. Pass NULL to disable protein interaction integration.

`computingEngine`
: Character specifying computing device: 'cpu' or 'gpu' (if available). Default 'cpu'.

`nCores`
: Number of processors to be used if BLAS or MPI is active.

`gammaValue`
: Graining level of data (proportion of number of single cells in the initial dataset to the number of super-cells in the final dataset)

`nPC`
: Number of principal components to use for construction of single-cell kNN network.

`assocMethod`
: Association method. Must be one of 'pearson', 'spearman' or 'pcNet'.

`alphaValue`
: Numeric update parameter (0 to 1) controlling relative contribution of prior networks. Default 0.1.

`hammingValue`
: Numeric convergence threshold based on Hamming distance. Algorithm stops when updates fall below this. Default 0.001.

`nIter`
: Sets the maximum number of iterations PANDA can run before exiting.

`outNet`
: A vector containing which networks to return. Options include "regNet", "coregNet", "coopNet".

`zScaling`
: Boolean to indicate use of Z-Scores in output. FALSE will use [0,1] scale.

`showProgress`
: Boolean to indicate printing of output for algorithm progress.

`randomizationMethod`
: Method by which to randomize gene expression matrix. Default "None". Must be one of "None", "within.gene", "by.genes". "within.gene" randomization scrambles each row of the gene expression matrix, "by.gene" scrambles gene labels.

`scaleByPresent`
: Boolean to indicate scaling of correlations by percentage of positive samples.

`filterExpr`
: Boolean to remove genes with zero expression across all cells before network inference. Default FALSE.

Value

A list of 6 elements describing the inferred networks at convergence:

- regNet: Regulatory network matrix (TFs × genes)
- coregNet: Co-regulation network matrix (genes × genes)
- coopNet: Cooperation network matrix (TFs × TFs)
- numGenes: Number of genes in the network
- numTFs: Number of transcription factors
- numEdges: Total number of edges in regulatory network

Author(s)

Daniel Osorio <daniecos@uio.no>

See Also

`runSCORPION` for building networks across cell groups.

Examples

```
# Loading example data
data(scorpionTest)

# The structure of the data
str(scorpionTest)

# List of 4
# $ gex     :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
# .. ..@ i       : int [1:46171] 29 32 41 43 61 170 208 245 251 269 ...
# .. ..@ p       : int [1:1955] 0 11 62 97 112 163 184 215 257 274 ...
# .. ..@ Dim     : int [1:2] 300 1954
# .. ..@ Dimnames:List of 2
# .. .. ..$ : chr [1:300] "IGHM" "IGHG2" "IGLC3" "IGLL5" ...
# .. .. ..$ : chr [1:1954] "P31-T_AAACGGGTCGGTTAAC" "P31-T_AAAGATGGTGGCCCTA" ...
# .. ..@ x       : num [1:46171] 1 1 1 1 2 2 1 1 2 1 ...
# .. ..@ factors : list()
# $ tf      :'data.frame': 371738 obs. of  3 variables:
# ..$ source_genesymbol: chr [1:371738] "MYC" "SPI1" "JUN_JUND" "FOS_JUND" ...
# ..$ target_genesymbol: chr [1:371738] "TERT" "BGLAP" "JUN" "JUN" ...
# ..$ weight           : num [1:371738] 1 1 1 1 1 1 1 1 1 1 ...
# ..- attr(*, "origin")= chr "cache"
# ..- attr(*, "url")= chr "https://omnipathdb.org/interactions? __truncated__
# $ ppi     :'data.frame': 4076 obs. of  3 variables:
# ..$ source_genesymbol: chr [1:4076] "ZIC1" "HES5" "ATOH1" "DLL1" ...
# ..$ target_genesymbol: chr [1:4076] "ATOH1" "ATOH1" "HES5" "NOTCH1" ...
# ..$ weight           : num [1:4076] 1 1 1 1 1 1 1 1 1 1 ...
# ..- attr(*, "origin")= chr "cache"
# ..- attr(*, "url")= chr "https://omnipathdb.org/interactions?__truncated__
# $ metadata:'data.frame': 1954 obs. of  4 variables:
# ..$ cell_id  : chr [1:1954] "P31-T_AAACGGGTCGGTTAAC" "P31-T_AAAGATGGTGGCCCTA"...
# ..$ donor    : chr [1:1954] "P31" "P31" "P31" "P31" ...
# ..$ region   : chr [1:1954] "T" "T" "T" "T" ...
# ..$ cell_type: Factor w/ 1 level "Epithelial": 1 1 1 1 1 1 1 1 1 1 ...

# Running SCORPION for epithelial cells from the normal tissue
# We are using alphaValue = 0.8 for testing purposes (Default = 0.1).
scorpionOutput <- scorpion(
  tfMotifs = scorpionTest$tf,
  gexMatrix = scorpionTest$gex[, scorpionTest$metadata$region == "N"],
  ppiNet = scorpionTest$ppi,
  alphaValue = 0.8
)

# -- SCORPION ----------------------------------------------------------------------------------
# + Initializing and validating
# + Verified sufficient samples
# i Normalizing networks
# i Learning Network
# i Using tanimoto similarity
# + Successfully ran SCORPION on 281 Genes and 963 TFs
```

```
# Structure of the output.
str(scorpionOutput)

# List of 6
# $ regNet  : num [1:963, 1:281] -0.1556 -0.0455 -0.1461 1.6881 0.8746 ...
# ..- attr(*, "dimnames")=List of 2
# .. ..$ : chr [1:963] "AATF" "ABL1" "ACSS2" "ADNP" ...
# .. ..$ : chr [1:281] "ACKR1" "ACTA2" "ACTG2" "ADAMDEC1" ...
# $ coregNet: num [1:281, 1:281] 2.02e+06 3.84 4.10 -1.26 8.81e-01 ...
# ..- attr(*, "dimnames")=List of 2
# .. ..$ : chr [1:281] "ACKR1" "ACTA2" "ACTG2" "ADAMDEC1" ...
# .. ..$ : chr [1:281] "ACKR1" "ACTA2" "ACTG2" "ADAMDEC1" ...
# $ coopNet : num [1:963, 1:963] 1.17e+07 -2.66 8.13 -1.31 4.95 ...
# ..- attr(*, "dimnames")=List of 2
# .. ..$ : chr [1:963] "AATF" "ABL1" "ACSS2" "ADNP" ...
# .. ..$ : chr [1:963] "AATF" "ABL1" "ACSS2" "ADNP" ...
# $ numGenes: int 281
# $ numTFs  : int 963
# $ numEdges: int 270603
```

`scorpionTest` *Example single-cell colorectal cancer data for SCORPION*

Description

A list bundling the inputs required to build and compare gene regulatory networks with SCORPION, derived from a colorectal cancer single-cell RNA-sequencing experiment. It contains a gene expression matrix, a transcription factor motif prior, a protein-protein interaction prior, and cell-level metadata.

Usage

```
data(scorpionTest)
```

Format

A named list with four elements:

`gex` A `dgCMatrix` gene expression matrix with 300 genes (rows) and 1,954 cells (columns).

`tf` A `data.frame` of transcription factor-target motif pairs from DoRothEA with columns `source_genesymbol`, `target_genesymbol` and `weight` (371,738 rows).

`ppi` A `data.frame` of protein-protein interactions with columns `source_genesymbol`, `target_genesymbol` and `weight` (4,076 rows).

`metadata` A `data.frame` of cell-level annotations with columns `cell_id`, `donor`, `region` and `cell_type` (1,954 rows). Region codes are `T` (tumor), `B` (border) and `N` (normal).

Examples

```
# Loading example data
data(scorpionTest)

# The structure of the data
str(scorpionTest)
```

`testEdges` *Test edges from SCORPION networks*

Description

Performs statistical testing of network edges from runSCORPION output. Supports single-sample tests (testing if edges differ from zero) and two-sample tests (comparing edges between two groups).

Usage

```
testEdges(
  networksDF,
  testType = c("single", "two.sample"),
  group1,
  group2 = NULL,
  paired = FALSE,
  alternative = c("two.sided", "greater", "less"),
  padjustMethod = "BH",
  minLog2FC = 0,
  moderateVariance = TRUE,
  empiricalNull = TRUE,
  nCores = 1L,
  batchSize = NULL
)
```

Arguments

`networksDF`	A data.frame output from `runSCORPION` containing TF-target pairs as rows and network identifiers as columns.
`testType`	Character specifying the test type. Options are: • "single": Single-sample test (one-sample t-test against zero) • "two.sample": Two-sample comparison (t-test between two groups)
`group1`	Character vector of column names in `networksDF` representing the first group (or the only group for single-sample tests).
`group2`	Character vector of column names in `networksDF` representing the second group. Required for two-sample tests, ignored for single-sample tests.

`paired` Logical indicating whether to perform a paired t-test. Default FALSE. When TRUE, group1 and group2 must have the same length and be in matched order (e.g., group1[1] is paired with group2[1]). Useful for comparing matched samples such as Tumor vs Normal from the same patient.

`alternative` Character specifying the alternative hypothesis. Options: "two.sided" (default), "greater", or "less".

`padjustMethod` Character specifying the p-value adjustment method for multiple testing correction. See `p.adjust` for options. Default "BH" (Benjamini-Hochberg FDR).

`minLog2FC` Numeric threshold for minimum absolute log2 fold change to include in testing. For two-sample and paired tests, edges with |log2FoldChange| below this threshold are excluded. Not applicable for single-sample tests. Default 0.

`moderateVariance` Logical indicating whether to apply SAM-style variance moderation. When TRUE, adds a fudge factor (s0, the median of all standard errors) to the denominator of the t-statistic. This prevents edges with very small variance from producing extreme t-statistics, resulting in volcano plots more similar to limma output. Default TRUE.

`empiricalNull` Logical indicating whether to estimate the null distribution empirically from the observed t-statistics. When TRUE, uses the median and MAD (median absolute deviation) of all t-statistics to recenter and rescale them, then computes p-values from the standard normal. This is Efron's empirical null correction (as in locfdr) and is essential when testing millions of correlated edges. Runs in O(n) time. Default TRUE.

`nCores` Integer specifying the number of parallel workers. Default 1 (sequential processing). When greater than 1, edges are split into batches and processed in parallel using `furrr::future_map_dfr`. Requires the **furrr** and **future** packages to be installed.

`batchSize` Integer specifying the number of edges (rows) per batch for parallel processing. Default NULL, which auto-calculates as `ceiling(nrow(networksDF) / nCores)`. Only used when `nCores > 1`. Smaller batch sizes use less memory per worker but add communication overhead.

Details

For single-sample tests, the function tests whether the mean edge weight across replicates significantly differs from zero using a one-sample t-test.

For two-sample tests, the function compares edge weights between two groups using Welch's t-test (unequal variances assumed).

For paired tests, the function calculates the difference between matched pairs and performs a one-sample t-test on the differences (testing if mean difference differs from zero). This is appropriate when samples are matched (e.g., Tumor and Normal from the same patient).

The returned `SE` is the raw sampling standard error before optional SAM-style variance moderation. It is intended for downstream effect-size meta-analysis. The moderated SE is used only internally for calculating the test statistic and p-value.

Edges are tested independently, and p-values are adjusted for multiple testing using the specified method.

The function uses fully vectorized computations for efficiency, making it suitable for large-scale analyses with millions of edges. T-statistics and p-values are calculated using matrix operations without iteration.

Value

A data.frame containing:

- tf: Transcription factor
- target: Target gene
- meanEdge: Mean edge weight
- SE: Raw, unmoderated sampling standard error
- tStatistic: Test statistic
- pValue: Raw p-value
- pAdj: Adjusted p-value
- For two-sample tests: meanGroup1, meanGroup2, cohensD, log2FoldChange (Group1 - Group2)

Author(s)

Daniel Osorio <daniecos@uio.no>

See Also

runSCORPION, maEdges, circosEdges

Examples

```
## Not run:
# Load test data and build networks by donor and region
# Note: T = Tumor, N = Normal, B = Border regions
data(scorpionTest)
nets <- runSCORPION(
  gexMatrix = scorpionTest$gex,
  tfMotifs = scorpionTest$tf,
  ppiNet = scorpionTest$ppi,
  cellsMetadata = scorpionTest$metadata,
  groupBy = c("donor", "region")
)

# Single-sample test: Test if edges in Tumor region differ from zero
tumor_nets <- grep("--T$", colnames(nets), value = TRUE)
results_single <- testEdges(
  networksDF = nets,
  testType = "single",
  group1 = tumor_nets
)

# Two-sample test: Compare Tumor vs Border regions
tumor_nets <- grep("--T$", colnames(nets), value = TRUE)
```

```
border_nets <- grep("--B$", colnames(nets), value = TRUE)
results_tumor_vs_border <- testEdges(
  networksDF = nets,
  testType = "two.sample",
  group1 = tumor_nets,
  group2 = border_nets
)

# View top differential edges (Tumor vs Border)
head(results_tumor_vs_border[order(results_tumor_vs_border$pAdj), ])

# Compare Tumor vs Normal regions
normal_nets <- grep("--N$", colnames(nets), value = TRUE)
results_tumor_vs_normal <- testEdges(
  networksDF = nets,
  testType = "two.sample",
  group1 = tumor_nets,
  group2 = normal_nets
)

# Filter by minimum log2 fold change for focused analysis
results_filtered <- testEdges(
  networksDF = nets,
  testType = "two.sample",
  group1 = tumor_nets,
  group2 = normal_nets,
  minLog2FC = 0.5
)

# Paired t-test: Compare matched Tumor vs Normal samples
tumor_nets_ordered <- c("P31--T", "P32--T", "P33--T")
normal_nets_ordered <- c("P31--N", "P32--N", "P33--N")
results_paired <- testEdges(
  networksDF = nets,
  testType = "two.sample",
  group1 = tumor_nets_ordered,
  group2 = normal_nets_ordered,
  paired = TRUE
)

## End(Not run)
```

Index