

Package: scTenifoldKnk (via r-universe)

October 8, 2026

Type Package

Title In-Silico Knockout Experiments from Single-Cell Gene Regulatory Networks

Version 2.0.0

Description A workflow based on 'scTenifoldNet' to perform in-silico knockout experiments using single-cell RNA sequencing (scRNA-seq) data from wild-type (WT) control samples as input. First, the package constructs a single-cell gene regulatory network (scGRN) and knocks out a target gene from the adjacency matrix of the WT scGRN by setting the gene's outdegree edges to zero. Then, it compares the knocked out scGRN with the WT scGRN to identify differentially regulated genes, called virtual-knockout perturbed genes, which are used to assess the impact of the gene knockout and reveal the gene's function in the analyzed cells. It also predicts the direction (up or down) of the response of each gene from the WT expression, and reads all knockouts of a network from a single heat kernel, which makes transcriptome-wide knockout screens practical.

URL <https://github.com/cailab-tamu/scTenifoldKnk>

BugReports <https://github.com/cailab-tamu/scTenifoldKnk/issues>

License GPL (>=2)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Imports Matrix, methods, stats, MASS, scTenifoldNet (>= 1.4.3), cli, enrichR, igraph, reshape2, grDevices, graphics

Suggests testthat (>= 2.1.0), locfdr

Config/pak/sysreqs libglpk-dev libicu-dev libxml2-dev libssl-dev perl

Repository <https://cailab-tamu.r-universe.dev>

Date/Publication 2026-10-08 13:09:18 UTC

RemoteUrl <https://github.com/cailab-tamu/scTenifoldKnk>

RemoteRef HEAD
RemoteSha e003c72d3a153ec07512271c1ae0fbeeadc319ca

Contents

dRegulation	2
heatKernel	4
hkManifoldAlignment	5
knockoutDirection	6
perturbationMap	7
plotKO	9
scQC	10
scTenifoldKnk	11
Index	17

dRegulation	<i>Evaluates gene differential regulation based on manifold alignment distances.</i>
-------------	--

Description

Using the output of the non-linear manifold alignment, this function computes the Euclidean distance between the coordinates for the same gene in both conditions. Calculated distances are then transformed using Box-Cox power transformation, and standardized to ensure normality. P-values are assigned following the chi-square distribution over the fold-change of the squared distance computed with respect to the expectation, the mean squared distance of the genes that were not knocked out (gKO), or, when empiricalNull = TRUE, using Efron’s empirical null estimated from the Z-scores with locfdr. Genes whose distance is at the level of floating-point noise (at most sqrt(.Machine\$double.eps) times the largest absolute coordinate) did not move between conditions and get a p-value of 1; if this applies to every gene, for example when the knockout leaves the network unchanged, a warning is raised.

Usage

```
dRegulation(  
  manifoldOutput,  
  gKO = NULL,  
  empiricalNull = FALSE,  
  direction = NULL  
)
```

Arguments

manifoldOutput A matrix. The output of the non-linear manifold alignment, a labeled matrix with two times the number of shared genes as rows (X_ genes followed by Y_ genes in the same order) and d number of columns.

gKO	A character vector with the knocked-out genes, or NULL. These genes are perturbed by construction, so they are left out of the expectation used to compute the fold-changes; otherwise their large distances inflate it and hide the other genes. They are still reported in the output. Default: NULL, the expectation is computed from all genes.
empiricalNull	A boolean value (TRUE/FALSE). If TRUE, p-values are assigned using Efron's empirical null: the null distribution of the Z-scores is estimated from the bulk of the data with <code>locfdr::locfdr</code> instead of assuming the theoretical chi-square null. Requires the <code>locfdr</code> package. Default: FALSE.
direction	Optional named numeric vector of direction scores (names: genes), for example a row of knockoutDirection . If given, the columns <code>direction</code> and <code>directionScore</code> are appended. Default: NULL.

Value

A data frame with 6 columns (8 when `direction` is given) as follows:

- `gene` A character vector with the gene id identified from the `manifoldAlignment` output.
- `distance` A numeric vector of the Euclidean distance computed between the coordinates of the same gene in both conditions.
- `Z` A numeric vector of the Z-scores computed after Box-Cox power transformation.
- `FC` A numeric vector of the FC computed with respect to the expectation, the mean squared distance of the genes not in gKO.
- `p.value` A numeric vector of the p-values associated to the fold-changes, probabilities are assigned as $P[X > x]$ using the Chi-square distribution with one degree of freedom.
- `p.adj` A numeric vector of adjusted p-values using Benjamini & Hochberg (1995) FDR correction.
- `direction` Only when `direction` is given: "up" or "down", the predicted direction of the change of the gene; the knocked-out genes are "down" by construction.
- `directionScore` Only when `direction` is given: the signed direction score (NA for the knocked-out genes).

References

- Benjamini, Y., and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, 29, 1165-1188. doi: 10.1214/aos/1013699998.

Examples

```
library(scTenifoldKnk)

# Simulating a dataset following a negative binomial distribution with high sparsity (~67%)
nCells = 2000
nGenes = 100
set.seed(1)
X <- rnbinom(n = nGenes * nCells, size = 20, prob = 0.98)
X <- round(X)
```

```

X <- matrix(X, ncol = nCells)
rownames(X) <- c(paste0('ng', 1:90), paste0('mt-', 1:10))

# Performing single-cell quality control
qcOutput <- scQC(
  X = X,
  minLibSize = 30,
  removeOutlierCells = TRUE,
  minPCT = 0.05,
  maxMTratio = 0.1
)

# Computing 3 gene regulatory networks from subsamples of 500 cells
xNetworks <- scTenifoldNet::makeNetworks(
  X = qcOutput,
  nNet = 3,
  nCells = 500,
  nComp = 3,
  scaleScores = TRUE,
  symmetric = FALSE,
  q = 0.95
)

# Computing a K = 3 CANDECOMP/PARAFAC (CP) Tensor Decomposition
tdOutput <- scTenifoldNet::tensorDecomposition(xNetworks, K = 3, maxError = 1e5, maxIter = 1e3)

## Not run:
# Computing the manifold alignment
maOutput <- scTenifoldNet::manifoldAlignment(tdOutput$X, tdOutput$X)

# Evaluating differential regulation
drOutput <- dRegulation(maOutput)
head(drOutput)

# Plotting - genes with FDR < 0.05 colored in red
geneColor <- ifelse(drOutput$p.adj < 0.05, 'red', 'black')
qqnorm(drOutput$Z, main = 'Standardized Distance', pch = 16, col = geneColor)
qqline(drOutput$Z)

## End(Not run)

```

heatKernel

Spectral heat kernel of a gene-gene matrix

Description

Computes the heat kernel

$$H = \sum_k \exp(t(\lambda_k / \lambda_{max} - 1)) v_k v_k^T$$

from the eigendecomposition $X = \sum_k \lambda_k v_k v_k^T$ of a symmetric gene-gene matrix (a gene regulatory network or a gene-gene correlation matrix). Row i of H describes how a change in gene i diffuses over the whole network, summing paths of every length; larger t concentrates the kernel on the dominant network modules.

Usage

```
heatKernel(X, t = 10, symmetric = TRUE)
```

Arguments

<code>X</code>	A square numeric matrix (or <code>Matrix</code>) with genes as row and column names.
<code>t</code>	A non-negative number. Diffusion time. <code>t = 0</code> returns the identity.
<code>symmetric</code>	A boolean value (TRUE/FALSE). If TRUE, <code>X</code> is replaced by its symmetric part $(X + t(X)) / 2$ before the eigendecomposition; required for directed networks such as the <code>scTenifoldKnk</code> WT network. Default: TRUE.

Value

A square numeric matrix with the same dimension names as `X`.

Examples

```
set.seed(1)
A <- matrix(rnorm(25), 5, 5, dimnames = list(letters[1:5], letters[1:5]))
H <- heatKernel(A, t = 10)
isSymmetric(H)
```

hkManifoldAlignment *Heat-kernel (hk) manifold alignment for in-silico knockouts*

Description

Kernel counterpart of the non-linear manifold alignment used by `scTenifoldKnk`. The heat kernel of the WT network is computed once ([heatKernel](#)), and the knockout of each gene (or gene set) is read from its rows: removing gene x lowers it by its mean expression and the change diffuses over the network,

$$\Delta_g = -\frac{\text{mean}(x)}{\text{sd}(x)} H[x, g] \text{sd}(g)$$

on $\log_{10}(\text{CPM})$ expression (summed over the genes of a multi-gene knockout). The absolute value $|\Delta_g|$ plays the role of the manifold alignment distance: it ranks the perturbed genes like the manifold alignment does, without recomputing an alignment for every knockout, which makes transcriptome-wide perturbation efficient.

Usage

```
hkManifoldAlignment(WT, X, gKO = NULL, t = 10, H = NULL)
```

Arguments

WT	The WT gene regulatory network (genes x genes, regulators as rows), as returned in <code>scTenifoldKnk(...)\$tensorNetworks\$WT</code> .
X	Raw counts matrix (genes x cells) of the WT cells, with row names matching the genes of WT (typically the quality-controlled matrix used to build the network).
gK0	A character vector or a list. Each element is a gene (or a character vector of genes knocked out together) to perturb. If NULL, every gene of the network is knocked out separately.
t	A non-negative number. Diffusion time of the heat kernel. Default: 10.
H	Optional pre-computed heat kernel of WT (from <code>heatKernel(WT, t)</code>), to reuse it across calls.

Value

A numeric matrix of perturbation distances $|\Delta_g|$ (for the knocked-out genes themselves, their mean $\log_{1p}(\text{CPM})$ expression, the amount they lose) with one row per knockout (named by the knocked-out genes, joined with "+" for multi-gene knockouts) and one column per gene of the network.

See Also

[heatKernel](#), [knockoutDirection](#)

knockoutDirection	<i>Direction of the response to an in-silico knockout</i>
-------------------	---

Description

Predicts whether each gene goes up or down after knocking out gK0, using only the WT expression data: the heat kernel ([heatKernel](#)) of the gene-gene Pearson correlation matrix of $\log_{1p}(\text{CPM})$ expression is diffused from the knocked-out gene(s),

$$s_g = - \sum_{x \in gK0} \frac{\text{mean}(x)}{\text{sd}(x)} H[x, g] \text{sd}(g)$$

, and the sign of s_g is the predicted direction.

Evaluated against bulk knockdown/knockout profiles of the same cell lines, this direction is correct more often than chance, but it mostly reflects the response shared by most knockdowns along the dominant WT expression program rather than regulation specific to gK0, and its accuracy varies between cell types and WT data sets.

In single-cell data, $\log_{1p}(\text{CPM})$ expression often keeps a dependence on sequencing depth that makes nearly all genes correlate positively; the heat kernel then follows that axis and predicts almost every gene to go down. With `regressLibSize = TRUE`, log library size is regressed out of each gene before computing the correlations, which removes the depth axis. It is off by default: it corrected tissue data where nearly all genes were predicted down, but in cell lines it left the direction unchanged or made it slightly worse, because the expression that follows library size can be biological.

Usage

```
knockoutDirection(X, gK0, genes = rownames(X), t = 5, regressLibSize = FALSE)
```

Arguments

X Raw counts matrix (genes x cells) of the WT cells.

gK0 A character vector or a list, as in [hkManifoldAlignment](#).

genes A character vector with the genes to score (for example the genes of the WT network). Default: all genes of X.

t A non-negative number. Diffusion time of the heat kernel. Default: 5.

regressLibSize A boolean value (TRUE/FALSE). If TRUE, the log library size of each cell (column sums of X) is regressed out of the log1p(CPM) expression of each gene before computing the gene-gene correlation matrix. Default: FALSE.

Value

A numeric matrix of direction scores s_g (positive: predicted up, negative: predicted down) with one row per knockout and one column per gene.

See Also

[heatKernel](#), [dRegulation](#)

perturbationMap	<i>Map of transcriptome-wide perturbation profiles</i>
-----------------	--

Description

Places every virtual knockout from a transcriptome-wide run of [scTenifoldKnk](#) on a two-dimensional map together with a reference signature, such as a disease-vs-control expression signature, and highlights the genes whose knockout reproduces it. Each knockout is described by its signed perturbation profile: the rank of the perturbation distance of every gene, scaled to (0, 1], multiplied by its predicted direction. The similarity of a profile to the signature is the cosine between both over the genes they share; the knocked-out gene itself is left out of its own profile.

Usage

```
perturbationMap(
  X,
  signature,
  genes = NULL,
  layout = c("signature", "pca"),
  nLabels = 10,
  plot = TRUE
)
```

Arguments

<code>X</code>	A list. Output from <code>scTenifoldKnk</code> with <code>transcriptomeWide = TRUE</code> and <code>dr_direction = TRUE</code> , which contains <code>perturbationDistances</code> and <code>perturbationDirections</code> .
<code>signature</code>	A named numeric vector, for example the log fold changes of a disease-vs-control comparison in the same cell type. Names must be gene identifiers matching those of the network.
<code>genes</code>	Character. Optional genes to highlight, such as known causal genes. Default: <code>NULL</code> .
<code>layout</code>	Character, "signature" or "pca". "signature" (default) places each knockout by its similarity to the signature (x axis) and by the main axis of the remaining variation of the profiles (y axis). "pca" shows the first two principal components of the profiles and projects the signature into the same space.
<code>nLabels</code>	Integer. Number of knockouts with the highest similarity to label, in addition to genes. Default: 10.
<code>plot</code>	Logical. If <code>TRUE</code> (default), draw the map.

Value

Invisibly, a data frame with one row per knockout, sorted by decreasing similarity: gene, similarity (cosine to the signature), rank (1 = most similar), percentile (fraction of knockouts at least as similar), and the map coordinates x and y. With `layout = "pca"`, the coordinates of the signature are stored in the "signature" attribute.

Examples

```
library(scTenifoldKnk)

# Counts with three gene modules
set.seed(1)
modules <- matrix(rgamma(3 * 600, shape = 2), 3, 600)
loadings <- matrix(0, 60, 3)
loadings[cbind(1:60, rep(1:3, length.out = 60))] <- runif(60, 0.5, 2)
X <- matrix(rpois(60 * 600, lambda = 5 * loadings %*% modules), 60, 600)
rownames(X) <- paste0('g', 1:60)
colnames(X) <- paste0('c', 1:600)

# Knock out every gene of the network
O <- scTenifoldKnk(X, transcriptomeWide = TRUE, qc = FALSE,
                  nc_nNet = 3, nc_nCells = 300, nCores = 1)

# A signature that resembles the knockout of g1
sig <- O$perturbationDistances['g1', ] * O$perturbationDirections['g1', ]
sig <- sig + rnorm(length(sig), sd = sd(sig))

M <- perturbationMap(O, signature = sig, genes = 'g1')
head(M)
```

plotKO

*Plot KO network***Description**

Generate and plot a KO-centered subnetwork from the output of `scTenifoldKnk`. The function selects genes with significant differential regulation ($\text{FDR} < 0.05$), extracts their interactions from the reconstructed WT network, filters edges by weight quantile, and displays the network using `igraph`. When `annotate = TRUE` the function queries enrichment databases via `enrichR` and overlays category pies on nodes with a legend of significant terms.

Usage

```
plotKO(
  X,
  gKO,
  q = 0.99,
  annotate = TRUE,
  nCategories = 20,
  fdrThreshold = 0.05
)
```

Arguments

<code>X</code>	A list. Output from <code>scTenifoldKnk</code> .
<code>gKO</code>	Character. Gene symbol(s) of the simulated knockout gene(s), as passed to <code>scTenifoldKnk</code> .
<code>q</code>	Numeric. Edge-weight quantile used to threshold weak edges. Default: 0.99.
<code>annotate</code>	Logical. If TRUE, query enrichment databases and overlay category pies on enriched nodes. Default: TRUE.
<code>nCategories</code>	Integer. Maximum number of enrichment categories to show in the legend. Default: 20.
<code>fdrThreshold</code>	Numeric. Adjusted p-value cutoff (FDR) for reporting enriched terms. Default: 0.05.

Value

Invisibly returns NULL. Called for the side effect of plotting the network.

Examples

```
## Not run:
library(scTenifoldKnk)

# Load example data
scRNAseq <- system.file("single-cell/example.csv", package = "scTenifoldKnk")
```

```

scRNAseq <- read.csv(scRNAseq, row.names = 1)

# Run scTenifoldKnk
output <- scTenifoldKnk(countMatrix = scRNAseq, gK0 = "G100", qc_minLibSize = 0)

# Plot the K0-centered subnetwork with enrichment annotation
plotK0(output, gK0 = "G100")

# Plot without enrichment annotation
plotK0(output, gK0 = "G100", annotate = FALSE)

## End(Not run)

```

scQC

Performs single-cell data quality control

Description

This function performs quality control filters over the provided input matrix. It checks for minimum cell library size, mitochondrial ratio, outlier cells, and the fraction of cells where a gene is expressed.

Usage

```

scQC(
  X,
  minLibSize = 1000,
  removeOutlierCells = TRUE,
  minPCT = 0.05,
  maxMTratio = 0.1,
  label = NULL
)

```

Arguments

X	Raw counts matrix with cells as columns and genes (symbols) as rows.
minLibSize	An integer value. Defines the minimum library size required for a cell to be included in the analysis.
removeOutlierCells	A boolean value (TRUE/FALSE), if TRUE, the identified cells with library size greater than 1.58 IQR/sqrt(n) computed from the sample, are removed. For further details see: <code>?boxplot.stats</code>
minPCT	A decimal value between 0 and 1. Defines the minimum fraction of cells where the gene needs to be expressed to be included in the analysis.
maxMTratio	A decimal value between 0 and 1. Defines the maximum ratio of mitochondrial reads (mitochondrial reads / library size) present in a cell to be included in the analysis. It's computed using the symbol genes starting with 'MT-' non-case sensitive.
label	Optional character label prepended to progress messages when running inside a pipeline.

Value

A dgCMatrx object with the cells and the genes that pass the quality control filters.

References

Ilicic, Tomislav, et al. "Classification of low quality cells from single-cell RNA-seq data." *Genome biology* 17.1 (2016): 29.

Examples

```
library(scTenifoldKnk)

# Simulating a dataset following a negative binomial distribution with high sparsity (~67%)
nCells = 2000
nGenes = 100
set.seed(1)
X <- rnbinom(n = nGenes * nCells, size = 20, prob = 0.98)
X <- round(X)
X <- matrix(X, ncol = nCells)
rownames(X) <- c(paste0('ng', 1:90), paste0('mt-', 1:10))

# Performing single-cell quality control
qcOutput <- scQC(
  X = X,
  minLibSize = 30,
  removeOutlierCells = TRUE,
  minPCT = 0.05,
  maxMTratio = 0.1
)

# Comparing dimensions before and after QC
dim(X)
dim(qcOutput)
```

scTenifoldKnk

scTenifoldKNK

Description

Predict gene perturbations using in-silico knockout experiments from single-cell gene regulatory networks.

Usage

```
scTenifoldKnk(
  countMatrix,
  gKO = NULL,
  transcriptomeWide = FALSE,
  qc = TRUE,
```

```

qc_minLibSize = 1000,
qc_removeOutlierCells = TRUE,
qc_minPCT = 0.05,
qc_maxMTratio = 0.1,
nc_lambda = 0,
nc_nNet = 10,
nc_nCells = 500,
nc_nComp = 3,
nc_scaleScores = TRUE,
nc_symmetric = FALSE,
nc_q = 0.9,
nc_priorNetwork = NULL,
td_K = 3,
td_maxIter = 1000,
td_maxError = 1e-05,
td_nDecimal = 3,
ma_nDim = 2,
ma_method = NULL,
ma_heatT = 10,
dr_empiricalNull = FALSE,
dr_direction = TRUE,
dr_directionT = 5,
dr_directionRegressLibSize = FALSE,
nCores = parallel::detectCores(),
seed = 1
)

```

Arguments

countMatrix	Raw counts matrix with cells as columns and genes (symbols) as rows, as a matrix or a sparse dgCMatix. A data.frame is not accepted; convert it with as.matrix() first.
gKO	Character. In knockout mode (transcriptomeWide = FALSE), the gene symbol of the gene to knock out, or a character vector of several genes to knock out together in a single simulated experiment (e.g. c("Hnf4a", "Hnf4g")). In transcriptome-wide mode (transcriptomeWide = TRUE), an optional character vector defining the subset of genes to perturb, each one knocked out separately; if NULL, every gene in the WT network is perturbed.
transcriptomeWide	A boolean value (TRUE/FALSE). If TRUE, the WT network is built once and each target gene is knocked out in turn, returning the manifold-alignment distances for every perturbation. Default: FALSE.
qc	A boolean value (TRUE/FALSE), if TRUE, a quality control is applied over the data.
qc_minLibSize	An integer value. Defines the minimum library size required for a cell to be included in the analysis.

qc_removeOutlierCells	A boolean value (TRUE/FALSE), if TRUE, cells with library size identified as outliers are removed. For further details see: <code>?boxplot.stats</code>
qc_minPCT	A decimal value between 0 and 1. Defines the minimum fraction of cells where the gene needs to be expressed to be included in the analysis.
qc_maxMTratio	A decimal value between 0 and 1. Defines the maximum ratio of mitochondrial reads (mitochondrial reads / library size) present in a cell to be included in the analysis. It's computed using the symbol genes starting with 'MT-' non-case sensitive.
nc_lambda	A continuous value between 0 and 1. Defines the multiplicative value (1-lambda) to be applied over the weaker edge connecting two genes to maximize the adjacency matrix directionality.
nc_nNet	An integer value. The number of networks based on principal components regression to generate.
nc_nCells	An integer value. The number of cells to subsample each time to generate a network.
nc_nComp	An integer value. The number of principal components in PCA to generate the networks. Should be greater than 2 and lower than the total number of genes.
nc_scaleScores	A boolean value (TRUE/FALSE), if TRUE, the weights will be normalized such that the maximum absolute value is 1.
nc_symmetric	A boolean value (TRUE/FALSE), if TRUE, the weights matrix returned will be symmetric.
nc_q	A decimal value between 0 and 1. Defines the cut-off threshold of top q% relationships to be returned.
nc_priorNetwork	A data.frame containing a prior gene regulatory network. The data.frame must have two columns: 'regulators' and 'targets'. Default: NULL.
td_K	An integer value. Defines the number of rank-one tensors used to approximate the data using CANDECOMP/PARAFAC (CP) Tensor Decomposition.
td_maxIter	An integer value. Defines the maximum number of iterations if error stay above <code>td_maxError</code> .
td_maxError	A decimal value between 0 and 1. Defines the relative Frobenius norm error tolerance.
td_nDecimal	An integer value indicating the number of decimal places to be used.
ma_nDim	An integer value. Defines the number of dimensions of the low-dimensional feature space to be returned from the non-linear manifold alignment.
ma_method	Character, "manifold" or "heat". How the WT and KO networks are compared: "manifold" runs the non-linear manifold alignment for each knockout; "heat" uses the heat manifold alignment (hkManifoldAlignment), which computes the heat kernel of the WT network once and reads every knockout from it. If NULL (default), "manifold" is used for single and multi-gene knockouts and "heat" when <code>transcriptomeWide = TRUE</code> .
ma_heatT	A non-negative number. Diffusion time of the heat kernel used when <code>ma_method = "heat"</code> . Default: 10.

dr_empiricalNull	A boolean value (TRUE/FALSE). If TRUE, the differential regulation p-values are assigned using Efron's empirical null (estimated with locfdr) instead of the theoretical chi-square null. Requires the locfdr package. Default: FALSE.
dr_direction	A boolean value (TRUE/FALSE). If TRUE, the predicted direction of the change of each gene (up/down) is added to the output, computed from the WT expression with knockoutDirection . Default: TRUE.
dr_directionT	A non-negative number. Diffusion time of the correlation heat kernel used to predict the direction. Default: 5.
dr_directionRegressLibSize	A boolean value (TRUE/FALSE). If TRUE, the log library size is regressed out of each gene's expression before computing the gene-gene correlations used to predict the direction (see knockoutDirection); consider it when nearly all genes are predicted down. Default: FALSE.
nCores	An integer value. Defines the number of cores to be used.
seed	An integer value. The RNG is set to this seed before each random stage (network construction, tensor decomposition and manifold alignment), so results are reproducible and independent of the caller's RNG state; the caller's RNG state is restored on exit. Use different values to assess run-to-run variability. If NULL, the RNG is never reseeded and the caller's RNG state (e.g. a previous <code>set.seed()</code>) drives all random stages. Default: 1.

Value

In single knockout mode (`transcriptomeWide = FALSE`), a list with 3 slots as follows:

- `tensorNetworks`: A list with the WT weight-averaged denoised gene regulatory network. The KO network is the WT network with the rows of `gKO` set to 0; it is not returned, which roughly halves the size of the output.
- `manifoldAlignment`: The generated low-dimensional features result of the non-linear manifold alignment (only when `ma_method = "manifold"`).
- `diffRegulation`: The results of the differential regulation analysis (see [dRegulation](#)), including the `direction` and `directionScore` columns when `dr_direction = TRUE`.

In transcriptome-wide mode (`transcriptomeWide = TRUE`), a list with 2 slots as follows:

- `tensorNetworks`: A list with the WT weight-averaged denoised gene regulatory network.
- `perturbationDistances`: A numeric matrix of perturbation distances (heat manifold alignment by default, manifold alignment when `ma_method = "manifold"`) with the perturbed genes as rows and all genes in the WT network as columns.
- `perturbationDirections`: Only when `dr_direction = TRUE`: a numeric matrix of the same dimensions with the predicted direction of each gene (1 up, -1 down, 0 undetermined).

Author(s)

Daniel Osorio <dcosorih@gmail.com>

Examples

```

library(scTenifoldKnk)

# Simulating a dataset following a negative binomial distribution with high sparsity (~67%)
nCells = 2000
nGenes = 100
set.seed(1)
X <- rnbinom(n = nGenes * nCells, size = 20, prob = 0.98)
X <- round(X)
X <- matrix(X, ncol = nCells)
rownames(X) <- c(paste0('ng', 1:90), paste0('mt-', 1:10))

## Not run:
# Running scTenifoldKnk – simulating knockout of gene ng10
output <- scTenifoldKnk(
  countMatrix = X,
  gKO = "ng10",
  nc_nNet = 10,
  nc_nCells = 500,
  td_K = 3,
  qc_minLibSize = 30
)

# Structure of the output
str(output)

# Accessing the WT gene regulatory network, and rebuilding the KO network
dim(output$tensorNetworks$WT)
KO <- output$tensorNetworks$WT
KO['ng10', ] <- 0

# Accessing the manifold alignment result
head(output$manifoldAlignment)

# Differential regulation results – top perturbed genes
head(output$diffRegulation, n = 10)

# Plotting the KO-centered subnetwork
plotKO(output, gKO = "ng10")

# Multi-gene knockout – ng10 and ng20 knocked out together
dkoOutput <- scTenifoldKnk(
  countMatrix = X,
  gKO = c("ng10", "ng20"),
  nc_nNet = 10,
  nc_nCells = 500,
  td_K = 3,
  qc_minLibSize = 30
)
head(dkoOutput$diffRegulation, n = 10)
plotKO(dkoOutput, gKO = c("ng10", "ng20"))

```

```
# Transcriptome-wide perturbation – knock out every gene in the WT network
twOutput <- scTenifoldKnk(
  countMatrix = X,
  transcriptomeWide = TRUE,
  nc_nNet = 10,
  nc_nCells = 500,
  td_K = 3,
  qc_minLibSize = 30
)

# Matrix of distances: perturbed genes (rows) by all genes (columns)
dim(twOutput$perturbationDistances)
twOutput$perturbationDistances[1:5, 1:5]

## End(Not run)
```

Index

dRegulation, [2](#), [7](#), [14](#)

heatKernel, [4](#), [5–7](#)

hkManifoldAlignment, [5](#), [7](#), [13](#)

knockoutDirection, [3](#), [6](#), [6](#), [14](#)

perturbationMap, [7](#)

plotKO, [9](#)

scQC, [10](#)

scTenifoldKnk, [7–9](#), [11](#)