Package ‘scTenifoldNet’

January 8, 2020

**Type** Package

**Title** Construct and Compare scGRN from Single-Cell Transcriptomic Data

**Version** 1.1.0

**Description**
A workflow based on machine learning methods to construct and compare single-cell gene regulatory networks (scGRN) using single-cell RNA-seq (scRNA-seq) data collected from different conditions. Uses principal component regression, tensor decomposition, and manifold alignment, to accurately identify even subtly shifted gene expression programs.

**URL** https://github.com/cailab-tamu/scTenifoldNet

**BugReports** https://github.com/cailab-tamu/scTenifoldNet/issues

**License** GPL (>= 2)

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.1

**biocViews**

**Imports** reticulate, pbapply, RSpectra, Matrix, rTensor, methods, stats, MASS

**Suggests** testthat (>= 2.1.0)

**NeedsCompilation** no

**Author** Daniel Osorio [aut, cre] (<https://orcid.org/0000-0003-4424-8422>), Yan Zhong [aut, ctb], Guanxun Li [aut, ctb], Jianhua Huang [aut, ctb], James Cai [aut, ctb, ths] (<https://orcid.org/0000-0002-8081-6725>)

**Maintainer** Daniel Osorio <dcosorioh@tamu.edu>

**Repository** CRAN

**Date/Publication** 2020-01-08 19:50:02 UTC
R topics documented:

cpmNormalization ......................................................... 2
dRegulation ................................................................. 3
makeNetworks ............................................................... 5
manifoldAlignment ......................................................... 7
pcNet ................................................................. 9
scQC ............................................................. 11
scTenifoldNet ............................................................. 14
tensorDecomposition .................................................... 16

Index 19

cpmNormalization  Performs counts per million (CPM) data normalization

Description

This function normalizes the count data present in a given matrix using counts per million normalization (CPM). Each gene count for each cell is divided by the total counts for that cell and multiplied by 1e6. No log-transformation is applied.

Usage

mpmNormalization(X)

Arguments

X  Raw counts matrix with cells as columns and genes (symbols) as rows

Value

A dgCMatrix object with the count per million (CPM) normalized values.

References


Examples

library(scTenifoldNet)

# Simulating of a dataset following a negative binomial distribution with high sparcity (~67%)
nCells = 2000
nGenes = 100
set.seed(1)
X <- rbinom(n = nGenes * nCells, size = 20, prob = 0.98)
X <- round(X)
X <- matrix(X, ncol = nCells)
# Evaluates gene differential regulation based on manifold alignment distances.

dRegulation <- function(manifoldOutput, minFC = 1.5) {
  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
  )

  # Performing Counts per million Normalization (CPM)
  normalizationOutput <- cpmNormalization(qcOutput)

  # Visualizing the differences
  oldPar <- par(no.readonly = TRUE)

  par(
    mfrow = c(1, 2),
    mar = c(3, 3, 1, 1),
    mgp = c(1.5, 0.5, 0)
  )

  plot(
    Matrix::colSums(qcOutput),
    ylab = 'Library Size',
    xlab = 'Cell',
    main = 'Before CPM Normalization'
  )

  plot(
    Matrix::colSums(normalizationOutput),
    ylab = 'Library Size',
    xlab = 'Cell',
    main = 'After CPM Normalization'
  )

  par(oldPar)
}

dRegulation

## 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 computed with respect to the expectation.

## Usage

dRegulation(manifoldOutput, minFC = 1.5)
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.

minFC  A decimal value. Defines the cut-off threshold of fold-change to limit the testing to genes that show, at least minFC deviation.

Value

A data frame with 5 columns 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.
- 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.

References


Examples

```r
library(scTenifoldNet)

# Simulating of a dataset following a negative binomial distribution with high sparcity (~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(‘Var’, 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 single-cell gene regulatory networks each one from a subsample of 500 cells
```
makeNetworks

Computes gene regulatory networks for subsamples of cells based on principal component regression.

Description

This function computes nNet gene regulatory networks for a randomly selected subsample of nCells cells based on principal component regression (PCR), a technique based on principal component analysis. In PCR, the outcome variable is regressed over a nComp number of for principal components computed from a set of covariates to estimate the unknown regression coefficients in the model. pcNet function computes the PCR coefficients for each gene one at a time using all the others as covariates, to construct an all by all gene regulatory network.

Usage

makeNetworks(X, nNet = 10, nCells = 500, nComp = 3, scaleScores = TRUE, symmetric = FALSE, q = 0.95)
makeNetworks

Arguments

- **X**: A filtered and normalized gene expression matrix with cells as columns and genes as rows.
- **nNet**: An integer value. The number of networks based on principal components regression to generate.
- **nCells**: An integer value. The number of cells to subsample each time to generate a network.
- **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.
- **scaleScores**: A boolean value (TRUE/FALSE), if TRUE, the weights will be normalized such that the maximum absolute value is 1.
- **symmetric**: A boolean value (TRUE/FALSE), if TRUE, the weights matrix returned will be symmetric.
- **q**: A decimal value between 0 and 1. Represent the cut-off threshold of top q% relationships to be returned.

Details

Principal component regression may be broadly divided into three major steps:

1. Perform PCA on the observed covariates data matrix to obtain nComp number of the principal components.
2. Regress the observed vector of outcomes on the selected principal components as covariates, using ordinary least squares regression to get a vector of estimated regression coefficients.
3. Transform this vector back to the scale of the actual covariates, using the eigenvectors corresponding to the selected principal components to get the final PCR estimator for estimating the regression coefficients characterizing the original model.

Value

A list with nNet gene regulatory networks in dgCMatrix format. Each one computed from a randomly selected subsample of nCells cells.

References

Examples

library(scTenifoldNet)

# 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('Var', 1:90), paste0('Var', 1:10))

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

# Computing 3 single-cell gene regulatory networks each one from a subsample of 500 cells
mnOutput <- makeNetworks(X = X,
  nNet = 3,
  nCells = 500,
  nComp = 3,
  scaleScores = TRUE,
  symmetric = FALSE,
  q = 0.95
)

# Verifying the class
class(mnOutput)

# Verifying the number of networks
length(mnOutput)

# Verifying the dimension of the networks
lapply(mnOutput,dim)

# Single-cell gene regulatory networks
mnOutput[[1]][1:10,1:10]
mnOutput[[2]][1:10,1:10]
mnOutput[[3]][1:10,1:10]

manifoldAlignment Performs non-linear manifold alignment of two gene regulatory networks.
Description

Build comparable low-dimensional features for two weight-averaged denoised single-cell gene regulatory networks. Using a non-linear network embedding method manifoldAlignment aligns two gene regulatory networks and finds the structural similarities between them. This function is a wrapper of the Python code provided by Vu et al., (2012) at https://github.com/all-umass/ManifoldWarping.

Usage

manifoldAlignment(X, Y, d = 30)

Arguments

X
A gene regulatory network.

Y
A gene regulatory network.

d
The dimension of the low-dimensional feature space.

Details

Manifold alignment builds connections between two or more disparate data sets by aligning their underlying manifolds and provides knowledge transfer across the data sets. For further information please see: Wang et al., (2009)

Value

A low-dimensional projection for two the two gene regulatory networks used as input. The output is 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.

References


Examples

library(scTenifoldNet)

# Simulating of a dataset following a negative binomial distribution with high sparcity (~67%)
nCells = 2000
nGenes = 100
set.seed(1)
X <- rbinom(n = nGenes * nCells, size = 20, prob = 0.98)
X <- round(X)
X <- matrix(X, ncol = nCells)ownames(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 single-cell gene regulatory networks each one from a subsample of 500 cells
xNetworks <- makeNetworks(X = X,
    nNet = 3,
    nCells = 500,
    nComp = 3,
    scaleScores = TRUE,
    symmetric = FALSE,
    q = 0.95
)

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

## Not run:
# Computing the alignment
# For this example, we are using the same input, the match should be perfect.
maOutput <- manifoldAlignment(tdOutput$X, tdOutput$X)

# Separating the coordinates for each gene
X <- maOutput[grepl('X_', rownames(maOutput)),]
Y <- maOutput[grepl('Y_', rownames(maOutput)),]

# Plotting
# X Points
plot(X, pch = 16)

# Y Points
points(Y, col = 'red')

# Legend
legend('topright', legend = c('X', 'Y'),
    col = c('black', 'red'), bty = 'n',
    pch = c(16,1), cex = 0.7)

## End(Not run)
Description

This function computes a gene regulatory network based on principal component regression (PCR), a technique based on principal component analysis. In PCR, the outcome variable is regressed over a \( n_{\text{Comp}} \) number of principal components computed from a set of covariates to estimate the unknown regression coefficients in the model. `pcNet` function computes the PCR coefficients for each gene one at a time using all the others as covariates, to construct an all by all gene regulatory network.

Usage

```r
pcNet(X, nComp = 3, scaleScores = TRUE, symmetric = FALSE, q = 0,
      verbose = TRUE)
```

Arguments

- **X**: A filtered and normalized gene expression matrix with cells as columns and genes as rows.
- **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.
- **scaleScores**: A boolean value (TRUE/FALSE), if TRUE, the weights will be normalized such that the maximum absolute value is 1.
- **symmetric**: A boolean value (TRUE/FALSE), if TRUE, the weights matrix returned will be symmetric.
- **q**: A decimal value between 0 and 1. Defines the cut-off threshold of top q% relationships to be returned.
- **verbose**: A boolean value (TRUE/FALSE), if TRUE, a progress bar is shown.

Details

Principal component regression may be broadly divided into three major steps:

1. Perform PCA on the observed covariates data matrix to obtain \( n_{\text{Comp}} \) number of the principal components.
2. Regress the observed vector of outcomes on the selected principal components as covariates, using ordinary least squares regression to get a vector of estimated regression coefficients.
3. Transform this vector back to the scale of the actual covariates, using the eigenvectors corresponding to the selected principal components to get the final PCR estimator for estimating the regression coefficients characterizing the original model.

Value

A gene regulatory network in dgCMatrix format.
References


Examples

```r
library(scTenifoldNet)

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

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

# Computing a single-cell gene regulatory network using principal component regression
# Non-symmetric
pcnetOutput <- pcNet(X = qcOutput, nComp = 3, scaleScores = TRUE, symmetric = FALSE, q = 0)
pcnetOutput[1:10,1:10]

# Symmetric
pcnetOutput <- pcNet(X = qcOutput, nComp = 3, scaleScores = TRUE, symmetric = TRUE, q = 0)
pcnetOutput[1:5,1:5]
```

scQC Performs single-cell data quality control

Description

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

```r
scQC(X, minLibSize = 1000, removeOutlierCells = TRUE, minPCT = 0.05,
    maxMTratio = 0.1)
```

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: `?boxplot.stats`
- **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.

Value

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

References


Examples

```r
library(scTenifoldNet)

# Simulating of a dataset following a negative binomial distribution with high sparcity (~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("Var", 1:90), paste0("Varmt-", 1:10))

# Performing Single cell quality control
qcOutput <- scQC(
    X = X,
    minLibSize = 30,
    removeOutlierCells = TRUE,
    minPCT = 0.05,
    maxMTratio = 0.1)
```
# Visualizing the Differences

oldPar <- par(no.readonly = TRUE)

par(
  mfrow = c(2, 2),
  mar = c(3, 3, 1, 1),
  mgp = c(1.5, 0.5, 0)
)

# Library Size
plot(
  Matrix::colSums(X),
  ylim = c(20, 70),
  ylab = 'Library Size',
  xlab = 'Cell',
  main = 'Library Size - Before QC'
)
abline(h = c(30, 58),
  lty = 2,
  col = 'red'
)

plot(
  Matrix::colSums(qcOutput),
  ylim = c(20, 70),
  ylab = 'Library Size',
  xlab = 'Cell',
  main = 'Library Size - After QC'
)
abline(h = c(30, 58),
  lty = 2,
  col = 'red'
)

# Mitochondrial ratio
mtGenes <- grepl('^mt-', rownames(X), ignore.case = TRUE)
plot(
  Matrix::colSums(X[mtGenes,]) / Matrix::colSums(X),
  ylim = c(0, 0.3),
  ylab = 'Mitochondrial Ratio',
  xlab = 'Cell',
  main = 'Mitochondrial Ratio - Before QC'
)
abline(h = c(0.1), lty = 2, col = 'red')

plot(
  Matrix::colSums(qcOutput[mtGenes,]) / Matrix::colSums(qcOutput),
  ylim = c(0, 0.3),
  ylab = 'Mitochondrial Ratio',
  xlab = 'Cell',
  main = 'Mitochondrial Ratio - Before QC'
)
abline(h = c(0.1), lty = 2, col = 'red')

par(oldPar)
Description

Construct and compare single-cell gene regulatory networks (scGRNs) using single-cell RNA-seq (scRNA-seq) data sets collected from different conditions based on principal component regression, tensor decomposition, and manifold alignment.

Usage

```r
scTenifoldNet(X, Y, qc_minLibSize = 1000, qc_removeOutlierCells = TRUE,
    qc_minPCT = 0.05, qc_maxMTratio = 0.1, nc_nNet = 10,
    nc_nCells = 500, nc_nComp = 3, nc_symmetric = FALSE,
    nc_scaleScores = TRUE, nc_q = 0.05, td_K = 3, td_maxIter = 1000,
    td_maxError = 1e-05, ma_nDim = 30, dc_minFC = 1.5)
```

Arguments

- **X**
  - Raw counts matrix with cells as columns and genes (symbols) as rows.

- **Y**
  - Raw counts matrix with cells as columns and genes (symbols) as rows.

- **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, the identified cells with library size greater than 1.58 IQR/sqrt(n) computed from the sample, are removed. For further details see: ?boxplot.stats

- **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_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_symmetric**
  - A boolean value (TRUE/FALSE), if TRUE, the weights matrix returned will be symmetric.

- **nc_scaleScores**
  - A boolean value (TRUE/FALSE), if TRUE, the weights will be normalized such that the maximum absolute value is 1.
nc_q  A decimal value between 0 and 1. Defines the cut-off threshold of top q% relationships to be returned.

dc_maxIter  An integer value. Defines the maximum number of iterations if error stay above td_maxError.

dc_maxError  A decimal value between 0 and 1. Defines the relative Frobenius norm error tolerance.

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.

dc_minFC  A decimal value. Defines the cut-off threshold of fold-change to limit the testing to genes that show, at least dc_minFC deviation.

**Value**

A list with 3 slots as follows:

- `manifoldAlignment`: The generated low-dimensional features result of the non-linear manifold alignment.
- `diffRegulation`: The results of the differential regulation analysis.

**Examples**

```r
library(scTenifoldNet)

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

# Generating a perturbed network modifying the expression of genes 10, 2 and 3
Y <- X
Y[10,] <- Y[50,]
Y[2,] <- Y[11,]
Y[3,] <- Y[5,]

## Not run:
# scTenifoldNet
Output <- scTenifoldNet(X = X, Y = Y, 
                         nc_nNet = 10, nc_nCells = 500, 
                         td_K = 3, qc_minLibSize = 30, 
                         dc_minDist = 0)
```
tensorDecomposition

Performs CANDECOMP/PARAFAC (CP) Tensor Decomposition.

Description

Generate weight-averaged denoised gene regulatory networks using CANDECOMP/PARAFAC (CP) Tensor Decomposition. The tensorDecomposition function takes one or two lists of gene regulatory matrices, if two list are provided, the shared genes are selected and the CP tensor decomposition is performed independently for each list (3d-tensor). The tensor decomposed matrices are then averaged to generate weight-averaged denoised networks.

Usage

tensorDecomposition(xList, yList = NULL, K = 5, maxError = 1e-05, maxIter = 1000)

Arguments

xList A list of gene regulatory networks.
yList Optional. A list of gene regulatory networks.
K The number of rank-one tensors used to approximate the data using CANDECOMP/PARAFAC (CP) Tensor Decomposition.
maxError A decimal value between 0 and 1. Defines the relative Frobenius norm error tolerance
maxIter An integer value. Defines the maximum number of iterations if error stay above maxError.

Details

CANDECOMP/PARAFAC (CP) tensor decomposition approximate a K-Tensor using a sum of K rank-1 K-Tensors. A rank-1 K-Tensor can be written as an outer product of K vectors. This is an iterative algorithm, with two possible stopping conditions: either relative error in Frobenius norm has gotten below maxError, or the maxIter number of iterations has been reached. For more details on CP decomposition, consult Kolda and Bader (2009) and Morup (2011).
Value

A list of weight-averaged denoised gene regulatory networks.

References


Examples

```r
library(scTenifoldNet)

# Simulating of a dataset following a negative binomial distribution with high sparcity (~67%)
nCells = 2000
nGenes = 100
set.seed(1)
X <- rbinom(n = nGenes * nCells, size = 20, prob = 0.98)
X <- round(X)
X <- matrix(X, ncol = nCells)ownames(X) <- c(paste0(quot('Var', 1:90), paste0('Var', 1:10))

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

# Computing 3 single-cell gene regulatory networks each one from a subsample of 500 cells
mnOutput <- makeNetworks(X = X,
    nNet = 3,
    nCells = 500,
    nComp = 3,
    scaleScores = TRUE,
    symmetric = FALSE,
    q = 0.95
)

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

# Verifying the number of networks
length(tdOutput)

# Veryfying the dimention of the networks
lapply(tdOutput,dim)
```
# Weight-averaged denoised single-cell gene regulatory networks

```plaintext
tdOutput[[1]][1:10,1:10]
```
Index

cpmNormalization, 2

dRegulation, 3

makeNetworks, 5

manifoldAlignment, 7

pcNet, 9

scQC, 11

scTenifoldNet, 14

tensorDecomposition, 16