Package ‘SparseDC’

January 4, 2018

Type Package

Title Implementation of SparseDC Algorithm

Version 0.1.17

Description Implements the algorithm described in Barron, M., Zhang, S. and Li, J. 2017, ‘A sparse differential clustering algorithm for tracing cell type changes via single-cell RNA-sequencing data’, Nucleic Acids Research, gkx1113, <doi:10.1093/nar/gkx1113>. This algorithm clusters samples from two different populations, links the clusters across the conditions and identifies marker genes for these changes. The package was designed for scRNA-Seq data but is also applicable to many other data types, just replace cells with samples and genes with variables. The package also contains functions for estimating the parameters for SparseDC as outlined in the paper. We recommend that users further select their marker genes using the magnitude of the cluster centers.

Depends R (>= 3.1.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

Imports stats

NeedsCompilation no

Author Jun Li [aut, cre], Martin Barron [aut]

Maintainer Jun Li <jun.li@nd.edu>

Repository CRAN

Date/Publication 2018-01-04 19:11:30 UTC
**R topics documented:**

- **cell_type_biase**
- **condition_biase**
- **data_biase**
- **generate_uni_dat**
- **lambda1_calculator**
- **lambda2_calculator**
- **pre_proc_data**
- **sim_data**
- **sparsedc_cluster**
- **sparsedc_gap**
- **S_func**
- **update_c**
- **update_mu**

---

### cell_type_biase

**Biase Data Cell Type**

**Description**

The cell type of each of the cells in the Biase data.

**Usage**

```r
cell_type_biase
```

**Format**

An R.Data object containing a vector with the cell type of each of the cells in the Biase Data.

### condition_biase

**Biase Data Conditions**

**Description**

The condition for each sample in the Biase data. To be used when splitting the data to demonstrate SparseDC.

**Usage**

```r
condition_biase
```

**Format**

An R.Data object containing a vector with the condition of the 49 cells in the Biase data.
**Description**

This dataset was created by Biase et al. to study cell fat inclination in mouse embryos. It contains FPKM gene expression measurements for 49 cells and 16,514 genes. There are three cell types in the dataset, zygote, two-cell embryo, and four-cell embryo cells.

**Usage**

data_biase

**Format**

An R.Data object storing FPKM gene expression measurements for each of the samples.

---

**generate_uni_dat**

Uniform data generator For use with the gap statistic. Generates datasets drawn from the reference distribution where each reference feature is generated uniformly over the range of observed values for that feature.

---

**Description**

Uniform data generator For use with the gap statistic. Generates datasets drawn from the reference distribution where each reference feature is generated uniformly over the range of observed values for that feature.

**Usage**

generate_uni_dat(data)

**Arguments**

data A dataset with rows as features and columns as samples.

**Value**

A dataset drawn from the reference distribution for use internally with the gap statistic.
Lambda 1 Calculator.

**Description**
Calculates the lambda 1 value for the sparseDC algorithm. The lambda 1 value controls the number of marker genes selected for each cluster in the output from SparseDC. It is calculated as the value of lambda 1 that results in no marker genes being selected when there are no meaningful clusters present in the data. Please see the original manuscript for further details.

**Usage**

```r
lambda1_calculator(pdat1, pdat2, ncluster, alpha1 = 0.5, nboot1 = 1000)
```

**Arguments**
- `pdat1`: The centered data from condition 1, columns should be samples (cells) and rows should be features (genes).
- `pdat2`: The centered data from condition 2, columns should be samples (cells) and rows should be features (genes). The number of genes should be the same as in `pdat1`.
- `ncluster`: The number of clusters present in the data.
- `alpha1`: The quantile of the bootstrapped lambda 1 values to use, range is (0,1). The default value is 0.5, the median of the calculated lambda 1 values.
- `nboot1`: The number of bootstrap repetitions used for estimating lambda 1, the default value is 1000.

**Value**
The calculated value of lambda 1 to use in the main SparseDC algorithm.

**See Also**
- `lambda2_calculator` for how to calculate the lambda 2 parameter.
- `sparsedc_cluster` for the main sparse differential clustering function.

**Examples**
```
set.seed(10)
# Select small dataset for example
data_test <- data_biase[1:1000,]
# Split data into conditions A and B
data_A <- data_test[, which(condition_biase == "A")]
data_B <- data_test[, which(condition_biase == "B")]
# Pre-process the data
pre_data <- pre_proc_data(data_A, data_B, norm = FALSE, log = TRUE, center = TRUE)
```
# lambda2_calculator

lambda2_calculator

# Calculate the lambda 1 value
lambda1_calculator(pdat1 = pre_data[[1]], pdat2 = pre_data[[2]], ncluster=3,
                   alpha1 = 0.5, nboot1 = 1000)

# Can also run

# Pre-process the data
pre_data <- pre_proc_data(data_A, data_B, norm = FALSE, log = TRUE,
                           center = TRUE)
pdata_A <- pre_data[[1]]
pdata_B <- pre_data[[2]]

# Calculate the lambda 1 value
lambda1_calculator(pdat1 = pdata_A, pdat2 = pdata_B, ncluster=3,
                    alpha1 = 0.5, nboot1 = 1000)

---

**lambda2_calculator**  
*Lambda 2 Calculator.*

**Description**

Calculates the lambda 2 values for use in the main SparseDC algorithm, the lambda 2 value controls the number of genes that show condition-dependent expression within each cell type. That is it controls the number of different mean values across the conditions for each cluster. It is calculated by estimating the value of lambda 2 that would result in no difference in mean values across conditions when there are no meaningful differences across between the conditions. For further details please see the original manuscript.

**Usage**

lambda2_calculator(pdat1, pdat2, ncluster, alpha2 = 0.5, nboot2 = 1000)

**Arguments**

- **pdat1**  
The centered data from condition 1, columns should be samples (cells) and rows should be features (genes).

- **pdat2**  
The centered data from condition 2, columns should be samples (cells) and rows should be features (genes). The number of genes should be the same as in pdat1.

- **ncluster**  
The number of clusters present in the data.

- **alpha2**  
The quantile of the bootstrapped lambda 2 values to use, range is (0,1). The default value is 0.5, the median of the calculated lambda 2 values.

- **nboot2**  
The number of bootstrap repetitions for estimating lambda 2, the default value is 1000.

**Value**

The calculated value of lambda 2 to use in the main SparseDC algorithm.
pre_proc_data

Pre-process Data

Description

This function pre-process the data so that SparseDC can be applied. SparseDC requires data that have been normalized for sequencing depth, log-transformed and centralized on a gene-by-gene basis. For the sequencing depth normalization we recommend that users use one of the many methods developed for normalizing scRNA-Seq data prior to using SparseDC and so can set norm = FALSE. However, here we normalize the data by dividing by the total number of reads. This function log transforms the data by applying \( \log(x + 1) \) to each of the data sets. By far the most important pre-processing step for SparseDC is the centralization of the data. Having centralized data is a core component of the SparseDC algorithm and is necessary for both accurate clustering of the cells and identifying marker genes. We therefore recommend that all users centralize their data using this function and that only experienced users set center = FALSE.

Usage

pre_proc_data(dat1, dat2, norm = TRUE, log = TRUE, center = TRUE)
**Arguments**

- **dat1**
  The data for the first condition with samples (cells) as columns and features (genes) as rows.

- **dat2**
  The data for the second condition with samples (cells) as columns and features (genes) as rows.

- **norm**
  This parameter controls whether the data is normalized for sequencing depth by dividing each column by the total number of reads for that sample. We recommend that the user use one of the many methods for normalizing scRNA-Seq data and so set this as `FALSE`. The default value is `TRUE`.

- **log**
  This parameter controls whether the data is transformed using $\log(x + 1)$. The default value is `TRUE`.

- **center**
  This parameter controls whether the data is centered on a gene by gene basis. We recommend all users center their data prior to applying SparseDC and only experienced users should set this as `FALSE`. The default value is `TRUE`.

**Value**

This function returns the two pre-processed datasets stored as a list.

**Examples**

```r
set.seed(10)
# Select small dataset for example
data_test <- data_biase[1:100,]
# Split data into condition A and B
data_A <- data_test[ , which(condition_biase == "A")]
data_B <- data_test[ , which(condition_biase == "B")]
# Pre-process the data
pre_data <- pre_proc_data(data_A, data_B, norm = FALSE, log = TRUE, center = TRUE)
# Extract Data
pdata_A <- pre_data[[1]]
pdata_B <- pre_data[[2]]
```

**Description**

Simulates two condition data for a range of conditions depending on the parameters used.

**Usage**

```r
sim_data(genes, cells, sig.genes, sig.genes.s, clus.t1, clus.t2, same.sig = FALSE, u.l = 1, u.h = 2)
```
Arguments

genes The number of genes to be simulated.
cells The number of cells to be simulated per condition.
sig.genes The number of marker genes for each cluster.
sig.genes.s The number of marker genes shared across conditions for each cluster. Should be less than or equal to sig.genes
clus.t1 A vector of clusters present in the first condition. Start at 1, e.g. c(1,2,3,4)
clus.t2 A vector of clusters present in the second condition. Does not have to match clus.t1, e.g c(3,4,5)
same.sig TRUE or FALSE. Should each cluster have a unique set of marker genes. default is FALSE.
u.l Lower bound for the cluster gene means, default is 1.
u.h Upper bound for the cluster gene means, default is 2.

Value

A list containing the two simulated data matrices, dat.1 and dat.2, true clusters for the cells in the first and second conditions, clusters1 and clusters2, a matrix indicating marker genes for the first and second condition, sig.gene.mat.1 and sig.gene.mat.2, the base mean values for each gene gene.means and the cluster specific additions for each gene clus.gene.means

Examples

set.seed(10)
genes <- 1000 # Simulate 1,000 genes
cells <- 100 # Simulate 100 cells per condition
clus.t1 <- c(1,2,3) # Generate 3 clusters present in condition A
clus.t2 <- c(1,2,3) # Generate 3 clusters present in condition B
sig.genes <- 30 # Generate 30 marker genes per cluster
sig.genes.s <- 15 # Let half of the 30 marker genes be shared.
temp_sim_dat <- sim_data(genes, cells, sig.genes, sig.genes.s, clus.t1, clus.t2)

---

sparsedc_cluster Sparse Differential Clustering

Description

The main SparseDC function. This function clusters the samples from the two conditions and links the clusters across the conditions. It also identifies marker genes for each of the clusters. There are three types of marker gene which SparseDC identifies. Please see the original manuscript for further details.
Usage

```r
sparsedc_cluster(pdat1, pdat2, ncluster, lambda1, lambda2, nitter = 20, 
nstarts = 50, init_iter = 5)
```

Arguments

- `pdat1`: The centered data from condition 1, columns should be samples (cells) and rows should be features (genes).
- `pdat2`: The centered data from condition 2, columns should be samples (cells) and rows should be features (genes). The number of genes should be the same as `pdat1` as in `pdat1`.
- `ncluster`: The number of clusters present in the data.
- `lambda1`: The lambda 1 value to use in the SparseDC function. This value controls the number of marker genes detected for each of the clusters in the final result. This can be calculated using the "lambda1_calculator" function or supplied by the user.
- `lambda2`: The lambda 2 value to use in the SparseDC function. This value controls the number of genes that show condition-dependent expression within each cell type. This can be calculated using the "lambda2_calculator" function or supplied by the user.
- `nitter`: The max number of iterations for each of the start values, the default value is 20.
- `nstarts`: The number of start values to use for SparseDC. The default value is 50.
- `init_iter`: The number of iterations used to generate the starting center values.

Value

A list containing the clustering solution, cluster centers and the score of each of the starts.

See Also

- `lambda1_calculator`
- `lambda2_calculator`
- `update_c`
- `update_mu`

Examples

```r
set.seed(10)
# Select small dataset for example
data_test <- data_biase[1:100,]
# Split data into conditions 1 and 2
data_1 <- data_test[, which(condition_biase == "A")]
data_2 <- data_test[, which(condition_biase == "B")]
# Preprocess data (log transform and center)
pre_data <- pre_proc_data(data_1, data_2, norm = FALSE, log = TRUE, 
center = TRUE)
# Calculate lambda 1 parameter
lambda1 <- lambda1_calculator(pdat1 = pre_data[1], pdat2 = pre_data[2], 
ncluster=3, alpha1 = 0.5, nboot1 = 1000)
# Calculate lambda 2 parameter
```
The function `sparsedc_gap` calculates the gap statistic for SparseDC. It is recommended for situations where the number of clusters in the data is unknown. The function uses the gap statistic to infer the optimal number of clusters. The `sparsedc_gap` function takes parameters such as `pdat1`, `pdat2`, `min_clus`, `max_clus`, `nboots`, `nitter`, and `nstarts` to perform the analysis.

**Description**

This function calculates the gap statistic for SparseDC. For use when the number of clusters in the data is unknown. We recommend using alternate methods to infer the number of clusters in the data.

**Usage**

```r
sparsedc_gap(pdat1, pdat2, min_clus, max_clus, nboots = 200, nitter = 20, nstarts = 10, l1_boot = 50, l2_boot = 50)
```
**S_func**

### Arguments

- **pdat1**
The centered data from condition 1, columns should be samples (cells) and rows should be features (genes).

- **pdat2**
The centered data from condition 2, columns should be samples (cells) and rows should be features (genes). The number of genes should be the same as pdat1.

- **min_clus**
The minimum number of clusters to try, minimum value is 2.

- **max_clus**
The maximum number of clusters to try.

- **nboots**
The number of bootstrap repetitions to use, default = 200.

- **nitter**
The max number of iterations for each of the start values, the default value is 20.

- **nstarts**
The number of start values to use for SparseDC. The default value is 10.

- **l1_boot**
The number of bootstrap repetitions used for estimating lambda 1.

- **l2_boot**
The number of bootstrap repetitions used for estimating lambda 2.

### Value

A list containing the optimal number of clusters, as well as gap statistics and the calculated standard error for each number of clusters.

### Examples

```r
# load a small dataset
data_test <- data_biase[1:50,]
# Split data into conditions 1 and 2
data_1 <- data_test[ , which(condition_biase == "A")]
data_2 <- data_test[ , which(condition_biase == "B")]
# Preprocess data (log transform and center)
pre_data <- pre_proc_data(data_1, data_2, norm = FALSE, log = TRUE, center = TRUE)
# Run with one bootstrap sample for example
gap_stat <- sparsedc_gap(pre_data[[1]], pre_data[[2]],
                        min_clus <- 2, max_clus <- 3, nboots <- 2)
```

---

**S_func**

The soft thresholding operator

### Description

Function to solve the soft thresholding problem

### Usage

```r
S_func(x, a)
```
Arguments

- \( x \) The data value
- \( a \) The lambda value

Value

The solution to the soft thresholding operator.

---

**update_c**

*Update Clusters*

**Description**

Updates the cluster membership for each iteration of SparseDC. Runs inside `sparse_dc_fun`.

**Usage**

\[
\text{update_c}(\text{mu}_1, \text{mu}_2, \text{pdat1}, \text{pdat2}, \text{ncluster})
\]

**Arguments**

- \( \text{mu}_1 \) The center values for each cluster in condition 1.
- \( \text{mu}_2 \) The center values for each cluster in condition 2.
- \( \text{pdat1} \) The centered data from condition 1, columns should be samples (cells) and rows should be features (genes).
- \( \text{pdat2} \) The centered data from condition 2, columns should be samples (cells) and rows should be features (genes). The number of genes should be the same as \( \text{pdat1} \) as in \( \text{pdat1} \).
- \( \text{ncluster} \) The number of clusters present in the data.

**Value**

A list containing the cluster membership for condition 1 and condition 2.
update_mu

**Update the Center Values**

**Description**

This function updates the center values for each cluster for each iteration of SparseDC. This function runs inside `sparse_dc_fun`.

**Usage**

```r
update_mu(c_1, c_2, pdat1, pdat2, ncluster, lambda1, lambda2)
```

**Arguments**

- `c_1`: The cluster membership for samples in condition 1.
- `c_2`: The cluster membership for samples in condition 2.
- `pdat1`: The centered data from condition 1, columns should be samples (cells) and rows should be features (genes).
- `pdat2`: The centered data from condition 2, columns should be samples (cells) and rows should be features (genes). The number of genes should be the same as in `pdat1`.
- `ncluster`: The number of clusters present in the data.
- `lambda1`: The lambda 1 value to use in the SparseDC function. This value controls the number of marker genes detected for each of the clusters in the final result. This can be calculated using the "lambda1_calculator" function or supplied by the user.
- `lambda2`: The lambda 2 value to use in the SparseDC function. This value controls the number of genes that show condition-dependent expression within each cell type. This can be calculated using the "lambda2_calculator" function or supplied by the user.

**Value**

Returns a list containing the center values for each of the clusters in condition 1 and 2.
Index

*Topic datasets
  cell_type_bias, 2
  condition_bias, 2
  data_bias, 3

  cell_type_bias, 2
  condition_bias, 2
  data_bias, 3

generate_uni_dat, 3

lambda1_calculator, 4, 6, 9
lambda2_calculator, 4, 5, 9

pre_proc_data, 6

S_func, 11
sim_data, 7
sparsedc_cluster, 4, 6, 8
sparsedc_gap, 10

update_c, 9, 12
update_mu, 9, 13