# Package 'SparseMDC'

August 2, 2018

**Type** Package

**Title** Implementation of SparseMDC Algorithm

**Version** 0.99.5

**Description** Implements the algorithm described in
Barron, M., and Li, J. (Not yet published). This algorithm clusters samples from multiple ordered 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 SparseMDC as outlined in the paper. We recommend that users further select their marker genes using the magnitude of the cluster centers.

**License** GPL-3

**Imports** stats, foreach, parallel, doParallel

**Depends** R (>= 3.1.0), doRNG

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.0

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

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

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

**Repository** CRAN

**Date/Publication** 2018-08-02 12:40:16 UTC

## R topics documented:

---

`cell_type_biase` *Biase Data Cell Type*

---

## Description

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

## Usage

```
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

```
condition_biase
```

## Format

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

## data_biase *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.

`lambda1_calculator` *Lambda 1 Calcualtor*

## Description

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

## Usage

```
lambda1_calculator(dat_l, dim, nclust, nboot = 1000, alpha1 = 0.5,
  delta = 1e-07)
```

## Arguments

| | |
|---|---|
| `dat_l` | list with D entries, each entry contains data d, p * n matrix. This data should be centered and log-transformed. |
| `dim` | Total number of conditions, D. |
| `nclust` | Total number of clusters. |
| `nboot` | The number of bootstrap repetitions used for estimating lambda 1, the default value is 1000. |
| `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. |
| `delta` | Small value term added to ensure existance, default value is 0.0000001. |

## Value

The estimated value of lambda1 for use in main SparseMDC algorithm

## Examples

```
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")]
data_C <- data_test[ , which(condition_biase == "C")]
# Store data as list
dat_l <- list(data_A, data_B, data_C)
# Pre-process the data
pdat <- pre_proc_data(dat_l, dim=3, norm = FALSE, log = TRUE,
center = TRUE)
lambda1 <- lambda1_calculator(pdat, dim = 3, nclust = 3 )
```

lambda2_calculator *Lambda 2 Calculator*

## Description

Calculates the lambda 2 values for use in the main SparseMDC 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(dat_l, dim, nclust, nboot = 1000, alpha2 = 0.5,
  delta = 1e-07, lambda1)
```

## Arguments

| | |
|---|---|
| dat_l | list with D entries, each entry contains data d, p * n matrix. This data should be centered and log-transformed. |
| dim | Total number of conditions, D. |
| nclust | Total number of clusters. |
| nboot | The number of bootstrap repetitions for estimating lambda 2, the default value is 1000. |
| 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. |
| delta | Small term to ensure existance of solution, default is 0.0000001. |
| lambda1 | Calcualted penalty parameter for mean size. |

## Value

The estimated value of lambda2

## Examples

```
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")]
data_C <- data_test[ , which(condition_biase == "C")]
# Store data as list
dat_l <- list(data_A, data_B, data_C)
# Pre-process the data
```

```
pdat <- pre_proc_data(dat_l, dim=3, norm = FALSE, log = TRUE,
center = TRUE)
lambda1 <- lambda1_calculator(pdat, dim = 3, nclust = 3)
lambda2 <- lambda2_calculator(pdat, dim = 3, nclust = 3, lambda1 = lambda1)
```

---

`mu_calc` *mu Calculator*

---

## Description

This function handles the calculations of the Mu Solver. This function runs inside `sparse_mdc`.

## Usage

```
mu_calc(d, v, EQ, dim, x, nk, p1, p2, delta)
```

## Arguments

| | |
|---|---|
| `d` | - The current condition. |
| `v` | - Relationship matrix, describing relationship between d-1 and d. |
| `EQ` | - Equality matrix specifying the number of following terms to which mu_d is equal. |
| `dim` | Total number of conditions, D. |
| `x` | - Matrix of mean values. |
| `nk` | - Vector with the number of samples in each dimension for this cluster. |
| `p1` | - Penalties on mean size. |
| `p2` | - Penalties on mean difference. |
| `delta` | Small term to ensure existance of solution. |

## Value

A list containing two vectors containing the calculated values of mu_d | mu_d < mu_d-1 and mu_d | mu_d > mu_d-1 respecitively.

## mu_solver *Mu Solver*

### Description

Calculates the regularized center values for a cluster. This function runs inside `sparse_mdc`.

### Usage

```
mu_solver(d, mu, v, EQ, dim, x, nk, p1, p2, delta)
```

### Arguments

| | |
|---|---|
| `d` | - The current condition. |
| `mu` | - A matrix of regularized mean values. |
| `v` | - Relationship matrix, describing relationship between d-1 and d. |
| `EQ` | - Equality matrix specifying the number of following terms to which mu_d is equal. |
| `dim` | - Total dimensions. |
| `x` | - Matrix of mean values. |
| `nk` | - Vector with the number of samples in each condition for this cluster. |
| `p1` | - Penalties on mean size. |
| `p2` | - Penalties on mean difference. |
| `delta` | Small term to ensure existance of solution. |

### Value

A matrix containing the regularized cluster means for each dimension.

## pen_calculator *Penalty calculator*

### Description

Calcualtes the penalty terms for penalizing mean size and mean difference. This function runs inside `sparse_mdc`.

### Usage

```
pen_calculator(lambda1, lambda2, nk, delta)
```

### Arguments

| | |
|---|---|
| lambda1 | Calculated penalty parameter for mean size. |
| lambda2 | Calculated penalty parameter for mean difference. |
| nk | Vector containing the number of samples in each dimension. |
| delta | Small term to ensure existance of solution, default is 0.0000001. |

### Value

a list with two vectors containing the penalty terms for mean size and mean difference respectively.

---

## pre_proc_data *Pre-process data*

---

### Description

This function centers on a gene-by-gene basis, normalizes and/or log transforms the data prior to the application of SparseMDC.For the sequencing depth normalization we recommend that users use one of the many methods developed for normalizing scRNA-Seq data prior to using SparseMDC 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 SparseMDC is the centralization of the data. Having centralized data is a core component of the SparseMDC 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(dat_l, dim, norm = FALSE, log = TRUE, center = TRUE)
```

### Arguments

| | |
|---|---|
| dat_l | list with D entries, each entry contains data d, p * n matrix. The entries should be ordered according the condition of each dataset. The rows of the data matrix should contain samples while the columns contain features or genes. |
| dim | Total number of conditions, D. |
| norm | True/False on if the data should be normalized. 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 user use one of the many methods for normalizing scRNA-Seq data and so set this as FALSE. The default value is FALSE |
| log | True/False of if the data should be transformed as 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 SparseMDC and only experienced users should set this as FALSE. The default value is TRUE. |

## Value

A list with D entries containing the pre-processed data.

## Examples

```
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")]
data_C <- data_test[ , which(condition_biase == "C")]
# Store data as list
dat_l <- list(data_A, data_B, data_C)
# Pre-process the data
pdat <- pre_proc_data(dat_l, dim=3, norm = FALSE, log = TRUE,
center = TRUE)
```

---

# score_calc *Score calculator*

---

## Description

Calculates the score for each combination of cluster assignments and center values. This function runs inside `sparse_mdc`.

## Usage

```
score_calc(pdat, clusters, mu, lambda1, lambda2, nclust, delta, dim)
```

## Arguments

| | |
|---|---|
| `pdat` | list with D entries, each entry contains data d, p * n matrix. This data should be centered and log-transformed. |
| `clusters` | List containig cluster assignments for each dimension as entries. |
| `mu` | list with D entries, each entry contains centers for data d, p*k matrix. |
| `lambda1` | Penalty parameter for mean size. |
| `lambda2` | Penalty parameter for mean difference. |
| `nclust` | Number of clusters in the data. |
| `delta` | Small term to ensure existance of solution, default is 0.0000001. |
| `dim` | Total number of conditions, D. |

## Value

The caluculated score.

sdc_mpar *SparseDC Multi Parallel*

## Description

Applies sparse clustering to data from multiple conditions, linking the clusters across conditions and selecting a set of marker variables for each cluster and condition. This is a wrapper function to run SparseMDC in parallel and choose the solution with the mimimum score for each run.

## Usage

```
sdc_mpar(pdat, nclust, dim, lambda1, lambda2, nitter = 20,
  nstarts = 50, init_iter = 5, delta = 1e-07, par_starts, cores)
```

## Arguments

| | |
|---|---|
| pdat | list with D entries, each entry contains data d, p * n matrix. This data should be centered and log-transformed. |
| nclust | Number of clusters in the data. |
| dim | Total number of conditions, D. |
| lambda1 | The lambda 1 value to use in the SparseMDC 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 SparseMDC 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 max number of possible starts. The default value is 50. |
| init_iter | The number of iterations used to initialize the algorithm. Higher values result in less starts but more accurate and vice versa. Default is 5. |
| delta | Small term to ensure existance of solution, default is 0.0000001. |
| par_starts | Number of parallel starts. |
| cores | Number of cores to use. |

## Value

A list containing cluster assignments, center values and the scores for each start.

## Examples

```
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")]
data_C <- data_test[ , which(condition_biase == "C")]
# Store data as list
dat_l <- list(data_A, data_B, data_C)
# Pre-process the data
pdat <- pre_proc_data(dat_l, dim=3, norm = FALSE, log = TRUE,
center = TRUE)
# Calculate lambda1
lambda1 <- lambda1_calculator(pdat, dim = 3, nclust = 3)
# Calcualte lambda2
lambda2 <- lambda2_calculator(pdat, dim = 3, nclust = 3, lambda1 = lambda1)
# Prepare parallel enviornment
library(doParallel) # Load package
library(foreach)  # Load the package
library(doRNG)
# Apply SparseMDC
smdc_res <- sdc_mpar(pdat, nclust = 3, dim = 3, lambda1 = lambda1,
lambda2 = lambda2, par_starts = 2, cores = 2)
```

---

# sparsemdc_gap *Gap Statistic Calculator*

---

## Description

This function calculates the gap statistic for SparseMDC. 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

```
sparsemdc_gap(pdat, dim, min_clus, max_clus, nboots = 200, nitter = 20,
  nstarts = 10, l1_boot = 50, l2_boot = 50)
```

## Arguments

| | |
|---|---|
| pdat | list with D entries, each entry contains data d, p * n matrix. This data should be centered and log-transformed. |
| dim | Total number of conditions, D. |
| 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

```
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")]
data_C <- data_test[ , which(condition_biase == "C")]
# Store data as list
dat_l <- list(data_A, data_B, data_C)
# Pre-process the data
pdat <- pre_proc_data(dat_l, dim=3, norm = FALSE, log = TRUE,
center = TRUE)
# Run with one bootstrap sample for example
gap_stat <- sparsemdc_gap(pdat, dim=3, min_clus = 2, max_clus =3, nboots =2,
nitter = 2, nstarts = 1, l1_boot = 5, l2_boot = 5)
```

---

# sparse_mdc *SparseDC Multi*

---

## Description

Applies sparse clustering to data from multiple conditions, linking the clusters across conditions and selecting a set of marker variables for each cluster and condition. See the manuscript for descriptions of the different categories of marker genes.

## Usage

```
sparse_mdc(pdat, nclust, dim, lambda1, lambda2, nitter = 20,
  nstarts = 50, init_iter = 5, delta = 1e-07)
```

## Arguments

| | |
|---|---|
| pdat | list with D entries, each entry contains data d, p * n matrix. This data should be centered and log-transformed. |
| nclust | Number of clusters in the data. |
| dim | Total number of conditions, D. |
| lambda1 | The lambda 1 value to use in the SparseMDC 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 SparseMDC 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 max number of possible starts. The default value is 50. |
| init_iter | The number of iterations used to initialize the algorithm. Higher values result in less starts but more accurate and vice versa. Default is 5. |
| delta | Small term to ensure existance of solution, default is 0.0000001. |

## Value

A list containing cluster assignments, center values and the scores for each start.

## Examples

```
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")]
data_C <- data_test[ , which(condition_biase == "C")]
# Store data as list
dat_l <- list(data_A, data_B, data_C)
# Pre-process the data
pdat <- pre_proc_data(dat_l, dim=3, norm = FALSE, log = TRUE,
center = TRUE)
lambda1 <- lambda1_calculator(pdat, dim = 3, nclust = 3)
lambda2 <- lambda2_calculator(pdat, dim = 3, nclust = 3, lambda1 = lambda1)
smdc_res <- sparse_mdc(pdat, nclust = 3, dim = 3, lambda1 = lambda1,
lambda2 = lambda2)
```

## S_func *The soft thresholding operator*

### Description

Function to solve the soft thresholding problem

### Usage

```
S_func(x, a)
```

### Arguments

| | |
|---|---|
| x | The data value |
| a | The lambda value |

### Value

The solution to the soft thresholding operator.

## update_c *update_c*

### Description

Assigns samples to clusters. This function runs inside sparse_mdc.

### Usage

```
update_c(mu, pdat, nclust, dim, lambda1, lambda2, delta)
```

### Arguments

| | |
|---|---|
| mu | list with D entries, each entry contains centers for data d, p*k matrix. |
| pdat | list with D entries, each entry contains data d, p * n matrix. This data should be centered and log-transformed. |
| nclust | Total number of clusters. |
| dim | Total number of conditions, D. |
| lambda1 | Calculated penalty parameter for mean size. |
| lambda2 | Calculated penalty parameter for mean difference. |
| delta | Small term to ensure existance of solution. |

### Value

A list with D entries containing cluster assignments for each sample.

update_mu *update_mu*

## Description

Update the mean/center values for each cluster and dimension. This function runs inside `sparse_mdc`.

## Usage

```
update_mu(clusters, pdat, nclust, dim, lambda1, lambda2, ngenes, delta)
```

## Arguments

| | |
|---|---|
| `clusters` | List containig cluster assignments for each dimension as entries. |
| `pdat` | list with D entries, each entry contains data d, p * n matrix. This data should be centered and log-transformed. |
| `nclust` | Number of clusters in the data. |
| `dim` | Total number of conditions, D. |
| `lambda1` | Calculated penalty parameter for mean size. |
| `lambda2` | Calculated penalty parameter for mean difference. |
| `ngenes` | The number of genes in the data. |
| `delta` | Small term to ensure existance of solution. |

## Value

A list containing the center values for the clusters in each dimensions as entries.

# Index