Package ‘rescue’

October 14, 2022

Type Package

Title Bootstrap Imputation for Single-Cell RNA-Seq Data

Version 1.0.3

Date 2020-07-15

Author Sam Tracy & Ruben Dries

Maintainer Sam Tracy <seasamgo@gmail.com>

Description Given a log-transformed expression matrix and list of informative genes:
subsample informative genes, cluster samples using shared nearest neighbors clustering,
estimate missing expression values with the distribution mean of means extrapolated
from these cell clusterings, and return an imputed expression matrix. See Tracy, S.,

Config/reticulate list( packages = list( list(package = "pandas"),
list(package = "networkx"), list(package = "python-louvain") )
)

Depends R (>= 3.4.0), utils

Imports data.table, dbscan (>= 1.1-3), igraph (>= 1.2.4.1), irlba,
Matrix, methods, parallel, reticulate (>= 1.14)

License GPL-2 | GPL-3

LazyData FALSE

URL https://github.com/seasamgo/rescue

BugReports http://github.com/seasamgo/rescue/issues

RoxygenNote 7.1.1

Encoding UTF-8

Suggests knitr, rmarkdown

NeedsCompilation no

Repository CRAN

Date/Publication 2020-07-18 10:00:03 UTC
R topics documented:

- bootstrapImputation .................................................. 2
- clusterLouvain ......................................................... 4
- computeHVG ............................................................. 5
- constructNN ............................................................. 6
- sampleImputation ....................................................... 7

Description

Subsample informative genes, cluster cells using SNN, estimate missing expression values with the distribution mean of means extrapolated from these cell clusterings

Usage

```r
bootstrapImputation(
  expression_matrix,
  select_cells = NULL,
  select_genes = NULL,
  log_transformed = TRUE,
  log_base = exp(1),
  proportion_genes = 0.6,
  bootstrap_samples = 100,
  number_pcs = 8,
  k_neighbors = 30,
  snn_resolution = 0.9,
  impute_index = NULL,
  use_mclapply = FALSE,
  cores = 2,
  return_individual_results = FALSE,
  python_path = NULL,
  verbose = FALSE
)
```

Arguments

- `expression_matrix`  
  Row by column log-normalized expression matrix
- `select_cells`  
  Subset cells if desired
- `select_genes`  
  A vector of highly variable of differentially expressed gene names, defaults to the most variable
- `log_transformed`  
  Whether the expression matrix has been log-transformed
**bootstrapImputation**

log_base If log-transformed, log-base used
proportion_genes Proportion of informative genes to sample
bootstrap_samples Number of samples for the bootstrap
number_pcs Number of dimensions to inform SNN clustering
k_neighbors Number of k neighbors to use for NN network
snn_resolution Resolution parameter for SNN
impute_index Index to impute, will default to all zeroes
use_mclapply Run in parallel, default FALSE
cores Number of cores for parallelization
return_individual_results Return a list of subsampled means
python_path path to your python binary (default = system path)
verbose Print progress output to the console

**Value**

Returns a list with the imputed and original expression matrices

**Examples**

```r
set.seed(0)
requireNamespace("Matrix")

## generate (meaningless) counts
c1 <- stats::rpois(5e3, 1)
c2 <- stats::rpois(5e3, 2)
m <- t(
  rbind(
    matrix(c1, nrow = 20),
    matrix(c2, nrow = 20)
  )
)

## construct an expression matrix m
colnames(m) <- paste0('cell', 1:ncol(m))
rownames(m) <- paste0('gene', 1:nrow(m))
m <- log(m/colSums(m)*1e4 + 1)
m <- methods::as(m, 'dgCMatrix')

## impute
m_imputed <- rescue::bootstrapImputation(
  expression_matrix = m,
  proportion_genes = .9,
  bootstrap_samples = 2,
  k_neighbors = 10
)`
clusterLouvain

Cluster Cells via Louvain Algorithm

Description
Cluster cells using a NN-network and the Louvain algorithm from the community module in Python

Usage
clusterLouvain(
    nn_network,
    python_path = NULL,
    resolution = 1,
    weight_col = NULL,
    louv_random = F,
    set_seed = T,
    seed_number = 0,
    ...
)

Arguments

- **nn_network**: Constructed nearest neighbor network to use
- **python_path**: Specify specific path to python if required
- **resolution**: Resolution
- **weight_col**: Weight column
- **louv_random**: Random
- **set_seed**: Set seed
- **seed_number**: Number for seed
- **...**: Additional parameters

Value
A character vector of cluster labels
**computeHVG**

**Compute Highly Variable Genes**

**Description**
Compute Highly Variable Genes

**Usage**

```r
computeHVG(
  expression_matrix,
  reverse_log_scale = T,
  log_base = exp(1),
  expression_threshold = 0,
  nr_expression_groups = 20,
  zscore_threshold = 1.5
)
```

**Arguments**

- `expression_matrix`:
  Expression matrix
- `reverse_log_scale`:
  Reverse log-scale of expression values
- `log_base`:
  If reverse_log_scale is TRUE, which log base was used?
- `expression_threshold`:
  Expression threshold to consider a gene detected
- `nr_expression_groups`:
  Number of expression groups for cov_groups
- `zscore_threshold`:
  $Z$-score to select hvg for cov_groups

**Value**
Character vector of highly variable genes

**Examples**

```r
set.seed(0)
requireNamespace("Matrix")

## generate (meaningless) counts
cl <- stats::rpois(5e3, 1)
c2 <- stats::rpois(5e3, 2)
m <- t(
  rbind(
    matrix(cl, nrow = 20),
    matrix(c2, nrow = 20)
  )
)
```r
matrix(c2, nrow = 20)

## construct an expression matrix m
colnames(m) <- paste0('cell', 1:ncol(m))
rownames(m) <- paste0('gene', 1:nrow(m))
m <- log(m/colSums(m)*1e4 + 1)
m <- methods::as(m, 'dgCMatrix')

## calculate HVGs
hvgs <- computeHVG(m)
```

---

**constructNN**

**Nearest Network**

**Description**

Construct a nearest neighbour network based on previously computed PCs

**Usage**

```r
constructNN(
  reduced_object,
  k_neighbors = 30,
  minimum_shared = 5,
  top_shared = 3,
  verbose = F,
  ...
)
```

**Arguments**

- `reduced_object`: PC reduction matrix
- `k_neighbors`: Number of k neighbors to use
- `minimum_shared`: Minimum shared neighbors
- `top_shared`: Keep at...
- `verbose`: Be verbose
- `...`: Additional parameters

**Value**

NN network as igraph object
sampleImputation  

**Sample-mean Estimation**

**Description**

Cluster cells using SNN and a list of given genes, estimate missing expression values for each cell-gene combination with the within-cluster non-zero expression mean.

**Usage**

```r
sampleImputation(
  expression_matrix,
  subset_genes = NULL,
  scale_data = TRUE,
  number_pcs = 8,
  k_neighbors = 30,
  snn_resolution = 0.9,
  impute_index = NULL,
  pseudo_zero = NULL,
  python_path = NULL,
  verbose = FALSE
)
```

**Arguments**

- `expression_matrix`  
  Row by column log-normalized expression matrix
- `subset_genes`  
  A vector of informative gene names, defaults to all genes
- `scale_data`  
  Whether to standardize expression by gene, default TRUE
- `number_pcs`  
  Number of dimensions to inform SNN clustering
- `k_neighbors`  
  Number of k neighbors to use for NN network
- `snn_resolution`  
  Resolution parameter for SNN
- `impute_index`  
  Index to impute, will default to all zeroes
- `pseudo_zero`  
  Pseudo-zero expression value
- `python_path`  
  path to your python binary (default = system path)
- `verbose`  
  Print progress output to the console

**Value**

Returns a sparse matrix of class ’dgCMatrix’
Examples

```r
set.seed(0)
requireNamespace("Matrix")

## generate (meaningless) counts
c1 <- stats::rpois(5e3, 1)
c2 <- stats::rpois(5e3, 2)
m <- t(
  rbind(
    matrix(c1, nrow = 20),
    matrix(c2, nrow = 20)
  )
)

## construct an expression matrix m
colnames(m) <- paste0('cell', 1:ncol(m))
rownames(m) <- paste0('gene', 1:nrow(m))
m <- log(m/colSums(m)*1e4 + 1)
m <- methods::as(m, 'dgCMatrix')

## impute
m_imputed <- rescue::sampleImputation(
  expression_matrix = m,
  k_neighbors = 10
)
```

Index

bootstrapImputation, 2
clusterLouvain, 4
computeHVG, 5
constructNN, 6
sampleImputation, 7