Package: ccImpute (via r-universe)

November 1, 2026

Type Package

Title ccImpute: an accurate and scalable consensus clustering based approach to impute dropout events in the single-cell RNA-seq data (https://doi.org/10.1186/s12859-022-04814-8)

Version 1.14.0

Description Dropout events make the lowly expressed genes indistinguishable from true zero expression and different than the low expression present in cells of the same type. This issue makes any subsequent downstream analysis difficult. ccImpute is an imputation algorithm that uses cell similarity established by consensus clustering to impute the most probable dropout events in the scRNA-seq datasets. ccImpute demonstrated performance which exceeds the performance of existing imputation approaches while introducing the least amount of new noise as measured by clustering performance characteristics on datasets with known cell identities.

License GPL-3

Imports Rcpp, sparseMatrixStats, stats, BiocParallel, irlba, SingleCellExperiment, Matrix, SummarizedExperiment

LinkingTo Rcpp, RcppEigen

Encoding UTF-8

LazyData FALSE

BugReports https://github.com/khazum/ccImpute/issues

URL https://github.com/khazum/ccImpute/

RoxygenNote 7.3.2

biocViews SingleCell, Sequencing, PrincipalComponent, DimensionReduction, Clustering, RNASeq, Transcriptomics

biocType Software

Suggests knitr, rmarkdown, BiocStyle, sessioninfo, scRNAseq, scater, mclust, testthat (>= 3.0.0), splatter

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:59:18 UTC

RemoteUrl https://github.com/bioc/ccImpute

RemoteRef RELEASE_3_23

RemoteSha b408f4545cd3b1f4aade299c5a5499f1031d64d0

Contents

ccImpute . . . 2
colRanks_fast . . . 5
computeDropouts . . . 5
cor_fast . . . 6
doSVD . . . 7
estkTW . . . 8
findDropouts . . . 8
getConsMtx . . . 9
getCorM . . . 10
getScale . . . 11
rowVars_fast . . . 11
runKM . . . 12
solver . . . 13
solver2 . . . 14
sparseColRanks_fast . . . 14
sparseSolver2 . . . 15
wCor_fast . . . 15

Index **17**

`ccImpute` *Impute Dropout Values in Single-Cell RNA Sequencing Data*

Description

Performs imputation of dropout values in single-cell RNA sequencing (scRNA-seq) data using a consensus clustering-based algorithm (ccImpute). This implementation includes performance enhancements over the original ccImpute method described in the paper "ccImpute: an accurate and scalable consensus clustering based algorithm to impute dropout events in the single-cell RNA-seq data" (DOI: https://doi.org/10.1186/s12859-022-04814-8).

Defines the generic function ‘ccImpute‘ and a specific method for ‘SingleCellExperiment‘ objects.

Usage

```
ccImpute.SingleCellExperiment(
  object,
  dist,
  nCeil = 2000,
  svdMaxRatio = 0.08,
  maxSets = 8,
  k,
  consMin = 0.75,
  kmNStart,
  kmMax = 1000,
  fastSolver = TRUE,
  BPPARAM = bpparam(),
  verbose = TRUE
)

ccImpute(
  object,
  dist,
  nCeil = 2000,
  svdMaxRatio = 0.08,
  maxSets = 8,
  k,
  consMin = 0.75,
  kmNStart,
  kmMax = 1000,
  fastSolver = TRUE,
  BPPARAM = bpparam(),
  verbose = TRUE
)

## S4 method for signature 'SingleCellExperiment'
ccImpute(
  object,
  dist,
  nCeil = 2000,
  svdMaxRatio = 0.08,
  maxSets = 8,
  k,
  consMin = 0.75,
  kmNStart,
  kmMax = 1000,
  fastSolver = TRUE,
  BPPARAM = bpparam(),
  verbose = TRUE
)
```

Arguments

object	A SingleCellExperiment class object containing the scRNA-seq data. The logcounts assay should contain matrix with log-normalized expression values. This code supports both dense and sparse (dgCMatrix) matrix format storage.
dist	(Optional) A distance matrix used for cell similarity. calculations. If not provided, a weighted Spearman correlation matrix is calculated.
nCeil	(Optional) The maximum number of cells used to compute the proportion of singular vectors (default: 2000).
svdMaxRatio	(Optional) The maximum proportion of singular vectors used for generating subsets (default: 0.08).
maxSets	(Optional) The maximum number of sub-datasets used for consensus clustering (default: 8).
k	(Optional) The number of clusters (cell groups) in the data. If not provided, it is estimated using the Tracy-Widom Bound.
consMin	(Optional) The low-pass filter threshold for processing the consensus matrix (default: 0.75).
kmNStart	nstart parameter passed to kmeans. function. Can be set manually. By default it is 1000 for up to 2000 cells and 50 for more than 2000 cells.
kmMax	iter.max parameter passed to kmeans.
fastSolver	(Optional) Whether to use mean of non-zero values for calculating dropout values or a linear equation solver (much slower and did show empirical difference in imputation performance) (default: TRUE).
BPPARAM	(Optional) A BiocParallelParam object for parallel processing (default: bpparam()).
verbose	(Optional) Whether to print progress messages (default: TRUE).

Value

A SingleCellExperiment class object with the imputed expression values stored in the ‘"imputed"‘ assay.

Examples

```
library(BiocParallel)
library(splatter)
library(scater)
sce <- splatSimulate(group.prob = rep(1, 5)/5, sparsify = FALSE,
        batchCells=100, nGenes=1000, method = "groups", verbose = FALSE,
        dropout.type = "experiment")
sce <- logNormCounts(sce)
cores <- 2
BPPARAM = MulticoreParam(cores)
sce <- ccImpute(sce, BPPARAM=BPPARAM)
```

colRanks_fast *Computes rankings for each column of a matrix in parallel.*

Description

Computes rankings for each column of a matrix in parallel.

Usage

```
colRanks_fast(x, n_cores)
```

Arguments

x	The input matrix to be ranked.
n_cores	The number of CPU cores to use for parallel processing.

Value

A matrix where each column contains the rankings for the corresponding column in the input matrix.

computeDropouts *Impute Dropout Values in a Log-normalized Expression Count Matrix*

Description

This function imputes dropout values (zeros) in a count matrix using either a fast numerical solver or a slower linear equations solver.

Usage

```
computeDropouts(consMtx, logX, dropIds, fastSolver = TRUE, nCores)
```

Arguments

consMtx	A numeric matrix representing the processed consensus matrix obtained from clustering analysis.
logX	A (sparse or dense) numeric matrix representing the transpose of a log-normalized gene expression matrix. Rows correspond to cells, and columns correspond to genes.
dropIds	A numeric vector containing the row/col indices of the dropouts to be imputed.
fastSolver	A logical value indicating whether to use the fast solver (default) or the slow solver.
nCores	An integer specifying the number of cores to use for parallel processing (if applicable).

Value

An imputed log-transformed count matrix (same dimensions as ‘logX‘).

Examples

```
library(scater)
library(BiocParallel)
library(splatter)

sce <- splatSimulate(group.prob = rep(1, 5)/5, sparsify = FALSE,
        batchCells=100, nGenes=1000, method = "groups", verbose = FALSE,
        dropout.type = "experiment")
sce <- logNormCounts(sce)
cores <- 2
logX <- as.matrix(logcounts(sce))
w <- rowVars_fast(logX, cores)
corMat <- getCorM("spearman", logcounts(sce), w, cores)
v <- doSVD(corMat, nCores=cores)
BPPARAM = MulticoreParam(cores)
consMtx <- runKM(logX, v, BPPARAM=bpparam())
dropIds <- findDropouts(logX, consMtx)
impLogX <- computeDropouts(consMtx, logX, dropIds, nCores=cores)
```

---

cor_fast *Computes a Pearson*

Description

This function calculates a Pearson correlation matrix

Usage

```
cor_fast(x, n_cores)
```

Arguments

x	The input matrix, where each column represents a set of observations.
n_cores	The number of CPU cores to utilize for parallel computation (optional, defaults to 1).

Value

A Pearson correlation matrix if ‘useRanks‘ is ‘false‘. If ‘useRanks‘ is ‘true‘, returns a Spearman correlation matrix.

doSVD *Perform Truncated Singular Value Decomposition (SVD)*

Description

Computes a truncated SVD on a matrix using the implicitly restarted Lanczos bidiagonalization algorithm (IRLBA).

Usage

```
doSVD(x, svdMaxRatio = 0.08, nCeil = 2000, nCores)
```

Arguments

x	A numeric matrix to perform SVD on.
svdMaxRatio	(Optional) The maximum proportion of singular vectors used for generating subsets (default: 0.08).
nCeil	(Optional) The maximum number of cells used to compute the proportion of singular vectors (default: 2000).
nCores	The number of cores to use for parallel processing.

Details

This function utilizes the ‘irlba‘ function from the ‘irlba‘ package to efficiently calculate the truncated SVD of the input matrix ‘x‘. The returned matrix contains ‘nv‘ right singular vectors, which are often used for dimensionality reduction and feature extraction in various applications.

Value

A matrix containing the right singular vectors of ‘x‘.

Examples

```
library(scater)
library(splatter)

sce <- splatSimulate(group.prob = rep(1, 5)/5, sparsify = FALSE,
        batchCells=100, nGenes=1000, method = "groups", verbose = FALSE,
        dropout.type = "experiment")
sce <- logNormCounts(sce)
cores <- 2
logX <- as.matrix(logcounts(sce))
w <- rowVars_fast(logX, cores)
corMat <- getCorM("spearman", logcounts(sce), w, cores)
v <- doSVD(corMat, nCores=cores)
```

estkTW *Estimate the Number of Clusters (k) Using the Tracy-Widom Bound*

Description

This function estimates the number of clusters (k) in a dataset using the Tracy-Widom distribution as a bound for the eigenvalues of the scaled data covariance matrix.

Usage

```
estkTW(x)
```

Arguments

x	A numeric matrix or data frame where rows are observations and columns are variables.

Value

The estimated number of clusters (k).

Examples

```
library(scater)
library(splatter)

sce <- splatSimulate(group.prob = rep(1, 5)/5, sparsify = FALSE,
        batchCells=100, nGenes=1000, method = "groups", verbose = FALSE,
        dropout.type = "experiment")
sce <- logNormCounts(sce)
logX <- as.matrix(logcounts(sce))
k <- estkTW(logX)
```

findDropouts *Identify Dropout Events in Single-Cell Expression Data*

Description

Determines which zero values within a transposed, log-normalized expression matrix are likely dropout events. The identification is based on a weighted cell voting scheme, where weights are derived from a processed consensus matrix.

Usage

```
findDropouts(logX, consMtx)
```

Arguments

logX	A (sparse or dense) numeric matrix representing the transpose of a log-normalized gene expression matrix. Rows correspond to cells, and columns correspond to genes.
consMtx	A numeric matrix representing the processed consensus matrix obtained from clustering analysis.

Value

A two-column matrix (or data frame) where each row indicates the location (row index, column index) of a potential dropout event in the input matrix ‘logX‘.

Examples

```
library(scater)
library(BiocParallel)
library(splatter)

sce <- splatSimulate(group.prob = rep(1, 5)/5, sparsify = FALSE,
        batchCells=100, nGenes=1000, method = "groups", verbose = FALSE,
        dropout.type = "experiment")
sce <- logNormCounts(sce)
cores <- 2
logX <- as.matrix(logcounts(sce))
w <- rowVars_fast(logX, cores)
corMat <- getCorM("spearman", logcounts(sce), w, cores)
v <- doSVD(corMat, nCores=cores)
BPPARAM = MulticoreParam(cores)
consMtx <- runKM(logX, v, BPPARAM=bpparam())
dropIds <- findDropouts(logX, consMtx)
```

---

getConsMtx *This function calculates an average consensus matrix from a set of clustering solutions. It filters out values below a specified minimum threshold (‘consMin‘) and normalizes the remaining non-zero columns to sum to 1.*

Description

This function calculates an average consensus matrix from a set of clustering solutions. It filters out values below a specified minimum threshold (‘consMin‘) and normalizes the remaining non-zero columns to sum to 1.

Usage

```
getConsMtx(dat, consMin, n_cores)
```

Arguments

dat	An integer matrix where each column represents a different clustering solution (cluster assignments for each data point).
consMin	The minimum consensus value to retain. Values below this threshold are set to zero.
n_cores	The number of cores to use for parallel processing. This can speed up the normalization step.

Value

A processed consensus matrix where each element (i, j) represents the proportion of times data points i and j were assigned to the same cluster with filtering and normalization applied.

---

getCorM *Calculate Column-wise Correlation Matrix*

Description

Efficiently computes a column-wise correlation matrix for a given input matrix. Supports Pearson and Spearman correlations, with optional weighting for features.

Usage

```
getCorM(method, x, w, nCores)
```

Arguments

method	A character string specifying the correlation metric to use. Currently supported options are: - "spearman": Spearman rank correlation - "pearson": Pearson correlation
x	A numeric matrix where each column represents a sample.
w	(Optional) A numeric vector of weights for each feature (row) in x. If not provided, all features are equally weighted.
nCores	The number of cores to use for parallel processing.

Value

A correlation matrix of the same dimensions as the number of columns in ‘x‘. The values represent the pairwise correlations between samples (columns) based on the chosen method and optional weights.

Examples

```
library(scater)
library(splatter)

sce <- splatSimulate(group.prob = rep(1, 5)/5, sparsify = FALSE,
        batchCells=100, nGenes=1000, method = "groups", verbose = FALSE,
        dropout.type = "experiment")
sce <- logNormCounts(sce)
cores <- 2
logX <- as.matrix(logcounts(sce))
w <- rowVars_fast(logX, cores)
corMat <- getCorM("spearman", logcounts(sce), w, cores)
```

---

getScale *Computes Means and Standard Deviations for Scaling*

Description

Computes Means and Standard Deviations for Scaling

Usage

```
getScale(x, n_cores)
```

Arguments

x	A numeric matrix representing the gene expression data, where rows are genes and columns are samples.
n_cores	The number of cores to use for parallel processing.

Value

A list containing: * ‘means‘: A numeric vector of column means. * ‘sds‘: A numeric vector of column standard deviations.

---

rowVars_fast *Computes Row Variances Efficiently*

Description

Computes Row Variances Efficiently

Usage

```
rowVars_fast(x, n_cores)
```

Arguments

x	A numeric dense matrix for which to compute row variances.
n_cores	The number of cores to utilize for parallel processing.

Value

A numeric vector containing the variance for each row of the input matrix.

Examples

```
library(Matrix)
rand_vals <- sample(0:10,1e4,replace=TRUE, p=c(0.99,rep(0.001,10)))
x <- as.matrix(Matrix(rand_vals,ncol=5))
cores <- 2
vars_vector <- rowVars_fast(x, cores)
```

---

runKM *Perform Consensus K-Means Clustering*

Description

Executes k-means clustering on multiple subsets of data defined by singular value decomposition (SVD) components, and then aggregates the results into a consensus matrix.

Usage

```
runKM(
  logX,
  v,
  maxSets = 8,
  k,
  consMin = 0.75,
  kmNStart,
  kmMax = 1000,
  BPPARAM = bpparam()
)
```

Arguments

logX	A (sparse or dense) numeric matrix representing the transpose of a log-normalized gene expression matrix. Rows correspond to cells, and columns correspond to genes.
v	A matrix of right singular vectors obtained from SVD of a distance matrix derived from 'logX'.
maxSets	(Optional) The maximum number of sub-datasets used for consensus clustering (default: 8).

k	(Optional) The number of clusters (cell groups) in the data. If not provided, it is estimated using the Tracy-Widom Bound.
consMin	(Optional) The low-pass filter threshold for processing the consensus matrix (default: 0.75).
kmNStart	nstart parameter passed to kmeans. function. Can be set manually. By default it is 1000 for up to 2000 cells and 50 for more than 2000 cells.
kmMax	iter.max parameter passed to kmeans.
BPPARAM	(Optional) A BiocParallelParam object for parallel processing (default: bpparam()).

Value

A consensus matrix summarizing the clustering results across multiple sub-datasets.

Examples

```
library(scater)
library(BiocParallel)
library(splatter)

sce <- splatSimulate(group.prob = rep(1, 5)/5, sparsify = FALSE,
        batchCells=100, nGenes=1000, method = "groups", verbose = FALSE,
        dropout.type = "experiment")
sce <- logNormCounts(sce)
cores <- 2
logX <- as.matrix(logcounts(sce))
w <- rowVars_fast(logX, cores)
corMat <- getCorM("spearman", logcounts(sce), w, cores)
v <- doSVD(corMat, nCores=cores)
BPPARAM = MulticoreParam(cores)
consMtx <- runKM(logX, v, BPPARAM=bpparam())
```

---

solver *Computes imputed expression matrix using linear eq solver*

Description

Computes imputed expression matrix using linear eq solver

Usage

```
solver(cm, em, ids, n_cores)
```

Arguments

cm	processed consensus matrix
em	expression matrix
ids	location of values determined to be dropout events
n_cores	number of cores to use for parallel computation.

Value

imputed expression matrix

`solver2` *Fast Calculation of "Dropout" values*

Description

Fast Calculation of "Dropout" values

Usage

```
solver2(cm, em, ids, n_cores)
```

Arguments

`cm`	A numeric matrix representing the consensus matrix.
`em`	A dense numeric matrix representing the gene expression data, where rows are genes and columns are samples.
`ids`	An integer matrix specifying the row and column indices of entries for which to calculate importance scores. Each row of ‘ids‘ should contain two integers: the row index (gene) and column index (sample) in the ‘em‘ matrix.
`n_cores`	The number of cores to use for parallel processing.

Value

A numeric vector of imputed dropout values, corresponding to the entries specified in the ‘ids‘ matrix.

`sparseColRanks_fast` *Computes rankings for each column of a matrix in parallel.*

Description

Computes rankings for each column of a matrix in parallel.

Usage

```
sparseColRanks_fast(x, n_cores)
```

Arguments

`x`	The input matrix to be ranked.
`n_cores`	The number of CPU cores to use for parallel processing.

Value

A matrix where each column contains the rankings for the corresponding column in the input matrix.

---

sparseSolver2 *Fast Calculation of "Dropout" values*

Description

Fast Calculation of "Dropout" values

Usage

```
sparseSolver2(cm, em, ids, n_cores)
```

Arguments

cm	A numeric matrix representing the consensus matrix.
em	A sparse numeric matrix representing the gene expression data, where rows are genes and columns are samples.
ids	An integer matrix specifying the row and column indices of entries for which to calculate importance scores. Each row of ‘ids‘ should contain two integers: the row index (gene) and column index (sample) in the ‘em‘ matrix.
n_cores	The number of cores to use for parallel processing.

Value

A numeric vector of imputed dropout values, corresponding to the entries specified in the ‘ids‘ matrix.

---

wCor_fast *Computes a Weighted Pearson*

Description

This function calculates weighted Pearson correlation matrix

Usage

```
wCor_fast(x, w, n_cores)
```

Arguments

x	The input matrix (dense), where each column represents a set of observations.
w	A vector of weights, one for each observation (must have the same number of elements as rows in ‘x‘).
n_cores	The number of CPU cores to utilize for parallel computation.

Value

A weighted Pearson correlation matrix

Index

ccImpute, 2
ccImpute,SingleCellExperiment-method
        *(*ccImpute*)*, 2
ccImpute.SingleCellExperiment
        *(*ccImpute*)*, 2
colRanks_fast, 5
computeDropouts, 5
cor_fast, 6

doSVD, 7

estkTW, 8

findDropouts, 8

getConsMtx, 9
getCorM, 10
getScale, 11

kmeans, *4*, *13*

rowVars_fast, 11
runKM, 12

solver, 13
solver2, 14
sparseColRanks_fast, 14
sparseSolver2, 15

wCor_fast, 15