Package: scCB2 (via r-universe)

November 12, 2026

Version 1.22.0

Date 2023/4/18

Title CB2 improves power of cell detection in droplet-based single-cell RNA sequencing data

Depends R (>= 3.6.0)

Imports SingleCellExperiment, SummarizedExperiment, Matrix, methods, utils, stats, edgeR, rhdf5, parallel, DropletUtils, doParallel, iterators, foreach, Seurat

Suggests testthat (>= 2.1.0), KernSmooth, beachmat, knitr, BiocStyle, rmarkdown

biocViews DataImport, RNASeq, SingleCell, Sequencing, GeneExpression, Transcriptomics, Preprocessing, Clustering

Description scCB2 is an R package implementing CB2 for distinguishing real cells from empty droplets in droplet-based single cell RNA-seq experiments (especially for 10x Chromium). It is based on clustering similar barcodes and calculating Monte-Carlo p-value for each cluster to test against background distribution. This cluster-level test outperforms single-barcode-level tests in dealing with low count barcodes and homogeneous sequencing library, while keeping FDR well controlled.

License GPL-3

NeedsCompilation yes

VignetteBuilder knitr

Encoding UTF-8

SystemRequirements C++11

RoxygenNote 7.1.2

URL https://github.com/zijianni/scCB2

BugReports https://github.com/zijianni/scCB2/issues

Config/pak/sysreqs libglpk-dev make libicu-dev libpng-dev libxml2-dev libssl-dev zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:53:07 UTC

RemoteUrl https://github.com/bioc/scCB2

RemoteRef RELEASE_3_23

RemoteSha 40bcb0ecddf3cd46b1364c878ff101c7efa85dd4

Contents

CB2FindCell 2
CheckBackgroundCutoff 4
FilterGB 5
GetCellMat 6
mbrainSub 7
QuickCB2 8
Read10xRaw 9

Index **11**

`CB2FindCell` *Main function of distinguish real cells from empty droplets using clustering-based Monte-Carlo test*

Description

The main function of `scCB2` package. Distinguish real cells from empty droplets using clustering-based Monte-Carlo test.

Usage

```
CB2FindCell(
  RawDat,
  FDR_threshold = 0.01,
  lower = 100,
  upper = NULL,
  GeneExpressionOnly = TRUE,
  Ncores = 2,
  TopNGene = 30000,
  verbose = TRUE
)
```

Arguments

`RawDat`	Matrix. Supports standard matrix or sparse matrix. This is the raw feature-by-barcode count matrix.
`FDR_threshold`	Numeric between 0 and 1. Default: 0.01. The False Discovery Rate (FDR) to be controlled for multiple testing.

`lower` Positive integer. Default: 100. All barcodes whose total count below or equal to this threshold are defined as background empty droplets. They will be used to estimate the background distribution. The remaining barcodes will be test against background distribution. If sequencing depth is deliberately made higher (lower) than usual, this threshold can be leveled up (down) correspondingly to get reasonable number of cells. Recommended sequencing depth for this default threshold: 40,000~80,000 reads per cell.

`upper` Positive integer. Default: `NULL`. This is the upper threshold for large barcodes. All barcodes whose total counts are larger or equal to upper threshold are directly classified as real cells prior to testing. If `upper = NULL`, the knee point of the log rank curve of barcodes total counts will serve as the upper threshold, which is calculated using package `DropletUtils`'s method. If `upper = Inf`, no barcodes will be retained prior to testing. If manually specified, it should be greater than pooling threshold.

`GeneExpressionOnly` Logical. Default: `TRUE`. For 10x Cell Ranger version >=3, extra features (surface proteins, cell multiplexing oligos, etc) besides genes are measured simultaneously. If `GeneExpressionOnly = TRUE`, only genes are used for testing. Removing extra features are recommended because the default pooling threshold (100) is chosen only for handling gene expression. Extra features expression level is hugely different from gene expression level. If using the default pooling threshold while keeping extra features, the estimated background distribution will be hugely biased and does not reflect the real background distribution of empty droplets.

`Ncores` Positive integer. Default: 2. Number of cores for parallel computation.

`TopNGene` Positive integer. Default: 30000. Number of top highly expressed genes to use. This threshold avoids high number of false positives in ultra-high dimensional datasets, e.g. 10x barnyard data.

`verbose` Logical. Default: `TRUE`. If `verbose = TRUE`, progressing messages will be printed.

Details

Input data is a feature-by-barcode matrix. Background barcodes are defined based on `lower`. Large barcodes are automatically treated as real cells based on `upper`. Remaining barcodes will be first clustered into subgroups, then tested against background using Monte-Carlo p-values simulated from Multinomial distribution. The rest barcodes will be further tested using EmptyDrops (Aaron T. L. Lun *et. al. 2019*). FDR is controlled based on `FDR_threshold`.

This function supports parallel computation. `Ncores` is used to specify number of cores.

Under CellRanger version >=3, extra features other than genes are simultaneously measured (e.g. surface protein, cell multiplexing oligo). We recommend filtering them out using `GeneExpressionOnly = TRUE` because the expression of extra features is not in the same scale as gene expression counts. If using the default pooling threshold while keeping extra features, the estimated background distribution will be hugely biased and does not reflect the real background distribution of empty droplets. The resulting matrix will contain lots of barcodes who have almost zero gene expression and relatively high extra features expression, which are usually not useful for RNA-Seq study.

Value

An object of class `SummarizedExperiment`. The slot `assays` contains the real cell barcode matrix distinguished during cluster-level test, single-barcode-level test plus large cells who exceed the upper threshold. The slot `metadata` contains (1) testing statistics (Pearson correlation to the background) for all candidate barcode clusters, (2) barcode IDs for all candidate barcode clusters, the name of each cluster is its median barcode size, (3) testing statistics (log likelihood under background distribution) for remaining single barcodes not clustered, (4) background distribution count vector without Good-Turing correction.

Examples

```
# raw data, all barcodes
data(mbrainSub)
str(mbrainSub)

# run CB2 on the first 10000 barcodes
CBOut <- CB2FindCell(mbrainSub[,1:10000], FDR_threshold = 0.01,
    lower = 100, Ncores = 2)
RealCell <- GetCellMat(CBOut, MTfilter = 0.05)

# real cells
str(RealCell)
```

CheckBackgroundCutoff *Check different background cutoffs and recommend an appropriate one*

Description

The key parameter of CB2 as well as other similar methods is the background cutoff, which divides barcodes into two groups: (1) small barcodes that are most likely to be background; (2) the rest barcodes that can be either background or cell, and remain to be tested. Those small barcodes will be used to estimate a background distribution, which guides the identification of cells from background. It is crucial to have an unbiased estimation of the background distribution.

Usage

```
CheckBackgroundCutoff(RawDat)
```

Arguments

`RawDat`	Matrix. Supports standard matrix or sparse matrix. This is the raw feature-by-barcode count matrix.

Details

An appropriate background cutoff should be reasonably large to contain enough background information, but shouldn't be too large to mistakenly include real cells. We recommend a background cutoff which (1) puts more than 90 (2) puts more than 10 The smallest cutoff satisfying either condition is the recommended cutoff.

Value

A list containing a data frame summarizing background information under different background cutoffs, and the recommended background cutoff for the input data. For the data frame, 'n_bg_bcs' is the number of barcodes less or equal to the cutoff, 'n_bg_counts' is the number of UMI counts within the barcodes less or equal to the cutoff, 'prop_bg_bcs' and 'prop_bg_counts' are the corresponding proportions.

Examples

```
data(mbrainSub)
CheckBackgroundCutoff(mbrainSub)
```

FilterGB *Filter out low count genes and barcodes from count matrix*

Description

This function is used for filtering out low count genes and barcodes from count matrix based on total gene expression count (row sums) and barcode expression count (column sums). `CB2FindCell` has already integrated this function into it with `g_threshold = 0` and `b_threshold = 0`. If users plan to customize their filtering threshold, this function can be applied to the raw expression count matrix prior to running `CB2FindCell`.

Usage

```
FilterGB(dat, g_threshold = 0, b_threshold = 0)
```

Arguments

`dat`	Input count matrix to be filtered.
`g_threshold`	Nonnegative integer. Default: `0`. Filtering threshold for genes. Any gene whose total expression count is less or equal to `g_threshold` will be filtered out.
`b_threshold`	Nonnegative integer. Default: `0`. Filtering threshold for barcodes. Any barcode whose total count is less or equal to `b_threshold` will be filtered out.

Value

A filtered matrix with the same format as input matrix.

Examples

```
data(mbrainSub)
dim(mbrainSub)
mbrainSub_f <- FilterGB(mbrainSub)
dim(mbrainSub_f)
```

`GetCellMat` *Extract real cell matrix from* `CB2FindCell` *output and optionally filter out low-quality cells*

Description

Handy function to extract real cell matrix from `CB2FindCell` output. It provides the option to filter out broken cells based on proportion of mitochondrial gene expressions. The input can also be a sparse matrix only for cell filtering.

Usage

```
GetCellMat(CBout, MTfilter = 0.25, MTgene = NULL)
```

Arguments

`CBout`	Output object from `CB2FindCell`, or a sparse matrix (for example, from `QuickCB2`).
`MTfilter`	Numeric value between 0 and 1. Default: `0.25`. For each barcode, if the proportion of mitochondrial gene expression exceeds `MTfilter`, this barcode will be filtered out. By default, cell barcodes with more than 25 filtered out. Set `MTfilter = 1` for no filtering. The proportion of mitochondrial gene expressions is usually a criterion for evaluating cell quality, and is calculated using the scaled sum of all genes starting with "MT-" (human) or "mt-" (mouse) if row names are gene symbols, or customized mitochondrial genes specified by `MTgene`.
`MTgene`	Character vector. User may specify customized mitochondrial gene names to perform the filtering. This should correspond to a subset of row names in raw data.

Value

A `dgCMatrix` count matrix of real cells.

Examples

```
# Please also refer to the example in function CB2FindCell.

# Simulate CB2FindCell output object.
library(SummarizedExperiment)
data(mbrainSub)
```

```
mbrainReal <- mbrainSub[,Matrix::colSums(mbrainSub)>500]

CBOut <- SummarizedExperiment(
    list(cell_matrix = mbrainReal[,sample(ncol(mbrainReal),
                                          200, replace = TRUE)]))

# Get cell matrix, filtering out barcodes with
# more than 10% of counts from mitochondrial genes.

RealCell <- GetCellMat(CBOut, MTfilter = 0.1)
str(RealCell)
```

`mbrainSub` *Subset of 1k Brain Cells from an E18 Mouse*

Description

1k Brain Cells from an E18 Mouse is a public dataset from 10X Genomics. This subset is the first 50,000 barcodes of original data.

Usage

```
data(mbrainSub)
```

Format

An object of class `"dgCMatrix"`.

Source

1k Brain Cells from an E18 Mouse

Examples

```
data(mbrainSub)
str(mbrainSub)
```

QuickCB2 *All-in-one function from raw data to filtered cell matrix*

Description

All-in-one function for scCB2. Take 10x output raw data as input and return either a matrix of real cells identified by CB2 or a Seurat object containing this matrix, which can be incorporated with downstream analysis using Seurat pipeline.

Usage

```
QuickCB2(
  dir = NULL,
  h5file = NULL,
  FDR_threshold = 0.01,
  MTfilter = 1,
  MTgene = NULL,
  AsSeurat = FALSE,
  Ncores = 2,
  ...
)
```

Arguments

dir	The directory of 10x output data. For Cell Ranger version <3, directory should include three files: barcodes.tsv, genes.tsv, matrix.mtx. For Cell Ranger version >=3, directory should include three files: barcodes.tsv.gz, features.tsv.gz, matrix.mtx.gz.
h5file	The path of 10x output HDF5 file (ended with .h5).
FDR_threshold	Numeric between 0 and 1. Default: 0.01. The False Discovery Rate (FDR) to be controlled for multiple testing.
MTfilter	Numeric value between 0 and 1. Default: 1 (No filtering). For each barcode, if the proportion of mitochondrial gene expression exceeds MTfilter, this barcode will be filtered out. No barcode exceeds 100% mitochondrial gene expression, thus the default (100%) corresponds to no filtering. The proportion of mitochondrial gene expressions is usually a criterion for evaluating cell quality, and is calculated using the scaled sum of all genes starting with "MT-" (human) or "mt-" (mouse) if row names are gene symbols, or customized mitochondrial genes specified by MTgene.
MTgene	Character vector. User may specify customized mitochondrial gene IDs to perform the filtering. This should correspond to a subset of row names in raw data.
AsSeurat	Logical. Default: FALSE. Decides whether a Seurat object is returned instead of cell matrix. Set to TRUE so that users can directly apply Seurat pipeline for downstream analyses.
Ncores	Positive integer. Default: detectCores() - 2. Number of cores for parallel computation.

... Additional arguments to be passed to `CB2FindCell`.

Details

`QuickCB2` is a quick function to apply CB2 on 10x Cell Ranger raw data by combining `Read10xRaw`, `Read10xRawH5`, `CB2FindCell` and `GetCellMat` into one simple function under default parameters.

Value

Either a sparse matrix of real cells identified by CB2 or a Seurat object containing real cell matrix.

Examples

```
# simulate 10x output files
data(mbrainSub)
mbrainSub <- mbrainSub[,1:10000]
data_dir <- file.path(tempdir(),"CB2example")
dir.create(data_dir)
gene_name <- rownames(mbrainSub)

# For simplicity, use gene names to generate gene IDs to fit the format.
gene_id <- paste0("ENSG_fake_",gene_name)
barcode_id <- colnames(mbrainSub)
Matrix::writeMM(mbrainSub,file = file.path(data_dir,"matrix.mtx"))
write.table(barcode_id,file = file.path(data_dir,"barcodes.tsv"),
    sep = "\t", quote = FALSE, col.names = FALSE, row.names = FALSE)
write.table(cbind(gene_id,gene_name),file = file.path(data_dir,"genes.tsv"),
    sep = "\t", quote = FALSE, col.names = FALSE, row.names = FALSE)

# Run QuickCB2 on 10x raw data and get cell matrix.
# Control FDR at 1%. Use 2-core parallel computation.

RealCell <- QuickCB2(dir = data_dir,
                     FDR_threshold = 0.01,
                     Ncores = 2)
str(RealCell)
```

`Read10xRaw` *Read 10x output data*

Description

`Read10xRaw` is a one-line handy function for reading 10x Cell Ranger output data, producing a count matrix for input to `CB2FindCell`. `Read10xRawH5` is for reading 10x Cell Ranger output HDF5 file (ended with .h5). Works under both old (<3) and new (>=3) Cell Ranger version.

Usage

```
Read10xRaw(dir = NULL, row.name = "symbol", meta = FALSE)

Read10xRawH5(h5file, row.name = "symbol", meta = FALSE)
```

Arguments

dir	The directory of 10x output data. For Cell Ranger version <3, directory should include three files: barcodes.tsv, genes.tsv, matrix.mtx. For Cell Ranger version >=3, directory should include three files: barcodes.tsv.gz, features.tsv.gz, matrix.mtx.gz.
row.name	Specify either using gene symbols (row.name = "symbol") or gene Ensembl IDs (row.name = "id") as row names of the count matrix. Default is row.name = "symbol".
meta	Logical. If TRUE, returns a list containing both the count matrix and metadata of genes (features). Metadata includes feature names, IDs and other additional information depending on Cell Ranger output. If FALSE (default), only returns the count matrix.
h5file	The path of 10x output HDF5 file (ended with .h5).

Value

If meta = TRUE, returns a list of two elements: a "dgCMatrix" sparse matrix containing expression counts and a data frame containing metadata of genes (features). For the count matrix, each row is a gene (feature) and each column is a barcode. If meta = FALSE, only returns the count matrix.

Examples

```
# simulate 10x output files
data(mbrainSub)
data_dir <- file.path(tempdir(),"CB2example")
dir.create(data_dir)
gene_name <- rownames(mbrainSub)

# For simplicity, use gene names to generate gene IDs to fit the format.
gene_id <- paste0("ENSG_fake_",gene_name)
barcode_id <- colnames(mbrainSub)
Matrix::writeMM(mbrainSub,file = file.path(data_dir,"matrix.mtx"))
write.table(barcode_id,file = file.path(data_dir,"barcodes.tsv"),
    sep = "\t", quote = FALSE, col.names = FALSE, row.names = FALSE)
write.table(cbind(gene_id,gene_name),file = file.path(data_dir,"genes.tsv"),
    sep = "\t", quote = FALSE, col.names = FALSE, row.names = FALSE)

# read files
list.files(data_dir)
mbrainSub_new <- Read10xRaw(data_dir)
str(mbrainSub_new)
identical(mbrainSub, mbrainSub_new)
```

Index

∗ **datasets**
mbrainSub, 7

CB2FindCell, 2
CheckBackgroundCutoff, 4

FilterGB, 5

GetCellMat, 6

mbrainSub, 7

QuickCB2, 8

Read10xRaw, 9
Read10xRawH5 *(*Read10xRaw*)*, 9