Package 'ondisc'

October 14, 2022

Title Fast, Universal, and Intuitive Computing on Large-Scale Single-Cell Data

Version 1.0.0

Description Single-cell datasets are growing in size, posing challenges as well as opportunities for biology researchers. 'ondisc' (short for ``on-disk single cell'') enables users to easily and efficiently analyze large-scale single-cell data. 'ondisc' makes computing on large-scale single-cell data FUN: Fast, Universal, and iNtuitive.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

URL https://timothy-barry.github.io/ondisc/

Suggests testthat, knitr, rmarkdown, covr

Imports readr, methods, magrittr, rhdf5, data.table, Matrix, Rcpp, crayon, dplyr

Depends R (>= 3.5.0)

VignetteBuilder knitr

LinkingTo Rcpp, Rhdf5lib

SystemRequirements GNU make

NeedsCompilation yes

Author Timothy Barry [aut, cre] (<https://orcid.org/0000-0002-4356-627X>), Eugene Katsevich [ths] (<https://orcid.org/0000-0003-0598-2050>), Kathryn Roeder [ths]

Maintainer Timothy Barry <tbarry2@andrew.cmu.edu>

Repository CRAN

Date/Publication 2021-03-05 09:10:02 UTC

R topics documented:

`create_ondisc_matrix_from_mtx` *Create an* `ondisc_matrix` *from a .mtx file.*

Description

Initializes an `ondisc_matrix` from a .mtx file, a features.tsv file, and a barcodes.tsv file. Returns an `ondisc_matrix` along with cell-specific and feature-specific covariate matrices.

Usage

```
create_ondisc_matrix_from_mtx(
  mtx_fp,
  barcodes_fp,
  features_fp,
  n_lines_per_chunk = 3e+08,
  on_disk_dir = NULL,
  file_name = NULL,
  return_metadata_ondisc_matrix = FALSE,
  progress = TRUE
)
```

Arguments

`mtx_fp`	file path to a .mtx file storing the expression data. The .mtx file can represent either an integer matrix or a logical (i.e., binary) matrix. If the .mtx file contains only two columns (after the initial three-column row of metadata), then the .mtx file is assumed to represent a logical matrix.
`barcodes_fp`	file path to the .tsv file containing the cell barcodes.

features_fp file path to the features.tsv file. The first column (required) contains the feature IDs (e.g., ENSG00000186092), and the second column (optional) contains the human-readable feature names (e.g., OR4F5). Subsequent columns are discarded.

n_lines_per_chunk (optional) number of lines in .mtx file to process per chunk. Defaults to 3e+08.

on_disk_dir (optional) directory in which to store the on-disk portion of the ondisc_matrix. Defaults to the directory in which the .mtx file is located.

file_name (optional) name of the file in which to store the .h5 data on-disk. Defaults to ondisc_matrix_x.h5, where x is a unique integer starting at 1.

return_metadata_ondisc_matrix (optional) return the output as a metadata_ondisc_matrix (instead of a list)? Defaults to FALSE.

progress (optional; default FALSE) print progress messages?

Details

The function can compute the following cell-specific and feature-specific covariates:

- cell-specific: (i) total number of features expressed in cell (n_nonzero_cell), (ii) total UMI count (n_umis_cell), and (iii) percentage of UMIs that map to mitochondrial genes (p_mito_cell).
- feature-specific: (i) total number of cells in which feature is expressed (n_nonzero_feature), (ii) mean expression of feature across cells (mean_expression_feature), (iii) coefficient of variation of feature expression across cells (coef_of_variation_feature).

The function decides which covariates to compute given the input; in general, the function computes the maximum set of covariates possible.

Value

A list containing (i) an ondisc_matrix, (ii) a cell-specific covariate matrix, and (iii) a feature-specific covariate matrix; if the parameter return_metadata_ondisc_matrix set to TRUE, converts the list to a metadata_ondisc_matrix before returning.

Examples

```
## Not run:
# First example: initialize a metadata_ondisc_matrix
# using simulated expression data; store output in tempdir()
file_locs <- system.file("extdata",package = "ondisc",
c("gene_expression.mtx", "genes.tsv", "cell_barcodes.tsv"))
names(file_locs) <- c("expressions", "features", "barcodes")
expression_data <- create_ondisc_matrix_from_mtx(mtx_fp = file_locs[["expressions"]],
barcodes_fp = file_locs[["barcodes"]],
features_fp = file_locs[["features"]],
on_disk_dir = tempdir(),
file_name = "expressions",
return_metadata_ondisc_matrix = TRUE)
saveRDS(object = expression_data, file = paste0(tempdir(), "/expressions.rds"))
```

```
# Second example: initialize a metadata_ondisc_matrix using simulated
# gRNA perturbation data; store in tempdir()
file_locs <- system.file("extdata", package = "ondisc",
c("perturbation.mtx", "guides.tsv", "cell_barcodes.tsv"))
names(file_locs) <- c("perturbations", "features", "barcodes")
perturbation_data <- create_ondisc_matrix_from_mtx(mtx_fp = file_locs[["perturbations"]],
barcodes_fp = file_locs[["barcodes"]],
features_fp = file_locs[["features"]],
on_disk_dir = tempdir(),
file_name = "perturbations",
return_metadata_ondisc_matrix = TRUE)
saveRDS(object = perturbation_data, file = paste0(tempdir(), "/perturbations.rds"))

## End(Not run)
```

`dim` *Get dimension*

Description

Return the dimension of an `ondisc_matrix`, `metadata_ondisc_matrix`, or `multimodal_ondisc_matrix`.

Usage

```
## S4 method for signature 'ondisc_matrix'
dim(x)

## S4 method for signature 'metadata_ondisc_matrix'
dim(x)

## S4 method for signature 'multimodal_ondisc_matrix'
dim(x)

ncol(x)

nrow(x)

## S4 method for signature 'multimodal_ondisc_matrix'
ncol(x)

## S4 method for signature 'multimodal_ondisc_matrix'
nrow(x)
```

Arguments

`x`	an `ondisc_matrix`, `metadata_ondisc_matrix`, or `multimodal_ondisc_matrix`.

Value

If x is an ondisc_matrix or metadata_ondisc_matrix, length-two integer vector containing the dimension of x; if x is a multimodal_ondisc_matrix, a list of integer vectors containing the dimensions of the constituent modalities of x.

Examples

```
# NOTE: You must create the RDS files "expressions.rds" and
# "perturbations.rds" to run this example. Navigate to the help file of
# "create_ondisc_matrix_from_mtx" (via ?create_ondisc_matrix_from_mtx),
# and execute both code blocks.

# dimension of an ondisc_matrix
h5_fp <- paste0(tempdir(), "/expressions.h5")
if (file.exists(h5_fp)) {
odm <- ondisc_matrix(h5_file = h5_fp)
dim(odm)
}

# dimension of a metadata_ondic_matrix
expressions_fp <- paste0(tempdir(), "/expressions.rds")
if (file.exists(expressions_fp)) {
expressions <- readRDS(expressions_fp)
dim(expressions)
}

# dimension of a multimodal_ondisc_matrix
expression_fp <- paste0(tempdir(), "/expressions.rds")
perturbations_fp <- paste0(tempdir(), "/perturbations.rds")
if (file.exists(expression_fp) && file.exists(perturbations_fp)) {
   crispr_experiment <- multimodal_ondisc_matrix(list(expressions = readRDS(expression_fp),
    perturbations = readRDS(perturbations_fp)))
    dim(crispr_experiment)
}
```

---

extract-odm *Pull a submatrix into memory using the* [[*operator.*

Description

Apply the [[operator to an ondisc_matrix to pull a submatrix into memory. You can pass logical, character, or numeric vectors to [[; character vectors are assumed to refer to feature IDs (for rows) and cell barcodes (for columns).

Usage

```
## S4 method for signature 'ondisc_matrix,missing,missing'
x[[i, j]]
```

```
## S4 method for signature 'ondisc_matrix,ANY,missing'
x[[i, j]]

## S4 method for signature 'ondisc_matrix,missing,ANY'
x[[i, j]]

## S4 method for signature 'ondisc_matrix,ANY,ANY'
x[[i, j]]

## S4 method for signature 'metadata_ondisc_matrix,ANY,ANY'
x[[i, j]]

## S4 method for signature 'multimodal_ondisc_matrix,ANY,ANY'
x[[i, j]]
```

Arguments

x	an ondisc_matrix object.
i	a vector (numeric, logical, or character) indicating features to pull into memory.
j	a vector (numeric, logical, or character) indicating cells to pull into memory.

Details

You can apply [[to ondisc_matrix objects only. You cannot apply [[to metadata_ondisc_matrix or multimodal_ondisc_matrix objects, because in the latter case the data to be accessed is ambiguous.

You can remember the difference between [and [[by thinking about R lists: [is used to subset a list, and [[is used to access elements stored *inside* a list. Similarly, [is used to subset an ondisc_matrix, and [[is used to access a submatrix usable within R.

Value

a matrix (as implemented by the Matrix package).

Examples

```
# NOTE: You must create the HDF5 file "expressions.h5" to run this example.
# Navigate to the help file of "create_ondisc_matrix_from_mtx"
# (via ?create_ondisc_matrix_from_mtx), and execute the code in the first code block.

h5_fp <- paste0(tempdir(), "/expressions.h5")
if (file.exists(h5_fp)) {
odm <- ondisc_matrix(h5_file = h5_fp)
# extract cells 100-110:
x <- odm[[,100:110]]
# extract genes ENSG00000188305, ENSG00000257284, ENSG00000251655:
x <- odm[[c("ENSG00000188305", "ENSG00000257284", "ENSG00000251655"),]]
# extract cells CTTAGGACACTGGCGT-1 and AAAGGATTCACATCAG-1:
x <- odm[[,c("CTTAGGACACTGGCGT-1", "AAAGGATTCACATCAG-1")]]
}
```

get-names *Get cell barcodes, feature names, and feature IDs*

Description

Obtain cell barcodes, feature names, and feature IDs of an ondisc_matrix, metadata_ondisc_matrix, or multimodal_ondisc_matrix.

Usage

```
get_feature_ids(x)

get_feature_names(x)

get_cell_barcodes(x)

## S4 method for signature 'ondisc_matrix'
get_feature_ids(x)

## S4 method for signature 'ondisc_matrix'
get_feature_names(x)

## S4 method for signature 'ondisc_matrix'
get_cell_barcodes(x)

## S4 method for signature 'metadata_ondisc_matrix'
get_feature_ids(x)

## S4 method for signature 'metadata_ondisc_matrix'
get_feature_names(x)

## S4 method for signature 'metadata_ondisc_matrix'
get_cell_barcodes(x)

## S4 method for signature 'multimodal_ondisc_matrix'
get_feature_ids(x)

## S4 method for signature 'multimodal_ondisc_matrix'
get_feature_names(x)

## S4 method for signature 'multimodal_ondisc_matrix'
get_cell_barcodes(x)
```

Arguments

x	an object of class ondisc_matrix, covaraite_ondisc_matrix, or multimodal_ondisc_matrix.

Details

The following functions can be used to obtain feature and cell identifiers:

- `get_cell_barcodes`: return the cell barcodes.
- `get_feature_names`: return the feature names.
- `get_feature_ids`: return the IDs of the features.

In general, these functions return a character vector containing the requested identifiers. When `get_feature_names` or `get_feature_ids` is called on a `multimodal_ondisc_matrix`, the function instead returns a list containing the feature names and feature IDs, respectively, of the modalities contained within the `multimodal_ondisc_matrix`.

Value

A character vector or list of character vectors containing the requested identifiers.

Examples

```
# NOTE: You must create the RDS files "expressions.rds" and
# "perturbations.rds" to run this example. Navigate to the help file of
# "create_ondisc_matrix_from_mtx" (via ?create_ondisc_matrix_from_mtx),
# and execute both code blocks.

# ondisc_matrix
h5_fp <- paste0(tempdir(), "/expressions.h5")
if (file.exists(h5_fp)) {
odm <- ondisc_matrix(h5_file = h5_fp)
barcodes <- get_cell_barcodes(odm)
feature_names <- get_feature_names(odm)
feature_ids <- get_feature_ids(odm)
}

# metadata_ondic_matrix
expressions_fp <- paste0(tempdir(), "/expressions.rds")
if (file.exists(expressions_fp)) {
expressions <- readRDS(expressions_fp)
barcodes <- get_cell_barcodes(odm)
feature_names <- get_feature_names(odm)
feature_ids <- get_feature_ids(odm)
}

# multimodal_ondisc_matrix
expression_fp <- paste0(tempdir(), "/expressions.rds")
perturbations_fp <- paste0(tempdir(), "/perturbations.rds")
if (file.exists(expression_fp) && file.exists(perturbations_fp)) {
   crispr_experiment <- multimodal_ondisc_matrix(list(expressions = readRDS(expression_fp),
    perturbations = readRDS(perturbations_fp)))
    barcodes <- get_cell_barcodes(crispr_experiment)
    feature_ids <- get_feature_ids(crispr_experiment)
}
```

head,ondisc_matrix-method *head*

Description

Print the first few rows and columns of an ondisc_matrix.

Usage

```
## S4 method for signature 'ondisc_matrix'
head(x)
```

Arguments

x	an ondisc_matrix.

Value

NULL; called for printing

Examples

```
# NOTE: You must create the HDF5 file "expressions.h5" to run this example.
# Navigate to the help file of "create_ondisc_matrix_from_mtx"
# (via ?create_ondisc_matrix_from_mtx), and execute the code in the first code block.

h5_fp <- paste0(tempdir(), "/expressions.h5")
if (file.exists(h5_fp)) {
odm <- ondisc_matrix(h5_file = h5_fp)
head(odm)
}
```

metadata_ondisc_matrix *metadata_ondisc_matrix class*

Description

A metadata_ondisc_matrix stores an ondisc_matrix, along with cell-specific and feature-specific covariate matrices.

Construct a metadata_ondisc_matrix by passing an ondisc_matrix, along with its associated cell_covariates and feature_covariates.

Usage

```
metadata_ondisc_matrix(ondisc_matrix, cell_covariates, feature_covariates)

metadata_ondisc_matrix(ondisc_matrix, cell_covariates, feature_covariates)
```

Arguments

ondisc_matrix	an ondisc_matrix.
cell_covariates	a data frame storing the cell-specific covariates.
feature_covariates	a data frame storing the feature-specific covariates.

Value

a metadata_ondisc_matrix.

Slots

ondisc_matrix an ondisc_matrix.

cell_covariates a data frame of cell covariates.

feature_covariates a data frame of feature covariates.

Examples

```
# NOTE: You must create the HDF5 file "expressions.h5" and the RDS file
# "expressions.rds" to run this example. Navigate to the help file of
# "create_ondisc_matrix_from_mtx" (via ?create_ondisc_matrix_from_mtx),
# and execute the code in the first code block.
covariates_fp <- paste0(tempdir(), "/expressions.rds")
h5_fp <-  paste0(tempdir(), "/expressions.h5")
if (file.exists(covariates_fp) && file.exists(h5_fp)) {
covariate_odm <- readRDS(covariates_fp)
cell_covariate_matrix <- covariate_odm@cell_covariates
feature_covariate_matrix <- covariate_odm@feature_covariates
covariate_odm_copy <- metadata_ondisc_matrix(
ondisc_matrix = ondisc_matrix(h5_file = h5_fp),
cell_covariates = cell_covariate_matrix,
feature_covariates = feature_covariate_matrix)
}
```

multimodal_ondisc_matrix multimodal_ondisc_matrix *class*

Description

A multimodal_ondisc_matrix represents multimodal data.

Construct a multimodal_ondisc_matrix from a list of metadata_ondisc_matrix objects.

Usage

```
multimodal_ondisc_matrix(metadata_ondisc_matrix_list)

multimodal_ondisc_matrix(metadata_ondisc_matrix_list)
```

Arguments

metadata_ondisc_matrix_list	a named list containing metadata_ondisc_matrices; the names are taken to be the names of the modalities.

Value

a multimodal_ondisc_matrix

Slots

modalities	a list containing metadata_ondisc_matrix objects representing different modalities.
global_cell_covariates	a data frame containing the cell-specific covariates pooled across all modalities.

Examples

```
# NOTE: You must create the RDS files "expressions.rds" and
# "perturbations.rds" to run this example. Navigate to the help file of
# "create_ondisc_matrix_from_mtx" (via ?create_ondisc_matrix_from_mtx),
# and execute both code blocks.
expression_fp <- paste0(tempdir(), "/expressions.rds")
perturbations_fp <- paste0(tempdir(), "/perturbations.rds")
if (file.exists(expression_fp) && file.exists(perturbations_fp)) {
    expressions <- readRDS(expression_fp)
    perturbations <- readRDS(perturbations_fp)
    crispr_experiment <- multimodal_ondisc_matrix(list(expressions = expressions,
    perturbations = perturbations))
}
```

ondisc *ondisc: A package for out-of-memory computing on single-cell data*

Description

Single-cell datasets are large and are growing in size as sequencing costs drop. The ondisc package is designed to facilitate large-scale computing on single-cell expression data by providing access to expression matrices out-of-memory. ondisc is functional (i.e., all objects are persistent) and efficient (i.e., all algorithms are theoretically optimal in time).

ondisc_matrix `ondisc_matrix` *class*

Description

An `ondisc_matrix` represents a feature-by-cell expression matrix stored on-disk.

Construct an `ondisc_matrix` from an initialized .h5 file.

Usage

```
ondisc_matrix(h5_file)

ondisc_matrix(h5_file)
```

Arguments

`h5_file`	a .h5 file storing the on-disk portion of an initialized `ondisc_matrix` object.

Details

It is best to avoid interacting with the slots of an `ondisc_matrix` directly. Instead, use the functions and operators provided by the package.

Value

an initialized `ondisc_matrix` object.

Slots

`h5_file` path to an initialized .h5 file stored on-disk.

`logical_mat` logical value indicating whether the matrix is logical.

`cell_subset` integer vector recording the cells currently in use.

`feature_subset` integer vector recording the features currently in use.

`underlying_dimension` the dimension of the (unsubset) expression matrix.

Examples

```
# NOTE: You must create the HDF5 file "expressions.h5" to run this example.
# Navigate to the help file of "create_ondisc_matrix_from_mtx"
# (via ?create_ondisc_matrix_from_mtx), and execute the code in the first code block.
h5_fp <- paste0(tempdir(), "/expressions.h5")
if (file.exists(h5_fp)) {
  odm <- ondisc_matrix(h5_file = h5_fp)
}
```

---

show *Print basic information to the console*

Description

Print basic information to the console

Usage

```
## S4 method for signature 'ondisc_matrix'
show(object)

## S4 method for signature 'metadata_ondisc_matrix'
show(object)

## S4 method for signature 'multimodal_ondisc_matrix'
show(object)
```

Arguments

object	an object of class ondisc_matrix, covaraite_ondisc_matrix, or multimodal_ondisc_matrix

Value

NULL; called for printing

Examples

```
# NOTE: You must create the HDF5 file "expressions.h5" to run this example.
# Navigate to the help file of "create_ondisc_matrix_from_mtx"
# (via ?create_ondisc_matrix_from_mtx), and execute the code in the first code block.

h5_fp <- paste0(tempdir(), "/expressions.h5")
if (file.exists(h5_fp)) {
odm <- ondisc_matrix(h5_file = h5_fp)
show(odm)
}
```

subset-odm *Subset using the* [*operator.*

Description

Apply the [operator to an ondisc_matrix, metadata_ondisc_matrix, or multimodal_ondisc_matrix to subset the object. You can pass logical, character, or numeric vectors to [; character vectors are assumed to refer to feature IDs (for rows) and cell barcodes (for columns).

Usage

```
## S4 method for signature 'ondisc_matrix,missing,missing,missing'
x[i, j, drop]

## S4 method for signature 'ondisc_matrix,ANY,missing,missing'
x[i, j]

## S4 method for signature 'ondisc_matrix,missing,ANY,missing'
x[i, j]

## S4 method for signature 'ondisc_matrix,ANY,ANY,missing'
x[i, j]

## S4 method for signature 'metadata_ondisc_matrix,ANY,ANY,missing'
x[i, j, drop]

## S4 method for signature 'metadata_ondisc_matrix,ANY,missing,missing'
x[i, j, drop]

## S4 method for signature 'metadata_ondisc_matrix,missing,ANY,missing'
x[i, j, drop]

## S4 method for signature 'metadata_ondisc_matrix,missing,missing,missing'
x[i, j, drop]

## S4 method for signature 'multimodal_ondisc_matrix,missing,missing,missing'
x[i, j, drop]

## S4 method for signature 'multimodal_ondisc_matrix,missing,ANY,missing'
x[i, j, drop]

## S4 method for signature 'multimodal_ondisc_matrix,ANY,ANY,ANY'
x[i, j, drop]
```

Arguments

x	an ondisc_matrix, metadata_ondisc_matrix, or multimodal_ondisc_matrix object.

i	a vector (numeric, logical, or character) indicating features to keep.
j	a vector (numeric, logical, or character) indicating cells to keep.
drop	not used

Details

You can subset an ondisc_matrix and a metadata_ondisc_matrix by cell and/or feature. You can subset a multimodal_ondisc_matrix by cell only (because the features differ across modalities).

Value

An appropriately subset object of the same class as x.

Examples

```
# NOTE: You must create the RDS files "expressions.rds" and
# "perturbations.rds" to run this example. Navigate to the help file of
# "create_ondisc_matrix_from_mtx" (via ?create_ondisc_matrix_from_mtx),
# and execute both code blocks.

# subset an ondisc_matrix
h5_fp <- paste0(tempdir(), "/expressions.h5")
if (file.exists(h5_fp)) {
odm <- ondisc_matrix(h5_file = h5_fp)
# keep cells 100-110
x <- odm[,100:110]
# keep all cells except 50, 100, 150
x <- odm[,-c(50, 100, 150)]
# keep genes ENSG00000188305, ENSG00000257284, and ENSG00000251655:
x <- odm[c("ENSG00000188305", "ENSG00000257284", "ENSG00000251655"),]
# keep the cells CTTAGGACACTGGCGT-1 and AAAGGATTCACATCAG-1:
x <- odm[,c("CTTAGGACACTGGCGT-1", "AAAGGATTCACATCAG-1")]
# keep all genes except ENSG00000188305 and ENSG00000257284
x <- odm[!(get_feature_ids(odm) %in% c("ENSG00000188305", "ENSG00000257284")),]
}

# subset a metadata_ondic_matrix
expressions_fp <- paste0(tempdir(), "/expressions.rds")
if (file.exists(expressions_fp)) {
expressions <- readRDS(expressions_fp)
# keep cells 100-110
x <- expressions[,100:110]
# keep genes ENSG00000188305, ENSG00000257284, and ENSG00000251655
x <- expressions[c("ENSG00000188305", "ENSG00000257284", "ENSG00000251655"),]
}

# subset a multimodal ondisc_matrix
expression_fp <- paste0(tempdir(), "/expressions.rds")
perturbations_fp <- paste0(tempdir(), "/perturbations.rds")
if (file.exists(expression_fp) && file.exists(perturbations_fp)) {
    expressions <- readRDS(expression_fp)
    perturbations <- readRDS(expression_fp)
```

```
    crispr_experiment <- multimodal_ondisc_matrix(list(expressions = expressions,
    perturbations = perturbations))
    # Keep all cells except 10,100, and 105.
    x <- crispr_experiment[,-c(10,100,105)]
    # Keep the first 5 cells
    x <- crispr_experiment[,1:5]
}
```

Index