Package 'scMitoMut'

May 25, 2024

Title Single-cell Mitochondrial Mutation Analysis Tool

Version 1.1.0

Description This package is designed for analyzing mitochondrial mutations using single-cell sequencing data, such as scRNASeq and scATACSeq (preferably the latter due to RNA editing issues). It includes functions for mutation filtering and visualization. In the future, the visualization tool will become an independent package. Mutation filtering is performed by fitting a statistical model to account for various sources of noise, including PCR error, sequencing error, mtDNA sampling and/or heteroplasmy dynamics. The model tests whether the observed allele frequency of a locus in a cell can be explained by the noise model. If not, we classify it as a mutation. The input for this analysis is the allele frequency. The noise model consists of three independent models: binomial, binomial-mixture, and beta-binomial models.

License Artistic-2.0

Encoding UTF-8

Depends R (>= 4.3.0)

Imports data.table, Rcpp, magrittr, plyr, stringr, utils, stats, methods, ggplot2, pheatmap, zlibbioc, RColorBrewer, rhdf5, readr, parallel, grDevices

LinkingTo Rcpp, RcppArmadillo

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Suggests testthat (>= 3.0.0), BiocStyle, knitr, rmarkdown, VGAM, R.utils

Config/testthat/edition 3

BugReports https://github.com/wenjie1991/scMitoMut/issues

URL http://github.com/wenjie1991/scMitoMut

VignetteBuilder knitr

NeedsCompilation yes

biocViews Preprocessing, Sequencing, SingleCell

git_url https://git.bioconductor.org/packages/scMitoMut

git_branch devel

git_last_commit 954512a

git_last_commit_date 2024-04-30

Repository Bioconductor 3.20

Date/Publication 2024-05-24

Author Wenjie Sun [cre, aut] (<https://orcid.org/0000-0002-3100-2346>), Leila Perie [ctb]

Maintainer Wenjie Sun <sunwjie@gmail.com>

Contents

`export_dt` *Export the mutation matrix*

Description

The helper functions to export the mutation results for further analysis. The output format can be data.frame, data.table or matrix for p value, allele frequency or binary mutation status.

Usage

```
export_dt(mtmutObj, percent_interp = 1, n_interp = 3, all_cell = FALSE)

export_df(mtmutObj, ...)

export_pval(mtmutObj, memoSort = TRUE, ...)

export_binary(mtmutObj, memoSort = TRUE, ...)

export_af(mtmutObj, memoSort = TRUE, ...)
```

Arguments

mtmutObj	The scMtioMut object.
percent_interp	A numeric value, the overlapping percentage threshold for triggering interpolation. The default is 1, which means no interpolation.
n_interp	A integer value, the minimum number of overlapped cells with mutation for triggering interpolation, the default is 3.
all_cell	A boolean to indicate whether to include all cells or only cells with mutation. By default, only cells with mutation are included.
...	Other parameters passed to export_dt or export_df.
memoSort	A boolean to indicate whether to sort the loci by mutation frequency. The default is TRUE, the advanced user will know when to set it to FALSE.

Value

data.frame, data.table or matrix or p

Examples

```
## Use the example data
f <- system.file("extdata", "mini_dataset.tsv.gz", package = "scMitoMut")

## Create a temporary h5 file
## In real case, we keep the h5 in project folder for future use
f_h5_tmp <- tempfile(fileext = ".h5")

## Load the data with parse_table function
f_h5 <- parse_table(f, sep = "\t", h5_file = f_h5_tmp)
## open the h5 file and create a mtmutObj object
x <- open_h5_file(f_h5)
run_model_fit(x)
x <- filter_loc(x, min_cell = 5, model = "bb", p_threshold = 0.05, p_adj_method = "fdr")
x
export_df(x)
export_pval(x)
export_af(x)
export_binary(x)
```

`filter_loc` *Filter mutations*

Description

This function filters the mutations based on the mutation calling model and parameters. The loci passed the filter will be saved in the h5 file, together with the filter parameters.

Usage

```
filter_loc(
  mtmutObj,
  min_cell = 5,
  model = "bb",
  p_threshold = 0.05,
  alt_count_threshold = 0,
  p_adj_method = "fdr"
)
```

Arguments

`mtmutObj`	a mtmutObj object.
`min_cell`	a integer of the minimum number of cells with mutation, the default is 5.
`model`	a string of the model for mutation calling, it can be "bb", "bm" or "bi" which stands for beta binomial, binomial mixture and binomial model respectively, the default is "bb".
`p_threshold`	a numeric of the p-value threshold, the default is 0.05.
`alt_count_threshold`	a integer of the minimum number of alternative base count, the default is 0.
`p_adj_method`	a string of the method for p-value adjustment, . refer to `p.adjust`. The default is "fdr".

Value

a mtmutObj object with loc_pass and loc_filter updated.

Examples

```
## Use the example data
f <- system.file("extdata", "mini_dataset.tsv.gz", package = "scMitoMut")

## Create a temporary h5 file
## In real case, we keep the h5 in project folder for future use
f_h5_tmp <- tempfile(fileext = ".h5")

## Load the data with parse_table function
f_h5 <- parse_table(f, sep = "\t", h5_file = f_h5_tmp)
```

```
## open the h5 file and create a mtmutObj object
x <- open_h5_file(f_h5)
run_model_fit(x)
x <- filter_loc(x, min_cell = 5, model = "bb", p_threshold = 0.05, p_adj_method = "fdr")
x
```

`format.mtmutObj` *Print mtmutObj object*

Description

The print method for mtmutObj object.

Usage

```
## S3 method for class 'mtmutObj'
format(x, ...)

## S3 method for class 'mtmutObj'
print(x, ...)

is.mtmutObj(x)
```

Arguments

`x`	a mtmutObj object.
`...`	other parameters passed to `format` or `print`.

Value

a string

Examples

```
## Use the example data
f <- system.file("extdata", "mini_dataset.tsv.gz", package = "scMitoMut")
## Create a temporary h5 file
## In real case, we keep the h5 in project folder for future use
f_h5_tmp <- tempfile(fileext = ".h5")
## Load the data with parse_table function
f_h5 <- parse_table(f, sep = "\t", h5_file = f_h5_tmp)
f_h5
## open the h5 file and create a mtmutObj object
x <- open_h5_file(f_h5)
x
print(x)
```

get_pval *Get p-value list for single locus*

Description

This function returns the p-value list for a single locus.

Usage

```
get_pval(mtmutObj, loc, model = "bb", method = "fdr")
```

Arguments

mtmutObj	a mtmutObj object.
loc	a string of the locus.
model	a string of the model for mutation calling, it can be "bb", "bm" or "bi" which stands for beta binomial, binomial mixture and binomial model respectively.
method	a string of the method for p-value adjustment, refer to p.adjust.

Value

a vector of p-value for each cell.

Examples

```
## Use the example data
f <- system.file("extdata", "mini_dataset.tsv.gz", package = "scMitoMut")
## Create a temporary h5 file
## In real case, we keep the h5 in project folder for future use
f_h5_tmp <- tempfile(fileext = ".h5")

## Load the data with parse_table function
f_h5 <- parse_table(f, sep = "\t", h5_file = f_h5_tmp)
## open the h5 file and create a mtmutObj object
x <- open_h5_file(f_h5)
run_model_fit(x)
get_pval(x, "chrM.1000", "bb", "fdr")
get_pval(x, "chrM.1000", "bm", "fdr")
get_pval(x, "chrM.1000", "bi", "fdr")
```

`open_h5_file` *Open H5 file*

Description

This function opens the H5 file and create a mtmutObj object.

Usage

```
open_h5_file(h5_file)
```

Arguments

`h5_file`	a string of the h5 file directory

Details

The mtmutObj object is a S3 class for handling mitochondrial mutation data. It contains the following elements:

file a string of the h5 file directory.

h5f H5 file handle.

mut_table allele count table H5 group handle.

loc_list list of available loci.

loc_selected selected loci, the default is all loci.

cell_list list of available cell ids.

cell_selected selected cell ids, the default is all cells.

loc_pass loci passed the filter, the default is NULL

loc_filter filter parameters.

loc_filter$min_cell a integer of the minimum number of cells with mutation, the default is 1.

loc_filter$model a string of the model for mutation calling, it can be "bb", "bm" or "bi", the default is "bb".

loc_filter$p_threshold a numeric of the p-value threshold, the default is 0.05.

loc_filter$p_adj_method a string of the method for p-value adjustment, refer to `p.adjust`, the default is "fdr".

Value

a mtmutObj object

Examples

```
## Use the example data
f <- system.file("extdata", "mini_dataset.tsv.gz", package = "scMitoMut")
## Create a temporary h5 file
## In real case, we keep the h5 in project folder for future use
f_h5_tmp <- tempfile(fileext = ".h5")
## Load the data with parse_table function
f_h5 <- parse_table(f, sep = "\t", h5_file = f_h5_tmp)
f_h5
## open the h5 file and create a mtmutObj object
x <- open_h5_file(f_h5)
x
```

`parse_mgatk` *Load mtGATK output*

Description

This function loads the mtGATK output and save it to a H5 file.

Usage

```
parse_mgatk(dir, prefix, h5_file = "mut.h5")
```

Arguments

`dir`	a string of the mtGATK output `final` fold.
`prefix`	a string of the prefix of the mtGATK output directory.
`h5_file`	a string of the output h5 file directory.

Value

a string of the output h5 file directory.

Examples

```
## Use the allele count table data
f <- system.file("extdata", "mini_dataset.tsv.gz", package = "scMitoMut")
## Create a temporary h5 file
## In real case, we keep the h5 in project folder for future use
f_h5_tmp <- tempfile(fileext = ".h5")
## Load the data with parse_table function
f_h5 <- parse_table(f, sep = "\t", h5_file = f_h5_tmp)
f_h5
## Use the mgatk output
f <- system.file("extdata", "mini_mgatk_out", package = "scMitoMut")
f_h5_tmp <- tempfile(fileext = ".h5")
f_h5 <- parse_mgatk(paste0(f, "/final/"), prefix = "sample", h5_file = f_h5_tmp)
```

```
f_h5
x <- open_h5_file(f_h5)
x
##
```

`parse_table` *Load allele count table*

Description

This function loads the allele count table and save it to a H5 file.

Usage

```
parse_table(file, h5_file = "mut.h5", ...)
```

Arguments

`file`	a string of the allele count table file directory.
`h5_file`	a string of the output h5 file directory.
`...`	other parameters passed to `fread`.

Details

The allele count table should be a data.table with the following columns:

loc a string of the locus

cell_barcode a string of the cell barcode.

fwd_depth a integer of the forward depth.

rev_depth a integer of the reverse depth.

alt a string of the alternative base.

ref a string of the reference base.

coverage a integer of the coverage.

Value

a string of the output h5 file directory.

Examples

```
## Use the example data
f <- system.file("extdata", "mini_dataset.tsv.gz", package = "scMitoMut")

## Create a temporary h5 file
## In real case, we keep the h5 in project folder for future use
f_h5_tmp <- tempfile(fileext = ".h5")

## Load the data with parse_table function
f_h5 <- parse_table(f, sep = "\t", h5_file = f_h5_tmp)
f_h5

## open the h5 file and create a mtmutObj object
x <- open_h5_file(f_h5)
x
```

`plot_af_coverage` *QC plot: 2D scatter plot for coverage ~ AF*

Description

QC plot: 2D scatter plot for coverage ~ AF

Usage

```
plot_af_coverage(
  mtmutObj,
  loc,
  model = NULL,
  p_threshold = NULL,
  alt_count_threshold = NULL,
  p_adj_method = NULL
)
```

Arguments

`mtmutObj`	an object of class "mtmutObj".
`loc`	a string of genome location, e.g. "chrM.200".
`model`	a string of model name, one of "bb", "bm", "bi", the default value is "bb".
`p_threshold`	a numeric value of p-value threshold, the default is 0.05.
`alt_count_threshold`	a numeric value of alternative allele count threshold, the default is NULL, which means use the value in mtmutObj.
`p_adj_method`	a string of p-value adjustment method, the default is FDR.

Value

a ggplot object.

Examples

```
## Use the example data
f <- system.file("extdata", "mini_dataset.tsv.gz", package = "scMitoMut")

## Create a temporary h5 file
## In real case, we keep the h5 in project folder for future use
f_h5_tmp <- tempfile(fileext = ".h5")

## Load the data with parse_table function
f_h5 <- parse_table(f, sep = "\t", h5_file = f_h5_tmp)

# open the h5f file
x <- open_h5_file(f_h5)

# run the model fit
run_model_fit(x)
x

# Filter the loci based on the model fit results
x <- filter_loc(x, min_cell = 5, model = "bb", p_threshold = 0.05, p_adj_method = "fdr")

# plot the locus profile for chrM.200
plot_af_coverage(x, "chrM.204")
```

---

plot_heatmap *Heatmap plot*

Description

Heatmap plot

Usage

```
plot_heatmap(mtmutObj, type = "p", cell_ann = NULL, ann_colors = NULL, ...)
```

Arguments

mtmutObj	an object of class "mtmutObj".
type	a string of plot type, "p" for p-value, "af" for allele frequency.
cell_ann	a data.frame of cell annotation, with rownames as cell barcodes, please refer to pheatmap for details.
ann_colors	a list of colors for cell annotation with cell annotation as names, please refer to pheatmap for details.
...	other parameters for export_df and pheatmap.

Value

The pheatmap output

Examples

```
# load the data
## Use the example data
f <- system.file("extdata", "mini_dataset.tsv.gz", package = "scMitoMut")

## Create a temporary h5 file
## In real case, we keep the h5 in project folder for future use
f_h5_tmp <- tempfile(fileext = ".h5")

## Load the data with parse_table function
f_h5 <- parse_table(f, sep = "\t", h5_file = f_h5_tmp)

# open the h5f file
x <- open_h5_file(f_h5)
# run the model fit
run_model_fit(x)
x
# Filter the loci based on the model fit results
x <- filter_loc(x, min_cell = 5, model = "bb", p_threshold = 0.05, p_adj_method = "fdr")

# set the cell annotation
f <- system.file("extdata", "mini_dataset_cell_ann.csv", package = "scMitoMut")
cell_ann <- read.csv(f, row.names = 1)
# Prepare the color for cell annotation
colors <- c(
  "Cancer Epi" = "#f28482",
  Blood = "#f6bd60"
)
ann_colors <- list("SeuratCellTypes" = colors)

# plot the heatmap for p-value
plot_heatmap(x, type = "p", cell_ann = cell_ann, ann_colors = ann_colors, percent_interp = 0.2)
# plot the heatmap for allele frequency
plot_heatmap(x, type = "af", cell_ann = cell_ann, ann_colors = ann_colors, percent_interp = 0.2)
# plot the heatmap for binary mutation
plot_heatmap(x, type = "binary", cell_ann = cell_ann, ann_colors = ann_colors, percent_interp = 0.2)
```

`process_locus_bmbb` *Fit tree models for one locus*

Description

This function fit binomial mixture model, beta binomial model and calculate the VMR and consistency of fwd rev strand for one locus.

Usage

```
process_locus_bmbb(
  mtmutObj,
  loc,
  dom_allele = NULL,
  return_data = FALSE,
  bb_over_bm = TRUE,
  bb_over_bm_p = 0.05,
  bb_over_bm_adj = "fdr",
  ...
)
```

Arguments

mtmutObj	a mtmutObj object.
loc	string given the locus name (e.g. "chrM1000").
dom_allele	string given the dominant allele (e.g. "A"), if NULL auto detect the dominant allele.
return_data	logical whether to return the allele count data, if FALSE, the data in the return value will be NULL. The default is FALSE.
bb_over_bm	logical weather to use binomial mixture model result to define the wildtype cells for training beta binomial model.
bb_over_bm_p	numeric the binomial mixutre model p value threshold for selecting the wildtype cells for training beta binomial model.
bb_over_bm_adj	string the method for adjusting the binomial mixture p value, default is "fdr".
...	other parameters control the model fitting.

Value

A list of three elements:

data	data.frame of the allele count data.
locus	data.table of the VMR and consistency of fwd rev strand.
model	list of the model fitting results.

Examples

```
## Use the example data
f <- system.file("extdata", "mini_dataset.tsv.gz", package = "scMitoMut")

## Create a temporary h5 file
## In real case, we keep the h5 in project folder for future use
f_h5_tmp <- tempfile(fileext = ".h5")

## Load the data with parse_table function
f_h5 <- parse_table(f, sep = "\t", h5_file = f_h5_tmp)
```

```
x <- open_h5_file(f_h5)
res <- process_locus_bmbb(x, loc = "chrM.1000")
res
```

`rm_mtmutObj` *Remove mtmutObj object*

Description

This function closes the H5 file and remove mtmutObj object. Because the H5 file is not closed automatically when the mtmutObj object is removed. We need to close the H5 file manually. By using this function, we can remove the mtmutObj object and close the H5 file at the same time.

Usage

```
rm_mtmutObj(x, envir = .GlobalEnv)
```

Arguments

x	a mtmutObj object.

Value

no return value.

Examples

```
## Use the example data
f <- system.file("extdata", "mini_dataset.tsv.gz", package = "scMitoMut")
## Create a temporary h5 file
## In real case, we keep the h5 in project folder for future use
f_h5_tmp <- tempfile(fileext = ".h5")
## Load the data with parse_table function
f_h5 <- parse_table(f, sep = "\t", h5_file = f_h5_tmp)
f_h5
## open the h5 file and create a mtmutObj object
x <- open_h5_file(f_h5)
x
rm_mtmutObj(x)
```

`run_model_fit` *Fit binomial mixture model for every candidate locus*

Description

Fit binomial mixture model for every candidate locus

Usage

```
run_model_fit(
  mtmutObj,
  mc.cores = getOption("mc.cores", 1L),
  bb_over_bm = TRUE,
  bb_over_bm_p = 0.05,
  bb_over_bm_adj = "fdr"
)
```

Arguments

`mtmutObj`	a mtmutObj object.
`mc.cores`	integer number of cores to use.
`bb_over_bm`	logical weather to use binomial mixture model result to define the wildtype cells for training beta binomial model.
`bb_over_bm_p`	numeric the binomial mixutre model p value threshold for selecting the wildtype cells for training beta binomial model.
`bb_over_bm_adj`	string the method for adjusting the binomial mixture p value, default is "fdr".

Details

This function will fit three models for every candidate locus:

- binomial mixture model
- beta binomial model
- binomial model

The results are saved in the h5f file.

Value

NULL, the results are saved in the h5f file.

Examples

```
## Use the example data
f <- system.file("extdata", "mini_dataset.tsv.gz", package = "scMitoMut")
## Load the data with parse_table function
f_h5_tmp <- tempfile(fileext = ".h5")
f_h5 <- parse_table(f, sep = "\t", h5_file = f_h5_tmp)
# open the h5f file
x <- open_h5_file(f_h5)
# run the model fit
run_model_fit(x)
x
# Filter the loci based on the model fit results
x <- filter_loc(x, min_cell = 5, model = "bb", p_threshold = 0.05, p_adj_method = "fdr")
x
```

`scMitoMut` *scMitoMut*

Description

scMitoMut is a tool for detecting mitochondrial mutations in single-cell sequencing data.

`subset_cell` *Subset cell and loci*

Description

Functions to subset cell and loci for fitting models and plotting.

Usage

```
subset_cell(mtmutObj, cell_list)

subset_loc(mtmutObj, loc_list)
```

Arguments

`mtmutObj`	a mtmutObj object
`cell_list`	a list of cell barcodes
`loc_list`	a list of loci

Value

a mtmutObj object with cell and loci selected

Examples

```
## Use the example data
f <- system.file("extdata", "mini_dataset.tsv.gz", package = "scMitoMut")

## Create a temporary h5 file
## In real case, we keep the h5 in project folder for future use
f_h5_tmp <- tempfile(fileext = ".h5")

## Load the data with parse_table function
f_h5 <- parse_table(f, sep = "\t", h5_file = f_h5_tmp)

## open the h5 file and create a mtmutObj object
x <- open_h5_file(f_h5)
x
## subset cell and loci
x <- subset_cell(x, x$cell_list[1:10])
x <- subset_loc(x, x$loc_list[1:10])
x
```

Index