Package: presto (via r-universe)

October 14, 2024

Title Fast Functions for Differential Expression using Wilcox and AUC

Version 1.0.0

Description Scalable implementation of the Wilcoxon rank sum test and auROC statistic. Interfaces to dense and sparse matrices, as well as genomics analysis frameworks Seurat and SingleCellExperiment.

License GPL-3

LazyData true

Encoding UTF-8

Depends R (>= 3.4.0), Rcpp, data.table

LinkingTo Rcpp, RcppArmadillo

Imports dplyr, methods, tidyr, purrr, tibble, Matrix, rlang, stats, utils

RoxygenNote 7.2.1

Suggests knitr, rmarkdown, testthat, Seurat, SingleCellExperiment, SummarizedExperiment, broom, BiocStyle, DESeq2

NeedsCompilation yes

VignetteBuilder knitr

Repository https://satijalab.r-universe.dev

RemoteUrl https://github.com/immunogenomics/presto

RemoteRef HEAD

RemoteSha 7636b3d0465c468c35853f82f1717d3a64b3c8f6

Contents

`collapse_counts` *Collapse counts based on multiple categorical metadata columns*

Description

Collapse counts based on multiple categorical metadata columns

Usage

```
collapse_counts(
  counts_mat,
  meta_data,
  varnames,
  min_cells_per_group = 0,
  keep_n = FALSE,
  how = c("sum", "mean")[1]
)
```

Arguments

`counts_mat`	counts matrix where columns represent cells and rows represent features
`meta_data`	data.frame containing cell metadata
`varnames`	subset of ‘meta_data‘ column names
`min_cells_per_group`	minimum cells to keep collapsed group
`keep_n`	keep or drop the ‘N‘ column containing the number of cells in each group. Default is ‘FALSE‘
`how`	method of collapsing counts from groups. ‘sum‘ or ‘mean‘

Examples

```
m <- matrix(sample.int(8, 100*500, replace=TRUE), nrow=100, ncol=500)
rownames(m) <- paste0("G", 1:100)
colnames(m) <- paste0("C", 1:500)
md1 <- sample(c("a", "b"), 500, replace=TRUE)
md2 <- sample(c("c", "d"), 500, replace=TRUE)
df <- data.frame(md1, md2)
data_collapsed <- collapse_counts(m, df, c("md1", "md2"))
head(data_collapsed$counts_mat)
```

---

compute_hash — *Compute unique hash for each row of data.frame*

Description

Compute unique hash for each row of data.frame

Usage

```
compute_hash(data_df, vars_use)
```

Arguments

data_df	data.frame
vars_use	vector of column names to use when computing the hash

---

exprs — *Small gene expression matrix*

Description

Small gene expression matrix

Usage

```
exprs
```

Format

matrix of 25 features by 150 observations

nnzeroGroups *nnzeroGroups*

Description

Utility function to compute number of zeros-per-feature within group

Usage

```
nnzeroGroups(X, y, MARGIN = 2)

## S3 method for class 'dgCMatrix'
nnzeroGroups(X, y, MARGIN = 2)

## S3 method for class 'matrix'
nnzeroGroups(X, y, MARGIN = 2)
```

Arguments

X	matrix
y	group labels
MARGIN	whether observations are rows (=2) or columns (=1)

Value

Matrix of groups by features

Examples

```
data(exprs)
data(y)
nnz_res <- nnzeroGroups(exprs, y, 1)
nnz_res <- nnzeroGroups(t(exprs), y, 2)
```

object_sce *SingleCellExperiment object with fake data*

Description

SingleCellExperiment object with fake data

Usage

```
object_sce
```

Format

A single cell experiment

---

object_seurat — *Seurat V3 object with fake data*

Description

Seurat V3 object with fake data

Usage

```
object_seurat
```

Format

A Seurat object

---

presto — *presto*

Description

Fast differential expression

---

pseudobulk_deseq2 — *Pseudobulk DESeq2*

Description

Pseudobulk DESeq2

Usage

```
pseudobulk_deseq2(
  dge_formula,
  meta_data,
  counts_df,
  verbose = TRUE,
  min_counts_per_sample = 10,
  present_in_min_samples = 5,
  collapse_background = TRUE,
  vals_test = NULL,
  mode = c("one_vs_all", "pairwise", "within")[1]
)
```

Arguments

dge_formula	differential gene expression formula for DESeq2
meta_data	data.frame of cell metadata
counts_df	A feature-by-sample matrix
verbose	verbose
min_counts_per_sample	minimum counts per sample to include in differential gene expression
present_in_min_samples	minimum samples with gene counts to include in differential gene expression
collapse_background	collapse background. Default is ‘TRUE‘
vals_test	cell metadata columns
mode	kind of pseudobulk testing to perform. One of ‘one_vs_all‘, ‘pairwise‘, or ‘within‘

Examples

```
## Not run:
    m <- matrix(sample.int(8, 100*500, replace=TRUE),nrow=100, ncol=500)
    rownames(m) <- paste0("G", 1:100)
    colnames(m) <- paste0("C", 1:500)
    md1 <- sample(c("a", "b"), 500, replace=TRUE)
    md2 <- sample(c("c", "d"), 500, replace=TRUE)
    df <- data.frame(md1, md2)
    data_collapsed <- collapse_counts(m, df, c("md1", "md2"))
    res_mat <- pseudobulk_deseq2(
        ~md1 + md1,
        data_collapsed$meta_data,
        data_collapsed$counts_mat,
        verbose = TRUE,
        present_in_min_samples = 1
    )
    head(res_mat)

## End(Not run)
```

---

pseudobulk_one_vs_all *Pseudobulk one versus all*

Description

Pseudobulk one versus all

Usage

```
pseudobulk_one_vs_all(
  dge_formula,
  counts_df,
  meta_data,
  contrast_var,
  vals_test,
  collapse_background,
  verbose
)
```

Arguments

dge_formula	differential gene expression formula for DESeq2
counts_df	counts matrix
meta_data	data.frame of cell metadata
contrast_var	cell metadata column to use for differential gene expression
vals_test	cell metadata columns
collapse_background	collapse background counts according to ‘contrast_var‘
verbose	verbose

---

pseudobulk_pairwise *Pseudobulk pairwise*

Description

Pseudobulk pairwise

Usage

```
pseudobulk_pairwise(
  dge_formula,
  counts_df,
  meta_data,
  contrast_var,
  vals_test,
  verbose,
  min_counts_per_sample,
  present_in_min_samples
)
```

Arguments

dge_formula	differential gene expression formula for DESeq2
counts_df	counts matrix
meta_data	data.frame of cell metadata
contrast_var	cell metadata column to use for differential gene expression
vals_test	cell metadata columns
verbose	verbose
min_counts_per_sample	minimum counts per sample to include in differential gene expression
present_in_min_samples	minimum samples with gene counts to include in differential gene expression

---

pseudobulk_within *Pseudobulk within*

Description

Pseudobulk within

Usage

```
pseudobulk_within(
  dge_formula,
  counts_df,
  meta_data,
  split_var,
  vals_test,
  verbose,
  min_counts_per_sample,
  present_in_min_samples
)
```

Arguments

dge_formula	differential gene expression formula for DESeq2
counts_df	counts matrix
meta_data	data.frame of cell metadata
split_var	-
vals_test	cell metadata columns
verbose	verbose
min_counts_per_sample	minimum counts per sample to include in differential gene expression
present_in_min_samples	minimum samples with gene counts to include in differential gene expression

rank_matrix *rank_matrix*

Description

Utility function to rank columns of matrix

Usage

```
rank_matrix(X)

## S3 method for class 'dgCMatrix'
rank_matrix(X)

## S3 method for class 'matrix'
rank_matrix(X)
```

Arguments

X	feature by observation matrix.

Value

List with 2 items

- X_ranked - matrix of entry ranks
- ties - list of tied group sizes

Examples

```
data(exprs)
rank_res <- rank_matrix(exprs)
```

sumGroups *sumGroups*

Description

Utility function to sum over group labels

Usage

```
sumGroups(X, y, MARGIN = 2)

## S3 method for class 'dgCMatrix'
sumGroups(X, y, MARGIN = 2)

## S3 method for class 'matrix'
sumGroups(X, y, MARGIN = 2)
```

Arguments

X	matrix
y	group labels
MARGIN	whether observations are rows (=2) or columns (=1)

Value

Matrix of groups by features

Examples

```
data(exprs)
data(y)
sumGroups_res <- sumGroups(exprs, y, 1)
sumGroups_res <- sumGroups(t(exprs), y, 2)
```

---

summarize_dge_pairs *Summarize differential gene expression pairs*

Description

Summarize differential gene expression pairs

Usage

```
summarize_dge_pairs(dge_res, mode = c("min", "max")[1])
```

Arguments

dge_res	table returned by pseudobulk_deseq2() function when ‘mode‘ is ‘pairwise‘
mode	-

top_markers — *Get top n markers from wilcoxauc*

Description

Useful summary of the most distinguishing features in each group.

Usage

```
top_markers(
  res,
  n = 10,
  auc_min = 0,
  pval_max = 1,
  padj_max = 1,
  pct_in_min = 0,
  pct_out_max = 100
)
```

Arguments

res	table returned by wilcoxauc() function.
n	number of markers to find for each.
auc_min	filter features with auc < auc_min.
pval_max	filter features with pval > pval_max.
padj_max	filter features with padj > padj_max.
pct_in_min	Minimum percent (0-100) of observations with non-zero entries in group.
pct_out_max	Maximum percent (0-100) of observations with non-zero entries out of group.

Value

table with the top n markers for each cluster.

Examples

```
data(exprs)
data(y)

## first, run wilcoxauc
res <- wilcoxauc(exprs, y)

## top 10 markers for each group
## filter for nominally significant (p<0.05) and over-expressed (auc>0.5)
top_markers(res, 10, auc_min = 0.5, pval_max = 0.05)
```

top_markers_dds — *Get top n markers from pseudobulk DESeq2*

Description

Useful summary of the most distinguishing features in each group.

Usage

```
top_markers_dds(res, n = 10, pval_max = 1, padj_max = 1, lfc_min = 1)
```

Arguments

res	table returned by pseudobulk_deseq2() function.
n	number of markers to find for each.
pval_max	filter features with pval > pval_max.
padj_max	filter features with padj > padj_max.
lfc_min	filter features with log2FoldChange < lfc_min @return table with the top n markers for each cluster.

wilcoxauc — *Fast Wilcoxon rank sum test and auROC*

Description

Computes auROC and Wilcoxon p-value based on Gaussian approximation. Inputs can be

- Dense matrix or data.frame
- Sparse matrix, such as dgCMatrix
- Seurat V3 object
- SingleCellExperiment object

For detailed examples, consult the presto vignette.

Usage

```
wilcoxauc(X, ...)

## S3 method for class 'seurat'
wilcoxauc(X, ...)

## S3 method for class 'Seurat'
wilcoxauc(
  X,
```

```
  group_by = NULL,
  assay = "data",
  groups_use = NULL,
  seurat_assay = "RNA",
  ...
)

## S3 method for class 'SingleCellExperiment'
wilcoxauc(X, group_by = NULL, assay = NULL, groups_use = NULL, ...)

## Default S3 method:
wilcoxauc(X, y, groups_use = NULL, verbose = TRUE, ...)
```

Arguments

X	A feature-by-sample matrix, Seurat object, or SingleCellExperiment object
...	input specific parameters.
group_by	(Seurat & SCE) name of groups variable ('e.g. Cluster').
assay	(Seurat & SCE) name of feature matrix slot (e.g. 'data' or 'logcounts').
groups_use	(optional) which groups from y vector to test.
seurat_assay	(Seurat) name of Seurat Assay (e.g. 'RNA').
y	vector of group labels.
verbose	boolean, TRUE for warnings and messages.

Value

table with the following columns:

- **feature** - feature name (e.g. gene name).
- **group** - group name.
- **avgExpr** - mean value of feature in group.
- **logFC** - log fold change between observations in group vs out.
- **statistic** - Wilcoxon rank sum U statistic.
- **auc** - area under the receiver operator curve.
- **pval** - nominal p value.
- **padj** - Benjamini-Hochberg adjusted p value.
- **pct_in** - Percent of observations in the group with non-zero feature value.
- **pct_out** - Percent of observations out of the group with non-zero feature value.

Examples

```
## Not run:
 data(exprs)
 data(y)
```

```
## on a dense matrix
head(wilcoxauc(exprs, y))

## with only some groups
head(wilcoxauc(exprs, y, c('A', 'B')))

## on a sparse matrix
exprs_sparse <- as(exprs, 'dgCMatrix')
head(wilcoxauc(exprs_sparse, y))

## on a Seurat V3 object
if (requireNamespace("Seurat", quietly = TRUE)) {
    pkg_version <- packageVersion('Seurat')
    if (pkg_version >= "3.0" & pkg_version < "4.0") {
        data(object_seurat)
        head(wilcoxauc(object_seurat, 'cell_type'))
    }
}

## on a SingleCellExperiment object
if (requireNamespace("SingleCellExperiment", quietly = TRUE)) {
    data(object_sce)
    head(wilcoxauc(object_sce, 'cell_type'))
}

## End(Not run)
```

---

y *Group labels for observations in gene expression matrix*

Description

Group labels for observations in gene expression matrix

Usage

```
y
```

Format

a factor with 3 levels

Index