Package 'TREG'

December 21, 2024

Title Tools for finding Total RNA Expression Genes in single nucleus RNA-seq data

Version 1.11.0

Date 2023-06-16

Description RNA abundance and cell size parameters could improve RNA-seq deconvolution algorithms to more accurately estimate cell type proportions given the different cell type transcription activity levels. A Total RNA Expression Gene (TREG) can facilitate estimating total RNA content using single molecule fluorescent in situ hybridization (smFISH). We developed a data-driven approach using a measure of expression invariance to find candidate TREGs in postmortem human brain single nucleus RNA-seq. This R package implements the method for identifying candidate TREGs from snRNA-seq data.

License Artistic-2.0

URL https://github.com/LieberInstitute/TREG, http://research.libd.org/TREG/

BugReports https://support.bioconductor.org/t/TREG

biocViews Software, SingleCell, RNASeq, GeneExpression, Transcriptomics, Transcription, Sequencing

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Depends R (>= 4.2), SummarizedExperiment

Suggests BiocFileCache, BiocStyle, dplyr, ggplot2, knitr, pheatmap, sessioninfo, RefManageR, rmarkdown, testthat (>= 3.0.0), tibble, tidyr, SingleCellExperiment

Config/testthat/edition 3

Imports Matrix, purrr, rafalib

VignetteBuilder knitr

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

git_branch devel

git_last_commit b9d8f58

git_last_commit_date 2024-12-10

Repository Bioconductor 3.21

Date/Publication 2024-12-20

Author Louise Huuki-Myers [aut, cre] (ORCID:
<https://orcid.org/0000-0001-5148-3602>),
Leonardo Collado-Torres [ctb] (ORCID:
<https://orcid.org/0000-0003-2140-308X>)

Maintainer Louise Huuki-Myers <lahuuki@gmail.com>

Contents

`filter_prop_zero` *Filter Genes for by Max Proportion Zero Among Groups*

Description

This function uses the `data.frame()` generated to find the maximum Proportion Zero across groups, then filter to a set of genes that pass the max Prop Zero cutoff defined by the user.

Usage

```
filter_prop_zero(prop_zero_df, cutoff = 0.9, na.rm = TRUE)
```

Arguments

`prop_zero_df`	`data.frame()` containing proportion of zero counts, genes as rows, groups as columns.
`cutoff`	A `numeric()` cutoff value for maximum Proportion Zero. The cutoff value should be < 1.
`na.rm`	a logical indicating whether missing values should be removed when max prop_zero is calculated.

Value

A `character()` of gene names that are under the cutoff. These names are from the `rownames()` of the expression data.

See Also

Other Proportion Zero functions: `get_prop_zero()`

Examples

```
## Get Proportion Zero data.frame
prop_zero <- get_prop_zero(sce_zero_test, group_col = "group")

## Filter with max Proportion Zero cutoff = 0.59
filter_prop_zero(prop_zero, cutoff = 0.59)

## NA handling
prop_zero[1, 1] <- NA

## If NA are present in the prop_zero_df the user can choose to:

## Remove NAs (default)
## genes with NA counts may be included in the final list
filter_prop_zero(prop_zero, cutoff = 0.59, na.rm = TRUE)

## Include NA values
## any gene with genes with NA counts (so max prop_zero will be NA)
## will be removed from the final list, this will throw warning
filter_prop_zero(prop_zero, cutoff = 0.59, na.rm = FALSE)
```

---

get_prop_zero *Get the Proportion of Zero Counts for Each Gene in Each Group*

Description

This function calculates the Proportion Zero for each gene in each user defined group. Proportion Zero = number of zero counts for a gene for a group of cells/number of cells in the group.

Usage

```
get_prop_zero(sce, group_col = "cellType")
```

Arguments

`sce`	SummarizedExperiment-class object
`group_col`	name of the column in the colData() of `sce` that defines the group of interest.

Details

For more information about calculating Proportion Zero, check equation 1 from the vignette in section "Calculate Proportion Zero and Pick Cutoff".

Value

A `data.frame()` containing proportion of zero counts, genes as rows, groups as columns.

See Also

Other Proportion Zero functions: `filter_prop_zero()`

Examples

```
## Basic Proportion counts == 0
rowSums(assays(sce_zero_test)$counts == 0) / ncol(sce_zero_test)

## Get proportion by the default group "cellType"
get_prop_zero(sce_zero_test)

## Get proportion by user defined grouping of the data
get_prop_zero(sce_zero_test, group_col = "group")

## Groups with missing levels will be dropped
get_prop_zero(sce_zero_test, group_col = "cellType_na")
```

---

rank_cells *Get the Rank of the Expression for each Gene in each Cell*

Description

This function finds the rank of each gene's expression for each cell, grouped by the user defined variable. This data is used to compute the rank invariance value for each gene later with `rank_invariance()`.

Usage

```
rank_cells(sce, group_col = "cellType", assay = "logcounts")
```

Arguments

`sce`	SummarizedExperiment-class object with the `assay` (defaults to `logcounts`).
`group_col`	name of the column in the colData() of `sce` that defines the group of interest.
`assay`	A `character(1)` specifying the name of the assay() in the `sce` object to use to rank expression values. Defaults to `logcounts` since it typically contains the normalized expression values.

Value

A named `list()` of `matrix()` objects. Each `matrix()` contains the rank values for the cells belonging to one group.

See Also

Other invariance functions: `rank_group()`, `rank_invariance_express()`, `rank_invariance()`

Examples

```
## Rank the genes for each cell, organized by "group" column
rank_cells(sce_zero_test, group_col = "group")
```

`rank_group` *Get the Rank of the Mean expression for each Gene in each Group*

Description

This function finds the rank of each gene's mean expression all cells in a group. This data is used to compute the rank invariance value for each gene later with `rank_invariance()`.

Usage

```
rank_group(sce, group_col = "cellType", assay = "logcounts")
```

Arguments

`sce`	SummarizedExperiment-class object with the `assay` (defaults to `logcounts`).
`group_col`	name of the column in the colData() of `sce` that defines the group of interest.
`assay`	A `character(1)` specifying the name of the assay() in the `sce` object to use to rank expression values. Defaults to `logcounts` since it typically contains the normalized expression values.

Value

Named `list()` of ranks for each gene.

See Also

Other invariance functions: `rank_cells()`, `rank_invariance_express()`, `rank_invariance()`

Examples

```
## Rank the genes for each group defined by "group" column
rank_group(sce_zero_test, group_col = "group")
```

rank_invariance *Calculate the Rank Invariance of Each Gene from Cell and Group Ranks*

Description

This function computes the Rank Invariance value for each gene, from the cell and group ranks computed by `rank_cells()` and `rank_group()` respectively. Genes with high RI values are considered good candidate TREGs.

Usage

```
rank_invariance(group_rank, cell_rank)
```

Arguments

`group_rank`	A `data.frame()` created with `rank_group()`.
`cell_rank`	A `data.frame()` created with `rank_cells()`.

Value

A `numeric()` with the rank of invariance for each gene. High values represent low Rank Invariance, these genes are considered good candidate TREGs.

See Also

Other invariance functions: `rank_cells()`, `rank_group()`, `rank_invariance_express()`

Examples

```
## Get the rank of the gene in each group
group_rank_test <- rank_group(sce_zero_test, group_col = "group")

## Get the rank of the gene for each cell
cell_rank_test <- rank_cells(sce_zero_test, group_col = "group")

## Use both rankings to calculate rank_invariance
rank_invar_test <- rank_invariance(group_rank_test, cell_rank_test)
## Highest RI value is best candidate TREG
sort(rank_invar_test, decreasing = TRUE)
```

rank_invariance_express *Calculate the Rank Invariance of Each Gene from SCE object*

Description

This function computes the Rank Invariance value for each gene, over the groups defined by the user. This function computes the same RI values as running `rank_cells()` + `rank_group()` + `rank_invariance()`. Genes with high RI values are considered good candidate TREGs. This function is more efficient than running the previous three functions.

Usage

```
rank_invariance_express(sce, group_col = "cellType", assay = "logcounts")
```

Arguments

`sce`	SummarizedExperiment-class object with the `assay` (defaults to `logcounts`).
`group_col`	name of the column in the colData() of `sce` that defines the group of interest.
`assay`	A `character(1)` specifying the name of the assay() in the `sce` object to use to rank expression values. Defaults to `logcounts` since it typically contains the normalized expression values.

Value

A `numeric()` with the rank of invariance for each gene. High values represent low Rank Invariance, these genes are considered good candidate TREGs.

See Also

Other invariance functions: `rank_cells()`, `rank_group()`, `rank_invariance()`

Examples

```
## Calculate RI for the sce object
## Highest RI value is best candidate TREG, and can change based on the grouping of interest
rank_invariance_express(sce_zero_test, group_col = "group")
rank_invariance_express(sce_zero_test, group_col = "cellType")
```

`sce_zero_test` *Test SummarizedExperiment data*

Description

A simulated SummarizedExperiment-class object representing the expression of 100 cells across 5 genes. This object is used as an example dataset throughout `TREG`.

Usage

```
sce_zero_test
```

Format

A SummarizedExperiment-class object with 5 rows and 100 columns.

Details

The `colData` of the object contains demo sample information about the cell, that were designed to be used as `group_col`, (such as `cell_type`, `donor`, and `region`).

The expression of the 5 genes were designed to show a range of Proportion Zeros across groups for different genes. The overall `prop_zero` for these theoretical genes are:

- g100: `prop_zero` = `1.00`
- g50: `prop_zero` = `0.50`
- g0: `prop_zero` = `0.00`
- gOffOn: `prop_zero` = `0.50`
- gVar: `prop_zero` = `0.54`

Index