Package 'DifferentialRegulation'

October 18, 2022

Type Package

Title Differentially regulated genes from scRNA-seq data

Version 1.0.7

Description DifferentialRegulation is a method for detecting differentially regulated genes between two groups of samples (e.g., healthy vs. disease, or treated vs. untreated samples), by targeting differences in the balance of spliced and unspliced mRNA abundances, obtained from single-cell RNA-sequencing (scRNA-seq) data.
DifferentialRegulation accounts for the sample-to-sample variability, and embeds multiple samples in a Bayesian hierarchical model.
In particular, when reads are compatible with multiple genes or multiple splicing versions of a gene (unspliced spliced or ambiguous),
the method allocates these multi-mapping reads to the gene of origin and their splicing version.
Parameters are inferred via Markov chain Monte Carlo (MCMC) techniques (Metropolis-within-Gibbs).

biocViews DifferentialSplicing, Bayesian, Genetics, RNASeq, Sequencing, DifferentialExpression, GeneExpression, MultipleComparison, Software, Transcription, StatisticalMethod, Visualization, SingleCell, GeneTarget

License GPL-3

Depends R (>= 4.2.0)

Imports methods, Rcpp, doRNG, MASS, data.table, doParallel, parallel, foreach, stats, BANDITS, Matrix, SingleCellExperiment, SummarizedExperiment, ggplot2

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown, testthat, BiocStyle

SystemRequirements C++11

VignetteBuilder knitr

RoxygenNote 7.2.0

ByteCompile true

URL https://github.com/SimoneTiberi/DifferentialRegulation

BugReports https://github.com/SimoneTiberi/DifferentialRegulation/issues

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

git_branch RELEASE_3_15

git_last_commit 013f193

git_last_commit_date 2022-07-25

Date/Publication 2022-10-18

Author Simone Tiberi [aut, cre] (<https://orcid.org/0000-0002-3054-9964>)

Maintainer Simone Tiberi <simone.tiberi@uzh.ch>

R topics documented:

DifferentialRegulation-package 2
compute_PB_counts 3
DifferentialRegulation 5
load_EC 8
load_USA 10
plot_pi 11

Index **14**

`DifferentialRegulation-package`
Differentially regulated genes from scRNA-seq data

Description

DifferentialRegulation is a method for detecting differentially regulated genes between two groups of samples (e.g., healthy vs. disease, or treated vs. untreated samples), by targeting differences in the balance of spliced and unspliced mRNA abundances, obtained from single-cell RNA-sequencing (scRNA-seq) data. DifferentialRegulation accounts for the sample-to-sample variability, and embeds multiple samples in a Bayesian hierarchical model. In particular, when reads are compatible with multiple genes or multiple splicing versions of a gene (unspliced spliced or ambiguous), the method allocates these multi-mapping reads to the gene of origin and their splicing version. Parameters are inferred via Markov chain Monte Carlo (MCMC) techniques (Metropolis-within-Gibbs).

Details

The DESCRIPTION file: This package was not yet installed at build time.
Questions relative to DifferentialRegulation should be reported as a new issue at https://github.com/SimoneTiberi/Differential

To access the vignettes, type: `browseVignettes("DifferentialRegulation")`.

Index: This package was not yet installed at build time.

Author(s)

Simone Tiberi <simone.tiberi@uzh.ch>

compute_PB_counts *Discover differentially regulated genes*

Description

compute_PB_counts computese the pseudo-bulk (PB) counts, needed to perform differential testing by DifferentialRegulation.

Usage

```
compute_PB_counts(
  sce,
  EC_list = NULL,
  design,
  sample_col_name = "sample",
  group_col_name = "group",
  sce_cluster_name = "cell_type",
  min_cells_per_cluster = 100,
  min_counts_per_gene_per_group = 20,
  min_counts_ECs = 0
)
```

Arguments

sce	a SingleCellExperiment object, computed via load_USA.
EC_list	a list, computed via load_EC.
design	a data.frame indicating the design of the experiment with one row for each sample; 'design' must contain a column with the sample id and one with the group id.
sample_col_name	a character ("sample" by default), indicating the column name of the 'design' element which stores the sample id.
group_col_name	a character ("group" by default), indicating the column name of the 'design' element which stores the group id.
sce_cluster_name	a character ("cell_type" by default), indicating the name of the 'colData(sce)' element, which stores the cluster id of each cell (i.e., colData(sce)$name_cluster).
min_cells_per_cluster	cell cluster (e.g., cell-type) filter. 'min_cells_per_cluster' is the minimum number of cells, across all samples and groups, for a cell cluster to be considered. Cell clusters with less than 'min_cells_per_cluster' cells will not be analyzed.
min_counts_per_gene_per_group	minimum number of counts per gene, in each cell, across all samples of every group. In each cell cluster, only genes with at least 'min_counts_per_gene_per_group' counts in both groups of samples will be analyzed.

`min_counts_ECs` equivalence classes (ECs) filter (NB: only used when 'EC_list' is provided) 'min_counts_ECs' indicates the minimum number of counts (across all cells in a cell cluster) for each equivalence class; by default all ECs are considered (min_counts_ECs = 0). ECs with less or equal than 'min_counts_ECs' will be discarded. Increasing 'min_counts_ECs' will marginally decrease computational cost computational at the cost of a marginal loss in performance.

Value

A `list` of objects required perform differential testing by `DifferentialRegulation`.

Author(s)

Simone Tiberi <simone.tiberi@uzh.ch>

See Also

`load_EC`, `load_USA`, `plot_pi`,

Examples

```
# load internal data to the package:
data_dir = system.file("extdata", package = "DifferentialRegulation")

# specify samples ids:
sample_ids = paste0("organoid", c(1:3, 16:18))
# set directories of each sample input data (obtained via alevin-fry):
base_dir = file.path(data_dir, "alevin-fry", sample_ids)
file.exists(base_dir)

# set paths to USA counts, cell id and gene id:
# Note that alevin-fry needs to be run with '--use-mtx' option
# to store counts in a 'quants_mat.mtx' file.
path_to_counts = file.path(base_dir,"/alevin/quants_mat.mtx")
path_to_cell_id = file.path(base_dir,"/alevin/quants_mat_rows.txt")
path_to_gene_id = file.path(base_dir,"/alevin/quants_mat_cols.txt")

# load USA counts:
sce = load_USA(path_to_counts,
               path_to_cell_id,
               path_to_gene_id,
               sample_ids)

# define the design of the study:
design = data.frame(sample = sample_ids,
                    group = c( rep("3 mon", 3), rep("6 mon", 3) ))
design

# cell types should be assigned to each cell;
# here we load pre-computed cell types:
path_to_DF = file.path(data_dir,"DF_cell_types.txt")
DF_cell_types = read.csv(path_to_DF, sep = "\t", header = TRUE)
```

```
matches = match(colnames(sce), DF_cell_types$cell_id)
sce$cell_type = DF_cell_types$cell_type[matches]

PB_counts = compute_PB_counts(sce = sce,
                              EC_list = NULL,
                              design =  design,
                              sample_col_name = "sample",
                              group_col_name = "group",
                              sce_cluster_name = "cell_type",
                              min_cells_per_cluster = 100,
                              min_counts_per_gene_per_group = 20)
```

`DifferentialRegulation`

Discover differentially regulated genes

Description

`DifferentialRegulation` identified differentially regulated genes between two conditions (e.g., healthy vs. disease or treated vs. untreated) in each cluster of cells. Parameters are inferred via Markov chain Monte Carlo (MCMC) techniques and a differential testing is performed via a multivariate Wald test on the posterior densities of the group-level USA (Unspliced, Spliced and Ambiguous) counts relative abundance.

Usage

```
DifferentialRegulation(
  PB_counts,
  EC = TRUE,
  n_cores = NULL,
  N_MCMC = 2000,
  burn_in = 500
)
```

Arguments

`PB_counts`	a `list`, computed via `compute_PB_counts`
`EC`	a logical, indicating whether to use equivalence classes (if TRUE, default) or USA estimated counts (if FALSE).
`n_cores`	the number of cores to parallelize the tasks on. Since parallelization is at the cluster level (each cluster is parallelized on a thread), we suggest setting n_cores to the number of clusters (e.g., cell-types), as set by default if 'n_cores' is not specified.
`N_MCMC`	the number of iterations for the MCMC algorithm (including burn-in). Min 2*10^3. If our algorithm does not converge (according to Heidelberger and Welch's convergence diagnostic), we automatically double N_MCMC and burn_in, and run it a second time (a message will be printed on screen to inform users).

burn_in — the length of the burn-in; i.e., the initial part of the MCMC chain to be discarded (before convergence is reached). Min 500. If no convergence is reached, the 'burn_in' is authomatically increased (up to N_MCMC/2) according to the convergence detected by Heidelberger and Welch's convergence diagnostic. If our algorithm does not converge even after increasing the burn-in, we automatically double N_MCMC and burn_in, and run it a second time (a message will be printed on screen to inform users).

Value

A `list` of 4 `data.frame` objects. 'Differential_results' contains results from differential testing only; 'US_results' has results for the proportion of Spliced and Unspliced counts (Ambiguous counts are allocated 50:50 to Spliced and Unspliced); 'USA_results' includes results for the proportion of Spliced, Unspliced and Ambiguous counts (Ambiguous counts are reported separately from Spliced and Unspliced counts); 'Convergence_results' contains information about convergence of posterior chains. Columns 'Gene_id' and 'Cluster_id' contain the gene and cell-cluster name, while 'p_val', 'p_adj.loc' and 'p_adj.glb' report the raw p-values, locally and globally adjusted p-values, via Benjamini and Hochberg (BH) correction. In locally adjusted p-values ('p_adj.loc') BH correction is applied to each cluster separately, while in globally adjusted p-values ('p_adj.glb') BH correction is performed to the results from all clusters. Columns 'pi' and 'sd' indicate the proportion and standard deviation, respectively, 'S', 'U' and 'A' refer to Spliced, Unspliced and Ambiguous counts, respectively, while 'gr_A' and 'gr_B' refer to group A and B, respectively. For instance, columns 'pi_S-gr_A' and 'sd_S-gr_A' indicate the estimates and standard deviation (sd) for the proportion of Spliced (pi_S) and Unspliced (pi_U) counts in group A, respectively.

Author(s)

Simone Tiberi <simone.tiberi@uzh.ch>

See Also

`load_EC`, `load_USA`, `plot_pi`,

Examples

```
# load internal data to the package:
data_dir = system.file("extdata", package = "DifferentialRegulation")

# specify samples ids:
sample_ids = paste0("organoid", c(1:3, 16:18))
# set directories of each sample input data (obtained via alevin-fry):
base_dir = file.path(data_dir, "alevin-fry", sample_ids)
file.exists(base_dir)

# set paths to USA counts, cell id and gene id:
# Note that alevin-fry needs to be run with '--use-mtx' option
# to store counts in a 'quants_mat.mtx' file.
path_to_counts = file.path(base_dir,"/alevin/quants_mat.mtx")
path_to_cell_id = file.path(base_dir,"/alevin/quants_mat_rows.txt")
path_to_gene_id = file.path(base_dir,"/alevin/quants_mat_cols.txt")
```

```
# load USA counts:
sce = load_USA(path_to_counts,
               path_to_cell_id,
               path_to_gene_id,
               sample_ids)

# define the design of the study:
design = data.frame(sample = sample_ids,
                    group = c( rep("3 mon", 3), rep("6 mon", 3) ))
design

# cell types should be assigned to each cell;
# here we load pre-computed cell types:
path_to_DF = file.path(data_dir,"DF_cell_types.txt")
DF_cell_types = read.csv(path_to_DF, sep = "\t", header = TRUE)
matches = match(colnames(sce), DF_cell_types$cell_id)
sce$cell_type = DF_cell_types$cell_type[matches]

PB_counts = compute_PB_counts(sce = sce,
                              EC_list = NULL,
                              design =  design,
                              sample_col_name = "sample",
                              group_col_name = "group",
                              sce_cluster_name = "cell_type",
                              min_cells_per_cluster = 100,
                              min_counts_per_gene_per_group = 20)

# Differential regulation test based on estimated USA (unspliced, spliced, ambiguous) counts
set.seed(169612)
results_USA = DifferentialRegulation(PB_counts, EC = FALSE)

# DifferentialRegulation returns of a list of 3 data.frames:
# "Differential_results" contains results from differential testing only;
# "US_results" has estimates and standard deviation (SD) for pi_S and pi_U (proportion of Spliced and Unspliced coun
# "USA_results" has estimates and standard deviation (SD) for pi_S, pi_U and pi_A (proportion of Spliced, Unspliced
names(results_USA)

# We visualize differential results:
head(results_USA$Differential_results)

# For improved performance, at a higher computational cost,
# we recommend using equivalence classes (EC) (here not run for computational reasons)
if(FALSE){
  # set paths to EC counts and ECs:
  path_to_EC_counts = file.path(base_dir,"/alevin/geqc_counts.mtx")
  path_to_EC = file.path(base_dir,"/alevin/gene_eqclass.txt.gz")

  # load EC counts:
  EC_list = load_EC(path_to_EC_counts,
                    path_to_EC,
                    path_to_cell_id,
                    path_to_gene_id,
                    sample_ids)
```

```
  PB_counts = compute_PB_counts(sce = sce,
                                EC_list = EC_list,
                                design =  design,
                                sample_col_name = "sample",
                                group_col_name = "group",
                                sce_cluster_name = "cell_type",
                                min_cells_per_cluster = 100,
                                min_counts_per_gene_per_group = 20)

  # to reduce memory usage, we can remove the EC_list object:
  rm(EC_list)

  set.seed(169612)
  results_EC = DifferentialRegulation(PB_counts)

  names(results_EC)

  # We visualize differential results:
  head(results_EC$Differential_results)
}

# plot top (i.e., most significant) result:
# plot USA proportions:
plot_pi(results_USA,
        type = "USA",
        gene_id = results_USA$Differential_results$Gene_id[1],
        cluster_id = results_USA$Differential_results$Cluster_id[1])

# plot US proportions:
plot_pi(results_USA,
        type = "US",
        gene_id = results_USA$Differential_results$Gene_id[1],
        cluster_id = results_USA$Differential_results$Cluster_id[1])
```

---

load_EC *Create a list containing the equivalence classes objects object*

Description

load_EC imports the equivalence classes (computed by alevin-fry), and stores them into a list.

Usage

```
load_EC(
  path_to_EC_counts,
  path_to_EC,
  path_to_cell_id,
  path_to_gene_id,
```

```
  sample_ids
)
```

Arguments

path_to_EC_counts	a vector of length equals to the number of samples: each element indicates the path to the equivalence classes counts of the respective sample (i.e., geqc_counts.mtx file).
path_to_EC	a vector of length equals to the number of samples: each element indicates the path to the equivalence classes of the respective sample (i.e., gene_eqclass.txt.gz file).
path_to_cell_id	a vector of length equals to the number of samples: each element indicates the path to the cell ids of the respective sample (i.e., quants_mat_rows.txt file).
path_to_gene_id	a vector of length equals to the number of samples: each element indicates the path to the gene ids of the respective sample (i.e., quants_mat_cols.txt file).
sample_ids	a vector of length equals to the number of samples: each element indicates the name of the sample.

Value

A `list` object.

Author(s)

Simone Tiberi <simone.tiberi@uzh.ch>

See Also

load_USA, DifferentialRegulation

Examples

```
# load internal data to the package:
data_dir = system.file("extdata", package = "DifferentialRegulation")

# specify samples ids:
sample_ids = paste0("organoid", c(1:3, 16:18))
# set directories of each sample input data (obtained via alevin-fry):
base_dir = file.path(data_dir, "alevin-fry", sample_ids)
file.exists(base_dir)

# set paths to USA counts, cell id, gene id, EC counts and ECs:
# Note that alevin-fry needs to be run with `--use-mtx` option
# to store counts in a `quants_mat.mtx` file.
path_to_counts = file.path(base_dir,"/alevin/quants_mat.mtx")
path_to_cell_id = file.path(base_dir,"/alevin/quants_mat_rows.txt")
path_to_gene_id = file.path(base_dir,"/alevin/quants_mat_cols.txt")
```

```
path_to_EC_counts = file.path(base_dir,"/alevin/geqc_counts.mtx")
path_to_EC = file.path(base_dir,"/alevin/gene_eqclass.txt.gz")

# load EC counts:
EC_list = load_EC(path_to_EC_counts,
                  path_to_EC,
                  path_to_cell_id,
                  path_to_gene_id,
                  sample_ids)
```

---

load_USA *Create a list containing the equivalence classes objects object*

Description

load_USA imports the estimated USA (Unspliced, Spliced and Ambiguous) counts (computed by alevin-fry), and stores them into a SingleCellExperiment object.

Usage

```
load_USA(path_to_counts, path_to_cell_id, path_to_gene_id, sample_ids)
```

Arguments

path_to_counts	a vector of length equals to the number of samples: each element indicates the path to the USA estimated count matrix of the respective sample (i.e., quants_mat.mtx file).
path_to_cell_id	a vector of length equals to the number of samples: each element indicates the path to the cell ids of the respective sample (i.e., quants_mat_rows.txt file).
path_to_gene_id	a vector of length equals to the number of samples: each element indicates the path to the gene ids of the respective sample (i.e., quants_mat_cols.txt file).
sample_ids	a vector of length equals to the number of samples: each element indicates the name of the sample.

Value

A SingleCellExperiment object.

Author(s)

Simone Tiberi <simone.tiberi@uzh.ch>

See Also

load_EC, DifferentialRegulation

Examples

```
# load internal data to the package:
data_dir = system.file("extdata", package = "DifferentialRegulation")

# specify samples ids:
sample_ids = paste0("organoid", c(1:3, 16:18))
# set directories of each sample input data (obtained via alevin-fry):
base_dir = file.path(data_dir, "alevin-fry", sample_ids)
file.exists(base_dir)

# set paths to USA counts, cell id and gene id:
# Note that alevin-fry needs to be run with `--use-mtx` option
# to store counts in a `quants_mat.mtx` file.
path_to_counts = file.path(base_dir,"/alevin/quants_mat.mtx")
path_to_cell_id = file.path(base_dir,"/alevin/quants_mat_rows.txt")
path_to_gene_id = file.path(base_dir,"/alevin/quants_mat_cols.txt")

# load USA counts:
sce = load_USA(path_to_counts,
               path_to_cell_id,
               path_to_gene_id,
               sample_ids)
```

---

plot_pi *Plot the estimated proportions of US or USA counts in each group*

Description

plot_pi plots the posterior means of the proportions of US (if 'type' = 'US') or USA (if 'type' = 'USA') counts, in each group. If 'CI' is TRUE, a profile Wald type confidence interval will also be added; the level of the confidence interval is specified by 'CI_level'.

Usage

```
plot_pi(results, gene_id, cluster_id, type = "USA", CI = TRUE, CI_level = 0.95)
```

Arguments

results	a list of 3 data.frame objects, computed via DifferentialRegulation.
gene_id	a character, indicating the gene to plot.
cluster_id	a character, indicating the cell cluster to plot.
type	a character (either 'SU' or 'SUA'). If "SU", it plots the proportion of Spliced and Unspliced counts (Ambiguous counts are assigned 50:50 to Spliced and Unspliced counts). If "SUA" (default), it plots the proportion of Spliced, Unspliced and Ambiguous counts (Ambiguous counts are kept separately). Note that, although US reads are easier to interpret, USA reads more closely reflect what is being compared between conditions.

CI	a logical ('TRUE' by default), indicating whether to plot a profile Wald type confidence interval around the estimated proportions.
CI_level	a numeric between 0 and 1, indicating the level of the confidence interval.

Value

A ggplot object.

Author(s)

Simone Tiberi <simone.tiberi@uzh.ch>

See Also

DifferentialRegulation

Examples

```
# load internal data to the package:
data_dir = system.file("extdata", package = "DifferentialRegulation")

# specify samples ids:
sample_ids = paste0("organoid", c(1:3, 16:18))
# set directories of each sample input data (obtained via alevin-fry):
base_dir = file.path(data_dir, "alevin-fry", sample_ids)
file.exists(base_dir)

# set paths to USA counts, cell id and gene id:
# Note that alevin-fry needs to be run with '--use-mtx' option
# to store counts in a 'quants_mat.mtx' file.
path_to_counts = file.path(base_dir,"/alevin/quants_mat.mtx")
path_to_cell_id = file.path(base_dir,"/alevin/quants_mat_rows.txt")
path_to_gene_id = file.path(base_dir,"/alevin/quants_mat_cols.txt")

# load USA counts:
sce = load_USA(path_to_counts,
               path_to_cell_id,
               path_to_gene_id,
               sample_ids)

# define the design of the study:
design = data.frame(sample = sample_ids,
                    group = c( rep("3 mon", 3), rep("6 mon", 3) ))
design

# cell types should be assigned to each cell;
# here we load pre-computed cell types:
path_to_DF = file.path(data_dir,"DF_cell_types.txt")
DF_cell_types = read.csv(path_to_DF, sep = "\t", header = TRUE)
matches = match(colnames(sce), DF_cell_types$cell_id)
sce$cell_type = DF_cell_types$cell_type[matches]
```

```
PB_counts = compute_PB_counts(sce = sce,
                              EC_list = NULL,
                              design =  design,
                              sample_col_name = "sample",
                              group_col_name = "group",
                              sce_cluster_name = "cell_type",
                              min_cells_per_cluster = 100,
                              min_counts_per_gene_per_group = 20)

# Differential regulation test based on estimated USA (unspliced, spliced, ambiguous) counts
set.seed(169612)
results_USA = DifferentialRegulation(PB_counts, EC = FALSE)

# DifferentialRegulation returns of a list of 3 data.frames:
# "Differential_results" contains results from differential testing only;
# "US_results" has estimates and standard deviation (SD) for pi_S and pi_U (proportion of Spliced and Unspliced coun
# "USA_results" has estimates and standard deviation (SD) for pi_S, pi_U and pi_A (proportion of Spliced, Unspliced
names(results_USA)

# We visualize differential results:
head(results_USA$Differential_results)

# For improved performance, at a higher computational cost,
# we recommend using equivalence classes (EC) (here not run for computational reasons)
# see help(DifferentialRegulation) examples.

# plot top (i.e., most significant) result:
# plot USA proportions:
plot_pi(results_USA,
        type = "USA",
        gene_id = results_USA$Differential_results$Gene_id[1],
        cluster_id = results_USA$Differential_results$Cluster_id[1])

# plot US proportions:
plot_pi(results_USA,
        type = "US",
        gene_id = results_USA$Differential_results$Gene_id[1],
        cluster_id = results_USA$Differential_results$Cluster_id[1])
```

Index

∗ **package**
DifferentialRegulation-package, 2

compute_PB_counts, 3, *5*

data.frame, *3*, *11*
DifferentialRegulation, *3*, *4*, 5, *9–12*
DifferentialRegulation-package, 2

load_EC, *3*, *4*, *6*, 8, *10*
load_USA, *3*, *4*, *6*, *9*, 10

plot_pi, *4*, *6*, 11