Package 'distinct'

October 24, 2025

Type Package

Title distinct: a method for differential analyses via hierarchical permutation tests

Version 1.21.1

Author Simone Tiberi [aut, cre].

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

Description distinct is a statistical method to perform differential testing between two or more groups of distributions;
differential testing is performed via hierarchical non-parametric permutation tests on the cumulative distribution functions (cdfs) of each sample.
While most methods for differential expression target differences in the mean abundance between conditions,
distinct, by comparing full cdfs, identifies, both, differential patterns involving changes in the mean,
as well as more subtle variations that do not involve the mean (e.g., unimodal vs. bimodal distributions with the same mean).
distinct is a general and flexible tool: due to its fully non-parametric nature,
which makes no assumptions on how the data was generated, it can be applied to a variety of datasets.
It is particularly suitable to perform differential state analyses on single cell data (i.e., differential analyses within sub-populations of cells),
such as single cell RNA sequencing (scRNA-seq) and high-dimensional flow or mass cytometry (HDCyto) data.
To use distinct one needs data from two or more groups of samples (i.e., experimental conditions), with at least 2 samples (i.e., biological replicates) per group.

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

License GPL (>= 3)

Depends R (>= 4.3)

Imports Rcpp, stats, SummarizedExperiment, SingleCellExperiment, methods, Matrix, foreach, parallel, doParallel, doRNG, ggplot2, limma, scater

Suggests knitr, rmarkdown, testthat, UpSetR, BiocStyle

LinkingTo Rcpp, RcppArmadillo

SystemRequirements C++17

VignetteBuilder knitr

RoxygenNote 7.2.3

ByteCompile true

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

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

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

git_branch devel

git_last_commit 27ff38a

git_last_commit_date 2025-10-14

Repository Bioconductor 3.23

Date/Publication 2025-10-24

Contents

distinct-package . . . 2
distinct_test . . . 3
Kang_subset . . . 6
log2_FC . . . 7
plot_cdfs . . . 9
plot_densities . . . 11
res . . . 12
top_results . . . 13

Index **16**

`distinct-package` *distinct: a method for differential analyses via hierarchical permutation tests*

Description

distinct is a statistical method to perform differential testing between two or more groups of distributions; differential testing is performed via hierarchical non-parametric permutation tests on the cumulative distribution functions (cdfs) of each sample. While most methods for differential expression target differences in the mean abundance between conditions, distinct, by comparing full cdfs, identifies, both, differential patterns involving changes in the mean, as well as more subtle variations that do not involve the mean (e.g., unimodal vs. bi-modal distributions with the same mean). distinct is a general and flexible tool: due to its fully non-parametric nature, which makes no assumptions on how the data was generated, it can be applied to a variety of datasets. It is particularly

suitable to perform differential state analyses on single cell data (i.e., differential analyses within sub-populations of cells), such as single cell RNA sequencing (scRNA-seq) and high-dimensional flow or mass cytometry (HDCyto) data. To use distinct one needs data from two or more groups of samples (i.e., experimental conditions), with at least 2 samples (i.e., biological replicates) per group.

Questions relative to distinct should be either written to the Bioconductor support site, tagging the question with 'distinct', or reported as a new issue at BugReports (preferred choice).

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

Author(s)

Simone Tiberi [aut, cre].

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

`distinct_test` *Test for differential state between two groups of samples, based on scRNA-seq data.*

Description

`distinct_test` tests for differential state between two groups of samples.

Usage

```
distinct_test(
  x,
  name_assays_expression = "logcounts",
  name_cluster = "cluster_id",
  name_sample = "sample_id",
  design,
  column_to_test = 2,
  P_1 = 100,
  P_2 = 500,
  P_3 = 2000,
  P_4 = 10000,
  N_breaks = 25,
  min_non_zero_cells = 20,
  n_cores = 1
)
```

Arguments

`x`	a `linkS4class{SummarizedExperiment}` or a `linkS4class{SingleCellExperiment}` object.

`name_assays_expression`
: a character ("logcounts" by default), indicating the name of the assays(x) element which stores the expression data (i.e., assays(x)$name_assays_expression). We strongly encourage using normalized data, such as counts per million (CPM) or log2-CPM (e.g., 'logcounts' as created via `scater::logNormCounts`). In case additional covariates are provided (e.g., batch effects), we highly recommend using log-normalized data, such as log2-CPM (e.g., 'logcounts' as created via `scater::logNormCounts`).

`name_cluster`
: a character ("cluster_id" by default), indicating the name of the colData(x) element which stores the cluster id of each cell (i.e., colData(x)$name_cluster).

`name_sample`
: a character ("sample_id" by default), indicating the name of the colData(x) element which stores the sample id of each cell (i.e., colData(x)$name_sample).

`design`
: a `matrix` or `data.frame` with the design matrix of the study (e.g., built via model.matrix(~batches), design must contain one row per sample, while columns include intercept, group and eventual covariates such as batches. Row names of design must indicate the sample ids, and correspond to the names in colData(x)$name_sample.

`column_to_test`
: indicates the column(s) of the design one wants to test (do not include the intercept).

`P_1`
: the number of permutations to use on all gene-cluster combinations.

`P_2`
: the number of permutations to use, when a (raw) p-value is < 0.1 (500 by default).

`P_3`
: the number of permutations to use, when a (raw) p-value is < 0.01 (2,000 by default).

`P_4`
: the number of permutations to use, when a (raw) p-value is < 0.001 (10,000 by default). In order to obtain a finer ranking for the most significant genes, if computational resources are available, we encourage users to set P_4 = 20,000.

`N_breaks`
: the number of breaks at which to evaluate the comulative density function.

`min_non_zero_cells`
: the minimum number of non-zero cells (across all samples) in each cluster for a gene to be evaluated.

`n_cores`
: the number of cores to parallelize the tasks on (parallelization is at the cluster level: each cluster is parallelized on a thread).

Value

A `data.frame` object. Columns 'gene' 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 in each cluster separately, while in globally adjusted p-values ('p_adj.glb') BH correction is performed to the results from all clusters. Column 'filtered' indicates whether a gene-cluster result was filtered (if TRUE), or analyzed (if FALSE). A gene-cluster combination is filtered when fewer than 'min_non_zero_cells' non-zero cells are available. Filtered results have raw and adjusted p-values equal to 1.

Author(s)

Simone Tiberi <simone.tiberi@uzh.ch>

See Also

plot_cdfs, plot_densities, log2_FC, top_results

Examples

```
# load the input data:
data("Kang_subset", package = "distinct")
Kang_subset

# create the design of the study:
samples = Kang_subset@metadata$experiment_info$sample_id
group = Kang_subset@metadata$experiment_info$stim
design = model.matrix(~group)
# rownames of the design must indicate sample ids:
rownames(design) = samples
design

# Note that the sample names in `colData(x)$name_sample` have to be the same ones as those in `rownames(design)`.
rownames(design)
unique(SingleCellExperiment::colData(Kang_subset)$sample_id)

# In order to obtain a finer ranking for the most significant genes, if computational resources are available, we enc

# The group we would like to test for is in the second column of the design, therefore we will specify: column_to_tes

set.seed(61217)
res = distinct_test(
  x = Kang_subset,
  name_assays_expression = "logcounts",
  name_cluster = "cell",
  design = design,
  column_to_test = 2,
  min_non_zero_cells = 20,
  n_cores = 2)

# We can optionally add the fold change (FC) and log2-FC between groups:
res = log2_FC(res = res,
  x = Kang_subset,
  name_assays_expression = "cpm",
  name_group = "stim",
  name_cluster = "cell")

# Visualize significant results:
head(top_results(res))

# Visualize significant results from a specified cluster of cells:
top_results(res, cluster = "Dendritic cells")
```

```
# By default, results from 'top_results' are sorted by (globally) adjusted p-value;
# they can also be sorted by log2-FC:
top_results(res, cluster = "Dendritic cells", sort_by = "log2FC")

# Visualize significant UP-regulated genes only:
top_results(res, up_down = "UP",
  cluster = "Dendritic cells")

# Plot density and cdf for gene 'ISG15' in cluster 'Dendritic cells'.
plot_densities(x = Kang_subset,
  gene = "ISG15",
  cluster = "Dendritic cells",
  name_assays_expression = "logcounts",
  name_cluster = "cell",
  name_sample = "sample_id",
  name_group = "stim")

 plot_cdfs(x = Kang_subset,
   gene = "ISG15",
   cluster = "Dendritic cells",
   name_assays_expression = "logcounts",
   name_cluster = "cell",
   name_sample = "sample_id",
   name_group = "stim")
```

Kang_subset *Subset from the 'Kang18_8vs8()' object of the* muscData *package.*

Description

Subset from the 'Kang18_8vs8()' object of the muscData package.

Arguments

Kang_subset	contains a SingleCellExperiment object, representing a subset of 6 samples (3 individuals observed across 2 conditions) and 100 genes selected from the 'Kang18_8vs8()' object of the muscData package. Below the code used to subset the original dataset.

Author(s)

Simone Tiberi <simone.tiberi@uzh.ch>

See Also

distinct_test

Examples

```
####################
# Object 'Kang_subset' is generated as follows:
####################
# library(muscData)
# sce = Kang18_8vs8()
#
# library(scater)
# sce = computeLibraryFactors(sce)
# sce = logNormCounts(sce)
# cpm(sce) <- calculateCPM(sce)
#
# Select genes with at least 1000 non-zero cells:
# sce = sce[ rowSums(assays(sce)$counts > 0) >= 1000, ]
#
# randomly select 100 of these genes:
# set.seed(61217)
# sel = sample( rownames(sce), size = 100)
# sce = sce[ rownames(sce) %in% sel, ]
#
# select 3 individuals only:
# ind_selected = levels(factor(colData(sce)$ind))[1:3]
# sce = sce[, sce$ind %in% ind_selected]
#
# make a sample_id column:
# colData(sce)$sample_id = factor(paste(colData(sce)$stim, colData(sce)$ind, sep = "_"))
#
# create an experiment_info object containing sample-group information:
# experiment_info = unique(data.frame(sample_id = colData(sce)$sample_id,
#                                     stim = colData(sce)$stim) )
# metadata(sce)$experiment_info = data.frame(experiment_info, row.names = NULL)
#
# remove unnecessary information to reduce storage space:
# sce$cluster = NULL;
# sce$multiplets = NULL;
# rowData(sce) = NULL;
# colnames(sce) = NULL;
# reducedDim(sce) = NULL
# sce$ind = NULL
# sce$sizeFactor = NULL
# rm assays counts
# assays(sce) = assays(sce)[2:3]
#
# Kang_subset = sce
# save(Kang_subset, file = "Kang_subset.RData")
```

---

log2_FC *Compute FCs and log2-FCs.*

Description

log2_FC extends the results obtained via distinct_test, by computing fold changes (FC) and log2-FC between conditions.

Usage

```
log2_FC(
  res,
  x,
  name_assays_expression = "cpm",
  name_group = "group_id",
  name_cluster = "cluster_id"
)
```

Arguments

res	a data.frame with results as returned from distinct_test.
x	a linkS4class{SummarizedExperiment} or a linkS4class{SingleCellExperiment} object.
name_assays_expression	a character ("cpm" by default), indicating the name of the assays(x) element which stores the expression data (i.e., assays(x)$name_assays_expression). We strongly encourage using normalized data, such as counts per million (CPM). Do not use logarithm transformed data to compute FCs.
name_group	a character ("group_id" by default), indicating the name of the colData(x) element which stores the group id of each cell (i.e., colData(x)$name_group).
name_cluster	a character ("cluster_id" by default), indicating the name of the colData(x) element which stores the cluster id of each cell (i.e., colData(x)$name_cluster).

Value

A data.frame object, extending the results in 'res'. Two additional columns are added: FC_group1/group2 and log2FC_group1/group2, inicating the FC and log2-FC of group1/group2. A FC > 1 (or log2FC > 0) indicates up-regulation of group1 (compared to group2); while a FC < 1 (or log2FC < 0) indicates down-regulation of group1 (compared to group2).

Author(s)

Simone Tiberi <simone.tiberi@uzh.ch>

See Also

distinct_test, top_results

Examples

```
# load pre-computed results (obtaines via `distinct_test`)
data("res", package = "distinct")
# load the input data:
data("Kang_subset", package = "distinct")

# We can optionally add the fold change (FC) and log2-FC between groups:
res = log2_FC(res = res,
  x = Kang_subset,
  name_assays_expression = "cpm",
  name_group = "stim",
  name_cluster = "cell")

# Visualize significant results:
head(top_results(res))
```

`plot_cdfs` *Plot sample-specific CDFs.*

Description

`plot_cdfs` returns a `ggplot` object with the sample-specific cumulative density function (CDF) estimates, for a specified cluster and gene.

Usage

```
plot_cdfs(
  x,
  name_assays_expression = "logcounts",
  name_cluster = "cluster_id",
  name_sample = "sample_id",
  name_group = "group_id",
  cluster,
  gene,
  group_level = FALSE,
  pad = TRUE,
  size = 0.75
)
```

Arguments

`x`	a `SummarizedExperiment` or a `SingleCellExperiment` object.
`name_assays_expression`	a character ("logcounts" by default), indicating the name of the assays(x) element which stores the expression data (i.e., assays(x)$name_assays_expression). We strongly encourage using normalized data, such as counts per million (CPM) or log-CPM.

name_cluster	a character ("cluster_id" by default), indicating the name of the colData(x) element which stores the cluster id of each cell (i.e., colData(x)$name_colData_cluster).
name_sample	a character ("sample_id" by default), indicating the name of the colData(x) element which stores the sample id of each cell (i.e., colData(x)$name_colData_sample).
name_group	a character ("group_id" by default), indicating the name of the colData(x) element which stores the group id of each cell (i.e., colData(x)$name_colData_group).
cluster	a character, indicating the name of the cluster to plot.
gene	a character, indicating the name of the gene to plot.
group_level	a logical, indicating whether to plot group-level (if TRUE) or sample-level curves (if FALSE).
pad	a logical element indicating whether to plot the lines of the CDF when 0 and 1 (TRUE) or not (FALSE).
size	a numeric argument defining the width of lines, passed to stat_ecdf.

Value

A ggplot object.

Author(s)

Simone Tiberi <simone.tiberi@uzh.ch>

See Also

distinct_test, plot_densities

Examples

```
data("Kang_subset", package = "distinct")
Kang_subset

plot_cdfs(x = Kang_subset,
          gene = "ISG15",
          cluster = "Dendritic cells",
          name_assays_expression = "logcounts",
          name_cluster = "cell",
          name_sample = "sample_id",
          name_group = "stim")
```

plot_densities *Plot sample-specific densities.*

Description

plot_densities returns a ggplot object with the sample-specific density estimates, for a specified cluster and gene.

Usage

```
plot_densities(
  x,
  name_assays_expression = "logcounts",
  name_cluster = "cluster_id",
  name_sample = "sample_id",
  name_group = "group_id",
  cluster,
  gene,
  group_level = FALSE,
  adjust = 1,
  size = 0.75
)
```

Arguments

x	a SummarizedExperiment or a SingleCellExperiment object.
name_assays_expression	a character ("logcounts" by default), indicating the name of the assays(x) element which stores the expression data (i.e., assays(x)$name_assays_expression). We strongly encourage using normalized data, such as counts per million (CPM) or log-CPM.
name_cluster	a character ("cluster_id" by default), indicating the name of the colData(x) element which stores the cluster id of each cell (i.e., colData(x)$name_colData_cluster).
name_sample	a character ("sample_id" by default), indicating the name of the colData(x) element which stores the sample id of each cell (i.e., colData(x)$name_colData_sample).
name_group	a character ("group_id" by default), indicating the name of the colData(x) element which stores the group id of each cell (i.e., colData(x)$name_colData_group).
cluster	a character, indicating the name of the cluster to plot.
gene	a character, indicating the name of the gene to plot.
group_level	a logical, indicating whether to plot group-level (if TRUE) or sample-level curves (if FALSE).
adjust	a numeric, representing a multiplicate bandwidth adjustment, argument passed to stat_density.
size	a numeric argument defining the width of lines, passed to stat_density.

Value

A ggplot object.

Author(s)

Simone Tiberi <simone.tiberi@uzh.ch>

See Also

distinct_test, plot_cdfs

Examples

```
data("Kang_subset", package = "distinct")
Kang_subset

plot_densities(x = Kang_subset,
               gene = "ISG15",
               cluster = "Dendritic cells",
               name_assays_expression = "logcounts",
               name_cluster = "cell",
               name_sample = "sample_id",
               name_group = "stim")
```

---

res *Results from* distinct_test *function*

Description

Results from distinct_test function

Arguments

res	contains a data.frame object, with the results obtained applying distinct_test function to Kang_subset dataset. Below the code used to obtain 'res'.

Author(s)

Simone Tiberi <simone.tiberi@uzh.ch>

See Also

distinct_test

Examples

```
# load the input data:
# data("Kang_subset", package = "distinct")
# Kang_subset
#
# create the design of the study:
# samples = Kang_subset@metadata$experiment_info$sample_id
# group = Kang_subset@metadata$experiment_info$stim
# design = model.matrix(~group)
# rownames of the design must indicate sample ids:
# rownames(design) = samples
# design
#
# Note that the sample names in `colData(x)$name_sample` have to be the same ones as those in `rownames(design)`.
# rownames(design)
# unique(SingleCellExperiment::colData(Kang_subset)$sample_id)
#
# In order to obtain a finer ranking for the most significant genes, if computational resources are available, we enc
#
# The group we would like to test for is in the second column of the design, therefore we will specify: column_to_tes
#
# set.seed(61217)
# res = distinct_test(
#   x = Kang_subset,
#   name_assays_expression = "logcounts",
#   name_cluster = "cell",
#   design = design,
#   column_to_test = 2,
#   min_non_zero_cells = 20,
#   n_cores = 2)
#
#   save(res, file = "res.RData")
#   saveRDS(res, file = "res.rds")
```

`top_results` *Filter significant results.*

Description

`top_results` returns the significant results obtained via `distinct_test`.

Usage

```
top_results(
  res,
  cluster = "all",
  significance = 0.01,
  global = TRUE,
  up_down = "both",
```

```
  sort_by = "p_adj.glb"
)
```

Arguments

res	a data.frame with results as returned from distinct_test.
cluster	a character indicating the cluster(s) whose results have to be returned. Results from all clusters are returned by default ("all").
significance	numeric, results with adjusted p-value < significance will be returned.
global	logical indicating whether to filter results according to p_adj.glb (when TRUE), or p_adj.loc (when FALSE).
up_down	a character indicating whether to return: all results ("both" or "BOTH"), only up-regulated results ("up" or "UP") or down-regulated results ("down" or "DOWN"). In 'res', a FC > 1 (or log2FC > 0) indicates up-regulation of group1 (compared to group2); while a FC < 1 (or log2FC < 0) indicates down-regulation of group1 (compared to group2).
sort_by	a character indicating how results should be sorted. Results can be sorted by globally adjusted p-value ("p_adj.glb", default choice), locally adjusted p-value ("p_adj.loc"), raw p-value ("p_val") or (log2)fold-change ("FC" or "log2FC").

Value

A data.frame object. Columns 'gene' 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 in each cluster separately, while in globally adjusted p-values ('p_adj.glb') BH correction is performed to the results from all clusters.

Author(s)

Simone Tiberi <simone.tiberi@uzh.ch>

See Also

distinct_test, log2_FC

Examples

```
# load pre-computed results (obtaines via `distinct_test`)
data("res", package = "distinct")

# Visualize significant results:
head(top_results(res))

# Visualize significant results from a specified cluster of cells:
top_results(res, cluster = "Dendritic cells")

# We can optionally add the fold change (FC) and log2-FC between groups:
# load the input data:
```

```
data("Kang_subset", package = "distinct")

res = log2_FC(res = res,
  x = Kang_subset,
  name_assays_expression = "cpm",
  name_group = "stim",
  name_cluster = "cell")

# By default, results from 'top_results' are sorted by (globally) adjusted p-value;
# they can also be sorted by log2-FC:
top_results(res, cluster = "Dendritic cells", sort_by = "log2FC")

# Visualize significant UP-regulated genes only:
top_results(res, up_down = "UP",
  cluster = "Dendritic cells")
```

Index

∗ **differential distribution,**
distinct-package, 2
∗ **differential expression,**
distinct-package, 2
∗ **differential state,**
distinct-package, 2
∗ **hierarchical permutation tests**
distinct-package, 2
∗
distinct-package, 2

data.frame, *4*, *8*, *12*, *14*
distinct *(*distinct-package*)*, 2
distinct-package, 2
distinct_test, 3, *6*, *8*, *10*, *12–14*

ggplot, *9–12*

Kang_subset, 6, *12*

log2_FC, *5*, 7, *14*

matrix, *4*

plot_cdfs, *5*, 9, *12*
plot_densities, *5*, *10*, 11

res, 12

SingleCellExperiment, *6*, *9*, *11*
stat_density, *11*
stat_ecdf, *10*
SummarizedExperiment, *9*, *11*

top_results, *5*, *8*, 13