Package 'DESpace'

May 9, 2024

Type Package

Title DESpace: a framework to discover spatially variable genes

Version 1.4.0

Description Intuitive framework for identifying spatially variable genes (SVGs) via edgeR,
a popular method for performing differential expression analyses.
Based on pre-annotated spatial clusters as summarized spatial information,
DESpace models gene expression using a negative binomial (NB), via edgeR, with spatial clusters as covariates.
SVGs are then identified by testing the significance of spatial clusters.
The method is flexible and robust, and is faster than the most SV methods.
Furthermore, to the best of our knowledge, it is the only SV approach that allows:
- performing a SV test on each individual spatial cluster, hence identifying the key regions of the tissue affected by spatial variability;
- jointly fitting multiple samples, targeting genes with consistent spatial patterns across replicates.

biocViews Spatial, SingleCell, RNASeq, Transcriptomics,
GeneExpression, Sequencing,
DifferentialExpression,StatisticalMethod, Visualization

License GPL-3

Depends R (>= 4.3.0)

Imports edgeR, limma, dplyr, stats, Matrix, SpatialExperiment,
ggplot2, ggpubr, scales, SummarizedExperiment, S4Vectors,
BiocGenerics, data.table, assertthat, cowplot, ggforce,
ggnewscale, patchwork, BiocParallel, methods

Suggests knitr, rmarkdown, testthat, BiocStyle, ExperimentHub,
concaveman, spatialLIBD, purrr, scuttle, utils

VignetteBuilder knitr

RoxygenNote 7.2.3

ByteCompile true

Encoding UTF-8

URL https://github.com/peicai/DESpace

BugReports https://github.com/peicai/DESpace/issues

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

git_branch RELEASE_3_19

git_last_commit b33dd89

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-09

Author Peiying Cai [aut, cre] (<https://orcid.org/0009-0001-9229-2244>), Simone Tiberi [aut] (<https://orcid.org/0000-0002-3054-9964>)

Maintainer Peiying Cai <peiying.cai@uzh.ch>

Contents

`DESpace` *DESpace: A package for identifying spatially variable genes*

Description

An intuitive framework for identifying spatially variable genes (SVGs) via edgeR, one of the most common methods for performing differential expression analyses.

Details

Based on pre-annotated spatial clusters as summarized spatial information, `DESpace` models gene expression using a negative binomial (NB), via `edgeR`, with spatial clusters as covariates. SVGs are then identified by testing the significance of spatial clusters.

Our approach assumes that the spatial structure can be summarized by spatial clusters, which should reproduce the key features of the tissue (e.g., white matter and layers in brain cortex). These spatial clusters are therefore taken as proxy for the actual spatial distribution; a significant test of these covariates indicates that space influences gene expression, hence identifying spatially variable genes.

Our model is flexible and robust, and is significantly faster than the most SV methods. Furthermore, to the best of our knowledge, it is the only SV approach that allows: - performing a SV test on each individual spatial cluster, hence identifying the key regions affected by spatial variability; - jointly fitting multiple samples, targeting genes with consistent spatial patterns across replicates.

For an overview of the functionality provided by the package, please see the vignette: `vignette("DESpace", package="DESpace")`

Author(s)

Peiying Cai <peiying.cai@uzh.ch>, Simone Tiberi <simone.tiberi@unibo.it>

See Also

DESpace_test, individual_test, top_results, FeaturePlot

DESpace_test *DESpace_test*

Description

'DESpace_test' identifies spatially variable genes (SVGs) from spatially-resolved transcriptomics data, provided spatial clusters are available.

Usage

```
DESpace_test(
  spe,
  spatial_cluster,
  sample_col = NULL,
  replicates = FALSE,
  min_counts = 20,
  min_non_zero_spots = 10,
  filter_gene = TRUE,
  verbose = FALSE
)
```

Arguments

spe	SpatialExperiment or SingleCellExperiment.
spatial_cluster	Column name of spatial clusters in colData(spe).
sample_col	Column name of sample ids in colData(spe).
replicates	A logical, indicating whether biological replicates are provided (TRUE) or not (FALSE). If biological replicates are provided, DESpace_test performs a joint test across all replicates, searching for SVGs with consistent spatial patterns across samples.
min_counts	Minimum number of counts per sample (across all spots) for a gene to be analyzed.
min_non_zero_spots	Minimum number of non-zero spots per sample, for a gene to be analyzed.
filter_gene	A logical. If TRUE, DESpace_test filters genes: genes have to be expressed in at least 'min_non_zero_spots' spots, and a gene requires at least 'min counts' counts per sample (across all locations).
verbose	A logical. If TRUE, DESpace_test returns two more results: 'DGEGLM' and 'DGELRT' objects contain full statistics from 'edgeR::glmFit' and 'edgeR::glmLRT'.

Details

If 'sample_col' is not specified and 'replicates == FALSE', DESpace_test assumed that data comes from an individual sample, and performs SV testing on it.

If 'sample_col' is provided and 'replicates == FALSE', DESpace_test tests each sample individually and returns a list of results for each sample.

If 'sample_col' is provided and 'replicates == TRUE', DESpace_test performs a joint multi-sample test.

Value

A list of results:

- "gene_results": a dataframe contains main edgeR test results;

- "estimated_y": a DGEList object contains the estimated common dispersion, which can later be used to speed-up calculation when testing individual clusters.

- "glmFit" (only if verbose = TRUE): a DGEGLM object contains full statistics from "edgeR::glmFit".

- "glmLRT" (only if verbose = TRUE): a DGELRT object contains full statistics from "edgeR::glmLRT".

See Also

top_results, individual_test, FeaturePlot

Examples

```
# load the input data:
data("LIBD_subset", package = "DESpace")
LIBD_subset

# Fit the model via \code{\link{DESpace_test}} function.
set.seed(123)
results_DESpace <- DESpace_test(spe = LIBD_subset,
                           spatial_cluster = "layer_guess_reordered",
                           verbose = FALSE)

# DESpace_test returns of a list of 2 objects:
# "gene_results": a dataframe contains main edgeR test results;
# "estimated_y": a DGEList object contains the estimated common dispersion,
#  which can later be used to speed-up calculation when testing individual clusters.

# We visualize differential results:
head(results_DESpace$gene_results, 3)
```

FeaturePlot *FeaturePlot*

Description

Plot spatial gene expression. This function is a modified version of the FeaturePlot function from BayesSpace R package. In comparison to the original BayesSpace function, this function allows plotting multiple genes simultaneously and drawing an outline around a specified cluster.

Usage

```
FeaturePlot(
  spe,
  feature,
  coordinates = NULL,
  assay.type = "logcounts",
  Annotated_cluster = FALSE,
  diverging = FALSE,
  low = NULL,
  high = NULL,
  mid = NULL,
  color = NULL,
  platform = "Visium",
  spatial_cluster = NULL,
  cluster = NULL,
  legend_cluster = FALSE,
  label = FALSE,
  ncol = 3,
  title = FALSE,
  linewidth = 0.4,
  legend_exprs = FALSE,
  title_size = 10
)
```

Arguments

spe	SpatialExperiment or SingleCellExperiment. If feature is specified and is a string, it must exist as a row in the specified assay of spe.
feature	Feature vector used to color each spot. May be the name of a gene/row in an assay of spe, or a vector of continuous values.
coordinates	Column names of spatial coordinates of spots stored in colData(spe).
assay.type	String indicating which assay in spe the expression vector should be taken from.
Annotated_cluster	A logical. TRUE or FALSE, indicating whether to plot the annotated spatial clusters next to expression plots.

diverging	A logical. If true, use a diverging color gradient in FeaturePlot (e.g. when plotting a fold change) instead of a sequential gradient (e.g. when plotting expression).
low, mid, high	Optional hex codes for low, mid, and high values of the color gradient used for continuous spot values.
color	Optional hex code to set color of borders around spots. Set to NA to remove borders.
platform	Spatial sequencing platform. If "Visium", the hex spot layout will be used, otherwise square spots will be plotted. NOTE: specifying this argument is only necessary if spe was not created by BayesSpace::patialCluster() or BayesSpace::spatialEnhance().
spatial_cluster	Column name of spatial clusters in colData(spe).
cluster	Names of the spatial clusters used for drawing a boundary around a group of points that belong to the specify cluster. It can be NULL, "all"/"ALL", or a vector of cluster names.
legend_cluster	A logical. TRUE of FALSE, indicating whether to plot the legend for the shaped clusters (TRUE), or not (FALSE). Only used when 'spatial_cluster' and 'cluster' are specified.
label	A logical. TRUE of FALSE. Adding a label and an arrow pointing to a group.
ncol	The dimensions of the grid to create. By default, 1, if the length of feature equals to 1, and 3, otherwise.
title	A logical. TRUE or FALSE. If true, the title name of each (subplot) is the gene name.
linewidth	The width of the boundary line around the cluster. The default ('0.4') size of the boundary line is one.
legend_exprs	A logical. TRUE of FALSE, indicating whether to plot the legend for the expression level (TRUE), or not (FALSE).
title_size	Text size.

Value

Returns a ggplot object.

See Also

DESpace_test, individual_test, top_results

Examples

```
# load the input data:
data("LIBD_subset", package = "DESpace")

# load pre-computed results (obtained via `DESpace_test`)
data("results_DESpace_test", package = "DESpace")
```

```
# Visualize the gene expression of the top three genes
feature = results_DESpace_test$gene_results$gene_id[seq_len(3)]
FeaturePlot(LIBD_subset, feature, coordinates = c("array_row", "array_col"),
            ncol = 3, title = TRUE)
```

`individual_test` *individual_test*

Description

DESpace can also be used to reveal the specific areas of the tissue affected by SVGs; i.e., spatial clusters that are particularly over/under abundant compared to the average signal. This function can be used to identify SVGs for each individual cluster.

Usage

```
individual_test(
  spe,
  spatial_cluster,
  sample_col = "sample_id",
  edgeR_y = NULL,
  min_counts = 20,
  min_non_zero_spots = 10,
  filter_gene = TRUE,
  replicates = FALSE,
  BPPARAM = NULL
)
```

Arguments

`spe`	SpatialExperiment or SingleCellExperiment.
`spatial_cluster`	Column name of spatial clusters in `colData(spe)`.
`sample_col`	Column name of sample ids in `colData(spe)`.
`edgeR_y`	Pre-estimated dispersion; if it's null, compute dispersion.
`min_counts`	Minimum number of counts per sample (across all spots) for a gene to be analyzed.
`min_non_zero_spots`	Minimum number of non-zero spots per sample, for a gene to be analyzed.
`filter_gene`	A logical. If TRUE, `DESpace_test` filters genes: genes have to be expressed in at least 'min_non_zero_spots' spots, and a gene requires at least 'min counts' counts per sample (across all locations).
`replicates`	Single sample or multi-sample test.

BPPARAM An optional parameter passed internally to bplapply. We suggest using as many cores as the number of spatial clusters. If unspecified, the script does not run in parallel. Note that parallel coding performs better only when dispersion estimations are not provided beforehand. Moreover, parallelizing the script will increase the memory requirement; if memory is an issue, leave 'BPPARAM' unspecified and, hence, avoid parallelization.

Details

For every spatial cluster we test, edgeR would normally re-compute the dispersion estimates based on the specific design of the test. However, this calculation represents the majority of the overall computing time. Therefore, to speed-up calculations, we propose to use the dispersion estimates which were previously computed for the gene-level tests. This introduces a minor approximation which, in our benchmarks, does not lead to decreased accuracy. If you want to use pre-computed gene-level dispersion estimates, set edgeR_y to 'estimated_y'. Alternatively, if you want to re-compute dispersion estimates (significantly slower, but marginally more accurate option), leave edgeR_y empty.

Value

A list of results, with one result per spatial cluster in each element. Specifically, each item in the list is a "gene_results" dataframe which contains main edgeR test results.

See Also

top_results, DESpace_test, FeaturePlot

Examples

```
# load the input data:
data("LIBD_subset", package = "DESpace")
LIBD_subset

# load pre-computed results (obtaines via `DESpace_test`)
data("results_DESpace_test", package = "DESpace")

# DESpace_test returns of a list of 2 objects:
# "gene_results": a dataframe contains main edgeR test results;
# "estimated_y": a DGEList object contains the estimated common dispersion,
#  which can later be used to speed-up calculation when testing individual clusters.

# We visualize differential results:
head(results_DESpace_test$gene_results, 3)

# Individual cluster test: identify SVGs for each individual cluster
# set parallel computing; we suggest using as many cores as the number of spatial clusters.
# Note that parallelizing the script will increase the memory requirement;
# if memory is an issue, leave 'BPPARAM' unspecified and, hence, avoid parallelization.
set.seed(123)
results_individual_test <- individual_test(LIBD_subset,
                                           edgeR_y = results_DESpace_test$estimated_y,
```

```
                                                spatial_cluster = "layer_guess_reordered")

# We visualize results for the cluster 'WM'
results_WM <- results_individual_test[[7]]
head(results_WM,3)
```

LIBD_subset *Subset from the human DLPFC 10x Genomics Visium dataset of the* spatialLIBD *package*

Description

Subset from the human DLPFC 10x Genomics Visium dataset of the spatialLIBD package

Arguments

LIBD_subset	contains a SpatialExperiment object, representing a subset of the sample 151673 from the full real data of the spatialLIBD package. Below the code used to subset the original dataset.

Author(s)

Peiying Cai <peiying.cai@uzh.ch>, Simone Tiberi <simone.tiberi@unibo.it>

See Also

DESpace_test, individual_test

Examples

```
# Connect to ExperimentHub
# ehub <- ExperimentHub::ExperimentHub()
# Download the example spe data
# spe_all <- spatialLIBD::fetch_data(type = "spe", eh = ehub)
# Select one sample only:
# LIBD_subset <- spe_all[, colData(spe_all)$sample_id == '151673']
# Select small set of random genes for faster runtime
# set.seed(123)
# sel_genes <- sample(dim(LIBD_subset)[1],500)
# LIBD_subset <- LIBD_subset[sel_genes,]
# keep_col <- c("array_row","array_col","layer_guess_reordered")
# library(SingleCellExperiment)
# LIBD_subset <- SpatialExperiment(assay = list(counts = assay(LIBD_subset),
#                                               logcounts = logcounts(LIBD_subset)),
#                                  colData = colData(LIBD_subset)[keep_col])
# save(LIBD_subset, file = "./DESpace/data/LIBD_subset.RData")
```

results_DESpace_test *Results from* DESpace_test *function*

Description

Results from DESpace_test function

Arguments

results_DESpace_test
: contains a list object, with the results obtained applying DESpace_test function to an external dataset from the spatialLIBD package. Below the code used to obtain 'results_DESpace_test'.

Author(s)

Peiying Cai <peiying.cai@uzh.ch>, Simone Tiberi <simone.tiberi@unibo.it>

See Also

DESpace_test

Examples

```
# load the input data:
# data("LIBD_subset", package = "DESpace")
# LIBD_subset
#
# Fit the model via `DESpace_test` function.
# Parameter `spe` specifies the input `SpatialExperiment` or `SingleCellExperiment` object,
# while `spatial_cluster` defines the column names of `colData(spe)` containing spatial clusters.
# To obtain all statistics, set `verbose` to `TRUE`.
# DESpace_test returns of a list of 4 objects:
# "gene_results": a dataframe contains main edgeR test results;
# "estimated_y": a DGEList object contains the estimated common dispersion,
#  which can later be used to speed-up calculation when testing individual clusters.
# "glmFit": a DGEGLM object contains full statistics from "edgeR::glmFit"
# "glmLRT" a DGELRT object contains full statistics from "edgeR::glmLRT"
#
# set.seed(123)
# results_DESpace_test <- DESpace_test(spe = LIBD_subset,
#                                      spatial_cluster = "layer_guess_reordered",
#                                      verbose = FALSE)
#
# save(results_DESpace_test, file = "./DESpace/data/results_DESpace_test.RData")
```

results_individual_test *Results from* individual_test *function*

Description

Results from individual_test function

Arguments

results_individual_test
: contains a list object, with the results obtained applying individual_test function to an external dataset from the spatialLIBD package. Below the code used to obtain 'results_individual_test'.

Author(s)

Peiying Cai <peiying.cai@uzh.ch>, Simone Tiberi <simone.tiberi@unibo.it>

See Also

DESpace_test, individual_test

Examples

```
# load the input data:
# data("LIBD_subset", package = "DESpace")
# LIBD_subset
# load pre-computed results (obtained via `DESpace_test`)
# data("results_DESpace_test", package = "DESpace")
# results_DESpace_test

# Function `individual_test()` can be used to identify SVGs for each individual cluster.
# Parameter `spatial_cluster` indicates the column names of `colData(spe)`
# containing spatial clusters.
# set.seed(123)
# results_individual_test <- individual_test(LIBD_subset,
#                                            edgeR_y = results_DESpace_test$estimated_y,
#                                            spatial_cluster = "layer_guess_reordered")
# save(results_individual_test, file = "./DESpace/data/results_individual_test.RData")
```

top_results *top_results*

Description

Filter significant results. top_results returns the significant results obtained via DESpace_test and individual_test. It can also be used to merge gene- and cluster-level results into a single object.

Usage

```
top_results(
  gene_results = NULL,
  cluster_results,
  cluster = NULL,
  select = "both",
  high_low = NULL
)
```

Arguments

gene_results	Results returned from DESpace_test.
cluster_results	Results returned from individual_test.
cluster	A character indicating the cluster(s) whose results have to be returned. Results from all clusters are returned by default ("NULL").
select	A character indicating what results should be returned ("FDR", "logFC", or "both"). Only used if "cluster_results" are provided. By default ("both"), both FDR and logFC are returned.
high_low	A character indicating whether to filter results or not. Only used if "cluster_results" are provided, and one cluster is specified in "cluster" parameter. By default (NULL), all results are returned in a single data.frame. If "high" or "HIGH", we only return SVGs with average abundace in "cluster" higher than in the rest of the tissue (i.e., logFC > 0). If "low" or "LOW", we only return SVGs with average abundace in "cluster" lower than in the rest of the tissue (i.e., logFC < 0). If "both" or "BOTH", then both "high" and "low" results are returned, but in two separate data.frames.

Value

A data.frame object or a list of data.frame with results.

A data.frame object or a list of data.frame with results.

- When only "cluster_results" is provided, results are reported as a data.frame with columns for gene names (gene_id), spatial clusters affected by SV (Cluster), cluster-specific likelihood ratio test statistics (LR), cluster-specific average (across spots) log-2 counts per million (logCPM),

cluster-specific log2-fold changes (logFC), cluster-specific raw p-values (PValue), and Benjamini-Hochberg adjusted p-values (FDR) for each spatial cluster.

- When "gene_results" and "cluster_results" are given, results are reported as a data.frame that merges gene- and cluster-level results.

- If "cluster" is specified, the function returns a subset data.frame for the given cluster, which contains cluster name, gene name, LR, logCPM, logFC, PValue and FDR, ordered by FDR for the specified cluster.

- If "high_low" is set, the function returns a list of data.frame that contains subsets of results for genes with higher and/or lower expression in the given cluster compared to the rest of the tissue.

See Also

DESpace_test, individual_test, FeaturePlot

Examples

```
# load pre-computed results (obtained via `DESpace_test`)
data("results_DESpace_test", package = "DESpace")

# DESpace_test returns of a list of 2 objects:
# "gene_results": a dataframe contains main edgeR test results;
# "estimated_y": a DGEList object contains the estimated common dispersion,
#  which can later be used to speed-up calculation when testing individual clusters.

# We visualize differential results:
head(results_DESpace_test$gene_results, 3)

# load pre-computed results (obtained via `individual_test`)
data("results_individual_test", package = "DESpace")
# Function `individual_test()` can be used to identify SVGs for each individual cluster.
# `individual_test()` returns a list containing the results of individual clusters.
# For each cluster, results are reported as a data.frame,
# where columns For each cluster, results are reported as a data.frame,
# where columns contain gene names (`genes`), likelihood ratio (`LR`),
# log2-fold changes (`logFC`) and adjusted p-value (`FDR`).
#
# Combine gene-and cluster-level results
merge_res = top_results(results_DESpace_test$gene_results,
                        results_individual_test)
head(merge_res,3)
# 'select = "FDR"' can be used to visualize adjusted p-values for each spatial cluster.
merge_res = top_results(results_DESpace_test$gene_results,
                        results_individual_test, select = "FDR")
head(merge_res,3)
# Specify the cluster of interest and check top genes detected by DESpace_test.
results_WM_both = top_results(cluster_results = results_individual_test,
                              cluster = "WM", high_low = "both")
head(results_WM_both$high_genes, 3)
head(results_WM_both$low_genes, 3)
```

Index