Package 'DeconvoBuddies'

April 5, 2026

Title Helper Functions for LIBD Deconvolution

Version 1.2.0

Date 2025-07-28

Description
Funtions helpful for LIBD deconvolution project. Includes tools for marker finding with mean ratio, expression plotting, and plotting deconvolution results. Working to include DLPFC datasets.

License Artistic-2.0

URL https://github.com/lahuuki/DeconvoBuddies

BugReports https://github.com/LieberInstitute/DeconvoBuddies/issues

biocViews Software, SingleCell, RNASeq, GeneExpression, Transcriptomics, ExperimentHubSoftware

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports AnnotationHub, BiocFileCache, DelayedMatrixStats, dplyr, ExperimentHub, ggplot2, graphics, grDevices, MatrixGenerics, methods, purrr, rafalib, reshape2, S4Vectors, scran, SingleCellExperiment, spatialLIBD, stats, stringr, SummarizedExperiment, tibble, utils

Suggests Biobase, BiocStyle, covr, HDF5Array, knitr, RColorBrewer, RefManageR, rmarkdown, sessioninfo, testthat (>= 3.0.0), tidyr, tidyverse

Config/testthat/edition 3

Depends R (>= 4.4.0)

VignetteBuilder knitr

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

git_branch RELEASE_3_22

git_last_commit 1c038d1

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2026-04-05

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

`DeconvoBuddies-package`

DeconvoBuddies: Helper Functions for LIBD Deconvolution

Description

Funtions helpful for LIBD deconvolution project. Includes tools for marker finding with mean ratio, expression plotting, and plotting deconvolution results. Working to include DLPFC datasets.

Author(s)

Maintainer: Louise Huuki-Myers <lahuuki@gmail.com> (ORCID)

Other contributors:

- Leonardo Collado-Torres <lcolladotor@gmail.com> (ORCID) [contributor]

See Also

Useful links:

- https://github.com/lahuuki/DeconvoBuddies
- Report bugs at https://github.com/LieberInstitute/DeconvoBuddies/issues

create_cell_colors *Create a cell color palette for plots*

Description

This function returns a `character()` vector with valid R colors for a given input `character()` of unique cell types. These were colors that have been useful in our experience.

Usage

```
create_cell_colors(
 cell_types = c("Astro", "Micro", "Endo", "Oligo", "OPC", "Excit", "Inhib", "Other"),
  palette_name = c("classic", "gg", "tableau"),
  palette = NULL,
  split = NA,
  preview = FALSE
)
```

Arguments

`cell_types`	A `character()` vector listing unique cell types.
`palette_name`	A `character(1)` indicating choice of included palettes: • `"classic"`: classic set of 8 cell type colors from LIBD, checked for visability and color blind accessibility. Default palette. • `"gg"` : mimic colors automatically picked by ggplot. • `"tableau"` : 20 distinct colors from tableau color palette, good for large number of cell type.
`palette`	A `character()` vector listing user provided color palette. If provided, overrides palette selection with palette_name.
`split`	delineating `character(1)` after which suffixes will be ignored. This is useful for cases when say `A.1` and `A.2` are both to be considered fine subtypes of broad cell type `A` (here `split = "\\."`). When used the function returns a nested list of borad and fine cell types.
`preview`	A `logical(1)` indicating whether to make a plot to preview the colors.

Value

A named `character()` vector of R and hex color values compatible with `ggplot2:scale_color_manual()`.

Examples

```
## create cell colors with included palettes
create_cell_colors(palette_name = "classic")
create_cell_colors(palette_name = "classic", preview = TRUE)
create_cell_colors(palette_name = "tableau", preview = TRUE)

## use custom colors
my_colors <- c("darkorchid4", "deeppink4", "aquamarine3", "darkolivegreen1")
create_cell_colors(
    cell_type = c("A", "B", "C", "D"),
```

```
    palette = my_colors,
    preview = TRUE
)

## use Rcolor brewer
create_cell_colors(
    cell_type = c("A", "B", "C"),
    palette = RColorBrewer::brewer.pal(n = 3, name = "Set1"),
    previe = TRUE
)

## Options for subtype handling
## Provide unique colors for cell subtypes (DEFAULT) - returns one level list
create_cell_colors(
    cell_types = c("A.1", "A.2", "B.1", "C", "D"),
    palette_name = "classic",
    preview = FALSE
)

## Provide gradient colors for A.1 and A.2 by using the "split" argument
## returns a nested list with broad & fine cell type colors, fine cell types
## are gradient with the top level matching the broad cell type
create_cell_colors(
    cell_types = c("A.1", "A.2", "B.1", "C", "D"),
    split = "\\.",
    palette_name = "classic",
    preview = TRUE
)

## try with custom colors
create_cell_colors(
    cell_types = c("A.1", "A.2", "B.1", "C", "D"),
    split = "\\.",
    palette = my_colors,
    preview = TRUE
)
```

`est_prop` *Bisque Estimated Cell Type Proportions for DLPFC bulk RNA-seq data*

Description

Cell type proportions estimated by Bisque for DLPFC bulk RNA-seq data set, utilizing DLPFC snRNA-seq data as the reference data.

Usage

```
data("est_prop")
```

Format

A `data.frame` object.

Details

16.79 kB

These are the columns of the `data.frame` object:

- Astro: estimated proportions of Astrocyte cells
- EndoMural: estimated proportions of Endothelia + Mural cells
- Micro: estimated proportions of Microglia cells
- Oligo: estimated proportions of Oligodendrocyte Cells
- OPC: estimated proportions of Oligodendrocyte Precursor Cells
- Excit: estimated proportions for Excitatory Neurons
- Inhib: estimated proportions for Inhibitory Neurons

Source

https://github.com/LieberInstitute/DeconvoBuddies/blob/master/inst/extdata/data-raw/est_prop.R

Examples

```
## R Note that the `rowSums(est_prop)` is equal to 1,
## with a small error tolerance.
data("est_prop")
summary(rowSums(est_prop) - 1)

## You can check this yourself with:
all(round(rowSums(est_prop), 3) == 1)

# To view source
system.file("extdata", "data-raw", "est_prop.R", package = "DeconvoBuddies")
```

`est_prop_test` *Test Estimated Cell Type Proportions*

Description

A test dataset of estimated proportions for 5 cell types over 100 samples.

Usage

```
data("est_prop_test")
```

Format

A `data.frame` object.

Details

11.62 kB

These are the columns of the `data.frame` object:

- cell_A: estimated proportions for cell type A
- cell_B: estimated proportions for cell type B
- cell_C: estimated proportions for cell type C
- cell_D: estimated proportions for cell type D
- cell_E: estimated proportions for cell type E

Source

https://github.com/LieberInstitute/DeconvoBuddies/blob/master/inst/extdata/data-raw/est_prop_test.R

Examples

```
## R Note that the `rowSums(est_prop_test)` is equal to 1,
## with a small error tolerance.
data("est_prop_test")
summary(rowSums(est_prop_test) - 1)

## You can check this yourself with:
all(round(rowSums(est_prop_test), 3) == 1)

# To view source
system.file("extdata", "data-raw", "est_prop_test.R", package = "DeconvoBuddies")
```

`fetch_deconvo_data` *Download Human DLPFC Deconvolution Data*

Description

This function downloads the processed data for the experiment documented at https://github.com/LieberInstitute/Human_DLPFC_Deconvolution. Internally, this function downloads the data from `ExperimentHub`.

Usage

```
fetch_deconvo_data(
  type = c("rse_gene", "sce", "sce_DLPFC_example"),
  destdir = tempdir(),
  eh = ExperimentHub::ExperimentHub(),
  bfc = BiocFileCache::BiocFileCache()
)
```

Arguments

type	A character(1) specifying which file you want to download. • rse_gene: A RangedSummarizedExperiment-class with 110 bulk RNA-seq samples x 21k genes. (41 MB) • sce: A SingleCellExperiment object with Human DLPFC snRNA-seq data. 77k nuclei x 36k genes (172 MB) • sce_DLPFC_example: An example subset of sec SingleCellExperiment with 10k nuclei x 557 genes (49 MB)
destdir	The destination directory to where files will be downloaded to in case the ExperimentHub resource is not available. If you already downloaded the files, you can set this to the current path where the files were previously downloaded to avoid re-downloading them.
eh	An ExperimentHub object ExperimentHub-class.
bfc	A BiocFileCache object BiocFileCache-class. Used when eh is not available.

Details

We are currently waiting for https://doi.org/10.1101/2024.02.09.579665 to pass peer review at a journal, which could lead to changes requested by the peer reviewers on the processed data for this study. Thus, this function temporarily downloads the files from Dropbox using BiocFileCache::bfcrpath() unless the files are present already at destdir.

Note that ExperimentHub and BiocFileCache will cache the data and automatically detect if you have previously downloaded it, thus making it the preferred way to interact with the data.

This function is based on spatialLIBD::fetch_data().

Value

The requested object: rse_gene that you assign to an object

Examples

```
## Download the bulk RNA gene expression data
## A RangedSummarizedExperiment (41.16 MB)

if (!exists("rse-gene")) rse_gene <- fetch_deconvo_data("rse_gene")

## explore bulk data
rse_gene

## load example snRNA-seq data
## A SingleCellExperiment (4.79 MB)
if (!exists("sce_DLPFC_example")) sce_DLPFC_example <- fetch_deconvo_data("sce_DLPFC_example")

## explore example sce data
sce_DLPFC_example

## check the logcounts
SingleCellExperiment::logcounts(sce_DLPFC_example)[1:5, 1:5]


## download the full sce experiment object
sce_path_zip <- fetch_deconvo_data("sce")
```

```
sce_path <- unzip(sce_path_zip, exdir = tempdir())
sce <- HDF5Array::loadHDF5SummarizedExperiment(
    file.path(tempdir(), "sce_DLPFC_annotated")
)
```

`findMarkers_1vAll` *Calculate 1 vs. All standard fold change for each gene x cell type, wrapper function for* `scran::findMarkers()`.

Description

For each cell type, this function computes the statistics comparing that cell type (the "1") against all other cell types combined ("All").

Usage

```
findMarkers_1vAll(
  sce,
  assay_name = "counts",
  cellType_col = "cellType",
  add_symbol = FALSE,
  mod = NULL,
  verbose = TRUE,
  direction = "up"
)
```

Arguments

`sce`	A SingleCellExperiment object.
`assay_name`	Name of the assay to use for calculation. See see `assayNames(sce)` for possible values.
`cellType_col`	Column name on `colData(sce)` that denotes the cell type.
`add_symbol`	A `logical(1)` indicating whether to add the gene symbol column to the marker stats table.
`mod`	A `character(1)` string specifying the model used as design in `scran::findMarkers()`. It can be `NULL` (default) if there are no blocking terms with uninteresting factors as documented at pairwiseTTests.
`verbose`	A `logical(1)` choosing whether to print progress messages or not.
`direction`	A `character(1)` for the choice of direction tested for gene cell type markers: `"up"` (default), `"any"`, or `"down"`. Impacts p-values: if `"up"` genes with logFC < 0 will have `p.value = 1`.

Details

See https://github.com/MarioniLab/scran/issues/57 for a more in depth discussion about the standard log fold change statistics provided by `scran::findMarkers()`.

See also https://youtu.be/IaclszgZb-g for a LIBD rstats club presentation on "Finding and interpreting marker genes in sc/snRNA-seq data". The companion notes are available at https://docs.google.com/document/d/1BeMtKgE7gpmNywInndVC9o_ufopn-U2EZHB32bO7ObM/edit?usp=sharing.

Value

A `tibble::tibble()` of 1 vs. ALL standard log fold change + p-values for each gene x cell type.

- `gene` is the name of the gene (from `rownames(sce)`).
- `logFC` the log fold change from the DE test
- `log.p.value` the log of the p-value of the DE test
- `log.FDR` the log of the False Discovery Rate adjusted p.value
- `std.logFC` the standard logFC.
- `cellType.target` the cell type we're finding marker genes for
- `std.logFC.rank` the rank of `std.logFC` for each cell type
- `std.logFC.anno` is an annotation of the `std.logFC` value helpful for plotting.

See Also

Other marker gene functions: `get_mean_ratio()`

Examples

```
## load example SingleCellExperiment
if (!exists("sce_DLPFC_example")) sce_DLPFC_example <- fetch_deconvo_data("sce_DLPFC_example")
## Explore properties of the sce object
sce_DLPFC_example

## this data contains logcounts of gene expression
SummarizedExperiment::assays(sce_DLPFC_example)$logcounts[1:5, 1:5]

## nuclei are classified in to cell types
table(sce_DLPFC_example$cellType_broad_hc)

## Get the 1vALL stats for each gene for each cell type defined in
## `cellType_broad_hc`
marker_stats_1vAll <- findMarkers_1vAll(
    sce = sce_DLPFC_example,
    assay_name = "logcounts",
    cellType_col = "cellType_broad_hc",
    mod = "~BrNum"
)

## explore output, top markers have high logFC
head(marker_stats_1vAll)
```

---

get_mean_ratio *Get Mean Ratio for Each Gene x Cell Type*

Description

Calculate the Mean Ratio value and rank for each gene for each cell type in the `sce` object, to identify effective marker genes for deconvolution.

Usage

```
get_mean_ratio(
  sce,
  cellType_col,
  assay_name = "logcounts",
  gene_ensembl = NULL,
  gene_name = NULL
)
```

Arguments

sce	SummarizedExperiment-class (or any derivative class) object containing single cell/nucleus gene expression data.
cellType_col	A character(1) name of the column in the colData() of sce that denotes the cell type or group of interest.
assay_name	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.
gene_ensembl	A character(1) specifying the rowData(sce_pseudo) column with the ENSEMBL gene IDs. This will be used by layer_stat_cor().
gene_name	A character(1) specifying the rowData(sce_pseudo) column with the gene names (symbols).

Details

Note if a cell type has < 10 cells the MeanRatio results may be unstable. See rational in OSCA: https://bioconductor.org/books/3.19/OSCA.multisample/multi-sample-comparisons.html#performing-the-de-analysis.

Value

A tibble::tibble() with the MeanRatio values for each gene x cell type.

- gene is the name of the gene (from rownames(sce)).
- cellType.target is the cell type we're finding marker genes for.
- mean.target is the mean expression of gene for cellType.target.
- cellType.2nd is the second highest non-target cell type.
- mean.2nd is the mean expression of gene for cellType.2nd.
- MeanRatio is the ratio of mean.target/mean.2nd.
- MeanRatio.rank is the rank of MeanRatio for the cell type.
- MeanRatio.anno is an annotation of the MeanRatio calculation helpful for plotting.
- gene_ensembl & gene_name optional columns from rowData(sce) specified by the user to add gene information.

See Also

Other marker gene functions: findMarkers_1vAll()

Examples

```
## load example SingleCellExperiment
if (!exists("sce_DLPFC_example")) sce_DLPFC_example <- fetch_deconvo_data("sce_DLPFC_example")
## Explore properties of the sce object
sce_DLPFC_example

## this data contains logcounts of gene expression
SummarizedExperiment::assays(sce_DLPFC_example)$logcounts[1:5, 1:5]

## nuclei are classified in to cell types
table(sce_DLPFC_example$cellType_broad_hc)

## Get the mean ratio for each gene for each cell type defined in
## `cellType_broad_hc`
get_mean_ratio(sce_DLPFC_example, cellType_col = "cellType_broad_hc")

# Option to specify gene_name as the "Symbol" column from rowData
# this will be added to the marker stats output
SummarizedExperiment::rowData(sce_DLPFC_example)

## specify rowData col names for gene_name and gene_ensembl
get_mean_ratio(sce_DLPFC_example,
    cellType_col = "cellType_broad_hc",
    gene_name = "gene_name",
    gene_ensembl = "gene_id"
)
```

`make_test_sce` *Simulate a SingleCellExperiment for testing.*

Description

The counts are simulated from a poisson distribution with `stats::rpois()`. Use `set.seed()` if you want the results to be reproducible.

Usage

```
make_test_sce(n_cell = 100, n_gene = 100, n_cellType = 4, n_donor = 2)
```

Arguments

`n_cell`	An `integer(1)` specifying the number of cells.
`n_gene`	An `integer(1)` specifying the number of genes.
`n_cellType`	An `integer(1)` specifying the number of cell types.
`n_donor`	An `integer(1)` specifying the number of donors.

Value

A SingleCellExperiment object with randomly generated counts and `colData()`.

Examples

```
## Create an example sce using default values.
set.seed(20240823)
test <- make_test_sce()

## Let's check the number of cells per cell type from each donor
addmargins(table(test$cellType, test$donor))
```

`marker_stats_1vAll` *1vAll Marker Statistics example data*

Description

A tibble contatinting the marker statistics calculated for 5k genes from DLPFC snRNA-seq dataset by `findMarkers_1vAll`.

Usage

```
data("marker_stats_1vAll")
```

Format

A `data.frame` object.

Details

3.47 MB

Source

https://github.com/LieberInstitute/DeconvoBuddies/blob/master/inst/extdata/data-raw/RNAScope_prop.R

Examples

```
# To view source
system.file("extdata", "data-raw", "marker_stats_1vAll.R", package = "DeconvoBuddies")
```

`marker_test` *Markers stats from sce_DLPFC_example*

Description

A tibble containing the marker stats from `get_mean_ratio()` for `sce_DLPFC_example`.

Usage

```
data("marker_test")
```

Format

A `tibble::tibble()`. See `get_mean_ratio()` for more details on the column names.

Details

402.60 kB

Source

https://github.com/LieberInstitute/DeconvoBuddies/blob/master/inst/extdata/data-raw/marker_test.R

Examples

```
# To view source
system.file("extdata", "data-raw", "marker_test.R", package = "DeconvoBuddies")
```

---

plot_composition_bar *Create barplot of average cell type composition*

Description

Given a long formatted `data.frame`, this function creates a barplot for the average cell type composition among a set of samples (donors) using `ggplot2`.

Usage

```
plot_composition_bar(
  prop_long,
  sample_col = "RNum",
  x_col = "ALL",
  prop_col = "prop",
  ct_col = "cell_type",
  add_text = TRUE,
  min_prop_text = 0
)
```

Arguments

`prop_long`	A `data.frame` of cell type portions in long form
`sample_col`	A `character(1)` specifying the name of column in `prop_long` that identifies samples.
`x_col`	A `character(1)` specifying the name of column in `prop_long` that specifies the category to divide samples by.
`prop_col`	A `character(1)` specifying the name of column in `prop_long` that contains proportion values.
`ct_col`	A `character(1)` specifying the name of column in `prop_long` containing cell type names.
`add_text`	A `logical(1)` determining whether to add the rounded proportion value to the bars.
`min_prop_text`	A `numeric(1)` specifying the minimum proportion to display text. Values greater than (>) `min_prop_text` will be displayed.

Value

A stacked barplot ggplot2 object representing the mean proportion of cell types for each group.

Examples

```
# Load example data
data("rse_bulk_test")
data("est_prop_test")

# extract relevant colData from the example RangedSummarizedExperiment object
pd <- SummarizedExperiment::colData(rse_bulk_test) |>
    as.data.frame()

# combine with the example estimated proportions in a long style table
est_prop_test_long <- est_prop_test |>
    tibble::rownames_to_column("RNum") |>
    tidyr::pivot_longer(!RNum, names_to = "cell_type", values_to = "prop") |>
    dplyr::inner_join(pd |> dplyr::select(RNum, Dx))

est_prop_test_long

# Create composition bar plots
# Mean composition of all samples
plot_composition_bar(est_prop_test_long)

# Mean composition by Dx
plot_composition_bar(est_prop_test_long, x_col = "Dx")

# control minimum value of text to add
plot_composition_bar(est_prop_test_long, x_col = "Dx", min_prop_text = 0.1)

# plot all samples, then facet by Dx
plot_composition_bar(est_prop_test_long, x_col = "RNum", add_text = FALSE) +
    ggplot2::facet_wrap(~Dx, scales = "free_x")
```

---

plot_gene_express *Plot the gene expression of a list of genes in a SCE object*

Description

This function plots the expression of one or more genes as a violin plot, over a user defined category, typically a cell type annotation. The plots are made using ggplot2.

Usage

```
plot_gene_express(
  sce,
  genes,
  assay_name = "logcounts",
  category = "cellType",
  color_pal = NULL,
  title = NULL,
```

```
  plot_points = FALSE,
  ncol = 2,
  plot_type = c("violin", "boxplot"),
  free_y = FALSE
)
```

Arguments

sce	A SummarizedExperiment-class object or one inheriting it.
genes	A `character()` vector specifying the genes to plot, this should match the format of `rownames(sce)`.
assay_name	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.
category	A `character(1)` specifying the name of the categorical variable to group the cells or nuclei by. Defaults to `cellType`.
color_pal	A named `character(1)` vector that contains a color palette matching the `category` values.
title	A `character(1)` to title the plot.
plot_points	A `logical(1)` indicating whether to plot points over the violin, defaults to `FALSE` as these often become over plotted and quite large (especially when saved as PDF).
ncol	An `integer(1)` specifying the number of columns for the facet in the final plot. Defaults to 2.
plot_type	A `character(1)` specifying whether to plot a 'violin' (default) or 'boxplot'.
free_y	`logical(1)` indicating whether to use "free" y-axis between genes (relevant to `facet_wrap`).

Value

A `ggplot()` violin plot for selected genes.

See Also

Other expression plotting functions: `plot_marker_express()`, `plot_marker_express_ALL()`, `plot_marker_express_List()`

Examples

```
## Using Symbol as rownames makes this more human readable
data("sce_ab")
plot_gene_express(sce = sce_ab, genes = c("G-D1_A"))
plot_gene_express(sce = sce_ab, genes = c("G-D1_A"), plot_type = "boxplot")

# Access example data
if (!exists("sce_DLPFC_example")) sce_DLPFC_example <- fetch_deconvo_data("sce_DLPFC_example")

## plot expression of two genes
plot_gene_express(
    sce = sce_DLPFC_example,
    category = "cellType_broad_hc",
    genes = c("GAD2", "CD22")
```

```
)

## plot as boxplot
plot_gene_express(
    sce = sce_DLPFC_example,
    category = "cellType_broad_hc",
    genes = c("GAD2", "CD22"),
    plot_type = "boxplot"
)

## plot points - note this creates large images and is easy to over plot
plot_gene_express(
    sce = sce_DLPFC_example,
    category = "cellType_broad_hc",
    genes = c("GAD2", "CD22"),
    plot_points = TRUE
)

## with boxplot
plot_gene_express(
    sce = sce_DLPFC_example,
    category = "cellType_broad_hc",
    genes = c("GAD2", "CD22"),
    plot_points = TRUE,
    plot_type = "boxplot"
)

## Use free y-axis between genes
plot_gene_express(
    sce = sce_DLPFC_example,
    category = "cellType_broad_hc",
    genes = c("GAD2", "CD22"),
    plot_points = TRUE,
    plot_type = "boxplot",
    free_y = FALSE
)

## Add title
plot_gene_express(
    sce = sce_DLPFC_example,
    category = "cellType_broad_hc",
    genes = c("GAD2", "CD22"),
    title = "My Genes"
)

## Add color pallet
my_cell_colors <- create_cell_colors(cell_types = levels(sce_DLPFC_example$cellType_broad_hc))

plot_gene_express(
    sce = sce_DLPFC_example,
    category = "cellType_broad_hc",
    genes = c("GAD2", "CD22"),
    color_pal =  my_cell_colors,
    plot_type = "boxplot",
    plot_points = TRUE
)
```

plot_marker_express *Plot gene expression violin plots for top marker genes for one cell type*

Description

This function plots the top n marker genes for a specified cell type based off of the `stats` table from `get_mean_ratio()`. The gene expression is plotted as violin plot with `plot_gene_express` and adds annotations to each plot.

Usage

```
plot_marker_express(
  sce,
  stats,
  cell_type,
  n_genes = 4,
  rank_col = "MeanRatio.rank",
  anno_col = "MeanRatio.anno",
  gene_col = "gene",
  cellType_col = "cellType",
  color_pal = NULL,
  plot_points = FALSE,
  ncol = 2
)
```

Arguments

`sce`	SummarizedExperiment-class object
`stats`	A `data.frame()` generated by `get_mean_ratio()` and/or `findMarkers_1vAll()`.
`cell_type`	A `character()` target cell type to plot markers for
`n_genes`	An `integer(1)` of number of markers you'd like to plot
`rank_col`	The `character(1)` name of column to rank genes by in `stats`.
`anno_col`	The `character(1)` name of column containing annotation in `stats`.
`gene_col`	The `character(1)` name of column containing gene name in `stats` should be the same syntax as `rownames(sce)`.
`cellType_col`	The `character(1)` name of `colData()` column containing cell type for `sce` data. It should match `cellType.target` in `stats`.
`color_pal`	A named `character(1)` vector that contains a color palette matching the `cell_type` values.
`plot_points`	A `logical(1)` indicating whether to plot points over the violin, defaults to `FALSE` as these often become over plotted and quite large (especially when saved as PDF).
`ncol`	An `integer(1)` specifying the number of columns for the facet in the final plot. Defaults to 2.

Value

A `ggplot2` object created with `plot_gene_express()`. It is a `scater::plotExpression()` style violin plot for selected marker genes.

See Also

Other expression plotting functions: plot_gene_express(), plot_marker_express_ALL(), plot_marker_express_L

Examples

```
## Download the processed study data from
## <https://github.com/LieberInstitute/Human_DLPFC_Deconvolution>.
if (!exists("sce_DLPFC_example")) sce_DLPFC_example <- fetch_deconvo_data("sce_DLPFC_example")

## load example marker stats
data("marker_test")

## Plot the top markers for Astrocytes
plot_marker_express(
    sce = sce_DLPFC_example,
    stat = marker_test,
    cellType_col = "cellType_broad_hc",
    cell_type = "Astro",
    gene_col = "gene"
)
```

---

plot_marker_express_ALL

Plot the top marker genes for ALL cell types

Description

This function plots the top n marker genes for a all cell types based off of the stats table from get_mean_ratio() in a multi-page PDF file. The gene expression is plotted as violin plot with plot_gene_express() and adds annotations to each plot.

Usage

```
plot_marker_express_ALL(
  sce,
  stats,
  pdf_fn = NULL,
  n_genes = 10,
  rank_col = "MeanRatio.rank",
  anno_col = "MeanRatio.anno",
  gene_col = "gene",
  cellType_col = "cellType",
  color_pal = NULL,
  plot_points = FALSE
)
```

Arguments

sce	SummarizedExperiment-class object
stats	A data.frame() generated by get_mean_ratio() and/or findMarkers_1vAll().
pdf_fn	A character() of the pdf filename to plot to, if NULL returns all plots

n_genes	An integer(1) of number of markers you'd like to plot
rank_col	The character(1) name of column to rank genes by in stats.
anno_col	The character(1) name of column containing annotation in stats.
gene_col	The character(1) name of column containing gene name in stats should be the same syntax as rownames(sce).
cellType_col	The character(1) name of colData() column containing cell type for sce data. It should match cellType.target in stats.
color_pal	A named character(1) vector that contains a color palette matching the cell_type values.
plot_points	A logical(1) indicating whether to plot points over the violin, defaults to FALSE as these often become over plotted and quite large (especially when saved as PDF).

Value

A PDF file with violin plots for the expression of top marker genes for all cell types.

See Also

Other expression plotting functions: plot_gene_express(), plot_marker_express(), plot_marker_express_List(

Examples

```
#' ## Fetch sce example data
if (!exists("sce_DLPFC_example")) sce_DLPFC_example <- fetch_deconvo_data("sce_DLPFC_example")

## load example marker stats
data("marker_test")

# Plot marker gene expression to PDF, one page per cell type in stats
pdf_file <- tempfile("test_marker_expression_ALL", fileext = ".pdf")

plot_marker_express_ALL(
    sce_DLPFC_example,
    cellType_col = "cellType_broad_hc",
    stat = marker_test,
    pdf_fn = pdf_file
)

if (interactive()) browseURL(pdf_file)
```

---

plot_marker_express_List

Plot a nested list of genes as a multi-page pdf

Description

This function plots a nested list of genes as a multi-page PDF, one for each sub list. A use case is plotting known marker genes for multiple cell types over cell type clusters with unknown identities.

Usage

```
plot_marker_express_List(
  sce,
  gene_list,
  pdf_fn = NULL,
  cellType_col = "cellType",
  gene_name_col = "gene_name",
  color_pal = NULL,
  plot_points = FALSE
)
```

Arguments

sce	SummarizedExperiment-class object
gene_list	A named list() of character() vectors containing the names of genes to plot.
pdf_fn	A character() of the pdf filename to plot to, if NULL returns all plots
cellType_col	The character(1) name of colData() column containing cell type for sce data. It should match cellType.target in stats.
gene_name_col	The character(1) name of rowData() matching the gene name from gene_list.
color_pal	A named character(1) vector that contains a color palette matching the cell_type values.
plot_points	A logical(1) indicating whether to plot points over the violin, defaults to FALSE as these often become over plotted and quite large (especially when saved as PDF).

Value

A PDF file with violin plots for the expression of top marker genes for all cell types.

See Also

Other expression plotting functions: plot_gene_express(), plot_marker_express(), plot_marker_express_ALL()

Examples

```
## Fetch sce example data
if (!exists("sce_DLPFC_example")) sce_DLPFC_example <- fetch_deconvo_data("sce_DLPFC_example")

## Create list-of-lists of genes to plot, names of sub-list become title of page
my_gene_list <- list(Inhib = c("GAD2", "SAMD5"), Astro = c("RGS20", "PRDM16"))

# Return a list of plots
plots <- plot_marker_express_List(
    sce_DLPFC_example,
    gene_list = my_gene_list,
    cellType_col = "cellType_broad_hc"
)

print(plots[[1]])

# Plot marker gene expression to PDF, one page per cell type in stats
pdf_file <- tempfile("test_marker_expression_List", fileext = ".pdf")
```

```
plot_marker_express_List(
    sce_DLPFC_example,
    gene_list = my_gene_list,
    pdf_fn = pdf_file,
    cellType_col = "cellType_broad_hc"
)

if (interactive()) browseURL(pdf_file)
```

RNAScope_prop *Cell Type Proportions estimated from RNAScope*

Description

Cell type proportion estimates from high quality images from Huuki-Myers et al., bioRxiv, 2024, doi: https://doi.org/10.1101/2024.02.09.579665.

Usage

```
data("RNAScope_prop")
```

Format

A `data.frame` object.

Details

11.49 kB

These are the columns of the `data.frame` object:

- `SAMPLE_ID` : DLPFC Tissue block + RNAScope combination.
- `Sample` : DLFPC Tissue block (Donor BrNum + DLPFC position).
- `Combo` : RNAScope probe combination, either "Circle" marking cell types Astro Endo, Inhib, or "Star" marking Excit, Micro, and OligoOPC.
- `cell_type` : The cell type measured.
- `Confidence` : Image confidence, this dataset has been filtered to the high & Ok confidence images.
- `n_cell` : the number of cells counted for the Sample and cell type.
- `prop` : the calculated cell type proportion from n_cell.
- `n_cell_sn` : number of nuclei in the corresponding snRNA-seq data.
- `prop_sn` : cell type proportion from the snRNA-seq data.

NOTE: For the RNAScope assay utilized here only 3 cell types could be measured at once. Two consecutive tissue slices were used from each brain block to measure one combination of three major cell types, then the other three (differentiated as circle and star). DAPI was used to mark the nuclei, in each tissue section, so the overall number of cells is recorded but only a fraction has a cell type label, unlabeled nuclei are classified "other".

The two sections combined should get close to identifying the type of all the cells, but often the combined "non-other" fractions are around 0.7 and not 1. This could be due to a few reasons: the sections while as similar as possible are not the same tissue, error in the assay and/or image processing not labeling all cells possible, or presence of rare cell types not marked by the selected probes.

With all of this considered, we still think the RNAScope estimates are good approximations of the cell type proportions in these samples.

For more info check out https://doi.org/10.1101/2024.02.09.579665 Figure 2, and Methods: RNAScope/Immunofluorescence Data Generation and HALO Analysis.

Source

https://github.com/LieberInstitute/DeconvoBuddies/blob/master/inst/extdata/data-raw/RNAScope_prop.R

Examples

```
# To view source
system.file("extdata", "data-raw", "RNAScope_prop.R", package = "DeconvoBuddies")
```

---

rse_bulk_test *Test bulk rse dataset*

Description

A test rse_gene object with data for 1000 genes across 100 samples.

Usage

```
data("rse_bulk_test")
```

Format

A SummarizedExperiment object.

Details

976.77 kB

Source

https://github.com/LieberInstitute/DeconvoBuddies/blob/master/inst/extdata/data-raw/rse_bulk_test.R

Examples

```
# To view source
system.file("extdata", "data-raw", "rse_bulk_test.R", package = "DeconvoBuddies")
```

sce_ab *Toy SCE object for testing*

Description

An example sce object for testing, with two cell types A and B.

Usage

```
data("sce_ab")
```

Format

A SingleCellExperiment object.

Details

Generated with `DeconvoBuddies::make_test_sce()` 38.26 kB

Source

https://github.com/LieberInstitute/DeconvoBuddies/blob/master/inst/extdata/data-raw/sce_ab.R

Examples

```
# To view source
system.file("extdata", "data-raw", "sce_ab.R", package = "DeconvoBuddies")
```

Index