

Package ‘spatialLIBD’

March 29, 2021

Title LIBD Visium spatial transcriptomics human pilot data inspector

Version 1.2.1

Date 2020-12-17

Description Inspect interactively the spatial transcriptomics 10x Genomics Visium data from Maynard, Collado-Torres et al, 2020 analyzed by Lieber Institute for Brain Development researchers and collaborators.

License Artistic-2.0

Encoding UTF-8

LazyData true

Imports shiny, golem, ggplot2, cowplot, plotly, viridisLite, shinyWidgets, Polychrome, sessioninfo, grid, grDevices, methods, AnnotationHub, utils, png, scater, DT, ExperimentHub, RColorBrewer, SummarizedExperiment, stats, graphics, S4Vectors, IRanges, fields, benchmarkme, SpatialExperiment, BiocFileCache, jsonlite, tibble, readbitmap

RoxygenNote 7.1.1

Roxygen list(markdown = TRUE)

URL <https://github.com/LieberInstitute/spatialLIBD>

BugReports <https://support.bioconductor.org/t/spatialLIBD>

Suggests knitr, RefManageR, rmarkdown, BiocStyle, testthat (>= 2.1.0), covr, here, BiocManager

VignetteBuilder knitr

biocViews Homo_sapiens_Data, ExperimentHub, SequencingData, SingleCellData, ExpressionData, Tissue, PackageTypeData

Depends SingleCellExperiment, R (>= 3.6)

git_url <https://git.bioconductor.org/packages/spatialLIBD>

git_branch RELEASE_3_12

git_last_commit a78c6d3

git_last_commit_date 2020-12-17

Date/Publication 2021-03-29

Author Leonardo Collado-Torres [aut, cre]
 (<<https://orcid.org/0000-0003-2140-308X>>),
 Kristen R. Maynard [ctb] (<<https://orcid.org/0000-0003-0031-8468>>),
 Andrew E. Jaffe [ctb] (<<https://orcid.org/0000-0001-6886-1454>>),
 Brenda Pardo [ctb] (<<https://orcid.org/0000-0001-8103-7136>>)

Maintainer Leonardo Collado-Torres <lcolladotor@gmail.com>

R topics documented:

check_image_path	2
check_modeling_results	3
check_sce	4
check_sce_layer	5
enough_ram	6
fetch_data	7
gene_set_enrichment	8
gene_set_enrichment_plot	10
geom_spatial	12
get_colors	13
layer_boxplot	14
layer_matrix_plot	16
layer_stat_cor	18
layer_stat_cor_plot	19
libd_layer_colors	21
read_image	21
run_app	22
sce_image_clus	23
sce_image_clus_p	25
sce_image_gene	26
sce_image_gene_p	28
sce_image_grid	29
sce_image_grid_gene	31
sce_to_ve	33
sig_genes_extract	34
sig_genes_extract_all	35
sort_clusters	36
tstats_Human_DLPFC_snRNAseq_Nguyen_topLayer	37
update_scaleFactors	38
ve_image_colData	38

Index **40**

check_image_path	<i>Check input image_path</i>
------------------	-------------------------------

Description

This function checks that the `image_path` vector has the appropriate structure. For more details please check the vignette documentation.

Usage

```
check_image_path(image_path, sce)
```

Arguments

`image_path` A path to the directory containing the low resolution histology images that is needed for the interactive visualizations with `plotly`. See <https://github.com/LieberInstitute/spatialLIBD> for an example of how these files should be organized.

`sce` Defaults to the output of `fetch_data(type = 'sce')`. This is a [SingleCellExperiment](#) object with the spot-level Visium data and information required for visualizing the histology. See `fetch_data()` for more details.

Value

The input object if all checks are passed.

See Also

Other Check input functions: `check_modeling_results()`, `check_sce_layer()`, `check_sce()`

Examples

```
if (enough_ram()) {  
  ## Obtain the necessary data  
  if (!exists("sce")) sce <- fetch_data("sce")  
  
  ## Get the path to the images  
  img_path <- system.file("app", "www", "data", package = "spatialLIBD")  
  
  ## Check the object  
  check_image_path(img_path, sce)  
}
```

check_modeling_results

Check input modeling_results

Description

This function checks that the `modeling_results` object has the appropriate structure. For more details please check the vignette documentation.

Usage

```
check_modeling_results(modeling_results)
```

Arguments

modeling_results

Defaults to the output of `fetch_data(type = 'modeling_results')`. This is a list of tables with the columns `f_stat_*` or `t_stat_*` as well as `p_value_*` and `fdr_*` plus `ensembl`. The column name is used to extract the statistic results, the p-values, and the FDR adjusted p-values. Then the `ensembl` column is used for matching in some cases. See `fetch_data()` for more details.

Value

The input object if all checks are passed.

See Also

Other Check input functions: `check_image_path()`, `check_sce_layer()`, `check_sce()`

Examples

```
if (!exists("modeling_results")) {
  modeling_results <- fetch_data(type = "modeling_results")
}

## Check the object
xx <- check_modeling_results(modeling_results)
```

check_sce

Check input sce

Description

This function checks that the sce object has the appropriate structure. For more details please check the vignette documentation.

Usage

```
check_sce(
  sce,
  variables = c("GraphBased", "Layer", "Maynard", "Martinowich", paste0("SNN_k50_k",
    4:28), "layer_guess_reordered_short", "cell_count", "sum_umi", "sum_gene",
    "expr_chrM", "expr_chrM_ratio", "SpatialDE_PCA", "SpatialDE_pool_PCA", "HVG_PCA",
    "pseudobulk_PCA", "markers_PCA", "SpatialDE_UMAP", "SpatialDE_pool_UMAP", "HVG_UMAP",
    "pseudobulk_UMAP", "markers_UMAP", "SpatialDE_PCA_spatial",
    "SpatialDE_pool_PCA_spatial", "HVG_PCA_spatial", "pseudobulk_PCA_spatial",
    "markers_PCA_spatial", "SpatialDE_UMAP_spatial", "SpatialDE_pool_UMAP_spatial",
    "HVG_UMAP_spatial", "pseudobulk_UMAP_spatial", "markers_UMAP_spatial")
)
```

Arguments

sce	Defaults to the output of <code>fetch_data(type = 'sce')</code> . This is a SingleCellExperiment object with the spot-level Visium data and information required for visualizing the histology. See <code>fetch_data()</code> for more details.
variables	A <code>character()</code> vector of variable names expected to be present in <code>colData(sce)</code> .

Value

The input object if all checks are passed.

See Also

Other Check input functions: [check_image_path\(\)](#), [check_modeling_results\(\)](#), [check_sce_layer\(\)](#)

Examples

```
if (enough_ram()) {
  ## Obtain the necessary data
  if (!exists("sce")) sce <- fetch_data("sce")

  ## Check the object
  check_sce(sce)
}
```

check_sce_layer	<i>Check input sce_layer</i>
-----------------	------------------------------

Description

This function checks that the `sce_layer` object has the appropriate structure. For more details please check the vignette documentation.

Usage

```
check_sce_layer(sce_layer, variables = "layer_guess_reordered_short")
```

Arguments

sce_layer	Defaults to the output of <code>fetch_data(type = 'sce_layer')</code> . This is a SingleCellExperiment object with the spot-level Visium data compressed via pseudo-bulking to the layer-level (group-level) resolution. See <code>fetch_data()</code> for more details.
variables	A <code>character()</code> vector of variable names expected to be present in <code>colData(sce_layer)</code> .

Value

The input object if all checks are passed.

See Also

Other Check input functions: [check_image_path\(\)](#), [check_modeling_results\(\)](#), [check_sce\(\)](#)

Examples

```
## Obtain the necessary data
if (!exists("sce_layer")) sce_layer <- fetch_data("sce_layer")

## Check the object
check_sce_layer(sce_layer)
```

enough_ram	<i>Determine if you have enough RAM memory</i>
------------	--

Description

This function determines if you have enough RAM memory on your system.

Usage

```
enough_ram(how_much = 3e+09)
```

Arguments

how_much The number of bytes you want to compare against.

Details

If `benchmarkme::get_ram()` fails, this function will return FALSE as a save bet.

Value

A `logical(1)` indicating whether your system has enough RAM memory.

Examples

```
## Do you have ~ 3 GB in your system?
enough_ram(3e9)

## Do you have ~ 100 GB in your system
enough_ram(100e9)
```

 fetch_data

 Download the Human DLPFC Visium data from LIBD

Description

This function downloads from ExperimentHub the dorsolateral prefrontal cortex (DLPFC) human Visium data and results analyzed by LIBD. If ExperimentHub is not available, it will download the files from Dropbox using `utils::download.file()` unless the files are present already at `destdir`. Note that ExperimentHub will cache the data and automatically detect if you have previously downloaded it, thus making it the preferred way to interact with the data.

Usage

```
fetch_data(
  type = c("sce", "sce_layer", "modeling_results", "sce_example", "ve"),
  destdir = tempdir(),
  eh = ExperimentHub::ExperimentHub(),
  bfc = BiocFileCache::BiocFileCache()
)
```

Arguments

type	A character(1) specifying which file you want to download. It can either be: <code>sce</code> for the SingleCellExperiment object containing the spot-level data that includes the information for visualizing the clusters/genes on top of the Visium histology, <code>sce_layer</code> for the SingleCellExperiment object containing the layer-level data (pseudo-bulked from the spot-level), or <code>modeling_results</code> for the list of tables with the enrichment, pairwise, and anova model results from the layer-level data. It can also be <code>sce_example</code> which is a reduced version of <code>sce</code> just for example purposes. As of BioC version 3.12 <code>ve</code> downloads a VisiumExperiment-class object.
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 <code>eh</code> is not available.

Details

The data was initially prepared by scripts at <https://github.com/LieberInstitute/HumanPilot> and further refined by https://github.com/LieberInstitute/spatialLIBD/blob/master/inst/scripts/make-data_spatialLIBD.R.

Value

The requested object: `sce`, `sce_layer`, `ve` or `modeling_results` that you have to assign to an object. If you didn't you can still avoid re-loading the object by using `.Last.value`.

Examples

```
## Download the SingleCellExperiment object
## at the layer-level
if (!exists("sce_layer")) sce_layer <- fetch_data("sce_layer")

## Explore the data
sce_layer
```

gene_set_enrichment *Evaluate the enrichment for a list of gene sets*

Description

Using the layer-level (group-level) data, this function evaluates whether list of gene sets (Ensembl gene IDs) are enrichment among the significant genes (FDR < 0.1 by default) genes for a given model type result.

Usage

```
gene_set_enrichment(
  gene_list,
  fdr_cut = 0.1,
  modeling_results = fetch_data(type = "modeling_results"),
  model_type = names(modeling_results)[1],
  reverse = FALSE
)
```

Arguments

- | | |
|------------------|---|
| gene_list | A named list object (could be a data.frame) where each element of the list is a character vector of Ensembl gene IDs. |
| fdr_cut | A numeric(1) specifying the FDR cutoff to use for determining significance among the |
| modeling_results | Defaults to the output of <code>fetch_data(type = 'modeling_results')</code> . This is a list of tables with the columns <code>f_stat_*</code> or <code>t_stat_*</code> as well as <code>p_value_*</code> and <code>fdr_*</code> plus <code>ensembl</code> . The column name is used to extract the statistic results, the p-values, and the FDR adjusted p-values. Then the <code>ensembl</code> column is used for matching in some cases. See fetch_data() for more details. |
| model_type | A named element of the <code>modeling_results</code> list. By default that is either <code>enrichment</code> for the model that tests one human brain layer against the rest (one group vs the rest), <code>pairwise</code> which compares two layers (groups) denoted by <code>layerA-layerB</code> such that <code>layerA</code> is greater than <code>layerB</code> , and <code>anova</code> which determines if any layer (group) is different from the rest adjusting for the mean expression level. The statistics for <code>enrichment</code> and <code>pairwise</code> are t-statistics while the <code>anova</code> model ones are F-statistics. |
| reverse | A logical(1) indicating whether to multiply by -1 the input statistics and reverse the <code>layerA-layerB</code> column names (using the <code>-</code>) into <code>layerB-layerA</code> . |

Details

Check https://github.com/LieberInstitute/HumanPilot/blob/master/Analysis/Layer_Guesses/check_clinical_gene_sets.R to see a full script from where this family of functions is derived from.

Value

A table in long format with the enrichment results using `stats::fisher.test()`.

Author(s)

Andrew E Jaffe, Leonardo Collado-Torres

See Also

Other Gene set enrichment functions: `gene_set_enrichment_plot()`

Examples

```
## Read in the SFARI gene sets included in the package
asd_sfari <- utils::read.csv(
  system.file(
    "extdata",
    "SFARI-Gene_genes_01-03-2020release_02-04-2020export.csv",
    package = "spatialLIBD"
  ),
  as.is = TRUE
)

## Format them appropriately
asd_sfari_geneList <- list(
  Gene_SFARI_all = asd_sfari$ensembl.id,
  Gene_SFARI_high = asd_sfari$ensembl.id[asd_sfari$gene.score < 3],
  Gene_SFARI_syndromic = asd_sfari$ensembl.id[asd_sfari$syndromic == 1]
)

## Obtain the necessary data
if (!exists("modeling_results")) {
  modeling_results <- fetch_data(type = "modeling_results")
}

## Compute the gene set enrichment results
asd_sfari_enrichment <- gene_set_enrichment(
  gene_list = asd_sfari_geneList,
  modeling_results = modeling_results,
  model_type = "enrichment"
)

## Explore the results
asd_sfari_enrichment
```

```
gene_set_enrichment_plot
```

Plot the gene set enrichment results

Description

This function takes the output of `gene_set_enrichment()` and creates a heatmap visualization of the results.

Usage

```
gene_set_enrichment_plot(
  enrichment,
  xlabs = unique(enrichment$ID),
  PThresh = 12,
  ORcut = 3,
  enrichOnly = FALSE,
  layerHeights = c(0, seq_len(length(unique(enrichment$test)))) * 15,
  mypal = c("white", (grDevices::colorRampPalette(RColorBrewer::brewer.pal(9,
    "YlOrRd")))(50)),
  cex = 1.2
)
```

Arguments

<code>enrichment</code>	The output of <code>gene_set_enrichment()</code> .
<code>xlabs</code>	A vector of names in the same order and length as <code>unique(enrichment\$ID)</code> . Gets passed to <code>layer_matrix_plot()</code> .
<code>PThresh</code>	A <code>numeric(1)</code> specifying the P-value threshold for the maximum value in the $-\log_{10}(p)$ scale.
<code>ORcut</code>	A <code>numeric(1)</code> specifying the P-value threshold for the minimum value in the $-\log_{10}(p)$ scale for printing the odds ratio values in the cells of the resulting plot.
<code>enrichOnly</code>	A <code>logical(1)</code> indicating whether to show only odds ratio values greater than 1.
<code>layerHeights</code>	A <code>numeric()</code> vector of length equal to <code>length(unique(enrichment\$test)) + 1</code> that starts at 0 specifying where to plot the y-axis breaks which can be used for re-creating the length of each brain layer. Gets passed to <code>layer_matrix_plot()</code> .
<code>mypal</code>	A vector with the color palette to use. Gets passed to <code>layer_matrix_plot()</code> .
<code>cex</code>	Passed to <code>layer_matrix_plot()</code> .

Details

Check https://github.com/LieberInstitute/HumanPilot/blob/master/Analysis/Layer_Guesses/check_clinical_gene_sets.R to see a full script from where this family of functions is derived from.

Value

A plot visualizing the gene set enrichment odds ratio and p-value results.

Author(s)

Andrew E Jaffe, Leonardo Collado-Torres

See Also

layer_matrix_plot

Other Gene set enrichment functions: [gene_set_enrichment\(\)](#)

Examples

```
## Read in the SFARI gene sets included in the package
asd_sfari <- utils::read.csv(
  system.file(
    "extdata",
    "SFARI-Gene_genes_01-03-2020release_02-04-2020export.csv",
    package = "spatialLIBD"
  ),
  as.is = TRUE
)

## Format them appropriately
asd_sfari_geneList <- list(
  Gene_SFARI_all = asd_sfari$ensembl.id,
  Gene_SFARI_high = asd_sfari$ensembl.id[asd_sfari$gene.score < 3],
  Gene_SFARI_syndromic = asd_sfari$ensembl.id[asd_sfari$syndromic == 1]
)

## Obtain the necessary data
if (!exists("modeling_results")) {
  modeling_results <- fetch_data(type = "modeling_results")
}

## Compute the gene set enrichment results
asd_sfari_enrichment <- gene_set_enrichment(
  gene_list = asd_sfari_geneList,
  modeling_results = modeling_results,
  model_type = "enrichment"
)

## Visualize the gene set enrichment results
## with a custom color palette
gene_set_enrichment_plot(
  asd_sfari_enrichment,
  xlabs = gsub(".*_", "", unique(asd_sfari_enrichment$ID)),
  mypal = c(
    "white",
    grDevices::colorRampPalette(
      RColorBrewer::brewer.pal(9, "BuGn")
    )(50)
  )
)

## Specify the layer heights so it resembles more the length of each
## layer in the brain
gene_set_enrichment_plot(
```

```

    asd_sfari_enrichment,
    xlabs = gsub(".*_", "", unique(asd_sfari_enrichment$ID)),
    layerHeights = c(0, 40, 55, 75, 85, 110, 120, 135),
  )

```

 geom_spatial

A ggplot2 layer for visualizing the Visium histology

Description

This function defines a `ggplot2::layer()` for visualizing the histology image from Visium. It can be combined with other `ggplot2` functions for visualizing the clusters as in `sce_image_clus_p()` or gene-level information as in `sce_image_gene_p()`.

Usage

```

geom_spatial(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = FALSE,
  ...
)

```

Arguments

<code>mapping</code>	Passed to <code>ggplot2::layer(mapping)</code> where <code>grob</code> , <code>x</code> and <code>y</code> are required.
<code>data</code>	Passed to <code>ggplot2::layer(data)</code> .
<code>stat</code>	Passed to <code>ggplot2::layer(stat)</code> .
<code>position</code>	Passed to <code>ggplot2::layer(position)</code> .
<code>na.rm</code>	Passed to <code>ggplot2::layer(params = list(na.rm))</code> .
<code>show.legend</code>	Passed to <code>ggplot2::layer(show.legend)</code> .
<code>inherit.aes</code>	Passed to <code>ggplot2::layer(inherit.aes)</code> .
<code>...</code>	Other arguments passed to <code>ggplot2::layer(params = list(...))</code> .

Value

A `ggplot2::layer()` for the histology information.

Author(s)

10x Genomics

Examples

```

if (enough_ram()) {
  ## Obtain the necessary data
  if (!exists("ve")) ve <- fetch_data("ve")

  ## Select the first sample and extract the data
  sample_id <- unique(SpatialExperiment::spatialCoords(ve)$sample_name)[1]
  ve_sub <- ve[, SpatialExperiment::spatialCoords(ve)$sample_name == sample_id]
  sample_df <- as.data.frame(SpatialExperiment::spatialCoords(ve_sub))

  ## Make a plot using geom_spatial
  p <- ggplot2::ggplot(
    sample_df,
    ggplot2::aes(
      x = imagecol,
      y = imagerow,
    )
  ) +
    geom_spatial(
      data = read_image(ve_sub, sample_id),
      ggplot2::aes(grob = grob),
      x = 0.5,
      y = 0.5
    )

  ## Show the plot
  print(p)

  ## Clean up
  rm(ve_sub)
}

```

get_colors

Obtain the colors for a set of cluster names

Description

This function returns a vector of colors based on a vector of cluster names. It can be used to automatically assign colors.

Usage

```
get_colors(colors = NULL, clusters)
```

Arguments

colors	A vector of colors. If NULL then a set of default colors will be used when clusters has less than 12 unique values, otherwise palette36.colors will be used which can generate up to 36 unique colors. If the number of unique clusters is beyond 36 then this function will fail.
clusters	A vector of cluster names.

Value

A named vector where the values are the colors to use for displaying them different clusters. For some use cases, you might have to either change the names or use `unname()`.

Examples

```
## Obtain the necessary data
if (!exists("sce_layer")) sce_layer <- fetch_data("sce_layer")

## Example layer colors with the corresponding names
get_colors(libd_layer_colors, sce_layer$layer_guess)
get_colors(libd_layer_colors, sce_layer$layer_guess_reordered_short)

## Example where colors are assigned automatically
## based on a pre-defined set of colors
get_colors(clusters = sce_layer$kmeans_k7)

## Example where Polychrome::palette36.colors() gets used
get_colors(clusters = letters[seq_len(13)])
```

layer_boxplot

*Layer-level (group-level) boxplots***Description**

This function uses the output of `sig_genes_extract_all()` as well as the logcounts from the layer-level (group-level) data to visualize the expression of a given gene and display the modeling results for the given gene.

Usage

```
layer_boxplot(
  i = 1,
  sig_genes = sig_genes_extract(),
  short_title = TRUE,
  sce_layer = fetch_data(type = "sce_layer"),
  col_bkg_box = "grey80",
  col_bkg_point = "grey40",
  col_low_box = "violet",
  col_low_point = "darkviolet",
  col_high_box = "skyblue",
  col_high_point = "dodgerblue4",
  cex = 2
)
```

Arguments

<code>i</code>	A integer(1) indicating which row of <code>sig_genes</code> do you want to plot.
<code>sig_genes</code>	The output of <code>sig_genes_extract_all()</code> .
<code>short_title</code>	A logical(1) indicating whether to print a short title or not.

sce_layer	Defaults to the output of <code>fetch_data(type = 'sce_layer')</code> . This is a Single-CellExperiment object with the spot-level Visium data compressed via pseudo-bulking to the layer-level (group-level) resolution. See <code>fetch_data()</code> for more details.
col_bkg_box	Box background color for layers not used when visualizing the pairwise model results.
col_bkg_point	Similar to <code>col_bkg_box</code> but for the points.
col_low_box	Box background color for layer(s) with the expected lower expression based on the actual test for row <code>i</code> of <code>sig_genes</code> .
col_low_point	Similar to <code>col_low_box</code> but for the points.
col_high_box	Similar to <code>col_low_box</code> but for the expected layer(s) with higher expression.
col_high_point	Similar to <code>col_high_box</code> but for the points.
cex	Controls the size of the text, points and axis legends.

Value

This function creates a boxplot of the layer-level data (group-level) separated by layer and colored based on the model type from row `i` of `sig_genes`.

References

Adapted from https://github.com/LieberInstitute/HumanPilot/blob/master/Analysis/Layer_Guesses/layer_specificity.R

See Also

Other Layer modeling functions: [sig_genes_extract_all\(\)](#), [sig_genes_extract\(\)](#)

Examples

```
## Obtain the necessary data
if (!exists("modeling_results")) {
  modeling_results <- fetch_data(type = "modeling_results")
}
if (!exists("sce_layer")) sce_layer <- fetch_data(type = "sce_layer")

## Top 2 genes from the enrichment model
sig_genes <- sig_genes_extract_all(
  n = 2,
  modeling_results = modeling_results,
  sce_layer = sce_layer
)

## Example default boxplot
set.seed(20200206)
layer_boxplot(sig_genes = sig_genes, sce_layer = sce_layer)

## Now show the long title version
set.seed(20200206)
layer_boxplot(
  sig_genes = sig_genes,
  short_title = FALSE,
  sce_layer = sce_layer
)
```

```

)

set.seed(20200206)
layer_boxplot(
  i = which(sig_genes$model_type == "anova")[1],
  sig_genes = sig_genes,
  sce_layer = sce_layer
)

set.seed(20200206)
layer_boxplot(
  i = which(sig_genes$model_type == "pairwise")[1],
  sig_genes = sig_genes,
  sce_layer = sce_layer
)

## Viridis colors displayed in the shiny app
library("viridisLite")
set.seed(20200206)
layer_boxplot(
  sig_genes = sig_genes,
  sce_layer = sce_layer,
  col_low_box = viridis(4)[2],
  col_low_point = viridis(4)[1],
  col_high_box = viridis(4)[3],
  col_high_point = viridis(4)[4]
)

## Paper colors displayed in the shiny app
set.seed(20200206)
layer_boxplot(
  sig_genes = sig_genes,
  sce_layer = sce_layer,
  col_low_box = "palegreen3",
  col_low_point = "springgreen2",
  col_high_box = "darkorange2",
  col_high_point = "orange1"
)

## Blue/red colors displayed in the shiny app
set.seed(20200206)
layer_boxplot(
  i = which(sig_genes$model_type == "pairwise")[1],
  sig_genes = sig_genes,
  sce_layer = sce_layer,
  col_bkg_box = "grey90",
  col_bkg_point = "grey60",
  col_low_box = "skyblue2",
  col_low_point = "royalblue3",
  col_high_box = "tomato2",
  col_high_point = "firebrick4",
  cex = 3
)

```


Description

This function visualizes a numerical matrix where the Y-axis represents the human brain layers and can be adjusted to represent the length of each brain layer. Cells can optionally have text values. This function is used by [gene_set_enrichment_plot\(\)](#) and [layer_stat_cor_plot\(\)](#).

Usage

```
layer_matrix_plot(
  matrix_values,
  matrix_labels = NULL,
  xlabs = NULL,
  layerHeights = NULL,
  mypal = c("white", (grDevices::colorRampPalette(RColorBrewer::brewer.pal(9,
    "YlOrRd")))(50)),
  breaks = NULL,
  axis.args = NULL,
  srt = 45,
  mar = c(8, 4, 4, 2) + 0.1,
  cex = 1.2
)
```

Arguments

<code>matrix_values</code>	A <code>matrix()</code> with one column per set of interest and one row per layer (group) with numeric values.
<code>matrix_labels</code>	Optionally a character <code>matrix()</code> with the same dimensions and <code>dimnames()</code> as <code>matrix_values</code> with text labels for the cells.
<code>xlabs</code>	A vector of names in the same order and length as <code>colnames(matrix_values)</code> .
<code>layerHeights</code>	A <code>numeric()</code> vector of length equal to <code>nrow(matrix_values) + 1</code> that starts at 0 specifying where to plot the y-axis breaks which can be used for re-creating the length of each brain layer.
<code>mypal</code>	A vector with the color palette to use.
<code>breaks</code>	Passed to <code>fields::image.plot()</code> . Used by <code>layer_stat_cor_plot()</code> .
<code>axis.args</code>	Passed to <code>fields::image.plot()</code> . Used by <code>layer_stat_cor_plot()</code> .
<code>srt</code>	The angle for the x-axis labels. Used by <code>layer_stat_cor_plot()</code> .
<code>mar</code>	Passed to <code>graphics::par()</code> .
<code>cex</code>	Used for the x-axis labels and the text inside the cells.

Value

A base R plot visualizing the input `matrix_values` with optional text labels for `matrix_labels`.

Author(s)

Andrew E Jaffe, Leonardo Collado-Torres

Examples

```
## Create some random data
set.seed(20200224)
mat <- matrix(runif(7 * 8, min = -1), nrow = 7)
rownames(mat) <- c("WM", paste0("L", rev(seq_len(6))))
colnames(mat) <- paste0("Var", seq_len(8))

## Create some text labels
mat_text <- matrix("", nrow = 7, ncol = 8, dimnames = dimnames(mat))
diag(mat_text) <- as.character(round(diag(mat), 2))

## Make the plot
layer_matrix_plot(mat, mat_text)

## Try to re-create the anatomical proportions of the human brain layers
layer_matrix_plot(
  mat,
  mat_text,
  layerHeights = c(0, 40, 55, 75, 85, 110, 120, 135),
  cex = 2
)
```

layer_stat_cor

*Layer modeling correlation of statistics***Description**

Layer modeling correlation of statistics

Usage

```
layer_stat_cor(
  stats,
  modeling_results = fetch_data(type = "modeling_results"),
  model_type = names(modeling_results)[1],
  reverse = FALSE
)
```

Arguments

stats A data.frame where the row names are Ensembl gene IDs, the column names are labels for clusters of cells or cell types, and where each cell contains the given statistic for that gene and cell type. These statistics should be computed similarly to the modeling results from the data we provide. For example, like the enrichment t-statistics that are derived from comparing one layer against the rest. The stats will be matched and then correlated with our statistics.

modeling_results Defaults to the output of `fetch_data(type = 'modeling_results')`. This is a list of tables with the columns `f_stat_*` or `t_stat_*` as well as `p_value_*` and `fdr_*` plus `ensembl`. The column name is used to extract the statistic results, the p-values, and the FDR adjusted p-values. Then the `ensembl` column is used for matching in some cases. See [fetch_data\(\)](#) for more details.

model_type	A named element of the modeling_results list. By default that is either enrichment for the model that tests one human brain layer against the rest (one group vs the rest), pairwise which compares two layers (groups) denoted by layerA-layerB such that layerA is greater than layerB, and anova which determines if any layer (group) is different from the rest adjusting for the mean expression level. The statistics for enrichment and pairwise are t-statistics while the anova model ones are F-statistics.
reverse	A logical(1) indicating whether to multiply by -1 the input statistics and reverse the layerA-layerB column names (using the -) into layerB-layerA.

Details

Check https://github.com/LieberInstitute/HumanPilot/blob/master/Analysis/Layer_Guesses/dlpfc_snRNAseq_annotation for a full analysis from which this family of functions is derived from.

Value

A correlation matrix between stats and our statistics using only the Ensembl gene IDs present in both tables. The columns are sorted using a hierarchical cluster.

Author(s)

Andrew E Jaffe, Leonardo Collado-Torres

See Also

Other Layer correlation functions: [layer_stat_cor_plot\(\)](#)

Examples

```
## Obtain the necessary data
if (!exists("modeling_results")) {
  modeling_results <- fetch_data(type = "modeling_results")
}

## Compute the correlations
cor_stats_layer <- layer_stat_cor(
  tstats_Human_DLPFC_snRNAseq_Nguyen_topLayer,
  modeling_results,
  "enrichment"
)

## Explore the correlation matrix
head(cor_stats_layer[, seq_len(3)])
```

layer_stat_cor_plot *Visualize the layer modeling correlation of statistics*

Description

This function makes a heatmap from the [layer_stat_cor\(\)](#) correlation matrix between a given set of cell cluster/type statistics derived from scRNA-seq or snRNA-seq data (among other types) and the layer statistics from the Human DLPFC Visium data (when using the default arguments).

Usage

```
layer_stat_cor_plot(
  cor_stats_layer,
  max = 0.81,
  min = -max,
  layerHeights = NULL,
  cex = 1.2
)
```

Arguments

cor_stats_layer	The output of <code>layer_stat_cor()</code> .
max	A numeric(1) specifying the highest correlation value for the color scale (should be between 0 and 1).
min	A numeric(1) specifying the lowest correlation value for the color scale (should be between 0 and -1).
layerHeights	A numeric() vector of length equal to <code>ncol(cor_stats_layer) + 1</code> that starts at 0 specifying where to plot the y-axis breaks which can be used for re-creating the length of each brain layer. Gets passed to <code>layer_matrix_plot()</code> .
cex	Passed to <code>layer_matrix_plot()</code> .

Details

Check https://github.com/LieberInstitute/HumanPilot/blob/master/Analysis/Layer_Guesses/dlpfc_snRNAseq_annotation for a full analysis from which this family of functions is derived from.

Value

A heatmap for the correlation matrix between statistics.

Author(s)

Andrew E Jaffe, Leonardo Collado-Torres

See Also

`layer_matrix_plot`
Other Layer correlation functions: `layer_stat_cor()`

Examples

```
## Obtain the necessary data
if (!exists("modeling_results")) {
  modeling_results <- fetch_data(type = "modeling_results")
}

## Compute the correlations
cor_stats_layer <- layer_stat_cor(
  tstats_Human_DLPFC_snRNAseq_Nguyen_topLayer,
  modeling_results,
  "enrichment"
```

```

)

## Visualize the correlation matrix
layer_stat_cor_plot(cor_stats_layer)

## Restrict the range of colors
layer_stat_cor_plot(cor_stats_layer, max = 0.3)

```

libd_layer_colors *Vector of LIBD layer colors*

Description

A named vector of colors to use for the LIBD layers designed by Lukas M. Weber with feedback from the spatialLIBD collaborators.

Usage

```
libd_layer_colors
```

Format

A vector of length 9 with colors for Layers 1 through 9, WM, NA and a special WM2 that is present in some of the unsupervised clustering results.

read_image *Read image*

Description

Helper function for `geom_plot()` that was needed in order to complete `sce_to_ve()`. It generates the `grid::rasterGrob()` data needed by `geom_plot()` that was previously included in the output from `fetch_data("sce")`.

Usage

```
read_image(ve, sample_id = names(imagePaths(ve))[1])
```

Arguments

`ve` A [VisiumExperiment-class](#) object created, such as one created by `sce_to_ve()`.
`sample_id` A character(1) specifying the sample ID to work with.

Value

A `tibble::tibble()` with a `grob` column that is a `list()` with a `grid::rasterGrob()` object.
 @author Brenda Pardo, Leonardo Collado-Torres

See Also

Other VisiumExperiment-related functions: [sce_to_ve\(\)](#), [ve_image_colData\(\)](#)

run_app

*Run the spatialLIBD Shiny Application***Description**

This function runs the shiny application that allows users to interact with the Visium spatial transcriptomics data from LIBD (by default) or any other data that you have shaped according to our object structure.

Usage

```
run_app(
  sce = fetch_data(type = "sce"),
  sce_layer = fetch_data(type = "sce_layer"),
  modeling_results = fetch_data(type = "modeling_results"),
  sig_genes = sig_genes_extract_all(n = nrow(sce_layer), modeling_results =
    modeling_results, sce_layer = sce_layer),
  image_path = system.file("app", "www", "data", package = "spatialLIBD"),
  sce_discrete_vars = c("GraphBased", "Layer", "Maynard", "Martinowich",
    paste0("SNN_k50_k", 4:28), "SpatialDE_PCA", "SpatialDE_pool_PCA", "HVG_PCA",
    "pseudobulk_PCA", "markers_PCA", "SpatialDE_UMAP", "SpatialDE_pool_UMAP", "HVG_UMAP",
    "pseudobulk_UMAP", "markers_UMAP", "SpatialDE_PCA_spatial",
    "SpatialDE_pool_PCA_spatial", "HVG_PCA_spatial", "pseudobulk_PCA_spatial",
    "markers_PCA_spatial", "SpatialDE_UMAP_spatial", "SpatialDE_pool_UMAP_spatial",
    "HVG_UMAP_spatial", "pseudobulk_UMAP_spatial", "markers_UMAP_spatial"),
  sce_continuous_vars = c("cell_count", "sum_umi", "sum_gene", "expr_chrM",
    "expr_chrM_ratio"),
  spatial_libd_var = "layer_guess_reordered_short",
  ...
)
```

Arguments

sce	Defaults to the output of <code>fetch_data(type = 'sce')</code> . This is a SingleCellExperiment object with the spot-level Visium data and information required for visualizing the histology. See <code>fetch_data()</code> for more details.
sce_layer	Defaults to the output of <code>fetch_data(type = 'sce_layer')</code> . This is a SingleCellExperiment object with the spot-level Visium data compressed via pseudo-bulking to the layer-level (group-level) resolution. See <code>fetch_data()</code> for more details.
modeling_results	Defaults to the output of <code>fetch_data(type = 'modeling_results')</code> . This is a list of tables with the columns <code>f_stat_*</code> or <code>t_stat_*</code> as well as <code>p_value_*</code> and <code>fdr_*</code> plus <code>ensembl</code> . The column name is used to extract the statistic results, the p-values, and the FDR adjusted p-values. Then the <code>ensembl</code> column is used for matching in some cases. See <code>fetch_data()</code> for more details.
sig_genes	The output of <code>sig_genes_extract_all()</code> which is a table in long format with the modeling results.
image_path	A path to the directory containing the low resolution histology images that is needed for the interactive visualizations with <code>plotly</code> . See https://github.com/LieberInstitute/spatialLIBD for an example of how these files should be organized.

`sce_discrete_vars`
 A `character()` vector of discrete variables that will be available to visualize in the app. Basically, the set of variables with spot-level groups. They will have to be present in `colData(sce)`.

`sce_continuous_vars`
 A `character()` vector of continuous variables that will be available to visualize in the app using the same scale as genes. They will have to be present in `colData(sce)`.

`spatial_libd_var`
 A `character(1)` with the name of the main cluster variable to use. It will have to be present in both `colData(sce)` and `colData(sce_layer)`.

...
 Other arguments passed to the list of `golem` options for running the application.

Value

A `shiny.appobj` that contains the input data.

Examples

```
## Not run:
## The default arguments will download the data from the web
## using fetch_data(). If this is the first time you have run this,
## the files will need to be cached by ExperimentHub. Otherwise it
## will re-use the files you have previously downloaded.
if (enough_ram(4e9)) {
  run_app()
}

## End(Not run)
```

`sce_image_clus` *Sample spatial cluster visualization*

Description

This function visualizes the clusters for one given sample at the spot-level using (by default) the histology information on the background. To visualize gene-level (or any continuous variable) use `sce_image_gene()`.

Usage

```
sce_image_clus(
  sce,
  sampleid,
  clustervar,
  colors = c("#b2df8a", "#e41a1c", "#377eb8", "#4daf4a", "#ff7f00", "gold", "#a65628",
    "#999999", "black", "grey", "white", "purple"),
  spatial = TRUE,
  ...
)
```

Arguments

sce	Defaults to the output of <code>fetch_data(type = 'sce')</code> . This is a SingleCellExperiment object with the spot-level Visium data and information required for visualizing the histology. See <code>fetch_data()</code> for more details.
sampleid	A character(1) specifying which sample to plot from <code>colData(sce)\$sample_name</code> .
clustervar	A character(1) with the name of the <code>colData(sce)</code> column that has the cluster values.
colors	A vector of colors to use for visualizing the clusters from <code>clustervar</code> . If the vector has names, then those should match the values of <code>clustervar</code> .
spatial	A logical(1) indicating whether to include the histology layer from <code>geom_spatial()</code> . If you plan to use <code>ggplotly()</code> then it's best to set this to FALSE.
...	Passed to <code>paste0()</code> for making the title of the plot following the <code>sampleid</code> .

Details

This function subsets `sce` to the given sample and prepares the data and title for `sce_image_clus_p()`.

Value

A [ggplot2](#) object.

See Also

Other Spatial cluster visualization functions: `sce_image_clus_p()`, `sce_image_grid()`

Examples

```
if (enough_ram()) {
  ## Obtain the necessary data
  if (!exists("sce")) sce <- fetch_data("sce")

  ## Check the colors defined by Lukas M Weber
  libd_layer_colors

  ## Use the manual color palette by Lukas M Weber
  sce_image_clus(
    sce = sce,
    clustervar = "layer_guess_reordered",
    sampleid = "151673",
    colors = libd_layer_colors,
    ... = " LIBD Layers"
  )

  ## Works also with VisiumExperiment objects.
  sce_image_clus(
    sce = sce_to_ve(sce),
    clustervar = "layer_guess_reordered",
    sampleid = "151673",
    colors = libd_layer_colors,
    ... = " LIBD Layers"
  )
}
```



```

## Without histology
sce_image_clus(
  sce = sce,
  clustervar = "layer_guess_reordered",
  sampleid = "151673",
  colors = libd_layer_colors,
  ... = " LIBD Layers",
  spatial = FALSE
)
}

```

sce_image_clus_p *Sample spatial cluster visualization workhorse function*

Description

This function visualizes the clusters for one given sample at the spot-level using (by default) the histology information on the background. This is the function that does all the plotting behind [sce_image_clus\(\)](#). To visualize gene-level (or any continuous variable) use [sce_image_gene_p\(\)](#).

Usage

```
sce_image_clus_p(sce, d, clustervar, sampleid, colors, spatial, title)
```

Arguments

sce	Defaults to the output of <code>fetch_data(type = 'sce')</code> . This is a SingleCellExperiment object with the spot-level Visium data and information required for visualizing the histology. See fetch_data() for more details.
d	A data.frame with the sample-level information. This is typically obtained using <code>as.data.frame(colData(sce))</code> .
clustervar	A character(1) with the name of the <code>colData(sce)</code> column that has the cluster values.
sampleid	A character(1) specifying which sample to plot from <code>colData(sce)\$sample_name</code> .
colors	A vector of colors to use for visualizing the clusters from <code>clustervar</code> . If the vector has names, then those should match the values of <code>clustervar</code> .
spatial	A logical(1) indicating whether to include the histology layer from geom_spatial() . If you plan to use ggplotly() then it's best to set this to FALSE.
title	The title for the plot.

Value

A [ggplot2](#) object.

See Also

Other Spatial cluster visualization functions: [sce_image_clus\(\)](#), [sce_image_grid\(\)](#)

Examples

```

if (enough_ram()) {
  ## Obtain the necessary data
  if (!exists("sce")) sce <- fetch_data("sce")
  sce_sub <- sce[, sce$sample_name == "151673"]

  ## Use the manual color palette by Lukas M Weber
  ## Don't plot the histology information
  sce_image_clus_p(
    sce = sce_sub,
    d = as.data.frame(colData(sce_sub)),
    clustervar = "layer_guess_reordered",
    sampleid = "151673",
    colors = libd_layer_colors,
    title = "151673 LIBD Layers",
    spatial = FALSE
  )

  ## Clean up
  rm(sce_sub)
}

```

sce_image_gene

Sample spatial gene visualization

Description

This function visualizes the gene expression stored in `assays(sce)` or any continuous variable stored in `colData(sce)` for one given sample at the spot-level using (by default) the histology information on the background. To visualize clusters (or any discrete variable) use `sce_image_clus()`.

Usage

```

sce_image_gene(
  sce,
  sampleid,
  geneid = "SCGB2A2; ENSG00000110484",
  spatial = TRUE,
  assayname = "logcounts",
  minCount = 0,
  viridis = TRUE,
  ...
)

```

Arguments

<code>sce</code>	Defaults to the output of <code>fetch_data(type = 'sce')</code> . This is a SingleCellExperiment object with the spot-level Visium data and information required for visualizing the histology. See <code>fetch_data()</code> for more details.
<code>sampleid</code>	A character(1) specifying which sample to plot from <code>colData(sce)\$sample_name</code> .

geneid	A character(1) specifying the gene ID stored in <code>rowData(sce)\$gene_search</code> or a continuous variable stored in <code>colData(sce)</code> to visualize.
spatial	A logical(1) indicating whether to include the histology layer from <code>geom_spatial()</code> . If you plan to use <code>ggplotly()</code> then it's best to set this to FALSE.
assayname	The name of the assays(<code>sce</code>) to use for extracting the gene expression data. Defaults to <code>logcounts</code> .
minCount	A numeric(1) specifying the minimum gene expression (or value in the continuous variable) to visualize. Values at or below this threshold will be set to NA. Defaults to 0.
viridis	A logical(1) whether to use the color-blind friendly palette from <code>viridis</code> or the color palette used in the paper that was chosen for contrast when visualizing the data on top of the histology image. One issue is being able to differentiate low values from NA ones due to the purple-ish histology information that is dependent on cell density.
...	Passed to <code>paste0()</code> for making the title of the plot following the <code>sampleid</code> .

Details

This function subsets `sce` to the given sample and prepares the data and title for `sce_image_gene_p()`. It also adds a caption to the plot.

Value

A `ggplot2` object.

See Also

Other Spatial gene visualization functions: `sce_image_gene_p()`, `sce_image_grid_gene()`

Examples

```
if (enough_ram()) {
  ## Obtain the necessary data
  if (!exists("sce")) sce <- fetch_data("sce")

  ## Valid `geneid` values are those in
  head(rowData(sce)$gene_search)
  ## or continuous variables stored in colData(sce)

  ## Visualize a default gene on the non-viridis scale
  sce_image_gene(
    sce = sce,
    sampleid = "151507",
    viridis = FALSE
  )

  ## Works also with VisiumExperiment objects
  sce_image_gene(
    sce = sce_to_ve(sce),
    sampleid = "151507",
    viridis = FALSE
  )
}
```

```

## Visualize a continuous variable, in this case, the ratio of chrM
## gene expression compared to the total expression at the spot-level
sce_image_gene(
  sce = sce,
  sampleid = "151507",
  geneid = "expr_chrM_ratio"
)
}

```

sce_image_gene_p

Sample spatial gene visualization workhorse function

Description

This function visualizes the gene expression stored in `assays(sce)` or any continuous variable stored in `colData(sce)` for one given sample at the spot-level using (by default) the histology information on the background. This is the function that does all the plotting behind `sce_image_gene()`. To visualize clusters (or any discrete variable) use `sce_image_clus_p()`.

Usage

```
sce_image_gene_p(sce, d, sampleid, spatial, title, viridis = TRUE)
```

Arguments

<code>sce</code>	Defaults to the output of <code>fetch_data(type = 'sce')</code> . This is a SingleCellExperiment object with the spot-level Visium data and information required for visualizing the histology. See <code>fetch_data()</code> for more details.
<code>d</code>	A <code>data.frame</code> with the sample-level information. This is typically obtained using <code>as.data.frame(colData(sce))</code> . The <code>data.frame</code> has to contain a column with the continuous variable data to plot stored under <code>d\$COUNT</code> .
<code>sampleid</code>	A <code>character(1)</code> specifying which sample to plot from <code>colData(sce)\$sample_name</code> .
<code>spatial</code>	A <code>logical(1)</code> indicating whether to include the histology layer from <code>geom_spatial()</code> . If you plan to use <code>ggplotly()</code> then it's best to set this to <code>FALSE</code> .
<code>title</code>	The title for the plot.
<code>viridis</code>	A <code>logical(1)</code> whether to use the color-blind friendly palette from viridis or the color palette used in the paper that was chosen for contrast when visualizing the data on top of the histology image. One issue is being able to differentiate low values from NA ones due to the purple-ish histology information that is dependent on cell density.

Value

A `ggplot2` object.

See Also

Other Spatial gene visualization functions: `sce_image_gene()`, `sce_image_grid_gene()`

Examples

```

if (enough_ram()) {
  ## Obtain the necessary data
  if (!exists("sce")) sce <- fetch_data("sce")

  ## Prepare the data for the plotting function
  sce_sub <- sce[, sce$sample_name == "151673"]
  df <- as.data.frame(colData(sce_sub))
  df$COUNT <- df$expr_chrM_ratio

  ## Use the manual color palette by Lukas M Weber
  ## Don't plot the histology information
  sce_image_gene_p(
    sce = sce_sub,
    d = df,
    sampleid = "151673",
    title = "151673 chrM expr ratio",
    spatial = FALSE
  )

  ## Or you can do this with a VisiumExperiment object
  ve_sub <- sce_to_ve(sce_sub)
  df2 <- colData(ve_sub)
  df2$COUNT <- df2$expr_chrM_ratio

  sce_image_gene_p(
    sce = ve_sub,
    d = df2,
    sampleid = "151673",
    title = "151673 chrM expr ratio",
    spatial = FALSE
  )

  ## Clean up
  rm(sce_sub)
  rm(ve_sub)
}

```

sce_image_grid

Sample spatial cluster visualization grid

Description

This function visualizes the clusters for a set of samples at the spot-level using (by default) the histology information on the background. To visualize gene-level (or any continuous variable) use [sce_image_grid_gene\(\)](#).

Usage

```

sce_image_grid(
  sce,

```

```

    clustervar,
    pdf_file,
    sort_clust = TRUE,
    colors = NULL,
    return_plots = FALSE,
    spatial = TRUE,
    ...
)

```

Arguments

sce	Defaults to the output of <code>fetch_data(type = 'sce')</code> . This is a SingleCellExperiment object with the spot-level Visium data and information required for visualizing the histology. See <code>fetch_data()</code> for more details.
clustervar	A character(1) with the name of the <code>colData(sce)</code> column that has the cluster values.
pdf_file	A character(1) specifying the path for the resulting PDF.
sort_clust	A logical(1) indicating whether you want to sort the clusters by frequency using <code>sort_clusters()</code> .
colors	A vector of colors to use for visualizing the clusters from <code>clustervar</code> . If the vector has names, then those should match the values of <code>clustervar</code> .
return_plots	A logical(1) indicating whether to print the plots to a PDF or to return the list of plots that you can then print using <code>plot_grid</code> .
spatial	A logical(1) indicating whether to include the histology layer from <code>geom_spatial()</code> . If you plan to use <code>ggplotly()</code> then it's best to set this to FALSE.
...	Passed to <code>paste0()</code> for making the title of the plot following the <code>sampleid</code> .

Details

This function prepares the data and then loops through `sce_image_clus()` for computing the list of `ggplot2` objects.

Value

A list of `ggplot2` objects.

See Also

Other Spatial cluster visualization functions: `sce_image_clus_p()`, `sce_image_clus()`

Examples

```

if (enough_ram()) {
  ## Obtain the necessary data
  if (!exists("sce")) sce <- fetch_data("sce")

  ## Subset to two samples of interest
  sce_sub <- sce[, sce$sample_name %in% c("151673", "151674")]

  ## Obtain the plot list
  p_list <-

```

```

    sce_image_grid(
      sce_sub,
      "layer_guess_reordered",
      spatial = FALSE,
      return_plots = TRUE,
      sort_clust = FALSE,
      colors = libd_layer_colors
    )

  ## Or you can do this with a VisiumExperiment object
  ve_sub <- sce_to_ve(sce_sub)
  p_list <-
    sce_image_grid(
      ve_sub,
      "layer_guess_reordered",
      spatial = FALSE,
      return_plots = TRUE,
      sort_clust = FALSE,
      colors = libd_layer_colors
    )

  ## Clean up
  rm(sce_sub)

  ## Visualize the spatial adjacent replicates for position = 0 micro meters
  ## for subject 3
  cowplot::plot_grid(plotlist = p_list, ncol = 2)
}

```

sce_image_grid_gene *Sample spatial gene visualization grid*

Description

This function visualizes the gene expression stored in `assays(sce)` or any continuous variable stored in `colData(sce)` for a set of samples at the spot-level using (by default) the histology information on the background. To visualize clusters (or any discrete variable) use `sce_image_grid()`.

Usage

```

sce_image_grid_gene(
  sce,
  geneid = "SCGB2A2; ENSG00000110484",
  pdf_file,
  assayname = "logcounts",
  minCount = 0,
  return_plots = FALSE,
  spatial = TRUE,
  viridis = TRUE,
  ...
)

```

Arguments

sce	Defaults to the output of <code>fetch_data(type = 'sce')</code> . This is a SingleCellExperiment object with the spot-level Visium data and information required for visualizing the histology. See <code>fetch_data()</code> for more details.
geneid	A character(1) specifying the gene ID stored in <code>rowData(sce)\$gene_search</code> or a continuous variable stored in <code>colData(sce)</code> to visualize.
pdf_file	A character(1) specifying the path for the resulting PDF.
assayname	The name of the assays(sce) to use for extracting the gene expression data. Defaults to <code>logcounts</code> .
minCount	A numeric(1) specifying the minimum gene expression (or value in the continuous variable) to visualize. Values at or below this threshold will be set to NA. Defaults to 0.
return_plots	A logical(1) indicating whether to print the plots to a PDF or to return the list of plots that you can then print using <code>plot_grid</code> .
spatial	A logical(1) indicating whether to include the histology layer from <code>geom_spatial()</code> . If you plan to use <code>ggplotly()</code> then it's best to set this to FALSE.
viridis	A logical(1) whether to use the color-blind friendly palette from <code>viridis</code> or the color palette used in the paper that was chosen for contrast when visualizing the data on top of the histology image. One issue is being able to differentiate low values from NA ones due to the purple-ish histology information that is dependent on cell density.
...	Passed to <code>paste0()</code> for making the title of the plot following the <code>sampleid</code> .

Details

This function prepares the data and then loops through `sce_image_gene()` for computing the list of `ggplot2` objects.

Value

A list of `ggplot2` objects.

See Also

Other Spatial gene visualization functions: `sce_image_gene_p()`, `sce_image_gene()`

Examples

```
if (enough_ram()) {
  ## Obtain the necessary data
  if (!exists("sce")) sce <- fetch_data("sce")

  ## Subset to two samples of interest
  sce_sub <- sce[, sce$sample_name %in% c("151673", "151674")]

  ## Obtain the plot list
  p_list <-
    sce_image_grid_gene(
      sce_sub,
      spatial = FALSE,
```



```

        return_plots = TRUE
    )

    ## Or you can do this with a VisiumExperiment object
    ve_sub <- sce_to_ve(sce_sub)
    p_list <-
      sce_image_grid_gene(
        ve_sub,
        spatial = FALSE,
        return_plots = TRUE
      )

    ## Clean up
    rm(sce_sub)

    ## Visualize the spatial adjacent replicates for position = 0 micro meters
    ## for subject 3
    cowplot::plot_grid(plotlist = p_list, ncol = 2)
  }

```

sce_to_ve

Convert a SCE object to a VE one

Description

This function converts a spot-level [SingleCellExperiment-class](#) (SCE) object as generated by `fetch_data()` to a [VisiumExperiment-class](#) (VE) object.

Usage

```
sce_to_ve(sce = fetch_data("sce"), bfc = BiocFileCache::BiocFileCache())
```

Arguments

sce	Defaults to the output of <code>fetch_data(type = 'sce')</code> . This is a SingleCellExperiment object with the spot-level Visium data and information required for visualizing the histology. See <code>fetch_data()</code> for more details.
bfc	A BiocFileCache object BiocFileCache-class . Used when eh is not available.

Details

Note that the resulting object is a bit more complex than a regular VE because it contains the data for multiple images. Also, the VE validity code checks that the images files exist locally instead of being available remotely through a URL. Thus, we decided to download the images the moment the VE object is created. To deal with the fact that we have multiple images, we implemented the function `update_scaleFactors()` and `read_image()` which is used in the example of `geom_plot()`.

Value

A [VisiumExperiment-class](#) object with some customization for our data. See `update_scaleFactors()` for more details.

Author(s)

Brenda Pardo, Leonardo Collado-Torres

See Also

Other VisiumExperiment-related functions: [read_image\(\)](#), [ve_image_colData\(\)](#)

Examples

```
if (enough_ram()) {
  ## Download the sce data
  sce <- fetch_data("sce")
  ## Transform it to a VisiumExperiment object
  ve <- sce_to_ve(sce)
}
```

<code>sig_genes_extract</code>	<i>Extract significant genes</i>
--------------------------------	----------------------------------

Description

From the layer-level modeling results, this function extracts the top n significant genes. This is the workhorse function used by [sig_genes_extract_all\(\)](#) through which we obtain the information that can then be used by functions such as [layer_boxplot\(\)](#) for constructing informative titles.

Usage

```
sig_genes_extract(
  n = 10,
  modeling_results = fetch_data(type = "modeling_results"),
  model_type = names(modeling_results)[1],
  reverse = FALSE,
  sce_layer = fetch_data(type = "sce_layer")
)
```

Arguments

<code>n</code>	The number of the top ranked genes to extract.
<code>modeling_results</code>	Defaults to the output of <code>fetch_data(type = 'modeling_results')</code> . This is a list of tables with the columns <code>f_stat_*</code> or <code>t_stat_*</code> as well as <code>p_value_*</code> and <code>fdr_*</code> plus <code>ensembl</code> . The column name is used to extract the statistic results, the p-values, and the FDR adjusted p-values. Then the <code>ensembl</code> column is used for matching in some cases. See fetch_data() for more details.
<code>model_type</code>	A named element of the <code>modeling_results</code> list. By default that is either <code>enrichment</code> for the model that tests one human brain layer against the rest (one group vs the rest), <code>pairwise</code> which compares two layers (groups) denoted by <code>layerA-layerB</code> such that <code>layerA</code> is greater than <code>layerB</code> , and <code>anova</code> which determines if any layer (group) is different from the rest adjusting for the mean expression level. The statistics for <code>enrichment</code> and <code>pairwise</code> are t-statistics while the <code>anova</code> model ones are F-statistics.

reverse	A logical(1) indicating whether to multiply by -1 the input statistics and reverse the layerA-layerB column names (using the -) into layerB-layerA.
sce_layer	Defaults to the output of <code>fetch_data(type = 'sce_layer')</code> . This is a Single-CellExperiment object with the spot-level Visium data compressed via pseudo-bulking to the layer-level (group-level) resolution. See <code>fetch_data()</code> for more details.

Value

A `data.frame()` with the top n significant genes (as ordered by their statistics in decreasing order) in long format. The specific columns are described further in the vignette.

References

Adapted from https://github.com/LieberInstitute/HumanPilot/blob/master/Analysis/Layer_Guesses/layer_specificity_fun

See Also

Other Layer modeling functions: `layer_boxplot()`, `sig_genes_extract_all()`

Examples

```
## Obtain the necessary data
if (!exists("modeling_results")) {
  modeling_results <- fetch_data(type = "modeling_results")
}
if (!exists("sce_layer")) sce_layer <- fetch_data(type = "sce_layer")

## anova top 10 genes
sig_genes_extract(
  modeling_results = modeling_results,
  sce_layer = sce_layer
)

## Extract all genes
sig_genes_extract(
  modeling_results = modeling_results,
  sce_layer = sce_layer,
  n = nrow(sce_layer)
)
```

`sig_genes_extract_all` *Extract significant genes for all modeling results*

Description

This function combines the output of `sig_genes_extract()` from all the layer-level (group-level) modeling results and builds the data required for functions such as `layer_boxplot()`.

Usage

```
sig_genes_extract_all(
  n = 10,
  modeling_results = fetch_data(type = "modeling_results"),
  sce_layer = fetch_data(type = "sce_layer")
)
```

Arguments

n	The number of the top ranked genes to extract.
modeling_results	Defaults to the output of <code>fetch_data(type = 'modeling_results')</code> . This is a list of tables with the columns <code>f_stat_*</code> or <code>t_stat_*</code> as well as <code>p_value_*</code> and <code>fdr_*</code> plus <code>ensembl</code> . The column name is used to extract the statistic results, the p-values, and the FDR adjusted p-values. Then the <code>ensembl</code> column is used for matching in some cases. See fetch_data() for more details.
sce_layer	Defaults to the output of <code>fetch_data(type = 'sce_layer')</code> . This is a Single-CellExperiment object with the spot-level Visium data compressed via pseudo-bulking to the layer-level (group-level) resolution. See fetch_data() for more details.

Value

A [DataFrame-class](#) with the extracted statistics in long format. The specific columns are described further in the vignette.

See Also

Other Layer modeling functions: [layer_boxplot\(\)](#), [sig_genes_extract\(\)](#)

Examples

```
## Obtain the necessary data
if (!exists("modeling_results")) {
  modeling_results <- fetch_data(type = "modeling_results")
}
if (!exists("sce_layer")) sce_layer <- fetch_data(type = "sce_layer")

## top 10 genes for all models
sig_genes_extract_all(
  modeling_results = modeling_results,
  sce_layer = sce_layer
)
```

 sort_clusters

Sort clusters by frequency

Description

This function takes a vector with cluster labels and sorts it by frequency such that the most frequent cluster is the first one and so on.

Usage

```
sort_clusters(clusters, map_subset = NULL)
```

Arguments

`clusters` A vector with cluster labels.

`map_subset` A logical vector of length equal to `clusters` specifying which elements of `clusters` to use to determine the ranking of the clusters.

Value

A factor of length equal to `clusters` where the levels are the new ordered clusters and the names of the factor are the original values from `clusters`.

Examples

```
## Build an initial set of cluster labels
clus <- letters[unlist(lapply(4:1, function(x) rep(x, x)))]

## In this case, it's a character vector
class(clus)

## Sort them and obtain a factor
sort_clusters(clus)
```

```
tstats_Human_DLPFC_snRNAseq_Nguyen_topLayer
```

Cell cluster t-statistics from Tran et al

Description

Using the DLPFC snRNA-seq data from Matthew N Tran et al we computed enrichment t-statistics for the cell clusters. This is a subset of them used in examples such as in [layer_stat_cor_plot\(\)](#).

Usage

```
tstats_Human_DLPFC_snRNAseq_Nguyen_topLayer
```

Format

A matrix with 692 rows and 31 variables where each column is a given cell cluster from Tran et al and each row is one gene. The row names are Ensembl gene IDs which are used by [layer_stat_cor\(\)](#) to match to our modeling results.

Source

https://github.com/LieberInstitute/HumanPilot/blob/master/Analysis/Layer_Guesses/dlpfc_snRNAseq_annotation.R and https://github.com/LieberInstitute/spatialLIBD/blob/master/dev/02_dev.R#L107-L194.

update_scaleFactors *Update the VisiumExperiment scaleFactors*

Description

Given that our data is a custom [VisiumExperiment-class](#) object with 12 images instead of 1, this function is useful for updating the [scaleFactors](#) for a given image.

Usage

```
update_scaleFactors(ve, sample_id)
```

Arguments

ve A [VisiumExperiment-class](#) object created with `fetch_data()`.
sample_id A character(1) with the name of sample of interest.

Value

A [VisiumExperiment-class](#) object with the updated [scaleFactors](#) for the given input `sample_id`.

Examples

```
if (enough_ram()) {
  ## Obtain the necessary data
  if (!exists("ve")) ve <- fetch_data("ve")

  ## Load the SpatialExperiment package
  library("SpatialExperiment")

  ## Check the default scale factors
  scaleFactors(ve)[1:4]

  ## Replace them for those for the second image and check them
  ve <- update_scaleFactors(ve, "151508")
  scaleFactors(ve)[1:4]

  ## Restore the original ones from the first image
  ve <- update_scaleFactors(ve, "151507")
  scaleFactors(ve)[1:4]
}
```

ve_image_colData *Create a data frame with required columns of the ve object to use
sce_image_gene_p() function*

Description

Function that creates a data frame containing the `colData` columns from the `ve` object and the `array_row` and `array_col` from the `spatialCoords` slot. This allows to use the functions `sce_image_gene_p()` and `sce_image_clus_p()` with `ve` objects.

Usage

```
ve_image_colData(ve, meta = as.data.frame(colData(ve)))
```

Arguments

ve A [VisiumExperiment-class](#) object created, such as one created by `sce_to_ve()`. It is a subset containing data of just one sample.

meta A `data.frame` containing `colData` columns from the `ve` object.

Value

A `data.frame` containing `colData` columns from the `ve` object and the `pxl_col_in_fullres` and `pxl_row_in_fullres` from the `spatialCoords` slot renamed as `imagecol` and `imagerow`.

Author(s)

Brenda Pardo, Leonardo Collado-Torres

See Also

Other `VisiumExperiment`-related functions: [read_image\(\)](#), [sce_to_ve\(\)](#)

Index

- * **Check input functions**
 - check_image_path, 2
 - check_modeling_results, 3
 - check_sce, 4
 - check_sce_layer, 5
- * **Gene set enrichment functions**
 - gene_set_enrichment, 8
 - gene_set_enrichment_plot, 10
- * **Layer correlation functions**
 - layer_stat_cor, 18
 - layer_stat_cor_plot, 19
- * **Layer modeling functions**
 - layer_boxplot, 14
 - sig_genes_extract, 34
 - sig_genes_extract_all, 35
- * **Spatial cluster visualization functions**
 - sce_image_clus, 23
 - sce_image_clus_p, 25
 - sce_image_grid, 29
- * **Spatial gene visualization functions**
 - sce_image_gene, 26
 - sce_image_gene_p, 28
 - sce_image_grid_gene, 31
- * **VisiumExperiment-related functions**
 - read_image, 21
 - sce_to_ve, 33
 - ve_image_colData, 38
- * **datasets**
 - libd_layer_colors, 21
 - tstats_Human_DLPPFC_snRNAseq_Nguyen_topLayer, 37
- BiocFileCache-class, 7, 33
- check_image_path, 2, 4, 5
- check_modeling_results, 3, 3, 5
- check_sce, 3, 4, 4, 5
- check_sce_layer, 3–5, 5
- DataFrame-class, 36
- enough_ram, 6
- ExperimentHub-class, 7
- fetch_data, 7
- fetch_data(), 3–5, 8, 15, 18, 22, 24–26, 28, 30, 32–36
- fields::image.plot(), 17
- gene_set_enrichment, 8, 11
- gene_set_enrichment(), 10
- gene_set_enrichment_plot, 9, 10
- gene_set_enrichment_plot(), 17
- geom_spatial, 12
- geom_spatial(), 24, 25, 27, 28, 30, 32
- get_colors, 13
- ggplot2, 24, 25, 27, 28, 30, 32
- ggplot2::layer(), 12
- ggplotly(), 24, 25, 27, 28, 30, 32
- graphics::par(), 17
- layer_boxplot, 14, 35, 36
- layer_boxplot(), 34, 35
- layer_matrix_plot, 16
- layer_matrix_plot(), 10, 20
- layer_stat_cor, 18, 20
- layer_stat_cor(), 19, 20, 37
- layer_stat_cor_plot, 19, 19
- layer_stat_cor_plot(), 17, 37
- libd_layer_colors, 21
- palette36.colors, 13
- paste0(), 24, 27, 30, 32
- plot_grid, 30, 32
- read_image, 21, 34, 39
- run_app, 22
- scaleFactors, 38
- sce_image_clus, 23, 25, 30
- sce_image_clus(), 25, 26, 30
- sce_image_clus_p, 24, 25, 30
- sce_image_clus_p(), 12, 24, 28
- sce_image_gene, 26, 28, 32
- sce_image_gene(), 23, 28, 32
- sce_image_gene_p, 27, 28, 32
- sce_image_gene_p(), 12, 25, 27
- sce_image_grid, 24, 25, 29
- sce_image_grid(), 31
- sce_image_grid_gene, 27, 28, 31

sce_image_grid_gene(), 29
sce_to_ve, 21, 33, 39
shiny.appobj, 23
sig_genes_extract, 15, 34, 36
sig_genes_extract(), 35
sig_genes_extract_all, 15, 35, 35
sig_genes_extract_all(), 14, 22, 34
SingleCellExperiment, 3, 5, 7, 15, 22,
24–26, 28, 30, 32, 33, 35, 36
SingleCellExperiment-class, 33
sort_clusters, 36
sort_clusters(), 30
stats::fisher.test(), 9

tstats_Human_DLPCF_snRNAseq_Nguyen_topLayer,
37

unname(), 14
update_scaleFactors, 38
utils::download.file(), 7

ve_image_colData, 21, 34, 38
viridis, 27, 28, 32
VisiumExperiment-class, 7, 21, 33, 38, 39