

Package ‘D0tools’

November 4, 2025

Title Convenient functions to streamline your single cell data analysis workflow

Version 1.1.0

Description This package provides functions for creating various visualizations, convenient wrappers, and quality-of-life utilities for single cell experiment objects. It offers a streamlined approach to visualize results and integrates different tools for easy use.

License MIT + file LICENSE

BugReports <https://github.com/MarianoRuzJurado/D0tools/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

biocViews SingleCell, RNASeq, Visualization, Clustering, Annotation, WorkflowStep, QualityControl, GeneExpression

Depends R (>= 4.5.0)

Imports Seurat (>= 5.2.0), SeuratObject (>= 5.1.0), ggplot2 (>= 3.5.0), ggpubr (>= 0.6.0), ggtext (>= 0.1.2), ggalluvial (>= 0.12.5), tidyverse (>= 2.0.0), reshape2 (>= 1.4.4), dplyr (>= 1.1.4), tidyr (>= 1.3.1), rstatix (>= 0.7.2), cowplot (>= 1.1.3), reticulate (>= 1.41.0.1), zellkonverter (>= 1.16.0), progress (>= 1.2.3), ggiraphExtra (>= 0.3.0), grid (>= 4.4.3), SCpubr (>= 2.0.2), DropletUtils (>= 1.26.0), scCustomize (>= 3.0.1), openxlsx (>= 4.2.8), tibble (>= 3.2.1), scDbfFinder (>= 1.20.0), ggcorrplot (>= 0.1.4.1), DESeq2 (>= 1.48.1), enrichR (>= 3.4), cli (>= 3.6.5), curl (>= 6.3.0), magrittr (>= 2.0.3), Matrix (>= 1.7.3), purrr (>= 1.0.4), rlang (>= 1.1.6), scales (>= 1.4.0), SingleCellExperiment (>= 1.30.1), S4Vectors (>= 0.46.0), basilisk (>= 1.20.0), methods, stats, utils

Suggests SummarizedExperiment, knitr, kableExtra, pkgdown, RefManageR, BiocStyle, roxygen2, httr, magick, rmarkdown, assertthat, plyr, rsvg, scan, scater, igraph, sessioninfo, testthat (>= 3.0.0), mockery

VignetteBuilder knitr

URL <https://marianoruzjurado.github.io/DOtools/>

StagedInstall yes

Config/testthat/edition 3

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

git_branch devel

git_last_commit 527ea73

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2025-11-04

Author Mariano Ruz Jurado [aut, cre] (ORCID:

[<https://orcid.org/0000-0001-5354-5336>](https://orcid.org/0000-0001-5354-5336)),

David Rodriguez Morales [aut] (ORCID:

[<https://orcid.org/0000-0002-1819-6991>](https://orcid.org/0000-0002-1819-6991)),

David John [aut] (ORCID: [<https://orcid.org/0000-0003-3217-5449>](https://orcid.org/0000-0003-3217-5449)),

DFG SFB 1366, Project B04 [fnd],

DFG SFB 1531, Project 456687919 [fnd]

Maintainer Mariano Ruz Jurado <ruzjurado@med.uni-frankfurt.de>

Contents

.annoSegment	3
.DO.BarcodeRanks	5
.example_10x	5
.QC_Vlnplot	6
DO.BarplotClustert	6
DO.BarplotWilcox	8
DO.BoxPlot	10
DO.CellBender	12
DO.CellComposition	13
DO.CellTypist	15
DO.Correlation	17
DO.DietSCE	18
DO.Dotplot	19
DO.enrichR	21
DO.FullRecluster	23
DO.Heatmap	24
DO.Import	27
DO.Integration	29
DO.MultiDGE	32
DO.PyEnv	33
DO.scVI	34
DO.SplitBarGSEA	35
DO.Subset	37
DO.TransferLabel	39
DO.UMAP	40

<code>.annoSegment</code>	3
DO.VlnPlot	41
DOtools	43
Index	45

<code>.annoSegment</code>	<i>Annotation modifier for plots</i>
---------------------------	--------------------------------------

Description

Used for segment the plot for further annotations

Usage

```
.annoSegment(
  object = NULL,
  relSideDist = 0.1,
  aesGroup = FALSE,
  aesGroName = NULL,
  annoPos = "top",
  xPosPosition = NULL,
  yPosPosition = NULL,
  pCol = NULL,
  segWidth = 1,
  lty = NULL,
  lwd = 10,
  alpha = NULL,
  lineend = "square",
  annoManual = FALSE,
  mArrow = NULL,
  addBranch = FALSE,
  bArrow = NULL,
  branDirection = 1,
  branRelSegLen = 0.3,
  addText = FALSE,
  textCol = NULL,
  textSize = NULL,
  fontfamily = NULL,
  fontface = NULL,
  textLabel = NULL,
  textRot = 0,
  textHVjust = 0.2,
  hjust = NULL,
  vjust = NULL,
  myFacetGrou = NULL,
  aes_x = NULL,
  aes_y = NULL
)
```

Arguments

<code>object</code>	ggplot list. Default(NULL).
<code>relSideDist</code>	The relative distance ratio to the y axis range. Default(0.1).
<code>aesGroup</code>	Whether use your group column to add rect annotation. Default("FALSE").
<code>aesGroName</code>	The mapping column name. Default(NULL).
<code>annoPos</code>	The position for the annotation to be added. Default("top").
<code>xPosition</code>	The x axis coordinate for the segment. Default(NULL).
<code>yPosition</code>	The y axis coordinate for the segment. Default(NULL).
<code>pCol</code>	The segment colors. Default(NULL).
<code>segWidth</code>	The relative segment width. Default(1).
<code>lty</code>	The segment line type. Default(NULL).
<code>lwd</code>	The segment line width. Default(NULL).
<code>alpha</code>	The segment color alpha. Default(NULL).
<code>lineend</code>	The segment line end. Default("square").
<code>annoManual</code>	Whether annotate by yourself by supplying with x and y coordinates. Default(FALSE).
<code>mArrow</code>	Whether add segment arrow. Default(FALSE).
<code>addBranch</code>	Whether add segment branch. Default(FALSE).
<code>bArrow</code>	Whether add branch arrow. Default(FALSE).
<code>branDirection</code>	The branch direction. Default(1).
<code>branRelSegLen</code>	The branch relative length to the segment. Default(0.3).
<code>addText</code>	Whether add text label on segment. Default(FALSE).
<code>textCol</code>	The text colors. Default(NULL).
<code>textSize</code>	The text size. Default(NULL).
<code>fontfamily</code>	The text fontfamily. Default(NULL).
<code>fontface</code>	The text fontface. Default(NULL).
<code>textLabel</code>	The text textLabel. Default(NULL).
<code>textRot</code>	The text angle. Default(NULL).
<code>textHVjust</code>	The text distance from the segment. Default(0.2).
<code>hjust</code>	The text hjust. Default(NULL).
<code>vjust</code>	The text vjust. Default(NULL).
<code>myFacetGrou</code>	Your facet group name to be added with annotation when object is a faceted object. Default(NULL).
<code>aes_x</code>	= NULL You should supply the plot X mapping name when annotate a facetd plot. Default(NULL).
<code>aes_y</code>	= NULL You should supply the plot Y mapping name when annotate a facetd plot. Default(NULL).

Value

ggplot

Author(s)

Mariano Ruz Jurado (edited from: Jun Zhang)

.DO.BarcodeRanks

DO.BarcodeRanks

Description

Given a raw count matrix (e.g. from a CellRanger HDF5 file), estimate the number of expected cells and droplets using the knee and inflection points from barcodeRanks.

Usage

```
.DO.BarcodeRanks(SCE_obj)
```

Arguments

SCE_obj A Single cell experiment object.

Value

A named numeric vector: 'c(xpc_cells = ..., total_cells = ...)

Author(s)

Mariano Ruz Jurado

.example_10x

Download example dataset 10x

Description

Download example dataset 10x

Usage

```
.example_10x(base_dir = tempfile("dotools_datasets_"))
```

Value

directory path where the data was saved

<code>.QC_Vlnplot</code>	<code>.QC_Vlnplot</code>
--------------------------	--------------------------

Description

Generates violin plots for common quality control (QC) metrics of single-cell RNA-seq data from a Seurat object. The function displays three violin plots for the number of detected genes per cell (nFeature_RNA), total UMI counts per cell (nCount_RNA), and mitochondrial gene content percentage (pt_mito). Useful for visual inspection of QC thresholds and outliers.

Usage

```
.QC_Vlnplot(  
  sce_object,  
  id,  
  layer = "counts",  
  features = c("nFeature_RNA", "nCount_RNA", "pt_mito")  
)
```

Arguments

- sce_object A Seurat object containing single-cell RNA-seq data.
- layer A character string specifying the assay layer to use (default is "counts").
- features A character vector of length 3 indicating the feature names to plot. Default is c("nFeature_RNA", "nCount_RNA", "pt_mito").

Value

A ggplot object arranged in a single row showing violin plots for the specified features with overlaid boxplots.

Author(s)

Mariano Ruz Jurado

DO.BarplotClustert	<i>SEM Graph with t test on cluster level</i>
--------------------	---

Description

Perform SEM-based graphs with t-test on cluster level for SCE objects. Calculates mean expression values and SEM for selected features and visualizes them. Performs pairwise t-tests comparing conditions, with optional custom control condition and clustering. Optionally returns a summary data frame.

Usage

```
DO.BarplotClustert(
  sce_object,
  Feature,
  ListTest = NULL,
  returnValues = FALSE,
  ctrl.condition = NULL,
  group.by = "condition",
  returnPlot = TRUE,
  bar_colours = NULL,
  stat_pos_mod = 1.15,
  step_mod = 0.2,
  x_label_rotation = 45,
  y_limits = NULL,
  log1p_nUMI = TRUE
)
```

Arguments

sce_object	Combined SCE object or Seurat
Feature	gene name
ListTest	List with conditions t-test will be performed, if NULL always against provided CTRL
returnValues	return df.melt.sum data frame containing means and SEM for the set group
ctrl.condition	set your ctrl condition, relevant if running with empty comparison List
group.by	select the seurat object slot where your conditions can be found, default conditon
returnPlot	IF TRUE returns ggplot
bar_colours	colour vector
stat_pos_mod	Defines the distance to the graphs of the statistic
step_mod	Defines the distance between each statistics bracket
x_label_rotation	Rotation of x-labels
y_limits	set limits for y-axis
log1p_nUMI	If nUMIs should be log1p transformed

Value

a ggplot or a dataframe

Author(s)

Mariano Ruz Jurado

Examples

```
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))

set.seed(123)
sce_data$orig.ident <-
  sample(rep(c("A", "B", "C"), length.out = ncol(sce_data)))

ListTest <- list()
ListTest[[1]] <- c("healthy", "disease")

D0.BarplotClustert(
  sce_object = sce_data,
  Feature = "NKG7",
  ListTest = ListTest,
  ctrl.condition = "healthy",
  group.by = "condition"
)
```

D0.BarplotWilcox

SEM Graph with wilcox test on single cell level

Description

Perform SEM-based graphs with Wilcox test on single-cell level for Seurat and SCE objects. Calculates mean expression values and SEM for the selected feature, and visualizes them. Performs pairwise Wilcox tests comparing conditions, with optional custom control condition and clustering. Optionally returns a summary data frame, statistical test results, and the generated plot.

Usage

```
D0.BarplotWilcox(
  sce_object,
  Feature,
  ListTest = NULL,
  returnValues = FALSE,
  ctrl.condition = NULL,
  group.by = "condition",
  wilcox_test = TRUE,
  bar_colours = NULL,
  stat_pos_mod = 1.15,
  step_mod = 0.2,
  x_label_rotation = 45,
  plotPvalue = FALSE,
  y_limits = NULL,
  log1p_nUMI = TRUE
)
```

Arguments

sce_object	combined SCE object or Seurat
Feature	name of the feature/gene
ListTest	List for which conditions wilcoxon test will be performed, if NULL always CTRL group against everything
returnValues	return data frames needed for the plot, containing df.melt, df.melt.sum, df.melt.orig and wilcoxstats
ctrl.condition	set your ctrl condition, relevant if running with empty comparison List
group.by	select the seurat object slot where your conditions can be found, default conditon
wilcox_test	perform wilcox test. boolean default TRUE
bar_colours	colour vector
stat_pos_mod	Defines the distance to the graphs of the statistic
step_mod	Defines the distance between each statistics bracket
x_label_rotation	Rotation of x-labels
plotPvalue	plot the non adjusted p-value without correcting for multiple tests
y_limits	set limits for y-axis
log1p_nUMI	If nUMIs should be log1p transformed

Value

a ggplot or a list with plot and data frame

Author(s)

Mariano Ruz Jurado

Examples

```
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))

ListTest <- list()
ListTest[[1]] <- c("healthy", "disease")

D0.BarplotWilcox(
  sce_object = sce_data,
  Feature = "NKG7",
  ListTest = ListTest,
  ctrl.condition = "healthy",
  group.by = "condition"
)
```

DO.BoxPlot

*Box Graph with wilcox test on single cell level***Description**

Creates a box plot using a pseudo-bulk approach and performs a Wilcoxon test on single-cell level. Allows customization of outlier removal, statistical labels, and color schemes. Supports comparison of conditions with optional second grouping. Useful for visualizing gene expression and statistical differences.

Usage

```
DO.BoxPlot(
  sce_object,
  Feature,
  sample.column = "orig.ident",
  ListTest = NULL,
  group.by = "condition",
  group.by.2 = NULL,
  ctrl.condition = NULL,
  outlier_removal = TRUE,
  plot_sample = TRUE,
  vector_colors = c("#1f77b4", "#ea7e1eff", "royalblue4", "tomato2", "darkgoldenrod",
    "palegreen4", "maroon", "thistle3"),
  wilcox_test = TRUE,
  stat_pos_mod = 1.15,
  step_mod = 0,
  hjust.wilcox = 0.5,
  vjust.wilcox = 0.25,
  size.wilcox = 3.33,
  hjust.wilcox.2 = 0.5,
  vjust.wilcox.2 = 0,
  sign_bar = 0.8,
  orderAxis = NULL
)
```

Arguments

sce_object	The SCE object or Seurat
Feature	name of the feature/gene
sample.column	meta data column containing sample IDs
ListTest	List for which conditions wilcox will be performed, if NULL always CTRL group against everything
group.by	group name to look for in meta data
group.by.2	second group name to look for in meta data

<code>ctrl.condition</code>	select condition to compare to
<code>outlier_removal</code>	Outlier calculation
<code>plot_sample</code>	Plot individual sample dot in graph
<code>vector_colors</code>	get the colours for the plot
<code>wilcox_test</code>	If you want to have wilcoxon performed, boolean default TRUE
<code>stat_pos_mod</code>	modifier for where the p-value is plotted increase for higher
<code>step_mod</code>	value for defining the space between one test and the next one
<code>hjust.wilcox</code>	value for adjusting height of the text
<code>vjust.wilcox</code>	value for vertical of text
<code>size.wilcox</code>	value for size of text of statistical test
<code>hjust.wilcox.2</code>	value for adjusting height of the text, with group.by.2 specified
<code>vjust.wilcox.2</code>	value for vertical of text, with group.by.2 specified
<code>sign_bar</code>	adjusts the sign_bar with group.by.2 specified
<code>orderAxis</code>	vector for xaxis sorting, alphabetically by default

Value

a ggplot

Author(s)

Mariano Ruz Jurado

Examples

```
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))

set.seed(123)
sce_data$orig.ident <-
  sample(rep(c("A", "B", "C"), length.out = ncol(sce_data)))

ListTest <- list()
ListTest[[1]] <- c("healthy", "disease")

D0.BoxPlot(
  sce_object = sce_data,
  Feature = "NKG7",
  sample.column = "orig.ident",
  ListTest = ListTest,
  group.by = "condition",
  ctrl.condition = "healthy",
)
```

DO.CellBender

*DO.CellBender***Description**

This function wraps a system call to a bash script for running CellBender on CellRanger outputs. It ensures required inputs are available and optionally installs CellBender in a conda env.

Usage

```
DO.CellBender(
    cellranger_path,
    output_path,
    samplenames = NULL,
    cuda = TRUE,
    cpu_threads = 15,
    epochs = 150,
    lr = 1e-05,
    estimator_multiple_cpu = FALSE,
    log = TRUE,
    conda_path = NULL,
    BarcodeRanking = TRUE,
    bash_script = system.file("bash", "_run_CellBender.sh", package = "D0tools")
)
```

Arguments

cellranger_path	Path to folder with CellRanger outputs.
output_path	Output directory for CellBender results.
samplenames	Optional vector of sample names. If NULL, will autodetect folders in cellranger_path.
cuda	Logical, whether to use GPU (CUDA).
cpu_threads	Number of CPU threads to use.
epochs	Number of training epochs.
lr	Learning rate.
estimator_multiple_cpu	Use estimator with multiple CPU threads.
log	Whether to enable logging.
conda_path	Optional path to the conda environment.
BarcodeRanking	Optional Calculation of estimated cells in samples through DropletUtils implementation
bash_script	Path to the bash script that runs CellBender.

Value

None

Examples

```
## Not run:
# Define paths
cellranger_path <- "/mnt/data/cellranger_outputs"
output_path <- "/mnt/data/cellbender_outputs"

# Optional: specify sample names if automatic detection is not desired
samplernames <- c("Sample_1", "Sample_2")

# Run CellBender (uses GPU by default)
DO.CellBender(
  cellranger_path = cellranger_path,
  output_path = output_path,
  samplernames = samplernames,
  cuda = TRUE,
  cpu_threads = 8,
  epochs = 100,
  lr = 0.00001,
  estimator_multiple_cpu = FALSE,
  log = TRUE
)

## End(Not run)
```

DO.CellComposition *DO CellComposition*

Description

Computes and visualizes cell composition changes from a Seurat object using Scanpro. Integrates R and Python via reticulate to convert Seurat to AnnData and run Scanpro. Generates customizable plots with options for transformation, grouping, and bootstrapping. Returns a ggplot object or a list containing plot data and visualisation.

Usage

```
DO.CellComposition(
  sce_object,
  assay_normalized = "RNA",
  cluster_column = "seurat_clusters",
  sample_column = "orig.ident",
  condition_column = "condition",
  transform_method = "logit",
  sort_x = NULL,
```

```

sub_ident = NULL,
sort_fill = NULL,
scanpro_plots = FALSE,
scanpro_group = NULL,
outputFolder = NULL,
return_df = FALSE,
bar_colors = NULL,
n_reps = NULL,
legend.pos.x = 0.48,
legend.pos.y = 0,
cowplot_width = 0.9,
cowlegend_width = 0.9,
...
)

```

Arguments

sce_object	The SCE object or Seurat
assay_normalized	Assay with raw counts
cluster_column	Column in meta data which will be used to segment the bar plot
sample_column	Column in meta data containing individual sample names
condition_column	Column in meta data plotted on the xaxis
transform_method	Method of transformation of properties, default: "logit"
sort_x	Vector sorting the xaxis
sub_ident	vector to subset the whole plot by
sort_fill	Vector sorting the bar segments
scanpro_plots	Boolean, will create plots provided in scanpro package, default: FALSE
scanpro_group	Defines the clusters showed in scanpro plots
outputFolder	Scanpro plots will be saved in this directory, defaults to current working directory
return_df	Boolean, makes a list with first entry being the dataframe used to create the ggplot and second entry is the plot
bar_colors	(named) vector of colors used for plotting, the names will be used to factorize the segments
n_reps	Number of replicates generated by scanpro
legend.pos.x	adjusts the position of the legend in horizontal
legend.pos.y	adjusts the position of the legend in vertical
cowplot_width	Changes the width on the plotting device for plot
cowlegend_width	Changes the width on the plotting device for legend, useful for adjusting the legend position in combination with legend.pos.x and y
...	Further arguments passed to scanpro plotting functions

Value

ggplot object or list

Author(s)

Mariano Ruz Jurado

Examples

```
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))

DO.CellComposition(
  sce_object = sce_data,
  cluster_column = "annotation",
  condition_column = "condition",
  scanpro_plots = FALSE,
  n_reps = 5
)
```

DO.CellTypist

DO Celltypist

Description

Runs the CellTypist model on a Seurat or SCE object to predict cell type labels, storing the results as metadata. If the number of cells is less than the specified threshold, it returns NAs for the labels. Optionally updates the CellTypist models and returns the probability matrix. Useful for annotating cell types in single-cell RNA sequencing datasets.

Usage

```
DO.CellTypist(
  sce_object,
  modelName = "Healthy_Adult_Heart.pkl",
  minCellsToRun = 200,
  runCelltypistUpdate = TRUE,
  over_clustering = "seurat_clusters",
  assay_normalized = "RNA",
  returnProb = FALSE,
  SeuV5 = TRUE
)
```

Arguments

sce_object	The seurat or sce object
modelName	Specify the model you want to use for celltypist
minCellsToRun	If the input seurat or SCE object has fewer than this many cells, NAs will be added for all expected columns and celltypist will not be run.
runCelltypistUpdate	If true, -update-models will be run for celltypist prior to scoring cells.
over_clustering	Column in metadata in object with clustering assignments for cells, default seurat_clusters
assay_normalized	Assay with log1p normalized expressions
returnProb	will additionally return the probability matrix, return will give a list with the first element being the object and second prob matrix
SeuV5	Specify if the Seurat object is made with Seuratv5

Value

a seurat or sce object

Author(s)

Mariano Ruz Jurado

Examples

```
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))

sce_data <- DO.CellTypist(
  sce_object = sce_data,
  modelName = "Healthy_Adult_Heart.pkl",
  runCelltypistUpdate = TRUE,
  over_clustering = "seurat_clusters",
  minCellsToRun = 5,
  SeuV5 = TRUE
)
```

DO.Correlation

*DO Correlation Plot for visualizing similarity between categories***Description**

Generates a correlation heatmap from expression data to visualize similarity across sample groups. Allows customization of plot type, correlation method, and color scaling using the ggcorrplot2 and ggplot2 architectures. Ideal for comparing transcriptional profiles between conditions or clusters.

Usage

```
DO.Correlation(
  sce_object,
  group_by = "orig.ident",
  assay = "RNA",
  features = NULL,
  method = "spearman",
  plotdesign = "square",
  plottype = "full",
  auto_limits = TRUE,
  outline.color = "white",
  colormap = c("royalblue4", "lightsteelblue", "tomato", "firebrick4"),
  lab_size = 10,
  lab = TRUE,
  lab_col = "white",
  axis_size_x = 12,
  axis_size_y = 12,
  ...
)
```

Arguments

sce_object	Seurat or SCE Object
group_by	Column to aggregate the expression over it, default "orig.ident"
assay	Assay in object to use, default "RNA"
features	What genes to include by default all, default "None"
method	Correlation method, default "spearman"
plotdesign	Plot design, default "circle"
plottype	Show the full plot or only half of it, default "full"
auto_limits	Automatically rescales the colour bar based on the values in the correlation matrix, default "TRUE"
outline.color	the outline color of square or circle. Default value is "white".
colormap	Defines the colormap used in the plot, default c("royalblue4", "royalblue2", "firebrick", "firebrick4")
lab_size	Size to be used for the correlation coefficient labels. used when lab = TRUE.

lab	logical value. If TRUE, add correlation coefficient on the plot.
lab_col	color to be used for the correlation coefficient labels. used when lab = TRUE.
axis_size_x	Controls x labels size
axis_size_y	Controls y labels size
...	Additionally arguments passed to ggcorrplot function

Value

ggplot2

Author(s)

Mariano Ruz Jurado

Examples

```
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))

D0.Correlation(
  sce_object = sce_data,
  group_by = "orig.ident",
  assay = "RNA",
  features = NULL,
  method = "spearman",
  plotdesign = "square",
  plotype = "full",
  auto_limits = TRUE,
  outline.color = "white",
  colormap = c("royalblue4", "lightsteelblue", "tomato", "firebrick4"),
  lab_size = 10,
  lab = TRUE,
  lab_col = "white"
)
```

DO.DietSCE

Remove Layers from Seurat or SCE Object by Pattern

Description

This function removes layers from a Seurat or SCE object's RNA assay based on a specified regular expression pattern. It is supposed to remove no longer needed layers from the object.

Usage

```
D0.DietSCE(sce_object, assay = "RNA", pattern = "^scale\\.data\\.")
```

Arguments

sce_object	Seurat or SCE object.
assay	Name of the assay from where to remove layers from
pattern	regular expression pattern to match layer names. Default "^scale\\.data\\."

Value

Seurat or SCE object with specified layers removed.

Author(s)

Mariano Ruz Jurado

Examples

```
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))

sce_data <- D0.DietSCE(sce_data, pattern = "data")
```

DO.Dotplot

DO Dot plot

Description

This function generates a dot plot for multiple genes, comparing expression levels across one or two specified groups. It supports both individual and pseudobulk expression calculations. Highly variable customization options allow control over dot size, color scaling, annotations, and axis orientation. The function integrates seamlessly with SCE objects for single-cell RNA-seq analysis.

Usage

```
D0.Dotplot(
  sce_object,
  Feature,
  group.by.x = NULL,
  group.by.y = NULL,
  group.by.y2 = NULL,
  across.group.by.x = FALSE,
  across.group.by.y = FALSE,
  sort_x = NULL,
  dot.size = c(1, 6),
  plot.margin = c(1, 1, 1, 1),
  midpoint = 0.5,
  scale_gene = FALSE,
  returnValue = FALSE,
```

```

log1p_nUMI = TRUE,
hide_zero = TRUE,
annotation_x = FALSE,
annotation_x_position = 0.25,
annotation_x_rev = FALSE,
point_stroke = 0.2,
limits_colorscale = NULL,
coord_flip = FALSE,
stats_x = FALSE,
stats_y = TRUE,
sig_size = 6,
nudge_x = 0.3,
nudge_y = 0.2,
...
)

```

Arguments

sce_object	The SCE object or Seurat
Feature	Genes or DF of interest, Data frame should have columns with gene and annotation information, e.g. output of FindAllMarkers
group.by.x	group name to plot on x-axis
group.by.y	group name to look for in meta data
group.by.y2	second group name to look for in meta data
across.group.by.x	calculate a pseudobulk expression approach for the x-axis categories
across.group.by.y	calculate a pseudobulk expression approach for the y-axis categories
sort_x	Vector sorting the xaxis
dot.size	Vector of dot size
plot.margin	= plot margins
midpoint	midpoint in color gradient
scale_gene	If True calculates the Z-score of the average expression per gene
returnValue	return the dataframe behind the plot
log1p_nUMI	log1p the plotted values, boolean
hide_zero	Removes dots for genes with 0 expression
annotation_x	Adds annotation on top of x axis instead on y axis
annotation_x_position	specifies the position for the annotation
annotation_x_rev	reverses the annotations label order
point_stroke	Defines the thickness of the black stroke on the dots
limits_colorscale	Set manually colorscale limits

<code>coord_flip</code>	flips the coordinates of the plot with each other
<code>stats_x</code>	Perform statistical test over categories on the xaxis
<code>stats_y</code>	Perform statistical test over categories on the yaxis
<code>sig_size</code>	Control the size of the significance stars in the plot
<code>nudge_x</code>	Control the position of the star on x axis
<code>nudge_y</code>	Control the position of the star on y axis
<code>...</code>	Further arguments passed to <code>annoSegment</code> function if <code>annotation_x == TRUE</code>

Value

a `ggplot`

Author(s)

Mariano Ruz Jurado

Examples

```
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))

D0.Dotplot(
  sce_object = sce_data,
  Feature = c("NKG7", "IL6", "MALAT1"),
  group.by.x = "condition"
)
```

DO.enrichR

DO.enrichR

Description

Performs Gene Ontology enrichment analysis on differentially expressed genes using the EnrichR API. Separately analyzes upregulated and downregulated genes and returns results.

Usage

```
DO.enrichR(
  df_DGE,
  gene_column,
  pval_column,
  log2fc_column,
  pval_cutoff = 0.05,
  log2fc_cutoff = 0.25,
  path = NULL,
  filename = "",
```

```

species = "Human",
go_catgs = c("GO_Molecular_Function_2023", "GO_Cellular_Component_2023",
             "GO_Biological_Process_2023")
)

```

Arguments

df_DGE	data.frame containing differential gene expression results.
gene_column	column name in df with gene symbols.
pval_column	column name in df with p-values.
log2fc_column	column name in df with log2 fold changes.
pval_cutoff	adjusted p-value threshold for significance (default = 0.05).
log2fc_cutoff	log2 fold change threshold for up/down regulation (default = 0.25).
path	folder path where the output Excel file will be saved. A subfolder "GSA_Tables" will be created.
filename	suffix used in the Excel filename (e.g., "GSA_CellType_MyAnalysis.xlsx").
species	species name for enrichment analysis. Options include "Human", "Mouse", "Yeast", etc. (default = "Mouse").
go_catgs	GO databases to use. Defaults to c(GO_Biological_Process_2023).

Value

data.frame with GO enrichment results if path is NULL, otherwise writes an Excel file.

Examples

```

library(enrichR)

sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))
DGE_result <- DO.MultiDGE(sce_data,
  sample_col = "orig.ident",
  method_sc = "wilcox",
  annotation_col = "annotation",
  ident_ctrl = "healthy"
)

DGE_result <- DGE_result[DGE_result$celltype == "CD4_T_cells", ]

result_GO <- DO.enrichR(
  df_DGE = DGE_result,
  gene_column = "gene",
  pval_column = "p_val_SC_wilcox",
  log2fc_column = "avg_log2FC_SC_wilcox",
  pval_cutoff = 0.05,
  log2fc_cutoff = 0.25,
  path = NULL,
  filename = "",

```

```

    species = "Human",
    go_catgs = "GO_Biological_Process_2023"
  )

```

DO.FullRecluster

DO.FullRecluster

Description

Performs iterative reclustering on each major cluster found by FindClusters in a Seurat or SCE object. It refines the clusters using the FindSubCluster function for better resolution and fine-tuned annotation. The new clustering results are stored in a metadata column called `annotation_recluster`. Suitable for improving cluster precision and granularity after initial clustering.

Usage

```

DO.FullRecluster(
  sce_object,
  over_clustering = "seurat_clusters",
  res = 0.5,
  algorithm = 4,
  graph.name = "RNA_snn"
)

```

Arguments

<code>sce_object</code>	The seurat or SCE object
<code>over_clustering</code>	Column in metadata in object with clustering assignments for cells, default <code>seurat_clusters</code>
<code>res</code>	Resolution for the new clusters, default 0.5
<code>algorithm</code>	Set one of the available algorithms found in FindSubCluster function, default = 4: <code>leiden</code>
<code>graph.name</code>	A builded neirest neighbor graph

Value

a Seurat or SCE Object with new clustering named `annotation_recluster`

Author(s)

Mariano Ruz Jurado

Examples

```
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))

sce_data <- D0.FullRecluster(
  sce_object = sce_data
)
```

DO.Heatmap

DO Heatmap of the mean expression of genes across a groups

Description

Wrapper around heatmap.py, which generates a heatmap of showing the average nUMI for a set of genes in different groups. Additional an argument can be made to show foldchanges between two conditions. Differential gene expression analysis between the different groups can be performed.

Usage

```
D0.Heatmap(
  sce_object,
  features,
  assay_normalized = "RNA",
  group_by = "seurat_clusters",
  groups_order = NULL,
  value_plot = "expr",
  group_fc = "condition",
  group_fc_ident_1 = NULL,
  group_fc_ident_2 = NULL,
  clip_value = FALSE,
  max_fc = 5,
  z_score = NULL,
  path = NULL,
  filename = "Heatmap.svg",
  swap_axes = TRUE,
  cmap = "Reds",
  title = NULL,
  title_fontprop = NULL,
  clustering_method = "complete",
  clustering_metric = "euclidean",
  cluster_x_axis = FALSE,
  cluster_y_axis = FALSE,
  axs = NULL,
  figsize = c(5, 6),
  linewidth = 0.1,
  ticks_fontdict = NULL,
```

```

xticks_rotation = NULL,
yticks_rotation = NULL,
vmin = 0,
vcenter = NULL,
vmax = NULL,
legend_title = "LogMean(nUMI)\nin group",
add_stats = TRUE,
df_pvals = NULL,
stats_x_size = NULL,
square_x_size = NULL,
test = "wilcox",
pval_cutoff = 0.05,
log2fc_cutoff = 0,
only_pos = TRUE,
square = TRUE,
showP = TRUE,
logcounts = TRUE
)

```

Arguments

sce_object	SCE object or Seurat with meta.data
features	gene names or continuous value in meta data
assay_normalized	Assay with raw counts
group_by	meta data column name with categorical values
groups_order	order for the categories in the group_by
value_plot	plotted values correspond to expression values or foldchanges
group_fc	if foldchanges specified than the groups must be specified that will be compared
group_fc_ident_1	Defines the first group in the test
group_fc_ident_2	Defines the second group in the test
clip_value	Clips the colourscale to the 99th percentile, useful if one gene is driving the colourscale
max_fc	Clips super high foldchanges to this value, so changes can still be appreciated
z_score	apply z-score transformation, "group" or "var"
path	path to save the plot
filename	name of the file
swap_axes	whether to swap the axes or not
cmap	color map
title	title for the main plot
title_fontprop	font properties for the title (e.g., 'weight' and 'size')

<code>clustering_method</code>	clustering method to use when hierarchically clustering the x and y-axis
<code>clustering_metric</code>	metric to use when hierarchically clustering the x- and y-axis
<code>cluster_x_axis</code>	hierarchically clustering the x-axis
<code>cluster_y_axis</code>	hierarchically clustering the y-axis
<code>axs</code>	matplotlib axis
<code>figsize</code>	figure size
<code>linewidth</code>	line width for the border of cells
<code>ticks_fontdict</code>	font properties for the x and y ticks (e.g., 'weight' and 'size')
<code>xticks_rotation</code>	rotation of the x-ticks
<code>yticks_rotation</code>	rotations of the y-ticks
<code>vmin</code>	minimum value
<code>vcenter</code>	center value
<code>vmax</code>	maximum value
<code>legend_title</code>	title for the color bar
<code>add_stats</code>	add statistical annotation, will add a square with an '*' in the center if the expression is significantly different in a group with respect to the others
<code>df_pvals</code>	dataframe with the p-values, should be gene x group or group x gene in case of <code>swap_axes</code> is False
<code>stats_x_size</code>	size of the asterisk
<code>square_x_size</code>	size and thickness of the square percentual, vector
<code>test</code>	test to use for test for significance
<code>pval_cutoff</code>	cutoff for the p-value
<code>log2fc_cutoff</code>	minimum cutoff for the log2FC
<code>only_pos</code>	if set to TRUE, only use positive genes in the condition
<code>square</code>	whether to make the cell square or not
<code>showP</code>	if set to false return a dictionary with the axis
<code>logcounts</code>	whether the input is logcounts or not

Value

Depending on `showP`, returns the plot if set to TRUE or a dictionary with the axes.

Author(s)

Mariano Ruz Jurado & David Rodriguez Morales

Examples

```

#'
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))

D0.Heatmap(
  sce_object = sce_data,
  assay_normalized = "RNA",
  group_by="seurat_clusters",
  features = rownames(sce_data)[1:10],
  z_score = NULL,
  path = NULL,
  filename = "Heatmap.svg",
  swap_axes = TRUE,
  cmap = "Reds",
  title = NULL,
  title_fontprop = NULL,
  clustering_method = "complete",
  clustering_metric = "euclidean",
  cluster_x_axis = FALSE,
  cluster_y_axis = FALSE,
  axs = NULL,
  figsize = c(5, 6),
  linewidth = 0.1,
  ticks_fontdict = NULL,
  xticks_rotation = 45,
  yticks_rotation = NULL,
  vmin = 0.0,
  vcenter = NULL,
  vmax = NULL,
  legend_title = "LogMean(nUMI)\nin group",
  add_stats = TRUE,
  df_pvals = NULL,
  stats_x_size = NULL,
  square_x_size = NULL,
  test = "wilcox",
  pval_cutoff = 0.05,
  log2fc_cutoff = 0,
  only_pos = TRUE,
  square = TRUE,
  showP = FALSE,
  logcounts = TRUE
)

```

Description

Imports and processes single-cell RNA-seq data from various formats (10x Genomics, CellBender, or CSV), performs quality control (QC), filtering, normalization, variable gene selection, and optionally detects doublets. Returns a merged and processed Seurat or SCE object ready for downstream analysis.

Usage

```
DO.Import(
  pathways,
  ids,
  minCellGenes = 5,
  FilterCells = TRUE,
  cut_mt = 0.05,
  min_counts = NULL,
  max_counts = NULL,
  min_genes = NULL,
  max_genes = NULL,
  low_quantile = NULL,
  high_quantile = NULL,
  DeleteDoublets = TRUE,
  include_rbs = TRUE,
  Seurat = TRUE,
  ...
)
```

Arguments

<code>pathways</code>	A character vector of paths to directories or files containing raw expression matrices.
<code>ids</code>	A character vector of sample identifiers, matching the order of pathways.
<code>minCellGenes</code>	Integer. Minimum number of cells a gene must be expressed in to be retained. Default is 5.
<code>FilterCells</code>	Logical. If TRUE, applies QC filtering on cells based on mitochondrial content, counts, and feature thresholds. Default is TRUE.
<code>cut_mt</code>	Numeric. Maximum allowed mitochondrial gene proportion per cell. Default is 0.05.
<code>min_counts</code>	Numeric. Minimum UMI count threshold (optional, used only if <code>low_quantile</code> is NULL).
<code>max_counts</code>	Numeric. Maximum UMI count threshold (optional, used only if <code>high_quantile</code> is NULL).
<code>min_genes</code>	Numeric. Minimum number of genes detected per cell to retain. Optional.
<code>max_genes</code>	Numeric. Maximum number of genes detected per cell to retain. Optional.
<code>low_quantile</code>	Numeric. Quantile threshold (0 to 1) to filter low UMI cells (used if <code>min_counts</code> is NULL).

high_quantile	Numeric. Quantile threshold (0 to 1) to filter high UMI cells (used if max_counts is NULL).
DeleteDoublets	Logical. If TRUE, doublets are detected and removed using scDbtFinder. Default is TRUE.
include_rbs	Logical. If TRUE, calculates ribosomal gene content in addition to mitochondrial content. Default is TRUE.
Seurat	Logical. If TRUE, returns Seurat object otherwise SCE object.
...	Additional arguments passed to RunPCA().

Value

A merged Seurat or SCE object containing all samples, with normalization, QC, scaling, PCA, and optional doublet removal applied.

Author(s)

Mariano Ruz Jurado & David John

Examples

```
## Not run:
merged_obj <- DO.Import(
  pathways = c("path/to/sample1", "path/to/sample2"),
  ids = c("sample1", "sample2"),
  TenX = TRUE,
  CellBender = FALSE,
  minCellGenes = 5,
  FilterCells = TRUE,
  cut_mt = 0.05,
  min_counts = 1000,
  max_counts = 20000,
  min_genes = 200,
  max_genes = 6000,
  DeleteDoublets = TRUE
)

## End(Not run)
```

DO.Integration

DO.Integration SCE object integration directly

Description

Integrates single-cell RNA-seq data directly from SingleCellExperiment or Seurat objects. Supports detection of variable genes, scaling, PCA, neighbor graph construction, clustering, and UMAP embedding, with multiple integration methods.

Usage

```

DO.Integration(
  sce_object,
  split_key = "orig.ident",
  HVG = FALSE,
  scale = FALSE,
  pca = FALSE,
  neighbors = TRUE,
  neighbors_dim = seq_len(50),
  clusters = TRUE,
  clusters_res = 0.3,
  clusters_algorithm = 4,
  umap = TRUE,
  umap_key = "UMAP",
  umap_dim = seq_len(50),
  integration_method = "CCAIntegration",
  selection_method = "vst",
  loess_span = 0.3,
  clip_max = "auto",
  num_bin = 20,
  binning_method = "equal_width",
  scale_max = 10,
  pca_key = "PCA",
  integration_key = "INTEGRATED.CCA",
  npcs = 50,
  verbose = FALSE
)

```

Arguments

sce_object	Seurat or SCE Object
split_key	Character. Column in meta data to split the samples by, default orig.ident
HVG	Logical. Perform detection of highly variable genes
scale	Logical. Perform scaling of the expression data
pca	Logical. Perform principal component analysis
neighbors	Logical. Perform Nearest-neighbor graph after integration
neighbors_dim	Numeric range. Dimensions of reduction to use as input
clusters	Logical. Perform clustering of cells
clusters_res	Numeric. Value of the resolution parameter, use a value above (below) 1.0 if you want to obtain a larger (smaller) number of communities.
clusters_algorithm	Numeric. Define the algorithm for clustering, default 4 for "Leiden"
umap	Logical. Runs the Uniform Manifold Approximation and Projection
umap_key	Character name for
umap_dim	Numeric range. Which dimensions to use as input features

integration_method	Character. Define the integration method, please check what versions are supported in Seurat::IntegrateLayers function
selection_method	Character. Default "vst". Options: "mean.var.plot", "dispersion"
loess_span	Numeric. Loess span parameter used when fitting the variance-mean relationship
clip_max	Character. After standardization values larger than clip.max will be set to clip.max; default is 'auto' which sets this value to the square root of the number of cells
num_bin	Numeric. Total number of bins to use in the scaled analysis (default is 20)
binning_method	Character. "equal_width": each bin is of equal width along the x-axis (default). Options: "equal_frequency":
scale_max	Numeric. Max value to return for scaled data. The default is 10.
pca_key	Character. Key name to save the pca result in
integration_key	Character. Key name to save the integration result in
npcs	Numeric. Total Number of PCs to compute and store (50 by default)
verbose	Logical. Verbosity for all functions

Value

integrated sce/seurat object

Author(s)

Mariano Ruz Jurado

Examples

```
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))

DO.Integration(
  sce_object = sce_data,
  split_key = "orig.ident",
  HVG = TRUE,
  scale = TRUE,
  pca = TRUE,
  integration_method = "CCAIntegration"
)
```

DO.MultiDGE

*DO.MultiDGE***Description**

Performs differential gene expression analysis using both single-cell and pseudo-bulk approaches across all annotated cell types. The single-cell method uses Seurat's FindMarkers, while pseudo-bulk testing uses DESeq2 on aggregated expression profiles. Outputs a merged data frame with DGE statistics from both methods per condition and cell type.

Usage

```
DO.MultiDGE(
  sce_object,
  assay = "RNA",
  method_sc = "wilcox",
  group_by = "condition",
  annotation_col = "annotation",
  sample_col = "orig.ident",
  ident_ctrl = "ctrl",
  min_pct = 0,
  logfc_threshold = 0,
  only_pos = FALSE,
  min_cells_group = 3,
  ...
)
```

Arguments

sce_object	The seurat or SCE object
assay	Specified assay in Seurat or SCE object, default "RNA"
method_sc	method to use for single cell DEG analysis, see FindMarkers from Seurat for options, default "wilcox"
group_by	Column in meta data containing groups used for testing, default "condition"
annotation_col	Column in meta data containing information of cell type annotation
sample_col	Column in meta data containing information of sample annotation, default "orig.ident"
ident_ctrl	Name of the condition in group_by to test against as ctrl, default "ctrl"
min_pct	only test genes that are detected in a minimum fraction of min.pct cells in either of the two populations, default is 0
logfc_threshold	Limit testing to genes which show, on average, at least X-fold difference (log-scale) between the two groups of cells, default is 0.
only_pos	Only return positive markers, default FALSE
min_cells_group	Minimum number of cells in one of the groups, default 3
...	Additional arguments passed to FindMarkers function

Value

Dataframe containing statistics for each gene from the single cell and the Pseudobulk DGE approach.

Author(s)

Mariano Ruz Jurado

Examples

```
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))
DGE_result <- DO.MultiDGE(sce_data,
  sample_col = "orig.ident",
  method_sc = "wilcox",
  annotation_col = "annotation",
  ident_ctrl = "healthy"
)
```

DO.PyEnv

DO.PyEnv

Description

Sets up or connects to a conda Python environment for use with D0tools. If no environment path is provided, it will create one at ~/.venv/D0tools and install required Python packages: scvi-tools, celltypist, and scanpro.

Usage

```
DO.PyEnv(conda_path = NULL)
```

Arguments

conda_path character string specifying the path to an existing or new conda environment.

Value

None

Examples

```
# Creates D0tools environment at ~/.venv/D0tools if it doesn't exist
DO.PyEnv()

# Use an existing conda environment at a custom location
# DO.PyEnv(conda_path = "~/miniconda3/envs/my_d0tools_env")
```

DO.scVI

*DO.scVI***Description**

This function will run the scVI Integration from the scVI python package. It includes all parameters from the actual python package and runs it by using an internal python script. The usage of a gpu is incorporated and highly recommended.

Usage

```
DO.scVI(
    sce_object,
    batch_key,
    layer_counts = "counts",
    layer_logcounts = "logcounts",
    categorical_covariates = NULL,
    continuous_covariates = NULL,
    n_hidden = 128,
    n_latent = 30,
    n_layers = 3,
    dispersion = "gene-batch",
    gene_likelihood = "zinb",
    get_model = FALSE
)
```

Arguments

sce_object	Seurat or SCE object with annotation in meta.data
batch_key	meta data column with batch information.
layer_counts	layer with counts. Raw counts are required.
layer_logcounts	layer with log-counts. Log-counts required for calculation of HVG.
categorical_covariates	meta data column names with categorical covariates for scVI inference.
continuous_covariates	meta data column names with continuous covariates for scVI inference.
n_hidden	number of hidden layers.
n_latent	dimensions of the latent space.
n_layers	number of layers.
dispersion	dispersion mode for scVI.
gene_likelihood	gene likelihood.
get_model	return the trained model.

Value

Seurat or SCE Object with dimensionality reduction from scVI

Examples

```
## Not run:
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))

# Run scVI using the 'orig.ident' column as the batch key
sce_data <- D0.scVI(sce_data, batch_key = "orig.ident")

## End(Not run)
```

DO.SplitBarGSEA

DO Bar plot for GSEA df result

Description

This function generates a split barplot. This is a plot where the top 10 GO terms are shown, sorted based on a column ('col_split'). Two conditions are shown at the same time. One condition is shown in the positive axis, while the other in the negative one. The condition to be shown as positive is set with 'pos_col'.

The GO terms will be shown inside the bars, if the term is too long, using 'cutoff', you can control the maximum number of characters per line.

Pre-filter of the dataframe to contain significant Terms is recommended

Usage

```
D0.SplitBarGSEA(
  df_GSEA,
  term_col,
  col_split,
  cond_col,
  pos_cond,
  cutoff = 40,
  log10_transform = TRUE,
  figsize = c(12, 8),
  topN = 10,
  colors_pairs = c("sandybrown", "royalblue"),
  alpha_colors = 0.3,
  path = NULL,
  spacing = 5,
  txt_size = 12,
  filename = "SplitBar.svg",
  title = "Top 10 GO Terms in each Condition: ",
```

```

    showP = FALSE,
    celltype = "all"
  )

```

Arguments

<code>df_GSEA</code>	dataframe with the results of a gene set enrichment analysis
<code>term_col</code>	column in the dataframe that contains the terms
<code>col_split</code>	column in the dataframe that will be used to sort and split the plot
<code>cond_col</code>	column in the dataframe that contains the condition information
<code>pos_cond</code>	condition that will be shown in the positive side of the plot
<code>cutoff</code>	maximum number of characters per line
<code>log10_transform</code>	if <code>col_split</code> contains values between 0 and 1, assume they are pvals and apply a -log10 transformation
<code>figsize</code>	figure size
<code>topN</code>	how many terms are shown
<code>colors_pairs</code>	colors for each condition (1st color → negative axis; 2nd color → positive axis)
<code>alpha_colors</code>	alpha value for the colors of the bars
<code>path</code>	path to save the plot
<code>spacing</code>	space to add between bars and origin. It is a percentage value , indicating that the bars start at 5 % of the maximum X axis value.
<code>txt_size</code>	size of the go terms text
<code>filename</code>	filename for the plot
<code>title</code>	title of the plot
<code>showP</code>	if False, the axis is return
<code>celltype</code>	vector with cell types you want to subset for, use "all" for all celltypes contained in the dataframe column "celltype"

Value

: None or the axis

Author(s)

Mariano Ruz Jurado

Examples

```

library(enrichR)

sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))
DGE_result <- DO.MultiDGE(sce_data,
  sample_col = "orig.ident",

```

```

        method_sc = "wilcox",
        annotation_col = "annotation",
        ident_ctrl = "healthy"
    )
    DGE_result <- DGE_result[DGE_result$celltype == "CD4_T_cells", ]

    result_GO <- DO.enrichR(
        df_DGE = DGE_result,
        gene_column = "gene",
        pval_column = "p_val_SC_wilcox",
        log2fc_column = "avg_log2FC_SC_wilcox",
        pval_cutoff = 0.05,
        log2fc_cutoff = 0.25,
        path = NULL,
        filename = "",
        species = "Human",
        go_catgs = "GO_Biological_Process_2023"
    )

    result_GO$celltype <- "CM1"

    # Run SplitBarGSEA visualisation
    DO.SplitBarGSEA(
        df_GSEA = result_GO,
        term_col = "Term",
        col_split = "Combined.Score",
        cond_col = "State",
        pos_cond = "enriched",
        cutoff = 40,
        log10_transform = TRUE,
        figsize = c(12, 8),
        topN = 10,
        colors_pairs = c("sandybrown", "royalblue"),
        alpha_colors = 0.3,
        path = NULL,
        spacing = 5,
        txt_size = 12,
        filename = "SplitBar.svg",
        title = "Top 10 GO Terms in each Condition: ",
        showP = FALSE,
        celltype = "all"
    )

```

DO.Subset

DO.Subset

Description

Creates a subset of a Seurat or SCE object based on either categorical or numeric thresholds in metadata. Allows for subsetting by specifying the ident column, group name, or threshold criteria.

Ideal for extracting specific cell populations or clusters based on custom conditions. Returns a new Seurat or SCE object containing only the subsetted cells and does not come with the Seuratv5 subset issue. Please be aware that right now, after using this function the subset might be treated with Seuv5=False in other functions.

Usage

```
D0.Subset(
  sce_object,
  assay = "RNA",
  ident,
  ident_name = NULL,
  ident_thresh = NULL
)
```

Arguments

sce_object	The seurat or SCE object
assay	assay to subset by
ident	meta data column to subset for
ident_name	name of group of barcodes in ident of subset for
ident_thresh	numeric thresholds as character, e.g ">5" or c(">5", "<200"), to subset barcodes in ident

Value

a subsetted Seurat or SCE object

Author(s)

Mariano Ruz Jurado

Examples

```
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))

sce_data_sub <- D0.Subset(
  sce_object = sce_data,
  ident = "condition",
  ident_name = "healthy"
)
```

D0.TransferLabel	<i>DO.TransferLabel</i>
------------------	-------------------------

Description

Transfers cell-type annotations from a re-annotated subset of a Seurat or SCE object back to the full Seurat or SCE object. This is useful when clusters have been refined or re-labeled in a subset and need to be reflected in the original object.

Usage

```
D0.TransferLabel(sce_object, Subset_obj, annotation_column, subset_annotation)
```

Arguments

sce_object	Seurat or SCE object with annotation in meta.data
Subset_obj	subsetting Seurat or SCE object with re-annotated clusters
annotation_column	column name in meta.data with annotation
subset_annotation	column name in meta.data with annotation in the subsetting object

Value

Seurat or SCE Object with transferred labels

Examples

```
sce_data <-  
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))  
  
sce_data <- D0.TransferLabel(sce_data,  
  sce_data,  
  annotation_column = "annotation",  
  subset_annotation = "annotation"  
)
```

DO.UMAP

*DO.UMAP***Description**

Creates a polished UMAP plot using Seurat's DimPlot or FeaturePlot functions. It allows customization of colors, labels, and other plot elements for better visualisation. The function handles both cluster-based visualisations and gene-based visualisations in a UMAP plot. Ideal for refining UMAP outputs with added flexibility and enhanced presentation.

Usage

```
DO.UMAP(
  sce_object,
  FeaturePlot = FALSE,
  features = NULL,
  group.by = "seurat_clusters",
  umap_colors = NULL,
  text_size = 14,
  label = TRUE,
  order = TRUE,
  plot.title = TRUE,
  legend.position = "none",
  ...
)
```

Arguments

sce_object	The seurat or SCE object
FeaturePlot	Is it going to be a Dimplot or a FeaturePlot?
features	features for Featureplot
group.by	grouping of plot in DImplot and defines in featureplot the labels
umap_colors	what colors to use for UMAP, specify as vector
text_size	Size of text
label	label the clusters on the plot by group.by column
order	Boolean determining whether to plot cells in order of expression.
plot.title	title for UMAP
legend.position	specify legend position
...	Further arguments passed to DimPlot or FeaturePlot function from Seurat

Value

Plot with Refined colors and axes

Author(s)

Mariano Ruz Jurado

Examples

```
sce_data <-
  readRDS(system.file("extdata", "sce_data.rds", package = "D0tools"))

DO.UMAP(
  sce_object = sce_data,
  group.by = "seurat_clusters"
)

DO.UMAP(
  sce_object = sce_data,
  FeaturePlot = TRUE,
  features = c("BAG2", "CD74")
)
```

DO.VlnPlot

*Violin Graph with wilcox test on single cell level***Description**

Creates a violin plot to compare gene expression across different conditions or groups within a Seurat object. It incorporates Wilcoxon rank-sum tests to evaluate statistical differences between conditions. The plot can be customized with options for data transformation, jitter display, and significance annotations. The function also supports multiple conditions and allows for visualisation of statistical results from wilcoxon test.

Usage

```
DO.VlnPlot(
  sce_object,
  SeuV5 = TRUE,
  Feature,
  ListTest = NULL,
  returnValues = FALSE,
  ctrl.condition = NULL,
  group.by = "condition",
  group.by.2 = NULL,
  geom_jitter_args = c(0.2, 0.25, 0.25),
  geom_jitter_args_group_by2 = c(0.1, 0.1, 1),
  vector_colors = c("#1f77b4", "#ea7e1eff", "royalblue4", "tomato2", "darkgoldenrod",
    "palegreen4", "maroon", "thistle3"),
  wilcox_test = TRUE,
  stat_pos_mod = 1.15,
```

```

hjust.wilcox = 0.8,
vjust.wilcox = 2,
size.wilcox = 3.33,
step_mod = 0,
hjust.wilcox.2 = 0.5,
vjust.wilcox.2 = 0,
sign_bar = 0.8
)

```

Arguments

sce_object	combined SCE object or Seurat
SeuV5	Seuratv5 object? (TRUE or FALSE)
Feature	name of the feature
ListTest	List for which conditions wilcox will be performed, if NULL always CTRL group against everything
returnValues	return df.melt.sum data frame containing means and SEM for the set group
ctrl.condition	set your ctrl condition, relevant if running with empty comparison List
group.by	select the seurat sce_object slot where your conditions can be found, default conditon
group.by.2	relevant for multiple group testing, e.g. for each cell type the test between each of them in two conditions provided
geom_jitter_args	vector for dots visualisation in vlnplot: size, width, alpha value
geom_jitter_args_group_by2	controls the jittering of points if group.by.2 is specified
vector_colors	specify a minimum number of colours as you have entries in your condition, default 2
wilcox_test	Boolean if TRUE a bonferoni wilcoxon test will be carried out between ctrl.condition and the rest
stat_pos_mod	value for modifyng statistics height
hjust.wilcox	value for adjusting height of the text
vjust.wilcox	value for vertical of text
size.wilcox	value for size of text of statistical test
step_mod	value for defining the space between one test and the next one
hjust.wilcox.2	value for adjusting height of the text, with group.by.2 specified
vjust.wilcox.2	value for vertical of text, with group.by.2 specified
sign_bar	adjusts the sign_bar with group.by.2 specified

Value

a ggplot or a list used data frames

Author(s)

Mariano Ruz Jurado

Examples

```
sce_data <-  
  readRDS(system.file("extdata", "sce_data.rds", package = "DOtools"))  
  
ListTest <- list()  
ListTest[[1]] <- c("healthy", "disease")  
  
DO.VlnPlot(  
  sce_object = sce_data,  
  SeuV5 = TRUE,  
  Feature = "NKG7",  
  ListTest = ListTest,  
  ctrl.condition = "healthy",  
  group.by = "condition"  
)
```

DOtools*DOtools: A Toolkit for scRNA Data Analysis*

Description

The ‘DOtools’ package provides a set of functions for advanced data processing, visualisation, and statistical analysis in Seurat objects. It includes functions for cell-type prediction, reclustering, creating polished UMAP plots, subsetting Seurat objects, and various statistical analyses like Wilcoxon tests and SEM graphs.

Details

This package includes the following functions:

- [DO.BoxPlot](#): A function for creating box plots with Wilcoxon test results.
- [DO.CellTypist](#): A function for running CellTypist on Seurat and SCE objects to predict cell types.
- [DO.DietSCE](#): A function for diet-based analysis of Seurat and SCE objects.
- [DO.Dotplot](#): A function for creating dot plots for visualizing gene expression.
- [DO.FullRecluster](#): A function for fine-grained reclustering of Seurat and SCE objects.
- [DO.BarplotClustert](#): A function for generating mean and SEM graphs for cluster-based analysis with t-tests.
- [DO.BarplotWilcox](#): A function for generating mean and SEM graphs with Wilcoxon test results.
- [DO.Subset](#): A function for subsetting Seurat and SCE objects based on metadata.

- [D0.UMAP](#): A function for creating polished UMAP plots using either DimPlot or FeaturePlot.
- [D0.VlnPlot](#): A function for generating violin plots with Wilcoxon test results.
- [D0.CellComposition](#): A function for visualizing and statistically analyzing cell-type composition changes across conditions using the Scanpro Python package, with support for bootstrapping, proportion plots, and customizable output.
- [D0.Import](#): A function for building a merged Seurat and SCE object from 10x software output, or directly from provided tables.
- [D0.Integration](#): A function for integrating SCE objects and Seurat objects with the provided method.
- [D0.CellBender](#): A function for running CellBender in a virtual conda env with provided raw count h5 files.
- [D0.SplitBarGSEA](#): A function for visualizing GSEA result from a provided df from e.g. metascape
- [D0.scVI](#): A function for running the scVI Integration implemented in scvi-tools.
- [D0.TransferLabel](#): A function for transferring annotation from a subseted object to the original seurat and SCE object.
- [D0.PyEnv](#): A function for creating a conda environment holding all python packages needed for some functions.
- [D0.Correlation](#): A function for creating a correlation plot between provided samples in the category specified.
- [D0.Heatmap](#): A function for generating Heat maps on gene expression data.
- [D0.MultiDGE](#): A function for calculating DEGs on a single cell and pseudo bulk level.
- `dot-Do.BarcodeRanks`: A function for estimating the number of expected cells and droplets.
- `dot-QC.Vlnplot`: A function for estimating the number of expected cells and droplets.

Value

This is a package-level documentation file and does not return a value.

Author(s)

Mariano Ruz Jurado, David Rodriguez Morales

See Also

[D0.BoxPlot](#), [D0.CellTypist](#), [D0.DietSCE](#), [D0.Dotplot](#), [D0.FullRecluster](#), [D0.BarplotClustert](#), [D0.BarplotWilcox](#), [D0.Subset](#), [D0.UMAP](#), [D0.VlnPlot](#), [D0.Import](#), [D0.Integration](#), [D0.CellBender](#), [D0.SplitBarGSEA](#), [D0.scVI](#), [D0.TransferLabel](#), [D0.Heatmap](#), [D0.PyEnv](#), [D0.Correlation](#), [D0.MultiDGE](#), [D0.TransferLabel](#), `dot-Do.BarcodeRanks`, `dot-QC.Vlnplot`

Index

* internal

- .DO.BarcodeRanks, [5](#)
- .QC_Vlnplot, [6](#)
- .annoSegment, [3](#)
- .example_10x, [5](#)
- .DO.BarcodeRanks, [5](#)
- .QC_Vlnplot, [6](#)
- .annoSegment, [3](#)
- .example_10x, [5](#)

- DO.BarplotClustert, [6](#), [43](#), [44](#)
- DO.BarplotWilcox, [8](#), [43](#), [44](#)
- DO.BoxPlot, [10](#), [43](#), [44](#)
- DO.CellBender, [12](#), [44](#)
- DO.CellComposition, [13](#), [44](#)
- DO.CellTypist, [15](#), [43](#), [44](#)
- DO.Correlation, [17](#), [44](#)
- DO.DietSCE, [18](#), [43](#), [44](#)
- DO.Dotplot, [19](#), [43](#), [44](#)
- DO.enrichR, [21](#)
- DO.FullRecluster, [23](#), [43](#), [44](#)
- DO.Heatmap, [24](#), [44](#)
- DO.Import, [27](#), [44](#)
- DO.Integration, [29](#), [44](#)
- DO.MultiDGE, [32](#), [44](#)
- DO.PyEnv, [33](#), [44](#)
- DO.scVI, [34](#), [44](#)
- DO.SplitBarGSEA, [35](#), [44](#)
- DO.Subset, [37](#), [43](#), [44](#)
- DO.TransferLabel, [39](#), [44](#)
- DO.UMAP, [40](#), [44](#)
- DO.VlnPlot, [41](#), [44](#)
- DOtools, [43](#)