

# Package ‘D0tools’

April 7, 2026

**Title** Convenient functions to streamline your single cell data analysis workflow

**Version** 1.0.2

**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** 7.3.3

**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), FNN (>= 1.1.4.1), methods, stats, utils

**Suggests** SummarizedExperiment, knitr, kableExtra, pkgdown, RefManageR, BiocStyle, roxygen2, httr, magick, rmarkdown, assertthat, plyr, rsvg, scran, scater, igraph, sessioninfo, testthat (>= 3.0.0), leidenbase (>= 0.1.36), mockery

**VignetteBuilder** knitr

**URL** <https://marianoruzjurado.github.io/D0tools/>

**StagedInstall** yes

**Config/testthat/edition** 3**git\_url** <https://git.bioconductor.org/packages/DOtools>**git\_branch** RELEASE\_3\_22**git\_last\_commit** 9a910aa**git\_last\_commit\_date** 2026-02-16**Repository** Bioconductor 3.22**Date/Publication** 2026-04-07

**Author** Mariano Ruz Jurado [aut, cre] (ORCID:  
 <<https://orcid.org/0000-0001-5354-5336>>),  
 David Rodriguez Morales [aut] (ORCID:  
 <<https://orcid.org/0000-0002-1819-6991>>),  
 David John [aut] (ORCID: <<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](mailto:ruzjurado@med.uni-frankfurt.de)>**Contents**

.annoSegment . . . . .	3
.DO.BarcodeRanks . . . . .	5
.example_10x . . . . .	5
.kBET_fct . . . . .	6
.QC_Vlnplot . . . . .	7
.run_kbet . . . . .	8
DO.Barplot . . . . .	9
DO.BarplotClustert . . . . .	10
DO.BoxPlot . . . . .	12
DO.CellBender . . . . .	14
DO.CellComposition . . . . .	15
DO.CellTypist . . . . .	17
DO.Correlation . . . . .	18
DO.DietSCE . . . . .	20
DO.Dotplot . . . . .	21
DO.enrichR . . . . .	23
DO.EvalIntegration . . . . .	24
DO.FullRecluster . . . . .	26
DO.Heatmap . . . . .	27
DO.HeatmapFC . . . . .	30
DO.Import . . . . .	34
DO.Integration . . . . .	35
DO.MultiDGE . . . . .	37
DO.PyEnv . . . . .	39
DO.scVI . . . . .	40
DO.SplitBarGSEA . . . . .	41
DO.Subset . . . . .	43
DO.TransferLabel . . . . .	44
DO.UMAP . . . . .	45
DO.VlnPlot . . . . .	46
DOtools . . . . .	48

---

.annoSegment	<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 = 2,  
  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 = 45,  
  textHVjust = 0.2,  
  hjust = 0.1,  
  vjust = -0.5,  
  myFacetGrou = NULL,  
  aes_x = NULL,  
  aes_y = NULL,  
  segment_gap = 0.9  
)
```

### Arguments

- object ggplot list. Default(NULL).
- relSideDist The relative distance ratio to the y axis range. Default(0.1).

aesGroup	Whether use your group column to add rect annotation. Default("FALSE").
aesGroName	The mapping column name. Default(NULL).
annoPos	The position for the annotation to be added. Default("top").
xPosition	The x axis coordinate for the segment. Default(NULL).
yPosition	The y axis coordinate for the segment. Default(NULL).
pCol	The segment colors. Default(NULL).
segWidth	The relative segment width. Default(1).
lty	The segment line type. Default(NULL).
lwd	The segment line width. Default(NULL).
alpha	The segment color alpha. Default(NULL).
lineend	The segment line end. Default("square").
annoManual	Whether annotate by yourself by supplying with x and y coordinates. Default(FALSE).
mArrow	Whether add segment arrow. Default(FALSE).
addBranch	Whether add segment branch. Default(FALSE).
bArrow	Whether add branch arrow. Default(FALSE).
branDirection	The branch direction. Default(1).
branRelSegLen	The branch relative length to the segment. Default(0.3).
addText	Whether add text label on segment. Default(FALSE).
textCol	The text colors. Default(NULL).
textSize	The text size. Default(NULL).
fontfamily	The text fontfamily. Default(NULL).
fontface	The text fontface. Default(NULL).
textLabel	The text textLabel. Default(NULL).
textRot	The text angle. Default(NULL).
textHVjust	The text distance from the segment. Default(0.2).
hjust	The text hjust. Default(NULL).
vjust	The text vjust. Default(NULL).
myFacetGrou	Your facet group name to be added with annotation when object is a faceted object. Default(NULL).
aes_x	= NULL You should supply the plot X mapping name when annotate a facetd plot. Default(NULL).
aes_y	= NULL You should supply the plot Y mapping name when annotate a facetd plot. Default(NULL).
segment_gap	= 0.9 define the gap between segmentation brackets

**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

---

*.kBET\_fct**kBET - k-nearest neighbour batch effect test*

---

**Description**

kBET runs a chi square test to evaluate the probability of a batch effect.

**Usage**

```
.kBET_fct(
  df,
  batch,
  k0 = NULL,
  knn = NULL,
  testSize = NULL,
  do.pca = TRUE,
  dim.pca = 50,
  heuristic = TRUE,
  n_repeat = 100,
  alpha = 0.05,
  addTest = FALSE,
  verbose = FALSE,
  plot = TRUE,
  adapt = TRUE
)
```

**Arguments**

<code>df</code>	dataset (rows: cells, columns: features)
<code>batch</code>	batch id for each cell or a data frame with both condition and replicates
<code>k0</code>	number of nearest neighbours to test on (neighbourhood size)
<code>knn</code>	an n x k matrix of nearest neighbours for each cell (optional)
<code>testSize</code>	number of data points to test, (10 percent sample size default, but at least 25)
<code>do.pca</code>	perform a pca prior to knn search? (defaults to TRUE)
<code>dim.pca</code>	if do.pca=TRUE, choose the number of dimensions to consider (defaults to 50)
<code>heuristic</code>	compute an optimal neighbourhood size k (defaults to TRUE)
<code>n_repeat</code>	to create a statistics on batch estimates, evaluate 'n_repeat' subsets
<code>alpha</code>	significance level
<code>addTest</code>	perform an LRT-approximation to the multinomial test AND a multinomial exact test (if appropriate)
<code>verbose</code>	displays stages of current computation (defaults to FALSE)
<code>plot</code>	if stats > 10, then a boxplot of the resulting rejection rates is created
<code>adapt</code>	In some cases, a number of cells do not contribute to any neighbourhood and this may cause an imbalance in observed and expected batch label frequencies. Frequencies will be adapted if adapt=TRUE (default).

**Value**

list object

1. `summary` - a rejection rate for the data, an expected rejection rate for random labeling and the significance for the observed result
2. `results` - detailed list for each tested cells; p-values for expected and observed label distribution
3. `average.pval` - significance level over the averaged batch label distribution in all neighbourhoods
4. `stats` - extended test summary for every sample
5. `params` - list of input parameters and adapted parameters, respectively
6. `outsider` - only shown if `adapt=TRUE`. List of samples without mutual nearest neighbour:
  - `index` - index of each outsider sample)
  - `categories` - tabularised labels of outsiders
  - `p.val` - Significance level of outsider batch label distribution vs expected frequencies. If the significance level is lower than `alpha`, expected frequencies will be adapted

If the optimal neighborhood size (`k0`) is smaller than 10, NA is returned.

**Author(s)**

Mariano Ruz Jurado (edited from: Maren Buettner)

---

.QC\_Vlnplot

*.QC\_Vlnplot*

---

**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**

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

**Value**

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

**Author(s)**

Mariano Ruz Jurado

---

.run_kbet	<i>kBET function</i>
-----------	----------------------

---

**Description**

kBET quantifies batch mixing in single-cell data by testing whether local neighborhood batch composition deviates from the global distribution, with lower rejection indicating better integration.

**Usage**

```
.run_kbet(
  sce_object,
  embedding,
  batch,
  cells.use = NULL,
  subsample = NULL,
  min_per_batch = NULL,
  ...
)
```

**Arguments**

sce_object	Seurat or SCE object.
embedding	Name of the embedding to test
batch	Name of the sample column in meta.data
cells.use	specify a specific set of cells as vector
subsample	set a fraction for stratified subsetting
min_per_batch	should always be higher than or equal k0

**Value**

list with summary stats of kBET

**Author(s)**

Mariano Ruz Jurado

DO.Barplot

*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**

```
DO.Barplot(
  sce_object,
  Feature,
  ListTest = NULL,
  returnValues = FALSE,
  ctrl.condition = NULL,
  group.by = "condition",
  test_use = "wilcox",
  correction_method = "fdr",
  p_values = NULL,
  bar_colours = NULL,
  stat_pos_mod = 1.15,
  step_mod = 0.2,
  x_label_rotation = 45,
  plot_raw_pvalue = FALSE,
  y_limits = NULL,
  log1p_nUMI = TRUE,
  random_seed = 42
)
```

**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 condition
test_use	perform one of c("wilcox", "wilcox_limma", "bimod", "t", "negbinom", "poisson", "LR", "MAST", "DESeq2", "none"). default "wilcox"
correction_method	correction for p-value calculation. One of c("BH", "bonferroni", "holm", "BY", "fdr", "none")
p_values	Manually providing p-values for plotting, be aware of group size and if necessary make your test return the same amount of values

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
plot_raw_pvalue	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
random_seed	parameter for random state initialisation

**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")

DO.Barplot(
  sce_object = sce_data,
  Feature = "NKG7",
  test_use = "wilcox",
  correction_method="fdr",
  ListTest = ListTest,
  ctrl.condition = "healthy",
  group.by = "condition"
)
```

---

DO.BarplotClustert      *SEM Graph with t test on cluster level*

---

**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,
  random_seed = 42
)
```

**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 condition
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
random_seed	parameter for random state initialisation

**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")

DO.BarplotClustert(
  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"),
  test_use = "wilcox",
  correction_method = "fdr",
  p_values = NULL,
  stat_pos_mod = 1.15,
  step_mod = 0,
  hjust_test = 0.5,
  vjust_test = 0.25,
```

```

    size_test = 3.33,
    hjust_test_2 = 0.5,
    vjust_test_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
ctrl.condition	select condition to compare to
outlier_removal	Outlier calculation
plot_sample	Plot individual sample dot in graph
vector_colors	get the colours for the plot
test_use	perform one of c( "wilcox", "wilcox_limma", "bimod", "t", "negbinom", "poisson", "LR", "MAST", "DESeq2", "none" ). default "wilcox"
correction_method	correction for p-value calculation. One of c("BH", "bonferroni", "holm", "BY", "fdr", "none"). default "fdr"
p_values	Manually providing p-values for plotting, be aware of group size and if necessary make your test return the same amount of values
stat_pos_mod	modifier for where the p-value is plotted increase for higher
step_mod	value for defining the space between one test and the next one
hjust_test	value for adjusting height of the text
vjust_test	value for vertical of text
size_test	value for size of text of statistical test
hjust_test_2	value for adjusting height of the text, with group.by.2 specified
vjust_test_2	value for vertical of text, with group.by.2 specified
sign_bar	adjusts the sign_bar with group.by.2 specified
orderAxis	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")

DO.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
samplenames <- c("Sample_1", "Sample_2")

# Run CellBender (uses GPU by default)
DO.CellBender(
  cellranger_path = cellranger_path,
  output_path = output_path,
  samplenames = samplenames,
  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,
  random_seed = 42,
  ...
)
```

**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 beeaing 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 sued to factorize the segments
n_reps	Number of replicates generated by scanpro

<code>legend.pos.x</code>	adjusts the position of the legend in horizontal
<code>legend.pos.y</code>	adjusts the position of the legend in vertical
<code>cowplot_width</code>	Changes the width on the plotting device for plot
<code>cowlegend_width</code>	Changes the width on the plotting device for legend, useful for adjusting the legend position in combination with <code>legend.pos.x</code> and <code>y</code>
<code>random_seed</code>	parameter for random state initialisation
<code>...</code>	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",
  returnAll = 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, <code>-update-models</code> will be run for celltypist prior to scoring cells.
over_clustering	Column in metadata in object with clustering assignments for cells, default <code>seurat_clusters</code>
assay_normalized	Assay with log1p normalized expressions
returnAll	will additionally return the probability matrix, return will give a list with the first element being the object and second plot and third probability 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"))

DO.Correlation(
  sce_object = sce_data,
  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"
)
```

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**

```
DO.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 " <code>^scale\\.data\\.</code> "

**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 <- DO.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**

```
DO.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,
  sort_y = 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

<code>group.by.y2</code>	second group name to look for in meta data
<code>across.group.by.x</code>	calculate a pseudobulk expression approach for the x-axis categories
<code>across.group.by.y</code>	calculate a pseudobulk expression approach for the y-axis categories
<code>sort_x</code>	Vector sorting the xaxis
<code>sort_y</code>	Vector to sort the yaxis
<code>dot.size</code>	Vector of dot size
<code>plot.margin</code>	= plot margins
<code>midpoint</code>	midpoint in color gradient
<code>scale_gene</code>	If True calculates the Z-score of the average expression per gene
<code>returnValue</code>	return the dataframe behind the plot
<code>log1p_nUMI</code>	log1p the plotted values, boolean
<code>hide_zero</code>	Removes dots for genes with 0 expression
<code>annotation_x</code>	Adds annotation on top of x axis instead on y axis
<code>annotation_x_position</code>	specifies the position for the annotation
<code>annotation_x_rev</code>	reverses the annotations label order
<code>point_stroke</code>	Defines the thickness of the black stroke on the dots
<code>limits_colorscale</code>	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"))

DO.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**

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

---

## Description

This function calculates different metrics to evaluate the integration of scRNA expression matrices in a new dimension. Its a wrapper function around scib batch correction metrics

## Usage

```

DO.EvalIntegration(
  sce_object,
  label_key = "annotation",
  batch_key = "orig.ident",
  type_ = "embed",
  pcr_covariate = "orig.ident",
  pcr_n_comps = 30,
  scale = TRUE,
  verbose = FALSE,
  n_cores = 10,
  assay = "RNA",
  integration = "INTEGRATED.CCA",
  kBET = TRUE,
  cells.use = NULL,
  subsample = NULL,
  min_per_batch = NULL,
  all_scores_silhouette = FALSE,
  ...
)

```

**Arguments**

sce_object	Seurat or SCE object.
label_key	character, Annotation column
batch_key	character, Sample column
type_	character, default: "embed"
pcr_covariate	character, covariate column for pcr
pcr_n_comps	integer, number of components for pcr
scale	boolean, default: TRUE
verbose	boolean, default: FALSE
n_cores	integer, Number of cores used for calculations
assay	character, Name of the assay the integration is saved in
integration	character, Name of the integration to evaluate
kBET	boolean, if kBET should be run
cells.use	vector, named cells to use for kBET subsetting
subsample	float, for stratified subsampling,
min_per_batch	integer, minimum number of cells per batch
all_scores_silhouette	boolean, define if all scores of silhouette return
...	Additionally arguments for kBET

**Value**

DataFrame with score for the given integration

**Author(s)**

Mariano Ruz Jurado

**Examples**

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

DO.EvalIntegration(
  sce_object = sce_data,
  label_key = "annotation",
  batch_key = "orig.ident",
  type_ = "embed",
  pcr_covariate = "orig.ident",
  pcr_n_comps = 30,
  scale = TRUE,
  verbose = FALSE,
  n_cores = 10,
  assay = "RNA",
  integration = "INTEGRATED.CCA",
  kBET = TRUE,
  cells.use = NULL,
  subsample = NULL,
```

```

    min_per_batch = NULL,
    all_scores_silhouette = FALSE
  )

  ## End(Not run)

```

---

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",
  random_seed = 42
)

```

### 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: leiden
<code>graph.name</code>	A builded neirest neighbor graph
<code>random_seed</code>	parameter for random state initialisation

### 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 <- DO.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**

```
DO.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')
clustering_method	clustering method to use when hierarchically clustering the x and y-axis
clustering_metric	metric to use when hierarchically clustering the x- and y-axis
cluster_x_axis	hierarchically clustering the x-axis
cluster_y_axis	hierarchically clustering the y-axis
axs	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"))

DO.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
)

```

---

DO.HeatmapFC

*DO Heatmap of the mean expression of genes across a groups*


---

## Description

Wrapper around `heatmap_foldchange`, which generates a heatmap of showing the foldchange for a set of gene expressions between specified groups. Differential gene expression analysis between the different groups can be performed.

## Usage

```

DO.HeatmapFC(
  sce_object,
  features,
  reference = NULL,
  assay_normalized = "RNA",
  group_by = "seurat_clusters",
  condition_key = "condition",
  groups_order = NULL,
  conditions_order = NULL,
  layer = NULL,
  figsize = c(5, 6),
  ax = NULL,

```

```

swap_axes = TRUE,
title = NULL,
title_fontproperties = list(size = NULL, weight = NULL),
palette = "RdBu_r",
palette_conditions = "tab10",
ticks_fontproperties = list(size = NULL, weight = NULL),
xticks_rotation = NULL,
yticks_rotation = NULL,
vmin = NULL,
vcenter = NULL,
vmax = NULL,
colorbar_legend_title = "Log2FC",
groups_legend_title = "Comparison",
group_legend_ncols = 1,
path = NULL,
filename = "Heatmap.svg",
showP = TRUE,
add_stats = TRUE,
test = c("wilcox"),
correction_method = c("fdr"),
df_pvals = NULL,
stats_x_size = NULL,
square_x_size = NULL,
pval_cutoff = 0.05,
log2fc_cutoff = 0,
linewidth = 0.1,
color_axis_ratio = 0.15
)

```

### Arguments

sce_object	A SingleCellExperiment or Seurat object containing expression data and meta-data.
features	Character vector of gene names or metadata column names to be visualized.
reference	Reference condition used for fold-change calculation.
assay_normalized	Name of the assay containing normalized expression values (default: "RNA").
group_by	Metadata column defining the primary grouping variable (e.g. clusters).
condition_key	Metadata column defining the condition or comparison variable.
groups_order	Optional character vector specifying the order of groups in group_by.
conditions_order	Optional character vector specifying the order of conditions.
layer	Optional layer name to extract expression values from.
figsize	Numeric vector of length two specifying figure width and height.
ax	Optional matplotlib axis object (for Python backend usage).
swap_axes	Logical; whether to swap x- and y-axes.
title	Optional title for the heatmap.
title_fontproperties	Named list specifying font properties for the title (e.g. size, weight).

palette	Color palette used for the heatmap.
palette_conditions	Color palette used for condition annotations.
ticks_fontproperties	Named list specifying font properties for axis tick labels.
xticks_rotation	Rotation angle for x-axis tick labels.
yticks_rotation	Rotation angle for y-axis tick labels.
vmin	Minimum value for the color scale.
vcenter	Center value for the color scale.
vmax	Maximum value for the color scale.
colorbar_legend_title	Title for the color bar.
groups_legend_title	Title for the group legend.
group_legend_ncols	Number of columns in the group legend.
path	Optional path to save the output figure.
filename	Name of the output file.
showP	Logical; whether to display the plot.
add_stats	Logical; whether to add statistical annotations.
test	Statistical test to use (currently "wilcox").
correction_method	Multiple-testing correction method (currently "fdr").
df_pvals	Optional data frame containing precomputed p-values (groups x genes or genes x groups depending on axis orientation).
stats_x_size	Size of the statistical annotation symbol.
square_x_size	Size of the square annotation.
pval_cutoff	P-value significance threshold.
log2fc_cutoff	Minimum absolute log2 fold-change cutoff.
linewidth	Line width of heatmap cell borders.
color_axis_ratio	Relative size of the color bar axis.

**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"))

DO.HeatmapFC(
  sce_object = sce_data,
  features = c("HES4", "ISG15", "TNFRSF18", "TNFRSF4", "MMP23B"),
  reference = NULL,
  assay_normalized = "RNA",
  group_by = "seurat_clusters",
  condition_key = "condition",
  groups_order = c("1", "2", "3", "4", "5", "6", "7", "8"),
  conditions_order = NULL,
  layer = NULL,

  # Figure parameters
  figsize = c(5, 6),
  ax = NULL,
  swap_axes = TRUE,
  title = NULL,
  title_fontproperties = list(size = NULL, weight = NULL),
  palette = "RdBu_r",
  palette_conditions = "tab10",
  ticks_fontproperties = list(size = NULL, weight = NULL),
  xticks_rotation = 45,
  yticks_rotation = NULL,
  vmin = NULL,
  vcenter = NULL,
  vmax = NULL,
  colorbar_legend_title = "Log2FC",
  groups_legend_title = "Comparison",
  group_legend_ncols = 1,

  # IO
  path = NULL,
  filename = "Heatmap.svg",
  show = FALSE,

  # # Statistics
  add_stats = TRUE,
  test = c("wilcox"),
  correction_method = c("bonferroni"),
  df_pvals = NULL,
  stats_x_size = NULL,
  square_x_size = NULL,
  pval_cutoff = 0.05,
  log2fc_cutoff = 0.0,

  # Fx specific
  linewidth = 0.1,
  color_axis_ratio = 0.15
)
```

DO.Import

*DO.Import***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**

pathways	A character vector of paths to directories or files containing raw expression matrices.
ids	A character vector of sample identifiers, matching the order of pathways.
minCellGenes	Integer. Minimum number of cells a gene must be expressed in to be retained. Default is 5.
FilterCells	Logical. If TRUE, applies QC filtering on cells based on mitochondrial content, counts, and feature thresholds. Default is TRUE.
cut_mt	Numeric. Maximum allowed mitochondrial gene proportion per cell. Default is 0.05.
min_counts	Numeric. Minimum UMI count threshold (optional, used only if low_quantile is NULL).
max_counts	Numeric. Maximum UMI count threshold (optional, used only if high_quantile is NULL).
min_genes	Numeric. Minimum number of genes detected per cell to retain. Optional.
max_genes	Numeric. Maximum number of genes detected per cell to retain. Optional.
low_quantile	Numeric. Quantile threshold (0 to 1) to filter low UMI cells (used if min_counts 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 = "CCAIIntegration",
  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,
  random_seed = 42
)

```

**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
random_seed	parameter for random state initialisation

**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 = "CCAIIntegration"
)
```

---

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**

<code>sce_object</code>	The seurat or SCE object
<code>assay</code>	Specified assay in Seurat or SCE object, default "RNA"
<code>method_sc</code>	method to use for single cell DEG analysis, see FindMarkers from Seurat for options, default "wilcox"
<code>group_by</code>	Column in meta data containing groups used for testing, default "condition"
<code>annotation_col</code>	Column in meta data containing information of cell type annotation
<code>sample_col</code>	Column in meta data containing information of sample annotation, default "orig.ident"
<code>ident_ctrl</code>	Name of the condition in <code>group_by</code> to test against as ctrl, default "ctrl"
<code>min_pct</code>	only test genes that are detected in a minimum fraction of <code>min.pct</code> cells in either of the two populations, default is 0
<code>logfc_threshold</code>	Limit testing to genes which show, on average, at least X-fold difference (log-scale) between the two groups of cells, default is 0.
<code>only_pos</code>	Only return positive markers, default FALSE
<code>min_cells_group</code>	Minimum number of cells in one of the groups, default 3
<code>...</code>	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 <- D0.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_dotools_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**

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

```
DO.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**

df_GSEA	dataframe with the results of a gene set enrichment analysis
term_col	column in the dataframe that contains the terms
col_split	column in the dataframe that will be used to sort and split the plot
cond_col	column in the dataframe that contains the condition information
pos_cond	condition that will be shown in the positive side of the plot
cutoff	maximum number of characters per line
log10_transform	if col_split contains values between 0 and 1, assume they are pvals and apply a -log10 transformation
figsize	figure size
topN	how many terms are shown
colors_pairs	colors for each condition (1st color -> negative axis; 2nd color -> positive axis)
alpha_colors	alpha value for the colors of the bars
path	path to save the plot
spacing	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.
txt_size	size of the go terms text
filename	filename for the plot
title	title of the plot
showP	if False, the axis is return
celltype	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

```

DO.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 <- DO.Subset(
  sce_object = sce_data,
  ident = "condition",
  ident_name = "healthy"
)

```

---

DO.TransferLabel

*DO.TransferLabel*

---

### 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

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

### Arguments

sce_object	Seurat or SCE object with annotation in meta.data
Subset_obj	subsetted 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 subsetted object

**Value**

Seurat or SCE Object with transferred labels

**Examples**

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

sce_data <- DO.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 different 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 different test.

**Usage**

```

DO.VlnPlot(
  sce_object,
  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"),
test_use = "wilcox",
correction_method = "fdr",
p_values = NULL,
y_title = "log(nUMI)",
stat_pos_mod = 1.15,
hjust_test = 0.8,
vjust_test = 2,
size_test = 3.33,
step_mod = 0,
hjust_test_2 = 0.5,
vjust_test_2 = 0,
sign_bar = 0.8,
random_seed = 42
)

```

### Arguments

sce_object	combined SCE object or Seurat
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
test_use	perform one of c("wilcox", "wilcox_limma", "bimod", "t", "negbinom", "poisson", "LR", "MAST", "DESeq2", "none"). default "wilcox"
correction_method	correction for p-value calculation. One of c("BH", "bonferroni", "holm", "BY", "fdr", "none"). default "fdr"
p_values	Manually providing p-values for plotting, be aware of group size and if necessary make your test return the same amount of values
y_title	specify title on the y axis. default "log(nUMI)"
stat_pos_mod	value for modifyng statistics height
hjust_test	value for adjusting height of the text
vjust_test	value for vertical of text

size_test	value for size of text of statistical test
step_mod	value for defining the space between one test and the next one
hjust_test_2	value for adjusting height of the text, with group.by.2 specified
vjust_test_2	value for vertical of text, with group.by.2 specified
sign_bar	adjusts the sign_bar with group.by.2 specified
random_seed	parameter for random state initialisation

**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,
  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.Barplot](#): A function for generating mean and SEM graphs with a statistical test indicating significance.
- [DO.Subset](#): A function for subsetting Seurat and SCE objects based on metadata.
- [DO.UMAP](#): A function for creating polished UMAP plots using either DimPlot or FeaturePlot.
- [DO.VlnPlot](#): A function for generating violin plots with Wilcoxon test results.
- [DO.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.
- [DO.Import](#): A function for building a merged Seurat and SCE object from 10x software output, or directly from provided tables.
- [DO.Integration](#): A function for integrating SCE objects and Seurat objects with the provided method.
- [DO.CellBender](#): A function for running CellBender in a virtual conda env with provided raw count h5 files.
- [DO.SplitBarGSEA](#): A function for visualizing GSEA result from a provided df from e.g. metascape
- [DO.scVI](#): A function for running the scVI Integration implemented in scvi-tools.
- [DO.TransferLabel](#): A function for transferring annotation from a subseted object to the original seurat and SCE object.
- [DO.PyEnv](#): A function for creating a conda environment holding all python packages needed for some functions.
- [DO.Correlation](#): A function for creating a correlation plot between provided samples in the category specified.
- [DO.Heatmap](#): A function for generating Heat maps on gene expression data.
- [DO.HeatmapFC](#): A function for generating Heat maps showing foldchanges in expression between specified conditions.
- [DO.MultiDGE](#): A function for calculating DEGs on a single cell and pseudo bulk level.
- [DO.EvalIntegration](#): A function for calculating sciB metrics on integration embeddings
- `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.
- `dot-run_kbet`: A self-contained call for running the kBET function.
- `dot-kBet_fct`: A self-contained version of the kBET algorithm.

**Value**

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

**Author(s)**

Mariano Ruz Jurado, David Rodriguez Morales

**See Also**

[DO.BoxPlot](#), [DO.CellTypist](#), [DO.DietSCE](#), [DO.Dotplot](#), [DO.FullRecluster](#), [DO.BarplotClustert](#), [DO.Barplot](#), [DO.Subset](#), [DO.UMAP](#), [DO.VlnPlot](#), [DO.Import](#), [DO.Integration](#), [DO.CellBender](#), [DO.SplitBarGSEA](#), [DO.scVI](#), [DO.TransferLabel](#), [DO.Heatmap](#), [DO.HeatmapFC](#), [DO.PyEnv](#), [DO.Correlation](#), [DO.MultiDGE](#), [DO.EvalIntegration](#), [DO.TransferLabel](#), [dot-Do.BarcodeRanks](#), [dot-QC.Vlnplot](#), [dot-run\\_kbet](#), [dot-kBet\\_fct](#)

# Index

## \* internal

- .DO.BarcodeRanks, [5](#)
- .QC\_Vlnplot, [7](#)
- .annoSegment, [3](#)
- .example\_10x, [5](#)
- .kBET\_fct, [6](#)
- .run\_kbet, [8](#)
- .DO.BarcodeRanks, [5](#)
- .QC\_Vlnplot, [7](#)
- .annoSegment, [3](#)
- .example\_10x, [5](#)
- .kBET\_fct, [6](#)
- .run\_kbet, [8](#)
  
- DO.Barplot, [9](#), [49](#), [50](#)
- DO.BarplotClustert, [10](#), [49](#), [50](#)
- DO.BoxPlot, [12](#), [48](#), [50](#)
- DO.CellBender, [14](#), [49](#), [50](#)
- DO.CellComposition, [15](#), [49](#)
- DO.CellTypist, [17](#), [48](#), [50](#)
- DO.Correlation, [18](#), [49](#), [50](#)
- DO.DietSCE, [20](#), [48](#), [50](#)
- DO.Dotplot, [21](#), [48](#), [50](#)
- DO.enrichR, [23](#)
- DO.EvalIntegration, [24](#), [49](#), [50](#)
- DO.FullRecluster, [26](#), [49](#), [50](#)
- DO.Heatmap, [27](#), [49](#), [50](#)
- DO.HeatmapFC, [30](#), [49](#), [50](#)
- DO.Import, [34](#), [49](#), [50](#)
- DO.Integration, [35](#), [49](#), [50](#)
- DO.MultiDGE, [37](#), [49](#), [50](#)
- DO.PyEnv, [39](#), [49](#), [50](#)
- DO.scVI, [40](#), [49](#), [50](#)
- DO.SplitBarGSEA, [41](#), [49](#), [50](#)
- DO.Subset, [43](#), [49](#), [50](#)
- DO.TransferLabel, [44](#), [49](#), [50](#)
- DO.UMAP, [45](#), [49](#), [50](#)
- DO.VlnPlot, [46](#), [49](#), [50](#)
- DOtools, [48](#)