Package: leo.sc (via r-universe)

May 19, 2026

Title Layered Exploratory Omics (LEO for single-cell analysis)

Version 0.1.0

Description LEO for single-cell analysis. This package provides useful functions for single-cell analysis workflow.

License GPL (>= 3)

URL https://laleoarrow.github.io/leo.sc/, https://github.com/laleoarrow/leo.sc

BugReports https://github.com/laleoarrow/leo.sc/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

VignetteBuilder knitr

Imports cli, data.table, dplyr, leo.basic, magrittr, methods, Seurat, SeuratObject, ggplot2, patchwork, rlang, tibble, tidyselect, scales, cowplot, glue, matrixStats, Matrix, ggalluvial, ggbeeswarm, circlize, ComplexHeatmap, SingleCellExperiment, DoubletFinder, ROGUE, miloR, Nebulosa, ggrastr, ggsci

Suggests Augur, pbmcapply, devtools, knitr, rmarkdown, pkgdown, enrichplot

Remotes laleoarrow/leo.basic, chris-mcginnis-ucsf/DoubletFinder, PaulingLiu/ROGUE, neurorestore/Augur, powellgenomicslab/Nebulosa

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev libglpk-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev perl python3 zlib1g-dev

Repository https://laleoarrow.r-universe.dev

Date/Publication 2026-03-04 05:03:03 UTC

RemoteUrl https://github.com/laleoarrow/leo.sc

RemoteRef HEAD

RemoteSha 3e2d81c5590e42fe17d72d4fbb19a4bf46b71a13

Contents

`calcROGUE` *Calculate and plot ROGUE index for a Seurat object, with inline filtering*

Description

Filters raw counts by minimum cells and genes, computes entropy and ROGUE metrics.

Usage

```
calcROGUE(
  obj,
  assay = "RNA",
  layer = "counts",
  downsample = 3000,
  min.cells = 10,
  min.genes = 10,
  anno_label = "celltype_broad",
  sample_label = "orig.ident",
  ...
)
```

Arguments

`obj`	A Seurat object with `celltype_broad` and `orig.ident` metadata.
`assay`	Assay name containing raw counts (default "RNA").
`layer`	Layer name for raw counts (default "counts").
`downsample`	Number of cells to downsample per group (default 3000).
`min.cells`	Minimum cells per gene for filtering (default 10).
`min.genes`	Minimum genes per cell for filtering (default 10).
`anno_label`	Cell type annotation label (default "celltype_broad").
`sample_label`	Sample label for grouping (default "orig.ident").
`...`	Additional arguments passed to `rogue()`.or `mc_rogue()`.

Value

An object returned by `rogue()`, invisibly.

check_gene_stats_in_multi_batch
Inspection of gene in multi-batch

Description

Batch-wise gene statistics checking, average expression, and common-gene subsetting

Usage

```
check_gene_stats_in_multi_batch(srt, batch_col = "Batch", layer = "counts")

check_gene_avg_in_multi_batch(
  srt,
  genes,
  batch_col = "Batch",
  layer = "counts"
)

subset_common_gene_in_multi_batch(srt, common_genes, assay = "RNA")
```

Arguments

srt	A Seurat object.
batch_col	Metadata column indicating batch/sample. Default "Batch".
layer	Expression layer to inspect: "counts" for presence; "data" for averaged expression.
genes	Character vector of genes (only for check_gene_avg_in_multi_batch).
common_genes	Character vector of shared genes (only for subset_common_gene_in_multi_batch).
assay	Assay to subset. Default "RNA".

Value

- check_gene_stats_in_multi_batch: list with a tibble stats and two character vectors common, drop.
- check_gene_avg_in_multi_batch: tibble of mean expression (rows = genes, cols = batches).
- subset_common_gene_in_multi_batch: a Seurat object containing only the genes shared by all batches.

Examples

```
## Not run:
## ------------------------------------------------------------------------
## Tutorial: reconcile gene sets across multiple batches
## ------------------------------------------------------------------------
```

```
# Assume `cd4` is a Seurat object with metadata column "Batch"

# Step 1 - inspect gene overlap/uniqueness across batches
res <- check_gene_stats_in_multi_batch(cd4)  # returns stats, common, drop
res$stats                                    # view the tibble summary

# Step 2 - examine average expression of genes missing from >=1 batch
avg <- check_gene_avg_in_multi_batch(cd4, res$drop); head(avg); colSums(avg[-1])

# Step 3 - subset the Seurat object to the intersection gene set
cd4 <- subset_common_gene_in_multi_batch(cd4, res$common)

## End(Not run)
```

`doublet_rate_dictionary` *Lookup expected doublet rate based on cells loaded*

Description

Estimates the 10x Genomics multiplet rate (as a fraction) by finding the closest "# of Cells Loaded" bracket in the standard guideline table. The ratios can refer to: https://github.com/chris-mcginnis-ucsf/DoubletFinder/issues/76 We currently did not find any corresponding ratio for other single cell sequancing platforms.

Usage

```
doublet_rate_dictionary(n_cells)
```

Arguments

`n_cells`	Integer. Number of cells loaded onto the 10x Chromium.

Value

Numeric. Expected multiplet (doublet) rate.

`doublet_removal` *Remove doublets using DoubletFinder*

Description

Estimates doublet rate (if not provided), finds optimal pK, and removes heterotypic doublets from a Seurat object.

Usage

```
doublet_removal(
  seurat_obj = sc_matrix,
  out_path,
  pN = 0.25,
  PCs = 1:20,
  nPCs_for_pK = 1:10,
  doublet_rate = NULL,
  sct = FALSE,
  slim = TRUE,
  verbose = TRUE
)
```

Arguments

`seurat_obj`	Seurat object (after normalization & PCA).
`out_path`	Character. Directory to save UMAP plots. Default: "."
`pN`	Numeric. Proportion of artificial doublets. Default: 0.25.
`PCs`	Integer vector. PCs to use. Default: 1:20.
`nPCs_for_pK`	Integer vector. PCs to use for pK estimation. Default: 1:10.
`doublet_rate`	Numeric. Expected doublet rate (0-1). If NULL, looked up via `doublet_rate_dictionary`.
`sct`	Logical. Whether data is SCTransformed. Default: FALSE.
`slim`	Logical. Whether to slim the Seurat object to only leave RNA count. Default: TRUE.
`verbose`	Logical. Whether to print `leo.log` messages. Default: TRUE.

Value

Seurat object with doublets removed.

Note

- PCs: a vector of statistically significant PCs to use
- pN: the number of artificially generated doublets (default = 0.25); robust, normally do not need fine-tune
- pK: PC neighborhood size used to compute network; need fine-tune
- nExp: threshold used to make doublet/singlet call; based on empirical ratios

filter_clusters_by_percent_or_cell_count

Filter Seurat clusters by percentage or absolute cell count

Description

Filters clusters in a Seurat object based on either a percentage of total cells or an absolute count threshold.

Usage

```
filter_clusters_by_percent_or_cell_count(
  seurat_obj,
  cluster_col,
  pct_threshold = 0.001,
  abs_threshold = NULL
)
```

Arguments

seurat_obj	A Seurat object.
cluster_col	Character. Name of the metadata column for clusters.
pct_threshold	Numeric. Minimum fraction of total cells to keep a cluster (default 0.001).
abs_threshold	Numeric or NULL. If provided, minimum absolute cell count to keep a cluster (overrides pct_threshold).

Value

Character vector of cluster IDs to keep.

format_markers_for_upload

Format marker-gene lists for (bulk) upload

Description

Convert a Seurat **marker table** into plain-text lines of the form `cluster_<id>:geneA,geneB,...`, one line per cluster.

Usage

```
format_markers_for_upload(
  markers_tbl,
  top_n = 10,
  order_by = "avg_log2FC",
  verbose = TRUE
)
```

Arguments

markers_tbl	A tibble / data.frame **containing at least** the columns cluster, gene, and the column named in order_by.
top_n	Integer. Number of genes to keep for each cluster (default **10**).
order_by	Column used to rank genes *within* each cluster. Accepts either a **character string** (e.g. "avg_log2FC") or a *bare* column name (unquoted). Default "avg_log2FC".
verbose	Logical. Print leo_log() messages? Default **TRUE**.

Value

A length-one character vector

Examples

```
top10 <- data.frame(
  cluster = rep(1:2, each = 2),
  gene = c("GeneA", "GeneB", "GeneC", "GeneD"),
  p_val_adj = c(0.01, 0.05, 0.001, 0.02),
  avg_log2FC = c(1.5, 1.2, 2.0, 1.8),
  avg_logFC = c(1.5, 1.2, 2.0, 1.8)
)
txt <- format_markers_for_upload(top10, top_n = 15, order_by = "p_val_adj")
cat(txt)

## tidy-eval style (bare name):
txt2 <- format_markers_for_upload(top10, order_by = avg_log2FC)
```

---

get_cluster_counts *Get cluster counts sorted descending*

Description

Get cluster counts sorted descending

Usage

```
get_cluster_counts(seurat_obj, cluster_col)
```

Arguments

seurat_obj	A Seurat object.
cluster_col	String. Name of the metadata column containing cluster labels.

Value

A data.frame with columns cluster and count, sorted by count descending.

Examples

```
# Suppose your Seurat object is `all` and cluster column is "harmony_clusters"
# get_cluster_counts(all, "harmony_clusters")
```

---

gimme_marker *Give me marker!*

Description

This function gives the most distinguishing marker for a cluster

Usage

```
gimme_marker(markers, cluster_of_interest = 1)
```

Arguments

markers	A data frame or tibble containing marker genes with columns cluster, pct.1, and pct.2.
cluster_of_interest	The cluster number for which to find the most distinguishing marker. Default is 1.

Value

The most distinguishing marker for the specified cluster.

---

leo.augur *Run Augur analysis*

Description

We added functionality to subset srt obj so that you can compare differet group's results

Usage

```
leo.augur(
  srt,
  subset_col = NULL,
  subset_value = c("Control", "Inactive"),
  label_level = NULL,
  label_col = "Stage1",
  cell_type_col = "cell_anno",
  n_threads = 8,
  return = "plot"
)
```

Arguments

srt	Seurat object
subset_col	Character. Column in meta.data to subset by (e.g. "Stage1")
subset_value	Character vector. Values to retain in subset_col
label_level	Character vector. Factor level order for subset_col (default: same as subset_value)
label_col	Character. Column in meta.data used as label for Augur
cell_type_col	Character. Column in meta.data representing cell types
n_threads	Integer. Number of threads for parallel Augur computation
return	Character. If "plot" (default), returns a list of plot and data; otherwise returns raw Augur object

Value

If return = "plot", a list with plot (ggplot object) and dat (Augur result); else, Augur result only

---

leo.marker *Marker hub*

Description

A minimal nested list for quick access to all kinds of markers collected in leo.sc.

Usage

```
leo.marker
```

Format

List with two elements: $data, $source.

Examples

```
leo.marker$t_nk_marker$data # t_nk_marker
```

leo.milo *Run Milo differential abundance workflow on a Seurat object*

Description

Lightweight wrapper to build a Milo object, define neighbourhoods, count cells, and test differential abundance with optional batch-aware design.

Usage

```
leo.milo(
  all,
  sample = "orig.ident",
  milo_mode = "fast",
  group = "case_ctrl",
  group_level = c("ctrl", "vkh"),
  reduced.dim = "harmony",
  k = 50,
  prop = 0.1,
  adjust_k_p_manual = FALSE,
  contrast_list = NULL,
  batch = NULL,
  cell_type = NULL
)
```

Arguments

all	Seurat object
sample	Character. Meta column used as sample identifier (default: "orig.ident")
milo_mode	Character, either "fast" or other. Controls neighbourhood refinement and testing behavior
group	Character. Meta column encoding the biological group/condition (default: "Stage1")
group_level	Character vector. Desired factor levels for group (controls ordering)
reduced.dim	Character. Set to the batch-corrected dim (default: "harmony")
k	Integer. Number of nearest neighbours to use for graph construction (default: 50)
prop	Numeric (0.1-0.2). Proportion of cells to use for neighbourhood definition (default: 0.1) Note that for large data sets, it might be good to set k higher (50-100) and prop lower (0.01-0.1). See: https://github.com/MarioniLab/miloR/issues/108
adjust_k_p_manual	Logical. If TRUE, allows interactive adjustment of k and prop parameters.
contrast_list	List or NULL. Contrast vector/list for differential abundance testing.
batch	NULL or character. Feels useless yet as Seraut obj normally has already processed with batch-integration like harmony.
cell_type	Deprecated! Ignored in current version (placeholder for future stratification)

Value

A list with components `da_results` (Milo differential abundance result) and `milo_obj` (constructed Milo object)

Examples

```
## Not run:
res <- leo.milo(all)
milo_obj   <- res$milo_obj
da_results <- res$da_results

## End(Not run)
```

`leo.ROIE` *Compute Ro/e and draw heatmap*

Description

Build a Ro/e (observed / expected) matrix from single-cell metadata and, optionally, save a heatmap.

Usage

```
leo.ROIE(
  srt,
  filter_col = NULL,
  filter_criteria = NULL,
  anno_col = "cell_anno",
  group_col = "Stage1",
  group_col_order = NULL,
  plot = FALSE,
  plot_path = NULL,
  col_fun = NULL,
  width = NULL,
  height = NULL,
  fontsize = 6,
  heatmap_anno = c("num", "+++", "none"),
  sym_break = c(-Inf, 0.1, 1, 2, 3, Inf),
  ...
)
```

Arguments

`srt`	seurat object
`filter_col`	column used to subset (optional)
`filter_criteria`	values kept in `filter_col`
`anno_col`	row group column; default `"cell_anno"`

group_col	column group column; default "Stage1"
group_col_order	column order in heatmap; auto if NULL (based on the level of group_col if it is a factor)
plot	save heatmap if TRUE
plot_path	PDF path; default "./ROIE.pdf"
col_fun	colour scale: Defaut orange style or redblue
width, height	device size; auto when NULL
fontsize	Font size for cell annotations (default 10).
heatmap_anno	"num", "+++, or "none"
sym_break	numeric breakpoints; **only two presets supported** –> c(-Inf,.1,1,2,3,Inf) or c(-Inf,0,.2,.8,1,Inf)
...	Passed to ComplexHeatmap::Heatmap.

Value

Ro/e matrix

Examples

```
## Not run:
roe <- leo.ROIE(all, filter_col = 'lineage', filter_criteria = c('DC'), fontsize = 4,
                anno_col = 'cell_anno', group_col = 'Stage1', col_fun = "redblue",
                plot = TRUE, plot_path = "./figure/sc/celltype_analysis/dc.roe.pdf",
                heatmap_anno = "+++", border = NA, rect_gp = gpar(col = NA))
head(roe)

## End(Not run)
```

---

leo_milo_vis *Visualize MiloR DA results*

Description

Visualize MiloR DA results

Usage

```
leo_milo_vis(
  milo_obj,
  da_results,
  plot = c(1, 2),
  layout = "UMAP.HARMONY",
  alpha = 0.05,
  log2fc_colours = c(low = "#070091", mid = "white", high = "#910000"),
```

```
  log2fc_limits = c(-5, 5),
  cell_type = "cell_anno",
  mix_threshold = 0.7,
  bee_order = NULL,
  bee_labs = c(x = "Cell Type", y = "Log Fold Change"),
  bee_add_box = TRUE
)
```

Arguments

milo_obj	A Milo object.
da_results	Output from miloR::testNhoods().
plot	Integer vector. Which plots to return: 1 for neighbourhood graph, 2 for beeswarm plot.
layout	Embedding stored in the Milo object (default "UMAP.HARMONY").
alpha	FDR cutoff (default 0.05).
log2fc_colours	Named vector with three colours for the log2FC gradient (c(low, mid, high)).
log2fc_limits	Two-element numeric vector giving colour-scale limits.
cell_type	Character. Column in the metadata used to annotate neighbourhoods (default "cell_anno").
mix_threshold	Minimum fraction of a single annotation to avoid calling the neighbourhood "Mixed".
bee_order	Optional character vector specifying the display order of groups in the beeswarm plot.
bee_labs	Named character vector giving axis titles for the bee plot: (c(x = "Cell Type", y = "Log Fold Change")). If NULL, no axis labels are added.
bee_add_box	Logical. If TRUE, adds a boxplot to the beeswarm plot (default TRUE).

Value

1: UMAP of the DA; 2: bee plot of the DA results; 3: vertical bee plot; 4. DA results with cell type annotation (based on 'cell_type' params)

---

locate_most_different_g_in_2_group

Locate most distinguishing markers between two clusters

Description

Locate most distinguishing markers between two clusters

Usage

```
locate_most_different_g_in_2_group(
  sc_obj,
  ident1,
  ident2,
  assay = "RNA",
  test.use = "wilcox",
  pval.adj = 0.05,
  logfc = 0.5,
  min.pct1 = 0.5,
  max.pct2 = 0.2,
  return_top_n = 1
)
```

Arguments

sc_obj	Seurat object
ident1	First cluster id (string)
ident2	Second cluster id (string)
assay	Assay name, default "RNA"
test.use	DE test, one of "wilcox","roc","MAST"; default "wilcox"
pval.adj	Adjusted p-value cutoff, default 0.05
logfc	|log2FC| cutoff, default 0.5
min.pct1	Minimum detection pct in marker group, default 0.5
max.pct2	Maximum detection pct in other group, default 0.2
return_top_n	Number of top markers to return for each group, default 1.

Value

named vector c(marker1,marker2)

Examples

```
## Not run:
tops <- locate_most_different_g_in_2_group(pbmc, "seurat_clusters", "4","5")

## End(Not run)
```

metadata_drop *Drop specified metadata columns from a Seurat object*

Description

Drop specified metadata columns from a Seurat object

Usage

```
metadata_drop(seu, ..., cols = NULL, select = NULL, ignore_missing = TRUE)
```

Arguments

seu	Seurat object.
...	Bare metadata names or strings (e.g., seurat_clusters, "seurat_clusters").
cols	Optional character vector of metadata column names to drop.
select	A tidyselect expression (e.g., starts_with("predicted")).
ignore_missing	Logical; if TRUE, ignore missing columns and notify.

Value

Seurat object with columns removed from meta.data.

metadata_get_colnames *Get specified metadata columns from a Seurat object*

Description

Get specified metadata columns from a Seurat object

Usage

```
metadata_get_colnames(srt, pattern = "RNA_snn_res")
```

Arguments

srt	Seurat object.
pattern	Pattern to match in column names, default is "RNA_snn_res".

Value

A character vector of column names that match the pattern.

Examples

```
library(Seurat)
# Create a dummy Seurat object with clustering columns
srt <- SeuratObject::pbmc_small
srt$RNA_snn_res.0.8 <- Idents(srt)
metadata_get_colnames(srt)
```

`metadata_keep` *Keep only selected metadata columns in a Seurat object*

Description

Keep only selected metadata columns in a Seurat object

Usage

```
metadata_keep(
  seu,
  ...,
  cols = NULL,
  select = NULL,
  drop_na = FALSE,
  ignore_missing = FALSE
)
```

Arguments

`seu`	Seurat object.
`...`	Bare metadata names or strings (e.g., cell_anno, "cell_anno").
`cols`	Optional character vector of metadata column names to keep.
`select`	A tidyselect expression (e.g., starts_with("predicted")).
`drop_na`	Logical; if TRUE, drop cells with NA in the kept columns.
`ignore_missing`	Logical; if TRUE, ignore missing columns and notify.

Value

Seurat object with trimmed meta.data.

metadata_write_10x *Export Seurat meta.data aligned to barcodes*

Description

Export Seurat meta.data aligned to barcodes

Usage

```
metadata_write_10x(
  seu,
  out_dir,
  file_name = "metadata.tsv.gz",
  cols = NULL,
  compress = TRUE,
  overwrite = FALSE
)
```

Arguments

seu	Seurat object.
out_dir	Output directory.
file_name	Output file name.
cols	Optional character vector of metadata columns to save.
compress	Logical; if TRUE, write gzipped file.
overwrite	Logical; if TRUE, overwrite existing file.

Value

Invisibly returns the output file path.

plot_alluvial *Draw alluvial bars with optional custom palette*

Description

Quickly visualise stacked proportions (e.g. cell-type composition over conditions) as an alluvial plot.

Usage

```
plot_alluvial(
  df,
  x_col = "Group",
  weight_col = "Percentage",
  stratum_col = "Cluster",
  width = 0.3,
  border_size = 0.5,
  x_angle = 0,
  palette = NULL
)
```

Arguments

df	A long-format data frame.
x_col	Column mapped to the x-axis, default "Group".
weight_col	Numeric column summed within each x-stratum, default "Percentage".
stratum_col	Column defining each stacked segment, default "Cluster".
width	Numeric; width of each stratum (default 0.3).
border_size	Line width for stratum borders. Default 0.5.
x_angle	Rotation angle for x-axis labels. Default 0. Accepts 0, 45, or 90.
palette	Optional colour vector; unnamed = applied by order, named = matched by stratum_col.

Value

A ggplot object.

Examples

```
library(dplyr)
example_data <- tibble(
  Group      = rep(c("Ctrl","10","20","30"), each = 7),
  Cluster    = rep(c("Il1b+","Cxcl9+","Spp1+","Folr2+",
                     "Clps+","Mki67+","Marco+"), times = 4),
  Percentage = c(
     5,10,15,20,20,20,10,
    25,20,15,15,10,10, 5,
    30,20,15,10,10,10, 5,
    35,25,15,10, 5, 5, 5)
)

# default palette
plot_alluvial(example_data)

# custom palette (unnamed vector)
my_cols <- c("#A6CEE3","#1F78B4","#B2DF8A","#33A02C",
             "#FB9A99","#E31A1C","#FDBF6F")
```

```
plot_alluvial(example_data, palette = my_cols)

# custom palette (named vector)
named_cols <- c(
  "Il1b+"  = "#BDD7EE",
  "Cxcl9+" = "#6FA8DC",
  "Spp1+"  = "#C6E0B4",
  "Folr2+" = "#93C47D",
  "Clps+"  = "#F4B6C2",
  "Mki67+" = "#E06666",
  "Marco+" = "#F9CB9C")
plot_alluvial(example_data, palette = named_cols)
```

`plot_alluvial_sc` *Alluvial plot from a Seurat object*

Description

Aggregate cell counts by any two metadata fields and draw an alluvial plot with `plot_alluvial`. Extra arguments are passed straight to `plot_alluvial()`.

Usage

```
plot_alluvial_sc(
  obj,
  group_col = "Group",
  cluster_col = "Cluster",
  return = "plot",
  ...
)
```

Arguments

`obj`	A **Seurat** object.
`group_col`	Metadata field mapped to the x-axis (e.g. sample, time).
`cluster_col`	Metadata field defining strata (e.g. cell type / ident).
`return`	return a plot (set "plot") or a list with plot and data (set "both").
`...`	Additional arguments forwarded to `plot_alluvial()`.

Value

A **ggplot2** object.

Examples

```
library(Seurat)
data("pbmc_small")
pbmc_small$Group   <- pbmc_small$orig.ident   # mock group
pbmc_small$Cluster <- Idents(pbmc_small)      # use idents

## default colours
plot_alluvial_sc(pbmc_small)

## custom palette (unnamed)
plot_alluvial_sc(
  pbmc_small,
  palette = c("#8DD3C7","#FFFFB3","#BEBADA","#FB8072",
              "#80B1D3","#FDB462","#B3DE69"))
```

---

plot_dbee *Different-effect-variable Beeswarm Plot*

Description

Visualize continuous effect values (e.g., log2FC/beta/FC) across groups with a reference dashed line. Non-significant points (by p-value and/or deadband) are drawn in gray; significant points use a diverging palette.

Usage

```
plot_dbee(
  df,
  group.by,
  effect_col,
  p_col = NULL,
  p_thresh = 0.05,
  effect_thresh = 0,
  pal_color = c(low = "#5062A7", mid = "white", high = "#BC4B59"),
  log2fc_limits = NULL,
  insignificant_color = "gray80",
  deadband = NULL,
  flip_coord = TRUE,
  point_size = 1,
  seed = NULL,
  ...
)
```

Arguments

df	data.frame/tibble containing grouping and effect columns
group.by	character, column name for grouping

effect_col	character, column name for effect (e.g., "logFC", "beta")
p_col	character or NULL, column name for p-values; if provided, p >= p_thresh is treated as non-significant
p_thresh	numeric, p-value threshold for significance (default 0.05)
effect_thresh	numeric, reference threshold for dashed line and color midpoint (default 0)
pal_color	named vector c(low, mid, high) for diverging palette (default c(low="#5062A7", mid="white", high="#BC4B59"))
log2fc_limits	NULL or numeric length-2 c(L, R); if set, color scale limits are c(effect_thresh-L, effect_thresh+R)
insignificant_color	character, color for non-significant/gray-zone points (default "gray80")
deadband	NULL or non-negative numeric; if set, |effect - effect_thresh| <= deadband will be gray
flip_coord	logical, flip coordinates to show groups vertically (default TRUE)
point_size	numeric, point size (default 1)
seed	NULL or integer, for reproducible quasirandom placement
...	extra args passed to ggbeeswarm::geom_quasirandom()

Value

ggplot object

Examples

```
# ---- Example 1: MiloR-like DA results ----
 set.seed(1)
 milo_df <- tibble::tibble(
   Nhood = paste0("n", seq_len(1200)),
   `Cell Type` = sample(paste0("CT", 1:6), 1200, replace = TRUE),
   logFC = rnorm(1200, sd = 1.2),
   SpatialFDR = runif(1200)
 )
 # Visualize logFC by cell types; non-sig: SpatialFDR >= 0.1; dashed line at 0
 p1 <- plot_dbee(milo_df, group.by = "Cell Type", effect_col = "logFC",
                 p_col = "SpatialFDR", p_thresh = 0.1, effect_thresh = 0,
                 log2fc_limits = c(-.1, .1), deadband = 0.1, point_size = 2, seed = 42)
 print(p1)

 # ---- Example 2: scRNA-seq DEG-like results ----
 set.seed(123)
 deg_df <- tibble::tibble(
   gene = paste0("G", 1:900),
   cluster = sample(paste0("C", 1:5), 900, replace = TRUE),
   log2FC = rnorm(900, mean = rep(seq(-0.4, 0.4, length.out = 5), each = 180), sd = 1),
   p_val_adj = pmin(runif(900)^2, 1)
 )
 # Visualize log2FC by cluster
 p2 <- plot_dbee(
```

```
  deg_df, group.by = "cluster",
  effect_col = "log2FC",
  p_col = "p_val_adj", p_thresh = 0.05,
  effect_thresh = 0,
  pal_color = c(
    low = "#2C7BB6", mid = "#FFFFBF",
    high = "#D7191C"),
  flip_coord = FALSE,
  log2fc_limits = NULL, deadband = 0.05,
  point_size = 2, seed = 7)
print(p2)
```

`plot_gw_density` *Plot gene-weighted density*

Description

Quickly visualize expression gene-weighted density of one or more features based on **Nebulosa**.

Usage

```
plot_gw_density(
  data,
  features,
  reduction = "umap.harmony",
  size = 0.2,
  pal = "magma",
  ncol = 2,
  joint = FALSE,
  combine = TRUE,
  ...
)
```

Arguments

`data`	A Seurat object with a "harmony" reduction and UMAP computed on it.
`features`	Character vector of feature names to plot.
`reduction`	Name of the reduction to use for plotting (default "umap.harmony").
`size`	Size of the geom to be plotted (e.g. point size)
`pal`	Choose from Nebulosa's palettes, e.g. "magma", "inferno", "plasma", "viridis", "cividis".
`ncol`	Number of columns in the output layout (default 2).
`joint`	Return joint density plot? By default FALSE
`combine`	Passed to `Nebulosa::plot_density()`.
`...`	Additional arguments passed to `Nebulosa::plot_density()`.

Value

A patchwork object arranging density plots in a grid.

Examples

```
## Not run:
# Assuming 'obj' is a Seurat object with harmony UMAP
library(Seurat)
library(Nebulosa)
library(patchwork)
data <- SeuratObject::pbmc_small
plot_gw_density(data, c("CD3D", "CD3E"),
                reduction = "tsne", ncol = 2)
# joint density
plot_gw_density(data, c("CD3D", "CD3E"),
                reduction = "tsne",
                joint = TRUE, combine = FALSE)

## End(Not run)
```

---

plot_highlight_cluster

Highlight a cluster

Description

This function highlight a cluster on a dimensional reduction in NPG palette with highlight on top.

Usage

```
plot_highlight_cluster(
  obj,
  cluster_id,
  reduction = NULL,
  group.by = NULL,
  highlight.col = NULL,
  other.col = "grey80",
  pt.size = 0.6,
  pt.shape = 16,
  raster = NULL,
  dpi = 300,
  legend = TRUE,
  legend_labels = NULL,
  legend_breaks = NULL,
  legend_pos = NULL
)
```

Arguments

obj	Seurat object.
cluster_id	Value to highlight (in Idents(obj) or obj[[group.by]]).
reduction	Dimred name; default active reduction.
group.by	Metadata column; NULL uses Idents.
highlight.col	Highlight color; NULL -> ggsci NPG red.
other.col	Background color (default "grey80").
pt.size	Point size.
pt.shape	Point shape (ggplot2 pch), default 16.
raster	Logical or NULL; NULL auto-enable when >1e5 cells.
dpi	Single numeric DPI for raster geoms (default 300).
legend	Show legend (default TRUE).
legend_labels	Named vector to rename legend entries; must include c("highlight","other"). Defaults to c(highlight=cluster_id, other="other").
legend_breaks	Character vector to set legend order/visibility; accepts keys in c("highlight","other") or their display labels in legend_labels. Default c("highlight","other"). Use "highlight" only to hide "other".
legend_pos	Legend position (inside); NULL uses ggplot2 default (outside, right).

Value

ggplot object.

Examples

```
## Not run:
# load demo data
data("pbmc_small", package = "SeuratObject")

# 1) Basic: highlight first cluster on PCA
plot_highlight_cluster(
  pbmc_small,
  cluster_id = levels(Seurat::Idents(pbmc_small))[1],
  reduction = "pca", pt.size = 0.5, raster = FALSE)

# 2) Use UMAP
pbmc_small <- Seurat::RunUMAP(
  pbmc_small, reduction = "pca", dims = 1:10)
plot_highlight_cluster(
  pbmc_small, cluster_id = "0",
  reduction = "umap", pt.size = 0.6, raster = FALSE)

# 3) Custom colors
plot_highlight_cluster(
  pbmc_small, cluster_id = "0", reduction = "pca",
  highlight.col = "#377EB8", other.col = "grey85")
```

```
# 4) Hide legend
plot_highlight_cluster(
  pbmc_small, cluster_id = "0", reduction = "pca",
  legend = FALSE)

# 5) Rename legend entries
plot_highlight_cluster(
  pbmc_small, cluster_id = "0", reduction = "pca",
  legend_labels = c(highlight = "Yes", other = "No"),
  legend_breaks = c("highlight", "other"))

# 6) Show only highlight in legend
plot_highlight_cluster(
  pbmc_small, cluster_id = "0", reduction = "pca",
  legend_labels = c(highlight = "Yes", other = "No"),
  legend_breaks = "highlight")

# 7) Rasterization
plot_highlight_cluster(
  pbmc_small, cluster_id = "0", reduction = "pca",
  raster = TRUE, dpi = 500)

# 8) Metadata column grouping
pbmc_small$Stage1 <- ifelse(
  as.character(Seurat::Idents(pbmc_small)) == "0",
  "Control", "Other")
plot_highlight_cluster(
  pbmc_small, cluster_id = "Control",
  group.by = "Stage1", reduction = "pca")

# 9) Inside legend position
plot_highlight_cluster(
  pbmc_small, cluster_id = "Control",
  group.by = "Stage1", reduction = "pca",
  legend_pos = c(0.8, 0.8))

## End(Not run)
```

`plot_qc` *Plot QC metrics*

Description

Generates violin and scatter plots for QC features and saves them as PDFs.

Usage

```
plot_qc(
  seurat_obj,
```

```
  out_path,
  prefix = "before_",
  save_plot = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

seurat_obj	A Seurat object with QC metrics.
out_path	Character. Directory to save plots.
prefix	Character. Filename prefix. Default: "before_".
save_plot	Logical. Whether to save the plots. Default: TRUE.
verbose	Logical. Whether to print leo.log messages. Default: TRUE.
...	Additional arguments passed to methods.

Value

Invisible NULL.

---

plot_score_signature_heatmap

Plot heatmap of signature scores

Description

This function generates a heatmap of signature scores. You should do this after score_signature().

Usage

```
plot_score_signature_heatmap(
  sc_obj,
  signature_list,
  group,
  group_prefix = NULL,
  scale = "none",
  signature_cat,
  signature_cat_col,
  heatmap_title = NULL,
  save_path = "./signature.pdf",
  width = 6,
  height = 6
)
```

Arguments

sc_obj	A Seurat object.
signature_list	list. A list of gene sets, where each element is a character vector of gene names.
group	A character string specifying the metadata column to group by. (e.g., group = "RNA_snn_res.0.6")
group_prefix	A character string to prefix the column names in the heatmap. Default is NULL.
scale	Whether to scale the matrix. Options are "none", "row", or "column". Default is "none".
signature_cat	Named vector. c("rowname1" = "category1", "rowname2" = "category2"). Provide to each row a category.
signature_cat_col	Named vector. c("category1" = "color1", "category2" = "color2"). Provide to each category a color.
heatmap_title	Character. Title for the heatmap. Default: NULL.
save_path	Path to save the heatmap PDF (Only support pdf). Default is "./signature.pdf".
width	Width of the saved heatmap PDF. Default is 6.
height	Height of the saved heatmap PDF. Default is 6.

Value

heatmap obj.

Examples

```
## Not run:
signature_category <- c("Naive"                        = "Differentiation",
"Activation/Effector function" = "Differentiation",
"Exhaustion"                   = "Differentiation",
"TCR Signaling"                = "Function",
"Cytotoxicity"                 = "Function",
"Cytokine/Cytokine receptor"   = "Function",
"Chemokine/Chemokine receptor" = "Function",
"Senescence"                   = "Function",
"Anergy"                       = "Function",
"NFKB Signaling"               = "Function",
"Stress response"              = "Function",
"MAPK Signaling"               = "Function",
"Adhesion"                     = "Function",
"IFN Response"                 = "Function",
"Oxidative phosphorylation"    = "Metabolism",
"Glycolysis"                   = "Metabolism",
"Fatty acid metabolism"        = "Metabolism",
"Pro-apoptosis"                = "Apoptosis",
"Anti-apoptosis"               = "Apoptosis")
signature_category_color <- c("Differentiation" = "#E49446",
                              "Function"        = "#4BA5DB",
                              "Metabolism"      = "#815F43",
```

```
                              "Apoptosis"       = "#B12F3A")
plot_score_signature_heatmap(cd8, cd8_signature, group = "RNA_snn_res.0.6",
                             group_prefix = "CD8_c", scale = "row",
                             signature_cat = signature_category,
                             signature_cat_col = signature_category_color,
                           save_path = "./figure/sc/annotation/tnk/cd8/demo.signature.pdf",
                             width = 6, height = 6)

## End(Not run)
```

`read_sc_data` *Read single-cell data into a Seurat object*

Description

Supports 10X Matrix, TXT, or RDS formats for raw count import.

Usage

```
read_sc_data(
  sc_path,
  project_name = NULL,
  method = c("10x", "txt", "rds"),
  gene.column = 1,
  min.cells = 3,
  min.features = 300,
  verbose = TRUE,
  ...
)
```

Arguments

`sc_path`	Character. Path to the data directory or file.
`project_name`	Character. Project name for Seurat object. Defaults to basename(sc_path) if NULL.
`method`	Character. Data format to import: "10X_matrix", "txt", or "rds". Default: "10X_matrix".
`gene.column`	Integer. Column index for gene names in 10X data. Default: 1.
`min.cells`	Integer. Minimum cells per feature for CreateSeuratObject. Default: 3.
`min.features`	Integer. Minimum features per cell for CreateSeuratObject. Default: 300.
`verbose`	Logical. Whether to print `leo.log` messages. Default: TRUE.
`...`	Additional arguments passed to CreateSeuratObject().

Value

A Seurat object of raw counts.

`remove_unwant_hvg` *Remove unwanted HVGs*

Description

Filter the current `VariableFeatures()` of a Seurat object and drop genes matching user-defined regular-expression patterns. Default setting (mitochondria / ribosome / TCR / hemoglobin).

Usage

```
remove_unwant_hvg(
  srt,
 pattern_list = list(mitochondrial_genes = "^MT-", ribosomal_genes = "^(RPL|RPS)",
    tcr_genes = "^TR[ABDG]", hemoglobin_genes = "^HB[AB]")
)
```

Arguments

`srt`	A Seurat object with HVGs already defined (e.g. after `FindVariableFeatures()`).
`pattern_list`	Named list of regex patterns. **Default**

- `mitochondrial_genes = "^MT-"`
- `ribosomal_genes = "^(RPL|RPS)"`
- `tcr_genes = "^TR[ABDG]"`
- `hemoglobin_genes = "^HB[AB]"`

Value

Seurat object with filtered `VariableFeatures()`. A concise summary is printed.

Examples

```
## Not run:
srt <- NormalizeData(srt) |>
       FindVariableFeatures(nfeatures = 2000) |>
       remove_unwant_hvg()

## End(Not run)
```

ROIE *Calculate ROIE*

Description

Calculate the Ro/e value from the given crosstab

Usage

```
ROIE(crosstab)
```

Arguments

crosstab	the contingency table of given distribution

Value

matrix of Ro/e values

Note

This function is from https://github.com/yuyang3/pan-B/blob/main/Figure4.R (line 545)

score_signature *Score sc_obj with signature list*

Description

This function calculates module scores for a Seurat object based on a list of gene sets.

Usage

```
score_signature(sc_obj, signature_list, seed = 1)
```

Arguments

sc_obj	A Seurat object.
signature_list	list. A list of gene sets, where each element is a character vector of gene names.
seed	An integer seed for reproducibility. Default is 1.

Value

A Seurat object with module scores added to the metadata.

Examples

```
## Not run:
sc_obj <- score_signature(sc_obj, signature_list))

## End(Not run)
```

`seurat_basic_info` *get basic summary of a seurat object*

Description

get basic summary of a seurat object

Usage

```
seurat_basic_info(seurat_obj, assay = NULL, layer = "counts")
```

Arguments

`seurat_obj`	Seurat object
`assay`	character. assay name, defaults to active assay
`layer`	character. data layer ("counts", "data" or "scale.data")

Value

named list with total_genes, total_cells, avg_genes_per_cell, genes_per_cell

---

`seurat_basic_qc` *Seurat basic QC*

Description

Calculates mitochondrial and hemoglobin percentages, filters cells by QC thresholds, and optionally plots QC metrics before and after filtering.

Usage

```
seurat_basic_qc(
  seurat_obj,
  nFeature_RNA_low = 200,
  nFeature_RNA_high = 7500,
  nCount_RNA_high = 10000,
  percent_mt_high = 10,
  percent_HB_high = 1,
  out_path = NULL,
  save_plot = FALSE,
  verbose = TRUE
)
```

Arguments

seurat_obj	A Seurat object.
nFeature_RNA_low	Lower boundary for number of features per cell. Default: 200.
nFeature_RNA_high	Upper boundary for number of features per cell. Default: 7500.
nCount_RNA_high	Upper boundary for number of counts per cell. Default: 10000.
percent_mt_high	Upper boundary for mitochondrial percentage. Default: 10.
percent_HB_high	Upper boundary for hemoglobin percentage. Default: 1.
out_path	Character. Directory to save QC plots. Required if save_plot is T.
save_plot	Logical. Whether to save QC plots. Default: FALSE.
verbose	Logical. Whether to print leo.log messages. Default: TRUE.

Value

A filtered Seurat object.

---

seurat_standard_normalize_and_scale

Standard seurat normalize_and_scale

Description

Performs data normalization, finds variable features, scales data, runs PCA, constructs graph-based neighbors and clusters, and generates UMAP embeddings.

Usage

```
seurat_standard_normalize_and_scale(
  seurat_obj,
  normalize_method = c("classic", "sctransform", "sctransform&regress"),
  conserve.memory = FALSE,
  cluster_resolution = 1,
  plot_dir = NULL,
  n_hv_gene = 10,
  verbose = TRUE
)
```

Arguments

seurat_obj	A Seurat object.
normalize_method	Character. Normalization method: • "classic" (LogNormalize + FindVariableFeatures + ScaleData) • "sctransform" (SCTransform). • "sctransform&regress" (SCTransform + regress out unwanted sources (currently only percent.mt)).
conserve.memory	Logical. Whether to conserve memory in SCTransform. Default: FALSE.
cluster_resolution	Numeric. Resolution for FindClusters. Default: 1.0.
plot_dir	Character. Directory to save variable feature plot. Default: NULL (i.e., not plot).
n_hv_gene	Integer. Number of top variable features to label. Default: 10.
verbose	Logical. Whether to print `leo.log` messages. Default: TRUE.

Value

A Seurat object after normalization, scaling, clustering, and UMAP embedding.

---

silico_ko *In-silico Knock-Out / Knock-In Analysis (single gene)*

Description

Performs a virtual knock-out or knock-in simulation for a target gene in a single-cell dataset. The function identifies cells with high or low expression of the gene, matches group sizes and cell-type composition, and performs differential expression and enrichment analysis to infer potential functional effects.

Usage

```
silico_ko(
  all,
  gene,
  sko_mode = c("ko", "ki"),
  cell_col = "cell_anno",
  filter_cell_threshold = 10,
  pct_threshold = 0.1,
  abs_threshold = NULL,
  deg_method = "default",
  enrichment_method = c("ORA", "GSEA"),
  enrichment_bg = c("GO", "KEGG", "MKEGG", "Reactome"),
  simplify = TRUE
)
```

Arguments

all	A Seurat object containing expression data and cell metadata.
gene	Character(1). The target gene symbol (must exist in the expression matrix).
sko_mode	One of c("ko", "ki"). Specifies virtual knock-out ("ko") or knock-in ("ki") mode.
cell_col	Character(1). The metadata column indicating cell type.
filter_cell_threshold	Integer. Minimum number of cells per cell type to retain.
pct_threshold	Numeric. Fraction of expressing cells to extract per group (ignored if abs_threshold is set).
abs_threshold	Integer. Absolute number of cells per group to extract; overrides pct_threshold if provided.
deg_method	Character. Differential expression method: "default" or "MAST".
enrichment_method	Character vector. Enrichment methods to run; default is c("ORA", "GSEA").
enrichment_bg	Character vector. Background databases for enrichment; default is c("GO", "KEGG", "MKEGG", "Reactome").
simplify	Logical. Whether to simplify redundant enrichment terms via clusterProfiler::simplify. Defaults to TRUE. Set to FALSE if enrichment results are sparse (e.g., small datasets) to avoid errors.

Details

The pipeline includes:

1. Filtering cells expressing the target gene.
2. Retaining cell types with sufficient cell counts.
3. Selecting top-expressing cells (high group) and matched low-expressing cells (low group).
4. Running differential expression between the two groups using Seurat::FindMarkers.
5. Performing enrichment analysis via leo.basic::leo_enrich.

Value

Invisibly returns a list containing:

- high_cells: Vector of selected high-expression cells.
- low_cells: Vector of selected low-expression cells.
- cell_rato: Data frame summarizing cell-type composition.
- deg_results: Differential expression results (data frame).
- enrichments: Enrichment analysis results.

See Also

FindMarkers, leo_enrich

Examples

```
## Not run:
res <- silico_ko(
  all = srt, gene = "LRRK2",
  sko_mode = "ko", cell_col = "cell_anno",
  pct_threshold = 0.1, deg_method = "default"
)

## End(Not run)
```

`sort_string_numeric_clusters` *Sort string-based cluster labels numerically and refactor*

Description

Sort string-based cluster labels numerically and refactor

Usage

```
sort_string_numeric_clusters(seurat_obj, cluster_col)
```

Arguments

`seurat_obj`	A Seurat object.
`cluster_col`	A character string. The name of the metadata column containing cluster labels as strings.

Value

The Seurat object with the specified cluster column re-factored so that its levels are sorted numerically.

Examples

```
# seurat_obj <- sort_string_numeric_clusters(seurat_obj, "cluster_labels")
```

subset_srt *Subset srt obj easily*

Description

Subset srt obj easily

Usage

```
subset_srt(srt, subset_col, subset_value)
```

Arguments

srt	Seurat object
subset_col	Character. Column name in meta.data to subset on.
subset_value	Character vector. Values to retain in subset_col.

Value

A subsetted Seurat object

write_10x_triple *Export Seurat object to 10x-style triple files*

Description

Export Seurat object to 10x-style triple files

Usage

```
write_10x_triple(
  seu,
  out_dir,
  assay = "RNA",
  slot = "counts",
  compress = TRUE,
  overwrite = FALSE
)
```

Arguments

seu	Seurat object.
out_dir	Output directory.
assay	Assay name (default "RNA").
slot	Slot/layer to export: "counts" (recommended) or "data".
compress	Logical; if TRUE, write gzipped files.
overwrite	Logical; if TRUE, overwrite existing files.

Value

Invisibly returns a list of output file paths.

Index