Package: STBench (via r-universe)

November 22, 2026

Type Package

Title Normalisation and Benchmarking for Spatial Transcriptomics

Version 0.99.2

Description Description: Provides a unified framework for applying and benchmarking normalisation methods for spatial transcriptomics data. The package supports quality control, spatially variable gene analysis, clustering, cell-type annotation, biological interpretation, visualisation, and other downstream analyses to evaluate the preservation of biological signals across normalisation methods.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports ggplot2, dplyr, magrittr, SummarizedExperiment, SingleCellExperiment, SpatialExperiment, scran, scater, sctransform, viridis, patchwork, SingleR, Polychrome, clusterProfiler, mclust, scales, methods, rlang, MatrixGenerics, tidyr, tidyselect, pROC, cluster, stats, withr, sparseMatrixStats

Suggests tibble, nnSVG, SpaNorm, Giotto, Banksy, knitr, rmarkdown, testthat (>= 3.0.0), CellChat, aricode, clevr, org.Mm.eg.db, presto

URL https://lahiru-lsb.github.io/STBench/

VignetteBuilder knitr

biocViews Spatial, GeneExpression, Normalization, QualityControl, Visualization, Clustering, Transcriptomics

BugReports https://github.com/Lahiru-lsb/STBench/issues

Depends R (>= 4.5)

LazyData false

Config/testthat/edition 3

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libmagick++-dev gsfonts libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev zlib1g-dev

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

Date/Publication 2026-11-22 11:30:01 UTC

RemoteUrl https://github.com/BiocStaging/STBench

RemoteRef HEAD

RemoteSha d15144270843e948606bf0e307f0c7f160442b64

Contents

add_qcs . . . 3
calculate_annotation_agreement . . . 4
calculate_clustering_metrics . . . 5
calculate_gene_selection_metrics . . . 6
calculate_go . . . 8
extract_svg_gene_set . . . 10
extract_svg_stats . . . 11
filter_qc . . . 12
get_umap_data . . . 13
norm_ccc_example . . . 14
norm_example . . . 15
norm_giotto . . . 15
norm_none . . . 16
norm_scran . . . 16
norm_sct . . . 17
norm_spanorm . . . 17
normalize_spe . . . 18
plot_annotation_agreement . . . 19
plot_ccc_circle . . . 20
plot_ccc_lr . . . 21
plot_ccc_pathways . . . 22
plot_ccc_sender_receiver . . . 24
plot_celltype_proportion . . . 25
plot_clustering_metrics . . . 26
plot_gene_selection_metrics . . . 27
plot_go . . . 29
plot_go_genes . . . 30
plot_library_size_comparison . . . 32
plot_marker_auc . . . 33
plot_marker_expression . . . 35
plot_marker_ttest . . . 36
plot_normalization_ranking . . . 38
plot_qc . . . 39
plot_qc_cutoffs . . . 40
plot_qc_removed_spots . . . 41
plot_spatial_annotation . . . 42

plot_spatial_clustering 44
plot_spatial_expression 45
plot_svg_gene_set_distribution 47
plot_svg_gene_sets 48
plot_svg_rank 50
plot_svg_rank_correlation 51
plot_svg_rank_vs_stat 52
plot_svg_stat 54
plot_svg_topn_gene_set 55
plot_umap_comparison 56
rank_normalization_methods 58
run_annotation 59
run_banksy_gene_selection 60
run_banksy_individual 61
run_ccc 63
run_svg 64
spe_example 65
svg_example 66

Index **67**

`add_qcs` *Add quality-control metrics*

Description

Calculate basic spot-level and gene-level quality-control metrics from raw counts in a SpatialExperiment object.

Usage

```
add_qcs(spe, assay_name = "counts", mito_pattern = "^MT-")
```

Arguments

`spe`	A SpatialExperiment object.
`assay_name`	Name of assay containing raw counts. Default is '"counts"'.
`mito_pattern`	Regular expression used to identify mitochondrial genes from row names. Default is '"^MT-"'.

Details

Spot-level metrics include library size, log library size, number of detected genes, and mitochondrial proportion.

Gene-level metrics include mean expression and log mean expression.

Value

A SpatialExperiment object with QC metrics added to ‘colData()‘ and ‘rowData()‘.

Examples

```
data(spe_example)

spe_qc <- add_qcs(
  spe_example,
  assay_name = "counts",
  mito_pattern = "^MT-"
)
#' @import sparseMatrixStats
```

---

calculate_annotation_agreement

Calculate pairwise cell-type annotation agreement

Description

Calculate the proportion of spatial locations receiving the same cell-type annotation between each pair of normalisation methods.

Usage

```
calculate_annotation_agreement(
  norm_list,
  annotation_col = "SingleR_label",
  method_order = NULL,
  remove_na = TRUE
)
```

Arguments

norm_list	Named list of annotated SpatialExperiment objects.
annotation_col	Name of the colData column containing cell-type annotations.
method_order	Optional character vector specifying the order of normalisation methods.
remove_na	Logical. If TRUE, spatial locations with a missing annotation in either method are removed before calculating agreement.

Value

A data frame containing pairwise annotation agreement.

Examples

```
data(norm_example)

agreement <- calculate_annotation_agreement(
  norm_list = norm_example,
  annotation_col = "cell_type",
  method_order = NULL,
  remove_na = TRUE
)

agreement
```

`calculate_clustering_metrics` *Calculate BANKSY clustering evaluation metrics*

Description

Calculates ARI, NMI, homogeneity, completeness, and spatial average silhouette width (ASW) for each normalisation method and spatial slice. Overall is the unweighted mean of NMI, HOM, and COM; it is a user-defined composite agreement score rather than a standard accuracy measure.

Usage

```
calculate_clustering_metrics(
  norm_list,
  ground_truth_col,
  cluster_col = "banksy_cluster",
  methods = names(norm_list),
  nmi_variant = "sum",
  asw_max_cells = 3000L,
  seed = 123L
)
```

Arguments

`norm_list`	Named list of SpatialExperiment objects, or a named list whose elements are named lists of SpatialExperiment objects (slices).
`ground_truth_col`	Name of the colData column containing reference labels.
`cluster_col`	Name of the colData column containing predicted clusters.
`methods`	Optional vector of normalisation methods to evaluate.
`nmi_variant`	NMI normalisation; one of "sum", "max", "min", "sqrt", or "joint". "sum" uses arithmetic-mean entropy normalization.
`asw_max_cells`	Maximum number of cells used to calculate spatial ASW per slice. Larger slices are randomly subsampled.
`seed`	Integer random seed for reproducible ASW subsampling.

Details

ASW uses Euclidean distances between the first two spatial coordinates, not expression-space distances. For large slices, ASW is estimated from a random subset of cells to limit memory requirements.

Value

Data frame with method, slice, n, ARI, NMI, HOM, COM, ASW, Overall, n_asw, and ASW_sampled.

Examples

```
data(norm_example)

banksy_norm <- run_banksy_individual(
  norm_list = norm_example,
  seed = 123
)

metrics_df <- calculate_clustering_metrics(
  norm_list = banksy_norm,
  ground_truth_col = "layer_guess",
  cluster_col = "banksy_cluster"
)

metrics_df
```

---

calculate_gene_selection_metrics

Calculate clustering metrics across gene-selection strategies

---

Description

Calculate ARI, NMI, HOM, COM, spatial ASW, and Overall clustering agreement for each gene-selection strategy.

Usage

```
calculate_gene_selection_metrics(
  banksy_list,
  ground_truth_col = "layer_guess",
  cluster_col = "banksy_cluster",
  asw_max_cells = 3000,
  seed = 123
)
```

Arguments

banksy_list	Output from run_banksy_gene_selection().
ground_truth_col	Ground-truth annotation column.
cluster_col	Predicted cluster column.
asw_max_cells	Maximum cells used for spatial ASW.
seed	Random seed.

Value

Data frame containing clustering metrics and gene sets. #' @examples data(norm_example)

banksy_genes <- run_banksy_gene_selection(norm_list = norm_example, methods = names(norm_example), gene_selection = "all", assay_name = "logcounts", k_geom = 15, lambda = 0.8, resolution = 0.2, seed = 123)

metrics_genes <- calculate_gene_selection_metrics(banksy_list = banksy_genes, ground_truth_col = "layer_guess")

Examples

```
data(norm_example)

banksy_results <- run_banksy_individual(
  norm_list = norm_example,
  methods = names(norm_example),
  assay_name = "logcounts",
  cluster_col = "banksy_cluster",
  k_geom = 15,
  lambda = 0.8,
  resolution = 0.2,
  seed = 123
)

banksy_list <- list(all = banksy_results)

metrics_genes <- calculate_gene_selection_metrics(
  banksy_list = banksy_list,
  ground_truth_col = "layer_guess",
  cluster_col = "banksy_cluster",
  asw_max_cells = 3000,
  seed = 123
)

head(metrics_genes)
```

`calculate_go` *Calculate GO enrichment from cluster-specific DE genes*

Description

Identify cluster-specific differentially expressed genes across normalisation methods and perform Gene Ontology enrichment for each cluster.

Usage

```
calculate_go(
  norm_list,
  cluster_col = "banksy_cluster",
  methods = names(norm_list),
  assay_name = "logcounts",
  OrgDb,
  key_type = "SYMBOL",
  ont = "BP",
  fdr_cutoff = 0.05,
  min_prop = 0.7,
  min_spots = 30,
  min_genes = 10,
  pvalue_cutoff = 0.05,
  qvalue_cutoff = 0.05,
  p_adjust_method = "BH",
  test_type = "wilcox",
  readable = TRUE
)
```

Arguments

`norm_list`	Named list of normalised SpatialExperiment objects.
`cluster_col`	Name of the colData column containing cluster assignments.
`methods`	Optional character vector specifying the normalisation methods to analyse.
`assay_name`	Name of the assay used for differential expression analysis.
`OrgDb`	Organism annotation database passed to ‘clusterProfiler::enrichGO()‘.
`key_type`	Gene identifier type.
`ont`	Gene Ontology category. One of ‘"BP"‘, ‘"MF"‘, ‘"CC"‘, or ‘"ALL"‘.
`fdr_cutoff`	FDR threshold used to define significant cluster-specific DE genes.
`min_prop`	Minimum proportion passed to ‘scran::findMarkers()‘.
`min_spots`	Minimum number of spatial locations required for a cluster.
`min_genes`	Minimum number of DE genes required to run GO enrichment.
`pvalue_cutoff`	GO enrichment p-value cutoff.

qvalue_cutoff	GO enrichment q-value cutoff.
p_adjust_method	Multiple-testing correction method.
test_type	Statistical test used for differential expression analysis in ‘scran::findMarkers()‘. Default is ‘"wilcox"‘.
readable	Logical indicating whether gene IDs should be converted to readable symbols.

Details

Each normalisation method should contain one SpatialExperiment object representing the same spatial sample.

Value

A list containing:

- ‘markers‘: full marker results for each method.
- ‘genes‘: significant DE genes used for GO.
- ‘go‘: GO enrichment results.
- ‘universe‘: background genes for each method.

Examples

```
data(norm_ccc_example)

go_result <- calculate_go(
  norm_list = norm_ccc_example,
  cluster_col = "layer_guess",
  methods = names(norm_ccc_example),
  assay_name = "logcounts",
  OrgDb = org.Mm.eg.db::org.Mm.eg.db,
  key_type = "SYMBOL",
  ont = "BP",
  fdr_cutoff = 0.05,
  min_prop = 0.4,
  min_spots = 10,
  min_genes = 20,
  pvalue_cutoff = 0.05,
  qvalue_cutoff = 0.05,
  p_adjust_method = "BH",
  readable = TRUE,
  test_type = "t"
)
```

`extract_svg_gene_set` *Extract SVG statistics for a gene set*

Description

Extract SVG statistics for a gene set

Usage

```
extract_svg_gene_set(
  svg_norm,
  genes,
  stat = "LR_stat",
  gene_id = "rownames",
  method_order = NULL
)
```

Arguments

`svg_norm`	Named SVG results.
`genes`	Character vector containing genes.
`stat`	SVG statistic.
`gene_id`	Gene identifier.
`method_order`	Optional method order.

Value

Data frame.

Examples

```
data(svg_example)

genes <- head(rownames(svg_example[[1]]), 5)

svg_df <- extract_svg_gene_set(
  svg_norm = svg_example,
  genes = genes,
  stat = "LR_stat",
  gene_id = "rownames",
  method_order = names(svg_example)
)

head(svg_df)
```

`extract_svg_stats` *Extract SVG statistics for selected genes*

Description

Extract SVG statistics for selected genes

Usage

```
extract_svg_stats(
  svg_norm,
  genes,
  stat = "LR_stat",
  gene_id = "rownames",
  method_order = NULL
)
```

Arguments

`svg_norm`	Named list of SVG results.
`genes`	Character vector of genes.
`stat`	SVG statistic column.
`gene_id`	Gene identifier.
`method_order`	Optional method order.

Value

Data frame.

Examples

```
# Create a small SpatialExperiment object
counts <- matrix(
  1:12,
  nrow = 3,
  dimnames = list(
    c("GeneA", "GeneB", "GeneC"),
    paste0("Cell", seq_len(4))
  )
)

spe <- SpatialExperiment::SpatialExperiment(
  assays = list(counts = counts)
)

SummarizedExperiment::rowData(spe)$LR_stat <- c(
  2.5, 4.0, 1.2
```

```
)

# Compare SVG statistics across normalization methods
svg_norm <- list(
  None = spe,
  scran = spe
)

result <- extract_svg_stats(
  svg_norm = svg_norm,
  genes = c("GeneA", "GeneB")
)

print(result)
```

`filter_qc` *Filter a SpatialExperiment using QC thresholds*

Description

Filter spatial locations using log library size and genes using log mean expression.

Usage

```
filter_qc(
  spe,
  library_cutoff = NULL,
  gene_cutoff = NULL,
  recalculate = TRUE,
  assay_name = "counts",
  mito_pattern = "^MT-"
)
```

Arguments

`spe`	A SpatialExperiment object containing QC metrics.
`library_cutoff`	Optional minimum log library size.
`gene_cutoff`	Optional minimum log mean gene expression.
`recalculate`	Logical. If TRUE, QC metrics are recalculated after filtering. Default is TRUE.
`assay_name`	Name of assay containing raw counts. Default is ‘"counts"‘.
`mito_pattern`	Regular expression identifying mitochondrial genes. Default is ‘"^MT-"‘.

Details

QC metrics must first be calculated using ‘add_qcs()‘.

Value

A filtered SpatialExperiment object.

Examples

```
data(spe_example)
spe_qc <- add_qcs(
  spe_example,
  assay_name = "counts",
  mito_pattern = "^MT-"
)
spe_filtered <- filter_qc(
  spe_qc,
  library_cutoff = 4.5,
  gene_cutoff = 0.1,
  recalculate = TRUE,
  assay_name = "counts",
  mito_pattern = "^MT-"
)
```

---

get_umap_data *Calculate UMAP embeddings across normalisation methods*

Description

Perform PCA and UMAP independently for each normalisation method using the same sampled spots and common genes.

Usage

```
get_umap_data(
  norm,
  methods = NULL,
  prop_spots = 1,
  seed = 123,
  assay_name = "logcounts",
  n_pcs = 30,
  n_neighbors = 15
)
```

Arguments

norm	Named list of normalised SpatialExperiment objects.
methods	Character vector of methods to compare. If NULL, all available methods are used.
prop_spots	Proportion of spots to sample.
seed	Random seed.

assay_name	Assay used for PCA and UMAP.
n_pcs	Number of principal components.
n_neighbors	Number of UMAP neighbours.

Details

All colData annotations from the reference object are retained for subsequent plotting.

Value

Data frame containing UMAP coordinates, method, spot identifiers, and all colData annotations.

Examples

```
data(norm_example)

umap_data <- get_umap_data(
  norm_example,
  methods = NULL,
  prop_spots = 1,
  seed = 123,
  assay_name = "logcounts",
  n_pcs = 30,
  n_neighbors = 15
)
```

---

norm_ccc_example *Example normalised data for cell-cell communication*

Description

A larger subset of the publicly available STARmap PLUS mouse brain dataset, prepared for demonstrating cell-cell communication and Gene Ontology analysis.

Format

A named list of SpatialExperiment objects.

Details

An initial random subset of 300 genes and 1000 cells was selected using seed 123. Genes and cells with zero total counts were removed.

Gene symbols were harmonised with the mouse CellChat database where unambiguous matches were available.

The subset was normalised using ‘normalize_spe()‘ with the ‘None‘ and ‘SpaNorm‘ methods.

Original cell-type and tissue-region annotations are retained for downstream analyses.

Source

Bioconductor STexampleData package: https://bioconductor.org/packages/STexampleData/

`norm_example` *Example normalised spatial transcriptomics datasets*

Description

Normalisation results generated from ‘spe_example‘ using the STBench ‘normalize_spe()‘ function.

Format

A named list of SpatialExperiment objects.

Details

Two normalisation methods are included: ‘None‘, representing the unnormalised baseline, and ‘SpaNorm‘, representing spatially aware normalisation.

The original cell-type and tissue-region annotations are retained for downstream benchmarking.

Source

Derived from the STARmap PLUS mouse brain dataset available through the Bioconductor STexampleData package.

`norm_giotto` *Giotto normalisation*

Description

Giotto normalisation

Usage

```
norm_giotto(spe, ...)
```

Arguments

`spe`	A SpatialExperiment object.
`...`	Additional arguments passed to Giotto normalisation.

Value

Normalised object.

norm_none *No normalisation*

Description

Applies log1p transformation to raw counts.

Usage

```
norm_none(spe)
```

Arguments

spe	A SpatialExperiment or SingleCellExperiment object.

Value

A modified object containing logcounts.

norm_scran *Scran normalisation*

Description

Scran normalisation

Usage

```
norm_scran(spe)
```

Arguments

spe	A SpatialExperiment or SingleCellExperiment object.

Value

Normalised object.

norm_sct *SCTransform normalisation*

Description

SCTransform normalisation

Usage

```
norm_sct(spe, seed = 123, vst_flavor = "v2", verbosity = 1, ...)
```

Arguments

spe	A SpatialExperiment or SingleCellExperiment object.
seed	Random seed.
vst_flavor	SCTransform vst flavour.
verbosity	Verbosity passed to sctransform.
...	Additional arguments passed to sctransform::vst().

Value

Normalised object.

norm_spanorm *SpaNorm normalisation*

Description

SpaNorm normalisation

Usage

```
norm_spanorm(
  spe,
  sample_p = NULL,
  df_tps = 6,
  adj_method = "logpac",
  seed = 123,
  ...
)
```

Arguments

spe	A SpatialExperiment object.
sample_p	Sampling proportion. If NULL, 3000/ncol(spe) is used.
df_tps	Degrees of freedom for thin-plate spline.
adj_method	Adjustment method.
seed	Random seed.
...	Additional arguments passed to SpaNorm.

Value

Normalised object.

---

normalize_spe *Apply multiple normalisation methods*

Description

Apply multiple normalisation methods

Usage

```
normalize_spe(
  spe,
  methods = c("none", "scran", "sct", "spanorm", "giotto"),
  params = list()
)
```

Arguments

spe	A SpatialExperiment object.
methods	Character vector containing normalisation methods.
params	Named list containing method-specific parameters.

Value

A named list of normalised objects.

Examples

```
data(spe_example)

norm <- normalize_spe(
  spe_example,
  methods = c("None", "SpaNorm"),
  params = list(
    spanorm = list(
```

```
      sample_p = 0.15,
      df_tps = 6,
      adj_method = "logpac"
    )
  )
)
```

`plot_annotation_agreement` *Plot pairwise cell-type annotation agreement*

Description

Plot pairwise annotation agreement between normalisation methods as a heatmap.

Usage

```
plot_annotation_agreement(
  norm_list,
  annotation_col = "SingleR_label",
  method_order = NULL,
  remove_na = TRUE,
  show_values = TRUE,
  digits = 2,
  colour_option = "viridis",
  theme_name = "classic"
)
```

Arguments

`norm_list`	Named list of annotated SpatialExperiment objects.
`annotation_col`	Name of the colData column containing cell-type annotations.
`method_order`	Optional character vector specifying the order of normalisation methods.
`remove_na`	Logical. If TRUE, missing annotations are removed before agreement is calculated.
`show_values`	Logical. If TRUE, agreement values are displayed inside heatmap cells.
`digits`	Number of decimal places shown in heatmap cells.
`colour_option`	Viridis colour option.
`theme_name`	Plot theme. One of ‘"classic"‘, ‘"bw"‘, ‘"minimal"‘, or ‘"gray"‘.

Value

A list containing agreement data and plot.

`plot_ccc_circle` *Plot CellChat circle networks*

Description

Produces CellChat circle plots for one or more pathways across normalization methods.

Usage

```
plot_ccc_circle(ccc_result, pathways, methods = NULL, method_order = NULL)
```

Arguments

`ccc_result`	Output from run_ccc().
`pathways`	Character vector of pathways.
`methods`	Optional normalization methods.
`method_order`	Optional method order.

Details

Rows correspond to pathways and columns correspond to normalization methods.

Value

Invisibly returns NULL.

Examples

```
data(norm_ccc_example)
data("CellChatDB.mouse", package = "CellChat")

ccc_results <- run_ccc(
  spelist = norm_ccc_example,
  celltype_col = "cell_type",
  database = CellChat::CellChatDB.mouse,
  spatial_factors = data.frame(
    ratio = 0.5,
    tol = 12.5
  ),
  interaction_range = 250,
  contact_range = 25,
  nboot = 10,
  min_cells = 5,
  verbose = FALSE
)

detected_pathways <- unique(unlist(
  lapply(ccc_results, function(x) {
```

```
    x$cellchat@netP$pathways
  })
))

if (length(detected_pathways) > 0L) {
  pathway <- detected_pathways[1]

  for (method in names(ccc_results)) {
    cellchat <- ccc_results[[method]]$cellchat

    if (!is.null(cellchat) &&
      pathway %in% cellchat@netP$pathways) {
      CellChat::netVisual_aggregate(
        cellchat,
        signaling = pathway,
        layout = "circle"
      )
    }
  }
}
```

`plot_ccc_lr` *Plot CCC ligand-receptor interactions*

Description

Plot CCC ligand-receptor interactions

Usage

```
plot_ccc_lr(
  ccc_result,
  pathways = NULL,
  interactions = NULL,
  top_n = 15,
  method_order = NULL,
  method_cols = NULL,
  log_scale = FALSE
)
```

Arguments

`ccc_result`	Output from run_ccc().
`pathways`	Optional pathways to include.
`interactions`	Optional LR interactions to include.
`top_n`	Number of strongest LR interactions.
`method_order`	Optional method order.

method_cols	Optional named vector of method colors.
log_scale	Logical. Use log10 y-axis.

Value

A ggplot object.

Examples

```
data(norm_ccc_example)
data("CellChatDB.mouse", package = "CellChat")

ccc_results <- run_ccc(
  spelist = norm_ccc_example,
  celltype_col = "cell_type",
  database = CellChat::CellChatDB.mouse,
  spatial_factors = data.frame(
    ratio = 0.5,
    tol = 12.5
  ),
  interaction_range = 250,
  contact_range = 25,
  nboot = 10,
  min_cells = 5,
  verbose = FALSE
)

p <- plot_ccc_lr(
  ccc_result = ccc_results,
  top_n = 15,
  method_order = names(ccc_results),
  log_scale = FALSE
)

p
```

---

plot_ccc_pathways *Plot CCC signaling pathway strength*

Description

Plot CCC signaling pathway strength

Usage

```
plot_ccc_pathways(
  ccc_result,
  pathways = NULL,
  top_n = 10,
```

```
  method_order = NULL,
  method_cols = NULL,
  log_scale = FALSE
)
```

Arguments

ccc_result	Output from run_ccc().
pathways	Optional character vector of pathways.
top_n	Number of strongest pathways when pathways = NULL.
method_order	Optional method order.
method_cols	Optional named vector of method colors.
log_scale	Logical. Use log10 y-axis.

Value

A ggplot object.

Examples

```
data(norm_ccc_example)
data("CellChatDB.mouse", package = "CellChat")

ccc_results <- run_ccc(
  spelist = norm_ccc_example,
  celltype_col = "cell_type",
  database = CellChat::CellChatDB.mouse,
  spatial_factors = data.frame(
    ratio = 0.5,
    tol = 12.5
  ),
  interaction_range = 250,
  contact_range = 25,
  nboot = 10,
  min_cells = 5,
  verbose = FALSE
)

p <- plot_ccc_pathways(
  ccc_result = ccc_results,
  top_n = 4,
  method_order = names(ccc_results),
  log_scale = FALSE
)

p
```

plot_ccc_sender_receiver *Plot CCC sender-receiver communication*

Description

Plot CCC sender-receiver communication

Usage

```
plot_ccc_sender_receiver(ccc_result, methods = NULL, method_order = NULL)
```

Arguments

ccc_result	Output from run_ccc().
methods	Optional methods to include.
method_order	Optional method order.

Value

A ggplot object.

Examples

```
data(norm_ccc_example)
data("CellChatDB.mouse", package = "CellChat")

ccc_results <- run_ccc(
  spelist = norm_ccc_example,
  celltype_col = "cell_type",
  database = CellChat::CellChatDB.mouse,
  spatial_factors = data.frame(
    ratio = 0.5,
    tol = 12.5
  ),
  interaction_range = 250,
  contact_range = 25,
  nboot = 10,
  min_cells = 5,
  verbose = FALSE
)

p <- plot_ccc_sender_receiver(
  ccc_result = ccc_results,
  methods = names(ccc_results),
  method_order = names(ccc_results)
)

p
```

plot_celltype_proportion *Plot cell-type proportions*

Description

Plot cell-type proportions stored in the colData of annotated SpatialExperiment objects across normalisation methods.

Usage

```
plot_celltype_proportion(
  norm_list,
  annotation_col = "SingleR_label",
  method_order = NULL,
  palette = NULL,
  show = "proportion",
  theme_name = "classic"
)
```

Arguments

norm_list	Named list of annotated SpatialExperiment objects.
annotation_col	Name of the colData column containing cell-type annotations.
method_order	Optional character vector specifying the order of normalisation methods.
palette	Optional named character vector specifying colours for cell types.
show	Character specifying whether to show ‘"proportion"‘ or ‘"count"‘.
theme_name	Plot theme. One of ‘"classic"‘, ‘"bw"‘, ‘"minimal"‘, or ‘"gray"‘.

Value

A list containing summary data and plot.

Examples

```
data(norm_example)

p <- plot_celltype_proportion(
  norm_list = norm_example,
  annotation_col = "cell_type",
  method_order = NULL,
  palette = "turbo",
  show = "proportion",
  theme_name = "classic"
)

p$plot
```

plot_clustering_metrics *Plot clustering metrics across normalisation methods*

Description

Each facet represents one clustering metric. Points are coloured by normalisation method.

Usage

```
plot_clustering_metrics(
  metrics_df,
  metrics = c("ARI", "NMI", "HOM", "COM", "ASW", "Overall"),
  point_size = 4,
  point_alpha = 0.9,
  ncol = 3,
  scales = "free_y",
  palette = NULL,
  theme_name = "classic",
  rotate_x = 45
)
```

Arguments

metrics_df	Data frame from calculate_clustering_metrics().
metrics	Metrics to plot.
point_size	Point size.
point_alpha	Point opacity.
ncol	Number of facet columns.
scales	Facet scales.
palette	Optional named colours for normalisation methods.
theme_name	One of "classic", "bw", "minimal", "gray".
rotate_x	Angle of method labels.

Value

List containing data and ggplot object (‘plot‘).

Examples

```
data(norm_example)

banksy_norm <- run_banksy_individual(
  norm_list = norm_example,
  seed = 123
```

```
)

metrics_df <- calculate_clustering_metrics(
  norm_list = banksy_norm,
  ground_truth_col = "layer_guess",
  cluster_col = "banksy_cluster"
)

p <- plot_clustering_metrics(
  metrics_df = metrics_df,
  metrics = c("ARI", "NMI", "HOM", "COM", "ASW", "Overall"),
  point_size = 4,
  ncol = 2,
  scales = "free_y",
  theme_name = "bw"
)

p$plot
```

---

plot_gene_selection_metrics *Plot BANKSY gene-selection clustering metrics*

Description

Visualise BANKSY clustering performance across normalisation methods and gene-selection strategies.

Usage

```
plot_gene_selection_metrics(
  metrics_df,
  metrics = c("ARI", "NMI", "HOM", "COM", "ASW", "Overall"),
  point_size = 4,
  point_alpha = 0.9,
  ncol = 3,
  scales = "free_y",
  palette = NULL,
  shapes = NULL,
  method_order = NULL,
  gene_selection_order = NULL,
  theme_name = "classic",
  rotate_x = 45
)
```

Arguments

metrics_df	Data frame returned by calculate_gene_selection_metrics().

metrics	Character vector of clustering metrics to display.
point_size	Point size.
point_alpha	Point opacity.
ncol	Number of facet columns.
scales	Facet scales. One of "fixed", "free", "free_x", or "free_y".
palette	Optional named colour vector for normalisation methods.
shapes	Optional named numeric vector specifying plotting symbols for gene-selection strategies.
method_order	Optional ordering of normalisation methods.
gene_selection_order	Optional ordering of gene-selection strategies.
theme_name	Plot theme. One of "classic", "bw", "minimal", or "gray".
rotate_x	Angle of x-axis labels.

Details

Points are coloured by normalisation method and distinguished by shape according to gene selection.

Metrics are displayed in separate facets.

Value

A list containing the plotting data and the combined ggplot object.

Examples

```
data(norm_example)

banksy_genes <- run_banksy_gene_selection(
  norm_list = norm_example,
  methods = names(norm_example),
  gene_selection = "all",
  assay_name = "logcounts",
  k_geom = 15,
  lambda = 0.8,
  resolution = 0.2,
  seed = 123
)

metrics_genes <- calculate_gene_selection_metrics(
  banksy_list = banksy_genes,
  ground_truth_col = "layer_guess"
)

p_genes <- plot_gene_selection_metrics(
  metrics_df = metrics_genes,
  point_size = 4,
```

```
  ncol = 3,
  scales = "free_y",
  theme_name = "classic"
)

p_genes$plot
```

---

plot_go *Plot GO enrichment results*

Description

Plot enriched Gene Ontology terms across normalisation methods and spatial clusters.

Usage

```
plot_go(
  go_result,
  top_n = 5,
  method_col = "method",
  cluster_col = "cluster",
  method_order = NULL,
  point_size_range = c(2, 7),
  theme_name = "classic",
  methods = NULL,
  title = NULL,
  subtitle = NULL,
  ncol = 3
)
```

Arguments

go_result	Output from calculate_go() or GO data frame.
top_n	Number of top GO terms per method and cluster. If NULL, all available GO terms are displayed.
method_col	Column containing normalisation methods.
cluster_col	Column containing cluster assignments.
method_order	Optional normalisation method order.
point_size_range	Minimum and maximum point sizes.
theme_name	Plot theme.
methods	Optional methods to display.
title	Optional plot title.
subtitle	Optional plot subtitle.
ncol	Number of facet columns.

Details

Point size represents gene count, colour represents enrichment significance, and shape represents the normalisation method.

Value

A list containing plotting data, full GO data, and the plot.

Examples

```
data(norm_ccc_example)

go_result <- calculate_go(
  norm_list = norm_ccc_example,
  cluster_col = "layer_guess",
  methods = names(norm_ccc_example),
  assay_name = "logcounts",
  OrgDb = org.Mm.eg.db::org.Mm.eg.db,
  key_type = "SYMBOL",
  ont = "BP",
  fdr_cutoff = 0.05,
  min_prop = 0.4,
  min_spots = 10,
  min_genes = 5,
  pvalue_cutoff = 0.05,
  qvalue_cutoff = 0.05,
  p_adjust_method = "BH",
  readable = TRUE,
  test_type = "t"
)

p_go <- plot_go(
  go_result = go_result,
  top_n = 5,
  method_col = "method",
  cluster_col = "cluster",
  point_size_range = c(2, 7),
  theme_name = "classic",
  ncol = 2
)

p_go$plot
```

---

plot_go_genes *Plot cluster-specific DE genes used for GO*

Description

Plot the top cluster-specific DE genes detected under each normalisation method.

Usage

```
plot_go_genes(
  go_result,
  top_n = 5,
  method_col = "method",
  cluster_col = "cluster",
  gene_col = "gene",
  method_order = NULL,
  point_size = 3,
  palette = NULL,
  theme_name = "classic"
)
```

Arguments

go_result	Result returned by ‘calculate_go()‘, or the gene data frame contained in ‘go_result$genes‘.
top_n	Number of top DE genes shown per normalisation method and cluster.
method_col	Name of the column containing normalisation methods.
cluster_col	Name of the column containing clusters.
gene_col	Name of the column containing gene names.
method_order	Optional character vector specifying the order of normalisation methods.
point_size	Point size.
palette	Optional named character vector specifying colours for normalisation methods.
theme_name	Plot theme. One of ‘"classic"‘, ‘"bw"‘, ‘"minimal"‘, or ‘"gray"‘.

Value

A list containing plotting data and plot.

Examples

```
data(norm_ccc_example)

go_result <- calculate_go(
  norm_list = norm_ccc_example,
  cluster_col = "layer_guess",
  methods = names(norm_ccc_example),
  assay_name = "logcounts",
  OrgDb = org.Mm.eg.db::org.Mm.eg.db,
  key_type = "SYMBOL",
  ont = "BP",
  fdr_cutoff = 0.05,
  min_prop = 0.4,
  min_spots = 10,
  min_genes = 5,
  pvalue_cutoff = 0.05,
  qvalue_cutoff = 0.05,
  p_adjust_method = "BH",
```

```
  readable = TRUE,
  test_type = "t"
)

if (nrow(go_result$genes) > 0L) {
  p_go_genes <- plot_go_genes(
    go_result = go_result,
    top_n = 5,
    method_col = "method",
    cluster_col = "cluster",
    gene_col = "gene",
    point_size = 3,
    theme_name = "classic"
  )

  p_go_genes$plot
}
```

`plot_library_size_comparison` *Compare normalised library sizes across methods*

Description

Calculate and visualise total normalised expression across spatial locations for different normalisation methods.

Usage

```
plot_library_size_comparison(
  norm,
  methods = NULL,
  assay_name = "logcounts",
  point_size = 0.5,
  theme_name = "classic"
)
```

Arguments

`norm`	Named list of normalised SpatialExperiment objects.
`methods`	Character vector of methods to compare. If NULL, all methods except sctransform and Giotto are used.
`assay_name`	Name of the log-normalised assay. Default is "logcounts".
`point_size`	Point size for spatial maps.
`theme_name`	Plot theme.

Details

Log-normalised expression is back-transformed using expm1(), summed across genes, and transformed again using log1p().

Produces a boxplot and spatial maps.

By default, sctransform and Giotto are excluded because their transformed expression values may not represent log1p-transformed non-negative expression.

Value

A list containing comparison data, boxplot, spatial plot, and combined plot.

Examples

```
data(norm_example)

p_ls <- plot_library_size_comparison(
  norm_example,
  methods = NULL,
  assay_name = "logcounts",
  point_size = 0.5,
  theme_name = "classic"
)

p_ls$plot
```

---

plot_marker_auc *Plot marker-gene discrimination across methods*

Description

Calculate ROC AUC for marker genes across normalisation methods using target cell-type labels.

Usage

```
plot_marker_auc(
  norm_list,
  markers,
  target_types,
  annotation_col = "SingleR_label",
  method_order = NULL,
  reference_method = NULL,
  palette = "viridis",
  point_size = 3,
  theme_name = "classic",
  ncol = 2
)
```

Arguments

norm_list	Named list of SpatialExperiment objects.
markers	Character vector of marker genes.
target_types	Named character vector mapping marker genes to their expected target cell types.
annotation_col	Cell-type annotation column.
method_order	Optional normalisation method order.
reference_method	Optional method used to define fixed cell-type labels across methods.
palette	Optional named colour vector or viridis palette name.
point_size	Size of AUC points.
theme_name	Plot theme.
ncol	number of column of the panel

Value

List containing AUC summary data and ggplot.

Examples

```
data(norm_example)

spe <- norm_example[[1]]

labels <- as.character(
  SummarizedExperiment::colData(spe)$cell_type
)

label_counts <- sort(
  table(labels),
  decreasing = TRUE
)

target_label <- names(label_counts)[1]

markers <- head(rownames(spe), 2)

target_types <- stats::setNames(
  rep(target_label, length(markers)),
  markers
)

p <- plot_marker_auc(
  norm_list = norm_example,
  markers = markers,
  target_types = target_types,
  annotation_col = "cell_type",
  reference_method = "None",
```

```
  palette = "magma",
  point_size = 3
)

p$plot
```

`plot_marker_expression` *Plot marker expression across normalisation methods*

Description

Compare marker-gene expression distributions across selected cell types and normalisation methods.

Usage

```
plot_marker_expression(
  norm_list,
  markers,
  annotation_col = "SingleR_label",
  methods = NULL,
  method_order = NULL,
  cell_types = NULL,
  reference_method = NULL,
  scale_expression = TRUE,
  winsorize = TRUE,
  winsor_quantile = 0.99,
  layout = "method_gene",
  palette = "viridis",
  box_width = 0.7,
  theme_name = "classic"
)
```

Arguments

`norm_list`	Named list of SpatialExperiment objects.
`markers`	Character vector of marker genes.
`annotation_col`	Cell-type annotation column.
`methods`	Optional character vector selecting normalisation methods. NULL includes all methods.
`method_order`	Optional character vector specifying the display order of selected methods.
`cell_types`	Optional character vector selecting cell types. NULL includes all cell types.
`reference_method`	Optional method used to define fixed cell-type annotations across methods.

scale_expression
: If TRUE, apply pooled gene-wise Z-score standardisation across methods and cells.

winsorize
: If TRUE, cap expression at the specified upper quantile before plotting or scaling.

winsor_quantile
: Upper winsorisation quantile.

layout
: Facet layout. Either "method_gene" or "gene_celltype".

palette
: Optional named colour vector or viridis palette name.

box_width
: Width of boxplots.

theme_name
: Plot theme.

Value

A list containing plotting data and ggplot.

Examples

```
data(norm_example)

markers <- head(
  rownames(norm_example[[1]]),
  2
)

p <- plot_marker_expression(
  norm_list = norm_example,
  annotation_col = "cell_type",
  markers = markers,
  palette = "turbo"
)

p$plot
```

---

plot_marker_ttest *Compare Marker Gene Expression Between Cell Groups*

Description

Performs Welch's two-sample t-test for selected marker genes, comparing expression in the specified cell group against all other cells across normalisation methods.

Usage

```
plot_marker_ttest(
  norm,
  genes,
  marker_labels,
  region_col = "cell_type",
  assay_name = "logcounts",
  method_order = NULL,
  method_colors = NULL,
  point_size = 3,
  scales = "free_y",
  ncol = NULL,
  nrow = NULL,
  theme_name = "classic"
)
```

Arguments

norm	Named list of normalised SpatialExperiment objects.
genes	Character vector of marker genes.
marker_labels	Expected cell type or region for each gene. Must have the same length as genes.
region_col	Column name in colData containing cell type or spatial region annotations.
assay_name	Expression assay. Default is "logcounts".
method_order	Optional order of normalisation methods.
method_colors	Optional named vector of method colours.
point_size	Point size. Default is 3.
scales	Facet scales: fixed, free, free_x, or free_y.
ncol	Number of facet columns.
nrow	Number of facet rows.
theme_name	Plot theme.

Value

A list containing t-test results and a ggplot object.

Examples

```
data(norm_example)

genes <- head(rownames(norm_example[[1]]), 3)

region <- as.character(
  SummarizedExperiment::colData(norm_example[[1]])$cell_type
)

marker_labels <- rep(
```

```
  names(sort(table(region), decreasing = TRUE))[1],
  length(genes)
)

p_marker <- plot_marker_ttest(
  norm = norm_example,
  genes = genes,
  marker_labels = marker_labels,
  region_col = "cell_type",
  assay_name = "logcounts",
  scales = "free_y",
  ncol = 3,
  theme_name = "bw"
)

p_marker$plot
```

`plot_normalization_ranking` *Plot normalisation method ranking*

Description

Display overall clustering agreement scores across normalisation methods.

Usage

```
plot_normalization_ranking(
  ranking_df,
  bar_width = 0.7,
  palette = NULL,
  label_size = 3.5,
  show_values = TRUE,
  theme_name = "classic"
)
```

Arguments

`ranking_df`	Data frame from rank_normalization_methods().
`bar_width`	Bar width.
`palette`	Optional named colours for normalisation methods.
`label_size`	Numeric label size.
`show_values`	Whether to show score labels.
`theme_name`	One of "classic", "bw", "minimal", "gray".

Value

List containing data and ggplot object (‘plot‘).

Examples

```
data(norm_example)

banksy_norm <- run_banksy_individual(
  norm_list = norm_example,
  seed = 123
)

metrics_df <- calculate_clustering_metrics(
  norm_list = banksy_norm,
  ground_truth_col = "layer_guess"
)

ranking_df <- rank_normalization_methods(
  metrics_df = metrics_df
)

p <- plot_normalization_ranking(
  ranking_df = ranking_df,
  bar_width = 0.7,
  label_size = 3.5,
  theme_name = "classic"
)

p$plot
```

---

plot_qc *Plot quality-control distributions*

Description

Plot distributions of the main spot-level and gene-level QC metrics stored in a SpatialExperiment object.

Usage

```
plot_qc(spe, bins = 45, show_density = TRUE, theme_name = "classic")
```

Arguments

spe	A SpatialExperiment object containing QC metrics generated by ‘add_qcs()‘.
bins	Number of histogram bins. Default is 45.
show_density	Logical. If TRUE, density curves are added to the histograms.
theme_name	Plot theme. One of ‘"classic"‘, ‘"bw"‘, ‘"minimal"‘, or ‘"gray"‘.

Details

The function can be used before or after filtering.

Value

A list containing plotting data, individual plots, and the combined plot.

Examples

```
data(spe_example)

spe_qc <- add_qcs(
  spe_example,
  assay_name = "counts",
  mito_pattern = "^MT-"
)

p_qc <- plot_qc(
  spe_qc,
  bins = 45,
  show_density = TRUE,
  theme_name = "classic"
)

p_qc$plot
```

---

plot_qc_cutoffs *Plot QC cutoff effects*

Description

Show how different QC cutoff values affect the proportion of spots and genes that would be removed.

Usage

```
plot_qc_cutoffs(
  spe,
  library_cutoff = NULL,
  gene_cutoff = NULL,
  n_points = 200,
  theme_name = "classic"
)
```

Arguments

spe	A SpatialExperiment object containing QC metrics generated by ‘add_qcs()‘.
library_cutoff	Optional selected cutoff for log library size.
gene_cutoff	Optional selected cutoff for log mean gene expression.
n_points	Number of cutoff values used to construct each curve. Default is 200.
theme_name	Plot theme. One of ‘"classic"‘, ‘"bw"‘, ‘"minimal"‘, or ‘"gray"‘.

Details

No filtering is performed by this function.

Value

A list containing cutoff data and plot.

Examples

```
data(spe_example)

spe_qc <- add_qcs(
  spe_example,
  assay_name = "counts",
  mito_pattern = "^MT-"
)

ls <- c(100, 200)
gc <- c(20, 50)

p <- lapply(seq_len(2), function(i) {
  p_cutoffs <- plot_qc_cutoffs(
    spe_qc,
    library_cutoff = ls[i],
    gene_cutoff = gc[i],
    n_points = 200,
    theme_name = "classic"
  )

  p_cutoffs$plot
})

patchwork::wrap_plots(p, ncol = 1) +
  patchwork::plot_annotation(tag_levels = "A")
```

---

plot_qc_removed_spots *Plot Spatial Distribution of Removed QC Spots*

Description

Visualises the spatial locations of spots retained and removed after quality control filtering. Retained spots are shown in blue, while removed spots are shown in red.

Usage

```
plot_qc_removed_spots(
  spe,
  spe_filtered,
  point_size = 0.8,
```

```
  theme_name = "classic"
)
```

Arguments

spe	Original SpatialExperiment object before QC filtering.
spe_filtered	SpatialExperiment object after QC filtering.
point_size	Numeric. Size of the plotted spots. Default is 0.8.
theme_name	Character. Plot theme supported by '.get_plot_theme()'. Default is "classic".

Details

Spots are classified by comparing their column names in the original and filtered SpatialExperiment objects. All spatial coordinates are obtained from the original object.

This function helps identify whether QC filtering removes spots from particular spatial regions.

Value

A ggplot object showing the spatial distribution of retained and removed spots.

Examples

```
data(spe_example)

spe_qc <- add_qcs(
  spe_example,
  assay_name = "counts",
  mito_pattern = "^MT-"
)

p_cutoffs <- plot_qc_cutoffs(
  spe_qc,
  library_cutoff = 3.5,
  gene_cutoff = 0.2,
  n_points = 200,
  theme_name = "classic"
)

p_cutoffs$plot
```

---

plot_spatial_annotation

Plot spatial cell-type annotations

Description

Plot cell-type annotations stored in the colData of SpatialExperiment objects across normalisation methods.

Usage

```
plot_spatial_annotation(
  norm_list,
  annotation_col = "SingleR_label",
  method_order = NULL,
  point_size = 0.5,
  palette = NULL,
  ncol = 1,
  theme_name = "classic"
)
```

Arguments

norm_list	Named list of annotated SpatialExperiment objects.
annotation_col	Name of the colData column containing cell-type annotations.
method_order	Optional character vector specifying the order of normalisation methods.
point_size	Point size for spatial locations.
palette	Colour palette. One of "viridis", "magma", "inferno", "plasma", "cividis", "rocket", "mako", "turbo", or a custom character vector of colours. If NULL, the default ggplot2 palette is used.
ncol	Number of facet columns.
theme_name	Plot theme. One of "classic", "bw", "minimal", or "gray".

Value

A list containing plotting data and plot.

Examples

```
data(norm_example)

p <- plot_spatial_annotation(
  norm_list = norm_example,
  annotation_col = "cell_type",
  point_size = 0.5,
  theme_name = "classic"
)

p$plot
```

plot_spatial_clustering *Plot spatial clustering results*

Description

Visualise predicted spatial clusters and ground-truth annotations across normalisation methods.

Usage

```
plot_spatial_clustering(
  norm_list,
  cluster_col = "banksy_cluster",
  methods = names(norm_list),
  method_order = NULL,
  point_size = 0.5,
  palette = NULL,
  ncol = 3,
  legend_point_size = 4,
  theme_name = "classic",
  ground_truth_col = NULL,
  show = c("predicted", "both", "truth"),
  truth_palette = NULL
)
```

Arguments

norm_list	Named list of SpatialExperiment objects or nested lists of SpatialExperiment objects.
cluster_col	Column containing predicted clusters.
methods	Optional normalisation methods to plot.
method_order	Optional method order.
point_size	Spatial point size.
palette	Optional named colours for predicted clusters.
ncol	Number of columns for spatial facets.
legend_point_size	Point size in legends.
theme_name	Plot theme.
ground_truth_col	Ground-truth annotation column.
show	One of "both", "predicted", or "truth".
truth_palette	Optional named colours for ground truth.

Details

When show = "both", the ground truth is displayed once per spatial slice, followed by predicted cluster maps across all selected normalisation methods.

Predicted clusters share one colour scale and legend. Ground truth uses a separate palette and legend.

Value

List containing data, palettes, individual plots, and combined plot.

Examples

```
data(norm_example)

banksy_norm <- run_banksy_individual(
  norm_list = norm_example,
  k_geom = 15,
  lambda = 0.8,
  resolution = 0.2,
  seed = 123
)

p <- plot_spatial_clustering(
  norm_list = banksy_norm,
  cluster_col = "banksy_cluster",
  ground_truth_col = "layer_guess",
  show = "both",
  point_size = 0.5,
  ncol = 2,
  theme_name = "classic"
)

p$plot
```

---

plot_spatial_expression

Plot spatial expression across normalisation methods

Description

Visualise spatial expression of selected genes across multiple normalisation methods.

Usage

```
plot_spatial_expression(
  norm_list,
  genes,
  assay_name = "logcounts",
```

```
  method_order = NULL,
  scale_expr = TRUE,
  point_size = 0.5,
  colour_option = "magma",
  ncol_genes = 1,
  theme_name = "classic",
  methods = NULL,
  scale_by = c("gene", "gene_method")
)
```

Arguments

norm_list	Named list of normalised SpatialExperiment objects.
genes	Character vector of genes to plot.
assay_name	Name of the expression assay.
method_order	Optional ordering of selected methods.
scale_expr	Logical. Whether to scale expression.
point_size	Spatial point size.
colour_option	Viridis colour option.
ncol_genes	Number of columns for unscaled plots.
theme_name	Plot theme.
methods	Optional character vector of normalisation methods. If NULL, all available methods are used.
scale_by	Scaling approach: "gene" or "gene_method".

Details

Expression can be scaled between 0 and 1 either across all selected methods for each gene or independently within each gene and normalisation method.

When scale_expr = FALSE, original assay values are used. Each gene receives a separate plot with a common colour scale across the selected methods.

Value

A list containing plotting data, individual gene plots when unscaled, and the combined plot.

Examples

```
data(norm_example)

genes <- head(rownames(norm_example[[1]]), 2)

p_expr <- plot_spatial_expression(
  norm_list = norm_example,
  genes = genes,
  assay_name = "logcounts",
  scale_expr = TRUE,
```

```
  scale_by = "gene",
  point_size = 0.5,
  colour_option = "magma",
  ncol_genes = 1,
  theme_name = "classic"
)

p_expr$plot
```

`plot_svg_gene_set_distribution` *Plot SVG statistic distribution for a gene set*

Description

Plot SVG statistic distribution for a gene set

Usage

```
plot_svg_gene_set_distribution(
  svg_norm,
  genes,
  stat = "LR_stat",
  method_order = NULL,
  method_colors = NULL,
  gene_id = "rownames",
  transform = c("log1p", "log", "none"),
  plot_type = c("boxplot", "violin"),
  add_jitter = TRUE,
  point_size = 1.5,
  alpha = 0.6,
  theme_name = "classic"
)
```

Arguments

`svg_norm`	Named SVG results.
`genes`	Gene set.
`stat`	SVG statistic.
`method_order`	Optional method order.
`method_colors`	Optional named colours.
`gene_id`	Gene identifier.
`transform`	Transformation.
`plot_type`	boxplot or violin.
`add_jitter`	Add gene-level points.

point_size	Point size.
alpha	Point transparency.
theme_name	Plot theme.

Value

List containing data and plot.

Examples

```
data(svg_example)

genes <- head(rownames(svg_example[[1]]), 10)

p <- plot_svg_gene_set_distribution(
  svg_norm = svg_example,
  genes = genes,
  stat = "LR_stat",
  transform = "log",
  plot_type = "boxplot"
)

p$plot
```

---

plot_svg_gene_sets *Compare SVG statistics across multiple gene sets*

Description

Plot distributions of spatial variability statistics for multiple user-defined gene sets across normalisation methods.

Usage

```
plot_svg_gene_sets(
  svg_norm,
  gene_sets,
  stat = "LR_stat",
  method_order = NULL,
  method_colors = NULL,
  gene_id = "rownames",
  transform = c("log1p", "log", "none"),
  plot_type = c("boxplot", "violin"),
  scales = c("free_y", "fixed", "free"),
  facet_ncol = 2,
  add_jitter = TRUE,
  point_size = 1,
  alpha = 0.65,
```

```
  jitter_alpha = 0.5,
  box_width = 0.65,
  title = NULL,
  subtitle = NULL,
  x_label = "Normalisation method",
  y_label = NULL,
  show_legend = FALSE,
  theme_name = "classic"
)
```

Arguments

svg_norm	Named list of SVG results.
gene_sets	Named list of character vectors containing genes for each gene set.
stat	SVG statistic. Default is "LR_stat".
method_order	Optional normalisation method order.
method_colors	Optional named vector of method colours.
gene_id	Gene identifier.
transform	Transformation: "log1p", "log", or "none".
plot_type	Plot type: "boxplot" or "violin".
scales	Facet scales: "fixed", "free_y", or "free".
facet_ncol	Number of facet columns.
add_jitter	Logical. Display individual gene points.
point_size	Size of jittered points.
alpha	Transparency of boxplots or violins.
jitter_alpha	Transparency of jittered points.
box_width	Width of boxplots.
title	Optional plot title.
subtitle	Optional plot subtitle.
x_label	X-axis label.
y_label	Optional y-axis label.
show_legend	Logical. Display method legend.
theme_name	Plot theme.

Details

Gene sets may include housekeeping genes, negative-control genes, biological marker genes, or any custom gene sets.

Each gene set is displayed in a separate facet.

Value

List containing plotting data and ggplot.

Examples

```
data(svg_example)

marker_genes <- head(rownames(svg_example[[1]]), 5)

p <- plot_svg_stat(
  svg_norm = svg_example,
  genes = marker_genes,
  stat = "LR_stat",
  transform = "log"
)

p$plot
```

`plot_svg_rank` *Plot SVG ranks for selected genes*

Description

Plot SVG ranks for selected genes

Usage

```
plot_svg_rank(
  svg_norm,
  genes,
  method_order = NULL,
  method_colors = NULL,
  point_size = 3,
  gene_id = "rownames",
  theme_name = "classic"
)
```

Arguments

`svg_norm`	Named list of SVG results.
`genes`	Genes to plot.
`method_order`	Optional method order.
`method_colors`	Optional method colours.
`point_size`	Point size.
`gene_id`	Gene identifier.
`theme_name`	Plot theme.

Value

List containing data and plot.

Examples

```
data(svg_example)

marker_genes <- head(rownames(svg_example[[1]]), 5)

p <- plot_svg_rank(
  svg_norm = svg_example,
  genes = marker_genes
)

p$plot
```

`plot_svg_rank_correlation` *Plot SVG Rank Correlation Across Normalisation Methods*

Description

Calculates pairwise Spearman rank correlations of SVG rankings across normalisation methods.

Usage

```
plot_svg_rank_correlation(
  svg_norm,
  methods = NULL,
  gene_id = "rownames",
  method_order = NULL,
  method_colors = c("#B2182B", "#F7F7F7", "#2166AC"),
  show_values = TRUE,
  digits = 2,
  theme_name = "classic"
)
```

Arguments

`svg_norm`	Named list of SVG results.
`methods`	Optional methods to compare.
`gene_id`	Gene identifier. Default is "rownames".
`method_order`	Optional order of normalisation methods.
`method_colors`	Three colours for low, middle, and high correlations.
`show_values`	Display correlation values. Default TRUE.
`digits`	Number of decimal places.
`theme_name`	Plot theme.

Details

Only genes shared by all selected methods are used. The heatmap displays the lower triangle of the correlation matrix, including the diagonal.

Value

A list containing the correlation matrix, heatmap data, shared gene ranks, and ggplot object.

Examples

```
data(svg_example)

p <- plot_svg_rank_correlation(
  svg_norm = svg_example,
  show_values = TRUE,
  digits = 2
)

p$plot
```

---

plot_svg_rank_vs_stat *Plot SVG Rank Against SVG Statistic*

Description

Visualises the relationship between SVG statistics and SVG ranks for selected genes across normalisation methods.

Usage

```
plot_svg_rank_vs_stat(
  svg_norm,
  genes,
  stat = "LR_stat",
  method_order = NULL,
  method_colors = NULL,
  transform = c("log1p", "log", "none"),
  scales = c("fixed", "free", "free_x", "free_y"),
  ncol = NULL,
  nrow = NULL,
  point_size = 3,
  gene_id = "rownames",
  theme_name = "classic"
)
```

Arguments

svg_norm	Named list of SVG results.
genes	Genes to plot.
stat	SVG statistic. Default is "LR_stat".
method_order	Optional method order.
method_colors	Optional named vector of method colours.
transform	Transformation: log1p, log, or none.
scales	Facet scales: fixed, free, free_x, or free_y.
ncol	Number of facet columns.
nrow	Number of facet rows.
point_size	Point size.
gene_id	Gene identifier.
theme_name	Plot theme.

Details

Each point represents a normalisation method, with genes displayed in separate facets.

Value

A list containing data and plot.

Examples

```
data(svg_example)

marker_genes <- head(rownames(svg_example[[1]]), 5)

p <- plot_svg_rank_vs_stat(
  svg_norm = svg_example,
  genes = marker_genes,
  stat = "LR_stat",
  transform = "log"
)

p$plot
```

`plot_svg_stat` *Plot SVG statistic for selected genes*

Description

Plot SVG statistic for selected genes

Usage

```
plot_svg_stat(
  svg_norm,
  genes,
  stat = "LR_stat",
  method_order = NULL,
  method_colors = NULL,
  transform = c("log1p", "log", "none"),
  point_size = 3,
  gene_id = "rownames",
  theme_name = "classic"
)
```

Arguments

`svg_norm`	Named list of SVG results.
`genes`	Genes to plot.
`stat`	SVG statistic.
`method_order`	Optional method order.
`method_colors`	Optional named vector of method colours.
`transform`	Transformation: log1p, log, or none.
`point_size`	Point size.
`gene_id`	Gene identifier.
`theme_name`	Plot theme.

Value

A list containing data and plot.

`plot_svg_topn_gene_set` *Proportion of a gene set among top-ranked SVGs*

Description

Calculate and visualise the proportion of genes belonging to a user-defined gene set among the top-ranked spatially variable genes (SVGs) across normalisation methods.

Usage

```
plot_svg_topn_gene_set(
  svg_norm,
  genes,
  top_n = c(50, 100, 200, 300),
  method_order = NULL,
  method_colors = NULL,
  gene_id = "rownames",
  theme_name = "classic",
  gene_set_name = "Gene set",
  title = NULL,
  subtitle = NULL,
  alpha = 1,
  point_size = 2,
  line_width = 1,
  line_type = "solid",
  show_points = TRUE,
  show_legend = TRUE,
  percentage = FALSE
)
```

Arguments

`svg_norm`	Named list of SVG results.
`genes`	Character vector containing genes of interest.
`top_n`	Numeric vector of top-N SVG thresholds.
`method_order`	Optional normalisation method order.
`method_colors`	Optional named vector of method colours.
`gene_id`	Gene identifier used in SVG results.
`theme_name`	Plot theme.
`gene_set_name`	Name of the gene set.
`title`	Optional plot title.
`subtitle`	Optional plot subtitle.
`alpha`	Transparency of points and lines.

point_size	Point size.
line_width	Line width.
line_type	Line type.
show_points	Logical. Display points at each top-N value.
show_legend	Logical. Display the method legend.
percentage	Logical. Display y-axis as percentages.

Details

The gene set may contain housekeeping genes, negative-control genes, marker genes, or any other user-defined gene set.

Value

List containing data and plot.

Examples

```
data(svg_example)

genes <- head(rownames(svg_example[[1]]), 20)

p <- plot_svg_topn_gene_set(
  svg_norm = svg_example,
  genes = genes,
  top_n = c(10, 20, 50),
  gene_set_name = "Example genes"
)

p$plot
```

---

plot_umap_comparison *Plot UMAP comparisons across normalisation methods*

Description

Visualise previously calculated UMAP embeddings using one or multiple colData annotations without recalculating PCA or UMAP.

Usage

```
plot_umap_comparison(
  umap_data,
  colour_by = "region",
  point_size = 0.58,
  alpha = 0.6,
  ncol = 3,
  theme_name = "classic"
)
```

Arguments

umap_data	Data frame returned by get_umap_data().
colour_by	Character vector containing one or more annotation columns.
point_size	Point size for UMAP plots.
alpha	Point transparency.
ncol	Number of columns for single-annotation facets.
theme_name	Plot theme.

Details

A single annotation uses facet_wrap by method. Multiple annotations use facet_grid with methods in rows and annotations in columns.

Value

A list containing plotting data and ggplot.

Examples

```
data(norm_example)

umap_data <- get_umap_data(
  norm_example,
  methods = NULL,
  prop_spots = 1,
  seed = 123,
  assay_name = "logcounts",
  n_pcs = 30,
  n_neighbors = 15
)

p_umap_layer <- plot_umap_comparison(
  umap_data,
  colour_by = "cell_type",
  point_size = 0.58,
  alpha = 0.6,
  ncol = 3,
  theme_name = "classic"
)

p_umap_layer$plot
```

rank_normalization_methods *Rank normalisation methods by overall clustering agreement*

Description

Methods are ranked using the Overall score, calculated as the mean of NMI, HOM and COM.

Usage

```
rank_normalization_methods(metrics_df)
```

Arguments

metrics_df	Data frame from calculate_clustering_metrics().

Value

Data frame containing Rank, method and Overall.

Examples

```
data(norm_example)

banksy_norm <- run_banksy_individual(
  norm_list = norm_example,
  seed = 123
)

metrics_df <- calculate_clustering_metrics(
  norm_list = banksy_norm,
  ground_truth_col = "layer_guess"
)

ranking_df <- rank_normalization_methods(
  metrics_df = metrics_df
)

ranking_df
```

run_annotation *Run cell-type annotation*

Description

Run cell-type annotation for a named list of normalised SpatialExperiment objects using SingleR.

Usage

```
run_annotation(
  norm_list,
  ref,
  labels,
  method = "SingleR",
  label_name = "SingleR_label",
  pruned_label_name = "SingleR_pruned_label",
  ...
)
```

Arguments

norm_list	Named list of normalised SpatialExperiment objects.
ref	Reference SingleCellExperiment or SummarizedExperiment object.
labels	Character vector containing reference cell-type labels. Length must equal the number of columns in ‘ref‘.
method	Cell-type annotation method. Currently only ‘"SingleR"‘ is supported.
label_name	Name of the colData column used to store predicted labels.
pruned_label_name	Name of the colData column used to store pruned SingleR labels.
...	Additional parameters passed to ‘SingleR::SingleR()‘.

Details

Predicted labels are added to the colData of each SpatialExperiment object so that the same objects can be used in downstream plotting and benchmarking functions.

Value

Named list of annotated SpatialExperiment objects.

Examples

```
data(norm_example)

# Use a subset of cells as an illustrative reference
ref <- norm_example$None[, seq_len(100)]
```

```
labels <- as.character(ref$cell_type)

# Annotate the remaining cells
query <- lapply(
  norm_example,
  function(spe) spe[, seq.int(101, ncol(spe))]
)

annotated_norm <- run_annotation(
  norm_list = query,
  ref = ref,
  labels = labels,
  method = "SingleR",
  label_name = "SingleR_label",
  pruned_label_name = "SingleR_pruned_label",
  de.method = "classic"
)

table(annotated_norm$None$SingleR_label)
```

`run_banksy_gene_selection` *Run BANKSY with different gene-selection strategies*

Description

Runs BANKSY clustering independently for each normalisation method using all genes, highly variable genes (HVGs), or spatially variable genes (SVGs).

Usage

```
run_banksy_gene_selection(
  norm_list,
  svg_list = NULL,
  methods = names(norm_list),
  gene_selection = c("all", "hvg500", "hvg1000", "svg100", "svg200", "svg500"),
  rank_col = "rank",
  assay_name = "logcounts",
  cluster_col = "banksy_cluster",
  k_geom = 15,
  lambda = 0.8,
  resolution = 0.2,
  seed = 123
)
```

Arguments

`norm_list`	Named list of SpatialExperiment objects, or named lists of SpatialExperiment slices.

svg_list	Optional named list of SVG results, corresponding to the normalisation methods.
methods	Normalisation methods to analyse.
gene_selection	Character vector containing any of "all", "hvg500", "hvg1000", "svg100", "svg200", "svg500".
rank_col	Column containing SVG rankings.
assay_name	Assay used for clustering and HVG selection.
cluster_col	Column for BANKSY cluster assignments.
k_geom	Number of spatial neighbours.
lambda	BANKSY spatial weighting parameter.
resolution	Clustering resolution.
seed	Random seed.

Details

HVGs are selected by expression variance within each normalisation method. SVGs are selected using the gene rankings supplied in svg_list.

Value

Nested list indexed by normalisation method, then gene-selection strategy. Each entry contains a clustered SpatialExperiment object or named slice list.

Examples

```
data(norm_example)

banksy_genes <- run_banksy_gene_selection(
  norm_list = norm_example,
  methods = names(norm_example),
  gene_selection = "all",
  assay_name = "logcounts",
  k_geom = 15,
  lambda = 0.8,
  resolution = 0.2,
  seed = 123
)
```

---

run_banksy_individual *Run BANKSY clustering independently for each slice*

Description

Run BANKSY clustering independently across normalisation methods and spatial slices.

Usage

```
run_banksy_individual(
  norm_list,
  methods = names(norm_list),
  assay_name = "logcounts",
  cluster_col = "banksy_cluster",
  k_geom = 15,
  lambda = 0.8,
  resolution = 0.2,
  seed = 123
)
```

Arguments

norm_list	Named list containing normalised SpatialExperiment objects.
methods	Optional character vector specifying the normalisation methods to analyse.
assay_name	Name of the assay used by BANKSY.
cluster_col	Name of the colData column used to store BANKSY cluster assignments.
k_geom	Number of spatial neighbours used by BANKSY.
lambda	BANKSY spatial weighting parameter.
resolution	Resolution parameter used for clustering.
seed	Random seed for reproducibility.

Details

BANKSY cluster assignments are added to the colData of each SpatialExperiment object so that the same objects can be used in downstream clustering evaluation, plotting, differential expression, and GO enrichment analyses.

The function supports either:

- a named list of SpatialExperiment objects, or
- a named list where each method contains a named list of SpatialExperiment objects corresponding to multiple slices.

Value

Named list with the same structure as ‘norm_list‘. BANKSY cluster assignments are added to the colData of each SpatialExperiment object.

Examples

```
data(norm_example)

banksy_norm <- run_banksy_individual(
  norm_list = norm_example,
  methods = names(norm_example),
  assay_name = "logcounts",
```

```
  cluster_col = "banksy_cluster",
  k_geom = 15,
  lambda = 0.8,
  resolution = 0.2,
  seed = 123
)
```

`run_ccc` *Run CellChat across normalization methods*

Description

Runs CellChat cell-cell communication analysis across multiple normalized versions of the same SpatialExperiment object.

Usage

```
run_ccc(
  spelist,
  celltype_col,
  database,
  spatial_factors,
  interaction_range,
  contact_range,
  assay_name = "logcounts",
  nboot = 100,
  min_cells = 10,
  trim = 0.1,
  scale_distance = 0.1,
  variable_both = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`spelist`	Named list of SpatialExperiment objects. Names should correspond to normalization methods.
`celltype_col`	Character. Column in colData(spe) containing cell-type labels.
`database`	CellChat database object, e.g. CellChatDB.mouse.
`spatial_factors`	Data frame containing CellChat spatial factors, typically ratio and tol.
`interaction_range`	Numeric. Maximum spatial interaction distance.
`contact_range`	Numeric. Contact-dependent interaction distance.
`assay_name`	Character. Assay containing normalized expression. Default is "logcounts".

nboot	Integer. Number of bootstrap iterations.
min_cells	Integer. Minimum number of spots/cells per group.
trim	Numeric. Trimming proportion for truncated mean.
scale_distance	Numeric. Distance scaling used by CellChat.
variable_both	Logical. Passed to identifyOverExpressedInteractions().
verbose	Logical. Print progress messages.
...	Additional arguments.

Value

Named list containing CellChat results for each method.

Examples

```
data(norm_ccc_example)

data("CellChatDB.mouse", package = "CellChat")

ccc_results <- run_ccc(
  spelist = norm_ccc_example,
  celltype_col = "cell_type",
  database = CellChat::CellChatDB.mouse,
  spatial_factors = data.frame(
    ratio = 0.5,
    tol = 12.5
  ),
  interaction_range = 250,
  contact_range = 25,
  assay_name = "logcounts",
  nboot = 10,
  min_cells = 5,
  trim = 0.1,
  scale_distance = 0.1,
  variable_both = FALSE,
  verbose = FALSE
)
```

---

run_svg *Run spatially variable gene detection*

Description

Run spatially variable gene detection

Usage

```
run_svg(norm_list, method = "nnSVG", ...)
```

Arguments

norm_list	Named list of normalised SpatialExperiment objects.
method	SVG detection method.
...	Additional parameters passed to the SVG method.

Value

Named list containing SVG results.

Examples

```
data(norm_example)

svg_results <- run_svg(
  norm_list = norm_example,
  method = "nnSVG",
  n_threads = 1,
  verbose = FALSE
)

names(svg_results)
head(as.data.frame(
  SummarizedExperiment::rowData(svg_results[[1]])
))
```

---

spe_example *Example spatial transcriptomics dataset*

Description

A subset of the publicly available STARmap PLUS mouse brain spatial transcriptomics dataset, obtained using the 'STARmapPLUS_mouseBrain()' function from the Bioconductor package 'STexampleData'.

Format

A SpatialExperiment object.

Details

The dataset was generated using STARmap PLUS, an imaging-based spatial transcriptomics technology.

A random subset of 100 genes and 200 cells was selected using seed 123. Genes and cells with zero total counts were removed.

The dataset contains spatial coordinates, raw gene counts, and original cell-type and tissue-region annotations.

Two additional annotation columns were created: 'cell_type', containing molecular cell-type annotations, and 'layer_guess', containing molecular tissue-region annotations.

Source

Bioconductor STexampleData package: https://bioconductor.org/packages/STexampleData/

`svg_example` *Example spatially variable gene results*

Description

Precomputed spatially variable gene detection results generated from ‘norm_example‘ using the STBench ‘run_svg()‘ function with the nnSVG method.

Format

A named list of SpatialExperiment objects containing nnSVG results.

Details

Results are provided for the normalisation methods ‘None‘ and ‘SpaNorm‘.

This dataset allows users to evaluate and visualise spatially variable gene statistics and rankings without repeating the SVG detection procedure.

Source

Derived from the STARmap PLUS mouse brain dataset available through the Bioconductor STexampleData package.

Index

add_qcs, 3

calculate_annotation_agreement, 4
calculate_clustering_metrics, 5
calculate_gene_selection_metrics, 6
calculate_go, 8

extract_svg_gene_set, 10
extract_svg_stats, 11

filter_qc, 12

get_umap_data, 13

norm_ccc_example, 14
norm_example, 15
norm_giotto, 15
norm_none, 16
norm_scran, 16
norm_sct, 17
norm_spanorm, 17
normalize_spe, 18

plot_annotation_agreement, 19
plot_ccc_circle, 20
plot_ccc_lr, 21
plot_ccc_pathways, 22
plot_ccc_sender_receiver, 24
plot_celltype_proportion, 25
plot_clustering_metrics, 26
plot_gene_selection_metrics, 27
plot_go, 29
plot_go_genes, 30
plot_library_size_comparison, 32
plot_marker_auc, 33
plot_marker_expression, 35
plot_marker_ttest, 36
plot_normalization_ranking, 38
plot_qc, 39
plot_qc_cutoffs, 40
plot_qc_removed_spots, 41
plot_spatial_annotation, 42
plot_spatial_clustering, 44
plot_spatial_expression, 45
plot_svg_gene_set_distribution, 47
plot_svg_gene_sets, 48
plot_svg_rank, 50
plot_svg_rank_correlation, 51
plot_svg_rank_vs_stat, 52
plot_svg_stat, 54
plot_svg_topn_gene_set, 55
plot_umap_comparison, 56

rank_normalization_methods, 58
run_annotation, 59
run_banksy_gene_selection, 60
run_banksy_individual, 61
run_ccc, 63
run_svg, 64

spe_example, 65
svg_example, 66