Package 'celliverse'

September 12, 2026

Type Package

Title An Ecosystem for Exploring the Universe of Single-Cell Data

Language en-US

Version 0.0.2

Description Contains functions for single-cell RNA sequencing data analysis. It provides core functionalities for clustering cells, identifying markers for predefined clusters, performing subclustering of major cell populations, and discovering markers within custom-selected subsets of cells. 'CelliVerse' also includes methods for analyzing cluster similarity and generating intuitive visualizations. Designed to be independent of library size and other sample- or cell-level confounding effects, 'CelliVerse' ensures reliable and interpretable results across a wide range of datasets.
Fred Viole and David Nawrocki (2013, ISBN:1490523995).
Csardi G, Nepusz T (2006). ``The 'igraph' software package for complex network research." InterJournal, Complex Systems, 1695.
Adopted algorithms and sources are referenced in function document.

Imports Matrix, Seurat, SummarizedExperiment, scales, ggplot2, magrittr, igraph, cli, dplyr, data.table, stringr, tidyr, RColorBrewer, ggnewscale, patchwork, tidyselect, methods, Rcpp

LinkingTo Rcpp, RcppEigen

Suggests R.rsp, htmltools, knitr, rmarkdown, plumber, jsonlite, httr2, processx, later, promises, callr, httpuv, openssl, fs, svglite, zip, withr, SeuratObject, ComplexHeatmap, hdf5r, rstudioapi

Depends R (>= 4.1.0)

URL https://github.com/asalavaty/celliverse, https://asalavaty.github.io/celliverse/

BugReports https://github.com/asalavaty/celliverse/issues

License GPL-3

Encoding UTF-8

LazyData true

LazyDataCompression bzip2

RoxygenNote 7.3.3

VignetteBuilder R.rsp

NeedsCompilation yes

Author Adrian Salavaty [aut, cre] (ORCID:
<https://orcid.org/0000-0001-9320-5889>)

Maintainer Adrian Salavaty <abbas.salavaty@gmail.com>

Repository CRAN

Date/Publication 2026-09-12 13:40:14 UTC

Contents

addClustoData *Add ClustoCell cluster annotations to a Seurat or SingleCellExperiment object*

Description

Adds major cluster and/or sub-cluster labels stored in a `ClustoCell` object to the cell-level metadata of a Seurat or SingleCellExperiment object.

Usage

```
addClustoData(
  obj,
  clustoCell,
  add_major_clusters = TRUE,
  add_sub_clusters = TRUE,
  major_cluster_name = "ClustoCell_Clusters",
  sub_cluster_name = "ClustoCell_SubClusters"
)
```

Arguments

`obj`	An object of class `Seurat` or `SingleCellExperiment`.
`clustoCell`	An object of class `ClustoCell`, generated via `clustoCell()` or `markoClust()`.
`add_major_clusters`	Logical; whether to add major cluster labels to the metadata of `obj`.
`add_sub_clusters`	Logical; whether to add sub-cluster labels to the metadata of `obj`.
`major_cluster_name`	Character; name of the metadata column to store major cluster labels.
`sub_cluster_name`	Character; name of the metadata column to store sub-cluster labels.

Details

This function transfers clustering results obtained using `clustoCell()` or `markoClust()` into an existing single-cell object by appending cluster labels as metadata columns. Major clusters and sub-clusters can be added independently and assigned custom column names.

Value

The input object `obj` with additional metadata columns containing ClustoCell cluster annotations.

See Also

`clustoCell`, `markoClust`

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

pbmc_small_cc <- clustoCell(
  data = pbmc_small,
  identify_subclusters = TRUE,
  num_threads = 1,
  verbose = FALSE
)

pbmc_small <- addClustoData(
  obj = pbmc_small,
  clustoCell = pbmc_small_cc,
  add_major_clusters = TRUE,
  add_sub_clusters = TRUE
)
```

`addTypoData` *Add TypoClust cell type annotations to a single-cell object*

Description

Adds inferred cell type annotations from a `TypoClust` object to the metadata of a Seurat or SingleCellExperiment object.

Usage

```
addTypoData(
  obj,
  typoClust,
  clusters,
  rank_thresh = 1,
  refine = TRUE,
  refine_thresh = 1,
  outNames = NULL
)
```

Arguments

`obj`	An object of class `Seurat` or `SingleCellExperiment`.
`typoClust`	An object of class `TypoClust`, generated using `typoClust()`.
`clusters`	Character vector; names of metadata columns in `obj` defining clusters or cell subsets to which cell types will be assigned.
`rank_thresh`	Integer; the top N ranked cell types to add for each cluster, stored as separate metadata columns.

refine	Logical; whether to refine inferred cell types by traversing deeper levels of the cell type hierarchy.
refine_thresh	Integer; depth of (lexical) hierarchical traversal for refinement. Ignored if refine = FALSE.
outNames	Character vector; names of output metadata columns. If NULL, defaults to paste0(clusters, "_Celltype").

Details

Cell type labels are assigned to specified cluster or subset columns and appended as new metadata columns. Multiple ranked cell types can be added, and hierarchical refinement can be applied to obtain more specific cell type annotations.

Value

The input object obj with additional metadata columns containing inferred cell type annotations.

See Also

typoClust, typoClustVis

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

cc <- clustoCell(
  data = pbmc_small,
  identify_subclusters = FALSE,
  num_threads = 1,
  verbose = FALSE
)

pbmc_small <- addClustoData(
  obj = pbmc_small,
  clustoCell = cc,
  add_major_clusters = TRUE,
  add_sub_clusters = FALSE
)

tc <- typoClust(
  objects = list(cc),
  tissue = "Blood",
  condition = "Healthy",
  use_neg_markers = FALSE,
  thresh = 10,
  mode = "markerDB",
  species = "human",
  verbose = FALSE
)

pbmc_small <- addTypoData(
```

```
  obj = pbmc_small,
  typoClust = tc,
  clusters = "ClustoCell_Clusters",
  refine = FALSE
)
```

---

ceLLMarkup *LLM-based cell-type annotation of clusters or marker sets (ceLLMarkup)*

Description

Annotates cell clusters, sub-clusters, or arbitrary marker sets by asking a large language model to name the most likely cell type for each set, given its marker genes and an optional tissue/condition/species context. Returns ranked candidate cell types per set in a `TypoClust`-compatible object.

Exactly one of `marker_set_list`, `panels`, or `seuratClusters` must be provided.

Usage

```
ceLLMarkup(
  sample_source = NULL,
  feature_type = "gene",
  marker_set_list = NULL,
  panels = NULL,
  seuratClusters = NULL,
  padj = 0.05,
  logFC = NULL,
  tissue = NULL,
  condition = NULL,
  species = c("human", "mouse"),
  provider = "ollama",
  model,
  api_key = NULL,
  host = NULL,
  temperature = 0.2,
  top_k = 3L,
  n_markers = 25L,
  max_retries = 1L,
  inherit_major_clusters = TRUE,
  verbose = TRUE
)
```

Arguments

`sample_source`	Free-text description of the sample origin shown to the model (e.g. `"human peripheral blood"`). Improves accuracy.

feature_type Feature type of the markers (e.g. "gene", "protein"). Default "gene".

marker_set_list
: Named list of marker data.frames, one per set. Each data.frame should have feature names in the first column and purity or fold changes in the second. Positive markers (or up-regulated features) are recommended.

panels Named list of marker panels: either a named list of character vectors (positive markers per set), or a list with pos_panels and/or neg_panels elements (each a named list of character vectors). This is the input typoClust(mode = "ceLLMarkup") uses.

seuratClusters A data.frame obtained from Seurat's FindAllMarkers() (must have avg_log2FC, p_val_adj, cluster, gene columns). Filtered to up-regulated markers by padj/logFC.

padj Adjusted p-value threshold for filtering seuratClusters (default 0.05; NULL disables).

logFC log-fold-change threshold for filtering seuratClusters (default NULL = no extra filter; only up-regulated genes kept).

tissue Optional tissue context (e.g. "blood").

condition Optional condition context (e.g. "healthy").

species Species for gene-symbol interpretation: "human", "mouse", or your desired species name.

provider LLM provider: one of "ollama", "lmstudio", "openai", "anthropic", "gemini", "deepseek", "groq", "openrouter", "cerebras".

model Model id for the provider (e.g. "qwen3:8b" for Ollama, "qwen/qwen3-8b" for LM Studio, "gpt-4o-mini" for OpenAI, "anthropic/claude-3-haiku" for OpenRouter).

api_key API key for cloud providers. Not needed for ollama/lmstudio. If NULL, falls back to the standard environment variable for the provider (e.g. OPENAI_API_KEY, OPENROUTER_API_KEY).

host Base URL for local providers. Defaults to http://localhost:11434 (Ollama) or http://localhost:1234/v1 (LM Studio). Ignored for cloud providers.

temperature Sampling temperature (default 0.2).

top_k Number of ranked candidate cell types per set (default 3).

n_markers Maximum number of markers per set shown to the model (default 25).

max_retries Retries per set when the model returns an unparseable or empty answer (default 1).

inherit_major_clusters
: Logical; whether a sub-cluster should be annotated *within the identity of its own major cluster*. Default TRUE.

When FALSE, all sets are annotated together in a single request and the function behaves exactly as it did before this argument existed.

When TRUE, the set names are inspected for a major/sub-cluster hierarchy — a set "C1" is the parent of "C1-Sub1", "C1-Sub2", and so on. If both levels are present, annotation runs in two stages. First the major clusters are annotated from *their own* markers. Then, for each major cluster separately, the model is

told what that cluster was identified as and asked which specific subtype or state of *that* cell type each of its sub-clusters represents, given the *sub-cluster's own* markers.

One request is issued per major cluster in the second stage. That is what keeps each sub-cluster tied to its own parent: a single combined request could not carry a different parent identity for each set. Cost is therefore one request plus one per major cluster that has sub-clusters, rather than one in total. `metadata$inheritance` records which parent was used for each sub-cluster and what it was called.

`verbose` Show progress messages (default `TRUE`).

Value

An object of class `TypoClust`: a list with `cell_types` (named list of ranked annotation data.frames) and `metadata`.

Examples

```
## Not run:
# Requires an externally running Ollama server and an installed
# qwen3:8b model.
markers <- list(
  Cluster1 = c("CD3D", "CD3E", "TRBC1", "IL7R", "LTB"),
  Cluster2 = c("MS4A1", "CD79A", "CD37", "CD74", "HLA-DRA")
)

annotations <- ceLLMarkup(
  panels = markers,
  sample_source = "human peripheral blood",
  feature_type = "gene",
  tissue = "Blood",
  condition = "Healthy",
  species = "human",
  provider = "ollama",
  model = "qwen3:8b",
  top_k = 3
)

## End(Not run)
```

`clustoCell` *Clustering and marker discovery for single-cell data using EWCSR-based similarity*

Description

Performs unsupervised clustering of single-cell data using expression-weighted centered scaled ranks (EWCSR), followed by identification of cluster-specific markers and optional sub-clustering. The function supports direct clustering on full datasets or scalable analysis via data sketching with subsequent label transfer to the full dataset.

Usage

```
clustoCell(
  data,
  assay = "RNA",
  layer = "counts",
  norm_assay = "RNA",
  norm_layer = "data",
  log1p = TRUE,
  subset_to_HVG = FALSE,
  hvg_selection.method = c("vst", "mean.var.plot", "dispersion"),
  hvg_var_thresh = 1,
  high_quantile = 0.25,
  low_quantile = 0.25,
  gini_thresh = 0.5,
  identify_subclusters = TRUE,
  sketch = FALSE,
  sketch_ncells = 5000L,
 label_transfer_method = c("ewcsr-cor", "seurat-project", "ewcsr-red-cor", "seurat-knn"),
  sketch_pca_dims = 30,
  refine_transferred_subClusters = FALSE,
  noise_feature_thresh = 4,
  random_marker_thresh = 5,
  mr_thresh = NULL,
  isolated_cluster_thresh = 5,
  leiden_obj_function = c("modularity", "CPM"),
  leiden_resolution = 1,
  leiden_n_iterations = 5,
  subcluster_resolution_weight = 0.75,
  num_threads = -1,
  seed = 121,
  verbose = TRUE
)
```

Arguments

data	Either a Seurat object or a numeric matrix with features (genes) as rows and cells as columns. Recommended to provide at least library-size normalized data when subset_to_HVG = TRUE.
assay	Character string specifying the assay to use for clustering.
layer	Character string specifying the layer of assay to use (e.g., "counts" or a normalized layer).
norm_assay	Character string specifying the assay used for selecting highly variable genes (HVGs).
norm_layer	Character string specifying the normalized layer used for HVG detection.
log1p	Logical; whether to apply log1p transformation to the input data. It is recommended to set this argument to TRUE (default) if the data is not already on a log scale.

`subset_to_HVG` Logical; whether to restrict the analysis to highly variable genes (HVGs) or use all features.

`hvg_selection.method`
Character string specifying the HVG selection strategy. One of `"vst"`, `"mean.var.plot"`, or `"dispersion"`.

`hvg_var_thresh` Numeric; variance threshold (in standard deviations above expected technical noise) for selecting HVGs.

`high_quantile` Numeric; upper quantile threshold used for identifying highly positive EWCSR values.

`low_quantile` Numeric; lower quantile threshold used for identifying highly negative EWCSR values.

`gini_thresh` Numeric; Gini coefficient threshold for identifying non-specific (global) markers.

`identify_subclusters`
Logical; whether to perform sub-clustering within major clusters.

`sketch` Logical; whether to apply Seurat's uniform sketching strategy to subsample cells prior to clustering. Recommended for very large datasets.

`sketch_ncells` Integer; number of cells to sample during sketching. Must be smaller than the total number of cells in the dataset.

`label_transfer_method`
Character string specifying the strategy used to transfer cluster labels from the sketched dataset back to the full dataset. Only used when `sketch = TRUE`. Options include:

- `"ewcsr-cor"`: Transfers labels by computing correlations between EWCSR profiles of query cells and EWCSR centroids of sketched clusters in the full feature space.
- `"seurat-project"`: Uses Seurat's `ProjectData()` workflow to transfer labels by projecting the full expression dataset (raw counts, log-normalized, or SCT-normalized) onto the low-dimensional embedding learned from the sketched dataset.
- `"ewcsr-red-cor"`: Similar to `"ewcsr-cor"`, but correlations are computed in a reduced dimensional space (PCA embedding).
- `"seurat-knn"`: Uses Seurat's `FindTransferAnchors()` and `TransferData()` workflow to transfer labels from the sketched dataset to the full dataset using nearest-neighbor matching. Supports standard Seurat normalization workflows (e.g., LogNormalize and SCTransform).

`sketch_pca_dims`
Integer; number of PCA dimensions used during label transfer. Only applicable for `"ewcsr-red-cor"` and `"seurat-knn"`.

`refine_transferred_subClusters`
Logical; whether to re-evaluate and refine transferred sub-cluster labels after label transfer.

`noise_feature_thresh`
Integer; features expressed in fewer than this number of cells are treated as noise and excluded.

`random_marker_thresh`
: Integer; markers detected in fewer than this number of cells are discarded.

`mr_thresh`
: Numeric; threshold applied to mutual rank similarity. If `NULL`, defaults to `sqrt(number of cells)`.

`isolated_cluster_thresh`
: Integer; clusters with fewer than this number of cells are treated as isolated.

`leiden_obj_function`
: Character string specifying the Leiden objective function. One of `"modularity"` or `"CPM"`.

`leiden_resolution`
: Numeric; resolution parameter controlling cluster granularity.

`leiden_n_iterations`
: Integer; number of Leiden algorithm iterations.

`subcluster_resolution_weight`
: Numeric; multiplier applied to `leiden_resolution` for sub-cluster detection.

`num_threads`
: Integer; number of CPU threads to use. Default `-1` uses all available cores.

`seed`
: Integer; random seed for reproducibility.

`verbose`
: Logical; whether to print progress messages.

Details

If `sketch = TRUE`, a representative subset of cells is sampled from the input data and used for clustering. Labels are transferred back to the full dataset using the method specified by `label_transfer_method`. The `"seurat-project"` and `"seurat-knn"` methods operate on the original expression matrix rather than the EWCSR representation. The supplied expression data may consist of raw counts, log-normalized expression values, or SCTransform-normalized data, provided that the sketched and full datasets were processed using the same normalization strategy. These methods leverage Seurat's native label transfer workflows (`ProjectData`, `FindTransferAnchors`, and `TransferData`) and do not require integer count matrices.

Value

An object of class `ClustoCell` containing:

- major and sub-cluster assignments
- EWCSR-transformed data
- cluster- and subcluster-specific marker tables
- similarity and mutual-rank matrices

See Also

`typoClust`, `markoClust`

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

cc <- clustoCell(
  data = pbmc_small,
  identify_subclusters = FALSE,
  num_threads = 1,
  verbose = FALSE
)
```

`clustoCell_TransferLabel` *Transfer ClustoCell cluster labels to a full-resolution dataset*

Description

Transfers major and sub-cluster labels from a `ClustoCell` object generated on a sketched (subsampled) dataset to a full-resolution dataset using EWCSR-based correlation or Seurat-based projection and anchor transfer strategies.

Usage

```
clustoCell_TransferLabel(
  clustoCell,
  query_ewcsr_mat = NULL,
  query_expr_mat = NULL,
  assay = "RNA",
  layer = "counts",
  method = c("ewcsr-cor", "seurat-project", "ewcsr-red-cor", "seurat-knn"),
  dims = 30,
  num_threads = -1,
  inherit_major_clusters = TRUE,
  seed = 121,
  verbose = TRUE
)
```

Arguments

`clustoCell`	An object of class `ClustoCell` obtained by running `clustoCell()` on a sketched dataset.
`query_ewcsr_mat`	A sparse matrix (`dgCMatrix`) containing EWCSR values for the full dataset. Required for EWCSR-based methods and ignored when `method = "seurat-knn"` or `method = "seurat-project"`.
`query_expr_mat`	Either a `Seurat` object or a sparse count matrix (`dgCMatrix`) for the full dataset. Required for `method = "seurat-knn"` or `method = "seurat-project"` and optional otherwise.

`assay` Character string specifying the assay used in `query_expr_mat` when a `Seurat` object is provided.

`layer` Character string specifying the layer of `assay` to use (e.g., `"counts"`).

`method` Character string specifying the label transfer strategy. Options include:

- `"ewcsr-cor"`: Transfers labels by computing correlations between EWCSR profiles of query cells and EWCSR centroids of sketched clusters in the full feature space.
- `"seurat-project"`: Uses Seurat's `ProjectData()` workflow to transfer labels by projecting the full expression dataset (raw counts, log-normalized, or SCT-normalized) onto the low-dimensional embedding learned from the sketched dataset.
- `"ewcsr-red-cor"`: Similar to `"ewcsr-cor"`, but correlations are computed in a reduced dimensional space (PCA embedding).
- `"seurat-knn"`: Uses Seurat's `FindTransferAnchors()` and `TransferData()` workflow to transfer labels from the sketched dataset to the full dataset using nearest-neighbor matching. Supports standard Seurat normalization workflows (e.g., LogNormalize and SCTransform).

`dims` Integer; number of dimensions used during sketching or PCA-based label transfer.

`num_threads` Integer; number of CPU threads to use. Default `-1` uses all available cores.

`inherit_major_clusters` Logical; whether to restrict sub-cluster label transfer within inherited major clusters when such labels are available.

`seed` Integer; random seed for reproducibility.

`verbose` Logical; whether to print progress messages.

Details

The `"seurat-project"` and `"seurat-knn"` methods operate on the original expression matrix (‘query_expr_mat‘) rather than the EWCSR representation. The supplied expression data may consist of raw counts, log-normalized expression values, or SCTransform-normalized data, provided that the sketched and full datasets were processed using the same normalization strategy. These methods leverage Seurat's native label transfer workflows (`ProjectData`, `FindTransferAnchors`, and `TransferData`) and do not require integer count matrices.

Value

An updated object of class `ClustoCell` containing transferred major and sub-cluster labels for the full dataset.

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

reference_cells <- colnames(pbmc_small)[1:60]

pbmc_reference <- pbmc_small[, reference_cells]
```

```
cc_reference <- clustoCell(
  data = pbmc_reference,
  identify_subclusters = FALSE,
  num_threads = 1,
  verbose = FALSE
)

full_counts <- SeuratObject::LayerData(
  pbmc_small,
  assay = "RNA",
  layer = "counts"
)

full_ewcsr <- ewcsr.sparse(full_counts, num_threads = 1)

cc_full <- clustoCell_TransferLabel(
  clustoCell = cc_reference,
  query_ewcsr_mat = full_ewcsr,
  method = "ewcsr-cor",
  num_threads = 1,
  verbose = FALSE
)
```

`ewcsr.sparse` *Compute expression-weighted centered scaled ranks (EWCSR)*

Description

Computes expression-weighted centered scaled ranks for a sparse or dense expression matrix, calculated per column (cell).

Usage

```
ewcsr.sparse(mat, num_threads = -1L)
```

Arguments

`mat`	A matrix with features (genes) as rows and cells or samples as columns.
`num_threads`	Integer; number of threads to use. The default is `-1` which uses all available cores.

Details

EWCSR transformation emphasizes relatively high and low expression features within each cell while accounting for expression magnitude. The output matrix retains the same dimensions as the input.

Value

A matrix of EWCSR-transformed values with the same dimensions as `mat`.

See Also

`gini.ewcsr.fs`, `markoCell`

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

mat <- SeuratObject::LayerData(
  pbmc_small,
  assay = "RNA",
  layer = "counts"
)

ewcsr_mat <- ewcsr.sparse(mat, num_threads = 1)
```

---

featureInspect *Inspect the Membership of Features Across ClustoCell Results*

Description

`featureInspect()` searches one or more features across all marker collections contained within a `ClustoCell` object, including global feature collections, cross-cluster markers, major cluster-specific markers, and sub-cluster-specific markers. All matches are returned in a single long-format data frame containing the feature level, membership, marker type, Gini score, purity, and rank. Optionally, a publication-quality **ggplot2** visualisation summarising feature memberships can also be generated.

Usage

```
featureInspect(
  clustoCell,
  features,
  level = NULL,
  type = NULL,
  sort_by = c("input", "rank", "gini"),
  plot = FALSE,
  title = NULL,
  subtitle = NULL,
  tag = NULL,
  nrow_panels = NULL,
  dotsize = 3,
  show_purity = TRUE,
  class_palette = NULL,
```

```
  color_low = "steelblue",
  color_high = "firebrick",
  panel_border_color = "black",
  panel_border_size = 0.5,
  axis_text_size = 8,
  axis_title_size = 9,
  plot_margin_right = 10,
  xlab = "Rank",
  ylab = "Feature",
  show_legend = TRUE,
  legend_position = "right",
  legend_box = "vertical",
  legend_box_just = "left"
)
```

Arguments

clustoCell	An object of class ClustoCell obtained using clustoCell or markoClust.
features	A character vector of feature names (e.g. gene symbols) to inspect across the ClustoCell object.
level	Character vector specifying which hierarchical level(s) to include in the output. One or more of "Global", "Cross-cluster", "Major cluster", and "Sub-cluster". If NULL (default), results from all levels are returned. If none of the queried features are found at the specified level(s), a zero-row data.frame is returned (with a warning) rather than an error.
type	Character vector specifying which marker type(s) to include in the output. One or more of "Positive", "Negative", "Medium", "Pure Ranked", "Pure High", "Pure Medium", and "Pure". If NULL (default), results of all types are returned. The value "Pure" is a convenience shorthand that expands to "Pure Ranked", "Pure High", and "Pure Medium" simultaneously, including all global feature categories. Individual pure types (e.g. "Pure High") can also be specified directly. The level and type filters are applied independently; if their combination yields no matching rows, a zero-row data.frame is returned (with a warning) rather than an error.
sort_by	Character string specifying how to order the rows of the output table. One of: "input" (Default) Rows follow the order of features as supplied by the user, then by level (Global → Cross-cluster → Major cluster → Sub-cluster). "rank" Ascending Rank (rows with NA rank appear last), then by input order within ties. "gini" Descending Gini_Score (rows with NA Gini score appear last), then by input order within ties.
plot	Logical. If FALSE (default), a data.frame is returned. If TRUE, a named list with elements $table and $plot is returned.
title	Character. Plot title. Ignored when plot = FALSE.
subtitle	Character. Plot subtitle. Ignored when plot = FALSE.
tag	Character. Plot tag (e.g. panel label). Ignored when plot = FALSE.

`nrow_panels` Integer. Number of rows used when faceting the plot by `Membership`. If `NULL` (default), the number of rows is determined automatically by `facet_wrap`. Ignored when `plot = FALSE`.

`dotsize` Numeric. Controls the size range of the dots in the plot. The actual `size` aesthetic is scaled between `dotsize * 0.4` and `dotsize * 1.8`. Default is `3`. Ignored when `plot = FALSE`.

`show_purity` Logical. If `TRUE` (default), dot colour encodes `Purity` via a continuous gradient. If `FALSE`, dot colour encodes `Type` as a discrete scale. Ignored when `plot = FALSE`.

`class_palette` Optional. Only used when `show_purity = FALSE`. Specifies the colour scale for `Type`. Can be one of:

- A **ggplot2** scale object (e.g. `ggplot2::scale_colour_brewer()`).
- A named or unnamed character vector of colours (e.g. `c("red", "blue", "green")`), which is passed to `scale_colour_manual`.
- `NULL` (default): the default **ggplot2** discrete colour scale is used.

Ignored when `plot = FALSE`.

`color_low` Character. The low-end colour of the continuous Purity gradient. Default is `"steelblue"`. Ignored when `show_purity = FALSE` or `plot = FALSE`.

`color_high` Character. The high-end colour of the continuous Purity gradient. Default is `"firebrick"`. Ignored when `show_purity = FALSE` or `plot = FALSE`.

`panel_border_color` Character. Colour of the panel border. Default is `"black"`. Ignored when `plot = FALSE`.

`panel_border_size` Numeric. Line width of the panel border. Default is `0.5`. Ignored when `plot = FALSE`.

`axis_text_size` Numeric. Font size (in points) for axis tick labels. Default is `8`. Ignored when `plot = FALSE`.

`axis_title_size` Numeric. Font size (in points) for axis titles. Default is `9`. Ignored when `plot = FALSE`.

`plot_margin_right` Numeric. Right margin of the plot in points. Default is `10`. Ignored when `plot = FALSE`.

`xlab` Character. Label for the x-axis. Default is `"Rank"`. Ignored when `plot = FALSE`.

`ylab` Character. Label for the y-axis. Default is `"Feature"`. Ignored when `plot = FALSE`.

`show_legend` Logical. Whether to display the plot legend. Default is `TRUE`. Ignored when `plot = FALSE`.

`legend_position` Character. Position of the legend. One of `"right"` (default), `"left"`, `"top"`, `"bottom"`, or `"none"`. Ignored when `plot = FALSE`.

`legend_box` Character. Arrangement of multiple legend keys. One of `"vertical"` (default) or `"horizontal"`. Ignored when `plot = FALSE`.

`legend_box_just`
: Character. Justification of legend boxes. Default is `"left"`. Ignored when `plot = FALSE`.

Details

Rank interpretation. The `Rank` column reflects the rank of the feature *within its specific collection* (i.e. within the combination of `Level`, `Membership`, and `Type`), as assigned by `clustoCell` or `markoClust`. It does *not* represent the row position in the output table returned by `featureInspect()`. A feature ranked 1st in cluster C1 positive markers and 5th in sub-cluster C1-Sub1 medium markers will appear in two separate rows with `Rank` values of 1 and 5, respectively.

Dot size in the plot. When `plot = TRUE`, dot size encodes the *inverted* Gini score: a lower Gini score indicates a purer marker and is represented by a *larger* dot. The size legend labels display the original Gini score values for interpretability. Features without a Gini score (i.e. global features stored in `globally_pure_ranked`, `globally_pure_high`, or `globally_pure_medium`) are rendered as large, semi-transparent grey dots with a heavier border stroke to signal that their size carries no quantitative meaning. These features are also plotted at $x = 0$ because no rank is assigned to them; the `0` tick label is suppressed in panels that contain only unranked features to avoid misinterpretation.

Level and type filtering. When `level` and/or `type` are specified, only rows matching the requested value(s) are returned. The two filters are applied sequentially and independently: `level` is applied first, then `type`. Specifying `type = "Pure"` expands to all three global pure-type categories (`"Pure Ranked"`, `"Pure High"`, `"Pure Medium"`) but does *not* override the `level` filter — if `level` is set to a non-global level, the combination will yield zero rows (with a warning). If no features are found after filtering, a zero-row `data.frame` is returned with a warning rather than an error, allowing `featureInspect()` to be used safely inside loops or `lapply()` calls.

Value

If `plot = FALSE` A `data.frame` with one row for each occurrence of each queried feature across all (or the selected) marker collections. Columns are:

`Feature` Feature name (character).

`Level` Hierarchical level at which the feature was found: `"Global"`, `"Cross-cluster"`, `"Major cluster"`, or `"Sub-cluster"` (character).

`Membership` The specific collection in which the feature was found, e.g. `"Global Features"`, `"Cross-cluster Marker"`, `"C1"`, or `"C1-Sub1"` (character).

`Type` Marker type: `"Pure High"`, `"Pure Medium"`, `"Pure Ranked"`, `"Positive"`, `"Negative"`, or `"Medium"` (character).

`Gini_Score` Gini score of the feature within the collection (numeric). `NA` for global features.

`Purity` Purity of the feature within the collection (numeric). `NA` for global and cross-cluster features.

`Rank` Rank of the feature within its specific collection (integer). `NA` for global features. See Details.

If no queried feature is found in any collection (or in the specified level(s)), a zero-row `data.frame` with the above columns is returned.

If `plot = TRUE` A named list with two elements:

`$table` The results `data.frame` described above.

`$plot` A `ggplot` object visualising the identified feature memberships as a dot plot faceted by `Membership`. Dot position (x-axis) encodes `Rank`, dot size encodes the inverted `Gini_Score` (larger = purer), dot shape encodes `Type`, and dot colour encodes `Purity` (or `Type` when `show_purity = FALSE`). Features without a Gini score are shown as large semi-transparent grey dots. See Details.

See Also

`clustoCell`, `markoClust`

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

cc <- clustoCell(
  data = pbmc_small,
  identify_subclusters = FALSE,
  num_threads = 1,
  verbose = FALSE
)

markers <- getDatasetMarkers(
  obj = cc,
  sub_clusters = FALSE,
  pos_thresh = 5,
  verbose = FALSE
)

features <- utils::head(
  markers$combined_markers,
  3
)

result <- featureInspect(
  clustoCell = cc,
  features = features
)

result
```

`getDatasetMarkers` *Collect marker genes from a ClustoCell object*

Description

Extracts positive, negative, and/or medium markers from major clusters and sub-clusters stored in a `ClustoCell` object.

Usage

```
getDatasetMarkers(
  obj,
  clusters = TRUE,
  sub_clusters = TRUE,
  positive_markers = TRUE,
  negative_markers = FALSE,
  medium_markers = FALSE,
  thresh_mode = c("n", "rank"),
  pos_thresh = 25,
  neg_thresh = 20,
  med_thresh = 10,
  verbose = TRUE
)
```

Arguments

`obj`	An object of class `ClustoCell`.
`clusters`	Logical; whether to collect markers from major clusters.
`sub_clusters`	Logical; whether to collect markers from sub-clusters.
`positive_markers`	Logical; whether to collect positive markers.
`negative_markers`	Logical; whether to collect negative markers.
`medium_markers`	Logical; whether to collect medium markers.
`thresh_mode`	Character; marker selection strategy. One of: • `"rank"`: include all markers with ranks up to the threshold. • `"n"`: include only the top n markers (rows) in rank order..
`pos_thresh`	Integer; threshold for positive markers.
`neg_thresh`	Integer; threshold for negative markers.
`med_thresh`	Integer; threshold for medium markers.
`verbose`	Logical; whether to display progress messages.

Details

Marker selection can be controlled using rank-based or fixed-size thresholds. Separate thresholds are applied for positive, negative, and medium markers.

Value

An object of class `DatasetMarkers`.

See Also

`clustoCell`, `markoClust`

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

cc <- clustoCell(
  data = pbmc_small,
  identify_subclusters = FALSE,
  num_threads = 1,
  verbose = FALSE
)

markers <- getDatasetMarkers(
  obj = cc,
  sub_clusters = FALSE,
  pos_thresh = 20,
  verbose = FALSE
)
```

`gini.ewcsr.fs` *Feature selection using Gini coefficient on EWCSR-transformed data*

Description

Performs feature selection based on the Gini inequality coefficient computed on expression-weighted centered scaled rank (EWCSR) data.

Usage

```
gini.ewcsr.fs(
  mat,
  gini_thresh = 0.5,
  ewcsr_high_thresh = NULL,
  ewcsr_low_thresh = NULL,
  noise_thresh = NULL,
  num_threads = -1
)
```

Arguments

`mat`	A matrix with features as rows and cells as columns.
`gini_thresh`	Numeric; Gini threshold for selecting specific features.
`ewcsr_high_thresh`	Numeric; upper EWCSR threshold for binarization. EWCSR values higher than this threshold will be converted to TRUE.
`ewcsr_low_thresh`	Numeric; lower EWCSR threshold for binarization. EWCSR values lower than this threshold will be converted to TRUE.
`noise_thresh`	Integer; minimum number of cells required for a feature to be retained.
`num_threads`	Integer; number of threads to use. `-1` uses all available cores.

Details

Features are categorized into specific, non-specific, and no-occurrence groups based on Gini thresholds and optional binarization of EWCSR values.

Value

A list containing `specific_features`, `non_specific_features`, and `no_occurrence`.

See Also

`ewcsr.sparse`, `gini.rank.fs`

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

mat <- SeuratObject::LayerData(
  pbmc_small,
  assay = "RNA",
  layer = "counts"
)

fs <- gini.ewcsr.fs(
  mat,
  num_threads = 1
)
```

`gini.rank.fs` *Feature selection using Gini coefficient on ranked expression data*

Description

Identifies specific and non-specific features based on the Gini inequality coefficient computed on ranked expression values.

Usage

```
gini.rank.fs(mat, gini_thresh = 0.5, noise_thresh = NULL, num_threads = -1)
```

Arguments

`mat`	A matrix with features as rows and cells as columns.
`gini_thresh`	Numeric; Gini threshold for selecting specific features.
`noise_thresh`	Integer; minimum number of cells required for a feature to be retained.
`num_threads`	Integer; number of threads to use. `-1` uses all available cores.

Details

This method removes globally low-ranked features while retaining features with consistently high ranks across cells for downstream analysis.

Value

A list containing `specific_features`, `non_specific_features`, and `no_occurrence`.

See Also

`gini.ewcsr.fs`

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

mat <- SeuratObject::LayerData(
  pbmc_small,
  assay = "RNA",
  layer = "counts"
)

fs <- gini.rank.fs(
  mat,
  num_threads = 1
)
```

---

install_celliverse_agent

Prepare this machine to run the CelliVerse agent

Description

The agent is **cloud-first**: it runs out of the box with a cloud provider (set an API key in Settings) and needs *no* local model. This installer verifies the R web-stack, creates `tools::R_user_dir("celliverse", "cache")` + a default config, and *optionally* sets up a local runtime - it detects Ollama (and pulls a tier model when found) and detects LM Studio (via its `lms` CLI). A local model is never required; you can install Ollama and/or LM Studio at any time later and switch providers in Settings. The config `default_model` is only pointed at a local Ollama model when Ollama is actually present (or you pass `model =` explicitly); otherwise the cloud default is left untouched.

Usage

```
install_celliverse_agent(
  tier = c("auto", "light", "recommended", "strong", "both", "all"),
  model = NULL,
  pull_model = TRUE
)
```

Arguments

tier	which local model tier(s) to pull when Ollama is present: ‘"auto"‘ (default; detect RAM and pick the best tier that fits), ‘"light"‘ (small, fast), ‘"recommended"‘ (MoE, strong tool-calling, needs ~24 GB), ‘"strong"‘ (largest, needs ~36 GB), ‘"both"‘ (light+strong), or ‘"all"‘ (all three). Ignored if ‘model‘ is given. No effect when Ollama is absent.
model	explicit local Ollama model id to pull. Overrides ‘tier‘ when supplied, and (unlike ‘tier‘) also sets the config ‘default_model‘ even if Ollama is not yet installed. Defaults to the hardware-recommended tier.
pull_model	actually run ‘ollama pull‘ if Ollama is found.

Value

invisibly, a named list summarising what was found/done.

---

jaccard.sparse *Compute Jaccard similarity for sparse matrices*

Description

Computes pairwise Jaccard similarity between columns of a sparse matrix.

Usage

```
jaccard.sparse(mat, num_threads = -1)
```

Arguments

mat	A sparse matrix of class `Matrix`.
num_threads	Integer; number of threads to use. `-1` uses all available cores.

Value

A numeric matrix of Jaccard similarity scores.

See Also

`mutual.rank`

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

mat <- SeuratObject::LayerData(
  pbmc_small,
  assay = "RNA",
  layer = "counts"
)

binary_mat <- mat
binary_mat@x[] <- 1

jac <- jaccard.sparse(
  binary_mat,
  num_threads = 1
)
```

---

markerDB *Cell-type marker database*

Description

A curated database of positive and negative cell-type marker genes for human and mouse, stored in sparse matrix format. This dataset is used internally by `typoClust()` for marker-based cell-type annotation.

Format

An object of class `CelliVerse_Data`, implemented as a named list with two elements:

human A list containing human marker databases:

- **positive_db** An object of class `CelliVerse_Sparse_Data` wrapping a sparse `dgCMatrix` of dimensions 16,899 genes × 9,410 cell types.
- **negative_db** An object of class `CelliVerse_Sparse_Data` wrapping a sparse `dgCMatrix` of dimensions 329 genes × 93 cell types.

mouse A list containing mouse marker databases:

- **positive_db** An object of class `CelliVerse_Sparse_Data` wrapping a sparse `dgCMatrix` of dimensions 4,779 genes × 2,112 cell types.
- **negative_db** An object of class `CelliVerse_Sparse_Data` wrapping a sparse `dgCMatrix` of dimensions 32 genes × 33 cell types.

Details

Rows correspond to marker genes and columns correspond to annotated cell types. Matrix values encode marker presence or strength as defined during database construction. Sparse representation is used to minimize memory usage.

Source

Curated from published cell-type marker resources and expert annotation.

See Also

typoClust, markerDictionary, tissueCondition_types

`markerDictionary` *Marker gene dictionary*

Description

A species-specific dictionary mapping marker identifiers to standardized gene annotations, including gene symbols, aliases, and database identifiers. This dataset supports marker harmonization and identifier resolution within the `celliverse` framework.

Format

An object of class `CelliVerse_Data`, implemented as a named list with two elements:

human A data frame with 20,887 rows and 6 variables:

- **Marker** Internal marker identifier
- **Symbol** Official gene symbol
- **Alias** Alternative gene symbols or aliases
- **Entrez** Entrez Gene identifier
- **Ensembl** Ensembl gene identifier
- **UniProt** UniProt protein identifier

mouse A data frame with 4,779 rows and 6 variables:

- **Marker** Internal marker identifier
- **Symbol** Official gene symbol
- **Alias** Alternative gene symbols or aliases
- **Entrez** Entrez Gene identifier
- **Ensembl** Ensembl gene identifier
- **UniProt** UniProt protein identifier

Details

This dictionary is used to standardize marker gene identifiers across datasets and species, enabling consistent matching between user-provided markers and curated cell-type marker databases.

Source

Integrated from public gene annotation resources including Ensembl, Entrez Gene, and UniProt.

See Also

markerDB, typoClust

`markerPurity` *Assess marker purity across clusters or cell subsets*

Description

Quantifies marker purity by evaluating expression specificity within clusters or user-defined cell subsets.

Usage

```
markerPurity(
  data,
  assay = "RNA",
  layer = "counts",
  desired_markers = NULL,
  cluster_labels = NULL,
  desired_clusters = NULL,
  desired_cells = NULL,
  log1p = TRUE,
  remove_quiescent_cells = TRUE,
  high_quantile = 0.25,
  low_quantile = 0.25,
  noise_feature_thresh = 4,
  num_threads = -1,
  seed = 121,
  verbose = TRUE
)
```

Arguments

`data`	Either a `Seurat` object or a numeric matrix with features (genes) as rows and cells as columns.
`assay`	Assay name used for marker purity assessment.
`layer`	Data layer used for assessment.
`desired_markers`	Character vector of markers to assess.
`cluster_labels`	Character vector. Required if 'desired_clusters' is specified. Either the name of the column in data@meta.data containing cluster labels, or a character vector of cluster labels with length equal to the number of columns (cells) in the 'data' argument.
`desired_clusters`	Character vector of clusters to assess. Required if 'desired_cells' is not specified. If not provided, marker purity will be assessed solely within 'desired_cells'.
`desired_cells`	Named list of character vectors specifying the names of the desired cells. Required if 'desired_clusters' is not specified.

log1p	Logical; whether to apply log1p transformation to the input data. It is recommended to set this argument to TRUE (default) if the data is not already on a log scale.
remove_quiescent_cells	Logical; whether to remove quiescent cells.
high_quantile	Quantile for defining high EWCSR values.
low_quantile	Quantile for defining low EWCSR values.
noise_feature_thresh	Threshold for filtering noise features.
num_threads	Number of threads to use.
seed	Random seed.
verbose	Logical; whether to display progress messages.

Details

Marker purity is assessed using EWCSR-based filtering and supports both Seurat objects and matrix-based inputs.

Value

An object of class MarkerPurity.

See Also

markoCell, getDatasetMarkers

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

pbmc_small$example_clusters <- as.character(
  SeuratObject::Idents(pbmc_small)
)

cluster_ids <- utils::head(
  unique(pbmc_small$example_clusters),
  2
)

mp <- markerPurity(
  data = pbmc_small,
  desired_markers = rownames(pbmc_small)[1:10],
  cluster_labels = "example_clusters",
  desired_clusters = cluster_ids,
  num_threads = 1,
  verbose = FALSE
)
```

markoCell *Rank markers for clusters, cell subsets, or individual cells*

Description

Identifies and ranks positive and negative marker genes for a specified set of cells, which may correspond to clusters, sub-clusters, arbitrary cell subsets, or even single cells. Marker ranking is based on expression-weighted centered scaled ranks (EWCSR), with Gini-based specificity assessment, and noise suppression.

Usage

```
markoCell(
  data,
  assay = "RNA",
  layer = "counts",
  norm_assay = "RNA",
  norm_layer = "data",
  cluster_labels = NULL,
  desired_clusters = NULL,
  desired_cells = NULL,
  log1p = TRUE,
  remove_quiescent_cells = TRUE,
  high_quantile = 0.25,
  low_quantile = 0.25,
  subset_to_HVG = FALSE,
  hvg_selection.method = c("vst", "mean.var.plot", "dispersion"),
  hvg_var_thresh = 1,
  gini_thresh = 0.5,
  noise_feature_thresh = 4,
  random_marker_thresh = 5,
  num_threads = -1,
  seed = 9999,
  verbose = TRUE
)
```

Arguments

data	Either a Seurat object or a numeric matrix with features (genes) as rows and cells as columns. Recommended to provide at least library-size normalized data when subset_to_HVG = TRUE.
assay	Character; assay used for marker ranking.
layer	Character; assay layer used for marker ranking. May be normalized.
norm_assay	Character; assay containing a normalized layer used for HVG detection.
norm_layer	Character; normalized layer used for HVG detection.

`cluster_labels` Optional; column name in `data@meta.data` containing cluster labels, or a character vector of cluster labels with length equal to the number of cells in `data`. Required if `desired_clusters` is specified.

`desired_clusters` Optional; character vector of cluster labels for which markers are ranked. Required if `desired_cells` is not specified.

`desired_cells` Optional; named list of character vectors specifying cell names for each subset. Required if `desired_clusters` is not specified.

`log1p` Logical; whether to apply `log1p` transformation to the input `data`. It is recommended to set this argument to TRUE (default) if the data is not already on a log scale.

`remove_quiescent_cells` Logical; whether to remove quiescent cells prior to marker ranking.

`high_quantile` Numeric; quantile threshold defining highly positive EWCSR values.

`low_quantile` Numeric; quantile threshold defining highly negative EWCSR values.

`subset_to_HVG` Logical; whether to restrict analysis to highly variable genes (HVGs).

`hvg_selection.method` Character; HVG selection strategy. One of `"vst"`, `"mean.var.plot"`, or `"dispersion"`.

`hvg_var_thresh` Numeric; variance threshold for selecting HVGs.

`gini_thresh` Numeric; Gini coefficient threshold for detecting non-specific markers.

`noise_feature_thresh` Integer; features expressed in fewer than this number of cells are considered noise.

`random_marker_thresh` Integer; markers detected in fewer than this number of cells are discarded.

`num_threads` Integer; number of threads to use. Default `-1` uses all available cores.

`seed` Integer; random seed for reproducibility.

`verbose` Logical; whether to display progress messages.

Details

When `subset_to_HVG = TRUE`, highly variable genes are detected using the normalized assay and layer specified by `norm_assay` and `norm_layer`. Gini-based filtering is applied to identify global (non-specific) versus specific markers.

Value

An object of class `"MarkoCell"` containing ranked marker tables and associated statistics for each requested cell set.

See Also

`markoClust`, `markerPurity`, `gini.ewcsr.fs`

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

pbmc_small$example_clusters <- as.character(
  SeuratObject::Idents(pbmc_small)
)

cluster_ids <- utils::head(
  unique(pbmc_small$example_clusters),
  2
)

mc <- markoCell(
  data = pbmc_small,
  cluster_labels = "example_clusters",
  desired_clusters = cluster_ids,
  num_threads = 1,
  verbose = FALSE
)
```

---

markoClust *Evaluate and refine cell clusters using marker ranking and graph partitioning*

Description

Identifies markers in predefined cell clusters, and optionally identifying sub-clusters using Leiden community detection. Supports large datasets via uniform sketching with label transfer back to the full dataset.

Usage

```
markoClust(
  data,
  assay = "RNA",
  layer = "counts",
  norm_assay = "RNA",
  norm_layer = "data",
  cluster_labels,
  log1p = TRUE,
  remove_quiescent_cells = TRUE,
  high_quantile = 0.25,
  low_quantile = 0.25,
  subset_to_HVG = FALSE,
  hvg_selection.method = c("vst", "mean.var.plot", "dispersion"),
  hvg_var_thresh = 1,
  gini_thresh = 0.5,
```

```
  sketch = FALSE,
  sketch_fraction = 0.5,
 label_transfer_method = c("ewcsr-cor", "seurat-project", "ewcsr-red-cor", "seurat-knn"),
  sketch_pca_dims = 30,
  noise_feature_thresh = 4,
  random_marker_thresh = 5,
  mr_thresh = NULL,
  isolated_cluster_thresh = 5,
  leiden_obj_function = c("modularity", "CPM"),
  leiden_resolution = 0.75,
  leiden_n_iterations = 5,
  identify_subclusters = FALSE,
  num_threads = -1,
  seed = 121,
  verbose = TRUE
)
```

Arguments

data	Either a Seurat object or a numeric matrix with features (genes) as rows and cells as columns. Recommended to provide at least library-size normalized data when subset_to_HVG = TRUE.
assay	Character; assay used for cluster evaluation.
layer	Character; assay layer used for cluster evaluation.
norm_assay	Character; assay containing normalized data for HVG detection.
norm_layer	Character; normalized layer used for HVG detection.
cluster_labels	Character vector. Either the name of the column in data@meta.data containing cluster labels, or a character vector of cluster labels with length equal to the number of columns (cells) in the ‘data‘ argument.
log1p	Logical; whether to apply log1p transformation to the input data. It is recommended to set this argument to TRUE (default) if the data is not already on a log scale.
remove_quiescent_cells	Logical; whether to remove quiescent cells prior to analysis.
high_quantile	Numeric; EWCSR high quantile threshold.
low_quantile	Numeric; EWCSR low quantile threshold.
subset_to_HVG	Logical; whether to restrict analysis to highly variable genes (HVGs).
hvg_selection.method	Character; HVG selection strategy.
hvg_var_thresh	Numeric; HVG variance threshold.
gini_thresh	Numeric; Gini coefficient threshold for detecting global markers.
sketch	Logical; whether to apply uniform sketching for sub-clustering.
sketch_fraction	Numeric between 0 and 1; fraction of cells per cluster to retain in sketch.

label_transfer_method
: Character; method for transferring labels from sketched to full data. One of "ewcsr-cor", "seurat-project", "ewcsr-red-cor", or "seurat-knn".

sketch_pca_dims
: Integer; number of PCA dimensions used during label transfer.

noise_feature_thresh
: Integer; threshold for identifying noise features.

random_marker_thresh
: Integer; threshold for discarding weak markers.

mr_thresh
: Numeric; mutual-rank threshold for filtering cell similarities.

isolated_cluster_thresh
: Integer; clusters with fewer cells are treated as isolated.

leiden_obj_function
: Character; Leiden objective function. One of "modularity" or "CPM".

leiden_resolution
: Numeric; Leiden resolution parameter.

leiden_n_iterations
: Integer; number of Leiden iterations.

identify_subclusters
: Logical; whether to identify sub-clusters.

num_threads
: Integer; number of threads to use.

seed
: Integer; random seed.

verbose
: Logical; whether to display progress messages.

Details

If sketch = TRUE, a representative subset of cells is sampled from each cluster and used for sub-clustering. Labels are transferred back to the full dataset using the method specified by label_transfer_method. The "seurat-project" and "seurat-knn" methods operate on the original expression matrix rather than the EWCSR representation. The supplied expression data may consist of raw counts, log-normalized expression values, or SCTransform-normalized data, provided that the sketched and full datasets were processed using the same normalization strategy. These methods leverage Seurat's native label transfer workflows (ProjectData, FindTransferAnchors, and TransferData) and do not require integer count matrices.

Value

An object of class "ClustoCell" containing refined clusters, sub-clusters, marker tables, and similarity structures.

See Also

markoCell, addClustoData

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

pbmc_small$example_clusters <- as.character(
  SeuratObject::Idents(pbmc_small)
)

cc <- markoClust(
  data = pbmc_small,
  cluster_labels = "example_clusters",
  identify_subclusters = FALSE,
  num_threads = 1,
  verbose = FALSE
)
```

---

markoClustVis *Visualize cluster and cell-subset markers*

Description

Generates a faceted dot plot for visualizing marker genes across clusters, sub-clusters, or cell subsets stored in a `ClustoCell` or `MarkoCell` object. Marker selection can be controlled by rank or by selecting the top `n` markers per group. Dot size and color can represent marker purity or marker class.

Usage

```
markoClustVis(
  obj,
  desired_sets = NULL,
  show_pos_markers = TRUE,
  show_neg_markers = FALSE,
  show_med_markers = FALSE,
  thresh_mode = c("n", "rank"),
  thresh = 5,
  title = NULL,
  subtitle = NULL,
  tag = NULL,
  nrow_panels = NULL,
  dotsize = 2,
  show_purity = TRUE,
  class_palette = NULL,
  color_low = "blue",
  color_high = "red",
  panel_border_color = "black",
  panel_border_size = 0.5,
  axis_text_size = 7,
```

```
  axis_title_size = 8,
  plot_margin_right = 10,
  xlab = "Rank",
  ylab = "Marker",
  show_legend = TRUE,
  legend_box = "vertical",
  legend_box_just = "left",
  legend_position = "right"
)
```

Arguments

obj	An object of class ClustoCell or MarkoCell containing marker information.
desired_sets	Optional character vector specifying the names of clusters, sub-clusters, and/or cell subsets to include. If NULL, all available sets in obj are used.
show_pos_markers	Logical; whether to include positive markers. Default is TRUE.
show_neg_markers	Logical; whether to include negative markers. Default is FALSE.
show_med_markers	Logical; whether to include medium markers. Default is FALSE.
thresh_mode	Character; method for selecting top markers. One of: • "rank": include all markers up to the specified rank threshold. • "n": include exactly the top n markers.
thresh	Integer; threshold for selecting markers based on thresh_mode. Default is 5.
title	Optional character string for the plot title.
subtitle	Optional character string for the plot subtitle.
tag	Optional character string for the plot tag.
nrow_panels	Optional integer specifying the number of rows in the faceted plot. If NULL, rows are determined automatically.
dotsize	Numeric; size of the dots in the plot. Default is 2.
show_purity	Logical; if TRUE, dot color represents marker purity. If FALSE, dot color represents marker class. Default is TRUE.
class_palette	Optional palette used when show_purity = FALSE. Can be either: • A ggplot2 scale object (e.g., ggplot2::scale_fill_hue()) • A character vector of colors
color_low	Character; low color for gradient (used when show_purity = TRUE). Default is "blue".
color_high	Character; high color for gradient (used when show_purity = TRUE). Default is "red".
panel_border_color	Character; color of panel borders.
panel_border_size	Numeric; size of panel borders.

`axis_text_size` Numeric; font size for axis text.

`axis_title_size`
Numeric; font size for axis titles.

`plot_margin_right`
Numeric; right margin of the plot.

`xlab` Character; label for the x-axis. Default is `"Rank"`.

`ylab` Character; label for the y-axis. Default is `"Marker"`.

`show_legend` Logical; whether to display the legend. Default is `TRUE`.

`legend_box` Character; layout of the legend box (e.g., `"vertical"`).

`legend_box_just`
Character; justification of the legend box.

`legend_position`
Character; position of the legend (e.g., `"right"`).

Details

This function provides a flexible visualization for exploring marker genes identified in clustering analyses. Marker selection can be based on rank or a fixed number of top markers. The resulting plot is faceted by cluster or subset, enabling comparison across groups.

When `show_purity = TRUE`, a continuous color scale is used to represent marker purity. Otherwise, discrete colors are used to represent marker classes (e.g., positive, negative, medium).

Value

A `ggplot2` object showing a faceted dot plot of selected markers across clusters, sub-clusters, or cell subsets.

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

pbmc_small$example_clusters <- as.character(
  SeuratObject::Idents(pbmc_small)
)

cc <- markoClust(
  data = pbmc_small,
  cluster_labels = "example_clusters",
  identify_subclusters = FALSE,
  num_threads = 1,
  verbose = FALSE
)

plt <- markoClustVis(
  obj = cc,
  show_pos_markers = TRUE,
  show_neg_markers = FALSE,
  thresh_mode = "n",
  thresh = 2
```

```
)

plt
```

`mutual.rank` *Compute mutual rank from a similarity matrix*

Description

Computes the mutual rank (MR) for all pairs in a symmetric similarity or dissimilarity matrix, a robust measure frequently used to stabilize pairwise similarity relationships.

Usage

```
mutual.rank(mat, num_threads = -1)
```

Arguments

`mat`	Numeric symmetric matrix (dense or sparse) representing pairwise similarities or dissimilarities.
`num_threads`	Integer; number of threads to use. Default `-1` uses all available cores.

Value

A numeric matrix of mutual ranks with the same dimensions as `mat`.

See Also

`jaccard.sparse`, `markoClust`

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

mat <- SeuratObject::LayerData(
  pbmc_small,
  assay = "RNA",
  layer = "counts"
)

binary_mat <- sign(mat)

similarity_matrix <- jaccard.sparse(
  binary_mat,
  num_threads = 1
)

mr <- mutual.rank(
```

```
  similarity_matrix,
  num_threads = 1
)
```

`run_celliverse_agent` *Launch the CelliVerse agent (API + UI)*

Description

Starts the local plumber API, serves the prebuilt React UI from 'inst/react-app/', binds to localhost, and (optionally) opens a browser. The agent runs cloud-first: no local model is required — pick a cloud provider + API key in Settings, or install Ollama / LM Studio later for fully-offline local models.

Usage

```
run_celliverse_agent(
  port = NULL,
  host = "127.0.0.1",
  provider = NULL,
  model = NULL,
  browser = interactive(),
  background = FALSE,
  port_scan = TRUE,
  max_port_tries = 20L
)
```

Arguments

`port`	TCP port to bind (default from config, else 8000).
`host`	host to bind; keep 127.0.0.1 for localhost-only (default).
`provider`	optional provider override for this run (writes to config).
`model`	optional model override for this run.
`browser`	open a browser window automatically.
`background`	run the server in a background process (returns a handle) instead of blocking the console.
`port_scan`	if 'TRUE' (default) and 'port' is already in use, bind the next free port instead of failing; if 'FALSE', a busy port is a hard error.
`max_port_tries`	how many ports above 'port' to probe when scanning (default 20).

Value

invisibly, the plumber router (foreground) or a process handle (background).

saveTypoPrompt *Save a TypoPrompt to a file*

Description

Saves a TypoPrompt object as either its exact plain-text prompt or a self-contained interactive HTML page.

Usage

```
saveTypoPrompt(x, file, format = c("txt", "html"))
```

Arguments

x	A TypoPrompt object returned by typoPrompt.
file	Character string giving the output file path.
format	Output format: "txt" for the raw prompt or "html" for the interactive HTML viewer.

Value

Invisibly returns the normalized output file path.

See Also

typoPrompt

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

cc <- clustoCell(
  data = pbmc_small,
  identify_subclusters = FALSE,
  num_threads = 1,
  verbose = FALSE
)

desired_set <- utils::head(
  sort(unique(as.character(cc$clusters$major_clusters))),
  1
)

prompt <- typoPrompt(
  object = cc,
  desired_sets = desired_set,
  sample_source = "human peripheral blood",
  tissue = "Blood",
  condition = "Healthy",
```

```
  species = "human",
  use_neg_markers = FALSE,
  thresh = 10,
  verbose = FALSE
)

txt_file <- tempfile(fileext = ".txt")

saveTypoPrompt(
  prompt,
  file = txt_file,
  format = "txt"
)
```

---

signatureDotHeatmap *Visualize per-signature marker expression across clusters*

Description

Generates a dot heatmap summarizing expression of signature-associated features across clusters.

Usage

```
signatureDotHeatmap(
  seurat_obj,
  cluster_col,
  row_data,
  features_col = NULL,
  signature_col = NULL,
  cell_type_colors = NULL,
  signature_colors = NULL,
  tag = NULL,
  tile_palette = RColorBrewer::brewer.pal(9, "YlOrRd"),
  tile_alpha_range = c(0.15, 0.65),
  dot_size_factor = 2,
  dot_range = c(0.5, 3),
  signature_label_size = 3.3,
  feature_label_size = 6,
  feature_label_angle = 45,
  show_cluster_labels = FALSE,
  show_signature_strip = TRUE,
  show_signature_labels = TRUE,
  show_cluster_strip = TRUE,
  show_cluster_legend = TRUE,
  show_signature_legend = FALSE,
```

```
  legend_ncol = 2,
  expression_legend_title = "Expression",
  percent_legend_title = "Percent\nExpressed",
  tile_fill_legend_title = NULL,
  signature_legend_title = "Signature",
  vline_color = "black",
  vline_width = 0.15,
  block_border_color = "grey90",
  feature_label_face = "plain"
)
```

Arguments

`seurat_obj`	A `Seurat` object containing expression data and metadata.
`cluster_col`	Character; column in `seurat_obj@meta.data` containing cluster labels. These could correspond to user defined clusters or cell types.
`row_data`	Data frame mapping features to signatures. Must contain one row per feature.
`features_col`	Character; column in `row_data` containing feature (e.g. gene) IDs corresponding to the row names of the `seurat_obj`.
`signature_col`	Character; column in `row_data` containing signature labels.
`cell_type_colors`	Color specification for cell types.
`signature_colors`	Color specification for signatures.
`tag`	Optional plot tag.
`tile_palette`	Character vector defining tile fill colors.
`tile_alpha_range`	Numeric vector of length two defining alpha range.
`dot_size_factor`	Numeric; scaling factor for dot sizes.
`dot_range`	Numeric vector defining minimum and maximum dot sizes.
`signature_label_size`	Numeric; font size of signature labels.
`feature_label_size`	Numeric; font size of feature labels.
`feature_label_angle`	Numeric; angle of feature labels.
`show_cluster_labels`	Logical; whether to show cluster labels.
`show_signature_strip`	Logical; whether to show signature strip.
`show_signature_labels`	Logical; whether to show signature labels.
`show_cluster_strip`	Logical; whether to show cluster strip.

`show_cluster_legend`
: Logical; whether to show cluster legend.

`show_signature_legend`
: Logical; whether to show signature legend.

`legend_ncol`
: Integer; number of legend columns.

`expression_legend_title`
: Character; title for expression legend.

`percent_legend_title`
: Character; title for percent expressed legend.

`tile_fill_legend_title`
: Character; title for tile fill legend.

`signature_legend_title`
: Character; title for signature legend.

`vline_color`
: Character; color of vertical separator lines.

`vline_width`
: Numeric; width of vertical separator lines.

`block_border_color`
: Character; color of block borders.

`feature_label_face`
: Character; font face for feature labels.

Details

Dot size typically encodes the percentage of cells expressing a feature, while color intensity reflects average expression.

Value

A `ggplot2` object.

See Also

`typoClustVis`, `markoCell`

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

pbmc_small$example_clusters <- as.character(
  SeuratObject::Idents(pbmc_small)
)

features <- rownames(pbmc_small)[1:6]

signatures <- data.frame(
  Features = features,
  Signature = rep(
    c("Signature 1", "Signature 2"),
    each = 3
```

```
  )
)

p <- signatureDotHeatmap(
  seurat_obj = pbmc_small,
  cluster_col = "example_clusters",
  row_data = signatures,
  features_col = "Features",
  signature_col = "Signature"
)

p
```

`tissueCondition_types` *Tissue and condition reference catalog*

Description

A reference catalog of tissues and disease conditions for human and mouse, used to contextualize cell-type annotation by tissue and pathological state.

Format

An object of class `CelliVerse_Data`, implemented as a named list with two elements:

human A list containing:

- **all_tissues** Character vector of all supported tissues (length 396)
- **healthy_tissue** Character vector of healthy tissues (length 367)
- **diseased_tissue** Character vector of diseased tissues (length 103)
- **all_conditions** Character vector of all supported conditions (length 265)
- **diseased_tissueCondition** A data frame with two variables:
  - **Tissue** Tissue name
  - **Condition** Associated disease condition

mouse A list containing:

- **all_tissues** Character vector of all supported tissues (length 110)
- **healthy_tissue** Character vector of healthy tissues (length 108)
- **diseased_tissue** Character vector of diseased tissues (length 19)
- **all_conditions** Character vector of all supported conditions (length 51)
- **diseased_tissueCondition** A data frame with two variables:
  - **Tissue** Tissue name
  - **Condition** Associated disease condition

Details

This dataset is used by `typoClust()` to restrict or guide cell-type annotation according to tissue context and disease state.

Source

Curated from public tissue ontologies and disease annotation resources.

See Also

typoClust, markerDB

---

typoClust *Cell type annotation of clusters, sub-clusters, or cell subsets*

Description

Annotates clusters, sub-clusters, or arbitrary cell subsets using curated cell-type marker databases or large language model (LLM)–based annotation. Annotation can be performed either from marker results stored in `ClustoCell` or `MarkoCell` objects, or directly from user-specified positive and/or negative marker panels.

Usage

```
typoClust(
  objects = NULL,
  desired_sets = NULL,
  tissue = NULL,
  condition = NULL,
  use_pos_markers = TRUE,
  use_neg_markers = TRUE,
  desired_pos_markers = NULL,
  desired_neg_markers = NULL,
  thresh_mode = c("n", "rank"),
  thresh = 20,
  mode = c("markerDB", "ceLLMarkup"),
  inherit_major_clusters = TRUE,
  inherit_score_ratio = 0.5,
  species = c("human", "mouse"),
  sample_source = NULL,
  feature_type = "gene",
  llm_provider = "ollama",
  llm_model = NULL,
  llm_api_key = NULL,
  llm_host = NULL,
  llm_top_k = 3,
  verbose = TRUE
)
```

Arguments

objects	A list of one or more objects of class ClustoCell or MarkoCell (e.g. list(obj1, obj2)). Mandatory if desired_pos_markers and/or desired_neg_markers are not specified.
desired_sets	Optional character vector specifying the names of clusters, sub-clusters, and/or cell subsets to annotate. These names must exist in the supplied objects. If NULL, all available sets are annotated.
tissue	Optional character vector specifying one or more tissue contexts used for annotation. If NULL, all available tissues are examined. Available tissue types can be accessed via data("tissueCondition_types", package = "celliverse").
condition	Optional character vector specifying one or more conditions (e.g. Healthy, disease states) used for annotation. If NULL, all available conditions are examined. Available condition types can be accessed via data("tissueCondition_types", package = "celliverse").
use_pos_markers	Logical; whether to use positive markers for cell-type annotation. Default is TRUE.
use_neg_markers	Logical; whether to use negative markers for cell-type annotation. Default is TRUE.
desired_pos_markers	Optional named list of character vectors specifying positive marker panels. Each list element corresponds to one cluster or cell subset (e.g. list(cluster1 = c("GeneA", "GeneB"))). Mandatory if objects and desired_neg_markers are not specified. List names must match those of desired_neg_markers, if provided.
desired_neg_markers	Optional named list of character vectors specifying negative marker panels. Each list element corresponds to one cluster or cell subset. Mandatory if objects and desired_pos_markers are not specified. List names must match those of desired_pos_markers, if provided.
thresh_mode	Character string specifying how to select top markers. One of: • "rank": Selects all markers with ranks up to the threshold. If multiple markers share the cutoff rank, all are included. • "n": Selects strictly the top n markers in rank order, even if additional markers share the same rank.
thresh	Integer specifying the marker selection threshold. Interpreted according to thresh_mode. Only used when objects is specified.
mode	Character string specifying the annotation strategy. One of: • "markerDB": Annotates cell sets using curated cell-type marker databases. • "ceLLMarkup": Annotates cell sets using large language models (LLM) via ceLLMarkup; configure the LLM with llm_provider, llm_model, llm_api_key, llm_host.

`inherit_major_clusters`

Logical; whether a sub-cluster should be annotated *within the identity of its own major cluster*. Default `TRUE`.

When `FALSE`, every set is annotated independently and the function behaves exactly as it did before this argument existed.

When `TRUE`, and the input contains both major clusters and sub-clusters, annotation becomes two-stage. Each requested sub-cluster's parent major cluster is annotated first, from the *major cluster's own* top markers; the sub-cluster is then annotated from *its own* top markers, constrained by that parent identity:

- `mode = "markerDB"`: the database is restricted to every cell type whose name contains an admitted parent identity as a whole phrase — so a parent called `"CD8+ T Cell"` leaves `"CD8+ T Cell"` itself plus `"Exhausted CD8+ T Cell"`, `"Memory CD8+ T Cell"`, `"GZMK+ CD8+ T Cell"` and so on — and the sub-cluster is scored against that restricted database only. Which identities count as admitted is set by `inherit_score_ratio`.
- `mode = "ceLLMarkup"`: the model is told the parent's identity and asked which specific subtype or state of *that* cell type the sub-cluster represents, given the sub-cluster's own markers.

A parent major cluster needed for this is annotated and returned even when it was not itself named in `desired_sets`; `metadata$inheritance` records, per sub-cluster, which parent was used and what it was called. The restriction is applied per sub-cluster, so one major cluster's identity can never constrain a different major cluster's sub-clusters.

`inherit_score_ratio`

Numeric in `(0, 1]`, default `0.5`. `mode = "markerDB"` **only**, and only when `inherit_major_clusters = TRUE`.

How close to the parent's rank-1 score a runner-up candidate must come before it *also* constrains that parent's sub-clusters. At `1` only the rank-1 label is used; at `0.6` any candidate scoring at least 60% of the rank-1 score is admitted alongside it, and the database is restricted to the union of their named varieties.

This exists because a major cluster's own label is often not certain, and treating it as certain propagates the doubt into every sub-cluster beneath it. Measured on the bundled pbmc3k `ClustoCell` (`thresh = 10`, Blood/Healthy), each parent's rank-2 candidate as a fraction of its rank-1 score:

Parent	rank-1 vs rank-2	ratio
C1	NK Cell vs CD8+ Alpha-Beta T Cell	0.653
C2	B Cell vs MS4A1+ B Cell	0.248
C3	T Cell vs CD4+ Alpha-Beta T Cell	0.965
C4	Mononuclear Phagocyte vs Monocyte	0.837
C5	Platelet vs Megakaryocyte	0.846

C1 is a mixed cytotoxic compartment whose sub-clustering separates CD8+ T cells from NK cells; on rank-1 alone its T-cell sub-cluster is folded back into NK. C4 is the other end of the same problem: exactly one database cell type is named as a variety of `"Mononuclear Phagocyte"`, so on rank-1 alone its sub-clusters can only repeat the parent's label, while admitting `"Monocyte"` restores

a vocabulary of 32. At the default, no parent on that dataset admits more than three identities.

`species` Character string specifying the species (either `"human"` or `"mouse"`). Other species names may also be supplied when the mode is set to `"ceLLMarkup"`. Default is `"human"`.

`sample_source` Free-text description of the sample origin shown to the model (e.g. `"human peripheral blood"`). Improves accuracy. Only used when mode is set to `"ceLLMarkup"`.

`feature_type` Feature type of the markers (e.g. `"gene"`, `"protein"`). Default `"gene"`. Only used when mode is set to `"ceLLMarkup"`.

`llm_provider` (mode = `"ceLLMarkup"` only) LLM provider. One of `"ollama"`, `"lmstudio"`, `"openai"`, `"anthropic"`, `"gemini"`, `"deepseek"`, `"groq"`, `"openrouter"`, `"cerebras"`. Default `"ollama"`.

`llm_model` (mode = `"ceLLMarkup"` only) Model id for `llm_provider` (e.g. `"qwen3:8b"` for Ollama, `"qwen/qwen3-8b"` for LM Studio, `"gpt-4o-mini"` for OpenAI, `"anthropic/claude-3-haiku"` for OpenRouter). Mandatory when mode = `"ceLLMarkup"`.

`llm_api_key` (mode = `"ceLLMarkup"` only) API key for cloud providers (not needed for Ollama/LM Studio). If `NULL`, falls back to the provider's standard environment variable (e.g. `OPENAI_API_KEY`, `OPENROUTER_API_KEY`).

`llm_host` (mode = `"ceLLMarkup"` only) Base URL for local providers. Defaults to `http://localhost:11434` (Ollama) or `http://localhost:1234/v1` (LM Studio).

`llm_top_k` (mode = `"ceLLMarkup"` only) Number of ranked candidate cell types returned per set. Default 3.

`verbose` Logical; whether to display progress messages.

Details

`typoClust()` identifies candidate cell types for each target set by comparing positive and/or negative marker genes against tissue- and condition-aware cell-type marker databases. Users may restrict annotation to specific tissues or conditions, control the number of markers used per cluster, and choose between rank-based or fixed-size marker selection.

Exactly one of the following inputs must be provided:

- One or more `ClustoCell` or `MarkoCell` objects via `objects`
- User-defined positive and/or negative marker panels via `desired_pos_markers` and/or `desired_neg_markers`

Value

An object of class `TypoClust` containing ranked cell-type annotations, supporting marker evidence, and summary statistics for each annotated set.

See Also

`typoClustVis`, `markoCell`, `markoClust`, `clustoCell`

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

cc <- clustoCell(
  data = pbmc_small,
  identify_subclusters = FALSE,
  num_threads = 1,
  verbose = FALSE
)

desired_set <- utils::head(
  sort(unique(as.character(cc$clusters$major_clusters))),
  1
)

tc <- typoClust(
  objects = list(cc),
  desired_sets = desired_set,
  tissue = "Blood",
  condition = "Healthy",
  use_neg_markers = FALSE,
  thresh = 10,
  mode = "markerDB",
  species = "human",
  verbose = FALSE
)
```

`typoClustVis` *Visualization of TypoClust cell-type annotations*

Description

Visualizes cell-type annotation results generated by `typoClust()` using a composite layout combining label, tile, and dot plots.

Usage

```
typoClustVis(
  typoClust,
  desired_sets = NULL,
  rank_thresh = 1,
  refine = TRUE,
  refine_thresh = 1,
 order_by = c("Cell Type", "Cluster", "Combined Score", "Combined Count", "Purity"),
  title = NULL,
  subtitle = NULL,
  tag = NULL,
  cellType_palette = NULL,
```

```
  flip = FALSE,
  tile_width = 0.2,
  tile_height = 0.6,
  tile_xlim = c(0.5, 1.5),
  tile_legend_title = "Cell Type",
  dot_color_low = "blue",
  dot_color_high = "red",
  dot_panel_border_color = "black",
  dot_panel_border_size = 0.5,
  dot_axis_text_size = 11,
  dot_axis_title_size = 12,
  dot_plot_margin_right = 10,
  dot_xlab = "Lead Cell Type Score",
  dot_size_title = "Combined\nCount",
  dot_color_title = "Avg Purity",
  label_size = 3,
  label_padding_lines = 0.5,
  legend_box = "vertical",
  legend_box_just = "left",
  legend_position = "right"
)
```

Arguments

typoClust	An object of class TypoClust generated by typoClust().
desired_sets	Optional character vector specifying which clusters, sub-clusters, or cell subsets to visualize. If NULL, all sets in typoClust are shown.
rank_thresh	Integer specifying the top N ranked cell types to display per set.
refine	Logical; whether to refine annotations by traversing deeper into the cell-type hierarchy.
refine_thresh	Integer specifying how many hierarchical levels to traverse when refining cell types. Ignored if refine = FALSE.
order_by	Character string specifying how to order clusters in the plot. One of "Cell Type", "Cluster", "Combined Score", "Combined Count", or "Purity".
title	Optional character string specifying the plot title.
subtitle	Optional character string specifying the plot subtitle.
tag	Optional character string specifying a plot tag.
cellType_palette	A scale specification for coloring cell types. Can be either a ggplot2 scale object (e.g. ggplot2::scale_fill_hue()) or a character vector of colors.
flip	Logical; whether to flip plot axes.
tile_width	Numeric specifying tile width.
tile_height	Numeric specifying tile height.
tile_xlim	Numeric vector of length two specifying x-axis limits for tile plots.

`tile_legend_title`
: Character string specifying the tile legend title.

`dot_color_low`
: Character string specifying the low-end color of the dot color scale.

`dot_color_high`
: Character string specifying the high-end color of the dot color scale.

`dot_panel_border_color`
: Character string specifying the border color of dot panels.

`dot_panel_border_size`
: Numeric specifying the border line width of dot panels.

`dot_axis_text_size`
: Numeric specifying axis text size in the dot plot.

`dot_axis_title_size`
: Numeric specifying axis title size in the dot plot.

`dot_plot_margin_right`
: Numeric specifying the right margin of the dot plot.

`dot_xlab`
: Character string specifying the x-axis label for the dot plot.

`dot_size_title`
: Character string specifying the dot size legend title.

`dot_color_title`
: Character string specifying the dot color legend title.

`label_size`
: Numeric specifying label text size.

`label_padding_lines`
: Numeric specifying vertical padding between labels.

`legend_box`
: Character string specifying legend box orientation.

`legend_box_just`
: Character string specifying legend box justification.

`legend_position`
: Character string specifying legend position.

Details

`typoClustVis()` displays the top-ranked cell types per cluster or cell subset, optionally refining annotations to deeper hierarchical levels. The visualization integrates categorical labels, quantitative annotation scores, marker support, and purity metrics.

Value

A `ggplot2` object representing the combined visualization.

See Also

`typoClust`, `signatureDotHeatmap`

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

cc <- clustoCell(
  data = pbmc_small,
  identify_subclusters = FALSE,
  num_threads = 1,
  verbose = FALSE
)

desired_sets <- utils::head(
  sort(unique(as.character(cc$clusters$major_clusters))),
  2
)

tc <- typoClust(
  objects = list(cc),
  desired_sets = desired_sets,
  tissue = "Blood",
  condition = "Healthy",
  use_neg_markers = FALSE,
  thresh = 10,
  mode = "markerDB",
  species = "human",
  verbose = FALSE
)

p <- typoClustVis(
  typoClust = tc,
  rank_thresh = 1,
  refine = FALSE,
  order_by = "Cluster"
)

p
```

---

typoPrompt *Generate an LLM-ready prompt for cell-type annotation*

Description

Generates a structured, copy-and-paste-ready prompt for annotation of clusters, sub-clusters, or cell subsets using any chatbot or large language model (LLM). Marker information is extracted directly from a `ClustoCell` or `MarkoCell` object and combined with optional biological context such as sample source, tissue, condition, and species.

In interactive sessions, the returned `TypoPrompt` object opens as a polished HTML interface in the RStudio Viewer, or in the default web browser when the Viewer is unavailable. The interface provides formatted and raw views, collapsible sections, light/dark appearance, one-click copying,

TXT/HTML downloads, and printing to PDF. The prompt can also be accessed directly as plain text or saved programmatically as `.txt` or `.html`.

Usage

```
typoPrompt(
  object,
  desired_sets = NULL,
  sample_source = NULL,
  feature_type = "gene",
  species = "human",
  tissue = NULL,
  condition = NULL,
  use_neg_markers = TRUE,
  thresh_mode = c("n", "rank"),
  thresh = 20,
  top_k = 3,
  verbose = TRUE
)
```

Arguments

`object`	An object of class `ClustoCell` or `MarkoCell` containing marker results for the clusters, sub-clusters, and/or cell subsets to be annotated.
`desired_sets`	Optional character vector specifying the names of clusters, sub-clusters, and/or cell subsets to include in the prompt. Names must be present in `object`. If `NULL`, all available sets are included.
`sample_source`	Optional free-text description of the sample origin provided to the LLM (e.g. `"human peripheral blood"` or `"melanoma tumor biopsy"`). Providing this context can help improve annotation specificity.
`feature_type`	Character string describing the marker feature type (e.g. `"gene"` or `"protein"`). Default is `"gene"`.
`species`	Character string specifying the species (e.g. `"human"` or `"mouse"`). Other species names may also be supplied. Default is `"human"`.
`tissue`	Optional character vector specifying one or more tissue contexts to provide to the LLM. Available tissue types can be accessed using `data("tissueCondition_types", package = "celliverse")`.
`condition`	Optional character vector specifying one or more biological or disease conditions to provide to the LLM. Available condition types can be accessed using `data("tissueCondition_types", package = "celliverse")`.
`use_neg_markers`	Logical; whether to include negative markers in the generated prompt. Negative markers provide exclusionary evidence that can help distinguish closely related cell types, subtypes, or states. Default is `TRUE`.
`thresh_mode`	Character string specifying how markers are selected from each marker table. One of:

- `"n"`: retains strictly the first `thresh` markers in rank order, even when additional markers share the final selected rank.
- `"rank"`: retains all markers with rank less than or equal to `thresh`. Ties at the cutoff rank are therefore retained.

Default is `"n"`.

`thresh` Integer specifying the marker-selection threshold. With `thresh_mode = "n"`, up to the first `thresh` markers are used for each set. With `thresh_mode = "rank"`, all markers with rank less than or equal to `thresh` are used. Default is `20`.

`top_k` Positive integer specifying the maximum number of ranked candidate annotations that the generated prompt asks the LLM to return for each set. Default is `3`.

`verbose` Logical; whether to display progress and status messages while preparing the prompt. Default is `TRUE`.

Details

`typoPrompt()` provides a model- and provider-independent workflow for LLM-assisted cell annotation. Unlike `ceLLMarkup`, it does not connect to an LLM directly and therefore requires no API key, model configuration, or local LLM server. Instead, it prepares the annotation task for submission to the user's preferred chatbot or LLM.

The generated prompt includes:

- positive markers and, optionally, negative markers for each set;
- available sample, tissue, condition, species, and feature context;
- instructions to consider the complete marker profile rather than individual markers;
- instructions to distinguish cell types, subtypes, and cellular states where supported by the marker evidence;
- a request for up to `top_k` ranked annotations with confidence scores and concise biological rationales; and
- a standardized Markdown-table response format followed by an overall interpretation.

For `ClustoCell` objects containing both major clusters and sub-clusters, the prompt additionally describes their hierarchy and asks the LLM to first establish the identity of each parent cluster and then interpret its sub-clusters as biologically meaningful subtypes or states within that context.

Printing a returned `TypoPrompt` object displays its formatted HTML interface:

```
prompt <- typoPrompt(...)
prompt
```

The underlying plain-text prompt remains directly accessible with `cat(prompt)` or `as.character(prompt)`. It can also be saved programmatically using `saveTypoPrompt` with `format = "txt"` or `format = "html"`. Interactive HTML rendering and HTML export require the optional **htmltools** package; when it is unavailable, printing falls back to the plain-text prompt.

Value

An object of class `TypoPrompt`, inheriting from `character`, that contains the complete LLM-ready annotation prompt.

In an interactive session, printing the object opens a formatted HTML interface in the RStudio Viewer when available, otherwise in the default web browser. The interface provides formatted and raw views, collapsible sections, light/dark appearance, one-click copying, TXT/HTML downloads, and printing to PDF. In non-interactive sessions, the plain-text prompt is printed instead. The raw prompt can also be obtained with `as.character()` or `cat()`, and exported programmatically with `saveTypoPrompt`.

See Also

`typoClust`, `ceLLMarkup`, `typoClustVis`, `saveTypoPrompt`, `markoCell`, `markoClust`, `clustoCell`

Examples

```
utils::data("pbmc_small", package = "SeuratObject")

cc <- clustoCell(
  data = pbmc_small,
  identify_subclusters = FALSE,
  num_threads = 1,
  verbose = FALSE
)

desired_set <- utils::head(
  sort(unique(as.character(cc$clusters$major_clusters))),
  1
)

prompt <- typoPrompt(
  object = cc,
  desired_sets = desired_set,
  sample_source = "human peripheral blood",
  tissue = "Blood",
  condition = "Healthy",
  species = "human",
  use_neg_markers = FALSE,
  thresh = 10,
  top_k = 3,
  verbose = FALSE
)

class(prompt)
```

Index