Package: scDiagnostics (via r-universe)

October 26, 2024

Type Package

Title Cell type annotation diagnostics

Version 0.99.11

Description The scDiagnostics package provides diagnostic plots to assess the quality of cell type assignments from single cell gene expression profiles. The implemented functionality allows to assess the reliability of cell type annotations, investigate gene expression patterns, and explore relationships between different cell types in query and reference datasets allowing users to detect potential misalignments between reference and query datasets. The package also provides visualization capabilities for diagnostics purposes.

License Artistic-2.0

URL https://github.com/ccb-hms/scDiagnostics

BugReports https://github.com/ccb-hms/scDiagnostics/issues

Depends R (>= 4.4.0)

Imports SingleCellExperiment, methods, isotree, ggplot2, ggridges, SummarizedExperiment, ranger, transport, speedglm, cramer, rlang, bluster, patchwork

Suggests AUCell, BiocStyle, knitr, Matrix, rmarkdown, scran, scRNAseq, SingleR, celldex, scuttle, scater, dplyr, testthat (>= 3.0.0)

VignetteBuilder knitr

biocViews Annotation, Classification, Clustering, GeneExpression, RNASeq, SingleCell, Software, Transcriptomics

Encoding UTF-8

LazyDataCompression xz

RoxygenNote 7.3.2

Config/testthat/edition 3

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

RemoteUrl https://github.com/bioc/scDiagnostics

RemoteRef HEAD

RemoteSha c58572e51c80ba9334e75e1cb1bc42fc90d5dc76

Contents

`boxplotPCA` *Plot Principal Components for Different Cell Types*

Description

This function generates a `ggplot2` boxplot visualization of principal components (PCs) for different cell types across two datasets (query and reference).

Usage

```
boxplotPCA(
  query_data,
  reference_data,
  query_cell_type_col,
  ref_cell_type_col,
  cell_types = NULL,
  pc_subset = 1:5,
  assay_name = "logcounts"
)
```

Arguments

query_data	A SingleCellExperiment object containing numeric expression matrix for the query cells.
reference_data	A SingleCellExperiment object containing numeric expression matrix for the reference cells.
query_cell_type_col	The column name in the colData of query_data that identifies the cell types.
ref_cell_type_col	The column name in the colData of reference_data that identifies the cell types.
cell_types	A character vector specifying the cell types to include in the plot. If NULL, all cell types are included.
pc_subset	A numeric vector specifying which principal components to include in the plot. Default is PC1 to PC5.
assay_name	Name of the assay on which to perform computations. Default is "logcounts".

Details

The function boxplotPCA is designed to provide a visualization of principal component analysis (PCA) results. It projects the query dataset onto the principal components obtained from the reference dataset. The results are then visualized as boxplots, grouped by cell types and datasets (query and reference). This allows for a comparative analysis of the distributions of the principal components across different cell types and datasets. The function internally calls projectPCA to perform the PCA projection. It then reshapes the output data into a long format suitable for ggplot2 plotting.

Value

A ggplot object representing the boxplots of specified principal components for the given cell types and datasets.

Author(s)

Anthony Christidis, <anthony-alexander_christidis@hms.harvard.edu>

Examples

```
# Load data
data("reference_data")
data("query_data")

# Plot the PC data
pc_plot <- boxplotPCA(query_data = query_data,
                      reference_data = reference_data,
                      cell_types = c("CD4", "CD8", "B_and_plasma", "Myeloid"),
                      query_cell_type_col = "SingleR_annotation",
                      ref_cell_type_col = "expert_annotation",
                      pc_subset = 1:6)
pc_plot
```

`calculateCategorizationEntropy` *Calculate Categorization Entropy*

Description

This function takes a matrix of category scores (cell type by cells) and calculates the entropy of the category probabilities for each cell. This gives a sense of how confident the cell type assignments are. High entropy = lots of plausible category assignments = low confidence. Low entropy = only one or two plausible categories = high confidence. This is confidence in the vernacular sense, not in the "confidence interval" statistical sense. Also note that the entropy tells you nothing about whether or not the assignments are correct – see the other functionality in the package for that. This functionality can be used for assessing how comparatively confident different sets of assignments are (given that the number of categories is the same).

Usage

```
calculateCategorizationEntropy(
  X,
  inverse_normal_transform = FALSE,
  plot = TRUE,
  verbose = TRUE
)
```

Arguments

`X`	A matrix of category scores.
`inverse_normal_transform`	If TRUE, apply inverse normal transformation to X. Default is FALSE.
`plot`	If TRUE, plot a histogram of the entropies. Default is TRUE.
`verbose`	If TRUE, display messages about the calculations. Default is TRUE.

Details

The function checks if X is already on the probability scale. Otherwise, it applies softmax column-wise.

You can think about entropies on a scale from 0 to a maximum that depends on the number of categories. This is the function for entropy (minus input checking): `entropy(p) = -sum(p*log(p))` . If that input vector p is a uniform distribution over the `length(p)` categories, the entropy will be a high as possible.

Value

A vector of entropy values for each column in X.

Author(s)

Andrew Ghazi, <andrew_ghazi@hms.harvard.edu>

Examples

```
# Simulate 500 cells with scores on 4 possible cell types
X <- rnorm(500 * 4) |> matrix(nrow = 4)
X[1, 1:250] <- X[1, 1:250] + 5 # Make the first category highly scored in the first 250 cells

# The function will issue a message about softmaxing the scores, and the entropy histogram will be
# bimodal since we made half of the cells clearly category 1 while the other half are roughly even.
entropy_scores <- calculateCategorizationEntropy(X)
```

---

calculateCellDistancesSimilarity

Function to Calculate Bhattacharyya Coefficients and Hellinger Distances

---

Description

This function computes Bhattacharyya coefficients and Hellinger distances to quantify the similarity of density distributions between query cells and reference data for each cell type.

Usage

```
calculateCellDistancesSimilarity(
  query_data,
  reference_data,
  query_cell_type_col,
  ref_cell_type_col,
  cell_names,
  pc_subset = 1:5,
  assay_name = "logcounts"
)
```

Arguments

query_data	A SingleCellExperiment object containing numeric expression matrix for the query cells.
reference_data	A SingleCellExperiment object containing numeric expression matrix for the reference cells.
query_cell_type_col	The column name in the colData of query_data that identifies the cell types.
ref_cell_type_col	The column name in the colData of reference_data that identifies the cell types.

cell_names	A character vector specifying the names of the query cells for which to compute distance measures.
pc_subset	A numeric vector specifying which principal components to include in the plot. Default is 1:5.
assay_name	Name of the assay on which to perform computations. Default is "logcounts".

Details

This function first computes distance data using the calculateCellDistances function, which calculates pairwise distances between cells within the reference data and between query cells and reference cells in the PCA space. Bhattacharyya coefficients and Hellinger distances are calculated to quantify the similarity of density distributions between query cells and reference data for each cell type. Bhattacharyya coefficient measures the similarity of two probability distributions, while Hellinger distance measures the distance between two probability distributions.

Bhattacharyya coefficients range between 0 and 1. A value closer to 1 indicates higher similarity between distributions, while a value closer to 0 indicates lower similarity

Hellinger distances range between 0 and 1. A value closer to 0 indicates higher similarity between distributions, while a value closer to 1 indicates lower similarity.

Value

A list containing distance data for each cell type. Each entry in the list contains:

ref_distances A vector of all pairwise distances within the reference subset for the cell type.

query_to_ref_distances A matrix of distances from each query cell to all reference cells for the cell type.

Author(s)

Anthony Christidis, <anthony-alexander_christidis@hms.harvard.edu>

Examples

```
# Load data
data("reference_data")
data("query_data")

# Plot the PC data
distance_data <- calculateCellDistances(query_data = query_data,
                                        reference_data = reference_data,
                                        query_cell_type_col = "SingleR_annotation",
                                        ref_cell_type_col = "expert_annotation",
                                        pc_subset = 1:10)

# Identify outliers for CD4
cd4_anomalies <- detectAnomaly(reference_data = reference_data,
                               query_data = query_data,
                               query_cell_type_col = "SingleR_annotation",
                               ref_cell_type_col = "expert_annotation",
                               pc_subset = 1:10,
```

```
                                n_tree = 500,
                                anomaly_treshold = 0.5)
cd4_top6_anomalies <- names(sort(cd4_anomalies$CD4$query_anomaly_scores, decreasing = TRUE)[1:6])

# Get overlap measures
overlap_measures <- calculateCellDistancesSimilarity(query_data = query_data,
                                                     reference_data = reference_data,
                                                     cell_names = cd4_top6_anomalies,
                                                query_cell_type_col = "SingleR_annotation",
                                                   ref_cell_type_col = "expert_annotation",
                                                     pc_subset = 1:10)
overlap_measures
```

`calculateCramerPValue` *Calculate Cramer Test P-Values for Two-Sample Comparison of Multivariate ECDFs*

Description

This function performs the Cramer test for comparing multivariate empirical cumulative distribution functions (ECDFs) between two samples.

Usage

```
calculateCramerPValue(
  reference_data,
  query_data = NULL,
  ref_cell_type_col,
  query_cell_type_col = NULL,
  cell_types = NULL,
  pc_subset = 1:5,
  assay_name = "logcounts"
)
```

Arguments

`reference_data`	A `SingleCellExperiment` object containing numeric expression matrix for the reference cells.
`query_data`	A `SingleCellExperiment` object containing numeric expression matrix for the query cells. If NULL, the PC scores are regressed against the cell types of the reference data.
`ref_cell_type_col`	The column name in the `colData` of `reference_data` that identifies the cell types.
`query_cell_type_col`	The column name in the `colData` of `query_data` that identifies the cell types.

cell_types	A character vector specifying the cell types to include in the plot. If NULL, all cell types are included.
pc_subset	A numeric vector specifying which principal components to include in the plot. Default is PC1 to PC5.
assay_name	Name of the assay on which to perform computations. Default is "logcounts".

Details

The function performs the following steps:

1. Projects the data into the PCA space.
2. Subsets the data to the specified cell types and principal components.
3. Performs the Cramer test for each cell type using the `cramer.test` function in the `cramer` package.

Value

A named vector of p-values from the Cramer test for each cell type.

References

Baringhaus, L., & Franz, C. (2004). "On a new multivariate two-sample test". Journal of Multivariate Analysis, 88(1), 190-206.

Examples

```
# Load data
data("reference_data")
data("query_data")

# Plot the PC data (with query data)
cramer_test <- calculateCramerPValue(reference_data = reference_data,
                                     query_data = query_data,
                                     ref_cell_type_col = "expert_annotation",
                                     query_cell_type_col = "SingleR_annotation",
                                   cell_types = c("CD4", "CD8", "B_and_plasma", "Myeloid"),
                                     pc_subset = 1:5)
cramer_test
```

`calculateHotellingPValue` *Perform Hotelling's T-squared Test on PCA Scores for Single-cell RNA-seq Data*

Description

Computes Hotelling's T-squared test statistic and p-values for each specified cell type based on PCA-projected data from query and reference datasets.

Usage

```
calculateHotellingPValue(
  query_data,
  reference_data,
  query_cell_type_col,
  ref_cell_type_col,
  cell_types = NULL,
  pc_subset = 1:5,
  n_permutation = 500,
  assay_name = "logcounts"
)
```

Arguments

query_data	A SingleCellExperiment object containing numeric expression matrix for the query cells.
reference_data	A SingleCellExperiment object containing numeric expression matrix for the reference cells.
query_cell_type_col	character. The column name in the colData of query_data that identifies the cell types.
ref_cell_type_col	character. The column name in the colData of reference_data that identifies the cell types.
cell_types	A character vector specifying the cell types to include in the plot. If NULL, all cell types are included.
pc_subset	A numeric vector specifying which principal components to include in the plot. Default is PC1 to PC5.
n_permutation	Number of permutations to perform for p-value calculation. Default is 500.
assay_name	Name of the assay on which to perform computations. Default is "logcounts".

Details

This function calculates Hotelling's T-squared statistic for comparing multivariate means between reference and query datasets, projected onto a subset of principal components (PCs). It performs a permutation test to obtain p-values for each cell type specified.

Value

A named numeric vector of p-values from Hotelling's T-squared test for each cell type.

Author(s)

Anthony Christidis, <anthony-alexander_christidis@hms.harvard.edu>

References

Hotelling, H. (1931). "The generalization of Student's ratio". *Annals of Mathematical Statistics*. 2 (3): 360–378. doi:10.1214/aoms/1177732979.

Examples

```
# Load data
data("reference_data")
data("query_data")

# Get the p-values
p_values <- calculateHotellingPValue(query_data = query_data,
                                     reference_data = reference_data,
                                     query_cell_type_col = "SingleR_annotation",
                                     ref_cell_type_col = "expert_annotation",
                                     pc_subset = 1:10)
round(p_values, 5)
```

`calculateHVGOverlap` *Calculate the Overlap Coefficient for Highly Variable Genes*

Description

Calculates the overlap coefficient between the sets of highly variable genes from a reference dataset and a query dataset.

Usage

```
calculateHVGOverlap(reference_genes, query_genes)
```

Arguments

`reference_genes`	A character vector of highly variable genes from the reference dataset.
`query_genes`	A character vector of highly variable genes from the query dataset.

Details

The overlap coefficient measures the similarity between two gene sets, indicating how well-aligned reference and query datasets are in terms of their highly variable genes. This metric is useful in single-cell genomics to understand the correspondence between different datasets.

The coefficient is calculated using the formula:

$$Coefficient(X, Y) = \frac{|X \cap Y|}{min(|X|, |Y|)}$$

where X and Y are the sets of highly variable genes from the reference and query datasets, respectively, $|X \cap Y|$ is the number of genes common to both X and Y, and $min(|X|, |Y|)$ is the size of the smaller set among X and Y.

Value

Overlap coefficient, a value between 0 and 1, where 0 indicates no overlap and 1 indicates complete overlap of highly variable genes between datasets.

Author(s)

Anthony Christidis, <anthony-alexander_christidis@hms.harvard.edu>

References

Luecken et al. Benchmarking atlas-level data integration in single-cell genomics. Nature Methods, 19:41-50, 2022.

Examples

```
# Load data
data("reference_data")
data("query_data")

# Selecting highly variable genes
ref_var <- scran::getTopHVGs(reference_data, n = 500)
query_var <- scran::getTopHVGs(query_data, n = 500)
overlap_coefficient <- calculateHVGOverlap(reference_genes = ref_var,
                                           query_genes = query_var)
overlap_coefficient
```

---

calculateVarImpOverlap

Compare Gene Importance Across Datasets Using Random Forest

Description

This function identifies and compares the most important genes for differentiating cell types between a query dataset and a reference dataset using Random Forest.

Usage

```
calculateVarImpOverlap(
  reference_data,
  query_data = NULL,
  ref_cell_type_col,
  query_cell_type_col = NULL,
  cell_types = NULL,
  n_tree = 500,
  n_top = 50
)
```

Arguments

reference_data	A SingleCellExperiment object containing numeric expression matrix for the reference cells.
query_data	A SingleCellExperiment object containing numeric expression matrix for the query cells. If NULL, then the variable importance scores are only computed for the reference data. Default is NULL.
ref_cell_type_col	A character string specifying the column name in the reference dataset containing cell type annotations.
query_cell_type_col	A character string specifying the column name in the query dataset containing cell type annotations.
cell_types	A character vector specifying the cell types to include in the plot. If NULL, all cell types are included.
n_tree	An integer specifying the number of trees to grow in the Random Forest. Default is 500.
n_top	An integer specifying the number of top genes to consider when comparing variable importance scores. Default is 50.

Details

This function uses the Random Forest algorithm to calculate the importance of genes in differentiating between cell types within both a reference dataset and a query dataset. The function then compares the top genes identified in both datasets to determine the overlap in their importance scores.

Value

A list containing three elements:

var_imp_ref	A list of data frames containing variable importance scores for each combination of cell types in the reference dataset.
var_imp_query	A list of data frames containing variable importance scores for each combination of cell types in the query dataset.
var_imp_comparison	A named vector indicating the proportion of top genes that overlap between the reference and query datasets for each combination of cell types.

Author(s)

Anthony Christidis, <anthony-alexander_christidis@hms.harvard.edu>

References

Breiman, L. (2001). "Random forests". *Machine Learning*, 45(1), 5-32. doi:10.1023/A:1010933404324.

Examples

```
# Load data
data("reference_data")
data("query_data")

# Compute important variables for all pairwise cell comparisons
rf_output <- calculateVarImpOverlap(reference_data = reference_data,
                                    query_data = query_data,
                                    query_cell_type_col = "SingleR_annotation",
                                    ref_cell_type_col = "expert_annotation",
                                    n_tree = 500,
                                    n_top = 50)

# Comparison table
rf_output$var_imp_comparison
```

`calculateWassersteinDistance` *Compute Wasserstein Distances Between Query and Reference Datasets*

Description

This function calculates Wasserstein distances between a query dataset and a reference dataset, as well as within the reference dataset itself, after projecting them into a shared PCA space.

Usage

```
calculateWassersteinDistance(
  query_data,
  reference_data,
  ref_cell_type_col,
  query_cell_type_col,
  pc_subset = 1:5,
  n_resamples = 300,
  assay_name = "logcounts"
)
```

Arguments

`query_data`	A `SingleCellExperiment` object containing a numeric expression matrix for the query cells.
`reference_data`	A `SingleCellExperiment` object with a numeric expression matrix for the reference cells.
`ref_cell_type_col`	The column name in the `colData` of `reference_data` that identifies cell types.

query_cell_type_col
: The column name in the colData of query_data that identifies cell types.

pc_subset
: A numeric vector specifying which principal components to use. Default is 1:10.

n_resamples
: An integer specifying the number of resamples to generate the null distribution. Default is 300.

assay_name
: The name of the assay to use for computations. Default is "logcounts".

Details

The function begins by projecting the query dataset onto the PCA space defined by the reference dataset. It then computes Wasserstein distances between randomly sampled pairs within the reference dataset to create a null distribution. Similarly, it calculates distances between the reference and query datasets. The function assesses overall differences in distances to understand the variation between the datasets.

Value

A list with the following components:

null_dist
: A numeric vector of Wasserstein distances computed from resampled pairs within the reference dataset.

query_dist
: The mean Wasserstein distance between the query dataset and the reference dataset.

cell_type
: A character vector containing the unique cell types present in the reference dataset.

References

Schuhmacher, D., Bernhard, S., & Book, M. (2019). "A Review of Approximate Transport in Machine Learning". In *Journal of Machine Learning Research* (Vol. 20, No. 117, pp. 1-61).

See Also

plot.calculateWassersteinDistanceObject

Examples

```
# Load data
data("reference_data")
data("query_data")

# Extract CD4 cells
ref_data_subset <- reference_data[, which(reference_data$expert_annotation == "CD4")]
query_data_subset <- query_data[, which(query_data$expert_annotation == "CD4")]

# Selecting highly variable genes (can be customized by the user)
ref_top_genes <- scran::getTopHVGs(ref_data_subset, n = 500)
query_top_genes <- scran::getTopHVGs(query_data_subset, n = 500)
```

```
# Intersect the gene symbols to obtain common genes
common_genes <- intersect(ref_top_genes, query_top_genes)
ref_data_subset <- ref_data_subset[common_genes,]
query_data_subset <- query_data_subset[common_genes,]

# Run PCA on reference data
ref_data_subset <- scater::runPCA(ref_data_subset)

# Compute Wasserstein distances and compare using quantile-based permutation test
wasserstein_data <- calculateWassersteinDistance(query_data = query_data_subset,
                                                 reference_data = ref_data_subset,
                                               query_cell_type_col = "expert_annotation",
                                                 ref_cell_type_col = "expert_annotation",
                                                  pc_subset = 1:5,
                                                  n_resamples = 100)
plot(wasserstein_data)
```

`histQCvsAnnotation` *Histograms: QC Stats and Annotation Scores Visualization*

Description

This function generates histograms for visualizing the distribution of quality control (QC) statistics and annotation scores associated with cell types in single-cell genomic data.

Usage

```
histQCvsAnnotation(
  se_object,
  cell_type_col,
  cell_types = NULL,
  qc_col,
  score_col
)
```

Arguments

`se_object`	A `SingleCellExperiment` containing the single-cell expression data and metadata.
`cell_type_col`	The column name in the `colData` of `se_object` that contains the cell type labels.
`cell_types`	A vector of cell types to plot (e.g., c("T-cell", "B-cell")). Defaults to `NULL`, which will include all the cells.
`qc_col`	A column name in the `colData` of `se_object` that contains the QC stats of interest.
`score_col`	The column name in the `colData` of `se_object` that contains the cell type scores.

Details

The particularly useful in the analysis of data from single-cell experiments, where understanding the distribution of these metrics is crucial for quality assessment and interpretation of cell type annotations.

Value

A object containing two histograms displayed side by side. The first histogram represents the distribution of QC stats, and the second histogram represents the distribution of annotation scores.

Examples

```
data("query_data")

# Generate histograms
histQCvsAnnotation(se_object = query_data,
                   cell_type_col = "SingleR_annotation",
                   cell_types = c("CD4", "CD8"),
                   qc_col = "percent_mito",
                   score_col = "annotation_scores")

histQCvsAnnotation(se_object = query_data,
                   cell_type_col = "SingleR_annotation",
                   cell_types = NULL,
                   qc_col = "percent_mito",
                   score_col = "annotation_scores")
```

---

plotCellTypeMDS *Plot Reference and Query Cell Types using MDS*

Description

This function facilitates the assessment of similarity between reference and query datasets through Multidimensional Scaling (MDS) scatter plots. It allows the visualization of cell types, color-coded with user-defined custom colors, based on a dissimilarity matrix computed from a user-selected gene set.

Usage

```
plotCellTypeMDS(
  query_data,
  reference_data,
  query_cell_type_col,
  ref_cell_type_col,
  cell_types = NULL,
  assay_name = "logcounts"
)
```

Arguments

query_data	A SingleCellExperiment containing the single-cell expression data and metadata.
reference_data	A SingleCellExperiment object containing the single-cell expression data and metadata.
query_cell_type_col	The column name in the colData of query_data that identifies the cell types.
ref_cell_type_col	The column name in the colData of reference_data that identifies the cell types.
cell_types	A character vector specifying the cell types to include in the plot. If NULL, all cell types are included.
assay_name	Name of the assay on which to perform computations. Default is "logcounts".

Details

To evaluate dataset similarity, the function selects specific subsets of cells from both reference and query datasets. It then calculates Spearman correlations between gene expression profiles, deriving a dissimilarity matrix. This matrix undergoes Classical Multidimensional Scaling (MDS) for visualization, presenting cell types in a scatter plot, distinguished by colors defined by the user.

Value

A ggplot object representing the MDS scatter plot with cell type coloring.

Author(s)

Anthony Christidis, <anthony-alexander_christidis@hms.harvard.edu>

References

- Kruskal, J. B. (1964). "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis". *Psychometrika*, 29(1), 1-27. doi:10.1007/BF02289565.
- Borg, I., & Groenen, P. J. F. (2005). *Modern multidimensional scaling: Theory and applications* (2nd ed.). Springer Science & Business Media. doi:10.1007/978-0-387-25975-1.

Examples

```
# Load data
data("reference_data")
data("query_data")

# Generate the MDS scatter plot with cell type coloring
mds_plot <- plotCellTypeMDS(query_data = query_data,
                            reference_data = reference_data,
                           cell_types = c("CD4", "CD8", "B_and_plasma", "Myeloid")[1:4],
                            query_cell_type_col = "SingleR_annotation",
                            ref_cell_type_col = "expert_annotation")
```

```
mds_plot
```

`plotCellTypePCA` *Plot Principal Components for Different Cell Types*

Description

This function plots the principal components for different cell types in the query and reference datasets.

Usage

```
plotCellTypePCA(
  query_data,
  reference_data,
  query_cell_type_col,
  ref_cell_type_col,
  cell_types = NULL,
  pc_subset = 1:5,
  assay_name = "logcounts"
)
```

Arguments

`query_data`	A `SingleCellExperiment` object containing numeric expression matrix for the query cells.
`reference_data`	A `SingleCellExperiment` object containing numeric expression matrix for the reference cells.
`query_cell_type_col`	The column name in the `colData` of `query_data` that identifies the cell types.
`ref_cell_type_col`	The column name in the `colData` of `reference_data` that identifies the cell types.
`cell_types`	A character vector specifying the cell types to include in the plot. If NULL, all cell types are included.
`pc_subset`	A numeric vector specifying which principal components to include in the plot. Default is 1:5.
`assay_name`	Name of the assay on which to perform computations. Default is "logcounts".

Details

This function projects the query dataset onto the principal component space of the reference dataset and then plots the specified principal components for the specified cell types. It uses the 'project-PCA' function to perform the projection and `ggplot2` to create the plots.

Value

A ggplot object representing the boxplots of specified principal components for the given cell types and datasets.

Author(s)

Anthony Christidis, <anthony-alexander_christidis@hms.harvard.edu>

Examples

```
# Load data
data("reference_data")
data("query_data")

# Plot the PC data
pc_plot <- plotCellTypePCA(query_data = query_data,
                           reference_data = reference_data,
                           cell_types = c("CD4", "CD8", "B_and_plasma", "Myeloid"),
                           query_cell_type_col = "expert_annotation",
                           ref_cell_type_col = "expert_annotation",
                           pc_subset = 1:5)
pc_plot
```

---

plotGeneExpressionDimred

Visualize gene expression on a dimensional reduction plot

Description

This function plots gene expression on a dimensional reduction plot using methods like t-SNE, UMAP, or PCA. Each single cell is color-coded based on the expression of a specific gene or feature.

Usage

```
plotGeneExpressionDimred(
  se_object,
  method = c("TSNE", "UMAP", "PCA"),
  pc_subset = 1:5,
  feature,
  assay_name = "logcounts"
)
```

Arguments

se_object	An object of class SingleCellExperiment containing log-transformed expression matrix and other metadata. It can be either a reference or query dataset.
method	The reduction method to use for visualization. It should be one of the supported methods: "TSNE", "UMAP", or "PCA".
pc_subset	An optional vector specifying the principal components (PCs) to include in the plot if method = "PCA". Default is 1:5.
feature	A character string representing the name of the gene or feature to be visualized.
assay_name	Name of the assay on which to perform computations. Default is "logcounts".

Value

A ggplot object representing the dimensional reduction plot with gene expression.

Author(s)

Anthony Christidis, <anthony-alexander_christidis@hms.harvard.edu>

Examples

```
# Load data
data("query_data")

# Plot gene expression on PCA plot
plotGeneExpressionDimred(se_object = query_data,
                         method = "PCA",
                         pc_subset = 1:5,
                         feature = "VPREB3")
```

---

plotGeneSetScores *Visualization of gene sets or pathway scores on dimensional reduction plot*

Description

Plot gene sets or pathway scores on PCA, TSNE, or UMAP. Single cells are color-coded by scores of gene sets or pathways.

Usage

```
plotGeneSetScores(
  se_object,
  method = c("PCA", "TSNE", "UMAP"),
  score_col,
  pc_subset = 1:5
)
```

Arguments

se_object	An object of class SingleCellExperiment containing numeric expression matrix and other metadata. It can be either a reference or query dataset.
method	A character string indicating the method for visualization ("PCA", "TSNE", or "UMAP").
score_col	A character string representing the name of the score_col (score) in the colData(se_object) to plot.
pc_subset	An optional vector specifying the principal components (PCs) to include in the plot if method = "PCA". Default is 1:5.

Details

This function plots gene set scores on reduced dimensions such as PCA, t-SNE, or UMAP. It extracts the reduced dimensions from the provided SingleCellExperiment object. Gene set scores are visualized as a scatter plot with colors indicating the scores. For PCA, the function automatically includes the percentage of variance explained in the plot's legend.

Value

A ggplot2 object representing the gene set scores plotted on the specified reduced dimensions.

Author(s)

Anthony Christidis, <anthony-alexander_christidis@hms.harvard.edu>

Examples

```
# Load data
data("query_data")

# Plot gene set scores on PCA
plotGeneSetScores(se_object = query_data,
                  method = "PCA",
                  score_col = "gene_set_scores",
                  pc_subset = 1:5)

# Note: Users can provide their own gene set scores in the colData of the 'se_object' object,
# using any dimension reduction of their choice.
```

---

plotMarkerExpression *Plot gene expression distribution from overall and cell type-specific perspective*

Description

This function generates density plots to visualize the distribution of gene expression values for a specific gene across the overall dataset and within a specified cell type.

Usage

```
plotMarkerExpression(
  reference_data,
  query_data,
  ref_cell_type_col,
  query_cell_type_col,
  cell_type,
  gene_name,
  assay_name = "logcounts"
)
```

Arguments

reference_data	A SingleCellExperiment object containing numeric expression matrix for the reference cells.
query_data	A SingleCellExperiment object containing numeric expression matrix for the query cells.
ref_cell_type_col	The column name in the colData of reference_data that identifies the cell types.
query_cell_type_col	The column name in the colData of query_data that identifies the cell types.
cell_type	A vector of cell type cell_types to plot (e.g., c("T-cell", "B-cell")).
gene_name	The gene name for which the distribution is to be visualized.
assay_name	Name of the assay on which to perform computations. Default is "logcounts".

Details

This function generates density plots to compare the distribution of a specific marker gene between reference and query datasets. The aim is to inspect the alignment of gene expression levels as a surrogate for dataset similarity. Similar distributions suggest a good alignment, while differences may indicate discrepancies or incompatibilities between the datasets. To make the gene expression scales comparable between the datasets, the gene expression values are transformed using z-rank normalization. This transformation ranks the expression values and then scales the ranks to have a mean of 0 and a standard deviation of 1, which helps in standardizing the distributions for comparison.

Value

A gtable object containing two arranged density plots as grobs. The first plot shows the overall gene expression distribution, and the second plot displays the cell type-specific expression distribution.

Author(s)

Anthony Christidis, <anthony-alexander_christidis@hms.harvard.edu>

Examples

```
# Load data
data("reference_data")
data("query_data")

# Note: Users can use SingleR or any other method to obtain the cell type annotations.
plotMarkerExpression(reference_data = reference_data,
                     query_data = query_data,
                     ref_cell_type_col = "expert_annotation",
                     query_cell_type_col = "SingleR_annotation",
                     gene_name = "VPREB3",
                     cell_type = "B_and_plasma")
```

`plotPairwiseDistancesDensity` *Ridgeline Plot of Pairwise Distance Analysis*

Description

This function calculates pairwise distances or correlations between query and reference cells of a specified cell type and visualizes the results using ridgeline plots, displaying the density distribution for each comparison.

Usage

```
plotPairwiseDistancesDensity(
  query_data,
  reference_data,
  query_cell_type_col,
  ref_cell_type_col,
  cell_type_query,
  cell_type_ref,
  pc_subset = 1:5,
  distance_metric = c("correlation", "euclidean"),
  correlation_method = c("spearman", "pearson"),
  assay_name = "logcounts"
)
```

Arguments

`query_data`	A `SingleCellExperiment` containing the single-cell expression data and metadata.
`reference_data`	A `SingleCellExperiment` object containing the single-cell expression data and metadata.
`query_cell_type_col`	The column name in the `colData` of `query_data` that identifies the cell types.

`ref_cell_type_col`
: The column name in the `colData` of `reference_data` that identifies the cell types.

`cell_type_query`
: The query cell type for which distances or correlations are calculated.

`cell_type_ref`
: The reference cell type for which distances or correlations are calculated.

`pc_subset`
: A numeric vector specifying which principal components to use in the analysis. Default is 1:5. If set to `NULL`, the assay data is used directly for computations without dimensionality reduction.

`distance_metric`
: The distance metric to use for calculating pairwise distances, such as euclidean, manhattan, etc. Set to "correlation" to calculate correlation coefficients.

`correlation_method`
: The correlation method to use when `distance_metric` is "correlation". Possible values are "pearson" and "spearman".

`assay_name`
: Name of the assay on which to perform computations. Default is "logcounts".

Details

Designed for `SingleCellExperiment` objects, this function subsets data for specified cell types, computes pairwise distances or correlations, and visualizes these measurements through ridgeline plots. The plots help evaluate the consistency and differentiation of annotated cell types within single-cell datasets.

Value

A ggplot2 object showing ridgeline plots of calculated distances or correlations.

See Also

`calculateWassersteinDistance`

Examples

```
# Load data
data("reference_data")
data("query_data")

# Example usage of the function
plotPairwiseDistancesDensity(query_data = query_data,
                             reference_data = reference_data,
                             query_cell_type_col = "SingleR_annotation",
                             ref_cell_type_col = "expert_annotation",
                             cell_type_query = "CD8",
                             cell_type_ref = "CD8",
                             pc_subset = 1:5,
                             distance_metric = "euclidean",
                             correlation_method = "pearson")
```

`plotQCvsAnnotation` *Scatter plot: QC stats vs Cell Type Annotation Scores*

Description

Creates a scatter plot to visualize the relationship between QC stats (e.g., library size) and cell type annotation scores for one or more cell types.

Usage

```
plotQCvsAnnotation(
  se_object,
  cell_type_col,
  cell_types = NULL,
  qc_col,
  score_col
)
```

Arguments

`se_object`	A `SingleCellExperiment` containing the single-cell expression data and metadata.
`cell_type_col`	The column name in the `colData` of `se_object` that contains the cell type labels.
`cell_types`	A vector of cell type labels to plot (e.g., c("T-cell", "B-cell")). Defaults to `NULL`, which will include all the cells.
`qc_col`	A column name in the `colData` of `se_object` that contains the QC stats of interest.
`score_col`	The column name in the `colData` of `se_object` that contains the cell type annotation scores.

Details

This function generates a scatter plot to explore the relationship between various quality control (QC) statistics, such as library size and mitochondrial percentage, and cell type annotation scores. By examining these relationships, users can assess whether specific QC metrics, systematically influence the confidence in cell type annotations, which is essential for ensuring reliable cell type annotation.

Value

A ggplot object displaying a scatter plot of QC stats vs annotation scores, where each point represents a cell, color-coded by its cell type.

Examples

```
# Load data
data("qc_data")

p1 <- plotQCvsAnnotation(se_object = qc_data,
                         cell_type_col = "SingleR_annotation",
                         cell_types = NULL,
                         qc_col = "total",
                         score_col = "annotation_scores")
p1 + ggplot2::xlab("Library Size")
```

`projectPCA` *Project Query Data Onto PCA Space of Reference Data*

Description

This function projects a query singleCellExperiment object onto the PCA space of a reference singleCellExperiment object. The PCA analysis on the reference data is assumed to be pre-computed and stored within the object.

Usage

```
projectPCA(
  query_data,
  reference_data,
  query_cell_type_col,
  ref_cell_type_col,
  pc_subset = 1:10,
  assay_name = "logcounts"
)
```

Arguments

`query_data`	A `SingleCellExperiment` object containing numeric expression matrix for the query cells.
`reference_data`	A `SingleCellExperiment` object containing numeric expression matrix for the reference cells.
`query_cell_type_col`	character. The column name in the `colData` of `query_data` that identifies the cell types.
`ref_cell_type_col`	character. The column name in the `colData` of `reference_data` that identifies the cell types.
`pc_subset`	A numeric vector specifying the subset of principal components (PCs) to compare. Default is 1:10.
`assay_name`	Name of the assay on which to perform computations. Defaults to `"logcounts"`.

Details

This function assumes that the "PCA" element exists within the `reducedDims` of the reference data (obtained using `reducedDim(reference_data)`) and that the genes used for PCA are present in both the reference and query data. It performs centering and scaling of the query data based on the reference data before projection.

Value

A `data.frame` containing the projected data in rows (reference and query data combined).

Author(s)

Anthony Christidis, <anthony-alexander_christidis@hms.harvard.edu>

Examples

```
# Load data
data("reference_data")
data("query_data")

# Project the query data onto PCA space of reference
pca_output <- projectPCA(query_data = query_data,
                         reference_data = reference_data,
                         query_cell_type_col = "SingleR_annotation",
                         ref_cell_type_col = "expert_annotation",
                         pc_subset = 1:10)
```

---

projectSIR *Project Query Data Onto SIR Space of Reference Data*

Description

This function projects a query `SingleCellExperiment` object onto the SIR (supervised independent component) space of a reference `SingleCellExperiment` object. The SVD of the reference data is computed on conditional means per cell type, and the query data is projected based on these reference components.

Usage

```
projectSIR(
  query_data,
  reference_data,
  query_cell_type_col,
  ref_cell_type_col,
  cell_types = NULL,
  multiple_cond_means = TRUE,
  assay_name = "logcounts",
```

```
  cumulative_variance_threshold = 0.7,
  n_neighbor = 1
)
```

Arguments

query_data	A SingleCellExperiment object containing numeric expression matrix for the query cells.
reference_data	A SingleCellExperiment object containing numeric expression matrix for the reference cells.
query_cell_type_col	A character string specifying the column in the colData of query_data that identifies the cell types.
ref_cell_type_col	A character string specifying the column in the colData of reference_data that identifies the cell types.
cell_types	A character vector of cell types for which to compute conditional means in the reference data.
multiple_cond_means	A logical value indicating whether to compute multiple conditional means per cell type (through PCA and clustering). Defaults to TRUE.
assay_name	A character string specifying the assay name on which to perform computations. Defaults to "logcounts".
cumulative_variance_threshold	A numeric value between 0 and 1 specifying the variance threshold for PCA when computing multiple conditional means. Defaults to 0.7.
n_neighbor	An integer specifying the number of nearest neighbors for clustering when computing multiple conditional means. Defaults to 1.

Details

The genes used for the projection (SVD) must be present in both the reference and query datasets. The function first computes conditional means for each cell type in the reference data, then performs SVD on these conditional means to obtain the rotation matrix used for projecting both the reference and query datasets. The query data is centered and scaled based on the reference data.

Value

A list containing:

cond_means	A matrix of the conditional means computed for the reference data.
rotation_mat	The rotation matrix obtained from the SVD of the conditional means.
sir_projections	A data.frame containing the SIR projections for both the reference and query datasets.
percent_var	The percentage of variance explained by each component of the SIR projection.

Author(s)

Anthony Christidis, <anthony-alexander_christidis@hms.harvard.edu>

Examples

```
# Load data
data("reference_data")
data("query_data")

# Project the query data onto SIR space of reference
sir_output <- projectSIR(query_data = query_data,
                         reference_data = reference_data,
                         query_cell_type_col = "SingleR_annotation",
                         ref_cell_type_col = "expert_annotation")
```

Index