Package 'scDiagnostics'

April 10, 2025

Type Package

Title Cell type annotation diagnostics

Version 1.1.0

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

git_url https://git.bioconductor.org/packages/scDiagnostics

git_branch devel

git_last_commit 1b71ee2

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2025-04-09

Author Anthony Christidis [aut, cre] (ORCID:
<https://orcid.org/0000-0002-4565-6279>),
Andrew Ghazi [aut],
Smriti Chawla [aut],
Nitesh Turaga [ctb],
Ludwig Geistlinger [aut],
Robert Gentleman [aut]

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

Contents

`scDiagnostics-package` *scDiagnostics: Single-Cell Diagnostics Package*

Description

'scDiagnostics' is a comprehensive toolkit designed for the analysis and diagnostics of single-cell RNA sequencing (scRNA-seq) data. This package provides functionalities for comparing principal components, visualizing canonical correlation analysis (CCA) outputs, and plotting cell type-specific MDS and PCA projections.

Details

The package includes the following key functionalities, organized by their specific purposes:

Visualization of Cell Type Annotations

Functions for visualizing differences between query and reference datasets across multiple cell types.

- `boxplotPCA`: Boxplots of PCA scores for cell types.
- `calculateDiscriminantSpace`: Calculates discriminant space, with a plot method for visualization.
- `plotCellTypeMDS`: Creates MDS plots for cell types using query and reference datasets.
- `plotCellTypePCA`: Plots principal components for different cell types.

Visualization of Marker Expressions

Functions for to visualize and compare the expression of markers between a reference and a query dataset.

- `plotGeneExpressionDimred`: Plots gene expression in a dimensionality reduction space.
- `plotMarkerExpression`: Plots marker expression levels.

Visualization of QC and Annotation Scores

Functions for visualizing quality control (QC) metrics or other characteristics of the data.

- `histQCvsAnnotation`: Plots histograms of QC metrics versus annotations.
- `plotGeneSetScores`: Plots scores of gene sets.
- `plotQCvsAnnotation`: Plots QC metrics versus annotations.

Evaluation of Dataset Alignment

Functions for visualizing differences between query and reference datasets for a specific cell type.

- `compareCCA`: Compares CCA results, with a plot method for visualization.
- `comparePCA`: Compares PCA results, with a plot method for visualization.
- `comparePCASubspace`: Compares PCA subspace, with a plot method for visualization.
- `plotPairwiseDistancesDensity`: Plots the density of pairwise distances.
- `calculateWassersteinDistance`: Wasserstein distance for different cell types.

Evaluation of Marker Gene Alignment

Functions for calculating overlap measures of genes between two datasets.

- `calculateHVGOverlap`: Calculates overlap of highly variable genes (HVG) between datasets.
- `calculateVarImpOverlap`: Calculates overlap of variable importance measures between datasets.

Calculation of Statistical Measures to Compare Two Datasets

Functions to compute statistical measures to compare two datasets.

- `calculateCramerPValue`: Calculates the p-value using Cramer's V.
- `calculateHotellingPValue`: Calculates the p-value using Hotelling's T-squared test.
- `calculateNearestNeighborProbabilities`: Calculates nearest neighbor probabilities, with a plot method for visualization.
- `calculateAveragePairwiseCorrelation`: Calculates average pairwise correlation, with a plot method for visualization.
- `regressPC`: Performs regression on principal components, with a plot method for visualization.

Anomaly Detection (Global and Cell Type-Specific)

Functions for detecting anomalies at both the global and cell type-specific levels.

- `detectAnomaly`: Detects anomalies in the data, with a plot method for visualization.
- `calculateCellSimilarityPCA`: Calculates cell similarity in PCA space, with a plot method for visualization.

Calculation of Distances Between Specific Cells and Cell Populations

Functions for calculating distances between specific cells and cell populations.

- `calculateCellDistances`: Calculates distances between cells, with a plot method for visualization.
- `calculateCellDistancesSimilarity`: Calculates similarity based on cell distances, with a plot method for visualization.

Misc

Miscellaneous functions for various tasks.

- `projectPCA`: Projects new data into PCA space.
- `calculateCategorizationEntropy`: Calculates categorization entropy for clusters.

The package is built to facilitate in-depth analysis and visualization of single-cell data, enhancing the understanding of cell type similarities and differences across datasets.

Author(s)

Maintainer: Anthony Christidis <anthony-alexander_christidis@hms.harvard.edu> (ORCID)

Authors:

- Andrew Ghazi
- Smriti Chawla
- Ludwig Geistlinger
- Robert Gentleman

Other contributors:

- Nitesh Turaga [contributor]

See Also

Useful links:

- https://github.com/ccb-hms/scDiagnostics
- Report bugs at https://github.com/ccb-hms/scDiagnostics/issues

adjustPValues *Adjust P-Values in Regression Results*

Description

Adjusts the p-values in the regression results using a specified adjustment method. The adjustment is performed either for each principal component (PC) by cell type or for each dataset, depending on the selected independent variable.

Usage

```
adjustPValues(
  regress_res,
 adjust_method = c("BH", "holm", "hochberg", "hommel", "bonferroni", "BY", "fdr",
    "none"),
  indep_var = c("cell_type", "dataset")
)
```

Arguments

regress_res	A list containing regression results. The structure of the list depends on the indep_var argument: if indep_var is "cell_type", the list should contain regression summaries for each principal component (PC); if indep_var is "dataset", it should contain summaries for each dataset.
adjust_method	A character string specifying the method to adjust the p-values. Options include "BH", "holm", "hochberg", "hommel", "bonferroni", "BY", "fdr", or "none". Default is "BH" (Benjamini-Hochberg). Default is "BH".
indep_var	A character string specifying the independent variable for the adjustment. Options are "cell_type" (default) or "dataset".

Details

This function adjusts p-values from regression results stored in a list. The adjustment can be applied either across cell types or datasets, depending on the user's choice. The method for adjusting p-values can be selected from various options such as Benjamini-Hochberg (BH), Holm, and others, which are supported by the 'p.adjust' function in R.

Value

A list similar to regress_res, but with an added column for adjusted p-values in the coefficients tables.

Author(s)

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

argumentCheck *Argument Validation for SingleCellExperiment Analysis*

Description

This function validates the input arguments for functions that analyze SingleCellExperiment objects. It checks that the inputs are of the correct types and formats, and that required columns and cell types are present in the data.

Usage

```
argumentCheck(
  query_data = NULL,
  reference_data = NULL,
  query_cell_type_col = NULL,
  ref_cell_type_col = NULL,
  cell_types = NULL,
  unique_cell_type = FALSE,
  plot_function = FALSE,
  cell_names_query = NULL,
  cell_names_ref = NULL,
  pc_subset_query = NULL,
  pc_subset_ref = NULL,
  common_rotation_genes = FALSE,
  assay_name = NULL
)
```

Arguments

query_data	A SingleCellExperiment object containing numeric expression matrix for the query cells. If 'NULL', no check is performed.
reference_data	A SingleCellExperiment object containing numeric expression matrix for the reference cells. If 'NULL', no check is performed.
query_cell_type_col	The column name in the colData of query_data that identifies the cell types. If 'NULL', no check is performed.
ref_cell_type_col	The column name in the colData of reference_data that identifies the cell types. If 'NULL', no check is performed.
cell_types	A character vector specifying the cell types to include in the plot. If 'NULL', no check is performed.
unique_cell_type	If 'TRUE', there should only be one cell type in the provided SingleCellExperiment objectss. Default is 'FALSE'.
plot_function	A logical value indicating whether the function is being called to generate a plot. Default is 'FALSE'.

`cell_names_query`
: A character vector of cell names in query data to be analyzed. If 'NULL', no check is performed.

`cell_names_ref`
: A character vector of cell names in reference data to be analyzed. If 'NULL', no check is performed.

`pc_subset_query`
: A numeric vector specifying the principal components to be used for the query data. If 'NULL', no check is performed.

`pc_subset_ref`
: A numeric vector specifying the principal components to be used for the reference data. If 'NULL', no check is performed.

`common_rotation_genes`
: If TRUE, check the rotation matrices of the reference and query data and ensure they have the same genes. Default is FALSE.

`assay_name`
: Name of the assay on which to perform computations. If 'NULL', no check is performed.

Details

The function performs a series of checks to ensure that:

- 'query_data' and 'reference_data' are `SingleCellExperiment` objects.
- 'query_cell_type_col' and 'ref_cell_type_col' exist in the column data of their respective `SingleCellExperiment` objects.
- The specified 'cell_types' are available in the provided datasets.
- If 'unique_cell_type' is 'TRUE', there should only be one cell type in the `SingleCellExperiment` objects.
- If 'plot_function' is 'TRUE', the number of unique 'cell_types' does not exceed 10.
- 'cell_names_query' are valid cell names in the provided query dataset.
- 'cell_names_ref' are valid cell names in the provided reference dataset.
- The PCA subsets specified by 'pc_subset_query' and 'pc_subset_ref' are valid.

Value

None.

Author(s)

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

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
```

---

calculateAveragePairwiseCorrelation

Compute Average Pairwise Correlation between Cell Types

Description

Computes the average pairwise correlations between specified cell types in single-cell gene expression data.

The S3 plot method takes the output of the ‘calculateAveragePairwiseCorrelation‘ function, which should be a matrix of pairwise correlations, and plots it as a heatmap.

Usage

```
calculateAveragePairwiseCorrelation(
  query_data,
  reference_data,
  query_cell_type_col,
  ref_cell_type_col,
  cell_types = NULL,
  pc_subset = 1:10,
  correlation_method = c("spearman", "pearson"),
  assay_name = "logcounts"
)
```

```
## S3 method for class 'calculateAveragePairwiseCorrelationObject'
plot(x, ...)
```

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 use in the analysis. Default is 1:10. If set to NULL then no dimensionality reduction is performed and the assay data is used directly for computations.
correlation_method	The correlation method to use for calculating pairwise correlations.
assay_name	Name of the assay on which to perform computations. Default is "logcounts".
x	Output matrix from ‘calculateAveragePairwiseCorrelation‘ function.
...	Additional arguments to be passed to the plotting function.

Details

This function operates on SingleCellExperiment objects, ideal for single-cell analysis workflows. It calculates pairwise correlations between query and reference cells using a specified correlation method, then averages these correlations for each cell type pair. This function aids in assessing the similarity between cells in reference and query datasets, providing insights into the reliability of cell type annotations in single-cell gene expression data.

The S3 plot method converts the correlation matrix into a dataframe, creates a heatmap using ggplot2, and customizes the appearance of the heatmap with updated colors and improved aesthetics.

Value

A matrix containing the average pairwise correlation values. Rows and columns are labeled with the cell types. Each element in the matrix represents the average correlation between a pair of cell types.

The S3 plot method returns a ggplot object representing the heatmap plot.

Author(s)

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

See Also

plot.calculateAveragePairwiseCorrelationObject

calculateAveragePairwiseCorrelation

Examples

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

# Compute pairwise correlations
cor_matrix_avg <- calculateAveragePairwiseCorrelation(query_data = query_data,
                                                      reference_data = reference_data,
                                                query_cell_type_col = "SingleR_annotation",
                                                   ref_cell_type_col = "expert_annotation",
                                               cell_types = c("CD4", "CD8", "B_and_plasma"),
                                                      pc_subset = 1:10,
                                                      correlation_method = "spearman")

# Visualize correlation output
plot(cor_matrix_avg)
```

---

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)
```

---

calculateCellDistances

Compute Cell Distances Between Reference and Query Data

Description

This function computes the distances within the reference dataset and the distances from each query cell to all reference cells for each cell type. It uses PCA for dimensionality reduction and Euclidean distance for distance calculation.

The S3 plot method plots the density functions for the reference data and the distances from a specified query cells to all reference cell within a specified cell type.

Usage

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

## S3 method for class 'calculateCellDistancesObject'
plot(x, ref_cell_type, cell_names, ...)
```

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 1:5.
assay_name	Name of the assay on which to perform computations. Default is "logcounts".
x	A list containing the distance data computed by calculatecellDistances.
ref_cell_type	A string specifying the reference cell type.
cell_names	A string specifying the query cell name for which to plot the distances.
...	Additional arguments passed to the plotting function.

Details

The function first performs PCA on the reference dataset and projects the query dataset onto the same PCA space. It then computes pairwise Euclidean distances within the reference dataset for each cell type, as well as distances from each query cell to all reference cells of a particular cell type. The results are stored in a list, with one entry per cell type.

The S3 plot method first checks if the specified cell type and cell names are present in the object. If the specified cell type or cell name is not found, an error is thrown. It then extracts the distances within the reference dataset and the distances from the specified query cell to the reference cells The function creates a density plot using ggplot2 to compare the distance distributions. The density plot

will show two distributions: one for the pairwise distances within the reference dataset and one for the distances from the specified query cell to each reference cell. These distributions are plotted in different colors to visually assess how similar the query cell is to the reference cells of the specified cell type.

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.

The S3 plot method returns a ggplot density plot comparing the reference distances and the distances from the specified cell to the reference cells.

Author(s)

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

See Also

plot.calculateCellDistancesObject

calculateCellDistances

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])

# Plot the densities of the distances
plot(distance_data, ref_cell_type = "CD4", cell_names = cd4_top6_anomalies)
plot(distance_data, ref_cell_type = "CD8", cell_names = cd4_top6_anomalies)
```

`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
```

`calculateCellSimilarityPCA` *Calculate Cell Similarity Using PCA Loadings*

Description

This function calculates the cosine similarity between cells based on the principal components (PCs) obtained from PCA (Principal Component Analysis) loadings.

The S3 plot method creates a heatmap plot to visualize the cosine similarities between cells and principal components (PCs).

Usage

```
calculateCellSimilarityPCA(
  se_object,
  cell_names,
  pc_subset = 1:5,
  n_top_vars = 50,
  assay_name = "logcounts"
)

## S3 method for class 'calculateCellSimilarityPCAObject'
plot(x, pc_subset = 1:5, ...)
```

Arguments

`se_object`	A `SingleCellExperiment` object containing expression data.
`cell_names`	A character vector specifying the cell names for which to compute the similarity.
`pc_subset`	A numeric vector specifying the subset of principal components to include in the plot. Default is 1:5.
`n_top_vars`	An integer indicating the number of top loading variables to consider for each PC. Default is 50.
`assay_name`	Name of the assay on which to perform computations. Default is "logcounts".
`x`	An object of class 'calculateCellSimilarityPCA' containing a dataframe of cosine similarity values between cells and PCs.
`...`	Additional arguments passed to the plotting function.

Details

This function calculates the cosine similarity between cells based on the loadings of the selected principal components obtained from PCA. It extracts the rotation matrix from the PCA results of the `SingleCellExperiment` object and identifies the high-loading variables for each selected PC. Then, it computes the cosine similarity between cells using the high-loading variables for each PC.

The S3 plot method reshapes the input data frame to create a long format suitable for plotting as a heatmap. It then creates a heatmap plot using ggplot2, where the x-axis represents the PCs, the y-axis represents the cells, and the color intensity represents the cosine similarity values.

Value

A data frame containing cosine similarity values between cells for each selected principal component.

The S3 plot method returns a ggplot object representing the cosine similarity heatmap.

Author(s)

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

See Also

plot.calculateCellSimilarityPCAObject

calculateCellSimilarityPCA

Examples

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

# Store PCA anomaly data and plots
anomaly_output <- detectAnomaly(reference_data = reference_data,
                                query_data = query_data,
                                ref_cell_type_col = "expert_annotation",
                                query_cell_type_col = "SingleR_annotation",
                                pc_subset = 1:10,
                                n_tree = 500,
                                anomaly_treshold = 0.5)
top6_anomalies <- names(sort(anomaly_output$Combined$reference_anomaly_scores,
                             decreasing = TRUE)[1:6])

# Compute cosine similarity between anomalies and top PCs
cosine_similarities <- calculateCellSimilarityPCA(reference_data,
                                                  cell_names = top6_anomalies,
                                                  pc_subset = 1:25,
                                                  n_top_vars = 50)
cosine_similarities

# Plot similarities
plot(cosine_similarities, pc_subset = 15:25)
```

---

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
```

`calculateDiscriminantSpace` *Project Query Data onto Discriminant Space of Reference Data*

Description

This function projects query single-cell RNA-seq data onto the discriminant space defined by reference data. The reference data is used to identify important variables and compute discriminant vectors, which are then used to project both reference and query data. Similarity between the query and reference projections is assessed using cosine similarity and Mahalanobis distance.

The S3 plot method plots the projected reference and query data on discriminant spaces.

Usage

```
calculateDiscriminantSpace(
  reference_data,
  query_data = NULL,
  ref_cell_type_col,
  query_cell_type_col = NULL,
  cell_types = NULL,
  n_tree = 500,
  n_top = 20,
  eigen_threshold = 0.1,
  calculate_metrics = FALSE,
  alpha = 0.01,
  assay_name = "logcounts"
)
```

```
## S3 method for class 'calculateDiscriminantSpaceObject'
plot(x, cell_types, plot_type = c("scatterplot", "boxplot"), ...)
```

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, only the projected reference data is returned. Default is NULL.
ref_cell_type_col	The column name in reference_data indicating cell type labels.
query_cell_type_col	The column name in query_data indicating cell type labels.
cell_types	A character vector specifying the cell types to plot. If not provided, all cell types will be plotted.
n_tree	An integer specifying the number of trees for the random forest used in variable importance calculation.
n_top	An integer specifying the number of top variables to select based on importance scores.
eigen_threshold	A numeric value specifying the threshold for retaining eigenvalues in discriminant analysis.
calculate_metrics	Parameter to determine if cosine similarity and Mahalanobis distance metrics should be computed. Default is FALSE.
alpha	A numeric value specifying the significance level for Mahalanobis distance cutoff.
assay_name	Name of the assay on which to perform computations. Default is "logcounts".
x	An object of class calculateDiscriminantSpace containing the projected data on the discriminant space.. Each element of the list represents a combination of cell types and datasets. Each element should contain 'ref_proj' and 'query_proj' data frames.
plot_type	Type of plot to generate. Options are "scatterplot" and "boxplot". Default is "scatterplot".
...	Additional arguments to be passed to the plotting functions.

Details

The function performs the following steps for each pairwise combination of cell types:

- Identifies the top important variables to distinguish the two cell types from the reference data.
- Computes the Ledoit-Wolf shrinkage estimate of the covariance matrix for each cell type using the top important genes.
- Constructs within-class and between-class scatter matrices.

- Solves the generalized eigenvalue problem to obtain discriminant vectors.
- Projects both reference and query data onto the discriminant space.
- Assesses similarity of the query data projection to the reference data using cosine similarity and Mahalanobis distance.

The S3 plot method generates either a scatterplot or a boxplot to visualize the projected data onto the discriminant spaces. For scatterplot, each point represents a projected data point, and colors are used to differentiate between different cell types and datasets. For boxplot, the distribution of the projected data values for each cell type is shown, separated by datasets.

Value

A list with the following components for each cell type combination:

`discriminant_eigenvalues`
: Eigenvalues from the discriminant analysis.

`discriminant_eigenvectors`
: Eigenvectors from the discriminant analysis.

`ref_proj`
: Reference data projected onto the discriminant space.

`query_proj`
: Query data projected onto the discriminant space.

`query_mahalanobis_dist`
: Mahalanobis distances of query projections.

`mahalanobis_crit`
: Cutoff value for Mahalanobis distance significance.

`query_cosine_similarity`
: Cosine similarity scores of query projections.

The S3 plot method returns a `ggplot` object representing the scatterplot or boxplot of the projected data.

Author(s)

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

References

- Fisher, R. A. (1936). "The Use of Multiple Measurements in Taxonomic Problems". *Annals of Eugenics*. 7 (2): 179–188. doi:10.1111/j.1469-1809.1936.tb02137.x.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer. Chapter 4: Linear Methods for Classification.
- Ledoit, O., & Wolf, M. (2004). "A well-conditioned estimator for large-dimensional covariance matrices". *Journal of Multivariate Analysis*. 88 (2): 365–411. doi:10.1016/S0047-259X(03)00096-4.
- De Maesschalck, R., Jouan-Rimbaud, D., & Massart, D. L. (2000). "The Mahalanobis distance". *Chemometrics and Intelligent Laboratory Systems*. 50 (1): 1–18. doi:10.1016/S0169-7439(99)00047-7.
- Breiman, L. (2001). "Random Forests". *Machine Learning*. 45 (1): 5–32. doi:10.1023/A:1010933404324.

See Also

plot.calculateDiscriminantSpaceObject

calculateDiscriminantSpace

Examples

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

# Compute important variables for all pairwise cell comparisons
disc_output <- calculateDiscriminantSpace(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,
                                          eigen_threshold  = 1e-1,
                                          calculate_metrics = FALSE,
                                          alpha = 0.01)

# Generate scatter and boxplot
plot(disc_output, plot_type = "scatterplot")
plot(disc_output, cell_types = "CD4-CD8", plot_type = "boxplot")

# Check comparison
table(Expert_Annotation = query_data$expert_annotation, SingleR = query_data$SingleR_annotation)
```

---

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
```

---

calculateNearestNeighborProbabilities

Calculate Nearest Neighbor Diagnostics for Cell Type Classification

---

Description

This function computes the probabilities for each query cell of belonging to either the reference or query dataset for each cell type using nearest neighbor analysis.

The S3 plot method generates a density plot showing the distribution of probabilities for each cell of belonging to either the reference or query dataset for each cell type.

Usage

```
calculateNearestNeighborProbabilities(
  query_data,
  reference_data,
  query_cell_type_col,
  ref_cell_type_col,
  cell_types = NULL,
  pc_subset = 1:5,
  n_neighbor = 20,
  assay_name = "logcounts"
)

## S3 method for class 'calculateNearestNeighborProbabilitiesObject'
plot(x, cell_types = NULL, ...)
```

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 name in the query dataset containing cell type annotations.
ref_cell_type_col	A character string specifying the column name in the reference dataset containing cell type annotations.
cell_types	A character vector specifying the cell types to include in the plot. If NULL, all cell types in x will be plotted. Default is NULL.
pc_subset	A vector specifying the subset of principal components to use in the analysis. Default is 1:5
n_neighbor	An integer specifying the number of nearest neighbors to consider. Default is 20.
assay_name	Name of the assay on which to perform computations. Default is "logcounts".
x	An object of class nearestNeighbotDiagnostics containing the probabilities calculated by the calculateNearestNeighborProbabilities function.
...	Additional arguments to be passed to geom_density.

Details

The function conducts PCA on both the query and reference datasets to reduce dimensionality. It then compares each query cell to its nearest neighbors in the reference dataset to estimate the probability of membership in each cell type. Sample sizes between datasets are balanced using data augmentation if necessary.

The S3 plot method creates a density plot to visualize the distribution of probabilities for each cell belonging to the reference or query dataset for each cell type. It utilizes the ggplot2 package for plotting.

Value

A list where each element corresponds to a cell type and contains:

n_neighbor	The number of nearest neighbors considered.
n_query	The number of cells in the query dataset for each cell type.
query_prob	The average probability of each query cell belonging to the reference dataset.

The list is assigned the class "calculateNearestNeighborProbabilities". Each element in the list is named after the respective cell type.

The S3 plot method returns a ggplot density plot.

Author(s)

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

See Also

plot.calculateNearestNeighborProbabilitiesObject

calculateNearestNeighborProbabilities

Examples

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

# Project the query data onto PCA space of reference
nn_output <- calculateNearestNeighborProbabilities(query_data = query_data,
                                                   reference_data = reference_data,
                                                query_cell_type_col = "SingleR_annotation",
                                                   ref_cell_type_col = "expert_annotation",
                                                   pc_subset = 1:10,
                                                   n_neighbor = 20)

# Plot output
plot(nn_output, cell_types = c("CD4", "CD8", "B_and_plasma", "Myeloid"))
```

---

calculateSIRSpace *Calculate Sliced Inverse Regression (SIR) Space for Different Cell Types*

Description

This function calculates the SIR space projections for different cell types in the query and reference datasets.

The S3 plot method visualizes the projected reference and query data on discriminant spaces using either a scatterplot, boxplot, or varplot.

Usage

```
calculateSIRSpace(
  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
)
```

```
## S3 method for class 'calculateSIRSpaceObject'
plot(
  x,
  plot_type = c("scatterplot", "boxplot", "varplot"),
  sir_subset = NULL,
  n_top_vars = 5,
  ...
)
```

Arguments

query_data	A SingleCellExperiment object containing the numeric expression matrix for the query cells.
reference_data	A SingleCellExperiment object containing the numeric expression matrix for the reference cells.
query_cell_type_col	A character string specifying the column name in the colData of query_data that identifies the cell types.
ref_cell_type_col	A character string specifying 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 analysis. If NULL, all common cell types between the query and reference data will be used.
multiple_cond_means	Logical. Whether to compute conditional means for multiple conditions in the reference dataset. Default is TRUE.
assay_name	A character string specifying the name of the assay on which to perform computations. Default is "logcounts".
cumulative_variance_threshold	A numeric value specifying the cumulative variance threshold for selecting principal components. Default is 0.7.
n_neighbor	A numeric value specifying the number of neighbors for computing the SIR space. Default is 1.
x	An object of class calculateSIRSpace, which contains projected data on the discriminant space. Each element should include 'ref_proj' and 'query_proj' data frames representing reference and query projections.
plot_type	A character string indicating the type of plot to generate. Options are "scatterplot", "boxplot", or "varplot". Default is "scatterplot".
sir_subset	A numeric vector specifying which discriminant axes (SIR components) to include in the plot. Default is the first 5 axes.
n_top_vars	Number of top contributing variables to display in varplot. Default is 5.
...	Additional arguments to be passed to the plotting functions.

Details

The function projects the query dataset onto the SIR space of the reference dataset based on shared cell types. It computes conditional means for the reference dataset, extracts the SVD components, and performs the projection of both the query and reference data. It uses the 'projectSIR' function to perform the actual projection and allows the user to specify particular cell types for analysis.

- **Scatterplot**: Displays projected data points, with colors used to differentiate between cell types and datasets. - **Boxplot**: Shows the distribution of projected data values for each cell type, separated by datasets. - **Varplot**: Highlights the top contributing variables for each discriminant axis, differentiating between positive and negative loadings.

Value

A list containing the SIR projections, rotation matrix, and percentage of variance explained for the given cell types.

A `ggplot` object representing the chosen visualization (scatterplot, boxplot, or varplot) of the projected data.

Author(s)

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

See Also

`plot.calculateSIRSpaceObject`

`calculateSIRSpace`

Examples

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

# Compute important variables for all pairwise cell comparisons
sir_output <- calculateSIRSpace(reference_data = reference_data,
                                query_data = query_data,
                                query_cell_type_col = "SingleR_annotation",
                                ref_cell_type_col = "expert_annotation",
                                multiple_cond_means = TRUE,
                                cumulative_variance_threshold = 0.9,
                                n_neighbor = 1)

# Generate boxplot of SIR projections
plot(sir_output, plot_type = "boxplot", sir_subset = 1:6)
```

`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)
```

`calculate_entropy` *Calculate Entropy*

Description

This function calculates the entropy of a probability distribution.

Usage

```
calculate_entropy(p)
```

Arguments

p	A numeric vector representing a probability distribution. The elements should sum to 1.

Details

The entropy is calculated using the formula $-\sum p \log(p)$, where the sum is over all non-zero elements of p.

Value

A numeric value representing the entropy of the probability distribution.

---

compareCCA *Compare Canonical Correlation Analysis (CCA) between Query and Reference Data*

Description

This function performs Canonical Correlation Analysis (CCA) between two datasets (query and reference) after performing PCA on each dataset. It projects the query data onto the PCA space of the reference data and then computes the cosine similarity of the canonical correlation vectors between the two datasets.

The S3 plot method generates a visualization of the output from the ‘compareCCA‘ function. The plot shows the cosine similarities of canonical correlation analysis (CCA) coefficients, with point sizes representing the correlations.

Usage

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

## S3 method for class 'compareCCAObject'
plot(x, ...)
```

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.

`pc_subset`
: A numeric vector specifying the subset of principal components (PCs) to compare. Default is the first five PCs.

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

`x`
: A list containing the output from the `compareCCA` function. This list should include `cosine_similarity` and `correlations`.

`...`
: Additional arguments passed to the plotting function.

Details

The function performs the following steps: 1. Projects the query data onto the PCA space of the reference data using the specified number of principal components. 2. Downsamples the datasets to ensure an equal number of rows for CCA. 3. Performs CCA on the projected datasets. 4. Computes the cosine similarity between the canonical correlation vectors and extracts the canonical correlations.

The cosine similarity provides a measure of alignment between the canonical correlation vectors of the two datasets. Higher values indicate greater similarity.

The S3 plot method converts the input list into a data frame suitable for plotting with `ggplot2`. Each point in the scatter plot represents the cosine similarity of CCA coefficients, with the size of the point indicating the correlation.

Value

A list containing the following elements:

coef_ref Canonical coefficients for the reference dataset.

coef_query Canonical coefficients for the query dataset.

cosine_similarity Cosine similarity values for the canonical variables.

correlations Canonical correlations between the reference and query datasets.

The S3 plot method returns a `ggplot` object representing the scatter plot of cosine similarities of CCA coefficients and correlations.

Author(s)

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

References

Hotelling, H. (1936). "Relations between two sets of variates". *Biometrika*, 28(3/4), 321-377. doi:10.2307/2333955.

See Also

plot.compareCCAObject

compareCCA

Examples

```
# Load libraries
library(scran)
library(scater)

# 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 <- getTopHVGs(ref_data_subset, n = 500)
query_top_genes <- 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 datasets separately
ref_data_subset <- runPCA(ref_data_subset)
query_data_subset <- runPCA(query_data_subset)

# Compare CCA
cca_comparison <- compareCCA(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)

# Visualize output of CCA comparison
plot(cca_comparison)
```

---

comparePCA *Compare Principal Components Analysis (PCA) Results*

Description

This function compares the principal components (PCs) obtained from separate PCA on reference and query datasets for a single cell type using either cosine similarity or correlation.

The S3 plot method generates a heatmap to visualize the cosine similarities between principal components from the output of the `comparePCA` function.

Usage

```
comparePCA(
  reference_data,
  query_data,
  query_cell_type_col,
  ref_cell_type_col,
  pc_subset = 1:5,
  n_top_vars = 50,
  metric = c("cosine", "correlation"),
  correlation_method = c("spearman", "pearson")
)

## S3 method for class 'comparePCAObject'
plot(x, ...)
```

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.
`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.
`pc_subset`	A numeric vector specifying the subset of principal components (PCs) to compare. Default is the first five PCs.
`n_top_vars`	An integer indicating the number of top loading variables to consider for each PC. Default is 50.
`metric`	The similarity metric to use. It can be either "cosine" or "correlation". Default is "cosine".
`correlation_method`	The correlation method to use if metric is "correlation". It can be "spearman" or "pearson". Default is "spearman".
`x`	A numeric matrix output from the `comparePCA` function, representing cosine similarities between query and reference principal components.
`...`	Additional arguments passed to the plotting function.

Details

This function compares the PCA results between the reference and query datasets by computing cosine similarities or correlations between the loadings of top variables for each pair of principal components. It first extracts the PCA rotation matrices from both datasets and identifies the top variables with highest loadings for each PC. Then, it computes the cosine similarities or correlations between the loadings of top variables for each pair of PCs. The resulting matrix contains the similarity values, where rows represent reference PCs and columns represent query PCs.

The S3 plot method converts the input matrix into a long-format data frame suitable for plotting with `ggplot2`. The rows in the heatmap are ordered in reverse to match the conventional display format. The heatmap uses a blue-white-red color gradient to represent cosine similarity values, where blue indicates negative similarity, white indicates zero similarity, and red indicates positive similarity.

Value

A similarity matrix comparing the principal components of the reference and query datasets. Each element (i, j) in the matrix represents the similarity between the i-th principal component of the reference dataset and the j-th principal component of the query dataset.

The S3 plot method returns a `ggplot` object representing the heatmap of cosine similarities.

Author(s)

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

See Also

`plot.comparePCAObject`

`comparePCA`

Examples

```
# Load libraries
library(scran)
library(scater)

# 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 <- getTopHVGs(ref_data_subset, n = 500)
query_top_genes <- 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 datasets separately
ref_data_subset <- runPCA(ref_data_subset)
query_data_subset <- runPCA(query_data_subset)

# Call the PCA comparison function
similarity_mat <- comparePCA(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_top_vars = 50,
                             metric = c("cosine", "correlation")[1],
                             correlation_method = c("spearman", "pearson")[1])

# Create the heatmap
plot(similarity_mat)
```

`comparePCASubspace` *Compare Subspaces Spanned by Top Principal Components*

Description

This function compares the subspace spanned by the top principal components (PCs) in a reference dataset to that in a query dataset. It computes the cosine similarity between the loadings of the top variables for each PC in both datasets and provides a weighted cosine similarity score.

The S3 plot method generates a visualization of the output from the `comparePCASubspace` function. The plot shows the cosine of principal angles between reference and query principal components, with point sizes representing the variance explained.

Usage

```
comparePCASubspace(
  reference_data,
  query_data,
  query_cell_type_col,
  ref_cell_type_col,
  pc_subset = 1:5,
  n_top_vars = 50
)

## S3 method for class 'comparePCASubspaceObject'
plot(x, ...)
```

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.
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.
pc_subset	A numeric vector specifying the subset of principal components (PCs) to compare. Default is the first five PCs.
n_top_vars	An integer indicating the number of top loading variables to consider for each PC. Default is 50.
x	A numeric matrix output from the comparePCASubspace function, representing cosine similarities between query and reference principal components.
...	Additional arguments passed to the plotting function.

Details

This function compares the subspace spanned by the top principal components (PCs) in a reference dataset to that in a query dataset. It first computes the cosine similarity between the loadings of the top variables for each PC in both datasets. The top cosine similarity scores are then selected, and their corresponding PC indices are stored. Additionally, the function calculates the average percentage of variance explained by the selected top PCs. Finally, it computes a weighted cosine similarity score based on the top cosine similarities and the average percentage of variance explained.

The S3 plot method converts the input list into a data frame suitable for plotting with ggplot2. Each point in the scatter plot represents the cosine of a principal angle, with the size of the point indicating the average variance explained by the corresponding principal components.

Value

A list containing the following components:

principal_angles_cosines	A numeric vector of cosine values of principal angles.
average_variance_explained	A numeric vector of average variance explained by each PC.
weighted_cosine_similarity	A numeric value representing the weighted cosine similarity.

The S3 plot method returns a ggplot object representing the heatmap of cosine similarities.

Author(s)

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

See Also

plot.comparePCASubspaceObject

comparePCASubspace

Examples

```
# Load libraries
library(scran)
library(scater)

# 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 <- getTopHVGs(ref_data_subset, n = 500)
query_top_genes <- 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 datasets separately
ref_data_subset <- runPCA(ref_data_subset)
query_data_subset <- runPCA(query_data_subset)

# Compare PCA subspaces
subspace_comparison <- comparePCASubspace(query_data = query_data_subset,
                                          reference_data = ref_data_subset,
                                          query_cell_type_col = "expert_annotation",
                                          ref_cell_type_col = "expert_annotation",
                                          n_top_vars = 50,
                                          pc_subset = 1:5)

# Plot output for PCA subspace comparison
plot(subspace_comparison)
```

---

conditionalMeans *Compute Conditional Means for Cell Types*

Description

This function computes conditional means for each cell type in the reference data. It can compute either a single conditional mean per cell type or multiple conditional means, depending on the specified settings. Principal component analysis (PCA) is used for dimensionality reduction before clustering when computing multiple conditional means.

Usage

```
conditionalMeans(
  reference_data,
  ref_cell_type_col,
  cell_types,
  multiple_cond_means = FALSE,
  assay_name = "logcounts",
  cumulative_variance_threshold = 0.7,
  n_neighbor = 1
)
```

Arguments

`reference_data`	A `SingleCellExperiment` object containing the reference data, where rows represent genes and columns represent cells.
`ref_cell_type_col`	A character string specifying the column in `colData(reference_data)` that contains the cell type labels.
`cell_types`	A character vector of cell types for which to compute conditional means.
`multiple_cond_means`	A logical value indicating whether to compute multiple conditional means per cell type. Defaults to `FALSE`.
`assay_name`	A character string specifying the name of the assay to use for the computation. 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 function offers two modes of operation: - **Single conditional mean per cell type**: For each cell type, it computes the mean expression across all observations. - **Multiple conditional means per cell type**: For each cell type, the function performs PCA to reduce dimensionality, followed by clustering to compute multiple conditional means.

Value

A numeric matrix with the conditional means for each cell type. If `multiple_cond_means = TRUE`, the matrix will contain multiple rows for each cell type, representing the different conditional means computed via clustering.

Author(s)

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

`detectAnomaly` *PCA Anomaly Scores via Isolation Forests with Visualization*

Description

This function detects anomalies in single-cell data by projecting the data onto a PCA space and using an isolation forest algorithm to identify anomalies.

The S3 plot method generates faceted scatter plots for specified principal component (PC) combinations within an anomaly detection object. It allows visualization of the relationship between specified PCs and highlights anomalies detected by the Isolation Forest algorithm.

Usage

```
detectAnomaly(
  reference_data,
  query_data = NULL,
  ref_cell_type_col,
  query_cell_type_col = NULL,
  cell_types = NULL,
  pc_subset = 1:5,
  n_tree = 500,
  anomaly_treshold = 0.6,
  assay_name = "logcounts",
  ...
)

## S3 method for class 'detectAnomalyObject'
plot(
  x,
  cell_type = NULL,
  pc_subset = NULL,
  data_type = c("query", "reference"),
  ...
)
```

Arguments

reference_data	A SingleCellExperiment object containing numeric expression matrix for the reference cells.
query_data	An optional SingleCellExperiment object containing numeric expression matrix for the query cells. If NULL, then the isolation forest anomaly scores are 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.
pc_subset	A numeric vector specifying the indices of the PCs to be included in the plots. If NULL, all PCs in reference_mat_subset will be included.
n_tree	An integer specifying the number of trees for the isolation forest. Default is 500
anomaly_treshold	A numeric value specifying the threshold for identifying anomalies, Default is 0.6.
assay_name	Name of the assay on which to perform computations. Default is "logcounts".
...	Additional arguments.
x	A list object containing the anomaly detection results from the detectAnomaly function. Each element of the list should correspond to a cell type and contain reference_mat_subset, query_mat_subset, var_explained, and anomaly.
cell_type	A character string specifying the cell type for which the plots should be generated. This should be a name present in x. If NULL, the "Combined" cell type will be plotted. Default is NULL.
data_type	A character string specifying whether to plot the "query" data or the "reference" data. Default is "query".

Details

This function projects the query data onto the PCA space of the reference data. An isolation forest is then built on the reference data to identify anomalies in the query data based on their PCA projections. If no query dataset is provided by the user, the anomaly scores are computed on the reference data itself. Anomaly scores for the data with all combined cell types are also provided as part of the output.

The S3 plot method extracts the specified PCs from the given anomaly detection object and generates scatter plots for each pair of PCs. It uses ggplot2 to create a faceted plot where each facet represents a pair of PCs. Anomalies are highlighted in red, while normal points are shown in black.

Value

A list containing the following components for each cell type and the combined data:

`anomaly_scores` Anomaly scores for each cell in the query data.

`anomaly` Logical vector indicating whether each cell is classified as an anomaly.

`reference_mat_subset` PCA projections of the reference data.

`query_mat_subset` PCA projections of the query data (if provided).

`var_explained` Proportion of variance explained by the retained principal components.

The S3 plot method returns a `ggplot` object representing the PCA plots with anomalies highlighted.

Author(s)

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

References

- Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining (pp. 413-422). IEEE.
- isotree: Isolation-Based Outlier Detection

See Also

`plot.detectAnomalyObject`

`detectAnomaly`

Examples

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

# Store PCA anomaly data
anomaly_output <- detectAnomaly(reference_data = reference_data,
                                query_data = query_data,
                                ref_cell_type_col = "expert_annotation",
                                query_cell_type_col = "SingleR_annotation",
                                pc_subset = 1:5,
                                n_tree = 500,
                                anomaly_treshold = 0.6)

# Plot the output for a cell type
plot(anomaly_output,
     cell_type = "CD4",
     pc_subset = 1:5,
     data_type = "query")
```

generateColors *Generate Paired Colors for Cell Types*

Description

This function assigns paired colors (light and dark) to a list of cell type names. The colors are selected from various color palettes in the 'pals' package.

Usage

```
generateColors(cell_type_names, paired = FALSE)
```

Arguments

cell_type_names	A character vector of cell type names that need to be assigned colors.
paired	If TRUE, the colored returned should be paired. Default is FALSE.

Details

The function uses color palettes from the 'pals' package to generate colors or pairs of colors (light and dark) for each cell type name provided. It cycles through different color families (blues, greens, reds, oranges, purples, purd and greys) to create the colors

Value

A named character vector where the names are the original cell type names, and the values are the assigned colors.

Author(s)

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

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")
```

hotellingT2 *Calculate Hotelling's T^2 Statistic*

Description

Calculates the Hotelling's T^2 statistic for comparing means of multivariate data.

Usage

```
hotellingT2(sample1, sample2)
```

Arguments

sample1	A numeric matrix or data frame of multivariate observations for sample 1, where rows are observations and columns are variables.
sample2	A numeric matrix or data frame of multivariate observations for sample 2, with the same structure as sample 1.

Value

The Hotelling's T^2 statistic.

inverse_normal_trans *Inverse Normal Transformation*

Description

This function performs an inverse normal transformation on a matrix or vector.

Usage

```
inverse_normal_trans(X, constant = 3/8)
```

Arguments

X	A numeric matrix or vector.
constant	A numeric value used in the transformation. Default is 3 / 8.

Details

The function ranks the elements of X and then applies the inverse normal transformation using the formula $qnorm((rank - constant)/(n - 2 * constant + 1))$.

Value

A matrix or vector with the same dimensions as X, with values transformed using the inverse normal transformation.

Author(s)

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

`ledoitWolf` *Ledoit-Wolf Covariance Matrix Estimation*

Description

Estimate the covariance matrix using the Ledoit-Wolf shrinkage method.

Usage

```
ledoitWolf(class_data)
```

Arguments

`class_data`	A numeric matrix or data frame containing the data for covariance estimation, where rows represent observations and columns represent variables.

Details

This function computes the Ledoit-Wolf shrinkage covariance matrix estimator, which improves the accuracy of the sample covariance matrix by shrinking it towards a structured estimator, typically the diagonal matrix with the mean of variances as its diagonal elements.

Value

A numeric matrix representing the Ledoit-Wolf estimated covariance matrix.

Author(s)

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

n_elements *Number of Elements*

Description

This function returns the number of elements in a matrix or vector.

Usage

```
n_elements(X)
```

Arguments

X	A matrix or vector.

Details

If X is a matrix, the function returns the product of its dimensions. If X is a vector, the function returns its length.

Value

An integer representing the number of elements in X.

Author(s)

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

plot.calculateWassersteinDistanceObject *Plot Density of Wasserstein Distances for Null Distribution*

Description

This function generates a density plot of Wasserstein distances for the null distribution of a 'calculateWassersteinDistanceObject'. Additionally, it overlays lines representing the significance threshold and the reference-query distance.

Usage

```
## S3 method for class 'calculateWassersteinDistanceObject'
plot(x, alpha = 0.05, ...)
```

Arguments

x	A list object containing the Wasserstein distance results from the calculateWassersteinDistance function.
alpha	A numeric value specifying the significance level for thresholding. Default is 0.05.
...	Additional arguments for future extensions.

Details

The density plot visualizes the distribution of Wasserstein distances calculated among reference samples, representing the null distribution. A vertical line marks the significance threshold based on the specified alpha. Another line indicates the mean Wasserstein distance between the reference and query datasets.

Value

A ggplot2 object representing the ridge plots of Wasserstein distances with annotated p-value.

Author(s)

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

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

calculateWassersteinDistance

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 null distribution using reference data and observed distances with query data
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)
```

`plot.regressPCObject` *Plot Regression Results on Principal Components*

Description

The S3 plot method generates plots to visualize the results of regression analyses performed on principal components (PCs) against cell types or dataset origin (query vs. reference).

This function performs linear regression of a covariate of interest onto one or more principal components, based on the data in a `SingleCellExperiment` object.

Usage

```
## S3 method for class 'regressPCObject'
plot(x, plot_type = c("r_squared", "p-value"), alpha = 0.05, ...)

regressPC(
  reference_data,
  query_data = NULL,
  ref_cell_type_col,
  query_cell_type_col = NULL,
  cell_types = NULL,
  pc_subset = 1:10,
 adjust_method = c("BH", "holm", "hochberg", "hommel", "bonferroni", "BY", "fdr",
    "none"),
  assay_name = "logcounts"
)
```

Arguments

`x`	An object of class `regressPC` containing the output of the `regressPC` function
`plot_type`	Type of plot to generate. Options are "r_squared" and "p-value". Default is "r-squared".
`alpha`	Significance threshold p-values of coefficients. Default is 0.05.
`...`	Additional arguments to be passed to the plotting functions.

`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.

`adjust_method` A character string specifying the method to adjust the p-values. Options include "BH", "holm", "hochberg", "hommel", "bonferroni", "BY", "fdr", or "none". Default is "BH" (Benjamini-Hochberg). Default is "BH".

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

Details

The S3 plot method generates, depending on the specified plot type, either the R-squared values or p-values resulting from the regression of principal components onto cell types or dataset origin (query vs. reference). For cell type regression, the plots show how well each PC correlates with different cell types. For dataset regression, the plots compare the PCs between query and reference datasets.

Principal component regression, derived from PCA, can be used to quantify the variance explained by a covariate of interest. Applications for single-cell analysis include quantification of batch effects, assessing clustering homogeneity, and evaluating alignment of query and reference datasets in cell type annotation settings.

Briefly, the R^2 is calculated from a linear regression of the covariate B of interest onto each principal component. The variance contribution of the covariate effect per principal component is then calculated as the product of the variance explained by the i-th principal component (PC) and the corresponding $R^2(PC_i|B)$. The sum across all variance contributions by the covariate effects in all principal components gives the total variance explained by the covariate as follows:

$$Var(C|B) = \sum_{i=1}^{G} \text{Var}(C|PC_i) \times R^2(PC_i|B)$$

where, $\text{Var}(C \mid PC_i)$ is the variance of the data matrix C explained by the i-th principal component. See references for details.

Value

The S3 plot method returns a `ggplot` object representing the specified plot type.

A `list` containing

- summaries of the linear regression models for each specified principal component,
- the corresponding R-squared (R2) values,
- the variance contributions for each principal component, and
- the total variance explained.

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.

See Also

regressPC

plot.regressPCObject

Examples

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

# Plot the PC data (no query data)
regress_res <- regressPC(reference_data = reference_data,
                         ref_cell_type_col = "expert_annotation",
                         cell_types = c("CD4", "CD8", "B_and_plasma", "Myeloid"),
                         pc_subset = 1:15)
# Plot results
plot(regress_res, plot_type = "r_squared")
plot(regress_res, plot_type = "p-value")

# Plot the PC data (with query data)
regress_res <- regressPC(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 = 1:15)
# Plot results
plot(regress_res, plot_type = "r_squared")
plot(regress_res, plot_type = "p-value")
```

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")
```

qc_data *Quality Control Single-Cell RNA-Seq Dataset*

Description

This dataset contains the processed query dataset from the Bunis haematopoietic stem and progenitor cell data. It has been preprocessed to include log-normalized counts, QC metrics, SingleR cell type predictions, and annotation scores.

Usage

```
qc_data
```

Format

An object of class `SingleCellExperiment` with 500 rows and 750 columns.

Details

This dataset underwent the following steps:

- Loads the `hpca` reference dataset using `fetchReference` from the `celldex` package.
- Loads the QC dataset (Bunis haematopoietic stem and progenitor cell data) from Bunis DG et al. (2021).
- Adds QC metrics to the QC dataset using the function `addPerCellQCMetrics` from the `scuttle` package.
- Performs log normalization on the QC dataset using the function `logNormCounts` from the `scuttle` package.
- Runs SingleR to predict cell types and assigns predicted labels to the QC dataset using the function `SingleR` from the `SingleR` package.
- Assigns annotation scores to the QC dataset.
- Selects specific columns (`total`, `SingleR_annotation`, `annotation_scores`) from the cell metadata for downstream analysis.
- Selects highly variable genes (HVGs) using the function `getTopHVGs` from the `scran` package on the QC dataset.

Source

Bunis DG et al. (2021). Single-Cell Mapping of Progressive Fetal-to-Adult Transition in Human Naive T Cells Cell Rep. 34(1): 108573

References

Bunis DG et al. (2021). Single-Cell Mapping of Progressive Fetal-to-Adult Transition in Human Naive T Cells Cell Rep. 34(1): 108573

See Also

Use `data("qc_data")` to load and access the resulting quality control dataset.

Examples

```
# Load and explore the quality control dataset
data("qc_data")
```

`query_data` *Query Single-Cell RNA-Seq Dataset*

Description

This dataset contains the processed query dataset from the HeOrganAtlas dataset for Marrow tissue. It has been preprocessed to include log-normalized counts, specific metadata columns, annotations based on SingleR cell type scoring, and PCA, t-SNE, and UMAP results.

Usage

```
query_data
```

Format

An object of class `SingleCellExperiment` with 392 rows and 503 columns.

Details

This dataset underwent the following steps:

- Loads the HeOrganAtlas dataset specifically for Marrow tissue from the `scRNAseq` package.
- Divides the loaded dataset into a query dataset used for downstream analysis.
- Performs log normalization on the query dataset using the function `logNormCounts` from the `scuttle` package.
- Selects specific columns (`percent_mito`, `expert_annotation`) from the cell metadata for downstream analysis.
- Adds SingleR annotations (`SingleR_annotation`) and annotation scores (`annotation_scores`) to the query dataset using the function `SingleR` from the `SingleR` package.
- Computes AUC gene set scores using the function `AUCell_calcAUC` from the `AUCell` package and adds these scores to the query dataset.
- Selects highly variable genes (HVGs) using the function `getTopHVGs` from the `scran` package on the query dataset.
- Intersects the highly variable genes between the query and reference datasets to obtain common genes for analysis.

- Performs Principal Component Analysis (PCA) on the query dataset using the function `runPCA` from the `scater` package.
- Performs t-Distributed Stochastic Neighbor Embedding (t-SNE) on the query dataset using the function `runTSNE` from the `scater` package.
- Performs Uniform Manifold Approximation and Projection (UMAP) on the query dataset using the function `runUMAP` from the `scater` package.

Source

The HeOrganAtlas dataset, available through the scRNAseq package.

References

He, et al. (2020). HeOrganAtlas: a comprehensive human organ atlas based on single-cell RNA sequencing.

See Also

Use `data("query_data")` to load and access the resulting query dataset and the `data("reference_data")` for comparison with the reference dataset.

Examples

```
# Load and explore the query dataset
data("query_data")
```

---

reference_data *Reference Single-Cell RNA-Seq Dataset*

Description

This dataset contains the processed reference dataset from the HeOrganAtlas dataset for Marrow tissue. It has been preprocessed to include log-normalized counts, specific metadata columns, and PCA, t-SNE, and UMAP results.

Usage

```
reference_data
```

Format

An object of class `SingleCellExperiment` with 392 rows and 1500 columns.

Details

This dataset underwent the following steps:

- Loads the HeOrganAtlas dataset specifically for Marrow tissue from the `scRNAseq` package.
- Divides the loaded dataset into a reference dataset used for downstream analysis.
- Performs log normalization on the reference dataset using the function `logNormCounts` from the `scuttle` package.
- Selects the column `expert_annotation`) from the cell metadata for downstream analysis.
- Selects highly variable genes (HVGs) using the function `getTopHVGs` from the `scran` package on the reference dataset.
- Performs Principal Component Analysis (PCA) on the reference dataset using the function `runPCA` from the `scater` package.
- Performs t-Distributed Stochastic Neighbor Embedding (t-SNE) on the reference dataset using the function `runTSNE` from the `scater` package.
- Performs Uniform Manifold Approximation and Projection (UMAP) on the reference dataset using the function `runUMAP` from the `scater` package.

Source

The HeOrganAtlas dataset, available through the scRNAseq package.

References

He, et al. (2020). HeOrganAtlas: a comprehensive human organ atlas based on single-cell RNA sequencing.

See Also

Use `data("reference_data")` to load and access the resulting reference dataset.

Examples

```
# Load and explore the reference dataset
data("reference_data")
```

Index