Package 'ADM'

September 25, 2024

Title Advanced Distance Measurement for High-Dimensional Data Analysis

Version 0.99.0

Description ADM provides a suite of advanced distance measurement methods for high-dimensional data analysis. The package implements various dimensionality reduction and visualization techniques, including UMAP, t-SNE, and PCA, along with tools for evaluating the performance of these methods. Additionally, ADM includes functions for calculating inter-group similarities and custom ggplot2 themes for data visualization.

License Artistic-2.0

Encoding UTF-8

Depends R (>= 4.3.0)

Imports stats, MASS, rARPACK, utils, grDevices, graphics, methods, phateR, ggplot2, mclust, cluster, magrittr, tidyr, dplyr, ggrepel, BiocParallel, uwot, aricode, DDoutlier, diffudist, fitdistrplus, igraph, Rtsne, dimRed

Suggests knitr, rmarkdown, MixGHD, testthat (>= 3.0.0), RDRToolbox, class, dbscan, survival, umap

Config/testthat/edition 3

VignetteBuilder knitr

biocViews Visualization, DimensionReduction, FeatureExtraction, Clustering

RoxygenNote 7.3.2

NeedsCompilation no

Author Junning Feng [aut, cre]

Maintainer Junning Feng <jnfeng3522@gmail.com>

R topics documented:

`adm` *Multidimensional Embedding Visualization (ADM)*

Description

Multidimensional Embedding Visualization (ADM)

Usage

```
adm(
  e,
  k.dim = NULL,
  dist.power = 0.5,
  conn.prop = 0.02,
  raw.d.pwr = 0.5,
  diffu.steps = NA,
  diffu.factor = 3.5,
  distr.template = "gamma",
  gamma.shape = 3,
  gamma.rate = 3,
  scale.dist = TRUE,
  symmetrize = "mean",
  dist.quantile = 0.25
)
```

Arguments

`e`	A list of matrices, each representing a dimension reduction result.
`k.dim`	Number of dimensions to use. If NULL, uses the number of columns in the first matrix of e.
`dist.power`	Power to raise distances to when merging. Default is 0.5.
`conn.prop`	Proportion of connections to consider. Default is 0.02.
`raw.d.pwr`	Power for raw distance calculation. Default is 0.5.
`diffu.steps`	Steps for diffusion. If NA, calculated based on graph properties.
`diffu.factor`	Factor to multiply diffusion steps by. Default is 3.5.

distr.template	Distribution template to use. Options are "none", "combine", "gamma", or "parametric". Default is "gamma".
gamma.shape	Shape parameter for gamma distribution. Default is 3.
gamma.rate	Rate parameter for gamma distribution. Default is 3.
scale.dist	Whether to scale distances. Default is TRUE.
symmetrize	Method to symmetrize distance matrix. Options are "mean", "min", or "max". Default is "mean".
dist.quantile	Quantile to use for distance calculations. Default is 0.25.

Value

A list containing the final merged distance matrix.

`cal_ari_nmi` *Calculate Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI)*

Description

Calculate Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI)

Usage

```
cal_ari_nmi(lowDim_data, k, method_name, seed, info = NULL)
```

Arguments

lowDim_data	A matrix or data frame of low-dimensional data.
k	The number of clusters for k-means clustering.
method_name	A string naming the method being evaluated.
seed	An integer seed for reproducibility.
info	A vector of true class labels. If NULL, it will use the global 'info' variable.

Details

This function performs k-means clustering on the input data and calculates the Adjusted Rand Index and Normalized Mutual Information between the clustering result and a pre-existing classification.

Value

A data frame with ARI and NMI scores.

Examples

```
data <- matrix(rnorm(200), 100, 2)
info <- sample(1:3, 100, replace = TRUE)  # Simulated cluster information
result <- cal_ari_nmi(data, k = 3, method_name = "Example Method", seed = 123, info = info)
```

cal_cci *Calculate Cluster Conservation Index (CCI)*

Description

This function calculates the Cluster Conservation Index (CCI) for both the ensemble and ADM methods.

Usage

```
cal_cci(ensemble.out, mev.out, info)
```

Arguments

ensemble.out	A list containing the output from the ensemble method, including the ensemble distance matrix.
mev.out	A list containing the output from the ADM method, including the diffusion distance matrix.
info	A vector or factor containing cluster information for each data point.

Details

The function calculates CCI for different numbers of neighbors (1, 2, 5, 10, 20) and for different representations of the data (raw, UMAP, PCA).

Value

A list containing two elements:

cci_viz	A matrix with CCI values for the ensemble method
cci_adm	A matrix with CCI values for the ADM method

candidate.visual *Perform Multiple Dimensionality Reduction Methods*

Description

This function applies various dimensionality reduction methods to the input data. It supports methods such as PCA, MDS, iMDS, Sammon mapping, LLE, HLLE, Isomap, kPCA, LEIM, UMAP, t-SNE, PHATE, and KEF.

Usage

```
candidate.visual(
  data,
  dim = 2,
 methods = c("PCA", "MDS", "iMDS", "Sammon", "HLLE", "Isomap", "kPCA", "LEIM", "UMAP",
    "tSNE", "PHATE", "KEF"),
  kpca.sigma = c(0.001, 0.002),
  umap.k = c(30, 50),
  tsne.perplexity = c(30, 50),
  phate.k = c(30, 50),
  cal_dist = TRUE
)
```

Arguments

data	A numeric matrix or data frame where rows are observations and columns are features.
dim	Integer. The number of dimensions for the reduced space. Default is 2.
methods	Character vector. Methods to be applied. Default includes all supported methods.
kpca.sigma	Numeric vector. Sigma values for kPCA. Default is c(0.001, 0.002).
umap.k	Integer vector. Number of neighbors for UMAP. Default is c(30, 50).
tsne.perplexity	Numeric vector. Perplexity values for t-SNE. Default is c(30, 50).
phate.k	Integer vector. Number of nearest neighbors for PHATE. Default is c(30, 50).
cal_dist	Logical. Whether to calculate distance matrix. Default is TRUE.

Details

The function applies each specified method to the input data. For some methods (kPCA, UMAP, t-SNE, PHATE), multiple parameter settings are tried, resulting in multiple outputs for these methods.

Value

A list containing two elements:

embed.list	A list of matrices, each representing the reduced data for a method
method_name	A character vector of method names corresponding to embed.list

Note

- This function requires several packages including rARPACK, MASS, dimRed, umap, and phateR. - The function prints progress messages to the console. - Some methods may fail if the required packages are not installed.

Examples

```
## Not run:
# Assuming 'data' is your input dataset
result <- candidate.visual(data, dim = 2,
                           methods = c("PCA", "MDS", "UMAP"),
```

```
                            umap.k = c(15, 30))
pca_result <- result$embed.list[[1]]
method_names <- result$method_name

## End(Not run)
```

`dataloader` *Load and Prepare Dataset for Analysis*

Description

Load and Prepare Dataset for Analysis

Usage

```
dataloader(dataset, base_dir = ".")
```

Arguments

`dataset`	A string specifying the dataset to load. Supported datasets are: "Oihane", "Gutierrez", "Spleen", "Quake", "pbmc", "metabolism", "kidney", "mir", "gene".
`base_dir`	A string specifying the base directory for data files. Default is ".".

Details

This function loads and prepares various datasets for analysis. Each dataset is processed differently based on its structure and content. The function sets the working directory to a specific path for each dataset before loading.

Value

A list containing:

`dat`	The main data matrix or data frame
`cell.type`	A factor of cell types
`info`	A factor of additional information (often identical to cell.type)

Note

- The function assumes a specific directory structure and file naming convention. - Some datasets require additional files to be present in the specified directories. - For "mir" and "gene" datasets, extensive preprocessing is performed.

Examples

```
## Not run:
# Load the Oihane dataset
oihane_data <- dataloader("Oihane")

# Load the PBMC dataset
pbmc_data <- dataloader("pbmc")

# Load the miRNA dataset
mir_data <- dataloader("mir")

## End(Not run)
```

`dist.grp` *Calculate distance-based group statistics*

Description

This function calculates distance-based group statistics for a given distance matrix and grouping.

Usage

```
dist.grp(distmat, grp, k = 1:20)
```

Arguments

`distmat`	A distance matrix.
`grp`	A vector of group labels for each data point.
`k`	Vector of integers; the number of neighbors to consider.

Value

A matrix with two columns: the k values and corresponding distance-based group statistics.

`get_mapping` *Get label mapping for a dataset*

Description

This function returns the appropriate label mapping based on the provided dataset name. It currently supports four datasets: "Oihane", "mir", "gene", and "Quake".

Usage

```
get_mapping(dataset)

get_mapping(dataset)
```

Arguments

dataset	A string specifying the name of the dataset. Possible values are "Oihane", "mir", "gene", or "Quake".

Value

A named vector where names are the original labels and values are the corresponding abbreviations or simplified labels. Returns NULL if the provided dataset name is not recognized.

A named vector where names are the original labels and values are the corresponding abbreviations or simplified labels. Returns NULL if the provided dataset name is not recognized.

Examples

```
get_mapping("Oihane")
get_mapping("Quake")

get_mapping("Oihane")
get_mapping("Quake")
```

---

move.outlier *Move Outliers Function*

Description

Move Outliers Function

Usage

```
move.outlier(x, d = NULL, fraction = 0.01)
```

Arguments

x	A numeric vector or matrix.
d	The distance threshold for outlier detection. If NULL, it's calculated from the data.
fraction	The fraction of points to consider as outliers.

Value

A numeric vector or matrix with outliers moved.

plot_cci_results *Plot Comparison of CCI Results*

Description

This function processes and visualizes the results of Cell-Cell Interaction (CCI) analyses for different methods and data transformations.

Usage

```
plot_cci_results(cci_list, dataset)
```

Arguments

cci_list	A list containing two matrices: one for visualization method and one for diffusion method.
dataset	A string specifying the name of the dataset (used for plot title).

Details

The function processes CCI matrices for visualization and diffusion methods, combines the data, and creates a boxplot comparing the results across different data types (raw, UMAP, PCA) and methods (metaspec, ADM).

Value

A ggplot object representing a boxplot of CCI results.

Examples

```
viz_matrix <- matrix(rnorm(30), nrow = 10,
                     dimnames = list(NULL, c("viz_raw", "viz_umap", "viz_pca")))
diffu_matrix <- matrix(rnorm(30), nrow = 10,
                       dimnames = list(NULL, c("diffu_raw", "diffu_umap", "diffu_pca")))
cci_list <- list(viz_matrix, diffu_matrix)
p <- plot_cci_results(cci_list, "Example Dataset")
if (interactive()) {
  print(p)
}
```

plot_legend *Create a scatter plot with a custom legend*

Description

This function generates a scatter plot from UMAP coordinates with a customized legend.

Usage

```
plot_legend(
  umap_viz,
  method_name,
  info,
  dataset,
  color_list,
  label_mapping = NULL
)
```

Arguments

umap_viz	A matrix or data frame with UMAP coordinates (at least two columns).
method_name	A string specifying the visualization method name (used for plot title).
info	A vector of original cluster labels for each data point.
dataset	A string specifying the dataset name (not used in the current implementation).
color_list	A named vector of colors for each cluster.
label_mapping	An optional named vector for mapping original labels to simplified ones.

Details

The function creates a scatter plot of UMAP coordinates, colors points by cluster, and applies a custom theme with a specially formatted legend. It uses the 'simplify_labels' function to potentially simplify cluster labels.

Value

A ggplot object representing the scatter plot with a customized legend.

Examples

```
umap_coords <- matrix(rnorm(200), ncol = 2)
cluster_info <- rep(c("Type A", "Type B"), each = 50)
color_list <- c("Type A" = "red", "Type B" = "blue")
label_map <- c("Type A" = "A", "Type B" = "B")
p <- plot_legend(umap_coords, "UMAP", cluster_info, "Example", color_list, label_map)
if (interactive()) {
  print(p)
}
```

---

process_and_visualize_meta_methods

Process and Visualize Meta-Methods

Description

Process and Visualize Meta-Methods

Usage

```
process_and_visualize_meta_methods(
  mev.out,
  ensemble.out = NULL,
  info,
  k,
  color_list,
  seed = 2024
)
```

Arguments

mev.out	Output from MEV method (assumed to contain $diffu.dist).
ensemble.out	Output from an ensemble method (assumed to contain $ensemble.dist.mat). Can be NULL.
info	A vector of true class labels.
k	The number of clusters for k-means clustering.
color_list	A list of colors for visualization.
seed	An integer seed for reproducibility. Default is 2024.

Details

This function processes and visualizes results from two meta-methods: meta-spec and ADM. If ensemble.out is NULL, only ADM method is executed. It performs UMAP dimensionality reduction, k-means clustering, and calculates various metrics.

Value

A list containing ARI and NMI scores, silhouette widths, and UMAP coordinates for meta-spec and/or ADM methods.

Note

This function requires the following packages: umap, mclust, aricode, cluster

Examples

```
# This function requires specific input structures and external functions
# An example cannot be easily provided without those dependencies
```

---

simplify_labels *Simplify labels based on a provided mapping*

Description

This function simplifies a vector of labels using an optional mapping. If no mapping is provided, it returns the original labels.

Usage

```
simplify_labels(info, mapping = NULL)
```

Arguments

info	A vector of original labels to be simplified.
mapping	An optional named vector where names are original labels and values are simplified labels. Default is NULL.

Details

If a mapping is provided, the function replaces each label in info with its corresponding value in the mapping. For any labels not found in the mapping, the original label is retained.

Value

A vector of simplified labels. If no mapping is provided, returns the original labels. If mapping is provided, returns mapped labels where possible, falling back to original labels where no mapping exists.

Examples

```
# Without mapping
original_labels <- c("Type A", "Type B", "Type C")
simplify_labels(original_labels)
# Returns: c("Type A", "Type B", "Type C")

# With mapping
mapping <- c("Type A" = "A", "Type B" = "B")
simplify_labels(original_labels, mapping)
# Returns: c("A", "B", "Type C")
```

---

umap5 *Perform UMAP and calculate distance-based group statistics*

Description

This function applies UMAP to the input data and calculates distance-based group statistics.

Usage

```
umap5(x, info, do.plot = TRUE, n_components, k = c(1, 2, 5, 10, 20))
```

Arguments

x	A distance matrix or object that can be coerced to one.
info	A vector of group labels for each data point.
do.plot	Logical; if TRUE, creates a pairs plot of the UMAP result.
n_components	Integer; the number of dimensions for UMAP.
k	Vector of integers; the number of neighbors to consider for distance calculations.

Value

A vector of mean distance-based group statistics across multiple UMAP runs.

`visualization_func` *Visualize data using ggplot2*

Description

This function creates a scatter plot of the input data, optionally coloring points by group.

Usage

```
visualization_func(data, method_name, color_list = NULL, info = NULL)
```

Arguments

`data`	A matrix or data frame with at least two columns for x and y coordinates.
`method_name`	A string specifying the name of the visualization method (used for plot title).
`color_list`	An optional named vector of colors for each group in 'info'.
`info`	An optional vector of group labels for each data point.

Value

A ggplot object representing the scatter plot.

`visualization_with_label`
Create a labeled scatter plot from UMAP visualization

Description

This function generates a scatter plot from UMAP coordinates with labeled cluster centers.

Usage

```
visualization_with_label(umap_viz, method, info, dataset, color_list)
```

Arguments

`umap_viz`	A matrix or data frame with UMAP coordinates (at least two columns).
`method`	A string specifying the visualization method name (used for plot title).
`info`	A vector of cluster labels for each data point.
`dataset`	A string specifying the dataset name (not used in the current implementation).
`color_list`	A named vector of colors for each cluster.

Details

The function creates a scatter plot of UMAP coordinates, colors points by cluster, calculates and labels cluster centers, and applies a custom theme. It uses ggrepel for non-overlapping text labels.

Value

A ggplot object representing the labeled scatter plot.

Examples

```
umap_coords <- matrix(rnorm(200), ncol = 2)
cluster_info <- rep(c("A", "B"), each = 50)
color_list <- c("A" = "red", "B" = "blue")
p <- visualization_with_label(umap_coords, "UMAP", cluster_info, "Example", color_list)
if (interactive()) {
  print(p)
}
```

`visualize_ari_nmi` *Visualize ARI and NMI Comparison with Bar Plot*

Description

This function generates a bar plot to compare ARI and NMI metrics across methods.

Usage

```
visualize_ari_nmi(ARI_list, dataset)
```

Arguments

`ARI_list`	A list containing one or two elements: ARI and NMI scores for ADM method, and optionally for meta-spec method.
`dataset`	A string naming the dataset being visualized.

Details

The function adapts to whether one (ADM only) or two (meta-spec and ADM) methods are present in the input. It creates a bar plot using ggplot2, with metrics on the x-axis and their values on the y-axis. Bars are grouped and colored by the 'group' variable. The plot uses a minimal theme and predefined colors.

Value

Invisibly returns a ggplot object visualizing ARI and NMI scores.

Examples

```
# Example with only ADM method
ARI_list_single <- list(data.frame(ARI = 0.7, NMI = 0.8))
visualize_ari_nmi(ARI_list_single, "Dataset A")

# Example with both methods
ARI_list_double <- list(
  data.frame(ARI = 0.8, NMI = 0.9),
  data.frame(ARI = 0.7, NMI = 0.8)
)
visualize_ari_nmi(ARI_list_double, "Dataset B")
```

`visualize_individual_methods` *Visualize Individual Methods*

Description

This function creates scatter plots for multiple matrices using different methods.

Usage

```
visualize_individual_methods(
  matrices,
  method_names,
  info = NULL,
  color_list = NULL
)
```

Arguments

`matrices`	A list of matrices, each representing a different method's output.
`method_names`	A character vector specifying the names of the visualization methods.
`info`	An optional vector of group labels for each data point. If not provided, all points will be the same color.
`color_list`	An optional named vector of colors for each group in 'info'. If not provided, a default color palette will be used.

Value

A list of ggplot objects representing the scatter plots.

Examples

```
matrices <- list(matrix(rnorm(200), 100, 2), matrix(rnorm(200), 100, 2))
method_names <- c("Method1", "Method2")
# Example with default coloring (all points same color)
plots1 <- visualize_individual_methods(matrices, method_names)

# Example with group information
info <- sample(c("A", "B", "C"), 100, replace = TRUE)
```

```
color_list <- c("A" = "red", "B" = "blue", "C" = "green")
plots2 <- visualize_individual_methods(matrices, method_names, info, color_list)
```

`visualize_silhouette_width` *Visualize Silhouette Width Comparison*

Description

This function creates a boxplot to compare Silhouette Width values across different clusters and methods.

Usage

```
visualize_silhouette_width(ASW_list, info, data_name, label_mapping = NULL)
```

Arguments

`ASW_list`	A list of vectors containing Silhouette Width values for each method.
`info`	A vector of cluster labels for each data point.
`data_name`	A string specifying the name of the dataset (used for plot title).
`label_mapping`	An optional named vector for mapping original cluster labels to simplified ones.

Value

A ggplot object representing a boxplot of Silhouette Width values.

Examples

```
ASW_list <- list(Method1 = rnorm(100, 0.7, 0.1), Method2 = rnorm(100, 0.6, 0.1))
info <- sample(c("A", "B", "C"), 100, replace = TRUE)
label_mapping <- c("A" = "Cluster 1", "B" = "Cluster 2", "C" = "Cluster 3")
p <- visualize_silhouette_width(ASW_list, info, "Example Dataset", label_mapping)
if (interactive()) {
  print(p)
}
```

`zp.quantile` *Quantile Normalization Function*

Description

Quantile Normalization Function

Usage

```
zp.quantile(x, y)
```

Arguments

x	A numeric vector. The reference vector whose quantiles will be used for normalization.
y	A numeric vector. The vector to be normalized based on the quantiles of ‘x‘.

Value

A numeric vector of the same length as ‘y‘, containing the quantile-normalized values.

Index