Package 'vizplore'

April 3, 2025

Type Package

Title Visualization of High-Dimensional Data Using Dimensionality Reduction

Version 0.1.0

Author Radoslava Hatalova, Radoslav Harman

Maintainer Radoslava Hatalova <radoslava.hatalova@g.fmph.uniba.sk>

Description The 'vizplore' package provides functions for dimensionality reduction and visualization of high-dimensional data using techniques like PCA, t-SNE, Canonical Correlation Analysis for categorical data, and neural networks. It supports 2D and 3D visualizations and is designed to help explore data patterns visually.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Imports Rtsne,
keras3,
plotly,
reshape2,
stats

Depends R (>= 3.5.0)

Contents

---

catcca_viz   *catCCA Visualization*

Description

Creates a 2D visualization of multidimensional data using categorical canonical correlation analysis.

Usage

```
catcca_viz(X, y, dim = 2, center.scale = TRUE)
```

Arguments

X	A matrix representing the input features (quantitative variables).
y	A vector representing the categories corresponding to the input data.
dim	Integer indicating the desired dimensionality of the visualization: 2 for 2D, 3 for 3D. Default is 2.
center.scale	A logical (boolean) value indicating whether the data should be centered and scaled before processing.
center	A logical value indicating whether to center the quantitative data.

Value

A list containing:

projected_data The projected data onto the canonical components.

transformation_matrix The eigenvectors corresponding to the canonical components.

Additionally, a Plotly plot representing the data points in the reduced feature space, with points colored by their categories is displayed.

Examples

```
data(iris)
X <- iris[,-5]
y <- iris[,5]
catcca_viz(X, y)         # Default 2D visualization
catcca_viz(X, y, dim = 3) # 3D visualization
```

---

independent_views   *Visualization of Independent Views*

Description

Creates 2 independent 2D views of the multidimensional data using the specified dimensionality reduction method, with subsequent views based on orthogonal projections to explore independent aspects of the data.

Usage

```
independent_views(X, y, dim = 2, method = "all", center.scale = TRUE)
```

Arguments

X	A matrix (n x m) representing the input features, where n is the number of samples and m is the number of features.
y	A vector of length n representing the categories or labels corresponding to the input data.
dim	Integer indicating the desired dimensionality of the visualization: 2 for 2D, 3 for 3D. Default is 2.
method	A character string specifying the reduction method to use ("pca", "nn", "catcca").
center.scale	A logical (boolean) value indicating whether the data should be centered and scaled before processing.

Value

A grid of plotly subplots representing the independent views.

Examples

```
if (!require("ContaminatedMixt")) {
     install.packages("ContaminatedMixt")}
library("ContaminatedMixt")
data(wine)
X <- wine[,-1]
y <- wine[,1]
independent_views(X, y, method = 'pca')
```

---

nn_viz   *Neural Network Dimensionality Reduction Visualization*

Description

Creates a 2D (or 3D) visualization of high-dimensional data using a simple neural network.

Usage

```
nn_viz(X, y, dim = 2, center.scale = TRUE)
```

Arguments

X	A numeric matrix (n x m) representing the input features, where n is the number of samples and m is the number of features.
y	A vector of length n representing the categories or labels corresponding to the input data.
dim	Integer indicating the desired dimensionality of the visualization: 2 for 2D, 3 for 3D. Default is 2.
center.scale	A logical (boolean) value indicating whether the data should be centered and scaled before processing.

Value

A list containing:

projected_data The data projected onto the lower-dimensional space using the neural network.

transformation_matrix The weight matrix of the neural network's first layer used for dimensionality reduction.

Additionally, a Plotly plot representing the data points in the reduced feature space, with points colored by their categories is displayed.

Examples

```
data(iris)
X <- iris[,-5]
y <- iris[,5]
nn_viz(X, y)          # Default 2D visualization
nn_viz(X, y, dim = 3) # 3D visualization
```

---

pca_viz *PCA Visualization*

Description

Creates a 2D or 3D visualization of multidimensional data using Principal Component Analysis (PCA).

Usage

```
pca_viz(X, y, dim = 2, center.scale = TRUE)
```

Arguments

X	A matrix representing the input features (quantitative variables).
y	A vector representing the categories corresponding to the input data.
dim	Integer indicating the desired dimensionality of the visualization: 2 for 2D, 3 for 3D. Default is 2.
center.scale	A logical (boolean) value indicating whether the data should be centered and scaled before processing.

Value

projected_data The data projected onto the principal components.

transformation_matrix The eigenvectors corresponding to the selected principal components.

Additionally, a plotly object representing the PCA visualization is displayed.

Examples

```
data(iris)
X <- iris[,-5]
y <- iris[,5]
pca_viz(X, y)           # Default 2D visualization
pca_viz(X, y, dim = 3) # 3D visualization
```

---

tsne_viz   *t-SNE Visualization*

Description

Creates a 2D visualization of multidimensional data using t-distributed Stochastic Neighbor Embedding.

Usage

```
tsne_viz(X, y, dim = 2, center.scale = TRUE)
```

Arguments

X	A matrix representing the input features (quantitative variables).
y	A vector representing the categories corresponding to the input data.
dim	Integer indicating the desired dimensionality of the visualization: 2 for 2D, 3 for 3D. Default is 2.
center.scale	A logical (boolean) value indicating whether the data should be centered and scaled before processing.

Value

A list containing:

projected_data The projected data onto the canonical components.

transformation_matrix The eigenvectors corresponding to the canonical components.

Additionally, a Plotly plot representing the data points in the reduced feature space, with points colored by their categories is displayed.

Examples

```
data(iris)
iris <- unique(iris)
X <- iris[,-5]
y <- iris[,5]
tsne_viz(X, y)           # Default 2D visualization
tsne_viz(X, y, dim = 3) # 3D visualization
```

Index