Package 'riemannianStats'

July 23, 2026

Type Package

Title Riemannian Methods for Principal Component Analysis, Regression and Visualization

Version 0.2.0

Description Provides tools for statistical analysis on Riemannian manifolds using local geometry derived from Uniform Manifold Approximation and Projection (UMAP), Isometric Mapping (Isomap), and Density-Based Spatial Clustering of Applications with Noise (DBSCAN). The package supports dimensionality reduction, visualization, Riemannian principal component analysis, and Riemannian linear regression for multivariate data analysis. Methods based on Uniform Manifold Approximation and Projection follow McInnes et al. (2018) <doi:10.21105/joss.00861>.

License BSD_3_clause + file LICENSE

Encoding UTF-8

Language en-US

Depends R (>= 4.1)

Imports rlang, ggplot2, ggrepel, grid, uwot, vegan, dbscan

Suggests scatterplot3d, plotly, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr, rmarkdown

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Oldemar Rodríguez Rojas [aut, cre], Jennifer Lobo Vásquez [aut]

Maintainer Oldemar Rodríguez Rojas <oldemar.rodriguez@ucr.ac.cr>

Repository CRAN

Date/Publication 2026-07-23 15:50:09 UTC

Contents

`coef.riem.lm` *Extract Riemannian Regression Coefficients*

Description

Extract Riemannian Regression Coefficients

Usage

```
## S3 method for class 'riem.lm'
coef(object, ...)
```

Arguments

object	An object of class ‘"riem.lm"‘.
...	Additional arguments passed to or from other methods.

Value

A named numeric vector of coefficients.

---

fitted.riem.lm *Extract Riemannian Fitted Values*

Description

Extract Riemannian Fitted Values

Usage

```
## S3 method for class 'riem.lm'
fitted(object, ...)
```

Arguments

object	An object of class ‘"riem.lm"‘.
...	Additional arguments passed to or from other methods.

Value

A numeric vector of fitted values in the Riemannian transformed scale.

---

print.riem.lm *Print a Riemannian Linear Regression Model*

Description

Print a Riemannian Linear Regression Model

Usage

```
## S3 method for class 'riem.lm'
print(x, ...)
```

Arguments

x	An object of class ‘"riem.lm"‘.
...	Additional arguments passed to or from other methods.

Value

Invisibly returns ‘x‘.

`print.riem.pca` *Print method for Riemannian PCA*

Description

Print method for Riemannian PCA

Usage

```
## S3 method for class 'riem.pca'
print(x, ...)
```

Arguments

`x`	An object of class ‘"riem.pca"‘.
`...`	Additional arguments.

Value

Invisibly returns ‘x‘.

`print.summary.riem.lm` *Print a Summary of a Riemannian Linear Regression Model*

Description

Print a Summary of a Riemannian Linear Regression Model

Usage

```
## S3 method for class 'summary.riem.lm'
print(x, ...)
```

Arguments

`x`	An object of class ‘"summary.riem.lm"‘.
`...`	Additional arguments passed to or from other methods.

Value

Invisibly returns ‘x‘.

residuals.riem.lm *Extract Riemannian Residuals*

Description

Extract Riemannian Residuals

Usage

```
## S3 method for class 'riem.lm'
residuals(object, ...)
```

Arguments

object	An object of class ‘"riem.lm"‘.
...	Additional arguments passed to or from other methods.

Value

A numeric vector of residuals in the Riemannian transformed scale.

riem.biplot *Riemannian PCA Biplot*

Description

Creates a biplot from a Riemannian PCA object or from manually supplied coordinates.

Usage

```
riem.biplot(data, ...)
```

Arguments

data	A ‘"riem.pca"‘ object or a numeric data frame/matrix used by the default biplot method.
...	Additional arguments passed to methods.

Value

A biplot.

`riem.biplot.default` *Default Method for Riemannian PCA Biplots*

Description

Generates a biplot showing individuals on the principal plane and variables as arrows projected onto the same component space.

Usage

```
## Default S3 method:
riem.biplot(
  data,
  components = NULL,
  correlations = NULL,
  clusters = NULL,
  explained.inertia = 0,
  title = "",
  var.scale = NULL,
  point.size = 2,
  alpha = 1,
  show.ind.labels = TRUE,
  show.var.labels = TRUE,
  ind.label.size = 3,
  var.label.size = 3,
  arrow.size = 0.4,
  var.color = "red",
  interactive = FALSE,
  ...
)
```

Arguments

`data`	A data frame containing the original data. Row names are used as individual labels and column names are used as variable labels.
`components`	A matrix or data frame containing at least two components. The first two columns are used to represent individuals on the principal plane.
`correlations`	A matrix or data frame containing the correlations between variables and components. The first two columns are used to draw the variable arrows.
`clusters`	Optional vector of cluster labels used to color individuals according to the cluster they belong to.
`explained.inertia`	Numeric percentage of inertia explained by the plotted components. Defaults to '0'.
`title`	Optional character string added above the default title.

var.scale	Numeric scaling factor for variable arrows. Defaults to 'NULL', which automatically scales arrows to the individual plane.
point.size	Numeric size of the points. Defaults to '2'.
alpha	Numeric transparency value between 0 and 1. Defaults to '1'.
show.ind.labels	Logical. If 'TRUE', displays individual labels. Defaults to 'TRUE'.
show.var.labels	Logical. If 'TRUE', displays variable labels. Defaults to 'TRUE'.
ind.label.size	Numeric size of individual labels. Defaults to '3'.
var.label.size	Numeric size of variable labels. Defaults to '3'.
arrow.size	Numeric line width for variable arrows. Defaults to '0.4'.
var.color	Character string indicating the color of variable arrows and variable labels. Defaults to '"red"'.
interactive	Logical. If 'TRUE', returns an interactive 'plotly' plot. If 'FALSE', returns a static 'ggplot2' plot. Defaults to 'FALSE'.
...	Additional arguments, currently unused.

Details

The biplot combines two types of information: the position of the individuals on the first two Riemannian components and the contribution of the variables represented as arrows. This allows the user to explore both the structure of the observations and the relationship between variables and components.

If 'interactive = TRUE', the static 'ggplot2' object is converted into an interactive 'plotly' object. In that case, hovering over the points displays information about the individuals, while hovering over the variable arrows or labels displays information about the corresponding variables.

Value

If 'interactive = FALSE', a 'ggplot' object. If 'interactive = TRUE', a 'plotly' object.

Examples

```
# Static biplot
data <- iris[, 1:4]
components <- prcomp(data, scale. = TRUE)$x
correlations <- cor(data, components[, 1:2])
riem.biplot(
  data = data,
  components = components,
  correlations = correlations,
  show.ind.labels = FALSE
)

# Interactive biplot
riem.biplot(
  data = data,
```

```
  components = components,
  correlations = correlations,
  show.ind.labels = FALSE,
  interactive = TRUE
)
```

`riem.biplot.riem.pca` *Riemannian PCA Biplot*

Description

Riemannian PCA Biplot

Usage

```
## S3 method for class 'riem.pca'
riem.biplot(data, axes = NULL, title = NULL, ...)
```

Arguments

`data`	An object of class ‘"riem.pca"‘.
`axes`	Integer vector of length 2 indicating the dimensions to plot. If ‘NULL‘, uses ‘data$axes‘.
`title`	Character string. Plot title.
`...`	Additional arguments passed to [riem.biplot.default()].

Value

A biplot.

`riem.center.data` *Center Data Around the Riemannian Mean Observation*

Description

Centers a dataset with respect to the Riemannian mean observation, defined as the row with the smallest total UMAP-based distance to all other rows.

Usage

```
riem.center.data(data, rho, umap.distance.matrix)
```

Arguments

`data`	A numeric data frame or matrix.
`rho`	A square numeric Rho matrix.
`umap.distance.matrix`	A square numeric UMAP-based distance matrix.

Value

A numeric matrix of Riemannian-centered data.

Examples

```
data <- iris[1:5, 1:4]
rho <- matrix(1, nrow = 5, ncol = 5)
dist.matrix <- as.matrix(dist(data))
centered <- riem.center.data(data, rho, dist.matrix)
head(centered)
```

`riem.cor` *Calculate the Riemannian Correlation Matrix*

Description

Calculates the Riemannian correlation matrix from a numeric data frame or matrix, a Rho matrix, and a UMAP-based distance matrix.

Usage

```
riem.cor(data, rho, umap.distance.matrix)
```

Arguments

`data`	A numeric data frame or matrix where rows are observations and columns are variables.
`rho`	A numeric Rho matrix.
`umap.distance.matrix`	A numeric UMAP-based distance matrix.

Value

A numeric Riemannian correlation matrix.

Examples

```
data <- iris[, 1:4]
similarities <- riem.similarities.umap(data, n.neighbors = 5)
rho <- riem.rho(similarities)
riemannian.diff <- riem.diff(data = data, rho = rho)
distance.matrix <- riem.dist(riemannian.diff)

correlation.matrix <- riem.cor(
  data = data,
  rho = rho,
  umap.distance.matrix = distance.matrix
)
```

---

riem.cov *Calculate a Riemannian Covariance Matrix*

Description

Calculates a covariance matrix using Riemannian-centered differences with respect to the Riemannian mean observation.

Usage

```
riem.cov(data, rho, umap.distance.matrix, unbiased = FALSE)
```

Arguments

data	A numeric data frame or matrix.
rho	A square numeric Rho matrix.
umap.distance.matrix	A square numeric UMAP-based distance matrix.
unbiased	Logical. If 'TRUE', divides by 'n - 1'; otherwise divides by 'n'. Defaults to 'FALSE'.

Value

A numeric Riemannian covariance matrix.

Examples

```
data <- iris[1:10, 1:4]
similarities <- diag(nrow(data))
rho <- riem.rho(similarities)
diffs <- riem.diff(data, rho)
dist.matrix <- riem.dist(diffs)
riem.cov(data, rho, dist.matrix)
```

riem.diff *Calculate Riemannian Vector Differences*

Description

Calculates weighted pairwise vector differences between all rows of a numeric dataset using the Rho matrix.

Usage

```
riem.diff(data, rho)
```

Arguments

data	A numeric data frame or matrix where rows are observations and columns are variables.
rho	A square numeric matrix with the same number of rows as ‘data‘.

Value

A three-dimensional numeric array. Entry ‘[i, j,]‘ contains ‘rho[i, j] * (data[i,] - data[j,])‘.

Examples

```
data <- iris[1:5, 1:4]
rho <- matrix(1, nrow = 5, ncol = 5)
diffs <- riem.diff(data, rho)
dim(diffs)
```

riem.dist *Calculate the Distance Matrix*

Description

Calculates a pairwise distance matrix from a tensor of Riemannian vector differences.

Usage

```
riem.dist(riemannian.diff)
```

Arguments

riemannian.diff	A three-dimensional array where entry ‘[i, j,]‘ contains the weighted vector difference between observations ‘i‘ and ‘j‘.

Value

A square numeric distance matrix.

Examples

```
data <- iris[1:5, 1:4]
rho <- matrix(1, nrow = 5, ncol = 5)
diffs <- riem.diff(data, rho)
riem.dist(diffs)
```

`riem.ind.coord` *Calculate Riemannian Principal Components*

Description

Performs Riemannian principal component analysis from a numeric data frame or matrix, a Riemannian correlation matrix, a Rho matrix, and a UMAP-based distance matrix.

Usage

```
riem.ind.coord(
  data,
  correlation.matrix,
  rho,
  umap.distance.matrix,
  sign.convention = c("largest.negative", "largest.positive", "none")
)
```

Arguments

`data`	A numeric data frame or matrix where rows are observations and columns are variables.
`correlation.matrix`	A square numeric correlation matrix.
`rho`	A numeric Rho matrix.
`umap.distance.matrix`	A numeric UMAP-based distance matrix.
`sign.convention`	Character string indicating how to orient the signs of the eigenvectors. Options are ‘"largest.negative"‘, ‘"largest.positive"‘, and ‘"none"‘. Defaults to ‘"largest.negative"‘.

Details

The sign of principal components is arbitrary. This function uses a sign convention to make component orientations reproducible.

Value

A numeric matrix of Riemannian principal components.

Examples

```
data <- iris[, 1:4]
similarities <- riem.similarities.umap(data, n.neighbors = 5)
rho <- riem.rho(similarities)
riemannian.diff <- riem.diff(data, rho = rho)
distance.matrix <- riem.dist(riemannian.diff)

correlation.matrix <- riem.cor(
  data = data,
  rho = rho,
  umap.distance.matrix = distance.matrix
)

components <- riem.ind.coord(
  data = data,
  correlation.matrix = correlation.matrix,
  rho = rho,
  umap.distance.matrix = distance.matrix
)

head(components)
```

---

riem.inertia *Calculate PCA Inertia for Two Components*

Description

Calculates the proportion of total inertia, or explained variance, associated with two selected principal components from a correlation matrix.

Usage

```
riem.inertia(correlation.matrix, component1, component2)
```

Arguments

correlation.matrix	A square numeric correlation matrix.
component1	Integer index of the first selected component.
component2	Integer index of the second selected component.

Details

In R, component indices start at 1. Therefore, the first principal component is ‘component1 = 1‘, not ‘component1 = 0‘.

Value

A numeric value between 0 and 1.

Examples

```
data <- iris[, 1:4]
similarities <- riem.similarities.umap(data, n.neighbors = 5)
rho <- riem.rho(similarities)
riemannian.diff <- riem.diff(data, rho = rho)
distance.matrix <- riem.dist(riemannian.diff)

correlation.matrix <- riem.cor(
  data = data,
  rho = rho,
  umap.distance.matrix = distance.matrix
)

components <- riem.ind.coord(
  data = data,
  correlation.matrix = correlation.matrix,
  rho = rho,
  umap.distance.matrix = distance.matrix
)
riem.inertia(correlation.matrix, component1 = 1, component2 = 2)
```

---

riem.lm *Riemannian Linear Regression*

Description

Fits a Riemannian linear regression model using a similarity matrix induced by UMAP, Isomap or DBSCAN.

Usage

```
riem.lm(
  formula = NULL,
  data = NULL,
  y = NULL,
  na.action = stats::na.omit,
  similarities = c("umap", "isomap", "dbscan"),
  n.neighbors = 3,
```

```
  min.dist = NULL,
  metric = "euclidean",
  tol = 1e-10,
  verbose = FALSE,
  max.neighbors = NULL,
  sigma = NULL,
  eps = NULL,
  eps.quantile = 0.9,
  minPts = NULL,
  density.power = 1,
  between.cluster.factor = 0.05,
  noise.factor = 0.1,
  use.only.eps.neighborhood = FALSE
)
```

Arguments

formula	An object of class [stats::formula()] describing the model, with the numeric response on the left of '~' and the explanatory variables on the right. Alternatively, a numeric data frame or matrix can be supplied here for compatibility with the former positional interface.
data	A data frame containing the variables referenced in 'formula'. With the former interface, it can instead be the numeric matrix of explanatory variables when 'formula = NULL'.
y	Numeric response vector used only by the former 'data' and 'y' interface. Its length must equal the number of rows in 'data'.
na.action	A function that indicates what should happen when the model data contain missing values. Defaults to [stats::na.omit()].
similarities	Character string. Similarity method used to construct the Riemannian structure. Must be one of '"umap"', '"isomap"' or '"dbscan"'. Defaults to '"umap"'.
n.neighbors	Integer. Number of neighbors used to construct the local similarity structure. Defaults to '3'.
min.dist	Numeric or 'NULL'. Kept for API compatibility with UMAP workflows and passed to [riem.similarities.umap()]. Defaults to 'NULL'.
metric	Character string. Distance metric passed to the selected similarity function. Defaults to '"euclidean"'.
tol	Numeric. Tolerance used to detect near-singular systems. Defaults to '1e-10'.
verbose	Logical. If 'TRUE', prints progress information from the selected similarity method. Defaults to 'FALSE'.
max.neighbors	Integer or 'NULL'. Maximum number of neighbors to try when 'similarities = "isomap"'. For other methods, it is kept only for API compatibility and is not used. Defaults to 'NULL'.
sigma	Numeric or 'NULL'. Scale parameter used when 'similarities = "dbscan"'. If 'NULL', it is estimated internally.
eps	Numeric or 'NULL'. DBSCAN radius parameter used when 'similarities = "dbscan"'. If 'NULL', it is estimated internally.

`eps.quantile` Numeric. Quantile used to estimate ‘eps‘ when ‘similarities = "dbscan"‘ and ‘eps = NULL‘. Defaults to ‘0.90‘.

`minPts` Integer or ‘NULL‘. Minimum number of points used by DBSCAN when ‘similarities = "dbscan"‘. If ‘NULL‘, uses ‘n.neighbors‘.

`density.power` Numeric. Exponent controlling the local density effect when ‘similarities = "dbscan"‘. Defaults to ‘1‘.

`between.cluster.factor`
Numeric between ‘0‘ and ‘1‘. Penalization factor for pairs of observations assigned to different non-noise DBSCAN clusters. Defaults to ‘0.05‘.

`noise.factor` Numeric between ‘0‘ and ‘1‘. Penalization factor for pairs where at least one observation is classified as noise by DBSCAN. Defaults to ‘0.10‘.

`use.only.eps.neighborhood`
Logical. If ‘TRUE‘, DBSCAN similarities are kept only for pairs of observations that belong to the same ‘eps‘ neighborhood. Defaults to ‘FALSE‘.

Details

The function first constructs a similarity matrix using one of [riem.similarities.umap()], [riem.similarities.isomap()] or [riem.similarities.dbscan()]. Then, it computes the corresponding Riemannian dissimilarity matrix with [riem.rho()], obtains Riemannian differences with [riem.diff()], and selects the Riemannian mean index using [riem.mean.index()].

The regression is fitted after centering the explanatory variables and response around the Riemannian center and weighting the centered data using the Riemannian weights.

Value

An object of class ‘"riem.lm"‘, which is a list containing:

- ‘beta.R‘: Riemannian regression coefficients.
- ‘yhat.R‘: fitted values in the Riemannian transformed scale.
- ‘y.R‘: response vector in the Riemannian transformed scale.
- ‘residuals.R‘: residuals in the Riemannian transformed scale.
- ‘R2.R‘: Riemannian coefficient of determination.
- ‘lambda‘: index of the Riemannian mean observation.
- ‘x.lambda‘: explanatory variables at the Riemannian center.
- ‘y.lambda‘: response value at the Riemannian center.
- ‘weights‘: Riemannian weights derived from ‘rho‘.
- ‘similarities‘: similarity matrix.
- ‘rho‘: Riemannian dissimilarity matrix.
- ‘distance.matrix‘: Riemannian distance matrix.

Examples

```
model <- riem.lm(
  Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width,
  data = iris,
  similarities = "umap",
  n.neighbors = 5
)

model
coef(model)
summary(model)

original <- riem.original.scale(model)
head(original$table)
```

`riem.mean.index` *Find the Riemannian Mean Observation Index*

Description

Returns the index of the observation with the smallest total distance to all other observations.

Usage

```
riem.mean.index(umap.distance.matrix)
```

Arguments

`umap.distance.matrix`
: A square numeric distance matrix.

Value

An integer index. In R, indexing starts at 1.

Examples

```
d <- as.matrix(dist(iris[1:5, 1:4]))
riem.mean.index(d)
```

riem.norm *Calculate the Euclidean Norm*

Description

Calculates the Euclidean norm of a vector or the Euclidean distance between two vectors.

Usage

```
riem.norm(x, y = NULL)
```

Arguments

x	Numeric vector.
y	Optional numeric vector. If supplied, the norm of ‘x - y‘ is computed.

Value

A single numeric value.

Examples

```
riem.norm(c(3, 4))
riem.norm(c(1, 2), c(4, 6))
```

riem.original.scale *Convert a Riemannian Regression Fit to the Original Scale*

Description

Converts fitted values and residuals from a Riemannian regression model back to the original response scale.

Usage

```
riem.original.scale(object)
```

Arguments

object	An object of class ‘"riem.lm"‘.

Details

The fitted values stored in a ‘"riem.lm"‘ object are computed in the Riemannian transformed scale. This function reconstructs fitted values in the original response scale using the Riemannian center.

Value

A list containing original response values, fitted values, residuals, original-scale sums of squares, original-scale coefficients of determination and a data frame with observation-level results.

Examples

```
model <- riem.lm(
  data = iris[, 1:4],
  y = iris$Sepal.Length,
  similarities = "umap",
  n.neighbors = 5
)

original <- riem.original.scale(model)
head(original$table)
```

---

riem.pca *Riemannian Principal Component Analysis Riemannian Principal Component Analysis*

Description

Performs Riemannian principal component analysis using a similarity matrix induced by UMAP, Isomap or DBSCAN.

Usage

```
riem.pca(
  data,
  scale.unit = TRUE,
  ncp = 5,
  axes = c(1, 2),
  n.neighbors = 3,
  min.dist = 0.1,
  metric = "euclidean",
  method = c("umap", "isomap", "dbscan"),
  verbose = FALSE,
  max.neighbors = NULL,
  sigma = NULL,
  eps = NULL,
  eps.quantile = 0.9,
  minPts = NULL,
  density.power = 1,
  between.cluster.factor = 0.05,
  noise.factor = 0.1,
  use.only.eps.neighborhood = FALSE
)
```

Arguments

data	A numeric data frame or matrix where rows are observations and columns are quantitative variables.
scale.unit	Logical. If ‘TRUE‘, variables are centered and scaled to unit variance before the analysis. If ‘FALSE‘, the original data are used without centering or scaling. Defaults to ‘TRUE‘.
ncp	Integer. Number of dimensions kept in the results. Defaults to ‘5‘.
axes	Integer vector of length 2. Axes to be used by plotting functions. Defaults to ‘c(1, 2)‘.
n.neighbors	Integer. Number of neighbors used to construct the local similarity structure. Defaults to ‘3‘.
min.dist	Numeric or ‘NULL‘. UMAP minimum distance parameter. It is used only when ‘method = "umap"‘. Defaults to ‘0.1‘.
metric	Character string. Distance metric passed to the selected similarity function. Defaults to ‘"euclidean"‘.
method	Character string. Similarity method used to construct the Riemannian structure. Must be one of ‘"umap"‘, ‘"isomap"‘ or ‘"dbscan"‘. Defaults to ‘"umap"‘.
verbose	Logical. If ‘TRUE‘, prints progress information from the selected similarity method. Defaults to ‘FALSE‘.
max.neighbors	Integer or ‘NULL‘. Maximum number of neighbors to try when ‘method = "isomap"‘ or ‘method = "dbscan"‘. Defaults to ‘NULL‘.
sigma	Numeric or ‘NULL‘. Scale parameter used when ‘method = "dbscan"‘. If ‘NULL‘, it is estimated internally.
eps	Numeric or ‘NULL‘. DBSCAN radius parameter used when ‘method = "dbscan"‘. If ‘NULL‘, it is estimated internally.
eps.quantile	Numeric. Quantile used to estimate ‘eps‘ when ‘method = "dbscan"‘ and ‘eps = NULL‘. Defaults to ‘0.90‘.
minPts	Integer or ‘NULL‘. Minimum number of points used by DBSCAN when ‘method = "dbscan"‘. If ‘NULL‘, uses ‘n.neighbors‘.
density.power	Numeric. Exponent controlling the local density effect when ‘method = "dbscan"‘. Defaults to ‘1‘.
between.cluster.factor	Numeric between ‘0‘ and ‘1‘. Penalization factor for pairs of observations assigned to different non-noise DBSCAN clusters. Defaults to ‘0.05‘.
noise.factor	Numeric between ‘0‘ and ‘1‘. Penalization factor for pairs where at least one observation is classified as noise by DBSCAN. Defaults to ‘0.10‘.
use.only.eps.neighborhood	Logical. If ‘TRUE‘, DBSCAN similarities are kept only for pairs of observations that belong to the same ‘eps‘ neighborhood. Defaults to ‘FALSE‘.

Value

An object of class ‘"riem.pca"‘, which is a list containing:

- ‘eig‘: eigenvalues, percentage of explained variance and cumulative percentage of explained variance.
- ‘var‘: results for the variables.
- ‘var$coord‘: coordinates of the variables on the retained dimensions.
- ‘var$cor‘: Riemannian correlation matrix.
- ‘var$cov‘: Riemannian covariance matrix.
- ‘ind‘: results for the individuals.
- ‘ind$similarities‘: similarity matrix.
- ‘ind$rho‘: Riemannian dissimilarity matrix.
- ‘ind$differences‘: Riemannian differences.
- ‘ind$distance.matrix‘: Riemannian distance matrix.
- ‘ind$coord‘: coordinates of the individuals on the retained dimensions.

Examples

```
model <- riem.pca(
  data = iris[, 1:4],
  scale.unit = TRUE,
  ncp = 2,
  method = "umap",
  n.neighbors = 5
)

model
round(model$eig, 4)
head(model$ind$coord)
model$var$coord
```

`riem.plot` *Plot Riemannian PCA Results*

Description

Plots individuals or variables from a Riemannian PCA object or from manually supplied coordinates.

Usage

```
riem.plot(data, ...)
```

Arguments

data	A ‘"riem.pca"‘ object or a numeric data frame/matrix used by the default plotting method.
...	Additional arguments passed to methods.

Value

A plot.

---

riem.plot.default *Default Method for Riemannian PCA Plots*

Description

Default Method for Riemannian PCA Plots

Usage

```
## Default S3 method:
riem.plot(
  data,
  choix = c("ind", "var"),
  components = NULL,
  correlations = NULL,
  clusters = NULL,
  explained.inertia = 0,
  title = "",
  ...
)
```

Arguments

data	A numeric data frame or matrix.
choix	Character string. Must be one of ‘"ind"‘ or ‘"var"‘.
components	Coordinates of the individuals.
correlations	Coordinates or correlations of the variables.
clusters	Optional vector of cluster labels used to color individuals.
explained.inertia	Percentage of explained inertia for the plotted axes.
title	Character string. Plot title.
...	Additional arguments.

Value

A plot.

Examples

```
data <- iris[, 1:4]
components <- prcomp(data, scale. = TRUE)$x
correlations <- cor(data, components[, 1:2])

riem.plot(
  data = data,
  choix = "ind",
  components = components
)

riem.plot(
  data = data,
  choix = "var",
  correlations = correlations
)
```

`riem.plot.riem.pca` *Plot Riemannian PCA Results*

Description

Plot Riemannian PCA Results

Usage

```
## S3 method for class 'riem.pca'
riem.plot(data, choix = c("ind", "var"), axes = NULL, title = NULL, ...)
```

Arguments

`data`	An object of class ‘"riem.pca"‘.
`choix`	Character string. Must be one of ‘"ind"‘ or ‘"var"‘.
`axes`	Integer vector of length 2 indicating the dimensions to plot. If ‘NULL‘, uses ‘data$axes‘.
`title`	Character string. Plot title.
`...`	Additional arguments passed to [riem.plot.default()].

Value

A plot.

`riem.plot3d` *Plot a Three-Dimensional Scatter Plot*

Description

Creates an interactive three-dimensional scatter plot from selected columns of a data frame. Points can optionally be colored according to a cluster column.

Usage

```
riem.plot3d(
  data,
  x.col,
  y.col,
  z.col,
  cluster.col = NULL,
  label.col = NULL,
  title = "",
  explained.inertia = 0,
  point.size = 5,
  alpha = 1
)
```

Arguments

`data`	A data frame containing the variables to plot.
`x.col`	Character string with the name of the x-axis column.
`y.col`	Character string with the name of the y-axis column.
`z.col`	Character string with the name of the z-axis column.
`cluster.col`	Optional character string with the name of the cluster column. If provided, points are colored according to the cluster they belong to. Defaults to 'NULL'.
`label.col`	Optional character string with the name of the column used as individual labels. If 'NULL', row names are used. Defaults to 'NULL'.
`title`	Optional character string added above the default title.
`explained.inertia`	Numeric percentage of inertia explained by the plotted components. Defaults to '0'.
`point.size`	Numeric size of the markers. Defaults to '5'.
`alpha`	Numeric transparency value between 0 and 1. Defaults to '1'.

Details

This function requires the optional package 'plotly'.

Value

A ‘plotly‘ htmlwidget.

Examples

```
riem.plot3d(
  data = iris,
  x.col = "Sepal.Length",
  y.col = "Sepal.Width",
  z.col = "Petal.Length"
)

riem.plot3d(
  data = iris,
  x.col = "Sepal.Length",
  y.col = "Sepal.Width",
  z.col = "Petal.Length",
  cluster.col = "Species"
)
```

---

riem.rho *Calculate the Rho Matrix*

Description

Calculates the Rho matrix as one minus the UMAP similarity matrix.

Usage

```
riem.rho(umap.similarities)
```

Arguments

umap.similarities	A numeric matrix of UMAP graph similarities.

Details

The Rho matrix is used to weight pairwise vector differences in the Riemannian analysis pipeline.

Value

A numeric matrix computed as ‘1 - umap.similarities‘.

Examples

```
similarities <- diag(3)
riem.rho(similarities)
```

`riem.similarities.dbscan` *Calculate DBSCAN-Induced Similarities*

Description

Calculates a dense matrix of similarities induced by DBSCAN clustering, local density and pairwise distances.

Usage

```
riem.similarities.dbscan(
  data,
  n.neighbors = 3,
  metric = "euclidean",
  sigma = NULL,
  max.neighbors = NULL,
  verbose = FALSE,
  eps = NULL,
  eps.quantile = 0.9,
  minPts = NULL,
  density.power = 1,
  between.cluster.factor = 0.05,
  noise.factor = 0.1,
  use.only.eps.neighborhood = FALSE
)
```

Arguments

`data`	A numeric data frame or matrix where rows are observations and columns are variables.
`n.neighbors`	Integer. Number of neighbors used as the default value for ‘minPts‘ when ‘minPts = NULL‘. Defaults to ‘3‘.
`metric`	Character string. Distance metric passed to [stats::dist()]. Supported values are ‘"euclidean"‘, ‘"maximum"‘, ‘"manhattan"‘, ‘"canberra"‘, ‘"binary"‘ and ‘"minkowski"‘. Unambiguous substrings are allowed, following [stats::dist()]. Defaults to ‘"euclidean"‘.
`sigma`	Numeric or ‘NULL‘. Scale parameter used in the distance-based similarity term. If ‘NULL‘, the median positive pairwise distance is used.
`max.neighbors`	‘NULL‘ or any value. Kept only for API compatibility with other similarity functions. It is not used by DBSCAN. Defaults to ‘NULL‘.
`verbose`	Logical. If ‘TRUE‘, prints information about the selected DBSCAN parameters and local densities. Defaults to ‘FALSE‘.
`eps`	Numeric or ‘NULL‘. DBSCAN radius parameter. If ‘NULL‘, it is estimated from the ‘eps.quantile‘ quantile of the ‘minPts‘ nearest-neighbor distances.

`eps.quantile` Numeric. Quantile used to estimate ‘eps‘ when ‘eps = NULL‘. Defaults to ‘0.90‘.

`minPts` Integer or ‘NULL‘. Minimum number of points used by DBSCAN. If ‘NULL‘, uses ‘n.neighbors‘.

`density.power` Numeric. Exponent controlling the effect of local density on the similarity matrix. Defaults to ‘1‘.

`between.cluster.factor` Numeric between ‘0‘ and ‘1‘. Penalization factor for pairs of observations assigned to different non-noise clusters. Defaults to ‘0.05‘.

`noise.factor` Numeric between ‘0‘ and ‘1‘. Penalization factor for pairs where at least one observation is classified as noise by DBSCAN. Defaults to ‘0.10‘.

`use.only.eps.neighborhood` Logical. If ‘TRUE‘, similarities are kept only for pairs of observations that belong to the same ‘eps‘ neighborhood. Defaults to ‘FALSE‘.

Details

This function combines distance-based similarity, local density information and DBSCAN cluster membership to construct a similarity matrix suitable for Riemannian statistical workflows.

The similarity between observations is computed from three components:

$$S_{dist}(i,j) = \exp(-d_{ij}/\sigma)$$

where ‘d_ij‘ is the pairwise distance between observations ‘i‘ and ‘j‘.

This distance-based similarity is then multiplied by a local density factor and a cluster-membership factor. Observations in the same non-noise DBSCAN cluster receive no cluster penalization, observations in different clusters receive the factor ‘between.cluster.factor‘, and pairs involving DBSCAN noise receive the factor ‘noise.factor‘.

The diagonal is set to zero to follow the same convention used by ‘riem.similarities.umap()‘.

Value

A dense numeric matrix of DBSCAN-induced similarities. The returned matrix has attributes ‘"eps"‘, ‘"minPts"‘, ‘"sigma"‘, ‘"clusters"‘, ‘"local.density"‘, ‘"density.scaled"‘, ‘"dbscan.object"‘, ‘"metric"‘ and ‘"type"‘.

Examples

```
similarities <- riem.similarities.dbscan(
  iris[, 1:4],
  n.neighbors = 5
)
dim(similarities)
attr(similarities, "eps")
table(attr(similarities, "clusters"))
```

`riem.similarities.isomap` *Calculate Isomap Graph Similarities*

Description

Calculates a dense matrix of similarities induced by Isomap geodesic distances from a numeric dataset.

Usage

```
riem.similarities.isomap(
  data,
  n.neighbors = 3,
  metric = "euclidean",
  max.neighbors = NULL,
  verbose = FALSE
)
```

Arguments

`data`	A numeric data frame or matrix where rows are observations and columns are variables.
`n.neighbors`	Integer. Initial number of nearest neighbors used to construct the Isomap graph. Defaults to ‘3‘.
`metric`	Character string. Distance metric passed to [stats::dist()]. Supported values are ‘"euclidean"‘, ‘"maximum"‘, ‘"manhattan"‘, ‘"canberra"‘, ‘"binary"‘ and ‘"minkowski"‘. Unambiguous substrings are allowed, following [stats::dist()]. Defaults to ‘"euclidean"‘.
`max.neighbors`	Integer or ‘NULL‘. Maximum number of neighbors to try when searching for a connected Isomap graph. If ‘NULL‘, uses ‘n - 1‘, where ‘n‘ is the number of observations.
`verbose`	Logical. If ‘TRUE‘, prints information about the selected number of neighbors. Defaults to ‘FALSE‘.

Details

This function builds an Isomap neighborhood graph using pairwise distances and converts the resulting geodesic distance matrix into a normalized similarity matrix.

The function first computes the original pairwise distances using [stats::dist()]. Then, it attempts to construct a connected Isomap graph using [vegan::isomapdist()]. If the graph is not connected with ‘n.neighbors‘, the number of neighbors is increased until a connected graph is obtained or ‘max.neighbors‘ is reached.

The Isomap geodesic distance matrix ‘D‘ is transformed into similarities as

$$S_{ij} = 1 - \frac{D_{ij}}{\max(D)}$$

.

The diagonal is set to zero to follow the same convention used by ‘riem.similarities.umap()‘.

Value

A dense numeric matrix of Isomap-induced similarities. The returned matrix has attributes ‘"k.used"‘, ‘"D.iso"‘, ‘"d.max"‘, ‘"metric"‘ and ‘"type"‘.

Examples

```
similarities <- riem.similarities.isomap(
  iris[, 1:4],
  n.neighbors = 5
)
dim(similarities)
attr(similarities, "k.used")
```

`riem.similarities.umap`

Calculate UMAP Graph Similarities

Description

Calculates a matrix of UMAP graph similarities from the local neighborhood graph of a numeric dataset.

Usage

```
riem.similarities.umap(
  data,
  n.neighbors = 3,
  min.dist = NULL,
  metric = "euclidean",
  sparse = FALSE,
  verbose = FALSE
)
```

Arguments

`data`	A numeric data frame or matrix where rows are observations and columns are variables.
`n.neighbors`	Integer. Number of nearest neighbors used to construct the local graph. Defaults to ‘3‘.

min.dist	Numeric or ‘NULL‘. Kept only for API compatibility with UMAP workflows. This parameter is not used by this function because [uwot::similarity_graph()] constructs the similarity graph directly from local neighborhoods and the distance metric. Defaults to ‘NULL‘.
metric	Character string. Distance metric used by UMAP. Supported values are ‘"euclidean"‘, ‘"cosine"‘, ‘"manhattan"‘, ‘"hamming"‘ and ‘"correlation"‘. Defaults to ‘"euclidean"‘.
sparse	Logical. If ‘TRUE‘, returns the sparse graph returned by ‘uwot‘; if ‘FALSE‘, returns a dense matrix. Defaults to ‘FALSE‘.
verbose	Logical. If ‘TRUE‘, prints information about the graph construction. Defaults to ‘FALSE‘.

Details

This function is the R equivalent of extracting ‘reducer.graph_‘ from ‘umap-learn‘ in Python and converting it to a similarity matrix.

The function constructs a fuzzy similarity graph using [uwot::similarity_graph()]. The diagonal is set to zero to follow the convention used by the other similarity functions in ‘riemannianStats‘.

When ‘sparse = FALSE‘, the graph is returned as a dense numeric matrix. When ‘sparse = TRUE‘, the sparse graph returned by ‘uwot‘ is preserved.

Value

A numeric matrix or sparse matrix of UMAP similarities. The returned object has attributes ‘"n.neighbors"‘, ‘"min.dist"‘, ‘"metric"‘ and ‘"type"‘.

Examples

```
similarities <- riem.similarities.umap(
  iris[, 1:4],
  n.neighbors = 5
)
dim(similarities)
attr(similarities, "type")
```

---

riem.var.coord *Calculate Riemannian Correlations Between Variables and Components*

Description

Calculates Riemannian correlations between the original variables and the first two principal components.

Usage

```
riem.var.coord(data, components, rho, umap.distance.matrix)
```

Arguments

data	A numeric data frame or matrix where rows are observations and columns are variables.
components	A numeric matrix or data frame with at least two columns.
rho	A numeric Rho matrix.
umap.distance.matrix	A numeric UMAP-based distance matrix.

Value

A data frame with columns 'Component.1' and 'Component.2'.

Examples

```
data <- iris[, 1:4]
similarities <- riem.similarities.umap(data, n.neighbors = 5)
rho <- riem.rho(similarities)
riemannian.diff <- riem.diff(data, rho = rho)
distance.matrix <- riem.dist(riemannian.diff)

correlation.matrix <- riem.cor(
  data = data,
  rho = rho,
  umap.distance.matrix = distance.matrix
)

components <- riem.ind.coord(
  data = data,
  correlation.matrix = correlation.matrix,
  rho = rho,
  umap.distance.matrix = distance.matrix
)

correlations <- riem.var.coord(
  data = data,
  components = components,
  rho = rho,
  umap.distance.matrix = distance.matrix
)
```

---

summary.riem.lm *Summarize a Riemannian Linear Regression Model*

Description

Summarize a Riemannian Linear Regression Model

Usage

```
## S3 method for class 'riem.lm'
summary(object, ...)
```

Arguments

`object`	An object of class ‘"riem.lm"‘.
`...`	Additional arguments passed to or from other methods.

Value

An object of class ‘"summary.riem.lm"‘.

Index