Package 'pandemonium'

November 3, 2025

Title High Dimensional Analysis in Linked Spaces

Version 0.2.4

Description A 'shiny' GUI that performs high dimensional cluster analysis.
This tool performs data preparation, clustering and visualisation within a dynamic GUI.
With interactive methods allowing the user to change settings all without having to to leave the GUI.
An earlier version of this package was described in Laa and Valencia (2022) <doi:10.1140/epjp/s13360-021-02310-1>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.5)

Imports tourr, stats, tibble, ggplot2, RColorBrewer, dplyr, dendextend, fpc, shiny, shinythemes, DT, tidyr, tidyselect, magrittr, detourr (>= 0.2.0), crosstalk, Rtsne, plotly, alphahull, uwot, shinyFeedback, ggpcp, rlang, viridis

LazyData true

Suggests knitr, readr, rmarkdown, testthat (>= 3.0.0), VIM

VignetteBuilder knitr

Config/testthat/edition 3

URL https://gabrielmccoy.github.io/pandemonium/

NeedsCompilation no

Author Gabriel McCoy [aut, cre],
Ursula Laa [aut] (ORCID: <https://orcid.org/0000-0002-0249-6439>),
German Valencia [aut] (ORCID: <https://orcid.org/0000-0001-6600-1290>)

Maintainer Gabriel McCoy <gabe.mccoy02@gmail.com>

Repository CRAN

Date/Publication 2025-11-03 08:30:02 UTC

Contents

`Bikes` *Bike sharing data with model information*

Description

The dataset contains daily counts of bikes rented with corresponding weather and seasonal information. The data is provided by Hadi Fanaee-T and available from https://doi.org/10.24432/C5W894. Additionally, model information from a single hidden layer neural network with eight nodes in the hidden layer has been added: the values of the activations for all observations (variables A1 to A8 in the cluster space) and the model prediction (pred) and residual (res) in the other variables.

Usage

```
Bikes
```

Format

a list of 4 dataframes

df dataframe 731 obs of 18 variables containing the entire bikes data set

space1 dataframe 731 obs of 8 variables (cluster space)

space2 dataframe 731 obs of 6 variables (linked space, predictors used in the model)

other dataframe 731 obs of 4 variables (other variables, including observed and predicted counts)

chi2score *Chi-squared scores function*

Description

Can be used as getScores input in pandemonium. Returns chi-squared values as the score and sigma bins as the bins.

Usage

```
chi2score(space1, covinv, exp, ...)
```

Arguments

space1	dataframe with variables in space1
covinv	inverse covariance matrix from space1
exp	reference point from space 1
...	other expected values of getScore

Value

named list containing scores for use in pandemonium

Examples

```
chi2score(Bikes$space1,solve(cov(Bikes$space1)),
            data.frame(value = colMeans(Bikes$space1)))
```

getBenchmarkInformation *Compute cluster information*

Description

The returned tibble contains the id of the cluster benchmark, the cluster radius and diameter, and group number for each cluster.

Usage

```
getBenchmarkInformation(dmat, groups)
```

Arguments

dmat	distance matrix
groups	groups resulting from clustering

Value

data frame with cluster information

Examples

```
dists <- getDists(Bikes$space1,"euclidean")
fit <- stats::hclust(dists, "ward.D2")
groups <- stats::cutree(fit, k = 4)
getBenchmarkInformation(as.matrix(dists), groups)
```

---

getClusterDists *Compute cluster distance summaries*

Description

The returned tibble contains the id of the cluster pairs, with benchmark distance (d1), minimum (d2) and maximum (d3) distances between any points in the two clusters.

Usage

```
getClusterDists(dmat, groups, benchmarks)
```

Arguments

dmat	distance matrix
groups	groups resulting from clustering
benchmarks	data frame with benchmark id and group number

Value

data frame with distance information

Examples

```
dists <- getDists(Bikes$space1,"euclidean")
fit <- stats::hclust(dists, "ward.D2")
groups <- stats::cutree(fit, k = 4)
bm <- getBenchmarkInformation(as.matrix(dists), groups)
getClusterDists(as.matrix(dists), groups, bm)
```

getDists *Compute distances between all points*

Description

Compute distances between all points

Usage

```
getDists(coord, metric, user_dist = NULL)
```

Arguments

coord	matrix with coordinate representation of all points
metric	name of distance metric to be used in stats::dist
user_dist	user distance returned with metric=user

Value

distances between all points

Examples

```
getDists(Bikes$space1[1:5,],"euclidean")
getDists(Bikes$space1[1:5,],"maximum")
```

makePlots *Generate a specified plot outside the GUI*

Description

An interface to generate a specific graph seen when using the GUI. Settings include: metric, linkage, k, plotType, for details see the vignette on using this function.

Usage

```
makePlots(
  space1,
  settings,
  cov = NULL,
  covInv = NULL,
  exp = NULL,
  space2 = NULL,
  space2.cov = NULL,
```

```
  space2.covInv,
  space2.exp = NULL,
  user_dist = NULL,
  getCoordsSpace1 = normCoords,
  getCoordsSpace2 = normCoords,
  getScore = NULL
)
```

Arguments

space1	dataframe of variables in cluster space
settings	list specifying parameters usually selected in the app
cov	covariance matrix for space 1
covInv	inverse covariance matrix for space 1
exp	reference point in space 1
space2	dataframe of variables in linked space
space2.cov	covariance matrix for space 2
space2.covInv	inverse covariance matrix for space 2
space2.exp	reference point in space 2
user_dist	user defined distances
getCoordsSpace1	function to calculate coordinates in space 1
getCoordsSpace2	function to calculate coordinates in space 2
getScore	function to calculate scores and bins

Value

ggplot, plotly or detourr plot depending on settings$plotType

Examples

```
makePlots(space1 = Bikes$space1,
  settings = list(
      plotType = "WC", x="hum", y="temp", k=4, metric="euclidean",
      linkage="ward.D2", WCa=0.5, showalpha=TRUE),cov = cov(Bikes$space1),
      space2 = Bikes$space2, getScore = outsidescore(Bikes$other$res,"Residual"))

makePlots(space1 = Bikes$space1,
  settings = list(
      plotType = "tour", k=4, metric="euclidean", linkage="ward.D2",
      tourspace="space1", colouring="clustering", out_dim=2, tour_path="grand",
     display="scatter",radial_start=NULL, radial_var=NULL, slice_width=NULL, seed = 2025),
      cov = cov(Bikes$space1), space2 = Bikes$space2,
      getScore = outsidescore(Bikes$other$res,"Residual"))
```

normCoords *Scaled coordinates*

Description

Using scale to center and scale the coordinates.

Usage

```
normCoords(df, ...)
```

Arguments

df	data frame
...	other expected values of getCoords

Value

matrix with coordinate representation of all points

Examples

```
head(normCoords(Bikes$space2))
```

outsidescore *Using externally computed score values*

Description

Can be used as getScores input in pandemonium, to use score values that are computed externally. Returns scores values as the score, and bins computed as below, between or above the first and third quartile.

Usage

```
outsidescore(scores, scoreName = NULL)
```

Arguments

scores	external scores to be passed to the app.
scoreName	name for scores

Value

named list containing scores for use in pandemonium

Examples

```
pandemonium(df = Bikes$space1, space2 = Bikes$space2,
              getScore = outsidescore(Bikes$other$res,"Residual"))
```

---

pandemonium *Shiny app for exploring clustering solutions*

Description

Opening the GUI to cluster the data points based on values in space2. Coordinates and distances are computed on the fly, or can be entered in the function call.

Usage

```
pandemonium(
  df,
  cov = NULL,
  is.inv = FALSE,
  exp = NULL,
  space2 = NULL,
  space2.cov = NULL,
  space2.exp = NULL,
  group = NULL,
  label = NULL,
  user_dist = NULL,
  dimReduction = list(tSNE = tSNE, umap = umap),
  getCoords = list(normal = normCoords),
  getScore = NULL
)
```

Arguments

df	data frame of data, assumes space 1 but variables can be re-assigned in the app
cov	covariance matrix (optional)
is.inv	is the covariance matrix an inverse default FALSE
exp	observable reference value (e.g. experimental measurement)
space2	data frame assumed to be in space 2 but variables can be re-assigned in the app
space2.cov	covariance matrix (optional)
space2.exp	observable reference value (e.g. experimental measurement)
group	grouping assignments
label	point labels
user_dist	input distance matrix (optional)

dimReduction	named list of functions used for dimension reduction
getCoords	named list containing functions to calculate coordinates
getScore	named list containing functions to calculate scores to be plotted as bins and continuous value.

Value

No return value, called to initiate 'shiny' app

`pullCoords` *Chi-Squared Loss Function Coordinates*

Description

Computes coordinate values by comparing observed values to the reference, using the covariance matrix as when computing the chi-squared loss.

Usage

```
pullCoords(df, covInv, exp, ...)
```

Arguments

df	data frame
covInv	inverse covariance matrix
exp	reference values
...	other expected values of getCoords

Value

matrix with coordinate representation of all points

Examples

```
head(pullCoords(Bikes$space2,solve(cov(Bikes$space2)),
            data.frame(value = colMeans(Bikes$space2))))
```

pullCoordsNoCov *Generic Loss Function Coordinates*

Description

Coordinates are computed as centered by the reference value and scaled with the standard deviation. Uses the i,ith entry of the covariance matrix as the standard deviation of the ith variable.

Usage

```
pullCoordsNoCov(df, cov, exp, ...)
```

Arguments

df	data frame
cov	covariance matrix
exp	reference values
...	other expected values of getCoords

Value

matrix with coordinate representation of all points

Examples

```
head(pullCoordsNoCov(Bikes$space2,cov(Bikes$space2),
                data.frame(value = colMeans(Bikes$space2))))
```

rawCoords *Raw coordinates*

Description

Returns the input data frame. This is used when other coordinate computations fail. In general, scaling of the inputs is recommended before clustering.

Usage

```
rawCoords(df, ...)
```

Arguments

df	data frame
...	other expected values of getCoords

Details

Externally calculated coordinates can be used through userCoords or as input data with rawCoords used as the coordinate function. The use of userCoords over rawCoords is in the treatment of input data. As pandemonium displays the input data in many plots the use of coordinates as input data will result in these plots being less meaningful for interpretation. Use userCoords where coordinates are necessary to calculate distances but interpretation from plots of clustering space is necessary.

Value

matrix with coordinate representation of all points

Examples

```
head(rawCoords(Bikes$space2))
```

---

tSNE *t-Distributed Stochastic Neighbor Embedding*

Description

Computes non-linear dimension reduction with Rtsne and default parameters.

Usage

```
tSNE(dist, ...)
```

Arguments

`dist`	a distance matrix
`...`	other parameters expected to be passed to dimReduction

Value

list containing a n x 2 matrix of reduced dimension data in Y

Examples

```
head(tSNE(getDists(Bikes$space1,"euclidean"))$Y)
```

umap *Uniform Manifold Approximation and Projection Embedding*

Description

Computes non-linear dimension reduction with uwot and default parameters.

Usage

```
umap(dist, ...)
```

Arguments

dist	a distance matrix
...	other parameters expected to be passed to dimReduction

Value

list containing a 2 x n matrix of reduced dimension data

Examples

```
head(umap(getDists(Bikes$space1,"euclidean"))$Y)
```

userCoords *User defined coordinate function*

Description

Allows the use of externally calculated coordinates in the app. Can only be used when variables are not reassigned between the two spaces.

Usage

```
userCoords(user_coords)
```

Arguments

user_coords	coordinate matrix the size of the space it will be used on

Details

Externally calculated coordinates can be used through userCoords or as input data with rawCoords used as the coordinate function. The use of userCoords over rawCoords is in the treatment of input data. As pandemonium displays the input data in many plots the use of coordinates as input data will result in these plots being less meaningful for interpretation. Use userCoords where coordinates are necessary to calculate distances but interpretation from plots of clustering space is necessary.

Value

function that returns the user defined coordinates user_coords

Examples

```
pandemonium(df = Bikes$space1, space2 = Bikes$space2,
             coords = list(normalised = normCoords, space2 = userCoords(Bikes$space2)))
```

---

writeResults *Write coordinates and cluster assignment to a CSV file*

Description

For working with the results outside the app. Settings used: metric, linkage, k

Usage

```
writeResults(
  space1,
  cov = NULL,
  covInv = NULL,
  exp = NULL,
  space2,
  space2.cov = NULL,
  space2.covInv = NULL,
  space2.exp = NULL,
  settings,
  filename,
  user_dist = NULL,
  getCoords.space1 = normCoords,
  getCoords.space2 = rawCoords
)
```

Arguments

space1	cluster space matrix
cov	covariance matrix
covInv	inverse covariance matrix
exp	observable reference value (e.g. experimental measurement)
space2	space2 matrix
space2.cov	covariance matrix
space2.covInv	inverse covariance matrix
space2.exp	observable reference value (e.g. experimental measurement)
settings	list specifying parameters usually selected in the app
filename	path to write the results file to
user_dist	input distance matrix (optional)
getCoords.space1	function to calculate coordinates on clustering space
getCoords.space2	function to calculate coordinates on linked space

Value

No return value, called for writing file

Examples

```
file<-tempfile()
writeResults(space1 = Bikes$space1, space2 = Bikes$space2,
settings = list(metric="euclidean",linkage="ward.D2",k=4), filename = file)
file.remove(file)
```

Index