

Package ‘pandemonium’

September 14, 2026

Title High Dimensional Analysis in Linked Spaces

Version 1.0.2

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 leave the GUI.

An earlier version of this package was described in Laa and Valencia (2022) <[doi:10.1140/epjp/s13360-021-02310-1](https://doi.org/10.1140/epjp/s13360-021-02310-1)>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.5)

Imports tourr (>= 1.2.6), stats, tibble (>= 3.3.1), ggplot2 (>= 4.0.0), RColorBrewer, dplyr, dendextend (>= 1.19.1), fpc, shiny (>= 1.13.0), shinythemes (>= 1.2.0), shinyFeedback (>= 0.4.0), DT (>= 0.34.0), tidyr (>= 1.3.2), tidyselect (>= 1.2.1), magrittr (>= 2.0.4), detourr (>= 0.2.0), crosstalk (>= 1.2.2), Rtsne (>= 0.17), plotly (>= 4.12.0), alphahull (>= 2.5), uwot (>= 0.2.4), ggpcp (>= 0.2.0), rlang (>= 1.1.7), viridis (>= 0.6.5), expm (>= 1.0)

LazyData true

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

VignetteBuilder knitr

Config/testthat/edition 3

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

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Gabriel McCoy [aut, cre] (ORCID:

<<https://orcid.org/0009-0008-3570-0361>>),

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 2026-09-14 02:50:09 UTC

Contents

Bikes	2
chi2Score	3
getBenchmarkInformation	4
getClusterDists	4
getDists	5
makePlots	6
makeResults	7
normCoords	8
outsideScore	9
pandemonium	10
pca	11
pullCoords	11
pullCoordsNoCov	12
pullCoordsSqrt	13
rawCoords	13
tSNE	14
umap	15
userCoords	15
writeResults	16
Index	18

Bikes	<i>Bike sharing data with model information</i>
-------	---

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	<i>Chi-squared scores function</i>
-----------	------------------------------------

Description

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

Usage

```
chi2Score(cluster, covinv, exp, ...)
```

Arguments

cluster	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

<code>dmat</code>	distance matrix
<code>groups</code>	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

<code>dmat</code>	distance matrix
<code>groups</code>	groups resulting from clustering
<code>benchmarks</code>	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(
  cluster,
  settings,
  cov = NULL,
  covInv = NULL,
  exp = NULL,
  linked = NULL,
  linked.cov = NULL,
  linked.covInv = NULL,
  linked.exp = NULL,
  user_dist = NULL,
  getCoordsSpace1 = normCoords,
  getCoordsSpace2 = normCoords,
  getScore = NULL,
  results = NULL
)
```

Arguments

cluster	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
linked	dataframe of variables in linked space
linked.cov	covariance matrix for space 2
linked.covInv	inverse covariance matrix for space 2
linked.exp	reference point in space 2
user_dist	user defined distances
getCoordsSpace1	function to calculate coordinates in cluster space
getCoordsSpace2	function to calculate coordinates in linked space
getScore	function to calculate scores and bins
results	an output of makeResults(), used to reduce computation when many plots are made.

Value

ggplot, plotly or detourr plot depending on settings\$plotType

Examples

```
makePlots(
  cluster = 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),
  linked = Bikes$space2, getScore = outsideScore(Bikes$other$res, "Residual")
)

makePlots(
  cluster = 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), linked = Bikes$space2,
  getScore = outsideScore(Bikes$other$res, "Residual")
)
```

makeResults

Generate Results for makePlots

Description

Settings are: metric, linkage, k. for details see the vignette on makePlots

Usage

```
makeResults(
  cluster,
  settings,
  cov = NULL,
  covInv = NULL,
  exp = NULL,
  linked = NULL,
  linked.cov = NULL,
  linked.covInv,
  linked.exp = NULL,
  user_dist = NULL,
  getCoordsSpace1 = normCoords,
  getCoordsSpace2 = normCoords,
  getScore = NULL
)
```

Arguments

cluster	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
linked	dataframe of variables in linked space
linked.cov	covariance matrix for space 2
linked.covInv	inverse covariance matrix for space 2
linked.exp	reference point in space 2
user_dist	user defined distances
getCoordsSpace1	function to calculate coordinates in cluster space
getCoordsSpace2	function to calculate coordinates in linked space
getScore	function to calculate scores and bins

Value

list of results to be passed to makePlots

Examples

```
r <- makeResults(cluster = Bikes$space1, settings = list(k = 4,
  metric = "euclidean", linkage = "ward.D2"), cov = cov(Bikes$space1),
  linked = Bikes$space2, getScore = outsideScore(Bikes$other$res, "Residual"))
makePlots(cluster = Bikes$space1, settings = list(plotType = "Obs",
  x = "hum", y = "temp", obs = "A1"), cov = cov(Bikes$space1),
  linked = Bikes$space2, getScore = outsideScore(Bikes$other$res, "Residual"),
  results = r)
```

normCoords	<i>Scaled coordinates</i>
------------	---------------------------

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	<i>Using externally computed score values</i>
--------------	---

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$space1, linked = Bikes$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 linked. 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,
  linked = NULL,
  linked.cov = NULL,
  linked.exp = NULL,
  group = NULL,
  label = NULL,
  user_dist = NULL,
  dimReduction = list(umap = umap, tSNE = tSNE, pca = pca),
  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	logical, is the covariance matrix an inverse default FALSE
exp	observable reference value (e.g. experimental measurement)
linked	data frame assumed to be in space 2 but variables can be re-assigned in the app
linked.cov	covariance matrix (optional)
linked.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

pca

Principal Component Analysis

Description

Computes and returns first two principal components of a given matrix using stats::prcomp

Usage

```
pca(mat, ...)
```

Arguments

mat	a coordinate matrix
...	other parameters expected to be passed to dimReduction

Value

list containing a 2 x n matrix of reduced dimension data

Examples

```
head(pca(Bikes$space1)$Y)
```

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 , i th entry of the covariance matrix as the standard deviation of the i th 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))
))
```

pullCoordsSqrt	<i>Chi-Squared Loss Function Coordinates</i>
----------------	--

Description

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

Usage

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

rawCoords	<i>Raw coordinates</i>
-----------	------------------------

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

<code>dist</code>	a distance matrix
<code>...</code>	other parameters expected to be passed to <code>dimReduction</code>

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

<code>user_coords</code>	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	<i>Write coordinates and cluster assignment to a CSV file</i>
--------------	---

Description

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

Usage

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

Arguments

cluster	cluster space matrix
cov	covariance matrix
covInv	inverse covariance matrix
exp	observable reference value (e.g. experimental measurement)
linked	linked space matrix
linked.cov	covariance matrix
linked.covInv	inverse covariance matrix

linked.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(
  cluster = Bikes$space1, linked = Bikes$space2,
  settings = list(metric = "euclidean", linkage = "ward.D2", k = 4), filename = file
)
file.remove(file)
```

Index

* **datasets**

Bikes, [2](#)

Bikes, [2](#)

chi2Score, [3](#)

getBenchmarkInformation, [4](#)

getClusterDists, [4](#)

getDists, [5](#)

makePlots, [6](#)

makeResults, [7](#)

normCoords, [8](#)

outsideScore, [9](#)

pandemonium, [10](#)

pca, [11](#)

pullCoords, [11](#)

pullCoordsNoCov, [12](#)

pullCoordsSqrt, [13](#)

rawCoords, [13](#)

tSNE, [14](#)

umap, [15](#)

userCoords, [15](#)

writeResults, [16](#)