Package 'ClusTorus'

May 7, 2026

Type Package

Title Prediction and Clustering on the Torus by Conformal Prediction

Description Provides various tools of for clustering multivariate angular data on the torus. The package provides angular adaptations of usual clustering methods such as the k-means clustering, pairwise angular distances, which can be used as an input for distance-based clustering algorithms, and implements clustering based on the conformal prediction framework. Options for the conformal scores include scores based on a kernel density estimate, multivariate von Mises mixtures, and naive k-means clusters. Moreover, the package provides some basic data handling tools for angular data.

Version 0.2.2

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

URL https://github.com/sungkyujung/ClusTorus

BugReports https://github.com/sungkyujung/ClusTorus/issues

Depends R (>= 3.6.0)

Imports BAMBI, igraph, purrr, ggplot2, rlang, stats, utils, cowplot

Suggests knitr, rmarkdown, tidyverse

VignetteBuilder knitr

NeedsCompilation no

Author Sungkyu Jung [aut, cph],
Seungki Hong [aut, cre],
Kiho Park [ctb],
Byungwon Kim [ctb]

Maintainer Seungki Hong <skgaboja@snu.ac.kr>

Repository CRAN

Date/Publication 2022-01-04 10:10:05 UTC

Contents

`ang.dist` *Angular distance*

Description

`ang.dist` computes element-wise angular distance between two angular values in $[0, 2\pi)$.

Usage

```
ang.dist(x, y)
```

Arguments

`x, y`	angular data(both scalar or vector) whose elements are in $[0, 2\pi)$

Value

angular data (scalar or vector) whose elements are in $[0, 2\pi)$

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

Examples

```
x <- c(pi/3, 0)
y <- c(pi/4, pi/2)

ang.dist(x, y)
```

`ang.minus` *Angular subtraction*

Description

`ang.minus` computes element-wise angular subtraction defined as

$$x - y := Arg(exp(i(x - y)))$$

Usage

```
ang.minus(x, y)
```

Arguments

`x, y`	angular data(scalar or vector) whose elements are in $[0, 2\pi)$

Value

returns a scalar or a vector whose elements are in $[-\pi, \pi)$.

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. The Annals of Applied Statistics, 15(4), 1583-1603.

Examples

```
x <- c(pi/2, 0)
y <- c(pi, pi/3)

ang.minus(x, y)
```

ang.pdist *Pairwise L2 angular distance*

Description

ang.pdist computes pairwise angular distances matrix.

Usage

```
ang.pdist(data)
```

Arguments

data	n x d angular data on $[0, 2\pi)^d$

Value

ang.pdist returns pairwise angular distances matrix with the class dist

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. The Annals of Applied Statistics, 15(4), 1583-1603.

See Also

ang.dist, dist

Examples

```
data <- matrix(c(pi/3, pi/3, pi/2,
                 pi, pi/4, pi/2,
                 0, pi/3, pi/6),
               ncol = 3, byrow = TRUE)

ang.pdist(data)
```

clus.torus *Clustering on the torus by conformal prediction*

Description

clus.torus returns clustering results of data on the torus based on inductive conformal prediction set

Usage

```
clus.torus(
  data,
  split.id = NULL,
  model = c("kmeans", "mixture"),
  mixturefitmethod = c("axis-aligned", "circular", "general"),
 kmeansfitmethod = c("general", "homogeneous-circular", "heterogeneous-circular",
    "ellipsoids"),
  J = NULL,
  level = NULL,
  option = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'clus.torus'
plot(
  x,
  panel = 1,
  assignment = "outlier",
  data = NULL,
  ellipse = TRUE,
  type = NULL,
  overlay = FALSE,
  out = FALSE,
  ...
)
```

Arguments

data	n x d matrix of toroidal data on $[0, 2\pi)^d$ or $[-\pi, \pi)^d$. Default is NULL.
split.id	a n-dimensional vector consisting of values 1 (estimation) and 2(evaluation)
model	A string. One of "mixture" and "kmeans" which determines the model or estimation methods. If "mixture", the model is based on the von Mises mixture, fitted with an EM algorithm. It supports the von Mises mixture and its variants based conformity scores. If "kmeans", the model is also based on the von Mises mixture, but the parameter estimation is implemented with the elliptical k-means

algorithm. It supports the log-max-mixture based conformity score only. If the dimension of data space is greater than 2, only "kmeans" is supported. Default is `model = "kmeans"`.

`mixturefitmethod`
: A string. One of "circular", "axis-aligned", and "general" which determines the constraint of the EM fitting. Default is "axis-aligned". This argument only works for `model = "mixture"`.

`kmeansfitmethod`
: A string. One of "general", ellipsoids", "heterogeneous-circular" or "homogeneous-circular". If "general", the elliptical k-means algorithm with no constraint is used. If "ellipsoids", only the one iteration of the algorithm is used. If"heterogeneous-circular", the same as above, but with the constraint that ellipsoids must be spheres. If "homogeneous-circular", the same as above but the radii of the spheres are identical. Default is "general". This argument only works for `model = "kmeans"`.

`J`
: the number of components for mixture model fitting. If `J` is a vector, then `hyperparam.torus` is used to choose optimal `J`. If `J == NULL`, then `J = 4:30` is used.

`level`
: a scalar in $[0, 1]$. The level of the conformal prediction set used for clustering. If `level == NULL`, then `hyperparam.alpha` is used to choose optimal `level`

`option`
: A string. One of "elbow", "risk", "AIC", or "BIC", which determines the criterion for the model selection. "risk" is based on the negative log-likelihood, "AIC" for the Akaike Information Criterion, and "BIC" for the Bayesian Information Criterion. "elbow" is based on minimizing the criterion used in Jung et. al.(2021). This argument is only used if `J` is a vector or `NULL`.

`verbose`
: boolean index, which indicates whether display additional details as to what the algorithm is doing or how many loops are done. Default is `TRUE`.

`...`
: Further arguments that will be passed to `icp.torus` and `hyperparam.torus`

`x`
: `clus.torus` object

`panel`
: One of 1 or 2 which determines the type of plot. If `panel = 1`, `x$cluster.obj` will be plotted, if `panel = 2`, `x$icp.torus` will be plotted. If `panel = 3`, `x$hyperparam.select` will be plotted. Default is `panel = 1`.

`assignment`
: A string. One of "outlier", "log.density", "posterior", "mahalanobis". Default is "outlier".

`ellipse`
: A boolean index which determines whether plotting ellipse-intersections. Default is `TRUE`. Only available for `panel = 2`.

`type`
: A string. One of "mix", "max" or "e". This argument is only available if `icp.torus` object is fitted with `model = "mixture"`. Default is `NULL`. If `type != NULL`, argument `ellipse` automatically becomes `FALSE`. If "mix", it plots based on von Mises mixture. If "max", it plots based on von Mises max-mixture. If "e", it plots based on ellipse-approximation.

`overlay`
: A boolean index which determines whether plotting ellipse-intersections on clustering plots. Default is `FALSE`. Only available for `panel = 1`.

`out`
: An option for returning the ggplot object. Default is `FALSE`.

Details

clus.torus is a user-friendly all-in-one function which implements following procedures automatically: 1. compute conformity scores for given model and fitting method, 2. choose optimal model and level based on prespecified criterion, and 3. make clusters based on the chosen model and level. Procedure 1-3 can be independently done with icp.torus, hyperparam.torus, hyperparam.J, hyperparam.alpha and cluster.assign.torus. If you want to see more detail for each procedure, please see icp.torus, hyperparam.J, hyperparam.alpha hyperparam.torus, cluster.assign.torus.

Value

clus.torus returns a clus.torus object, which consists of following 3 different S3 objects;

cluster.obj cluster.obj object; clustering assignment results for several methods. For detail, see cluster.assign.torus.

icp.torus icp.torus object; containing model parameters and conformity scores. For detail, see icp.torus.

hyperparam.select hyperparam.torus object (if J = NULL or a sequence of numbers, and level = NULL or a sequence of numbers), hyperparam.J object (if level is a scalar), or hyperparam.alpha object (if J is a scalar); contains information for the optimally chosen model (number of components J) and level (alpha) based on prespecified criterion. For detail, see hyperparam.torus, hyperparam.J, and hyperparam.alpha.

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

Mardia, K. V., Kent, J. T., Zhang, Z., Taylor, C. C., & Hamelryck, T. (2012). Mixtures of concentrated multivariate sine distributions with applications to bioinformatics. *Journal of Applied Statistics*, 39(11), 2475-2492.

Shin, J., Rinaldo, A., & Wasserman, L. (2019). Predictive clustering. *arXiv preprint arXiv:1903.08125*.

See Also

icp.torus, hyperparam.torus, hyperparam.J, hyperparam.alpha cluster.assign.torus

Examples

```
data <- toydata2[, 1:2]
n <- nrow(data)
clus.torus(data = data, model = "kmeans", kmeansfitmethod = "general", J = 5:30, option = "risk")
```

cluster.assign.torus *Clustering by connected components of ellipsoids*

Description

cluster.assign.torus returns clustering assignment for data given icp.torus objects, which can be constructed with icp.torus.

plot.clus.torus plots clustering results, which is given by cluster.obj object, with some options.

Usage

```
cluster.assign.torus(icp.object, data = NULL, level = NULL)

## S3 method for class 'cluster.obj'
plot(
  x,
  assignment = c("outlier", "log.density", "posterior", "mahalanobis"),
  overlay = FALSE,
  out = FALSE,
  ...
)
```

Arguments

icp.object	an object must be an icp.torus object, which contains all values to compute the conformity score constructed with icp.torus, or a hyperparam.torus object which is generated by hyperparam.torus.
data	n x d matrix of toroidal data on $[0, 2\pi)^d$ or $[-\pi, \pi)^d$. If data = NULL, then data within the icp.object is used.
level	a scalar in $[0, 1]$. If argument icp.object is an icp.torus object, the default value for level is 0.1. If argument icp.object is a hyperparam.torus object and level = NULL, then level is set as the optimal level hyperparam.torus$alphahat.
x	cluster.obj object
assignment	A string. One of "outlier", "log.density", "posterior", "mahalanobis". Default is "outlier".
overlay	A boolean index which determines whether plotting ellipse-intersections on clustering plots. Default is FALSE.
out	An option for returning the ggplot object. Default is FALSE.
...	additional parameter for ggplot2::ggplot()

Value

clustering assignment for data, given `icp.torus` objects

`cluster.id.by.log.density` cluster assignment result based on approximate log-density.

`cluster.id.by.posterior` cluster assignment result based on the posterior probability.

`cluster.id.outlier` cluster assignment result which regards data not included in conformal prediction set as outliers.

`cluster.id.by.Mah.dist` cluster assignment result based on Mahalanobis distance.

`level` used level which determines the size of clusters(conformal prediction set).

`data` input data which are assigned to each cluster.

`icp.torus` `icp.torus` object which is used for cluster assignment.

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

Gilitschenski, I., & Hanebeck, U. D. (2012, July). A robust computational test for overlap of two arbitrary-dimensional ellipsoids in fault-detection of kalman filters. In *2012 15th International Conference on Information Fusion* (pp. 396-401). IEEE.

See Also

`icp.torus`, `hyperparam.torus`

Examples

```
data <- toydata1[, 1:2]
icp.torus <- icp.torus(data, model = "kmeans",
                       kmeansfitmethod = "general",
                       J = 4, concentration = 25)
level <- 0.1

cluster.assign.torus(icp.torus, level = level)
```

---

cp.torus.kde *Conformal prediction set indices with kernel density estimation*

Description

`cp.torus.kde` computes conformal prediction set indices (TRUE if in the set) using kernel density estimation as conformity score.

Usage

```
cp.torus.kde(data, eval.point = grid.torus(), level = 0.1, concentration = 25)

## S3 method for class 'cp.torus.kde'
plot(x, level.id = 1, ...)
```

Arguments

`data`	n x d matrix of toroidal data on $[0, 2\pi)^d$
`eval.point`	N x N numeric matrix on $[0, 2\pi)^d$. Default input is `NULL`, which represents the fine grid points on $[0, 2\pi)^d$.
`level`	either a scalar or a vector, or even `NULL`. Default value is 0.1.
`concentration`	positive number which has the role of κ of von Mises distribution. Default value is 25.
`x`	`cp.torus.kde` object
`level.id`	an integer among `1:length(cp.torus$level)`.
`...`	additional parameter for ggplot2::ggplot()

Value

If `level` is `NULL`, then return kde at `eval.point` and at data points.

If `level` is a vector, return the above and prediction set indices for each value of level.

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

Di Marzio, M., Panzera, A., & Taylor, C. C. (2011). Kernel density estimation on the torus. *Journal of Statistical Planning and Inference*, 141(6), 2156-2173.

See Also

`kde.torus`, `grid.torus`

Examples

```
data <- ILE[1:200, 1:2]
cp.torus.kde(data, eval.point = grid.torus(),
             level = 0.05, concentration = 25)
```

`data_6VXX` *6VXX: Structure of the SARS-CoV-2 spike glycoprotein(closed state)*

Description

The torsion angle dataset of SARS-CoV-2 spike glycopreotein.

Usage

```
data_6VXX
```

Format

`data_6VXX` consists of following informations:

`phi` main chain torsion angle for atoms C,N,CA,C.

`psi` main chain torsion angle for atoms N,CA,C,N.

`omega` main chain torsion angle for atoms CA,C,N,CA.

`alpha` virtual torsion angle between consecutive C-alpha atoms.

`chi1` side chain torsion angle for atoms N,CA,CB,*G.

`chi2` side chain torsion angle for atoms CA,CB,*G,*D.

`chi3` side chain torsion angle for atoms CB,*G,*D,*E.

`chi4` side chain torsion angle for atoms *G,*D,*E,*Z.

`chi5` side chain torsion angle for atoms *D,*E,*Z, NH1.

`coords` numeric matrix of 'justified' coordinates.

`tbl` a numeric matrix of psi, phi and chi torsion angles.

Source

This data can be downloaded in https://www.rcsb.org/structure/6VXX, or with using R package `bio3d`. Precisely, we use the code: `bio3d::torsion.pdb(bio3d::read.pdb("6vxx"))`

References

Walls, A. C., Park, Y. J., Tortorici, M. A., Wall, A., McGuire, A. T., & Veesler, D. (2020). Structure, function, and antigenicity of the SARS-CoV-2 spike glycoprotein. *Cell*, 181(2), 281-292. Retrived from https://www.wwpdb.org/pdb?id=pdb_00006vxx

See Also

Description of the angluar information is from the 'value' part of `torsion.pdb` in the package `bio3d`.

`ellip.kmeans.torus` *K-Means Clustering to K-Spheres Clustering on Torus*

Description

`ellip.kmeans.torus` prepares the parameters for conformity scores which are derived by k-means clustering on torus.

Usage

```
ellip.kmeans.torus(
  data,
  centers = 10,
 type = c("homogeneous-circular", "heterogeneous-circular", "ellipsoids", "general"),
  init = c("kmeans", "hierarchical"),
  d = NULL,
  additional.condition = TRUE,
  THRESHOLD = 1e-10,
  maxiter = 200,
  verbose = TRUE,
  ...
)
```

Arguments

data	data n x d matrix of toroidal data on $[0, 2\pi)^d$
centers	either the number of clusters or a set of initial cluster centers. If a number, a random set of row in x is chosen as the initial centers.
type	character which must be "homogeneous-circular", "heterogeneous-circular", or "general". If "homogeneous-circular", the radii of k-spheres are identical. If "heterogeneous-circular", the radii of k-spheres may be different. If "ellipsoids", cluster with k-ellipsoids without optimized parameters. If, "general", clustering with k-ellipsoids. The parameters to construct the ellipses are optimized with elliptical k-means algorithm, which is modified for toroidal space. See references for the detail. Default is "homogeneous-circular".
init	determine the initial parameter for option "general". Must be "kmeans" or "hierarchical". If "kmeans", the initial parameters are obtained with extrinsic kmeans method. If "hierarchical", the initial parameters are obtained with hierarchical clustering method. Default is "hierarchical".
d	pairwise distance matrix(dist object) for `init = "hierarchical"`, which used in hierarchical clustering. If `init = "hierarchical"` and `d = NULL`, `d` will be automatically filled with `ang.pdist(data)`.
additional.condition	boolean index. If TRUE, a singular matrix will be altered to the scalar identity.
THRESHOLD	number of threshold for difference between updating and updated parameters. Default is 1e-10.
maxiter	the maximal number of iteration. Default is 200.
verbose	boolean index, which indicates whether display additional details as to what the algorithm is doing or how many loops are done. Default is TRUE.
...	Further arguments for argument `init`. If `init = "kmeans"`, these are for kmeans. If `init = "hierarchical"`, there are for hclust.

Value

returns a list, containing all values which determines the shape and location of spheres.

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

Mardia, K. V., Kent, J. T., Zhang, Z., Taylor, C. C., & Hamelryck, T. (2012). Mixtures of concentrated multivariate sine distributions with applications to bioinformatics. *Journal of Applied Statistics*, 39(11), 2475-2492.

Shin, J., Rinaldo, A., & Wasserman, L. (2019). Predictive clustering. *arXiv preprint arXiv:1903.08125*.

See Also

kmeans.torus

Examples

```
data <- ILE[1:200, 1:2]

ellip.kmeans.torus(data, centers = 3, type = "general", init = "hierarchical")
```

EMsinvMmix *Fitting mixtures of bivariate von Mises distribution*

Description

EMsinvMmix returns fitted parameters of J-mixture of bivariate sine von Mises distributions.

Usage

```
EMsinvMmix(
  data,
  J = 4,
  parammat = EMsinvMmix.init(data, J),
  THRESHOLD = 1e-10,
  maxiter = 100,
  type = c("circular", "axis-aligned", "general"),
  kmax = 500,
  verbose = TRUE
)
```

Arguments

data	n x 2 matrix of toroidal data on $[0, 2\pi)^2$
J	number of components of mixture density
parammat	6 x J parameter data with the following components: parammat[1,] : the weights for each von Mises sine density parammat[n + 1,] : κ_n for each von Mises sine density for n = 1, 2, 3 parammat[m + 4,] : μ_m for each von Mises sine density for m = 1, 2

THRESHOLD	number of threshold for difference between updating and updated parameters.
maxiter	the maximal number of iteration.
type	a string one of "circular", "axis-aligned", "general", and "Bayesian" which determines the fitting method.
kmax	the maximal number of kappa. If estimated kappa is larger than kmax, then put kappa as kmax.
verbose	boolean index, which indicates whether display additional details as to what the algorithm is doing or how many loops are done.

Details

This algorithm is based on ECME algorithm. That is, constructed with E - step and M - step and M - step maximizes the parameters with given type.

If type == "circular", then the mixture density is just a product of two independent von Mises.

If type == "axis-aligned", then the mixture density is the special case of type == "circular": only need to take care of the common concentration parameter.

Iftype == "general", then the fitting the mixture density is more complicated than before, check the detail of the reference article.

Value

returns approximated parameters for bivariate normal distribution with list:

list$Sigmainv[j] : approximated covariance matrix for j-th bivariate normal distribution, approximation of the j-th von Mises.

list$c[j] : approximated $|2\pi\Sigma|^{-1}$ for j-th bivariate normal distribution, approximation of the j-th von Mises.

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

Examples

```
data <- ILE[1:200, 1:2]

EMsinvMmix(data, J = 3,
           THRESHOLD = 1e-10, maxiter = 200,
           type = "general", kmax = 500, verbose = FALSE)
```

grid.torus *Grid on torus*

Description

grid.torus returns an equally-spaced grid on torus.

Usage

```
grid.torus(d = 2, grid.size = 100)
```

Arguments

d	number for dimension. Default is 2.
grid.size	number of grid for each axis. Default value is 100.

Value

returns (grid.size) x (grid.size) numeric matrix which indicates the grid points on torus.

Examples

```
grid.torus(d = 2, grid.size = 100)
```

hyperparam.alpha *Selecting optimal level based on the runs of the number of clusters*

Description

hyperparam.alpha evaluates the numbers of clusters for various levels, and select the optimal level based on the runs of the cluster numbers.

Usage

```
hyperparam.alpha(icp.torus, alphavec = NULL, alpha.lim = 0.15)

## S3 method for class 'hyperparam.alpha'
plot(x, ...)
```

Arguments

icp.torus	an object containing all values to compute the conformity score, which will be constructed with icp.torus.score.
alphavec	either a scalar or a vector, or even NULL for the levels. Default value is NULL. If NULL, then alphavec is automatically generated as a sequence from 0 to alpha.lim.
alpha.lim	a positive number lower than 1, which is the upper bound of Default is 0.15.
x	hyperparam.alpha object
...	additional parameter for ggplot2::ggplot()

Value

returns a hyperparam.alpha object which contains a data.frame for the numbers of clusters corresponding to the levels and the optimal level.

See Also

hyperparam.J, hyperparam.torus icp.torus

Examples

```
data <- toydata2[, 1:2]
n <- nrow(data)
split.id <- rep(2, n)
split.id[sample(n, floor(n/2))] <- 1
icp.torus <- icp.torus(data, split.id = split.id, model = "kmeans",
                       kmeansfitmethod = "ge", init = "h",
                       J = 25, verbose = TRUE)
hyperparam.alpha(icp.torus)
```

---

hyperparam.J *Selecting optimal number of mixture components based on various criteria*

Description

hyperparam.J evaluates criterion for each icp.torus objects, and select the optimal number of mixture components based on the evaluated criterion.

Usage

```
hyperparam.J(icp.torus.objects, option = c("risk", "AIC", "BIC"))

## S3 method for class 'hyperparam.J'
plot(x, ...)
```

Arguments

`icp.torus.objects`	a list whose elements are icp.torus objects, generated by `icp.torus`.
`option`	a string one of "risk", "AIC", or "BIC", which determines the criterion for the model selection. "risk" is based on the negative log-likelihood, "AIC" for the Akaike Information Criterion, and "BIC" for the Bayesian Information Criterion.
`x`	`hyperparam.J` object
`...`	additional parameter for ggplot2::ggplot()

Value

returns a `hyperparam.J` object which contains a `data.frame` for the evaluated criterion corresponding to each number of components, the optimal number of components, and the corresponding `icp.torus` object.

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6), 716-723.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, 461-464.

See Also

`icp.torus`, `hyperparam.torus`, `hyperparam.alpha`

Examples

```
data <- toydata1[,1:2]
n <- nrow(data)
split.id <- rep(2,n)
split.id[ sample(n,floor(n/2)) ] <- 1

Jvec = 4:20
icp.torus.objects <- icp.torus(data, split.id = split.id, model = "kmeans", J = Jvec)

hyperparam.J(icp.torus.objects, option = "AIC")
```

`hyperparam.torus` *Selecting optimal hyperparameters for the conformal prediction set*

Description

`hyperparam.torus` selects optimal hyperparameters for constructing the conformal prediction set, based on the type of postulated model and the criterion.

Usage

```
hyperparam.torus(
  icp.torus.objects,
  option = NULL,
  alphavec = NULL,
  alpha.lim = NULL,
  eval.point = NULL
)

## S3 method for class 'hyperparam.torus'
plot(x, color = "auto", ...)
```

Arguments

`icp.torus.objects`	list whose elements are icp.torus objects, generated by `icp.torus`
`option`	A string. One of "elbow", "risk", "AIC", or "BIC", which determines the criterion for the model selection. "risk" is based on the negative log-likelihood, "AIC" for the Akaike Information Criterion, and "BIC" for the Bayesian Information Criterion. "elbow" is based on minimizing the criterion used in Jung et. al.(2021). Default is `option = "elbow"` for 2-dimensional cases and `option = "risk"` for d(>2)-dimensional cases.
`alphavec`	either a scalar or a vector, or even `NULL` for the levels. Default value is `NULL`. If `NULL`, then `alphavec` is automatically generated as a sequence from 0 to `alpha.lim`.
`alpha.lim`	a positive number lower than 1. Default value is `NULL`. If `NULL`, then `alpha.vec` is is 0.5 for `option = "elbow"`, and 0.15 for options c("risk", "AIC", or "BIC").
`eval.point`	N x N numeric matrix on $[0, 2\pi)^2$. Default input is `grid.torus`.
`x`	`hyperparam.torus` object
`color`	A string for plotting `hyperparam.torus` object, whose criterion option is `option = "elbow"`. One of "auto", "sequential", or "qualitative". If `color = "auto"`, color assignment will be done automatically based on the number of J or concentration. If `color = "sequential"`, color assignment will be done by regarding each J or concentration as quantitative variable. If `color = "qualitative"`, color assignment will be done by regarding each J or concentration as qualitative variable. Default is `color = "auto"`.
`...`	additional parameter for ggplot2::ggplot()

Value

returns a list object which contains `data.frame` objects for the evaluated criterion corresponding to each hyperparameter, selected hyperparameters based on the designated criterion, and an `icp.torus` object based the selected hyperparameters.

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6), 716-723.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, 461-464.

Examples

```
data <- toydata2[, 1:2]
n <- nrow(data)
split.id <- rep(2, n)
split.id[sample(n, floor(n/2))] <- 1
Jvec <- 3:35
icp.torus.objects <- icp.torus(data, split.id = split.id, model = "kmeans",
                                     kmeansfitmethod = "ge", init = "h",
                                     J = Jvec, verbose = TRUE)
hyperparam.torus(icp.torus.objects, option = "risk")
```

`icp.torus` *Conformity score for inductive prediction sets*

Description

`icp.torus` prepares all values for computing the conformity score for specified methods.

`plot.icp.torus` plots `icp.torus` object with some options.

Usage

```
icp.torus(
  data,
  split.id = NULL,
  model = c("kmeans", "kde", "mixture"),
  mixturefitmethod = c("axis-aligned", "circular", "general"),
 kmeansfitmethod = c("general", "homogeneous-circular", "heterogeneous-circular",
    "ellipsoids"),
  init = c("hierarchical", "kmeans"),
  d = NULL,
  additional.condition = TRUE,
  J = 4,
```

```
  concentration = 25,
  kmax = 500,
  THRESHOLD = 1e-10,
  maxiter = 200,
  verbose = TRUE,
  ...
)

## S3 method for class 'icp.torus'
logLik(object, ...)

## S3 method for class 'icp.torus'
predict(object, newdata, ...)

## S3 method for class 'icp.torus'
plot(
  x,
  data = NULL,
  level = 0.1,
  ellipse = TRUE,
  out = FALSE,
  type = NULL,
  ...
)
```

Arguments

`data`	n x d matrix of toroidal data on $[0, 2\pi)^d$ or $[-\pi, \pi)^d$. Default is `NULL`.
`split.id`	a n-dimensional vector consisting of values 1 (estimation) and 2(evaluation)
`model`	A string. One of "kde", "mixture", and "kmeans" which determines the model or estimation methods. If "kde", the model is based on the kernel density estimates. It supports the kde-based conformity score only. If "mixture", the model is based on the von Mises mixture, fitted with an EM algorithm. It supports the von Mises mixture and its variants based conformity scores. If "kmeans", the model is also based on the von Mises mixture, but the parameter estimation is implemented with the elliptical k-means algorithm illustrated in Appendix. It supports the log-max-mixture based conformity score only. If the dimension of data space is greater than 2, only "kmeans" is supported. Default is `model = "kmeans"`.
`mixturefitmethod`	A string. One of "circular", "axis-aligned", and "general" which determines the constraint of the EM fitting. Default is "axis-aligned". This argument only works for `model = "mixture"`.
`kmeansfitmethod`	A string. One of "general", ellipsoids", "heterogeneous-circular" or "homogeneous-circular". If "general", the elliptical k-means algorithm with no constraint is used. If "ellipsoids", only the one iteration of the algorithm is used. If"heterogeneous-circular", the same as above, but with the constraint that ellipsoids must be spheres. If "homogeneous-circular", the same as above but the radii of the

spheres are identical. Default is "general". This argument only works for model = "kmeans".

init — Methods for choosing initial values of "kmeans" fitting. Must be "hierarchical" or "kmeans". If "hierarchical", the initial parameters are obtained with hierarchical clustering method. If "kmeans", the initial parameters are obtained with extrinsic k-means method. Additional arguments for k-means clustering and hierarchical clustering can be designated via argument .... If no options are designated, nstart=1 for init="kmeans" and method="complete" for init="hierarchical" are used. Default is "hierarchical".

d — pairwise distance matrix(dist object) for init = "hierarchical", which used in hierarchical clustering. If init = "hierarchical" and d = NULL, d will be automatically filled with ang.pdist(data).

additional.condition — boolean index. If TRUE, a singular matrix will be altered to the scaled identity.

J — A scalar or numeric vector for the number(s) of components for model = c("mixture", "kmeans"). Default is J = 4.

concentration — A scalar or numeric vector for the concentration parameter(s) for model = "kde". Default is concentration = 25.

kmax — the maximal number of kappa. If estimated kappa is larger than kmax, then put kappa as kmax.

THRESHOLD — number for difference between updating and updated parameters. Default is 1e-10.

maxiter — the maximal number of iteration. Default is 200.

verbose — boolean index, which indicates whether display additional details as to what the algorithm is doing or how many loops are done. Moreover, if additional.condition is TRUE, the warning message will be reported.

... — additional parameters. For plotting icp.torus, these parameters are for ggplot2::ggplot().

object — icp.torus object

newdata — n x d matrix of toroidal data on $[0, 2\pi)^d$. Dimension d must be the same as data used for icp.torus object.

x — icp.torus object

level — either a numeric scalar or a vector in $[0, 1]$. Default value is 0.1.

ellipse — A boolean index which determines whether plotting ellipses from mixture models. Default is TRUE. (This option is used only when the icp.torus object x is fitted by model kmeans or mixture.)

out — An option for returning the ggplot object. Default is FALSE.

type — A string. One of "mix", "max" or "e". This argument is only available if icp.torus object is fitted with model = "mixture". Default is NULL. If type != NULL, argument ellipse automatically becomes FALSE. If "mix", it plots based on von Mises mixture. If "max", it plots based on von Mises max-mixture. If "e", it plots based on ellipse-approximation.

Value

icp.torus returns an icp.torus object, containing all values to compute the conformity score (if J or concentration is a single value). if J or concentration is a vector containing multiple values, then icp.torus returns a list of icp.torus objects

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

Mardia, K. V., Kent, J. T., Zhang, Z., Taylor, C. C., & Hamelryck, T. (2012). Mixtures of concentrated multivariate sine distributions with applications to bioinformatics. *Journal of Applied Statistics*, 39(11), 2475-2492.

Di Marzio, M., Panzera, A., & Taylor, C. C. (2011). Kernel density estimation on the torus. *Journal of Statistical Planning and Inference*, 141(6), 2156-2173.

Shin, J., Rinaldo, A., & Wasserman, L. (2019). Predictive clustering. *arXiv preprint arXiv:1903.08125*.

Examples

```
data <- toydata1[, 1:2]

icp.torus <- icp.torus(data, model = "kmeans",
                       kmeansfitmethod = "general",
                       J = 4, concentration = 25)
```

---

icp.torus.eval *Inductive prediction sets for each level*

Description

icp.torus.eval evaluates whether each pre-specified evaluation point is contained in the inductive conformal prediction sets for each given level.

Usage

```
icp.torus.eval(icp.torus, level = 0.1, eval.point = grid.torus())
```

Arguments

icp.torus	an object containing all values to compute the conformity score, which will be constructed with icp.torus.
level	either a scalar or a vector, or even NULL. Default value is 0.1.
eval.point	N x N numeric matrix on $[0, 2\pi)^2$. Default input is grid.torus.

Value

returns a cp object with the boolean values which indicate whether each evaluation point is contained in the inductive conformal prediction sets for each given level.

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

See Also

grid.torus, icp.torus

Examples

```
data <- toydata1[, 1:2]

icp.torus <- icp.torus(data, model = "kmeans",
                       mixturefitmethod = "general",
                       J = 4, concentration = 25)

icp.torus.eval(icp.torus, level = c(0.1, 0.08), eval.point = grid.torus())
```

ILE *ILE: Structure of the Isoleucine*

Description

An isomer of leucine, essential branched-chain aliphatic amino acid found in many proteins.

Usage

```
ILE
```

Format

This list contains the following components:

`phi` main chain torsion angle for atoms C,N,CA,C.

`psi` main chain torsion angle for atoms N,CA,C,N.

`chi1` side chain torsion angle for atoms N,CA,CB,*G.

`chi2` side chain torsion angle for atoms CA,CB,*G,*D.

Details

ILE data is generated with collection of different pdb files. To select adequate protein data, we use PISCES server. (the method is introduced in articles of references.) To select high-quality protein data, we use several benchmarks: resolution : 1.6A(angstrom) or better, R-factor : 0.22 or better, Sequence percentage identity: <= 25 Then, we select ILE only angular data for each protein data. To see the detail code, visit https://github.com/sungkyujung/ClusTorus

Source

This data is extracted from PISCES server http://dunbrack.fccc.edu/pisces/

References

Data description is from https://www.rcsb.org/ligand/ILE.

The data extracting method is from Harder, T., Boomsma, W., Paluszewski, M., Frellsen, J., Johansson, K. E., & Hamelryck, T. (2010). Beyond rotamers: a generative, probabilistic model of side chains in proteins. *BMC bioinformatics*, 11(1), 1-13.

Mardia, K. V., Kent, J. T., Zhang, Z., Taylor, C. C., & Hamelryck, T. (2012). Mixtures of concentrated multivariate sine distributions with applications to bioinformatics. *Journal of Applied Statistics*, 39(11), 2475-2492.

See Also

Description of the angluar information is from the 'value' part of `torsion.pdb` in the package `bio3d`.

---

kde.torus *Kernel density estimation using circular von Mises distribution*

Description

`kde.torus` returns a kde using independent multivariate von mises kernel.

Usage

```
kde.torus(data, eval.point = NULL, concentration = 25)
```

Arguments

`data`	n x d matrix of toroidal data on $[0, 2\pi)^d$
`eval.point`	N x N numeric matrix on $[0, 2\pi)^d$. Default input is `NULL`, which represents the fine grid points on $[0, 2\pi)^d$.
`concentration`	positive number which has the role of κ of von Mises distribution. Default value is 25.

Value

kde.torus returns N-dimensional vector of kdes evaluated at eval.point

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

Di Marzio, M., Panzera, A., & Taylor, C. C. (2011). Kernel density estimation on the torus. *Journal of Statistical Planning and Inference*, 141(6), 2156-2173.

See Also

grid.torus

Examples

```
data <- ILE[1:200, 1:2]

kde.torus(data)
```

---

kmeans.torus *K-Means Clustering on Torus*

Description

kmeans.torus implements extrinsic k-means clustering on toroidal space.

Usage

```
kmeans.torus(data, centers = 10, ...)

## S3 method for class 'kmeans.torus'
predict(object, newdata, ...)
```

Arguments

data	n x d matrix of toroidal data on $[0, 2\pi)^d$
centers	either the number of clusters or a set of initial cluster centers. If a number, a random set of row in x is chosen as the initial centers.
...	additional parameter
object	kmeans.torus object
newdata	n x d matrix of toroidal data on $[0, 2\pi)^d$. Dimension d must be the same as data used for kmeans.torus object.

Details

In Euclidean space, we know that the total sum of squares is equal to the summation of the within cluster sum of squares and the between cluster centers sum of squares. However, toroidal space does not satisfy the property; the equality does not hold. Thus, you need to be careful to use the sum of squares.

Extrinsic k-means algorithm uses the ambient space for $[0, 2\pi)^d$. Each datum is transformed to a vector in 2d-dimensional Euclidean space, whose elements are sine and cosine values of the datum, then a usual k-means algorithm is applied to transformed data.

Value

returns a kmeans object, which contains

`extrinsic.results` extrinsic k-means clustering results using ordinary kmeans algorithm.

`centers` A matrix of cluster centers.

`membership` A vector of integers indicating the cluster to which each point is allocated.

`size` The number of points in each cluster.

`withinss` Vector of within-cluster sum of squares, one component per cluster.

`totss` The total sum of squares, based on angular distance.

`betweenss` The between-cluster sum of squares.

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

Gao, Y., Wang, S., Deng, M., & Xu, J. (2018). RaptorX-Angle: real-value prediction of protein backbone dihedral angles through a hybrid method of clustering and deep learning. *BMC bioinformatics*, 19(4), 73-84.

See Also

`kmeans`, `ang.minus`, `ang.dist`

Examples

```
data <- ILE[1:200, 1:2]

kmeans.torus(data, centers = 2,
             iter.max = 100, nstart = 1)
```

on.torus *Transform the angular data to be on principal interval*

Description

on.torus transforms d-dimensional angular data to be on $[0, 2\pi)^d$.

Usage

```
on.torus(x)
```

Arguments

x	d-dimensional angular data(vector or matrix) whose unit is the radian.

Value

d-dimensional radian-unit angular data on $[0, 2\pi)^d$.

Examples

```
data <- SARS_CoV_2 * pi / 180

on.torus(data)
```

SARS_CoV_2 *SARS-CoV-2: chain B of Structure of the SARS-CoV-2 spike glycoprotein(closed state)*

Description

The torsion angle dataset of the chain B of SARS-CoV-2 spike glycopreotein. This data is originally from first two main torsion angles of data_6VXX.

Usage

```
SARS_CoV_2
```

Format

This data.frame contains the following columns:

phi main chain torsion angle for atoms C,N,CA,C.

psi main chain torsion angle for atoms N,CA,C,N.

Details

This data is obtained with following codes:

Source

This data can be downloaded in https://www.rcsb.org/structure/6VXX, or with using R package bio3d. To see the precise extracting code, visit https://github.com/sungkyujung/ClusTorus/tree/master/data-raw

References

Walls, A. C., Park, Y. J., Tortorici, M. A., Wall, A., McGuire, A. T., & Veesler, D. (2020). Structure, function, and antigenicity of the SARS-CoV-2 spike glycoprotein. *Cell*, 181(2), 281-292. Retrived from https://www.wwpdb.org/pdb?id=pdb_00006vxx

See Also

Description of the angluar information is from the 'value' part of `torsion.pdb` in the package `bio3d`.

`tor.minus` *Toroidal subtraction*

Description

`tor.minus` computes angular subtraction bewtween n x d toroidal data and a d dimensional vector.

Usage

```
tor.minus(data, mu)
```

Arguments

`data`	n x d matrix of toroidal data
`mu`	a d-dimensinal vector

Value

angular subtraction bewtween n x d toroidal data and a d dimensional vector.

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

See Also

`ang.minus`

Examples

```
data <- ILE[1:200, 1:2]
Mu1 <- c(4.5, 3)
tor.minus(data, Mu1)
```

`toydata1` *toydata1: Labelled Data for 5 Clusters*

Description

Artificially generated data on the 2 dimensional torus

Usage

```
toydata1
```

Format

This `data.frame` contains the following components:

`phi` column for the first angle

`psi` column for the second angle

`label` column for the clustering membership

Details

toydata1 is an artificial data generated from a mixture of 5 clusters, where three clusters are sampled from bivariate normal distributions and the other two are each sampled from the uniform distribution on a rectangle.

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

toydata2 *toydata2: Labelled Data for 3 Clusters*

Description

Artificially generated data on the 2 dimensional torus

Usage

```
toydata2
```

Format

This `data.frame` contains the following components:

`phi` column for the first angle

`psi` column for the second angle

`label` column for the clustering membership

Details

toydata2 is an artificial data generated from a mixture of 3 clusters, where the first cluster is sampled from a spherical normal distribution, the second cluster is from the uniform distribution on a large "L"-shaped region, and the third cluster of size 50 is sampled from the uniform distribution on the entire 2-dimensional torus.

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

wtd.stat.ang *Weighted extrinsic mean direction and mean resultant length*

Description

`wtd.stat.ang` computes weighted extrinsic mean direction and mean resultant length.

Usage

```
wtd.stat.ang(data, w)
```

Arguments

`data`	angular data whose elements are in $[0, 2\pi)$
`w`	numeric vector whose each element is non-negative and `sum(w) == 1`. Moreover, the length of `w` is the same with `nrow(data)`.

Value

list which is consisting of the following components:

`Mean` weighted extrinsic mean direction

`R` mean resultant length

References

Jung, S., Park, K., & Kim, B. (2021). Clustering on the torus by conformal prediction. *The Annals of Applied Statistics*, 15(4), 1583-1603.

Examples

```
data <- matrix(c(pi/3, pi/3, pi/2,
                 pi, pi/4, pi/2,
                 0, pi/3, pi/6),
               ncol = 3, byrow = TRUE)
w <- c(0.3, 0.3, 0.4)
wtd.stat.ang(data, w)
```

Index