Package 'monoClust'

July 23, 2025

Title Perform Monothetic Clustering with Extensions to Circular Data

Version 1.2.1

Description Implementation of the Monothetic Clustering algorithm (Chavent, 1998 <doi:10.1016/S0167-8655(98)00087-7>) on continuous data sets. A lot of extensions are included in the package, including applying Monothetic clustering on data sets with circular variables, visualizations with the results, and permutation and cross-validation based tests to support the decision on the number of clusters.

License GPL (>= 2)

URL https://vinhtantran.github.io/monoClust/, https://github.com/vinhtantran/monoClust

BugReports https://github.com/vinhtantran/monoClust/issues

Depends R (>= 3.3.0)

Imports cluster (>= 2.0.5), doParallel, dplyr (>= 1.0.0), foreach, ggplot2, graphics, grDevices, parallel, permute, purrr (>= 0.3.0), rlang (>= 0.3.0), stats, stringr (>= 0.5), tibble (>= 3.0.0), tidyr (>= 1.0.0)

Suggests knitr, mice, rmarkdown, covr, testthat (>= 3.0.0)

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Config/testthat/edition 3

NeedsCompilation no

Author Tan Tran [aut, cre] (ORCID: <https://orcid.org/0000-0001-9881-6339>), Brian McGuire [aut], Mark Greenwood [aut] (ORCID: <https://orcid.org/0000-0001-6933-1201>)

Maintainer Tan Tran <vinhtantran@gmail.com>

Repository CRAN

Date/Publication 2021-02-15 15:00:02 UTC

Contents

`as_MonoClust` *Coerce Similar Object to MonoClust*

Description

The function turns a MonoClust-similar object into MonoClust object so it can use supported functions for MonoClust such as `print.MonoClust()` and `plot.MonoClust()`.

Usage

```
as_MonoClust(x, ...)

## Default S3 method:
as_MonoClust(x, ...)
```

Arguments

`x`	An object that can be coerced to MonoClust object.
`...`	For extensibility.

Details

`as_MonoClust()` is an S3 generic. The function itself doesn't run unless it is implemented for another similar object. Currently, this function is not implemented within `monoClust` package.

circ_arith *Add/Subtract Circular Values in Degrees/Radian*

Description

Add/subtract two circular variables in degrees (`%cd+%` and `%cd-%`) and radian (`%cr+%` and `%cr-%`).

Usage

```
x %cd+% y

x %cd-% y

x %cr+% y

x %cr-% y
```

Arguments

`x, y`	Circular values in degrees/radians.

Value

A value between [0, 360) in degrees or [0, 2*pi) in radian.

Examples

```
90 %cd+% 90

250 %cd+% 200

25 %cd-% 80

pi %cr+% (pi/2)
```

circ_dist *Distance Matrix of Circular Variables*

Description

Calculates the distance matrix of observations with circular variables using an adapted version of Gower's distance. This distance should be compatible with the Gower's distance for other variable types.

Usage

```
circ_dist(frame)
```

Arguments

frame	A data frame with all columns are circular measured in degrees.

Details

The distance between two observations i and j of a circular variable q is suggested to be

$$(y_{iq}, y_{jq}) = \frac{180 - |180 - |y_{iq} - y_{jq}||}{180}.$$

Value

Object of class "dist".

References

- Tran, T. V. (2019). Chapter 3. Monothetic Cluster Analysis with Extensions to Circular and Functional Data. Montana State University - Bozeman.

See Also

stats::dist()

Examples

```
# Make a sample data set of 20 observations with 2 circular variables
data <- data.frame(var1 = sample.int(359, 20),
                   var2 = sample.int(359, 20))
circ_dist(data)
```

---

cv.test *Cross-Validation Test on MonoClust*

Description

Perform cross-validation test for different different number of clusters of Monothetic Clustering.

Usage

```
cv.test(data, fold = 10L, minnodes = 2L, maxnodes = 10L, ncores = 1L, ...)
```

Arguments

`data`	Data set to be partitioned.
`fold`	Number of folds (k). `fold = 1` is the special case, when the function performs a Leave-One-Out Cross-Validation (LOOCV).
`minnodes`	Minimum number of clusters to be checked.
`maxnodes`	Maximum number of clusters to be checked.
`ncores`	Number of CPU cores on the current host. When set to NULL, all available cores are used.
`...`	Other parameters transferred to `MonoClust()`.

Details

The k-fold cross-validation randomly partitions data into k subsets with equal (or close to equal) sizes. $k-1$ subsets are used as the training data set to create a tree with a desired number of leaves and the other subset is used as validation data set to evaluate the predictive performance of the trained tree. The process repeats for each subset as the validating set ($m = 1, \ldots, k$) and the mean squared difference,

$$MSE_m = \frac{1}{n_m} \sum_{q=1}^{Q} \sum_{i \in m} d^2_{euc}(y_{iq}, \hat{y}_{(-i)q}),$$

is calculated, where $\hat{y}_{(-i)q}$ is the cluster mean on the variable q of the cluster created by the training data where the observed value, y_{iq}, of the validation data set will fall into, and $d^2_{euc}(y_{iq}, \hat{y}_{(-i)q})$ is the squared Euclidean distance (dissimilarity) between two observations at variable q. This process is repeated for the k subsets of the data set and the average of these test errors is the cross-validation-based estimate of the mean squared error of predicting a new observation,

$$CV_K = \overline{MSE} = \frac{1}{M} \sum_{m=1}^{M} MSE_m.$$

Value

A `MonoClust.cv` class containing a data frame of mean sum of square error and its standard deviation.

Note

This function supports parallel processing with `foreach::foreach()`. It distributes MonoClust calls to processes.

See Also

`plot.cv.MonoClust()`, `MonoClust()`, `predict.MonoClust()`

Examples

```
library(cluster)
data(ruspini)

# Leave-one-out cross-validation
cv.test(ruspini, fold = 1, minnodes = 2, maxnodes = 4)

# 5-fold cross-validation
cv.test(ruspini, fold = 5, minnodes = 2, maxnodes = 4)
```

---

ggcv *GGPlot the Mean Square Error with Error Bar for +/- 1 Standard Error*

Description

GGPlot the Mean Square Error with Error Bar for +/- 1 Standard Error

Usage

```
ggcv(
  cv.obj,
  title = "MSE for CV of monothetic clustering",
  xlab = "Number of clusters",
  ylab = "MSE +/- 1 SE",
  type = c("b", "p", "l"),
  linetype = 2,
  err.col = "red",
  err.width = 0.2
)
```

Arguments

cv.obj	A cv.MonoClust object (output of cv.test()).
title	Overall title for the plot.
xlab	Title for x axis.
ylab	Title for y axis.
type	What type of plot should be drawn. Choosing between "l" (line only), "p" (point only), and "b" (both line and point).
linetype	The line type. See vignette("ggplot2-specs").
err.col	Color of the error bars.
err.width	Width of the bars.

Value

A ggplot2 object.

See Also

Plot using base R `plot.cv.MonoClust()`

Examples

```
library(cluster)
data(ruspini)

# 10-fold cross-validation
cptable <- cv.test(ruspini, minnodes = 2, maxnodes = 4)
ggcv(cptable)
```

---

ggpcp *Parallel Coordinates Plot with Circular Variables*

Description

Making a parallel coordinates plot with the circular variables are plotted as ellipses. The function currently works well with data with one circular variable.

Usage

```
ggpcp(
  data,
  circ.var = NULL,
  is.degree = TRUE,
  rotate = 0,
  north = 0,
  cw = FALSE,
  order.appear = NULL,
  linetype = 1,
  size = 0.5,
  alpha = 0.5,
  clustering,
  medoids = NULL,
  cluster.col = NULL,
  show.medoids = FALSE,
  labelsize = 4,
  xlab = "Variables",
  ylab = NULL,
  legend.cluster = "groups"
)
```

Arguments

data	Data set.
circ.var	Circular variable(s) in the data set, indicated by names or index in the data set.
is.degree	Whether the unit of the circular variables is degree or not (radian). Default is TRUE.
rotate	The rotate (offset, shift) of the circular variable, in radians. Default is 0 (no rotation).
north	What value of the circular variable is labeled North. Default is 0 radian.
cw	Which direction of the circular variable is considered increasing in value, clockwise (TRUE) or counter-clockwise (FALSE). Default is TRUE.
order.appear	The order of appearance of the variables, listed by a vector of names or index. If set, length has to be equal to the number of variables in the data set.
linetype	Line type. Default is solid line. See details in vignette("ggplot2-specs").
size	Size of a line is its width in mm. Default is 0.5. See details in vignette("ggplot2-specs").
alpha	The transparency of the lines. Default is 0.1.
clustering	Cluster membership.
medoids	Vector of medoid observations of cluster. Only required when show.medoids = TRUE.
cluster.col	Color of clusters, indicating by a vector. If set, the length of this vector must be equal to the number of clusters in clustering.
show.medoids	Whether to highlight the median lines or not. Default is FALSE.
labelsize	The size of labels on the plot. Default is 4.
xlab	Labels for x-axis.
ylab	Labels for y-axis.
legend.cluster	Labels for group membership. Implemented by setting label for ggplot color aesthetics.

Value

A ggplot2 object.

Examples

```
# Set color constant
COLOR4 <- c("#e41a1c", "#377eb8", "#4daf4a", "#984ea3")
# Reduce the size of the data for for sake of example speed
set.seed(12345)
wind_reduced <- wind_sensit_2007[sample.int(nrow(wind_sensit_2007), 50), ]

sol42007 <- MonoClust(wind_reduced, cir.var = 3, nclusters = 4)

library(ggplot2)
ggpcp(data = wind_reduced,
      circ.var = "WDIR",
```

```
    # To improve aesthetics
    rotate = pi*3/4-0.3,
    order.appear = c("WDIR", "has.sensit", "WS"),
    alpha = 0.5,
    clustering = sol42007$membership,
    medoids = sol42007$medoids,
    cluster.col = COLOR4,
    show.medoids = TRUE) +
  theme(panel.background = element_rect(color = "white"),
        panel.border = element_rect(color = "white", fill = NA),
        panel.grid.major = element_line(color = "#f0f0f0"),
        panel.grid.minor = element_blank(),
        axis.line = element_line(color = "black"),
        legend.key = element_rect(color = NA),
        legend.position = "bottom",
        legend.direction = "horizontal",
        legend.title = element_text(face = "italic"),
        legend.justification = "center")
```

`inertia_calc` *Cluster Inertia Calculation*

Description

Calculate inertia for a given subset of the distance matrix from the original data set provided to `x`. Assumes that distance matrices are stored as matrices and not distance objects.

Usage

```
inertia_calc(x)
```

Arguments

`x`	Distance matrix, not an object of some distance measure.

Value

Inertia value of the matrix, formula in Chavent (1998). If `x` is a single number, return 0.

Examples

```
data(iris)

# Euclidean distance on first 20 rows of the 4 continuous variables
dist_mat <- as.matrix(dist(iris[1:20, 1:4]))
inertia_calc(dist_mat)
```

is_MonoClust *Test If The Object is A MonoClust*

Description

This function returns TRUE for MonoClust, and FALSE for all other objects.

Usage

```
is_MonoClust(mono_obj)
```

Arguments

mono_obj	An object.

Value

TRUE if the object inherits from the MonoClust class.

medoid *Find Medoid of the Cluster*

Description

Medoid is the point that has minimum distance to all other points in the cluster.

Usage

```
medoid(members, dist_mat)
```

Arguments

members	index vector indicating which observation belongs to the cluster.
dist_mat	distance matrix of the whole data set. A class of dist object must be coerced to a matrix before using.

Value

index of the medoid point in the members vector.

Examples

```
library(cluster)
data(ruspini)
ruspini4sol <- MonoClust(ruspini, nclusters = 4)
ruspini4sol

medoid(which(ruspini4sol$membership == 4), ruspini4sol$dist)

# Check with the output with "4" label
ruspini4sol$medoids
```

`MonoClust` *Monothetic Clustering*

Description

Creates a MonoClust object after partitioning the data set using Monothetic Clustering.

Usage

```
MonoClust(
  toclust,
  cir.var = NULL,
  variables = NULL,
  distmethod = NULL,
  digits = getOption("digits"),
  nclusters = 2L,
  minsplit = 5L,
  minbucket = round(minsplit/3),
  ncores = 1L
)
```

Arguments

`toclust`	Data set as a data frame.
`cir.var`	Index or name of the circular variable in the data set.
`variables`	List of variables selected for clustering procedure. It could be a vector of variable indexes, or a vector of variable names.
`distmethod`	Distance method to use with the data set. Can be chosen from "euclidean" (for Euclidean distance), "mahattan" (for Manhattan distance), or "gower" (for Gower distance). If not set, Euclidean distance is used unless `cir.var` is set, then it is Gower distance is used by default. Abbreviations can be used.
`digits`	Significant decimal number printed in the output.
`nclusters`	Number of clusters created. Default is 2.

minsplit	The minimum number of observations that must exist in a node in order for a split to be attempted. Default is 5.
minbucket	The minimum number of observations in any terminal leaf node. Default is minsplit/3.
ncores	Number of CPU cores on the current host. If greater than 1, parallel processing with foreach::foreach() is used to distribute cut search on variables to processes. When set to NULL, all available cores are used.

Value

A MonoClust object. See MonoClust.object.

References

1. Chavent, M. (1998). A monothetic clustering method. Pattern Recognition Letters, 19(11), 989-996. doi: 10.1016/S01678655(98)000877.
2. Tran, T. V. (2019). Monothetic Cluster Analysis with Extensions to Circular and Functional Data. Montana State University - Bozeman.

Examples

```
# Very simple data set
library(cluster)
data(ruspini)
ruspini4sol <- MonoClust(ruspini, nclusters = 4)
ruspini4sol

# data with circular variable
library(monoClust)
data(wind_sensit_2007)

# Use a small data set
set.seed(12345)
wind_reduced <- wind_sensit_2007[sample.int(nrow(wind_sensit_2007), 10), ]
circular_wind <- MonoClust(wind_reduced, cir.var = 3, nclusters = 2)
circular_wind
```

MonoClust.object *Monothetic Clustering Tree Object*

Description

The structure and objects contained in MonoClust, an object returned from the MonoClust() function and used as the input in other functions in the package.

Value

frame Data frame in the form of a `tibble::tibble()` representing a tree structure with one row for each node. The columns include:

- **number** Index of the node. Depth of a node can be derived by `number %/% 2`.
- **var** Name of the variable used in the split at a node or `"<leaf>"` if it is a leaf node.
- **cut** Splitting value, so values of `var` that are smaller than that go to left branch while values greater than that go to the right branch.
- **n** Cluster size, the number of observations in that cluster.
- **inertia** Inertia value of the cluster at that node.
- **bipartsplitrow** Position of the next split row in the data set (that position will belong to left node (smaller)).
- **bipartsplitcol** Position of the next split variable in the data set.
- **inertiadel** Proportion of inertia value of the cluster at that node to the inertia of the root.
- **medoid** Position of the data point regarded as the medoid of its cluster.
- **loc** y-coordinate of the splitting node to facilitate showing on the tree. See `plot.MonoClust()` for details.
- **split.order** Order of the splits with root is 0.
- **inertia_explained** Percent inertia explained as described in Chavent (2007). It is `1 - (sum(current inertia)/inert`
- **alt** A nested tibble of alternate splits at a node. It contains `bipartsplitrow` and `bipartsplitcol` with the same meaning above. Note that this is only for information purpose. Currently `monoClust` does not support choosing an alternate splitting route. Running `MonoClust()` with `nclusters = 2` step-by-step can be run if needed.

membership Vector of the same length as the number of rows in the data, containing the value of `frame$number` corresponding to the leaf node that an observation falls into.

dist Distance matrix calculated using the method indicated in `distmethod` argument of `MonoClust()`.

terms Vector of variable names in the data that were used to split.

centroids Data frame with one row for centroid value of each cluster.

medoids Named vector of positions of the data points regarded as medoids of clusters.

alt Indicator of having an alternate splitting route occurred when splitting.

circularroot List of values designed for circular variable in the data set. `var` is the name of circular variable and `cut` is its first best split value. If circular variable is not available, both objects are NULL.

References

- Chavent, M., Lechevallier, Y., & Briant, O. (2007). DIVCLUS-T: A monothetic divisive hierarchical clustering method. Computational Statistics & Data Analysis, 52(2), 687-701. doi: 10.1016/j.csda.2007.03.013.

See Also

`MonoClust()`.

perm.test *Permutation Test on Monothetic Tree*

Description

Testing the significance of each monothetic clustering split by permutation methods. The "simple-withhold" method ("sw") shuffles the observations between two groups without the splitting variable. The other two methods shuffle the values in the splitting variable to create a new data set, then it either splits again on that variable ("resplit-limit", "rl") or use all variables as the splitting candidates ("resplit-nolimit", "rn").

Usage

```
perm.test(
  object,
  data,
  auto.pick = FALSE,
  sig.val = 0.05,
  method = c("sw", "rl", "rn"),
  rep = 1000L,
  stat = c("f", "aw"),
  bon.adj = TRUE,
  ncores = 1L
)
```

Arguments

object	The MonoClust object as the result of the clustering.
data	The data set which is being clustered.
auto.pick	Whether the algorithm stops when p-value becomes larger than sig.val or keeps testing and let the researcher pick the final splitting tree. Default value is FALSE.
sig.val	Significance value to decide when to stop splitting. This option is ignored if auto.pick = FALSE, and is 0.05 by default when auto.pick = TRUE.
method	Can be chosen between sw (simple-withhold, default), rl (resplit-limit), or rn (resplit-nolimit). See Details.
rep	Number of permutations required to calculate test statistic.
stat	Statistic to use. Choosing between "f" (Calinski-Harabasz's pseudo-F (Calinski and Harabasz, 1974)) or "aw" (Average silhoutte width by Rousseeuw (1987)).
bon.adj	Whether to adjust for multiple testing problem using Bonferroni correction.
ncores	Number of CPU cores on the current host. When set to NULL, all available cores are used.

Details

Permutation Methods:

Simple-Withhold: Shuffle the observations between two proposed clusters:
The `stat` calculated from the shuffles create the reference distribution to find the p-value. Because the splitting variable that was chosen is already the best in terms of reduction of inertia, that variable is withheld from the distance matrix used in the permutation test.

Resplit-Limit: Shuffle splitting variable, split again on that variable:
This method shuffles the values of the splitting variables while keeping other variables fixed to create a new data set, then the chosen `stat` is calculated for each rep to compare with the observed `stat`.

Resplit-Nolimit: Shuffle splitting variable, split on all variables:
Similar to Method 2 but all variables are splitting candidates.

Bonferroni Correction:

A hypothesis test occurred lower in the monothetic clustering tree could have its p-value corrected for multiple tests happened before it in order to reach that node. The formula is

$$adj.p = unadj.p \times depth,$$

with $depth$ is 1 at the root node.

Value

The same `MonoClust` object with an extra column (p-value), as well as the `numofclusters` object if `auto.pick = TRUE`.

Note

This function uses `foreach::foreach()` to facilitate parallel processing. It distributes reps to processes.

References

Calinski, T. and Harabasz, J (1974). "A dendrite method for cluster analysis". en. In: *Communications in Statistics* 3.1, pp. 1-27. doi: 10.1080/03610927408827101.

Rousseeuw, P. J. (1987). "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20, pp. 53-65. ISSN: 03770427. doi: 10.1016/03770427(87)901257.

Examples

```
library(cluster)
data(ruspini)

ruspini6sol <- MonoClust(ruspini, nclusters = 6)
ruspini6.p_value <- perm.test(ruspini6sol, data = ruspini, method = "sw",
                              rep = 1000)
ruspini6.p_value
```

`plot.cv.MonoClust` *Plot the Mean Square Error with Error Bar for +/- 1 Standard Error*

Description

Plot the Mean Square Error with Error Bar for +/- 1 Standard Error

Usage

```
## S3 method for class 'cv.MonoClust'
plot(
  x,
  main = "MSE for CV of monothetic clustering",
  xlab = "Number of clusters",
  ylab = "MSE +/- 1 SE",
  type = "b",
  lty = 2,
  err.col = "red",
  err.width = 0.1,
  ...
)
```

Arguments

`x`	A `cv.MonoClust` object (output of `cv.test()`).
`main`	Overall title for the plot.
`xlab`	Title for x axis.
`ylab`	Title for y axis.
`type`	What type of plot should be drawn. See `graphics::par()`.
`lty`	The line type.
`err.col`	Color of the error bars.
`err.width`	Width of the bars.
`...`	Arguments to be passed to `graphics::plot.default()`.

Value

A line plot with error bars.

See Also

Plot using ggplot2 `ggcv()`

Examples

```
library(cluster)
data(ruspini)

# 10-fold cross-validation
cptable <- cv.test(ruspini, minnodes = 2, maxnodes = 4)
plot(cptable)
```

`plot.MonoClust` *Plot MonoClust Splitting Rule Tree*

Description

Print the MonoClust tree in the form of dendrogram.

Usage

```
## S3 method for class 'MonoClust'
plot(
  x,
  uniform = FALSE,
  branch = 1,
  margin = c(0.12, 0.02, 0, 0.05),
  minbranch = 0.3,
  text = TRUE,
  which = 4,
  stats = TRUE,
  abbrev = c("no", "short", "abbreviate"),
  digits = getOption("digits") - 2,
  cols = NULL,
  col.type = c("l", "p", "b"),
  rel.loc.x = TRUE,
  show.pval = TRUE,
  ...
)
```

Arguments

`x`	MonoClust result object.
`uniform`	If TRUE, uniform vertical spacing of the nodes is used; this may be less cluttered when fitting a large plot onto a page. The default is to use a non-uniform spacing proportional to the inertia in the fit.
`branch`	Controls the shape of the branches from parent to child node. Any number from 0 to 1 is allowed. A value of 1 gives square shouldered branches, a value of 0 give V shaped branches, with other values being intermediate.

margin	An extra fraction of white space to leave around the borders of the tree. (Long labels sometimes get cut off by the default computation).
minbranch	Set the minimum length for a branch to minbranch times the average branch length. This parameter is ignored if uniform = TRUE. Sometimes a split will give very little improvement, or even no improvement at all. A tree with branch lengths strictly proportional to improvement leaves no room to squeeze in node labels.
text	Whether to print the labels on the tree.
which	Labeling modes, which are: • 1: only splitting variable names are shown, no splitting rules. • 2: only splitting rules to the left branches are shown. • 3: only splitting rules to the right branches are shown. • 4 (default): splitting rules are shown on both sides of branches.
stats	Whether to show statistics (cluster sizes and medoid points) on the tree.
abbrev	Whether to print the abbreviated versions of variable names. Can be either "no" (default), "short", or "abbreviate". Short forms of them can also be used. If "no", the labels recorded in x$labels are used. If "short", variable names will be turned into "V1", "V2", ... If "abbreviate", abbreviate() function will be used. Use the optional arguments for this function.
digits	Number of significant digits to print.
cols	Whether to shown color bars at leaves or not. It helps matching this tree plot with other plots whose cluster membership were colored. It only works when text is TRUE. Either NULL, a vector of one color, or a vector of colors matching the number of leaves.
col.type	When cols is set, choose whether the color indicators are shown in a form of solid lines below the leaves ("l"), or big points ("p"), or both ("b").
rel.loc.x	Whether to use the relative distance between clusters as x coordinate of the leaves. Default is TRUE.
show.pval	If MonoClust object has been run through perm.test(), whether to show p-value on the tree.
...	Arguments to be passed to graphics::plot.default() and graphics::lines().

Value

A plot of splitting rule.

Examples

```
library(cluster)
data(ruspini)

# MonoClust tree
ruspini4sol <- MonoClust(ruspini, nclusters = 4)
plot(ruspini4sol)
```

```
# MonoClust tree after permutation test is run
ruspini6sol <- MonoClust(ruspini, nclusters = 6)
ruspini6_test <- perm.test(ruspini6sol,
                           data = ruspini,
                           method = "sw",
                           rep = 1000)
plot(ruspini6_test, branch = 1, uniform = TRUE)
```

`predict.MonoClust` *Predictions from a MonoClust Object*

Description

Predict the cluster memberships of a new data set from a `MonoClust` object.

Usage

```
## S3 method for class 'MonoClust'
predict(object, newdata, type = c("centroid", "medoid"), ...)
```

Arguments

`object`	MonoClust result object.
`newdata`	Data frame containing the values to be predicted. If missing, the memberships of the MonoClust object are returned.
`type`	Type of returned cluster representatives. Either `"centroid"` to return the centroid values of the terminal clusters, or `"medoid"` to return the index of the medoid observations in the clustered data set.
`...`	Further arguments passed to or from other methods.

Value

A tibble of cluster index in `cname` and either centroid values or medoid observations index based on the value of `type` argument.

Examples

```
library(cluster)
data(ruspini)

set.seed(1234)
test_index <- sample(1:nrow(ruspini), nrow(ruspini)/5)
train_index <- setdiff(1:nrow(ruspini), test_index)
ruspini_train <- ruspini[train_index, ]
ruspini_test <- ruspini[test_index, ]

ruspini_train_4sol <- MonoClust(ruspini_train, nclusters = 4)
predict(ruspini_train_4sol, newdata = ruspini_test)
```

print.cv.MonoClust *Print MonoClust Cross-Validation Result*

Description

Print MonoClust Cross-Validation Result

Usage

```
## S3 method for class 'cv.MonoClust'
print(x, ...)
```

Arguments

x	A cv.MonoClust object (output of cv.test()).
...	Further arguments passed to or from other methods.

Examples

```
library(cluster)
data(ruspini)

# 10-fold cross-validation
cp_table <- cv.test(ruspini, minnodes = 2, maxnodes = 4)
print(cp_table)
```

print.MonoClust *Print Monothetic Clustering Results*

Description

Render the MonoClust split tree in an easy to read format with important information such as terminal nodes, p-value (if possible), etc.

Usage

```
## S3 method for class 'MonoClust'
print(
  x,
  abbrev = c("no", "short", "abbreviate"),
  spaces = 2L,
  digits = getOption("digits"),
  ...
)
```

Arguments

x	MonoClust result object.
abbrev	Whether to print the abbreviated versions of variable names. Can be either "no" (default), "short", or "abbreviate". Short forms of them can also be used. If "no", the labels recorded in `x$labels` are used. If "short", variable names will be turned into "V1", "V2", ... If "abbreviate", `abbreviate()` function will be used. Use the optional arguments for this function.
spaces	Spaces indent between 2 tree levels.
digits	Number of significant digits to print.
...	Optional arguments to `abbreviate()`.

Value

A nicely displayed MonoClust split tree.

See Also

`abbreviate()`

Examples

```
library(cluster)
data(ruspini)
ruspini4sol <- MonoClust(ruspini, nclusters = 4)
print(ruspini4sol, digits = 2)
```

---

to_deg_rad *Transform Between Degree and Radian*

Description

This function transforms a circular angle from degree to radian or from radian to degree.

Usage

```
torad(x)

todeg(x)
```

Arguments

x	A degree value if `torad` or radian value if `todeg`.

Value

A radian value if `torad` or degree value if `todeg`.

Examples

```
torad(90)

torad(-45)

todeg(pi/2)
```

`wind_sensit_2007` *Existence of Microorganisms Carried in Wind*

Description

Data set is a part of a study on microorganisms carried in strong f\"ohn winds at the Bonney Riegel location of Taylor Valley, an ice free area in the Antarctic continent. Wind direction and wind speed data were obtained from the meteorological station. Wind direction was recorded every 30 seconds and wind speeds every 4 seconds at 1.15 meters above the ground surface. The recorded wind directions and speeds were averaged at 15 minute intervals. For wind direction, as discussed previously, winds from the north are defined as 0/360 degrees and from the east as 90 degrees. 2007 data were collected from August 4–11, 2007.

Usage

```
wind_sensit_2007
```

Format

A data frame with 671 rows and 3 variables:

has.sensit A binary variable of the existence of particles in the wind (1) or not (0).

WS Wind speed measured in m/s.

WDIR Wind direction in degree with 0 indicates "from the north" and 90 degrees indicate "from the east".

Source

Sabacka, M., Priscu, J. C., Basagic, H. J., Fountain, A. G., Wall, D. H., Virginia, R. A., and Greenwood, M. C. (2012). "Aeolian flux of biotic and abiotic material in Taylor Valley, Antarctica". In: Geomorphology 155-156, pp. 102-111. issn: 0169555X. doi: 10.1016/j.geomorph.2011.12.009.

`wind_sensit_2008` *Existence of Microorganisms Carried in Wind*

Description

Data set is a part of a study on microorganisms carried in strong f\"ohn winds at the Bonney Riegel location of Taylor Valley, an ice free area in the Antarctic continent. Wind direction and wind speed data were obtained from the meteorological station. Wind direction was recorded every 30 seconds and wind speeds every 4 seconds at 1.15 meters above the ground surface. The recorded wind directions and speeds were averaged at 15 minute intervals. For wind direction, as discussed previously, winds from the north are defined as 0/360 degrees and from the east as 90 degrees. 2008 data were collected from July 7–14, 2008.

Usage

```
wind_sensit_2008
```

Format

A data frame with 673 rows and 3 variables:

has.sensit A binary variable of the existence of particles in the wind (1) or not (0).

WS Wind speed measured in m/s.

WDIR Wind direction in degree with 0 indicates "from the north" and 90 degrees indicate "from the east".

Source

Sabacka, M., Priscu, J. C., Basagic, H. J., Fountain, A. G., Wall, D. H., Virginia, R. A., and Greenwood, M. C. (2012). "Aeolian flux of biotic and abiotic material in Taylor Valley, Antarctica". In: Geomorphology 155-156, pp. 102-111. issn: 0169555X. doi: 10.1016/j.geomorph.2011.12.009.

Index