Package 'ConsRankClass'

September 28, 2021

Type Package

Title Classification and Clustering of Preference Rankings

Version 1.0.1

Date 2021-09-28

Maintainer Antonio D'Ambrosio <antdambr@unina.it>

Depends ConsRank

Imports janitor, methods, pracma, rlist, proxy

Description Tree-based classification and soft-
clustering method for preference rankings, with tools for external validation of fuzzy clustering.
It contains the recursive partitioning algorithm for preference rankings, non-parametric tree-based method for a matrix of preference rankings as a response variable. It contains also the distribution-free soft clustering method for preference rankings, namely the K-median cluster component analysis (CCA).
The package depends on the 'ConsRank' R package.
Options for validate the tree-based method are both test-set procedure and V-fold cross validation.
The package contains the routines to compute the adjusted concordance index (a fuzzy version of the adjusted rand index) and the normalized degree of concordance (the corresponding fuzzy version of the rand index).
Essential references:
D'Ambrosio, A., Amodio, S., Iorio, C., Pandolfo, G., and Siciliano, R. (2021) <doi:10.1007/s00357-020-09367-0>
D'Ambrosio, A., and Heiser, W.J. (2019) <doi:10.1007/s41237-018-0069-5>;
D'Ambrosio, A., and Heiser W.J. (2016) <doi:10.1007/s11336-016-9505-1>;
Hullermeier, E., Rifqi, M., Henzgen, S., and Senge, R. (2012) <doi:10.1109/TFUZZ.2011.2179303>.

License GPL-3

Encoding UTF-8

URL https://www.r-project.org/

Repository CRAN

RoxygenNote 7.1.1

NeedsCompilation no

Author Antonio D'Ambrosio [aut, cre]

Date/Publication 2021-09-28 10:10:02 UTC

R topics documented:

`cca` *K-Median Cluster Component Analysis*

Description

K-Median Cluster Component Analysis, a distribution-free soft-clustering method for preference rankings.

Usage

```
cca(X, k, control = ccacontrol(...), ...)
```

Arguments

`X`	A n by m data matrix containing preference rankings, in which there are n judges and m objects to be judged. Each row is a ranking of the objects which are represented by the columns.
`k`	The number of cluster components
`control`	a list of options that control details of the `cca` algorithm governed by the function `ccacontrol`. The options govern maximum number of iterations of `cca` (itercca=1 is the default), the algorithm chosen to compute the median ranking (default, "quick"), and other options related to the consrank algorithm, which is called by `cca`
`...`	arguments passed bypassing `ccacontrol`

Details

The user can use any algorithm implemented in the `consrank` function from the **ConsRank** package. All algorithms allow the user to set the option 'full=TRUE' if the median ranking(s) must be searched in the restricted space of permutations instead of in the unconstrained universe of rankings of n items including all possible ties. There are two classification uncertainty measures: Us and Uprods. "Us" is the geometric mean of the membership probabilities of each individual, normalized in such a way that in the case of maximum uncertainty Us=1. "Ucca" is the average of all the "Us". "Uprods" is the product of the membership probabilities of each individual, normalized in such a way that in the case of maximum uncertainty Uprods=1. "Uprodscca" is the average of all the "Uprods".

Value

An object of the class "cca". It contains:

pk	the membership probability matrix
clc	cluster centers
oclc	cluster centers in terms of orderings
idc	crisp partition: id of the cluster component associated with the highest membership probability
Hcca	Global homogeneity measure (tau_X rank correlation coefficient)
hk	Homogeneity within cluster
props	estimated proportion of cases within cluster
Us	Uncertainty measure per-individual (see details)
Ucca	Global uncertainty measure
Uprods	Uncertainty measure per-individual (see details)
Uprodscca	Global uncertainty measure
consrankout	complete output of rank aggregation algorithm, containing eventually multiple median rankings

Author(s)

Antonio D'Ambrosio <antdambr@unina.it>

References

D'Ambrosio, A. and Heiser, W.J. (2019). A Distribution-free Soft Clustering Method for Preference Rankings. Behaviormetrika , vol. 46(2), pp. 333–351, DOI: 10.1007/s41237-018-0069-5

Heiser W.J., and D'Ambrosio A. (2013). Clustering and Prediction of Rankings within a Kemeny Distance Framework. In Berthold, L., Van den Poel, D, Ultsch, A. (eds). Algorithms from and for Nature and Life.pp-19-31. Springer international. DOI: 10.1007/978-3-319-00035-0_2.

Ben-Israel, A., and Iyigun, C. (2008). Probabilistic d-clustering. Journal of Classification, 25(1), pp.5-26. DOI: 10.1007/s00357-008-9002-z

See Also

`ccacontrol`

`ranktree`

Examples

```
data(Irish)
set.seed(135) #for reproducibility
# CCA with four components
ccares <- cca(Irish$rankings, 4, itercca=10)
summary(ccares)
```

`ccacontrol` *Utility function*

Description

Utility function to use to set the control arguments of `cca`

Usage

```
ccacontrol(
  algorithm = "quick",
  full = FALSE,
  itercca = 1,
  consrankitermax = 10,
  np = 15,
  gl = 100,
  ff = 0.4,
  cr = 0.9,
  proc = FALSE,
  ps = FALSE
)
```

Arguments

`algorithm`	The algorithm used to compute the median ranking. One among"BB", "quick" (default), "fast" and "decor"
`full`	Specifies if the median ranking must be searched in the universe of rankings including all the possible ties. Default: FALSE
`itercca`	Number of iterations of cca
`consrankitermax`	Number of iterations for "fast" and "decor" algorithms. itermax=10 is the default option.
`np`	(for "decor" only) the number of population individuals. np=15 is the default option.

gl	(for"decor" only) generations limit, maximum number of consecutive generations without improvement. gl=100 is the default option.
ff	(for"decor" only) the scaling rate for mutation. Must be in [0,1]. ff=0.4 is the default option.
cr	(for"decor" only) the crossover range. Must be in [0,1]. cr=0.9 is the default option.
proc	(for "BB" only) proc=TRUE allows the branch and bound algorithm to work in difficult cases, i.e. when the number of objects is larger than 15 or 25. proc=FALSE is the default option
ps	If PS=TRUE, on the screen some information about how many branches are processed are displayed. Default value: FALSE

Value

A list containing all the control parameters

Author(s)

Antonio D'Ambrosio <antdambr@unina.it>

See Also

cca

EVS *European Values Studies (EVS) data*

Description

Random sub-sample of 3584 cases of the survey conducted in 1999 in 32 countries analyzed by Vermunt (2003).

Usage

```
data("EVS")
```

Format

The format is: List of 3

$ data:'data.frame': 1911 obs. of 11 variables:

country, gender ,yearbird, mstatus (marital status), eduage (age of education completion), employment (Employment status: ordinal scale 1-8), householdinc (Household income: ordinal scale 1-10), A (Maintain order in Nation),Give people more say in Government decisions, (C) Fight rising prices, (D) Protect freedom of speech.

$ predictors:'data.frame' with all the predictors

$ rankings : matrix with the preferencres for "A" (Maintain order in Nation), "B" (Give people more say in Government decisions), "C" (Fight rising prices), "D" (Protect freedom of speech).

Details

Rankings were obtained by applying the post-materialism scale developed by Inglehart (1977). The scale is based upon an experiment of the type "pick 2 out of 4" most important political goals for your Governments. For this reason, replace the 'NA's with 3 before using the rankings with codes 'ranktree' or 'cca' (see D'Ambrosio and Heiser, 2016). About the predictors, the coding of the Countries are: G1 (Austria, Denmark, Netherlands, Sweden), G2 (Belgium, Croatia, France, Greece, Ireland, Northern Ireland, Spain), G3 (Bulgaria, Czechnia, East, Germany, Finland, Iceland, Luxembourg, Malta, Portugal, Romania, Slovenia, West Germany), G4 (Belarus, Estonia, Hungary, Latvia, Lithuania, Poland, Russia, Slovakia, Ukraine). Coding of predictor "mstatus" are: mar (married), wid (widowed), div (divorced), sep (separated), nevm (never married).

Source

http://statisticalinnovations.com/technicalsupport/choice_datasets.html

References

Vermunt, J. K. (2003). Multilevel latent class models. Sociological Methodology, 33(1), 213–239.

Inglehart, R. (1977). The silent revolution: Changing values and political styles among Western Publics. Princeton, NJ: Princeton University Press.

D'Ambrosio, A., and Heiser W.J. (2016). A recursive partitioning method for the prediction of preference rankings based upon Kemeny distances. Psychometrika, vol. 81 (3), pp.774-94.

Examples

```
data(EVS)

# EVS$rankings[is.na(EVS$rankings)] <- 3 #place unranked objects in a tie to the third position
# ccares <- cca(EVS$rankings,4) #solution with 4 components
```

`fuzzyconcordance` *Normalized Degree of Concordance (NDC) and Adjusted Concordance Index (ACI)*

Description

Given two fuzzy (Ruspini) partitions, it compute the NDC and the ACI. NDC is the fuzzy version of the Rand Index, as well as ACI is the fuzzy version of the Adjusted Rand Index

Usage

```
fuzzyconcordance(P, Q, nperms = 1000)
```

Arguments

P	A fuzzy partition. It has to be a matrix with n rows and k columns. Each column is expression of the degree of membership of the i-th row over the k partitions (see details).
Q	A fuzzy partition. It has to be a matrix with n rows and h columns. Each column is expression of the degree of membership of the i-th row over the h partitions (see details).
nperms	number of permutations necessary to compute ACI. Default: 1000

Details

Both P and Q, or only one of those, can be crisp (or hard) partitions. In this case, each row must contain either 0 or 1, and the sum of the i-th row must be 1. In other words, either P or Q (or both) are expressed in terms of dummy coding. If both partitions are crisp, then NDC is equal to Rand Index and ACI is equal to Adjusted Rand Index. This function can be used to externally validate the output of any fuzzy clustering method

Value

A list containing:

ACI	the Adjusted Concordance Index
NDC	the Normalized Degree of Concordance

Author(s)

Antonio D'Ambrosio <antdambr@unina.it>

References

D'Ambrosio, A., Amodio, S., Iorio, C., Pandolfo, G. and Siciliano, R. (2021). Adjusted Concordance Index: an Extension of the Adjusted Rand Index to Fuzzy Partitions. Journal of Classification vol. 38(1), pp. 112–128 (2021). DOI: 10.1007/s00357-020-09367-0

Hullermeier, E., Rifqi, M., Henzgen, S., and Senge, R. (2012). Comparing fuzzy partitions: a generalization of the Rand index and related measures. IEEE Transactions on Fuzzy Systems, 20(3), 546–556. DOI: 10.1109/TFUZZ.2011.2179303

See Also

cca

Examples

```
#two random fuzzy partitions
P = rbind(c(0.5259,  0.1656,    0.3085),
c(0.5623,    0.1036,    0.3341),
c(0.2508,    0.1849,    0.5643),
c(0.5654,    0.1934,    0.2413),
c(0.4529,    0.1679,    0.3792),
```

```
c(0.2390,    0.1758,    0.5852),
c(0.3114,    0.1743,    0.5143),
c(0.4188,    0.1392,    0.4420),
c(0.5830,    0.1655,    0.2514),
c(0.5860,    0.1171,    0.2969),
c(0.2630,    0.1706,    0.5664),
c(0.5882,    0.1032,    0.3086),
c(0.5829,    0.1277,    0.2894),
c(0.3942,    0.1046,    0.5012),
c(0.5201,    0.1097,    0.3702),
c(0.2568,    0.1823,    0.5609),
c(0.3687,    0.1695,    0.4618),
c(0.5663,    0.1317,    0.3020),
c(0.5169,    0.1950,    0.2881),
c(0.5838,    0.1034,    0.3128))

Q = rbind(c(0.4494,    0.3755,    0.1751),
c(0.5219,    0.3526,    0.1255),
c(0.3432,    0.5062,    0.1506),
c(0.3120,    0.5181,    0.1699),
c(0.5362,    0.2747,    0.1891),
c(0.4082,    0.3959,    0.1959),
c(0.4670,    0.3782,    0.1547),
c(0.4276,    0.4585,    0.1139),
c(0.4013,    0.4837,    0.1149),
c(0.3724,    0.5019,    0.1258),
c(0.5055,    0.3104,    0.1841),
c(0.4027,    0.4719,    0.1254),
c(0.3565,    0.4620,    0.1814),
c(0.6106,    0.2650,    0.1244),
c(0.5595,    0.2476,    0.1929),
c(0.4657,    0.3993,    0.1350),
c(0.2964,    0.5839,    0.1197),
c(0.5387,    0.3362,    0.1251),
c(0.4043,    0.4341,    0.1616),
c(0.5631,    0.2895,    0.1473))

ci <- fuzzyconcordance(P,Q)

#generate a random fuzzy partition with two components (clusters)
Q2 <- matrix(runif(20),ncol=1)
Q2 <- cbind(Q2,1-Q2)

ci2 <- fuzzyconcordance(P,Q2)

#generate a random crisp partition
P2 <- t(rmultinom(20,1,c(0.3,0.3,0.4)))

ci3 <- fuzzyconcordance(P2,Q)
#--------------------
## Not run:
# install.packages("Rankcluster")
library("Rankcluster") # model-based clustering algorithm for
```

```
                    #  ranking data by Biernacki and Jacques (2013)
                    #  <doi:10.1016/j.csda.2012.08.008>
data(APA)
set.seed(136) #for reproducibility
rcres <- rankclust(APA$data,K=3)  # solution with 3 centers, it takes about 75 seconds
##
ccares <- cca(APA$data,k=3) #solution with 3 components, it takes about 7 seconds
##
ci <- fuzzyconcordance(rcres[3]@tik,ccares$pk)
ci$ACI  # 0.0226 means that the two partitions are similar (see NDC below),
        # but their similarity is mainly due to chance
ci$NDC

## End(Not run)
```

`getsubtree` *Determine a tree from the main tree-based structure*

Description

Given a tree belonging to the class "ranktree", determine a subtree with a given number of terminal nodes

Usage

```
getsubtree(Tree, cut, tokeep = NULL)
```

Arguments

`Tree`	An object of the class "ranktree" coming form te function `ranktree`
`cut`	The maximum number of terminal nodes that the Tree must have
`tokeep`	parameter invoked by other internal functions

Details

If the pruning sequence returns a series of subtrees with, say, 1,2,4,7,9 terminal nodes and the user set cut=8, the function extract the subtree with 7 terminal nodes.

Value

An object of the class "ranktree", containing the same information of the output of the function `ranktree`

Author(s)

Antonio D'Ambrosio <antdambr@unina.it>

Examples

```
data("Univranks")
tree <- ranktree(Univranks$rankings,Univranks$predictors,num=50)
#see how many terminal nodes have the trees compomimg the nested sequence of subtrees
infoprun <- tree$pruneinfo$termnodes
#select the tree with, say, 6 terminal nodes
tree6 <- getsubtree(tree,6)
```

Irish *Irish Election data set*

Description

An opinion poll conducted by Irish Marketing Surveys one month prior to the election in 1997. Interviews were conducted on about 1100 respondents, drawn from 100 sampling areas. Interviews took place at randomly located homes, with respondents selected according to a socioeconomic quota. A range of sociological questions was asked of each respondent, as was their voting preference, if any, for each of the candidates.

Usage

```
data("Irish")
```

Format

The format is: List of 3

$ IrishElection: 'data.frame': 1083 obs. of 11 variables: Gender (male, housewife, nonhousewife), marital status (single, married, separated), age, socialclass (five unordered categories), Area (rural, city, town), government satisfaction (no opinion,m satisfied, dissatisfied), Bano , Roch, McAl, Nall, Scal

$ predictors :'data.frame' with all the predictors

$ rankings : matrix with the preferencres for "Bano" "Roch" "McAl" "Nall"

Details

In the original version of the data, the ranking matrix contains NAs. Here, NAs are replaced with the number 7, to indicate that all the non-stated preferences are in a tie at the last position (see D'Ambrosio and Heiser, 2016). For details about the data set see Gormley and Murphy, 2008.

Source

https://projecteuclid.org/journals/annals-of-applied-statistics/volume-2/issue-4/A-mixture-of-experts-model-for-rank-data-with/10.1214/08-AOAS178.full?tab=ArticleLinkSupplemental

References

Gormley, I.C., and Murphy, T.B. (2008). A mixture of experts model for rank data withapplications in election studies. Annals of Applied Statistics 2(4): 1452-1477. DOI: 10.1214/08-AOAS178

D'Ambrosio, A., and Heiser W.J. (2016). A recursive partitioning method for the prediction of preference rankings based upon Kemeny distances. Psychometrika, vol. 81 (3), pp.774-94. DOI: 10.1007/s11336-016-9505-1.

Examples

```
data(Irish)
```

`layouttree` *Utility function*

Description

A utility function completing the output of the function `ranktree`.

Usage

```
layouttree(Tree)
```

Arguments

`Tree`	an object of the class "ranktree"

Value

an object of the class "ranktree" completing the output of the function `ranktree`

Author(s)

Antonio D'Ambrosio <antdambr@unina.it>

nodepath *Path of a terminal node*

Description

Given an object of the class "ranktree", it visualize the path leading to the terminal node

Usage

```
nodepath(termnode, Tree)
```

Arguments

termnode	The terminal node of which the path has to be extracted
Tree	An object of the class "ranktree"

Value

The path leading to the terminal node

Author(s)

Antonio D'Ambrosio <antdambr@unina.it>

See Also

ranktree, treepaths, getsubtree

Examples

```
data(Irish)
#build the tree with default options
tree <- ranktree(Irish$rankings,Irish$predictors)
#get information about all the paths leading to terminal nodes
paths <- treepaths(tree)
#see the path for terminal node number 8
nodepath(termnode=8,tree)
```

plot.ranktree *Plot tree-based structure or pruning sequence of ranktree*

Description

Plot the tree coming from the ranktree or the pruning sequence of the ranktree

Usage

```
## S3 method for class 'ranktree'
plot(
  x,
  plot.type = "tree",
  dispclass = FALSE,
  valtree = NULL,
  taos = TRUE,
  ...
)
```

Arguments

x	An object of the class "ranktree"
plot.type	One among "tree" or "pruningseq"
dispclass	Display the median ranking above terminal nodes. Default option: FALSE
valtree	If plot.type="pruningseq", it shows the Tau_x rank correlation coefficient or the error along the pruning sequence on the training set. If valtree is the output of the function validatetree, it shows either the Tau_x rank correlation coefficient or the error along the pruning sequence of also the decision tree (validated by wither test set or cross-validation)
taos	If plot.type="pruningseq", it plots the Tau_x rank correlation coefficient along the pruning sequence. If taos=FALSE, it plots the error.
...	System reserved (No specific usage)

Value

the plot of either the tree or the pruning sequence

Author(s)

Antonio D'Ambrosio <antdambr@unina.it>

See Also

ranktree, validatetree

Examples

```
data("Univranks")
tree <- ranktree(Univranks$rankings,Univranks$predictors,num=50)
plot(tree,dispclass=TRUE)


data(EVS)
EVS$rankings[is.na(EVS$rankings)] <- 3
set.seed(654)
training=sample(1911,1434)
tree <- ranktree(EVS$rankings[training,],EVS$predictors[training,],decrmin=0.001,num=50)
plot(tree,dispclass=TRUE)
#test set validation
vtreetest <- validatetree(tree,testX=EVS$predictors[-training,],EVS$rankings[-training,])
dtree <- getsubtree(tree,vtreetest$best_tau)
plot(dtree,dispclass=TRUE)
#see the global weigthted tau_X rank correlation coefficients
plot(tree,plot.type="pruningseq",valtree=vtreetest)
#see the error rates
plot(tree,plot.type="pruningseq",valtree=vtreetest, taos=FALSE)
```

---

predict.ranktree *Predict the median rankings for new observations*

Description

Predict the median rankings in a tree-based structure built with `ranktree` for new observations

Usage

```
## S3 method for class 'ranktree'
predict(object, newx, ...)
```

Arguments

`object`	An object of the class "ranktree"
`newx`	A dataframe of the same nature of the predictor dataframe with which the tree has been built
`...`	System reserved (No specific usage)

Value

A list containing:

rankings	the fit in terms of rankings
orderings	the fit in terms of orderings
info	dataframe containing the terminal nodes in which the new x fall down, then the new x and the fit (in terms of ra

Author(s)

Antonio D'Ambrosio <antdambr@unina.it>

See Also

ranktree validatetree

Examples

```
data(EVS)
EVS$rankings[is.na(EVS$rankings)] <- 3
set.seed(654)
training=sample(1911,1434)
tree <- ranktree(EVS$rankings[training,],EVS$predictors[training,],decrmin=0.001,num=50)
#use the function predict ro predict rankings for new predictors
rankfit <- predict(tree,newx=EVS$predictors[-training,])
#fit in terms of rankings
rankfit$rankings
#fit in terms of orderings
rankfit$orderings
# information about the fit (terminal node, predictor and fit (in terms of rankings))
rankfit$info
```

---

print.cca *S3 methods for cca*

Description

Print methods for objects of class cca

Usage

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

Arguments

x	An object of the class "cca"
...	not used

Value

print a brief summary of the CCA

print.ranktree *S3 methods for ranktree*

Description

Print methods for objects of class ranktree

Usage

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

Arguments

x	An object of the class "ranktree"
...	not used

Value

print a brief summary of the prediction tree

Examples

```
data("Univranks")
tree <- ranktree(Univranks$rankings,Univranks$predictors,num=50)
tree
```

ranktree *Recursive partitioning method for the prediction of preference rankings based upon Kemeny distances*

Description

Recursive partitioning method for the prediction of preference rankings based upon Kemeny distances.

Usage

```
ranktree(Y, X, prunplot = FALSE, control = ranktreecontrol(...), ...)
```

Arguments

Y	A n by m data matrix, in which there are n judges and m objects to be judged. Each row is a ranking of the objects which are represented by the columns.
X	A dataframe containing the predictor, that must have n rows.
prunplot	prunplot=TRUE returns the plot of the pruning sequence. Default value: FALSE
control	a list of options that control details of the ranktree algorithm governed by the function ranktreecontrol. The options govern the minimum size within node to split (the default value is 0.1*n, where n is the total sample size), the bound on the decrease in impurity, (default, 0.01), the algorithm chosen to compute the median ranking (default, "quick"), and other options related to the consrank algorithm, which is called by ranktree
...	arguments passed bypassing ranktreecontrol

Details

The user can use any algorithm implemented in the consrank function from the **ConsRank** package. All algorithms allow the user to set the option 'full=TRUE' if the median ranking(s) must be searched in the restricted space of permutations instead of in the unconstrained universe of rankings of n items including all possible ties. The output consists in a object of the class "ranktree". It contains:

X		the predictors: it must be a dataframe
Y		the response variable: the matrix of the rankings
node		a list containing teh tree-based structure:
	number	node number
	terminal	logical: TRUE is terminal node
	father	father node number of the current node
	idfather	id of the father node of the current node
	size	sample size within node
	impur	impurity at node
	wimpur	weighted impurity at node
	idatnode	id of the observations within node
	class	median ranking within node in terms of orderings
	nclass	median ranking within node in terms of rankings
	mclass	eventual multiple median rankings
	tau	Tau_x rank correlation coefficient at node
	wtau	weighted Tau_x rank correlation coefficient at node
	error	error at node
	werror	weighted error at node
	varsplit	variables generating split
	varsplitid	id of variables generating split
	cutspli	splitting point
	children	children nodes generated by current node
	idchildren	id of children nodes generated by current node
	...	other info about node
control		parameters used to build the tree
numnodes		number of nodes of the tree

tsynt		list containing the synthesis of the tree:
	children	list containing all information about leaves
	parents	list containing all information about parent nodes
geneaoly		data frame containing information about all nodes
idgenealogy		data frame containing information about all nodes in terms of nodes id
idparents		id of the parents of all the nodes
goodness		goodness -and badness- of fit measures of the tree: Tau_X, error, impurity
nomin		information about nature of the predictors
alpha		alpha parameter for pruning sequence
pruneinfo		list containing information about the pruning sequence:
	prunelist	information about the pruning
	tau	tau_X rank correlation coefficient of each subtree
	error	error of each subtree
	termnodes	number of terminal nodes of each subtree
subtrees		list of each subtree created with the cost-complexity pruning procedure

Value

An object of the class ranktree. See details for detailed information.

Author(s)

Antonio D'Ambrosio <antdambr@unina.it>

References

D'Ambrosio, A., and Heiser W.J. (2016). A recursive partitioning method for the prediction of preference rankings based upon Kemeny distances. Psychometrika, vol. 81 (3), pp.774-94.

See Also

`ranktreecontrol`, `plot.ranktree`, `summary.ranktree`, `getsubtree`, `validatetree`, `treepaths`, `nodepath`

Examples

```
data("Univranks")
tree <- ranktree(Univranks$rankings,Univranks$predictors,num=50)


data(Irish)
#build the tree with default options
tree <- ranktree(Irish$rankings,Irish$predictors)

#plot the tree
plot(tree,dispclass=TRUE)

#visualize information
summary(tree)
```

```
#get information about the paths leading to terminal nodes (all the paths)
infopaths <- treepaths(tree)

#the terminal nodes
infopaths$leaves

#sample size within each terminal node
infopaths$size

#visualize the path of the second leave (terminal node number 8)
infopaths$paths[[2]]

#alternatively
nodepath(termnode=8,tree)


set.seed(132) #for reproducibility
#validation of the tree via v-fold cross-validation (default value of V=5)
vtree <- validatetree(tree,method="cv")

#extract the "best" tree
dtree <- getsubtree(tree,vtree$best_tau)

summary(dtree)

#plot the validated tree
plot(dtree,dispclass=TRUE)

#predicted rankings
rankfit <- predict(dtree,newx=Irish$predictors)

#fit of rankings
rankfit$rankings

#fit in terms of orderings
rankfit$orderings

#all info about the fit (id og the leaf, predictor values, and fit)
rankfit$orderings
```

`ranktreecontrol` *Utility function*

Description

Utility function to use to set the control arguments of `ranktree`

Usage

```
ranktreecontrol(
  num = NULL,
  decrmin = 0.01,
  algorithm = "quick",
  full = FALSE,
  itermax = 10,
  np = 15,
  gl = 100,
  ff = 0.4,
  cr = 0.9,
  proc = FALSE,
  ps = FALSE
)
```

Arguments

num	The maximum number of observations in a node to be split: default, 10% of the sample size
decrmin	Minimum decrease in impurity
algorithm	The algorithm used to compute the median ranking. One among"BB", "quick" (default), "fast" and "decor"
full	Specifies if the median ranking must be searched in the universe of rankings including all the possible ties. Default: FALSE
itermax	Number of iterations for "fast" and "decor" algorithms. itermax=10 is the default option.
np	(for "decor" only) the number of population individuals. np=15 is the default option.
gl	(for"decor" only) generations limit, maximum number of consecutive generations without improvement. gl=100 is the default option.
ff	(for"decor" only) the scaling rate for mutation. Must be in [0,1]. ff=0.4 is the default option.
cr	(for"decor" only) the crossover range. Must be in [0,1]. cr=0.9 is the default option.
proc	(for "BB" only) proc=TRUE allows the branch and bound algorithm to work in difficult cases, i.e. when the number of objects is larger than 15 or 25. proc=FALSE is the default option
ps	If PS=TRUE, on the screen some information about how many branches are processed are displayed. Default value: FALSE

Value

A list containing all the control parameters

Author(s)

Antonio D'Ambrosio <antdambr@unina.it>

See Also

ranktree

---

summary.cca *S3 methods for ranktree*

Description

Summary methods for objects of class cca

Usage

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

Arguments

object	An object of the class "cca"
...	not used

Value

it shows the summary of the prediction tree

---

summary.ranktree *S3 methods for ranktree*

Description

Summary methods for objects of class ranktree

Usage

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

Arguments

object	An object of the class "ranktree"
...	not used

Value

it shows the summary of the prediction tree

Examples

```
data("Univranks")
tree <- ranktree(Univranks$rankings,Univranks$predictors,num=50)
summary(tree)
```

`treepaths` *Path of a terminal node*

Description

Given an object of the class "ranktree", it extracts the paths of all terminal nodes

Usage

```
treepaths(Tree)
```

Arguments

`Tree`	An object of the class "ranktree"

Value

A list containing:

leaves	the number of the terminal nodes
size	the sample size within each terminal nodes
paths	a list containing all the paths

Author(s)

Antonio D'Ambrosio <antdambr@unina.it>

See Also

ranktree, nodepath, getsubtree

Examples

```
data(Irish)
#build the tree with default options
tree <- ranktree(Irish$rankings,Irish$predictors)
#get information about all the paths leading to terminal nodes
paths <- treepaths(tree)
#
#the terminal nodes
```

```
paths$leaves
#
#sample size within each terminal node
paths$size
#
#visualize the path of the second leave (terminal node number 8)
paths$paths[[2]]
```

Univranks *University rankings dataset.*

Description

University rankings dataset was analysed by Dittrich, Hatzinger and Katzenbeisser (1998) to investigate paired comparison data concerning European universities and student's characteristics with the goal to show that university rankings are different for different groups of students. Here both raw data (with paired comparisons) and the version with rankings are preesented (see details). A survey of 303 students studying at the Vienna University of Economics was carried out to examine the student's preference of six universities, namely London, Paris, Milan, St. Gallen, Barcelona and Stockholm. The data set contains 23 variables. The first 15 digits in each row indicate the preferences of a student. For a given comparison, responses were coded by 1 if the first preference was preferred, by 2 if the second university was preferred, and by 3 if universities are tied. All rows containing missing ranked Universities were skipped.

Usage

```
data("Univranks")
```

Format

The format is: List of 3

$ rawdata: 'data.frame': 212 obs. of 23 variables: the first 15 are the paired comparisons coded as follows: (1: the first is preferred to the second; 2: the second is preferred to the fisrt; 3 tied)

$ LP : comparison of London to Paris

$ LM : comparison of London to Milan

$ PM : comparison of London to Milan

$ LSg : comparison of London to St. Gallen

$ PSg : comparison of Paris to St. Gallen

$ MSg : comparison of Milan to St. Gallen

$ LB : comparison of London to Barcelona

$ PB : comparison of Paris to Barcelona

$ MB : comparison of Milan to Barcelona

$ SgB : comparison of St. Gallen to Barcelona

$ LSt : comparison of London to Stockholm

$ PSt : comparison of Paris to Stockholm

$ MSt : comparison of Milan to Stockholm

$ SgSt: comparison of St. Gallen to Stockholm

$ BSt : comparison of Barcelona to Stockholm

$ Stud: Factor w/ 2 levels "commerce","other"

$ Eng : Factor w/ 2 levels "good","poor""

$ Fra : Factor w/ 2 levels "good","poor"

$ Spa : Factor w/ 2 levels "good","poor"

$ Ita : Factor w/ 2 levels "good","poor"

$ Wor : Factor w/ 2 levels "no","yes"

$ Deg : Factor w/ 2 levels "no","yes"

$ Sex : Factor w/ 2 levels "female","male"

$ predictors:'data.frame': 212 obs. of 8 variables(the last 8 variables of the "rawdata" dataframe

$ rankings : matrix of preference rankings. The columns are: "L" (London), "P" (Paris), "M" (Milan), "Sg" (St. Gallen), "B" (Barcerlona), "St" (Stockholm)

Details

To obtain the preference rankings from the paired comparisons the procedure has been the following: the first row of the raw data is [1 3 2 1 2 1 1 2 1 1 1 2 1 1 2]. London is preferred to Paris, St. Gallen, Barcelona Stockholm (LP, LM, LSg, LB and LSt are always equal to 1), and there is no preference between London and Milan (they are tied); Milan is preferred to Paris (PM = 2), St. Gallen, Barcelona and Stockholm; and so on. The first ordering is then <L M Sg St B P> corresponding to a ranking [1,5,1,2,4,3], where the columns indicate L P M Sg B St.

Source

http://www.blackwellpublishers.co.uk/rss

References

Dittrich, R., Hatzinger, R., and Katzenbeisser, W. (1998). Modelling the effect of subject-specific covariates in paired comparison studies with an application to university rankings. Journal of the Royal Statistical Society: Series C (Applied Statistics), 47(4), 511-525. DOI: 10.1111/1467-9876.00125

D'Ambrosio, A. (2008). Tree based methods for data editing and preference rankings. Ph.D. thesis, University of Naples Federico II. https://www.doi.org/10.6092/UNINA/FEDOA/2746

Examples

```
data(Univranks)
```

validatetree *Validation of the tree for preference rankings*

Description

Validation of the tree either with a test set procedure or with v-fold cross validation

Usage

```
validatetree(
  Tree,
  testX = NULL,
  testY = NULL,
  method = "test",
  V = 5,
  plotting = TRUE
)
```

Arguments

Tree	An object of the class "ranktree" coming form te function ranktree
testX	The data frame containing the test set (predictors)
testY	The matrix ontaining the test set (response)
method	One between "test" (default) or "cv"
V	The cross-validation parameter. Default V=5
plotting	With the defaul option plotting=TRUE, the pruning sequence plot is visualized

Value

A list containing:

tau	the Tau_x rank correlation coefficient of the sequence of the trees
error	the error of the sequence of the trees
termnodes	the number of terminal nodes of the sequence of the trees
best_tau	the best tree in terms of Tau_x rank correlation coefficient
best_error	the best tree in terms of error (it is the same)
validation	information about the validation procedure

#'

Author(s)

Antonio D'Ambrosio <antdambr@unina.it>

Examples

```
data(EVS)
EVS$rankings[is.na(EVS$rankings)] <- 3
set.seed(654)
training=sample(1911,1434)
tree <- ranktree(EVS$rankings[training,],EVS$predictors[training,],decrmin=0.001,num=50)
#test set validation
vtreetest <- validatetree(tree,testX=EVS$predictors[-training,],EVS$rankings[-training,])
#cross-validation
vtreecv <- validatetree(tree,method="cv",V=10)
```

Index