Package 'C443'

May 7, 2026

Type Package

Title See a Forest for the Trees

Version 3.4.0

Imports MASS, partykit, rpart, RColorBrewer, grDevices, gridExtra, ggplot2, cluster, parallel, foreach, igraph, stats, graphics, plyr, ranger, randomForest, methods, doParallel

LazyData true

Encoding UTF-8

Date 2023-06-21

Description Get insight into a forest of classification trees, by calculating similarities between the trees, and subsequently clustering them. Each cluster is represented by it's most central cluster member. The package implements the methodology described in Sies & Van Mechelen (2020) <doi:10.1007/s00357-019-09350-4>.

URL https://github.com/KULeuven-PPW-OKPIV/C443

BugReports https://github.com/KULeuven-PPW-OKPIV/C443/issues

License GPL (>= 2)

RoxygenNote 7.2.3

NeedsCompilation no

Author Aniek Sies [aut, cre], Kristof Meers [ctb], Iven Van Mechelen [ths]

Maintainer Aniek Sies <aniek.sies@kuleuven.be>

Repository CRAN

Date/Publication 2023-06-21 09:30:02 UTC

Contents

`clusterforest` *Clustering the classification trees in a forest based on similarities*

Description

A function to get insight into a forest of classification trees by clustering the trees in a forest using Partitioning Around Medoids (PAM, Kaufman & Rousseeuw, 2009), based on user provided similarities, or based on similarities calculated by the package using a similarity measure chosen by the user (see Sies & Van Mechelen, 2020).

Usage

```
clusterforest(
  observeddata,
  treedata = NULL,
  trees,
  simmatrix = NULL,
  m = NULL,
  tol = NULL,
  weight = NULL,
  fromclus = 1,
  toclus = 1,
  treecov = NULL,
  sameobs = FALSE,
  seed = NULL,
  no_cores = detectCores(logical = FALSE)
)
```

Arguments

observeddata	The entire observed dataset
treedata	A list of dataframes on which the trees are based. Not necessary if the data set is included in the tree object already.
trees	A list of trees of class party, classes inheriting from party (e.g., glmtree), classes that can be coerced to party (i.e., rpart, Weka_tree, XMLnode), or a randomForest or ranger object.
simmatrix	A similaritymatrix with the similarities between all trees. Should be square, symmetric and have ones on the diagonal. Default=NULL
m	Similarity measure that should be used to calculate similarities, in the case that no similarity matrix was provided by the user. Default=NULL. m=1 is based on counting common predictors; m=2 is based on counting common predictor-split point combinations; m=3 is based on common ordered sets of predictor-range part combinations (see Shannon & Banks (1999)); m=4 is based on the agreement of partitions implied by leaf membership (Chipman, 1998); m=5 is based on the agreement of partitions implied by class labels (Chipman, 1998); m=6 is based on the number of predictor occurrences in definitions of leaves with same class label; m=7 is based on the number of predictor-split point combinations in definitions of leaves with same class label m=8 measures closeness to logical equivalence (applicable in case of binary predictors only)
tol	A vector with for each predictor a number that defines the tolerance zone within which two split points of the predictor in question are assumed equal. For example, if the tolerance for predictor X is 1, then a split on that predictor in tree A will be assumed equal to a split in tree B as long as the splitpoint in tree B is within the splitpoint in tree A + or - 1. Only applicable for m=1 and m=6. Default=NULL
weight	If 1, the number of dissimilar paths in the Shannon and Banks measure (m=2), should be weighted by 1/their length (Otherwise they are weighted equally). Only applicable for m=2. Default=NULL
fromclus	The lowest number of clusters for which the PAM algorithm should be run. Default=1.
toclus	The highest number of clusters for which the PAM algorithm should be run. Default=1.
treecov	A vector/dataframe with the covariate value(s) for each tree in the forest (1 column per covariate) in the case of known sources of variation underlying the forest, that should be linked to the clustering solution.
sameobs	Are the same observations included in every tree data set? For example, in the case of subsamples or bootstrap samples, the answer is no. Default=FALSE
seed	A seed number that should be used for the multi start procedure (based on which initial medoids are assigned). Default=NULL.
no_cores	Number of CPU cores used for computations. Default=detectCores(logical=FALSE)

Details

The user should provide the number of clusters that the solution should contain, or a range of numbers that should be explored. In the latter case, the resulting clusterforest object will contain

clustering results for each solution. On this clusterforest object, several methods, such as plot, print and summary, can be used.

Value

The function returns an object of class clusterforest, with attributes:

`medoids`	the position of the medoid trees in the forest (i.e., which element of the list of partytrees)
`medoidtrees`	the medoid trees
`clusters`	The cluster to which each tree in the forest is assigned
`avgsilwidth`	The average silhouette width for each solution (see Kaufman and Rousseeuw, 2009)
`accuracy`	For each solution, the accuracy of the predicted class labels based on the medoids.
`agreement`	For each solution, the agreement between the predicted class label for each observation based on the forest as a whole, and those based on the medoids only (see Sies & Van Mechelen,2020)
`withinsim`	Within cluster similarity for each solution (see Sies & Van Mechelen, 2020)
`treesimilarities`	Similarity matrix on which clustering was based
`treecov`	covariate value(s) for each tree in the forest
`seed`	seed number that was used for the multi start procedure (based on which initial medoids were assigned)

References

Kaufman, L., & Rousseeuw, P. J. (2009). Finding groups in data: an introduction to cluster analysis (Vol. 344). John Wiley & Sons.

Sies, A. & Van Mechelen I. (2020). C443: An R-package to see a forest for the trees. Journal of Classification.

Shannon, W. D., & Banks, D. (1999). Combining classification trees using MLE. Statistics in medicine, 18(6), 727-740.

Chipman, H. A., George, E. I., & McCulloh, R. E. (1998). Making sense of a forest of trees. Computing Science and Statistics, 84-92.

Examples

```
require(MASS)
require(ranger)
require(rpart)
#Function to draw a bootstrap sample from a dataset
DrawBoots <- function(dataset, i){
set.seed(2394 + i)
Boot <- dataset[sample(1:nrow(dataset), size = nrow(dataset), replace = TRUE),]
return(Boot)
}
```

```
#Function to grow a tree using rpart on a dataset
GrowTree <- function(x,y,BootsSample, minsplit = 40, minbucket = 20, maxdepth =3){
 controlrpart <- rpart.control(minsplit = minsplit, minbucket = minbucket, maxdepth = maxdepth,
   maxsurrogate = 0, maxcompete = 0)
 tree <- rpart(as.formula(paste(noquote(paste(y, "~")), noquote(paste(x, collapse="+")))),
 data = BootsSample, control = controlrpart)
 return(tree)
}

#Use functions to draw 10 boostrapsamples and grow a tree on each sample
Boots<- lapply(1:10, function(k) DrawBoots(Pima.tr ,k))
Trees <- lapply(1:10, function (i) GrowTree(x=c("npreg", "glu",  "bp",  "skin",
"bmi", "ped", "age"), y="type", Boots[[i]] ))

#Clustering the trees in this forest
ClusterForest<- clusterforest(observeddata=Pima.tr,treedata=Boots,trees=Trees,m=1,
fromclus=1, toclus=2, sameobs=FALSE, no_cores=2)

#Example RandomForest
Pima.tr.ranger <- ranger(type ~ ., data = Pima.tr, keep.inbag = TRUE, num.trees=20,
max.depth=3)

ClusterForest<- clusterforest(observeddata=Pima.tr,trees=Pima.tr.ranger,m=5,
                          fromclus=1, toclus=2, sameobs=FALSE, no_cores=2)
```

`clusters` *Get the cluster assignments for a solution of a clusterforest object*

Description

A function to get the cluster assignments for a given solution of a clusterforest object.

Usage

```
clusters(clusterforest, solution)
```

Arguments

`clusterforest`	A clusterforest object
`solution`	The solution for which cluster assignments should be returned. Default = 1

clusters.clusterforest *Get the cluster assignments for a solution of a clusterforest object*

Description

A function to get the cluster assignments for a given solution of a clusterforest object.

Usage

```
## S3 method for class 'clusterforest'
clusters(clusterforest, solution = 1)
```

Arguments

clusterforest	The clusterforest object
solution	The solution

clusters.default *Get the cluster assignments for a solution of a clusterforest object*

Description

A function to get the cluster assignments for a given solution of a clusterforest object.

Usage

```
## Default S3 method:
clusters(clusterforest, solution)
```

Arguments

clusterforest	The clusterforest object
solution	The solution

`drugs` *Drug consumption data set*

Description

A dataset collected by Fehrman et al. (2017), freely available on the UCI Machine Learning Repository (Lichman, 2013) containing records of 1885 respondents regarding their use of 18 types of drugs, and their measurements on 12 predictors. #' All predictors were originally categorical and were quantified by Fehrman et al. (2017). The meaning of the values can be found on https://archive.ics.uci.edu/dataset/373/drug+consumption+quantified. The original response categories for each drug were: never used the drug, used it over a decade ago, or in the last decade, year, month, week, or day. We transformed these into binary response categories, where 0 (non-user) consists of the categories never used the drug and used it over a decade ago and 1 (user) consists of all other categories.

Usage

```
drugs
```

Format

A data frame with 1185 rows and 32 variables:

ID Respondent ID

Age Age of respondent

Gender Gender of respondent, where 0.48 denotes female and -0.48 denotes male

Edu Level of education of participant

Country Country of current residence of participant

Ethn Ethnicity of participant

Neuro NEO-FFI-R Neuroticism score

Extr NEO-FFI-R Extraversion score

Open NEO-FFI-R Openness to experience score

Agree NEO-FFI-R Agreeableness score

Consc NEO-FFI-R Conscientiousness score

Impul Impulsiveness score measured by BIS-11

Sensat Sensation seeking score measured by ImpSS

Alc Alcohol user (1) or non-user (0)

Amphet Amphetamine user (1) or non-user (0)

Amyl Amyl nitrite user (1) or non-user (0)

Benzos Benzodiazepine user (1) or non-user (0)

Caff Caffeine user (1) or non-user (0)

Can Cannabis user (1) or non-user (0)

Choco Chocolate user (1) or non-user (0)

Coke Coke user (1) or non-user (0)

Crack Crack user (1) or non-user (0)

Ecst Ecstacy user (1) or non-user (0)

Her Heroin user (1) or non-user (0)

Ket Ketamine user (1) or non-user (0)

Leghighs Legal Highs user (1) or non-user (0)

LSD LSD user (1) or non-user (0)

Meth Methadone user (1) or non-user (0)

Mush Magical Mushroom user (1) or non-user (0)

Nico Nicotine user (1) or non-user (0)

Semeron Semeron user (1) or non-user (0), fictitious drug to identify over-claimers

VSA volatile substance abuse user(1) or non-user (0)

Source

https://archive.ics.uci.edu/dataset/373/drug+consumption+quantified

References

Fehrman, E., Muhammad, A. K., Mirkes, E. M., Egan, V., & Gorban, A. N. (2017). The Five Factor Model of personality and evaluation of drug consumption risk. In Data Science (pp. 231-242). Springer, Cham. Lichman, M. (2013). UCI machine learning repository.

`medoidtrees` *Get the medoid trees for a solution of a clusterforest object*

Description

A function to get the medoid trees for a given solution of a clusterforest object.

Usage

```
medoidtrees(clusterforest, solution)
```

Arguments

`clusterforest`	A clusterforest object
`solution`	The solution for which medoid trees should be returned. Default = 1

medoidtrees.clusterforest *Get the medoid trees for a solution of a clusterforest object*

Description

A function to get the medoid trees for a given solution of a clusterforest object.

Usage

```
## S3 method for class 'clusterforest'
medoidtrees(clusterforest, solution = 1)
```

Arguments

clusterforest	A clusterforest object
solution	The solution for which medoid trees should be returned. Default = 1

medoidtrees.default *Get the medoid trees for a solution of a clusterforest object*

Description

A function to get the medoid trees for a given solution of a clusterforest object.

Usage

```
## Default S3 method:
medoidtrees(clusterforest, solution)
```

Arguments

clusterforest	A clusterforest object
solution	The solution for which medoid trees should be returned. Default = 1

`plot.clusterforest` *Plot a clusterforest object*

Description

A function that can be used to plot a clusterforest object, either by returning plots with information such as average silhouette width and within cluster siiliarity on the cluster solutions, or plots of the medoid trees of each solution.

Usage

```
## S3 method for class 'clusterforest'
plot(x, solution = NULL, predictive_plots = FALSE, ...)
```

Arguments

`x`	A clusterforest object
`solution`	The solution to plot the medoid trees from. If NULL, plots with the average silhouette width, within cluster similiarty (and predictive accuracy) per solution are returned. Default = NULL
`predictive_plots`	Indicating whether predictive plots should be returned: A plot showing the predictive accuracy when making predictions based on the medoid trees, and a plot of the agreement between the class label for each object predicted on the basis of the random forest as a whole versus based on the medoid trees. Default = FALSE.
`...`	Additional arguments that can be used in generic plot function, or in plot.party.

Details

This function can be used to plot a clusterforest object in two ways. If it's used without specifying a solution, then the average silhouette width, and within cluster similarity measures are plotted for each solution. If additionally, predictive_plots=TRUE, two more plots are returned, namely a plot showing for each solution the predictive accuracy when making predictions based on the medoid trees, and a plot showing for each solution the agreement between the class label for each object predicted on the basis of the random forest as a whole versus based on the medoid trees. These plots may be helpful in deciding how many clusters are needed to summarize the forest (see Sies & Van Mechelen, 2020).

If the function is used with the clusterforest object and the number of the solution, then the medoid tree(s) of that solution are plotted.

References

Sies, A. & Van Mechelen I. (2020). C443: An R-package to see a forest for the trees. Journal of Classification.

Examples

```
require(MASS)
require(rpart)
#Function to draw a bootstrap sample from a dataset
DrawBoots <- function(dataset, i){
set.seed(2394 + i)
Boot <- dataset[sample(1:nrow(dataset), size = nrow(dataset), replace = TRUE),]
return(Boot)
}

#Function to grow a tree using rpart on a dataset
GrowTree <- function(x,y,BootsSample, minsplit = 40, minbucket = 20, maxdepth =3){
 controlrpart <- rpart.control(minsplit = minsplit, minbucket = minbucket,
 maxdepth = maxdepth, maxsurrogate = 0, maxcompete = 0)
 tree <- rpart(as.formula(paste(noquote(paste(y, "~")),
 noquote(paste(x, collapse="+")))), data = BootsSample,
 control = controlrpart)
 return(tree)
}

#Use functions to draw 20 boostrapsamples and grow a tree on each sample
Boots<- lapply(1:10, function(k) DrawBoots(Pima.tr ,k))
Trees <- lapply(1:10, function (i) GrowTree(x=c("npreg", "glu",  "bp",
 "skin",  "bmi", "ped", "age"), y="type",
Boots[[i]] ))

ClusterForest<- clusterforest(observeddata=Pima.tr,treedata=Boots,trees=Trees,m=1,
fromclus=1, toclus=5, sameobs=FALSE, no_cores=2)
plot(ClusterForest)
plot(ClusterForest,2)
```

`print.clusterforest` *Print a clusterforest object*

Description

A function that can be used to print a clusterforest object.

Usage

```
## S3 method for class 'clusterforest'
print(x, solution = 1, ...)
```

Arguments

`x`	A clusterforest object
`solution`	The solution to print the medoid trees from. Default = NULL
`...`	Additional arguments that can be used in the generic print function.

summary.clusterforest *Summarize a clusterforest object*

Description

A function to summarize a clusterforest object.

Usage

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

Arguments

object	A clusterforest object
...	Additional arguments that can be used in the generic summary function.

treesimilarities *Get the similarity matrix that wast used to create a clusterforest object*

Description

A function to get the similarity matrix used to obtain a clusterforest object.

Usage

```
treesimilarities(clusterforest)
```

Arguments

clusterforest	A clusterforest object

`treesimilarities.clusterforest` *Get the similarity matrix that wast used to create a clusterforest object*

Description

A function to get the similarity matrix used to obtain a clusterforest object.

Usage

```
## S3 method for class 'clusterforest'
treesimilarities(clusterforest)
```

Arguments

`clusterforest`	A clusterforest object

`treesimilarities.default` *Get the similarity matrix that wast used to create a clusterforest object*

Description

A function to get the similarity matrix used to obtain a clusterforest object.

Usage

```
## Default S3 method:
treesimilarities(clusterforest)
```

Arguments

`clusterforest`	A clusterforest object

`treesource` *Mapping the tree clustering solution to a known source of variation underlying the forest*

Description

A function that can be used to get insight into a clusterforest solution, in the case that there are known sources of variation underlying the forest. These known sources of variation must be included in the clusterforest object (and thus must be defined when running the clusterforest function) In case of a categorical covariate, it visualizes the number of trees from each value of the covariate that belong to each cluster. In case of a continuous covariate, it returns the mean and standard deviation of the covariate in each cluster.

Usage

```
treesource(clusterforest, solution)
```

Arguments

`clusterforest`	The clusterforest object, indluding the treecov attribute.
`solution`	The solution

Value

`multiplot`	In case of categorical covariate, for each value of the covariate, a bar plot with the number of trees that belong to each cluster
`heatmap`	In case of a categorical covariate, a heatmap with for each value of the covariate, the number of trees that belong to each cluster
`clustermeans`	In case of a continuous covariate, the mean of the covariate in each cluster
`clusterstds`	In case of a continuous covariate, the standard deviation of the covariate in each cluster

Examples

```
require(rpart)
data_Amphet <-drugs[,c ("Amphet","Age", "Gender", "Edu", "Neuro", "Extr", "Open", "Agree",
"Consc", "Impul","Sensat")]
data_cocaine <-drugs[,c ("Coke","Age", "Gender", "Edu", "Neuro", "Extr", "Open", "Agree",
                         "Consc", "Impul","Sensat")]


#Function to draw a bootstrap sample from a dataset
DrawBoots <- function(dataset, i){
set.seed(2394 + i)
Boot <- dataset[sample(1:nrow(dataset), size = nrow(dataset), replace = TRUE),]
return(Boot)
}
```

```
#Function to grow a tree using rpart on a dataset
GrowTree <- function(x,y,BootsSample, minsplit = 40, minbucket = 20, maxdepth =3){

 controlrpart <- rpart.control(minsplit = minsplit, minbucket = minbucket, maxdepth = maxdepth,
 maxsurrogate = 0, maxcompete = 0)
 tree <- rpart(as.formula(paste(noquote(paste(y, "~")), noquote(paste(x, collapse="+")))),
  data = BootsSample, control = controlrpart)
 return(tree)
}

#Draw bootstrap samples and grow trees
BootsA<- lapply(1:5, function(k) DrawBoots(data_Amphet,k))
BootsC<- lapply(1:5, function(k) DrawBoots(data_cocaine,k))
Boots = c(BootsA,BootsC)

TreesA <- lapply(1:5, function (i) GrowTree(x=c ("Age", "Gender", "Edu", "Neuro",
"Extr", "Open", "Agree","Consc", "Impul","Sensat"), y="Amphet", BootsA[[i]] ))
TreesC <- lapply(1:5, function (i) GrowTree(x=c ( "Age", "Gender", "Edu", "Neuro",
"Extr", "Open", "Agree", "Consc", "Impul","Sensat"), y="Coke", BootsC[[i]] ))
Trees=c(TreesA,TreesC)

#Cluster the trees
ClusterForest<- clusterforest(observeddata=drugs,treedata=Boots,trees=Trees,m=1,
fromclus=2, toclus=2, treecov=rep(c("Amphet","Coke"),each=5), sameobs=FALSE, no_cores=2)

#Link cluster result to known source of variation
treesource(ClusterForest, 2)
```

`treesource.clusterforest` *Mapping the tree clustering solution to a known source of variation underlying the forest*

Description

A function that can be used to get insight into a clusterforest solution, in the case that there is a known source of variation underlying the forest. It visualizes the number of trees from each source that belong to each cluster.

Usage

```
## S3 method for class 'clusterforest'
treesource(clusterforest, solution)
```

Arguments

`clusterforest`	The clusterforest object
`solution`	The solution

`treesource.default` *Mapping the tree clustering solution to a known source of variation underlying the forest*

Description

A function that can be used to get insight into a clusterforest solution, in the case that there is a known source of variation underlying the forest. It visualizes the number of trees from each source that belong to each cluster.

Usage

```
## Default S3 method:
treesource(clusterforest, solution)
```

Arguments

`clusterforest`	The clusterforest object
`solution`	The solution

Index