Package ‘C443’

April 25, 2020

Type Package

Title See a Forest for the Trees

Version 2.0.0

Imports MASS, partykit, rpart, RColorBrewer, grDevices, gridExtra, ggplot2, cluster, parallel, igraph, reshape2, qgraph, stats, graphics, plyr, plot.matrix

LazyData true

Encoding UTF-8

Date 2020-04-18


License GPL (>= 2)

RoxygenNote 7.1.0

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2020-04-25 15:30:02 UTC

R topics documented:

 as.treesimilarities ......................................................... 2
 as.treesimilarities.list .................................................. 3
 clusterforest .............................................................. 4
 drugs .............................................................................. 5
 forest ............................................................................. 7
 plot.clusterforest ......................................................... 8
as.treesimilarities

Function to coerce objects to a treesimilarities object.

Description

A function to coerce objects to treesimilarities objects. As an argument, it takes a list with a similarity matrix as its first element and a forest object as its second. This list is then coerced to a treesimilarities object (with attributes similarity matrix and forest).

Usage

as.treesimilarities(obj, ...)

Arguments

obj List with as first element the similarity matrix, and as second the forest on which it was calculated

... Additional arguments

Value

The function returns an object of class treesimilarities. It includes two attributes: forest (The forest for which similarities were calculated) and sim (similarity matrix with similarities between trees in forest based on chosen similarity measure)

Examples

require(rpart)
require(MASS)

#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),]
}
as.treesimilarities.list

A function to coerce objects to a treesimilarities object.

Description

A function to coerce objects to treesimilarities objects. It takes a list with a similarity matrix as its first element and a forest object as its second, as arguments. This list is then coerced to a treesimilarities object (with attributes similarity matrix and forest).

Usage

## S3 method for class 'list'
as.treesimilarities(obj, ...)

Arguments

obj List with as first element the similarity matrix, and as second the forest on which it was calculated

... Additional arguments
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 similarities (see Sies & Van Mechelen, 2020).

Usage

clusterforest(similarities, fromclus, toclus)

Arguments

- **similarities**: A treesimilarities object containing a similarity matrix and the forest object on which the similarities were calculated
- **fromclus**: The lowest number of clusters for which the PAM algorithm should be run
- **toclus**: The highest number of clusters for which the PAM algorithm should be run

Details

This function uses the Partitioning Around Medoids (PAM) algorithm, to cluster trees in a forest, based on similarities between the trees. The function takes a treesimilarities object as input (see treesimilarities() or as.treesimilarities() to create a treesimilarities object) and returns a clusterforest object.

The PAM algorithm can be run for solutions with varying number of clusters, depending on the fromclus and toclus arguments. Results will be returned for each solution.

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)
- **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)
References


Examples

```r
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 15 bootstrap samples and grow a tree on each sample
Boots<- lapply(1:15, function(k) DrawBoots(Pima.tr ,k))
Trees <- lapply(1:15, function (i) GrowTree(x=c("npreg", "glu", "bp", "skin", 
                                             "bmi", "ped", "age"), y="type", Boots[ [i]] ))
#Turn the lists of dataframes and rpart trees to a forest object
myforest<- forest(Pima.tr,Boots,Trees)
#Calculate similarities between trees in forest, based on similarity measure m=1.
Simmatrix1 <- treesimilarities(forest=myforest, m=1)
#Cluster forest
ClusterForest <- clusterforest(Simmatrix1, 1, 5)
```

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](https://).
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

Format

A data frame with 1185 rows and 32 variables:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID</strong></td>
<td>Respondent ID</td>
</tr>
<tr>
<td><strong>Age</strong></td>
<td>Age of respondent</td>
</tr>
<tr>
<td><strong>Gender</strong></td>
<td>Gender of respondent, where 0.48 denotes female and -0.48 denotes male</td>
</tr>
<tr>
<td><strong>Edu</strong></td>
<td>Level of education of participant</td>
</tr>
<tr>
<td><strong>Country</strong></td>
<td>Country of current residence of participant</td>
</tr>
<tr>
<td><strong>Ethn</strong></td>
<td>Ethnicity of participant</td>
</tr>
<tr>
<td><strong>Neuro</strong></td>
<td>NEO-FFI-R Neuroticism score</td>
</tr>
<tr>
<td><strong>Extr</strong></td>
<td>NEO-FFI-R Extraversion score</td>
</tr>
<tr>
<td><strong>Open</strong></td>
<td>NEO-FFI-R Openness to experience score</td>
</tr>
<tr>
<td><strong>Agree</strong></td>
<td>NEO-FFI-R Agreeableness score</td>
</tr>
<tr>
<td><strong>Consc</strong></td>
<td>NEO-FFI-R Conscientiousness score</td>
</tr>
<tr>
<td><strong>Impul</strong></td>
<td>Impulsiveness score measured by BIS-11</td>
</tr>
<tr>
<td><strong>Sensat</strong></td>
<td>Sensation seeking score measured by ImpSS</td>
</tr>
<tr>
<td><strong>Alc</strong></td>
<td>Alcohol user (1) or non-user (0)</td>
</tr>
<tr>
<td><strong>Amphet</strong></td>
<td>Amphetamine user (1) or non-user (0)</td>
</tr>
<tr>
<td><strong>Amyl</strong></td>
<td>Amyl nitrite user (1) or non-user (0)</td>
</tr>
<tr>
<td><strong>Benzos</strong></td>
<td>Benzodiazepine user (1) or non-user (0)</td>
</tr>
<tr>
<td><strong>Caff</strong></td>
<td>Caffeine user (1) or non-user (0)</td>
</tr>
<tr>
<td><strong>Can</strong></td>
<td>Cannabis user (1) or non-user (0)</td>
</tr>
<tr>
<td><strong>Choco</strong></td>
<td>Chocolate user (1) or non-user (0)</td>
</tr>
<tr>
<td><strong>Coke</strong></td>
<td>Coke user (1) or non-user (0)</td>
</tr>
<tr>
<td><strong>Crack</strong></td>
<td>Crack user (1) or non-user (0)</td>
</tr>
<tr>
<td><strong>Ecst</strong></td>
<td>Ecstasy user (1) or non-user (0)</td>
</tr>
<tr>
<td><strong>Her</strong></td>
<td>Heroin user (1) or non-user (0)</td>
</tr>
<tr>
<td><strong>Ket</strong></td>
<td>Ketamine user (1) or non-user (0)</td>
</tr>
<tr>
<td><strong>Leghighs</strong></td>
<td>Legal Highs user (1) or non-user (0)</td>
</tr>
<tr>
<td><strong>LSD</strong></td>
<td>LSD user (1) or non-user (0)</td>
</tr>
</tbody>
</table>
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


References


forest  Constructor of the forest class

Description

A function that coerces a list of party trees, trees that can be coerced to party trees, or classes inheriting from party (see Hothorn & Zeileis, 2015), a list of dataframes on which these trees were grown, and the original data set underlying the forest, into an object of class forest.

Usage

forest(fulldata, treedata, trees)

Arguments

fulldata  The full/original dataset
treedata  A list of dataframes on which the trees are based
trees     A list of trees of class party, classes inheriting from party (e.g., glm.tree), or classes that can be coerced to party (i.e., rpart, Weka_tree, XMLnode).

Details

Objects of class forest contain three attributes: partytrees, treedata and fulldata. The first is a list of trees of class party (or a class inheriting from party), the second is a list of dataframes on which the trees are based, and the last is a dataframe containing the original data set. There are a couple of methods that can be used on a forest object, such as plot.forest(), print.forest(), summary.forest() and predict.forest().
Value

The function returns an object of class forest with three attributes:

- **partytrees**: A list of classification trees stored as party objects
- **treedata**: A list of dataframes on which the trees are based
- **fulldata**: The full or original dataset underlying the forest

References


Examples

```r
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 boostraps samples and grow a tree on each sample
Boots<- lapply(1:20, function(k) DrawBoots(Pima.tr ,k))
Trees <- lapply(1:20, function (i) GrowTree(x=c("npreg", "glu", "bp", "skin", "bmi", "ped", "age"),y="type", Boots[[i]] ))

# Turn the lists of dataframes and rpart trees to a forest object
myforest<- forest(Pima.tr,Boots,Trees)
```

---

**plot.clusterforest**  
*Plot a clusterforest object*

Description

A function that can be used to plot a clusterforest object, either by returning plots with information on the cluster solutions (e.g., average silhouette width), or plots of the medoid trees of each solution.
Usage

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

Arguments

- **x**: A clusterforest object
- **...**: Additional arguments
- **solution**: The solution to plot the medoid trees from. Default = NULL

Details

This function can be used to plot a clusterforest object in two ways. If it’s used with as only argument the clusterforest object, then the average silhouette width, agreement in predicted labels between medoids in solution and forest, and within cluster similarity measures are plotted for each solution. 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 two arguments (the clusterforest object and the number of the solution), then the medoid tree(s) of that solution are plotted.

References


Examples

```r
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 15 bootstrapsamples and grow a tree on each sample
Boots<- lapply(1:15, function(k) DrawBoots(Pima.tr ,k))
Trees <- lapply(1:15, function (i) GrowTree(x=c("npreg", "glu", "bp", "skin", "bmi", "ped", "age"), y="type",
```
plot.treesimilarities

Boots[[i]] )

# Turn the lists of dataframes and rpart trees to a forest object
myforest <- forest(Pima.tr, Boots, Trees)

# Calculate similarities between trees in forest, based on similarity measure m=1.
Simmatrix1 <- treesimilarities(forest=myforest, m=1)

# Cluster forest
ClusterForest <- clusterforest(Simmatrix1, 1, 5)
pplot(ClusterForest)
pplot(ClusterForest, 2)

plot.forest

Plot a forest object

Description
A function to plot a tree from a forest.

Usage
## S3 method for class 'forest'
plot(x, ..., treetoplot = 1)

Arguments
x          A clusterforest object
...        Additional arguments
treetoplot The tree from the forest to be plotted. Default=1

plot.treesimilarities

Plot a treesimilarities object

Description
A function to plot a treesimilarities object.

Usage
## S3 method for class 'treesimilarities'
plot(x, ...)

Arguments
x          A treesimilarities object
...        Additional arguments
predict.forest

Description

A function to make predictions for observations in a dataset based on a forest object.

Usage

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

Arguments

object An object of class forest
newdata A data frame containing data for which to make predictions.
... Additional arguments

Details

This function returns predicted class labels for each observation, obtained by predicting the classification label for an observation based on each tree in the forest, and subsequently taking a majority vote.

Value

produces a vector of predicted class labels

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

# Turn the lists of dataframes and rpart trees to a forest object
myforest <- forest(Pima.tr, Boots, Trees)
myforest$partytrees

# Predict labels for observations in test set Pima.te, based on the forest
predict(myforest, Pima.te)

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
...
Additional arguments
solution
The solution to print the medoid trees from. Default = NULL

print.forest
Print a forest object

Description
A function to print a forest, or a tree from a forest.

Usage
## S3 method for class 'forest'
print(x, ..., treetoprint = NULL)

Arguments
x
A clusterforest object
...
Additional arguments
treetoprint
The tree to print. If NULL then all trees are printed. Default=NULL.
print.treesimilarities

*print a treesimilarities object*

**Description**

A function to print a treesimilarities object.

**Usage**

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

**Arguments**

- `x` A treesimilarities object
- `...` Additional arguments

---

summary.clusterforest  *Summarize a clusterforest object*

**Description**

A function to summarize a clusterforest object.

**Usage**

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

**Arguments**

- `object` A clusterforest object
- `...` Additional arguments
**summary.forest**

*Summarize a forest object*

**Description**

A function to summarize a forest.

**Usage**

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

**Arguments**

- `object` A forest object
- `...` Additional arguments

**summary.treesimilarities**

*Summarize a treesimilarities object*

**Description**

A function to summarize a treesimilarities object.

**Usage**

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

**Arguments**

- `object` A treesimilarities object
- `...` Additional arguments
treesimilarities  Calculating similarities between classification trees

Description
A function to calculate similarities between classification trees in a forest, based on 6 different possible similarity measures (described in Sies & Van Mechelen, 2020). Returns an object of class treesimilarities.

Usage
treesimilarities(forest, m, tol = NULL, weight = NULL)

Arguments
forest  Object of forest class, containing the trees between which similarities should be calculated.

m  Similarity measure that should be used to calculate similarities. Here, m=1 is based on counting equal predictors or predictor-split point combinations (Equation 3 or 6 in the Supplementary Materials of Sies & Van Mechelen (2020)); m=2 is the measure of Shannon & Banks (1999), based on counting the number of equal paths from rootnode to leaves (See also Sies & Van Mechelen (2020), Equation 2); m=3 is based on the agreement in predicted classification labels (Chipman, 1998, see also Sies & Van Mechelen (2020), Equation 7); m=4 is based on the agreement of partitions (Chipman, 1998; see also Sies & Van Mechelen (2020), Equation 5); m=5 is based on counting equal elementary conjunctions of trees transformed to disjunctive normal form ((limited to binary predictors), see Sies & Van Mechelen (2020), Supplementary materials Equation 15); m=6 is based on comparing sets of predictors or predictor split point combinations (taking into account directions of the splits) associated with a leaf, while also taking into account the classification label of that leaf (see Sies & Van Mechelen (2020), Supplementary Materials Equation 13 and 14).

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

Details
This function calculates similarities between trees in a forest object (use the forest function to coerce a list of trees, a list of dataframes on which the trees were estimated, and the full data set underlying
the forest, to a forest object). The six similarity measures that are implemented, are described in Sies & Van Mechelen (2020). Each measure takes into account different aspects of similarity between trees.

The function returns a treesimilarities object, which consists of two attributes: The forest object containing the trees between which the similarities were calculated, and the similarity matrix. There are a couple of methods that can be used on treesimilarity objects, such as print.treesimilarity(), plot.treesimilarity(), summary.treesimilarity(). With the coercion method as.treesimilarity(), one can coerce a forest object and a similarity matrix to a treesimilarity object.

Value

The function returns an object of class treesimilarities. It includes two attributes:

forest The forest for which similarities were calculated.
sim Similarity matrix with similarities between all trees in the forest based on the chosen similarity measure

References


Examples

```r
require(rpart)
require(MASS)

#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 bootstrapsamples and grow a tree on each sample
Boots<- lapply(1:20, function(k) DrawBoots(Pima.tr ,k))
Trees <- lapply(1:20, function(i) GrowTree(x=c("npreg", "glu", "bp", "skin",
                                          "bmi", "ped", "age"), y="type", Boots[[i]] ))
```
# Turn the lists of dataframes and rpart trees to a forest object
myforest <- forest(Pima.tr, Boots, Trees)

# Calculate similarities between trees in forest, based on similarity measure m=1.
Simmatrix1 <- treesimilarities(forest=myforest, m=1)

# Calculate similarities between trees in forest, based on similarity measure m=6, # taking split points into account.
Simmatrix6 <- treesimilarities(forest=myforest, m=6, tol=rep(1, 7))

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 is a known source of variation underlying the forest. It visualizes the number of trees from each source that belong to each cluster.

Usage

treesource(clusterforest, solution, source)

Arguments

- clusterforest: The clusterforest object
- solution: The solution
- source: A vector with the source for each tree in the forest

Value

- multiplot: For each value of the source, a bar plot with the number of trees that belong to each cluster
- heatmap: A heatmap with for each value of the source, the number of trees that belong to each cluster

Examples

```r
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)
}

#Draw bootstrap samples and grow trees
BootsA<- lapply(1:3, function(k) DrawBoots(data_Amphet,k))
BootsC<- lapply(1:3, function(k) DrawBoots(data_cocaine,k))
BootsL<- lapply(1:3, function(k) DrawBoots(data_LSD,k))
Boots = c(BootsA,BootsC,BootsL)

TreesC <- lapply(1:3, function (i) GrowTree(x=c("Age", "Gender", "Edu", "Neuro", "Extr", "Open", "Agree", "Consc", "Impul","Sensat"), y="Coke", BootsC[[i]] ))
TreesL <- lapply(1:3, function (i) GrowTree(x=c("Age", "Gender", "Edu", "Neuro", "Extr", "Open", "Agree", "Consc", "Impul","Sensat"), y="LSD", BootsL[[i]] ))
Trees=c(TreesA,TreesC,TreesL)

#Create forest object
myforest<- forest(drugs, Boots,Trees)

#calculate similarities
Simmatrix1<- treesimilarities(forest=myforest, m=1)

#Cluster
Clusters<- clusterforest(Simmatrix1, 1, 2)

#Link cluster result to known source of variation
treesource(Clusters, 2, rep(c("Amphet","Coke","LSD"),each=3) )

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

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

Arguments

- `clusterforest`: The clusterforest object
- `solution`: The solution
- `source`: A vector with the source for each tree in the forest

---

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

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

Arguments

- `clusterforest`: The clusterforest object
- `solution`: The solution
- `source`: A vector with the source for each tree in the forest
Index

*Topic **datasets**
  drugs, 5

as.treesimilarities, 2
as.treesimilarities.list, 3

clusterforest, 4

drugs, 5

forest, 7

plot.clusterforest, 8
plot.forest, 10
plot.treesimilarities, 10
predict.forest, 11
print.clusterforest, 12
print.forest, 12
print.treesimilarities, 13

summary.clusterforest, 13
summary.forest, 14
summary.treesimilarities, 14

treesimilarities, 15
treesource, 17
treesource.clusterforest, 18
treesource.default, 19