

# Package ‘TWIX’

March 29, 2012

**Title** Trees With eXtra splits

**Date** 28.03.2012

**Version** 0.2.19

**Author** Sergej Potapov <sergej.potapov@googlemail.com>, Martin Theus  
<martin@theusrus.de>

**Maintainer** Sergej Potapov <sergej.potapov@googlemail.com>

**Description** Trees with extra splits

**Depends** R (>= 1.8.0)

**Suggests** MASS, mlbench, ElemStatLearn, rpart, randomForest, parallel

**License** GPL-2

**URL** <http://www.rosuda.org/software/TWIX/>

**Repository** CRAN

**Date/Publication** 2012-03-29 08:01:03

## R topics documented:

bagg.default . . . . .	2
bootTWIX . . . . .	3
deviance.TWIX . . . . .	6
Devplot . . . . .	7
export . . . . .	8
fullrks . . . . .	9
get.splitvar . . . . .	9
get.tree . . . . .	10
olives . . . . .	10
plot.TWIX . . . . .	11
predict.TWIX . . . . .	12
print.id.tree . . . . .	13

print.single.tree . . . . .	14
print.TWIX . . . . .	15
scree.plot . . . . .	15
splitt . . . . .	16
splitt.padj . . . . .	17
summary.TWIX . . . . .	18
trace.plot . . . . .	18
tune.TWIX . . . . .	19
TWIX . . . . .	21

<b>Index</b>	<b>25</b>
--------------	-----------

---

bagg.default	<i>Predictions from TWIX's or Bagging Trees</i>
--------------	---

---

## Description

Prediction of a new observation based on multiple trees.

## Usage

```
## Default S3 method:
bagg(object,newdata = NULL, sq=1:length(object$trees),
      aggregation = "weighted", type = "class", ...)
## S3 method for class 'TWIX'
bagg(object,...)
## S3 method for class 'bootTWIX'
bagg(object,...)
```

## Arguments

object	object of classes TWIX or bootTWIX.
newdata	a data frame of new observations.
sq	Integer vector indicating for which trees predictions are required.
aggregation	character string specifying how to aggregate. There are three ways to aggregate the TWIX trees. One of them is the class majority voting (aggregation="majority"), another method is the weighted aggregation (aggregation="weighted").
type	character string indicating the type of predicted value returned. Either class predicted classes or prob estimated class probabilities are returned.
...	additional arguments.

## See Also

[TWIX](#), [predict.TWIX](#), [bootTWIX](#)

**Examples**

```

library(ElemStatLearn)
data(SAheart)

### response variable must be a factor
SAheart$chd <- factor(SAheart$chd)

### test and train data
###
set.seed(1234)
icv <- sample(nrow(SAheart),nrow(SAheart)/3)
itr <- setdiff(1:nrow(SAheart),icv)
train <- SAheart[itr,]
test <- SAheart[icv,]

### TWIX Ensemble
M1 <- TWIX(chd~.,data=train,topN=c(9,5),topn.method="single")

### Bagging with greedy decision trees as base classifier
M2 <- bootTWIX(chd~.,data=train,nbagg=50)

### Bagging with the p-value adjusted classification trees
### as base classifier
M3 <- bootTWIX(chd~.,data=train,nbagg=50,splitf="p-adj",Devmin=0.01)

preda <- bagg(M1,test,sq=1:length(M1$trees),aggregation="majority")
predb <- bagg(M1,test,sq=1:length(M1$trees),aggregation="weighted")
pred1 <- predict(M2,test,sq=1:length(M2$trees))
pred2 <- predict(M3,test,sq=1:length(M3$trees))

###
### CCR's

sum(preda == test$chd)/nrow(test)
sum(predb == test$chd)/nrow(test)
sum(pred1 == test$chd)/nrow(test)
sum(pred2 == test$chd)/nrow(test)

```

bootTWIX

*Bootstrap of the TWIX trees***Description**

Bootstrap samples of the Greedy-TWIX-trees or p-value adjusted TWIX-trees.

**Usage**

```
bootTWIX(formula, data = NULL, nbagg = 25, topN = 1, subset = NULL,
         comb = NULL, method = "deviance", topn.method = "complete",
```

```
replace = TRUE, ns=1, minsplit = 2, minbucket = round(minsplit/3),
splitf = "deviance", Devmin = 0.01, level = 30, tol = 0.01,
cluster = NULL, seed.cluster = NULL)
```

### Arguments

formula	formula of the form $y \sim x_1 + x_2 + \dots$ , where $y$ must be a factor and $x_1, x_2, \dots$ are numeric.
data	an optional data frame containing the variables in the model (training data).
nbagg	an integer giving the number of bootstrap replications.
comb	a list of additional model and its prediction function for model combination, see below for some examples.
splitf	kind of the splitting function to be used. It can be one of "deviance"(default) or "p-adj". If splitf set to "p-adj", the p-value adjusted classification tree will be performed.
replace	Should sampling be with replacement?
ns	data set of size $ns \leq nrow(data)$ obtained by sampling without replacement.
Devmin	the minimum improvement on entropy by splitting or by the p-value adjusted classification trees the significance level alpha.
topN	integer vector. How many splits will be selected and at which level? If length 1, the same size of splits will be selected at each level. If length > 1, for example $topN=c(3, 2)$ , 3 splits will be chosen at first level, 2 splits at second level and for all next levels 1 split.
subset	an optional vector specifying a subset of observations to be used.
method	Which split points will be used? This can be "deviance" (default), "grid" or "local". If the method is set to: "local" - the program uses the local maxima of the split function(entropy), "deviance" - all values of the entropy, "grid" - grid points.
topn.method	one of "complete"(default) or "single". A specification of the consideration of the split points. If set to "complete" it uses split points from all variables, else it uses split points per variable.
minsplit	the minimum number of observations that must exist in a node.
minbucket	the minimum number of observations in any terminal <leaf> node.
level	maximum depth of the trees. If level set to 1, trees consist of root node.
tol	parameter, which will be used, if topn.method is set to "single".
cluster	the name of the cluster, if parallel computing will be used.
seed.cluster	an integer to be supplied to set.seed, or NULL not to set reproducible seeds.

**Value**

a list with the following components :

`call`            the call generating the object.  
`trees`           a list of all constructed trees, which include ID, Dev, Fit, Splitvar, ... for each tree.

**See Also**

[TWIX](#), [get.tree](#), [predict.bootTWIX](#), [deviance.TWIX](#), [bagg.TWIX](#)

**Examples**

```
library(ElemStatLearn)
data(SAheart)

### response variable must be a factor
SAheart$chd <- factor(SAheart$chd)

### test and train data
###
set.seed(1234)
icv <- sample(nrow(SAheart),nrow(SAheart)*0.3)
itr <- setdiff(1:nrow(SAheart),icv)
train <- SAheart[itr,]
test <- SAheart[icv,]

### Bagging with greedy decision trees as base classifier
M1 <- bootTWIX(chd~.,data=train,nbagg=50)

### Bagging with the p-value adjusted classification trees (alpha is 0.01)
### as base classifier
M2 <- bootTWIX(chd~.,data=train,nbagg=50,splitf="p-adj",Devmin=0.01)

library(MASS)

### Double-Bagging: combine LDA and classification trees
comb.lda <- list(model=lda, predict=function(obj, newdata)
                predict(obj, newdata)$x)

M3 <- bootTWIX(chd~.,data=train,nbagg=50,comb=comb.lda)

### Double-Bagging: combine LDA and p-value adjusted classification trees
comb.lda <- list(model=lda, predict=function(obj, newdata)
                predict(obj, newdata)$x)
```

```

M4 <- bootTWIX(chd~.,data=train,nbagg=50,comb=comb.lda,
               splitf="p-adj",Devmin=0.01)

### Double-Bagging: combine GLM and classification trees
comb.glm <- list(model=function(x,...){glm(x,family=binomial,...)},
                 predict=function(obj, newdata) predict(obj, newdata))

M5 <- bootTWIX(chd~.,data=train,nbagg=50,comb=comb.glm)

### Double-Bagging: combine GLM and p-value adjusted classification trees
comb.glm <- list(model=function(x,...){glm(x,family=binomial,...)},
                 predict=function(obj, newdata) predict(obj, newdata))

M6 <- bootTWIX(chd~.,data=train,nbagg=50,comb=comb.glm,
               splitf="p-adj",Devmin=0.01)

pred1 <- predict(M1,test)
pred2 <- predict(M2,test)
pred3 <- predict(M3,test)
pred4 <- predict(M4,test)
pred5 <- predict(M5,test)
pred6 <- predict(M6,test)

###
### CCR's

sum(pred1 == test$chd)/nrow(test)
sum(pred2 == test$chd)/nrow(test)
sum(pred3 == test$chd)/nrow(test)
sum(pred4 == test$chd)/nrow(test)
sum(pred5 == test$chd)/nrow(test)
sum(pred6 == test$chd)/nrow(test)

```

---

deviance.TWIX

*Deviance of the TWIX-trees*


---

### Description

Returns the deviance of a fitted model object.

### Usage

```

## S3 method for class 'TWIX'
deviance(object, type = "training", ...)

```

**Arguments**

object	an object of class <i>TWIX</i> or <i>bootTWIX</i> for which the deviance is desired.
type	a deviance from test, training or from both data.
...	additional arguments.

**Value**

The value of the deviance extracted from the object object.

**See Also**

[TWIX](#), [predict.TWIX](#), [bootTWIX](#)

**Examples**

```
data(olives)
Tree <- TWIX(Region~., data=olives, topN=c(5,3), method="local")
deviance(Tree)
```

---

Devplot

*Deviance plot*


---

**Description**

Deviance plot.

**Usage**

```
Devplot(rsp, x, col = 1, classes = FALSE,
        pch = 16, nsample = 0, ...)
```

**Arguments**

rsp	response variable.
x	a dataframe of predictor variables.
col	the color for points.
classes	Scatterplot of Classes.
pch	a vector of plotting characters or symbols.
nsample	the number of bootstrap samples without replacement.
...	other parameters to be passed through to plotting functions.

**See Also**

[plot.TWIX](#), [TWIX](#), [trace.plot](#), [scree.plot](#)

### Examples

```
data(olives)

### for one predictor variable
Devplot(olives$Region,olives$oleic)

### with classes
Devplot(olives$Region,olives$oleic,classes=TRUE)

### for more as one predictor variable
Devplot(olives$Region,olives[,1:8])
```

---

export

*Export TWIX-trees for KLIMIT*

---

### Description

Export TWIX-trees for KLIMIT.

### Usage

```
export(x, sq = 1, directory = "ForKlimt")
```

### Arguments

x	an object of class TWIX.
sq	a vector of tree IDs.
directory	a name of directory, which will be created.

### References

Urbanek Simon (2002). KLIMIT - A COSADA Software Project.  
<http://rosuda.org/KLIMIT>

### See Also

[predict.TWIX](#), [print.TWIX](#)

### Examples

```
data(olives)
Tree <- TWIX(Region~.,data=olives,topN=c(3,2),method="local",level=5)

###
#export(Tree,c(1,3))
```

---

fullrks	<i>Internal function of TWIX.</i>
---------	-----------------------------------

---

**Description**

Internal function of TWIX.

**Usage**

```
fullrks(m)
```

**Arguments**

m                    a numeric or character vector

**Examples**

```
fullrks(c(3,1,2,4,5))
```

---

get.splitvar	<i>Method for getting split variables from a single tree.</i>
--------------	---

---

**Description**

This function extracts a split variables from a `single.tree` object.

**Usage**

```
get.splitvar(x, sq = 1:length(x$trees), parm = "Splitvar")
```

**Arguments**

x                    an object of class TWIX generated by TWIX  
sq                    Integer vector indicating for which trees a split variables are required.  
parm                  Which information must be returned? This can be "Splitvar", "Split" or "Dev"

**See Also**

[TWIX](#), [get.tree](#), [predict.TWIX](#)

**Examples**

```
data(olives)
TreeFR <- TWIX(Area~., data=olives, topN=c(2,1), method="local", level=5)
TreeSG <- get.splitvar(TreeFR, sq=1)
TreeSG
get.tree(TreeFR)
```

---

<code>get.tree</code>	<i>Method for getting appointed trees from a TWIX-multitree.</i>
-----------------------	--

---

**Description**

This function extracts a single tree from a TWIX object.

**Usage**

```
get.tree(m.tr, n = 1, id = NULL)
```

**Arguments**

<code>m.tr</code>	an object of class TWIX generated by TWIX
<code>n</code>	integer. Which tree must be extracted?
<code>id</code>	a vector of integers. ID of the tree.

**See Also**

[TWIX](#), [predict.TWIX](#)

**Examples**

```
data(olives)
TreeFR <- TWIX(Area~., data=olives, topN=c(2,1), method="local")
TreeSG <- get.tree(TreeFR, n=1)
TreeSG
```

---

<code>olives</code>	<i>Classification of olive oils from their fatty acid composition.</i>
---------------------	--

---

**Description**

The `olives` data frame has 572 rows and 10 columns.

**Usage**

```
olives
```

**Format**

This data frame contains the following columns:

Area a factor with 3 levels.

Region a factor with 9 levels.

and 8 variables of fatty acid measurements.

**Source**

Forina, M. and Armanino, C. and Lanteri, S. and Tiscornia, E. (1983)  
*Food Research and Data Analysis*,  
 Applied Science Publishers, CA 1983.

---

 plot.TWIX

---

*Plotting method for TWIX objects*


---

**Description**

Plot an TWIX or bootTWIX object generated by TWIX or bootTWIX function(s).

**Usage**

```
## S3 method for class 'TWIX'
plot(x, sq = 1:length(x$trees), type = "deviance",
     size = 3, freq = TRUE, breaks = "Sturges",
     pch = par("pch"), ...)
```

**Arguments**

x	an object of class TWIX.
sq	Integer vector giving the number of trees to be plotted.
type	one of "deviance", "ccr", "d&c".
size	value for largest circle (cex).
freq	logical. Should the frequency or density be plotted.
breaks	see histogram.
pch	a vector of plotting characters or symbols.
...	graphical parameters can be given as arguments to 'plot'.

**Details**

If type = "deviance":  
 the training deviance vs. test deviance will be plotted.  
 If type = "ccr":  
 the correct classification rate(CCR) for training data vs. the CCR for test data.  
 If type = "d&c":  
 the deviance vs. CCR for test data.

**See Also**

[TWIX](#), [get.tree](#)

**Examples**

```

data(olives)

### train and test data
set.seed(123)
i <- sample(572,150)
ic <- setdiff(1:572,i)
training <- olives[ic,]
test <- olives[i,]

TM1 <- TWIX(Region~.,data=training[,1:9],test.data=test,
            topN=c(5,3),method="local")

plot(TM1)
plot(TM1,type="ccr")
plot(TM1,type="d&c")

```

---

predict.TWIX

*Predictions from a TWIX object*


---

**Description**

The result is a data frame, whose rows are prediction values from appointed tree(s).

**Usage**

```

## S3 method for class 'TWIX'
predict(object,newdata,sq=1,ccr=FALSE,type="class", ...)

```

**Arguments**

object	an object returned from TWIX function.
newdata	data frame containing the new data(test data).
sq	Integer vector indicating for which trees predictions are required.
ccr	logical. If TRUE the result is a list of two components: a data frame with prediction values and correct classification rate of trees.
type	character string indicating the type of predicted value returned. Either class predicted classes or prob estimated class probabilities are returned.
...	additional arguments.

**See Also**

[bagg](#), [TWIX](#), [plot.TWIX](#)

## Examples

```
library(ElemStatLearn)
data(SAheart)

### response variable must be a factor
SAheart$chd <- factor(SAheart$chd)

### test and train data
###
set.seed(1234)
icv <- sample(nrow(SAheart),nrow(SAheart)/3)
itr <- setdiff(1:nrow(SAheart),icv)
train <- SAheart[itr,]
test <- SAheart[icv,]

M1 <- TWIX(chd~.,data=train,topN=c(4,3),topn.method="single")

### classification
pred <- predict(M1,newdata=test,sq=1:2)
pred

### for correct classification rate
predict(M1,newdata=test,sq=1:2,ccr=TRUE)$CCR

### estimated class probabilities
predict(M1,newdata=test,sq=1,type="prob")
```

---

print.id.tree

*Internal function of TWIX.*

---

## Description

Print names from id.tree object.

## Usage

```
## S3 method for class 'id.tree'
print(x, sq = 1:5, ...)
```

## Arguments

x	an object of class id.tree.
sq	a numeric vector.
...	further arguments passed to or from other methods.

## See Also

[TWIX](#)

---

print.single.tree      *Print tree from single.tree object.*

---

### Description

This is a method for the generic print() function for objects generating by the function get . tree.

### Usage

```
## S3 method for class 'single.tree'  
print(x, klimt=FALSE, Data=NULL, file="FromR.tree", ...)
```

### Arguments

x	an object of class single.tree.
klimt	logical. If TRUE, Klimt will be started with the tree baum und dataset Data.
Data	a data frame. It can be test or training data. This parameter is ignored if klimt == "FALSE".
file	a character string naming a file.
...	further arguments passed to or from other methods.

### References

Urbanek Simon (2002). KLIMT - A COSADA Software Project.  
<http://rosuda.org/KLIMT>

### See Also

[get.tree](#), [TWIX](#)

### Examples

```
data(olives)  
Tree <- TWIX(Area~.,data=olives,topN=c(2,2),method="local")  
Tree <- get.tree(Tree,n=1)  
Tree  
  
### for further analysis in KLIMT  
print(Tree,klimt=TRUE,Data=olives)
```

---

print.TWIX	<i>Print method for TWIX or BootTWIXtree object</i>
------------	---

---

**Description**

Print object of class TWIX or bootTWIX.

**Usage**

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

**Arguments**

x	object of class TWIX or bootTWIX.
...	additional arguments.

**Details**

An object of class TWIX or bootTWIX is printed.

**See Also**

[TWIX](#), [print.single.tree](#)

---

scree.plot	<i>Scree-plot</i>
------------	-------------------

---

**Description**

A scree plot shows the sorted maximum decrease in impurity for each variable's value.

**Usage**

```
scree.plot(formula, data = NULL, bars = TRUE, col = "grey",  
           type = "b", pch = 16, ylim = c(0, 1), ...)
```

**Arguments**

formula	formula of the form $y \sim x_1 + x_2 + \dots$ , where $y$ must be a factor and $x_1, x_2, \dots$ are numeric or factor.
data	an optional data frame containing the variables in the model.
bars	the type of plot: barplot or lines.
col	the colors for lines or Bar's.
type	the color for points.
pch	see <code>par(pch = ...)</code> .
ylim	the y limits of the plot.
...	other parameters to be passed through to plotting functions.

**See Also**

[plot.TWIX](#), [TWIX](#)

**Examples**

```
data(olives)
screen.plot(Region~.,data=olives,bars=FALSE,col=2)
screen.plot(Region~.,data=olives,bars=TRUE)
```

---

splitt

*Internal function of TWIX.*

---

**Description**

This function compute split-points and corresponding deviance gain.

**Usage**

```
splitt(sv, rsp, meth = "deviance", topn = 1,
       topn.meth = "complete", lstep = 1,
       test = FALSE, K = 0, level = 0, minbuck = 1)
```

**Arguments**

sv	a numeric vector of predicted variable.
rsp	response variable.
meth	Which split points will be used? This can be "deviance" (default), "grid" or "local".
topn	a numeric vector. How many splits will be selected and at which level?
topn.meth	one of "complete"(default) or "single".
lstep	step parameter for method "grid".

test	parameter for Devplot funcion.
K	k-fold cross-validation.
level	the maximum depth of the TWIX tree's.
minbuck	the minimum number of observations.

---

`splitt.padj`*Internal function of TWIX.*

---

### Description

This function compute the split-point and the corresponding p-value.

### Usage

```
splitt.padj(sv, rsp, minprop = 0.1, maxprop = 0.9,  
           minbuck = 1, test = FALSE)
```

### Arguments

sv	a numeric vector of predicted variable.
rsp	response variable.
minprop	at least minprop*100% of the observations in the first group.
maxprop	not more than maxprop*100% of the observations in the first group.
minbuck	the minimum number of observations.
test	logical: return the whole information.

### References

Lausen, B., Hothorn, T., Bretz, F. and Schmacher, M. (2004).  
Assessment of Optimally Selected Prognostic Factors.  
*Biometrical Journal* **46**, 364-374.

---

summary.TWIX

*Summarising TWIX*


---

### Description

summary method for objects returned by [TWIX](#) or bootTWIX.

### Usage

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

### Arguments

object            object of class TWIX or bootTWIX.  
 ...               further arguments to be passed to or from methods.

### See Also

[TWIX](#)

---

trace.plot

*Trace plot*


---

### Description

A Trace plot shows the structure of the trees

### Usage

```
trace.plot(obj, sq = 1, quality = NULL,
           color.palette = topo.colors, alpha = 1)
```

### Arguments

obj                formula of the form  $y \sim x_1 + x_2 + \dots$ , where  $y$  must be a factor and  $x_1, x_2, \dots$  are numeric or factor.  
 sq                 an optional data frame containing the variables in the model.  
 quality           the type of plot: barplot or lines.  
 color.palette    the colors for lines or Bar's.  
 alpha             the alpha transparency, a number in  $[0, 1]$ .

## References

- Urbanek, S. (2002). KLIMIT - A COSADA Software Project.  
<http://rosuda.org/KLIMIT>
- Urbanek, S. (2005). Following Traces of Lost Models.  
*Proc. of the joint Statistical Meetings 2005, Section on Statistical Graphics*  
 Mira Digital Publishing.

## See Also

[plot.TWIX](#), [TWIX](#), [scree.plot](#)

## Examples

```
data(olives)
Tree1 <- TWIX(Region~., data=olives[,1:9], topN=c(2,2), method="local")
get.tree(Tree1)

### Trace plot of classification tree
trace.plot(Tree1, sq=8)

### Trace plot of eight classification trees
trace.plot(Tree1, sq=1:8)
```

---

tune.TWIX

*Parameter tuning via K-fold cross-validation.*

---

## Description

This function tunes hyperparameters: minbuck, cp for TWIX or rpart, mtry for randomForest.

## Usage

```
tune.TWIX(formula, data = NULL, minbuck = seq(5,20,by=5),
  maxdepth=30, Devmin=0.0, splitf="deviance",
  method="deviance", topn.method="single",
  topN=1, tol=0.25, cp=0.0, xval=10, runs = 1,
  trace.plot=FALSE, score=1, predict="best",
  cluster=NULL, seed.cluster=NULL, multicore=FALSE)

tune.cp.TWIX(formula, data = NULL, cp=seq(0.01,0.1,length=10),
  minbuck=1, maxdepth=30, Devmin=0.0, splitf="deviance",
  method="deviance", topn.method="single",
  topN=1, tol=0, xval=10, runs = 1, trace.plot=FALSE,
  score=1, predict="best", cluster=NULL,
  seed.cluster=NULL, multicore=FALSE)

tune.rpart(formula, data = NULL, minbuck=seq(5,20,by=5),
```

```

parms=list(split="information"), maxdepth=30,
cp=0.0, xval=10, runs=1, trace.plot=FALSE,
cluster=NULL, seed.cluster=NULL, multicore=FALSE)

tune.cp.rpart(formula, data = NULL, cp=seq(0,0.3,0.05),
minbuck=0, parms=list(split="information"),
maxdepth=30, xval=10, runs=10, trace.plot=FALSE,
cluster=NULL, seed.cluster=NULL, multicore=FALSE)

tune.RF(formula, data = NULL, mtry = 2:(ncol(data)-1),
ntree=500, replace=TRUE, xval=10, runs = 1,
trace.plot = FALSE, cluster=NULL, seed.cluster=NULL,
multicore=FALSE)

```

### Arguments

formula	formula of the form $y \sim x_1 + x_2 + \dots$ , where $y$ must be a factor and $x_1, x_2, \dots$ are numeric or factor.
data	an optional data frame containing the variables in the model.
minbuck	the sampling space for parameter minbuck.
maxdepth	set the maximum depth of the final tree.
xval	number of cross-validations.
runs	number of runs.
splitf	kind of the splitting function to be used. It can be one of "deviance"(default) or "p-adj". If set to "p-adj", the p-value adjusted classification tree will be performed.
Devmin	the minimum improvement on entropy by splitting. If "splitf" set to "p-adj", "Devmin" will be the significance level alpha.
method	Which split points will be used? This can be "deviance" (default), "grid" or "local". If the method is set to: "local" - the program uses the local maxima of the split function (entropy), "deviance" - all values of the entropy, "grid" - grid points.
topn.method	one of "complete"(default) or "single". A specification of the consideration of the split points. If set to "complete" it uses split points from all variables, else it uses split points per variable.
topN	integer vector. How many splits will be selected and at which level?
tol	parameter, which will be used, if topn.method is set to "single".
cp	complexity parameter. See <a href="#">rpart.control</a> .
score	Specifies the method for model selection. See <a href="#">TWIX</a> .
predict	a string that specifies what prediction method will be used. Possible values are "best" for a best single-tree prediction and "ensemble" for prediction based on all trees (see <a href="#">bagg</a> ).
cluster	the name of the snow cluster, if parallel computing will be used

<code>seed.cluster</code>	an integer to be supplied to <code>set.seed</code> , or <code>NULL</code> not to set reproducible seeds.
<code>multicore</code>	a logical value for parallel execution with package <code>multicore</code> .
<code>parms</code>	optional parameters for the splitting function. See <a href="#">rpart</a> .
<code>mtry</code>	Number of variables randomly sampled as candidates at each split. See <code>randomForest</code> .
<code>ntree</code>	Number of trees to grow. See <code>randomForest</code> .
<code>replace</code>	a logical indicating whether sampling of observations is done with or without replacement.
<code>trace.plot</code>	Should trace plot be plotted?

**See Also**

[plot.TWIX](#), [TWIX](#)

**Examples**

```
library(mlbench)
data(PimaIndiansDiabetes2)
Pima <- na.omit(PimaIndiansDiabetes2)

tune.TWIX(diabetes~., data=Pima, minbuck=c(5, 10, 15, 20, 25), runs=5)
```

---

TWIX

*Trees with extra splits*

---

**Description**

Trees with extra splits

**Usage**

```
TWIX(formula, data=NULL, test.data=NULL, subset=NULL,
      method="deviance", topn.method="complete", minsplit=20,
      minbucket=round(minsplit/3), topN=1, splitf="deviance",
      Devmin=0.01, tol=0.25, cp=0.01, level=30, st=1, score=1,
      k=0, cluster=NULL, seed.cluster=NULL, cl.level=1, multicore=FALSE,
      trace=TRUE, trace.plot=FALSE, ...)
```

**Arguments**

<code>formula</code>	formula of the form $y \sim x_1 + x_2 + \dots$ , where $y$ must be a factor and $x_1, x_2, \dots$ are numeric or factor.
<code>data</code>	an optional data frame containing the variables in the model (training data).
<code>test.data</code>	This can be a data frame containing new data.
<code>subset</code>	an optional vector specifying a subset of observations to be used.

method	Which split points will be used? This can be "deviance" (default), "grid" or "local". If the method is set to: "local" - the program uses the local maxima of the split function (entropy), "deviance" - all values of the entropy, "grid" - grid points.
topn.method	one of "complete"(default) or "single". A specification of the consideration of the split points. If set to "complete" it uses split points from all variables, else it uses split points per variable.
minsplit	the minimum number of observations that must exist in a node.
minbucket	the minimum number of observations in any terminal <leaf> node.
topN	integer vector. How many splits will be selected and at which level? If length 1, the same size of splits will be selected at each level. If length > 1, for example topN=c(3, 2), 3 splits will be chosen at first level, 2 splits at second level and for all next levels 1 split.
splitf	kind of the splitting function to be used. It can be one of "deviance"(default) or "p-adj". If set to "p-adj", the p-value adjusted classification tree will be performed.
Devmin	the minimum improvement on entropy by splitting. If "splitf" set to "p-adj", "Devmin" will be the significance level alpha.
tol	parameter, which will be used, if topn.method is set to "single".
cp	complexity parameter.
level	the maximum depth of the trees. If level set to 1, trees consist of root node.
st	step parameter for method "grid".
cluster	the name of the snow cluster, if parallel computing will be used.
seed.cluster	an integer to be supplied to set.seed, or NULL not to set reproducible seeds.
cl.level	an internal parameter of parallel computing.
multicore	a logical value for parallel execution with package multicore.
score	Specifies the method for model selection. This can be 1(default), 2 or 3. If it is 1 the weighted correct classification rate will be used, if it is 2 the <i>sort</i> -function will be used, if it set to 3 the <i>weight</i> -function will be used $score = 0.25 * scale(dev.tr) + 0.6 * scale(fit.tr) + 0.15 * (structure)$
k	k-fold cross-validation of split-function. k specify the part of observations which will be take in hold-out sample (k can be (0,0.5)).
trace	A logical for printing a training log.
trace.plot	Should trace plot be plotted?
...	further arguments to be passed to or from methods.

### Details

This implementation can't handle missing values. Therefore, cases with missing values must be removed. For p-value adjusted classification trees, continuous and binary independent descriptors are implemented as predictors and a response variable must be categorical with two categories.

**Value**

a list with the following components :

call	the call generating the object.
trees	a list of all constructed trees, which include ID, Dev, Fit, Splitvar, ... for each tree.
greedy.tree	greedy tree
multitree	database
score	score values

**References**

Martin Theus, Sergej Potapov and Simon Urbanek (2006).  
 TWIX (Talk given at the 3rd Ensemble Workshop in Munich 2006).  
<http://theusrus.de/Talks/Talks/TWIX.pdf>

Lausen, B., Hothorn, T., Bretz, F. and Schmacher, M. (2004).  
 Assessment of Optimally Selected Prognostic Factors.  
*Biometrical Journal* **46**, 364-374.

**See Also**

[get.tree](#), [predict.TWIX](#), [print.single.tree](#), [plot.TWIX](#), [bootTWIX](#)

**Examples**

```
data(olives)

### train and test data
set.seed(123)
i <- sample(572,150)
ic <- setdiff(1:572,i)
training <- olives[ic,]
test <- olives[i,]

###
### TWIX Ensemble: 729 classification trees

TM <- TWIX(Region~.,data=training[,1:9],topN=c(9,9),method="local")
TM$trees
get.tree(TM,1)
pred <- predict(TM,newdata=test,sq=1)

### for correct classification rate
predict(TM,newdata=test,sq=1:36,ccr=TRUE)$CCCR

###
### the p-value adjusted classification tree
```

```
library(mlbench)
data(PimaIndiansDiabetes2)
Pima <- na.omit(PimaIndiansDiabetes2)

### train and test data
set.seed(1111)
N <- nrow(Pima)
icv <- sample(N,N/3)
itr <- setdiff(1:N,icv)
train <- Pima[itr,]
test <- Pima[icv,]

###
### the p-value adjusted classification tree with alpha = 0.05

TMa <- TWIX(diabetes~.,data=train,splitf="p-adj",Devmin=0.05)
get.tree(TMa)

### for correct classification rate
predict(TMa,newdata=test,ccr=TRUE)$CCR
```

# Index

- \*Topic **classif**
    - bootTWIX, 3
    - TWIX, 21
  - \*Topic **datasets**
    - olives, 10
  - \*Topic **hplot**
    - Devplot, 7
    - plot.TWIX, 11
    - scree.plot, 15
    - trace.plot, 18
  - \*Topic **print**
    - print.id.tree, 13
    - print.single.tree, 14
    - print.TWIX, 15
  - \*Topic **tree**
    - bagg.default, 2
    - bootTWIX, 3
    - deviance.TWIX, 6
    - Devplot, 7
    - export, 8
    - get.splitvar, 9
    - get.tree, 10
    - plot.TWIX, 11
    - predict.TWIX, 12
    - print.single.tree, 14
    - splitt, 16
    - splitt.padj, 17
    - TWIX, 21
  - \*Topic **utilities**
    - deviance.TWIX, 6
    - export, 8
    - fullrks, 9
    - summary.TWIX, 18
    - tune.TWIX, 19
- bagg, 12, 20  
bagg (bagg.default), 2  
bagg.default, 2  
bagg.TWIX, 5  
bootTWIX, 2, 3, 7, 23  
deviance.bootTWIX (deviance.TWIX), 6  
deviance.TWIX, 5, 6  
Devplot, 7  
export, 8  
fullrks, 9  
get.splitvar, 9  
get.tree, 5, 9, 10, 11, 14, 23  
olives, 10  
plot.bootTWIX (plot.TWIX), 11  
plot.TWIX, 7, 11, 12, 16, 19, 21, 23  
predict.bootTWIX, 5  
predict.bootTWIX (predict.TWIX), 12  
predict.bundlTWIX (predict.TWIX), 12  
predict.TWIX, 2, 7–10, 12, 23  
print.bootTWIX (print.TWIX), 15  
print.id.tree, 13  
print.single.tree, 14, 15, 23  
print.TWIX, 8, 15  
rpart, 21  
rpart.control, 20  
scree.plot, 7, 15, 19  
splitt, 16  
splitt.padj, 17  
summary.bootTWIX (summary.TWIX), 18  
summary.TWIX, 18  
trace.plot, 7, 18  
tune.cp.rpart (tune.TWIX), 19  
tune.cp.TWIX (tune.TWIX), 19  
tune.RF (tune.TWIX), 19  
tune.rpart (tune.TWIX), 19  
tune.TWIX, 19  
TWIX, 2, 5, 7, 9–16, 18–20, 21, 21