Package 'ipred'

November 5, 2018

Title Improved Predictors

Version 0.9-8

Date 2018-11-05

Description Improved predictive models by indirect classification and bagging for classification, regression and survival problems as well as resampling based estimators of prediction error.

Depends R (>= 2.10)

Imports rpart (>= 3.1-8), MASS, survival, nnet, class, prodlim

Suggests mvtnorm, mlbench, TH.data

License GPL (>= 2)

NeedsCompilation yes

Author Andrea Peters [aut],
Torsten Hothorn [aut, cre],
Brian D. Ripley [ctb],
Terry Therneau [ctb],
Beth Atkinson [ctb]

Maintainer Torsten Hothorn <Torsten.Hothorn@R-project.org>

Repository CRAN

Date/Publication 2018-11-05 13:40:03 UTC

R topics documented:

---

bagging *Bagging Classification, Regression and Survival Trees*

Description

Bagging for classification, regression and survival trees.

Usage

```
## S3 method for class 'factor'
ipredbagg(y, X=NULL, nbagg=25, control=
                rpart.control(minsplit=2, cp=0, xval=0),
                comb=NULL, coob=FALSE, ns=length(y), keepX = TRUE, ...)
## S3 method for class 'numeric'
ipredbagg(y, X=NULL, nbagg=25, control=rpart.control(xval=0),
                 comb=NULL, coob=FALSE, ns=length(y), keepX = TRUE, ...)
## S3 method for class 'Surv'
ipredbagg(y, X=NULL, nbagg=25, control=rpart.control(xval=0),
               comb=NULL, coob=FALSE, ns=dim(y)[1], keepX = TRUE, ...)
## S3 method for class 'data.frame'
bagging(formula, data, subset, na.action=na.rpart, ...)
```

Arguments

y	the response variable: either a factor vector of class labels (bagging classification trees), a vector of numerical values (bagging regression trees) or an object of class Surv (bagging survival trees).
X	a data frame of predictor variables.
nbagg	an integer giving the number of bootstrap replications.
coob	a logical indicating whether an out-of-bag estimate of the error rate (misclassification error, root mean squared error or Brier score) should be computed. See predict.classbagg for details.
control	options that control details of the rpart algorithm, see rpart.control. It is wise to set xval = 0 in order to save computing time. Note that the default values depend on the class of y.
comb	a list of additional models for model combination, see below for some examples. Note that argument method for double-bagging is no longer there, comb is much more flexible.
ns	number of sample to draw from the learning sample. By default, the usual bootstrap n out of n with replacement is performed. If ns is smaller than length(y), subagging (Buehlmann and Yu, 2002), i.e. sampling ns out of length(y) without replacement, is performed.
keepX	a logical indicating whether the data frame of predictors should be returned. Note that the computation of the out-of-bag estimator requires keepX=TRUE.
formula	a formula of the form lhs ~ rhs where lhs is the response variable and rhs a set of predictors.
data	optional data frame containing the variables in the model formula.
subset	optional vector specifying a subset of observations to be used.
na.action	function which indicates what should happen when the data contain NAs. Defaults to na.rpart.
...	additional parameters passed to ipredbagg or rpart, respectively.

Details

The random forest implementations randomForest and cforest are more flexible and reliable for computing bootstrap-aggregated trees than this function and should be used instead.

Bagging for classification and regression trees were suggested by Breiman (1996a, 1998) in order to stabilise trees.

The trees in this function are computed using the implementation in the rpart package. The generic function ipredbagg implements methods for different responses. If y is a factor, classification trees are constructed. For numerical vectors y, regression trees are aggregated and if y is a survival object, bagging survival trees (Hothorn et al, 2003) is performed. The function bagging offers a formula based interface to ipredbagg.

nbagg bootstrap samples are drawn and a tree is constructed for each of them. There is no general rule when to stop the tree growing. The size of the trees can be controlled by control argument or prune.classbagg. By default, classification trees are as large as possible whereas regression trees

and survival trees are build with the standard options of rpart.control. If nbagg=1, one single tree is computed for the whole learning sample without bootstrapping.

If coob is TRUE, the out-of-bag sample (Breiman, 1996b) is used to estimate the prediction error corresponding to class(y). Alternatively, the out-of-bag sample can be used for model combination, an out-of-bag error rate estimator is not available in this case. Double-bagging (Hothorn and Lausen, 2003) computes a LDA on the out-of-bag sample and uses the discriminant variables as additional predictors for the classification trees. comb is an optional list of lists with two elements model and predict. model is a function with arguments formula and data. predict is a function with arguments object, newdata only. If the estimation of the covariance matrix in lda fails due to a limited out-of-bag sample size, one can use slda instead. See the example section for an example of double-bagging. The methodology is not limited to a combination with LDA: bundling (Hothorn and Lausen, 2002b) can be used with arbitrary classifiers.

NOTE: Up to ipred version 0.9-0, bagging was performed using a modified version of the original rpart function. Due to interface changes in rpart 3.1-55, the bagging function had to be rewritten. Results of previous version are not exactly reproducible.

Value

The class of the object returned depends on class(y): classbagg, regbagg and survbagg. Each is a list with elements

y	the vector of responses.
X	the data frame of predictors.
mtrees	multiple trees: a list of length nbagg containing the trees (and possibly additional objects) for each bootstrap sample.
OOB	logical whether the out-of-bag estimate should be computed.
err	if OOB=TRUE, the out-of-bag estimate of misclassification or root mean squared error or the Brier score for censored data.
comb	logical whether a combination of models was requested.

For each class methods for the generics prune.rpart, print, summary and predict are available for inspection of the results and prediction, for example: print.classbagg, summary.classbagg, predict.classbagg and prune.classbagg for classification problems.

References

Leo Breiman (1996a), Bagging Predictors. *Machine Learning* **24**(2), 123–140.

Leo Breiman (1996b), Out-Of-Bag Estimation. *Technical Report* http://www.stat.berkeley.edu/~breiman/OOBestimation.pdf.

Leo Breiman (1998), Arcing Classifiers. *The Annals of Statistics* **26**(3), 801–824.

Peter Buehlmann and Bin Yu (2002), Analyzing Bagging. *The Annals of Statistics* **30**(4), 927–961.

Torsten Hothorn and Berthold Lausen (2003), Double-Bagging: Combining classifiers by bootstrap aggregation. *Pattern Recognition*, **36**(6), 1303–1309.

Torsten Hothorn and Berthold Lausen (2005), Bundling Classifiers by Bagging Trees. *Computational Statistics & Data Analysis*, 49, 1068–1078.

Torsten Hothorn, Berthold Lausen, Axel Benner and Martin Radespiel-Troeger (2004), Bagging Survival Trees. *Statistics in Medicine*, **23**(1), 77–91.

Examples

```
library("MASS")
library("survival")

# Classification: Breast Cancer data

data("BreastCancer", package = "mlbench")

# Test set error bagging (nbagg = 50): 3.7% (Breiman, 1998, Table 5)

mod <- bagging(Class ~ Cl.thickness + Cell.size
                + Cell.shape + Marg.adhesion
                + Epith.c.size + Bare.nuclei
                + Bl.cromatin + Normal.nucleoli
                + Mitoses, data=BreastCancer, coob=TRUE)
print(mod)

# Test set error bagging (nbagg=50): 7.9% (Breiman, 1996a, Table 2)
data("Ionosphere", package = "mlbench")
Ionosphere$V2 <- NULL # constant within groups

bagging(Class ~ ., data=Ionosphere, coob=TRUE)

# Double-Bagging: combine LDA and classification trees

# predict returns the linear discriminant values, i.e. linear combinations
# of the original predictors

comb.lda <- list(list(model=lda, predict=function(obj, newdata)
                                 predict(obj, newdata)$x))

# Note: out-of-bag estimator is not available in this situation, use
# errorest

mod <- bagging(Class ~ ., data=Ionosphere, comb=comb.lda)

predict(mod, Ionosphere[1:10,])

# Regression:

data("BostonHousing", package = "mlbench")

# Test set error (nbagg=25, trees pruned): 3.41 (Breiman, 1996a, Table 8)

mod <- bagging(medv ~ ., data=BostonHousing, coob=TRUE)
print(mod)

library("mlbench")
learn <- as.data.frame(mlbench.friedman1(200))

# Test set error (nbagg=25, trees pruned): 2.47 (Breiman, 1996a, Table 8)
```

```
mod <- bagging(y ~ ., data=learn, coob=TRUE)
print(mod)

# Survival data

# Brier score for censored data estimated by
# 10 times 10-fold cross-validation: 0.2 (Hothorn et al,
# 2002)

data("DLBCL", package = "ipred")
mod <- bagging(Surv(time,cens) ~ MGEc.1 + MGEc.2 + MGEc.3 + MGEc.4 + MGEc.5 +
                                 MGEc.6 + MGEc.7 + MGEc.8 + MGEc.9 +
                                 MGEc.10 + IPI, data=DLBCL, coob=TRUE)

print(mod)
```

---

bootest *Bootstrap Error Rate Estimators*

Description

Those functions are low-level functions used by `errorest` and are normally not called by users.

Usage

```
## S3 method for class 'factor'
bootest(y, formula, data, model, predict, nboot=25,
bc632plus=FALSE, list.tindx = NULL, predictions = FALSE,
both.boot = FALSE, ...)
```

Arguments

`y`	the response variable, either of class `factor` (classification), `numeric` (regression) or `Surv` (survival).
`formula`	a formula object.
`data`	data frame of predictors and response described in `formula`.
`model`	a function implementing the predictive model to be evaluated. The function `model` can either return an object representing a fitted model or a function with argument `newdata` which returns predicted values. In this case, the `predict` argument to `errorest` is ignored.
`predict`	a function with arguments `object` and `newdata` only which predicts the status of the observations in `newdata` based on the fitted model in `object`.
`nboot`	number of bootstrap replications to be used.

bc632plus	logical. Should the bias corrected version of misclassification error be computed?
predictions	logical, return a matrix of predictions. The ith column contains predictions of the ith out-of-bootstrap sample and 'NA's corresponding to the ith bootstrap sample.
list.tindx	list of numeric vectors, indicating which observations are included in each bootstrap sample.
both.boot	logical, return both (bootstrap and 632plus) estimations or only one of them.
...	additional arguments to model.

Details

See errorest.

`control.errorest` *Control Error Rate Estimators*

Description

Some parameters that control the behaviour of errorest.

Usage

```
control.errorest(k = 10, nboot = 25, strat = FALSE, random = TRUE,
                 predictions = FALSE, getmodels=FALSE, list.tindx = NULL)
```

Arguments

k	integer, specify k for k-fold cross-validation.
nboot	integer, number of bootstrap replications.
strat	logical, if TRUE, cross-validation is performed using stratified sampling (for classification problems).
random	logical, if TRUE, cross-validation is performed using a random ordering of the data.
predictions	logical, indicates whether the prediction for each observation should be returned or not (classification and regression only). For a bootstrap based estimator a matrix of size 'number of observations' times nboot is returned with predicted values of the ith out-of-bootstrap sample in column i and 'NA's for those observations not included in the ith out-of-bootstrap sample.
getmodels	logical, indicates a list of all models should be returned. For cross-validation only.
list.tindx	list of numeric vectors, indicating which observations are included in each bootstrap or cross-validation sample, respectively.

Value

A list with the same components as arguments.

cv *Cross-validated Error Rate Estimators.*

Description

Those functions are low-level functions used by `errorest` and are normally not called by users.

Usage

```
## S3 method for class 'factor'
cv(y, formula, data, model, predict, k=10, random=TRUE,
            strat=FALSE,
            predictions=NULL, getmodels=NULL, list.tindx = NULL, ...)
```

Arguments

`y`	response variable, either of class `factor` (classification), `numeric` (regression) or `Surv` (survival).
`formula`	a formula object.
`data`	data frame of predictors and response described in `formula`.
`model`	a function implementing the predictive model to be evaluated. The function `model` can either return an object representing a fitted model or a function with argument `newdata` which returns predicted values. In this case, the `predict` argument to `errorest` is ignored.
`predict`	a function with arguments `object` and `newdata` only which predicts the status of the observations in `newdata` based on the fitted model in `object`.
`k`	k-fold cross-validation.
`random`	logical, indicates whether a random order or the given order of the data should be used for sample splitting or not, defaults to `TRUE`.
`strat`	logical, stratified sampling or not, defaults to `FALSE`.
`predictions`	logical, return the prediction of each observation.
`getmodels`	logical, return a list of models for each fold.
`list.tindx`	list of numeric vectors, indicating which observations are included in each cross-validation sample.
`...`	additional arguments to `model`.

Details

See `errorest`.

DLBCL *Diffuse Large B-Cell Lymphoma*

Description

A data frame with gene expression data from diffuse large B-cell lymphoma (DLBCL) patients.

Usage

```
data("DLBCL")
```

Format

This data frame contains the following columns:

DLCL.Sample DLBCL identifier.

Gene.Expression Gene expression group.

time survival time in month.

cens censoring: 0 censored, 1 dead.

IPI International prognostic index.

MGEc.1 mean gene expression in cluster 1.

MGEc.2 mean gene expression in cluster 2.

MGEc.3 mean gene expression in cluster 3.

MGEc.4 mean gene expression in cluster 4.

MGEc.5 mean gene expression in cluster 5.

MGEc.6 mean gene expression in cluster 6.

MGEc.7 mean gene expression in cluster 7.

MGEc.8 mean gene expression in cluster 8.

MGEc.9 mean gene expression in cluster 9.

MGEc.10 mean gene expression in cluster 10.

Source

Except of `MGE`, the data is published at http://llmpp.nih.gov/lymphoma/data.shtml. `MGEc.*` is the mean of the gene expression in each of ten clusters derived by agglomerative average linkage hierarchical cluster analysis (Hothorn et al., 2002).

References

Ash A. Alizadeh et. al (2000), Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, **403**, 504–509.

Torsten Hothorn, Berthold Lausen, Axel Benner and Martin Radespiel-Troeger (2004), Bagging Survival Trees. *Statistics in Medicine*, **23**, 77–91.

Examples

```
set.seed(290875)

data("DLBCL", package="ipred")
library("survival")
survfit(Surv(time, cens) ~ 1, data=DLBCL)
```

---

dystrophy *Detection of muscular dystrophy carriers.*

Description

The dystrophy data frame has 209 rows and 10 columns.

Usage

```
data(dystrophy)
```

Format

This data frame contains the following columns:

OBS numeric. Observation number.

HospID numeric. Hospital ID number.

AGE numeric, age in years.

M numeric. Month of examination.

Y numeric. Year of examination.

CK numeric. Serum marker creatine kinase.

H numeric. Serum marker hemopexin.

PK numeric. Serum marker pyruvate kinase.

LD numeric. Serum marker lactate dehydroginase.

Class factor with levels, carrier and normal.

Details

Duchenne Muscular Dystrophy (DMD) is a genetically transmitted disease, passed from a mother to her children. Affected female offspring usually suffer no apparent symptoms, male offspring with the disease die at young age. Although female carriers have no physical symptoms they tend to exhibit elevated levels of certain serum enzymes or proteins.
The dystrophy dataset contains 209 observations of 75 female DMD carriers and 134 female DMD non-carrier. It includes 6 variables describing age of the female and the serum parameters serum marker creatine kinase (CK), serum marker hemopexin (H), serum marker pyruvate kinase (PK) and serum marker lactate dehydroginase (LD). The serum markers CK and H may be measured rather inexpensive from frozen serum, PK and LD requires fresh serum.

Source

D.Andrews and A. Herzberg (1985), Data. Berlin: Springer-Verlag.

References

Robert Tibshirani and Geoffry Hinton (1998), Coaching variables for regression and classification. Statistics and Computing 8, 25-33.

Examples

```
## Not run:

data("dystrophy")
library("rpart")
errorest(Class~CK+H~AGE+PK+LD, data = dystrophy, model = inbagg,
pFUN = list(list(model = lm, predict = mypredict.lm), list(model = rpart)),
ns = 0.75, estimator = "cv")

## End(Not run)
```

`errorest` *Estimators of Prediction Error*

Description

Resampling based estimates of prediction error: misclassification error, root mean squared error or Brier score for survival data.

Usage

```
## S3 method for class 'data.frame'
errorest(formula, data, subset, na.action=na.omit,
         model=NULL, predict=NULL,
         estimator=c("cv", "boot", "632plus"),
         est.para=control.errorest(), ...)
```

Arguments

`formula`	a formula of the form `lhs ~ rhs`. Either describing the model of explanatory and response variables in the usual way (see `lm`) or the model between explanatory and intermediate variables in the framework of indirect classification, see `inclass`.
`data`	a data frame containing the variables in the model formula and additionally the class membership variable if `model = inclass`. `data` is required for indirect classification, otherwise `formula` is evaluated in the calling environment.
`subset`	optional vector, specifying a subset of observations to be used.

na.action	function which indicates what should happen when the data contains NA's, defaults to na.omit.
model	function. Modelling technique whose error rate is to be estimated. The function model can either return an object representing a fitted model or a function with argument newdata which returns predicted values. In this case, the predict argument to errorest is ignored.
predict	function. Prediction method to be used. The vector of predicted values must have the same length as the the number of to-be-predicted observations. Predictions corresponding to missing data must be replaced by NA. Additionally, predict has to return predicted values comparable to the responses (that is: factors for classification problems). See the example on how to make this sure for any predictor.
estimator	estimator of the misclassification error: cv cross-validation, boot bootstrap or 632plus bias corrected bootstrap (classification only).
est.para	a list of additional parameters that control the calculation of the estimator, see control.errorest for details.
...	additional parameters to model.

Details

The prediction error for classification and regression models as well as predictive models for censored data using cross-validation or the bootstrap can be computed by errorest. For classification problems, the estimated misclassification error is returned. The root mean squared error is computed for regression problems and the Brier score for censored data (Graf et al., 1999) is reported if the response is censored.

Any model can be specified as long as it is a function with arguments model(formula, data, subset, na.action, ...). If a method predict.model(object, newdata, ...) is available, predict does not need to be specified. However, predict has to return predicted values in the same order and of the same length corresponding to the response. See the examples below.

k-fold cross-validation and the usual bootstrap estimator with est.para$nboot bootstrap replications can be computed for all kind of problems. The bias corrected .632+ bootstrap by Efron and Tibshirani (1997) is available for classification problems only. Use control.errorest to specify additional arguments.

errorest is a formula based interface to the generic functions cv or bootest which implement methods for classification, regression and survival problems.

Value

The class of the object returned depends on the class of the response variable and the estimator used. In each case, it is a list with an element error and additional information. print methods are available for the inspection of the results.

References

Brian D. Ripley (1996), *Pattern Recognition and Neural Networks*. Cambridge: Cambridge University Press.

Bradley Efron and Robert Tibshirani (1997), Improvements on Cross-Validation: The .632+ Bootstrap Estimator. *Journal of the American Statistical Association* **92**(438), 548–560.

Erika Graf, Claudia Schmoor, Willi Sauerbrei and Martin Schumacher (1999), Assessment and comparison of prognostic classification schemes for survival data. *Statistics in Medicine* **18**(17-18), 2529–2545.

Rosa A. Schiavo and David J. Hand (2000), Ten More Years of Error Rate Research. *International Statistical Review* **68**(3), 296-310.

David J. Hand, Hua Gui Li, Niall M. Adams (2001), Supervised Classification with Structured Class Definitions. *Computational Statistics & Data Analysis* **36**, 209–225.

Examples

```
# Classification

data("iris")
library("MASS")

# force predict to return class labels only
mypredict.lda <- function(object, newdata)
  predict(object, newdata = newdata)$class

# 10-fold cv of LDA for Iris data
errorest(Species ~ ., data=iris, model=lda,
         estimator = "cv", predict= mypredict.lda)

data("PimaIndiansDiabetes", package = "mlbench")
## Not run:
# 632+ bootstrap of LDA for Diabetes data
errorest(diabetes ~ ., data=PimaIndiansDiabetes, model=lda,
         estimator = "632plus", predict= mypredict.lda)

## End(Not run)

#cv of a fixed partition of the data
list.tindx <- list(1:100, 101:200, 201:300, 301:400, 401:500,
        501:600, 601:700, 701:768)

errorest(diabetes ~ ., data=PimaIndiansDiabetes, model=lda,
          estimator = "cv", predict = mypredict.lda,
          est.para = control.errorest(list.tindx = list.tindx))

## Not run:
#both bootstrap estimations based on fixed partitions

list.tindx <- vector(mode = "list", length = 25)
for(i in 1:25) {
  list.tindx[[i]] <- sample(1:768, 768, TRUE)
}

errorest(diabetes ~ ., data=PimaIndiansDiabetes, model=lda,
```

```
          estimator = c("boot", "632plus"), predict= mypredict.lda,
          est.para = control.errorest(list.tindx = list.tindx))


## End(Not run)
data("Glass", package = "mlbench")

# LDA has cross-validated misclassification error of
# 38% (Ripley, 1996, page 98)

# Pruned trees about 32% (Ripley, 1996, page 230)

# use stratified sampling here, i.e. preserve the class proportions
errorest(Type ~ ., data=Glass, model=lda,
         predict=mypredict.lda, est.para=control.errorest(strat=TRUE))

# force predict to return class labels
mypredict.rpart <- function(object, newdata)
  predict(object, newdata = newdata,type="class")

library("rpart")
pruneit <- function(formula, ...)
  prune(rpart(formula, ...), cp =0.01)

errorest(Type ~ ., data=Glass, model=pruneit,
         predict=mypredict.rpart, est.para=control.errorest(strat=TRUE))

# compute sensitivity and specifity for stabilised LDA

data("GlaucomaM", package = "TH.data")

error <- errorest(Class ~ ., data=GlaucomaM, model=slda,
  predict=mypredict.lda, est.para=control.errorest(predictions=TRUE))

# sensitivity

mean(error$predictions[GlaucomaM$Class == "glaucoma"] == "glaucoma")

# specifity

mean(error$predictions[GlaucomaM$Class == "normal"] == "normal")

# Indirect Classification: Smoking data

data(Smoking)
# Set three groups of variables:
# 1) explanatory variables are: TarY, NicY, COY, Sex, Age
# 2) intermediate variables are: TVPS, BPNL, COHB
# 3) response (resp) is defined by:

resp <- function(data){
  data <- data[, c("TVPS", "BPNL", "COHB")]
  res <- t(t(data) > c(4438, 232.5, 58))
```

```
  res <- as.factor(ifelse(apply(res, 1, sum) > 2, 1, 0))
  res
}

response <- resp(Smoking[ ,c("TVPS", "BPNL", "COHB")])
smoking <- cbind(Smoking, response)

formula <- response~TVPS+BPNL+COHB~TarY+NicY+COY+Sex+Age

# Estimation per leave-one-out estimate for the misclassification is
# 36.36% (Hand et al., 2001), using indirect classification with
# linear models
## Not run:
errorest(formula, data = smoking, model = inclass,estimator = "cv",
         pFUN = list(list(model=lm, predict = mypredict.lm)), cFUN = resp,
         est.para=control.errorest(k=nrow(smoking)))

## End(Not run)

# Regression

data("BostonHousing", package = "mlbench")

# 10-fold cv of lm for Boston Housing data
errorest(medv ~ ., data=BostonHousing, model=lm,
         est.para=control.errorest(random=FALSE))

# the same, with "model" returning a function for prediction
# instead of an object of class "lm"

mylm <- function(formula, data) {
  mod <- lm(formula, data)
  function(newdata) predict(mod, newdata)
}

errorest(medv ~ ., data=BostonHousing, model=mylm,
est.para=control.errorest(random=FALSE))


# Survival data

data("GBSG2", package = "TH.data")
library("survival")

# prediction is fitted Kaplan-Meier
predict.survfit <- function(object, newdata) object

# 5-fold cv of Kaplan-Meier for GBSG2 study
errorest(Surv(time, cens) ~ 1, data=GBSG2, model=survfit,
         predict=predict.survfit, est.para=control.errorest(k=5))
```

GlaucomaMVF *Glaucoma Database*

Description

The GlaucomaMVF data has 170 observations in two classes. 66 predictors are derived from a confocal laser scanning image of the optic nerve head, from a visual field test, a fundus photography and a measurement of the intra occular pressure.

Usage

```
data("GlaucomaMVF")
```

Format

This data frame contains the following predictors describing the morphology of the optic nerve head, the visual field, the intra occular pressure and a membership variable:

ag area global.

at area temporal.

as area superior.

an area nasal.

ai area inferior.

eag effective area global.

eat effective area temporal.

eas effective area superior.

ean effective area nasal.

eai effective area inferior.

abrg area below reference global.

abrt area below reference temporal.

abrs area below reference superior.

abrn area below reference nasal.

abri area below reference inferior.

hic height in contour.

mhcg mean height contour global.

mhct mean height contour temporal.

mhcs mean height contour superior.

mhcn mean height contour nasal.

mhci mean height contour inferior.

phcg peak height contour.

phct peak height contour temporal.

phcs peak height contour superior.

phcn peak height contour nasal.

phci peak height contour inferior.

hvc height variation contour.

vbsg volume below surface global.

vbst volume below surface temporal.

vbss volume below surface superior.

vbsn volume below surface nasal.

vbsi volume below surface inferior.

vasg volume above surface global.

vast volume above surface temporal.

vass volume above surface superior.

vasn volume above surface nasal.

vasi volume above surface inferior.

vbrg volume below reference global.

vbrt volume below reference temporal.

vbrs volume below reference superior.

vbrn volume below reference nasal.

vbri volume below reference inferior.

varg volume above reference global.

vart volume above reference temporal.

vars volume above reference superior.

varn volume above reference nasal.

vari volume above reference inferior.

mdg mean depth global.

mdt mean depth temporal.

mds mean depth superior.

mdn mean depth nasal.

mdi mean depth inferior.

tmg third moment global.

tmt third moment temporal.

tms third moment superior.

tmn third moment nasal.

tmi third moment inferior.

mr mean radius.

rnf retinal nerve fiber thickness.

mdic mean depth in contour.

emd effective mean depth.

mv mean variability.

tension intra occular pressure.

clv corrected loss variance, variability of the visual field.

cs contrast sensitivity of the visual field.

lora loss of rim area, measured by fundus photography.

Class a factor with levels `glaucoma` and `normal`.

Details

Confocal laser images of the eye background are taken with the Heidelberg Retina Tomograph and variables 1-62 are derived. Most of these variables describe either the area or volume in certain parts of the papilla and are measured in four sectors (temporal, superior, nasal and inferior) as well as for the whole papilla (global). The global measurement is, roughly, the sum of the measurements taken in the four sector.

The perimeter 'Octopus' measures the visual field variables `clv` and `cs`, stereo optic disks photographs were taken with a telecentric fundus camera and `lora` is derived.

Observations of both groups are matched by age and sex, to prevent for possible confounding.

Note

`GLaucomMVF` overlaps in some parts with `GlaucomaM`.

Source

Andrea Peters, Berthold Lausen, Georg Michelson and Olaf Gefeller (2003), Diagnosis of glaucoma by indirect classifiers. *Methods of Information in Medicine* **1**, 99-103.

Examples

```
## Not run:

data("GlaucomaMVF", package = "ipred")
library("rpart")

response <- function (data) {
  attach(data)
  res <- ifelse((!is.na(clv) & !is.na(lora) & clv >= 5.1 & lora >=
        49.23372) | (!is.na(clv) & !is.na(lora) & !is.na(cs) &
        clv < 5.1 & lora >= 58.55409 & cs < 1.405) | (is.na(clv) &
        !is.na(lora) & !is.na(cs) & lora >= 58.55409 & cs < 1.405) |
        (!is.na(clv) & is.na(lora) & cs < 1.405), 0, 1)
  detach(data)
  factor (res, labels = c("glaucoma", "normal"))
}

errorest(Class~clv+lora+cs~., data = GlaucomaMVF, model=inclass,
       estimator="cv", pFUN = list(list(model = rpart)), cFUN = response)
```

```
## End(Not run)
```

---

inbagg *Indirect Bagging*

Description

Function to perform the indirect bagging and subagging.

Usage

```
## S3 method for class 'data.frame'
inbagg(formula, data, pFUN=NULL,
  cFUN=list(model = NULL, predict = NULL, training.set = NULL),
  nbagg = 25, ns = 0.5, replace = FALSE, ...)
```

Arguments

formula	formula. A formula specified as y~w1+w2+w3~x1+x2+x3 describes how to model the intermediate variables w1, w2, w3 and the response variable y, if no other formula is specified by the elements of pFUN or in cFUN
data	data frame of explanatory, intermediate and response variables.
pFUN	list of lists, which describe models for the intermediate variables, details are given below.
cFUN	either a fixed function with argument newdata and returning the class membership by default, or a list specifying a classifying model, similar to one element of pFUN. Details are given below.
nbagg	number of bootstrap samples.
ns	proportion of sample to be drawn from the learning sample. By default, subagging with 50% is performed, i.e. draw 0.5*n out of n without replacement.
replace	logical. Draw with or without replacement.
...	additional arguments (e.g. subset).

Details

A given data set is subdivided into three types of variables: explanatory, intermediate and response variables.

Here, each specified intermediate variable is modelled separately following pFUN, a list of lists with elements specifying an arbitrary number of models for the intermediate variables and an optional element training.set = c("oob", "bag", "all"). The element training.set determines whether, predictive models for the intermediate are calculated based on the out-of-bag sample ("oob"), the default, on the bag sample ("bag") or on all available observations ("all"). The elements of pFUN, specifying the models for the intermediate variables are lists as described in

inclass. Note that, if no formula is given in these elements, the functional relationship of formula is used.

The response variable is modelled following cFUN. This can either be a fixed classifying function as described in Peters et al. (2003) or a list, which specifies the modelling technique to be applied. The list contains the arguments model (which model to be fitted), predict (optional, how to predict), formula (optional, of type y~w1+w2+w3+x1+x2 determines the variables the classifying function is based on) and the optional argument training.set = c("fitted.bag", "original", "fitted.subset") specifying whether the classifying function is trained on the predicted observations of the bag sample ("fitted.bag"), on the original observations ("original") or on the predicted observations not included in a defined subset ("fitted.subset"). Per default the formula specified in formula determines the variables, the classifying function is based on.

Note that the default of cFUN = list(model = NULL, training.set = "fitted.bag") uses the function rpart and the predict function predict(object, newdata, type = "class").

Value

An object of class "inbagg", that is a list with elements

mtrees	a list of length nbagg, describing the prediction models corresponding to each bootstrap sample. Each element of mtrees is a list with elements bindx (observations of bag sample), btree (classifying function of bag sample) and bfct (predictive models for intermediates of bag sample).
y	vector of response values.
W	data frame of intermediate variables.
X	data frame of explanatory variables.

References

David J. Hand, Hua Gui Li, Niall M. Adams (2001), Supervised classification with structured class definitions. *Computational Statistics & Data Analysis* **36**, 209–225.

Andrea Peters, Berthold Lausen, Georg Michelson and Olaf Gefeller (2003), Diagnosis of glaucoma by indirect classifiers. *Methods of Information in Medicine* **1**, 99-103.

See Also

rpart, bagging, lm

Examples

```
library("MASS")
library("rpart")
y <- as.factor(sample(1:2, 100, replace = TRUE))
W <- mvrnorm(n = 200, mu = rep(0, 3), Sigma = diag(3))
X <- mvrnorm(n = 200, mu = rep(2, 3), Sigma = diag(3))
colnames(W) <- c("w1", "w2", "w3")
```

```
colnames(X) <- c("x1", "x2", "x3")
DATA <- data.frame(y, W, X)


pFUN <- list(list(formula = w1~x1+x2, model = lm, predict = mypredict.lm),
list(model = rpart))

inbagg(y~w1+w2+w3~x1+x2+x3, data = DATA, pFUN = pFUN)
```

`inclass` *Indirect Classification*

Description

A framework for the indirect classification approach.

Usage

```
## S3 method for class 'data.frame'
inclass(formula, data, pFUN = NULL, cFUN = NULL, ...)
```

Arguments

`formula`	formula. A `formula` specified as `y~w1+w2+w3~x1+x2+x3` models each intermediate variable `w1, w2, w3` by `wi~x1+x2+x3` and the response by `y~w1+w2+w3` if no other formulas are given in `pFUN` or `cFUN`.
`data`	data frame of explanatory, intermediate and response variables.
`pFUN`	list of lists, which describe models for the intermediate variables, see below for details.
`cFUN`	either a function or a list which describes the model for the response variable. The function has the argument `newdata` only.
`...`	additional arguments, passed to model fitting of the response variable.

Details

A given data set is subdivided into three types of variables: those to be used predicting the class (explanatory variables) those to be used defining the class (intermediate variables) and the class membership variable itself (response variable). Intermediate variables are modelled based on the explanatory variables, the class membership variable is defined on the intermediate variables.

Each specified intermediate variable is modelled separately following `pFUN` and a formula specified by `formula`. `pFUN` is a list of lists, the maximum length of `pFUN` is the number of intermediate variables. Each element of `pFUN` is a list with elements:
`model` - a function with arguments `formula` and `data`;
`predict` - an optional function with arguments `object, newdata` only, if `predict` is not specified, the predict method of `model` is used;

`formula` - specifies the formula for the corresponding `model` (optional), the formula described in `y~w1+w2+w3~x1+x2+x3` is used if no other is specified.

The response is classified following `cFUN`, which is either a fixed function or a list as described below. The determined function `cFUN` assigns the intermediate (and explanatory) variables to a certain class membership, the list `cFUN` has the elements `formula, model, predict` and `training.set`. The elements `formula, model, predict` are structured as described by `pFUN`, the described model is trained on the original (intermediate variables) if `training.set="original"` or if `training.set = NULL`, on the fitted values if `training.set = "fitted"` or on observations not included in a specified subset if `training.set = "subset"`.

A list of prediction models corresponding to each intermediate variable, a predictive function for the response, a list of specifications for the intermediate and for the response are returned.
For a detailed description on indirect classification see Hand et al. (2001).

Value

An object of class `inclass`, consisting of a list of

`model.intermediate`
: list of fitted models for each intermediate variable.

`model.response`
: predictive model for the response variable.

`para.intermediate`
: list, where each element is again a list and specifies the model for each intermediate variable.

`para.response`
: a list which specifies the model for response variable.

References

David J. Hand, Hua Gui Li, Niall M. Adams (2001), Supervised classification with structured class definitions. *Computational Statistics & Data Analysis* **36**, 209–225.

Andrea Peters, Berthold Lausen, Georg Michelson and Olaf Gefeller (2003), Diagnosis of glaucoma by indirect classifiers. *Methods of Information in Medicine* **1**, 99-103.

See Also

`bagging`, `inclass`

Examples

```
data("Smoking", package = "ipred")
# Set three groups of variables:
# 1) explanatory variables are: TarY, NicY, COY, Sex, Age
# 2) intermediate variables are: TVPS, BPNL, COHB
# 3) response (resp) is defined by:

classify <- function(data){
  data <- data[,c("TVPS", "BPNL", "COHB")]
  res <- t(t(data) > c(4438, 232.5, 58))
```

```
  res <- as.factor(ifelse(apply(res, 1, sum) > 2, 1, 0))
  res
}

response <- classify(Smoking[ ,c("TVPS", "BPNL", "COHB")])
smoking <- data.frame(Smoking, response)

formula <- response~TVPS+BPNL+COHB~TarY+NicY+COY+Sex+Age

inclass(formula, data = smoking, pFUN = list(list(model = lm, predict =
mypredict.lm)), cFUN = classify)
```

---

ipredknn *k-Nearest Neighbour Classification*

Description

k-nearest neighbour classification with an interface compatible to bagging and errorest.

Usage

```
ipredknn(formula, data, subset, na.action, k=5, ...)
```

Arguments

formula	a formula of the form lhs ~ rhs where lhs is the response variable and rhs a set of predictors.
data	optional data frame containing the variables in the model formula.
subset	optional vector specifying a subset of observations to be used.
na.action	function which indicates what should happen when the data contain NAs.
k	number of neighbours considered, defaults to 5.
...	additional parameters.

Details

This is a wrapper to knn in order to be able to use k-NN in bagging and errorest.

Value

An object of class ipredknn. See predict.ipredknn.

Examples

```
library("mlbench")
learn <- as.data.frame(mlbench.twonorm(300))

mypredict.knn <- function(object, newdata)
                   predict.ipredknn(object, newdata, type="class")

errorest(classes ~., data=learn, model=ipredknn,
         predict=mypredict.knn)
```

---

kfoldcv *Subsamples for k-fold Cross-Validation*

Description

Computes feasible sample sizes for the k groups in k-fold cv if N/k is not an integer.

Usage

```
kfoldcv(k, N, nlevel=NULL)
```

Arguments

k	number of groups.
N	total sample size.
nlevel	a vector of sample sizes for stratified sampling.

Details

If N/k is not an integer, k-fold cv is not unique. Determine meaningful sample sizes.

Value

A vector of length k.

Examples

```
# 10-fold CV with N = 91

kfoldcv(10, 91)
```

mypredict.lm *Predictions Based on Linear Models*

Description

Function to predict a vector of full length (number of observations), where predictions according to missing explanatory values are replaced by NA.

Usage

```
mypredict.lm(object, newdata)
```

Arguments

object	an object of class lm.
newdata	matrix or data frame to be predicted according to object.

Value

Vector of predicted values.

Note

predict.lm delivers a vector of reduced length, i.e. rows where explanatory variables are missing are omitted. The full length of the predicted observation vector is necessary in the indirect classification approach (predict.inclass).

predict.classbagg *Predictions from Bagging Trees*

Description

Predict the outcome of a new observation based on multiple trees.

Usage

```
## S3 method for class 'classbagg'
predict(object, newdata=NULL, type=c("class", "prob"),
                          aggregation=c("majority", "average", "weighted"), ...)
## S3 method for class 'regbagg'
predict(object, newdata=NULL, aggregation=c("average",
                "weighted"), ...)
## S3 method for class 'survbagg'
predict(object, newdata=NULL,...)
```

Arguments

object	object of classes classbagg, regbagg or survbagg.
newdata	a data frame of new observations.
type	character string denoting the type of predicted value returned for classification trees. Either class (predicted classes are returned) or prob (estimated class probabilities are returned).
aggregation	character string specifying how to aggregate, see below.
...	additional arguments, currently not passed to any function.

Details

There are (at least) three different ways to aggregate the predictions of bagging classification trees. Most famous is class majority voting (aggregation="majority") where the most frequent class is returned. The second way is choosing the class with maximal averaged class probability (aggregation="average"). The third method is based on the "aggregated learning sample", introduced by Hothorn et al. (2003) for survival trees. The prediction of a new observation is the majority class, mean or Kaplan-Meier curve of all observations from the learning sample identified by the nbagg leaves containing the new observation. For regression trees, only averaged or weighted predictions are possible.

By default, the out-of-bag estimate is computed if newdata is NOT specified. Therefore, the predictions of predict(object) are "honest" in some way (this is not possible for combined models via comb in bagging). If you like to compute the predictions for the learning sample itself, use newdata to specify your data.

Value

The predicted class or estimated class probabilities are returned for classification trees. The predicted endpoint is returned in regression problems and the predicted Kaplan-Meier curve is returned for survival trees.

References

Leo Breiman (1996), Bagging Predictors. *Machine Learning* **24**(2), 123–140.

Torsten Hothorn, Berthold Lausen, Axel Benner and Martin Radespiel-Troeger (2004), Bagging Survival Trees. *Statistics in Medicine*, **23**(1), 77–91.

Examples

```
data("Ionosphere", package = "mlbench")
Ionosphere$V2 <- NULL # constant within groups

# nbagg = 10 for performance reasons here
mod <- bagging(Class ~ ., data=Ionosphere)

# out-of-bag estimate

mean(predict(mod) != Ionosphere$Class)
```

```
# predictions for the first 10 observations

predict(mod, newdata=Ionosphere[1:10,])

predict(mod, newdata=Ionosphere[1:10,], type="prob")
```

---

predict.inbagg *Predictions from an Inbagg Object*

Description

Predicts the class membership of new observations through indirect bagging.

Usage

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

Arguments

object	object of class inbagg, see inbagg.
newdata	data frame to be classified.
...	additional argumends corresponding to the predictive models.

Details

Predictions of class memberships are calculated. i.e. values of the intermediate variables are predicted following pFUN and classified following cFUN, see inbagg.

Value

The vector of predicted classes is returned.

References

David J. Hand, Hua Gui Li, Niall M. Adams (2001), Supervised classification with structured class definitions. *Computational Statistics & Data Analysis* **36**, 209–225.

Andrea Peters, Berthold Lausen, Georg Michelson and Olaf Gefeller (2003), Diagnosis of glaucoma by indirect classifiers. *Methods of Information in Medicine* **1**, 99-103.

See Also

inbagg

Examples

```
library("MASS")
library("rpart")
y <- as.factor(sample(1:2, 100, replace = TRUE))
W <- mvrnorm(n = 200, mu = rep(0, 3), Sigma = diag(3))
X <- mvrnorm(n = 200, mu = rep(2, 3), Sigma = diag(3))
colnames(W) <- c("w1", "w2", "w3")
colnames(X) <- c("x1", "x2", "x3")
DATA <- data.frame(y, W, X)

pFUN <- list(list(formula = w1~x1+x2, model = lm),
list(model = rpart))

RES <- inbagg(y~w1+w2+w3~x1+x2+x3, data = DATA, pFUN = pFUN)
predict(RES, newdata = X)
```

---

predict.inclass *Predictions from an Inclass Object*

Description

Predicts the class membership of new observations through indirect classification.

Usage

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

Arguments

object	object of class inclass, see inclass.
newdata	data frame to be classified.
...	additional arguments corresponding to the predictive models specified in inclass.

Details

Predictions of class memberships are calculated. i.e. values of the intermediate variables are predicted and classified following cFUN, see inclass.

Value

The vector of predicted classes is returned.

References

David J. Hand, Hua Gui Li, Niall M. Adams (2001), Supervised classification with structured class definitions. *Computational Statistics & Data Analysis* **36**, 209–225.

Andrea Peters, Berthold Lausen, Georg Michelson and Olaf Gefeller (2003), Diagnosis of glaucoma by indirect classifiers. *Methods of Information in Medicine* **1**, 99-103.

See Also

inclass

Examples

```
## Not run:
# Simulation model, classification rule following Hand et al. (2001)

theta90 <- varset(N = 1000, sigma = 0.1, theta = 90, threshold = 0)

dataset <- as.data.frame(cbind(theta90$explanatory, theta90$intermediate))
names(dataset) <- c(colnames(theta90$explanatory),
colnames(theta90$intermediate))

classify <- function(Y, threshold = 0) {
  Y <- Y[,c("y1", "y2")]
  z <- (Y > threshold)
  resp <- as.factor(ifelse((z[,1] + z[,2]) > 1, 1, 0))
  return(resp)
}

formula <- response~y1+y2~x1+x2

fit <- inclass(formula, data = dataset, pFUN = list(list(model = lm)),
 cFUN = classify)

predict(object = fit, newdata = dataset)


data("Smoking", package = "ipred")

# explanatory variables are: TarY, NicY, COY, Sex, Age
# intermediate variables are: TVPS, BPNL, COHB
# reponse is defined by:

classify <- function(data){
  data <- data[,c("TVPS", "BPNL", "COHB")]
  res <- t(t(data) > c(4438, 232.5, 58))
  res <- as.factor(ifelse(apply(res, 1, sum) > 2, 1, 0))
  res
}

response <- classify(Smoking[ ,c("TVPS", "BPNL", "COHB")])
smoking <- cbind(Smoking, response)
```

```
formula <- response~TVPS+BPNL+COHB~TarY+NicY+COY+Sex+Age

fit <- inclass(formula, data = smoking,
  pFUN = list(list(model = lm)), cFUN = classify)


predict(object = fit, newdata = smoking)

## End(Not run)

data("GlaucomaMVF", package = "ipred")
library("rpart")
glaucoma <- GlaucomaMVF[,(names(GlaucomaMVF) != "tension")]
# explanatory variables are derived by laser scanning image and intra occular pressure
# intermediate variables are: clv, cs, lora
# response is defined by

classify <- function (data) {
  attach(data)
  res <- ifelse((!is.na(clv) & !is.na(lora) & clv >= 5.1 & lora >=
        49.23372) | (!is.na(clv) & !is.na(lora) & !is.na(cs) &
        clv < 5.1 & lora >= 58.55409 & cs < 1.405) | (is.na(clv) &
        !is.na(lora) & !is.na(cs) & lora >= 58.55409 & cs < 1.405) |
        (!is.na(clv) & is.na(lora) & cs < 1.405), 0, 1)
  detach(data)
  factor (res, labels = c("glaucoma", "normal"))
}

fit <- inclass(Class~clv+lora+cs~., data = glaucoma,
             pFUN = list(list(model = rpart)), cFUN = classify)

data("GlaucomaM", package = "TH.data")
predict(object = fit, newdata = GlaucomaM)
```

`predict.ipredknn` *Predictions from k-Nearest Neighbors*

Description

Predict the class of a new observation based on k-NN.

Usage

```
## S3 method for class 'ipredknn'
predict(object, newdata, type=c("prob", "class"), ...)
```

Arguments

object	object of class ipredknn.
newdata	a data frame of new observations.
type	return either the predicted class or the the proportion of the votes for the winning class.
...	additional arguments.

Details

This function is a method for the generic function predict for class ipredknn. For the details see knn.

Value

Either the predicted class or the the proportion of the votes for the winning class.

---

predict.slda *Predictions from Stabilised Linear Discriminant Analysis*

Description

Predict the class of a new observation based on stabilised LDA.

Usage

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

Arguments

object	object of class slda.
newdata	a data frame of new observations.
...	additional arguments passed to predict.lda.

Details

This function is a method for the generic function predict for class slda. For the details see predict.lda.

Value

A list with components

class	the predicted class (a factor).
posterior	posterior probabilities for the classes.
x	the scores of test cases.

print.classbagg *Print Method for Bagging Trees*

Description

Print objects returned by bagging in nice layout.

Usage

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

Arguments

x	object returned by bagging.
digits	how many digits should be printed.
...	further arguments to be passed to or from methods.

Value

none

print.cvclass *Print Method for Error Rate Estimators*

Description

Print objects returned by errorest in nice layout.

Usage

```
## S3 method for class 'cvclass'
print(x, digits=4, ...)
```

Arguments

x	an object returned by errorest.
digits	how many digits should be printed.
...	further arguments to be passed to or from methods.

Value

none

print.inbagg *Print Method for Inbagg Object*

Description

Print object of class inbagg in nice layout.

Usage

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

Arguments

x	object of class inbagg.
...	additional arguments.

Details

An object of class inbagg is printed. Information about number and names of the intermediate variables, and the number of drawn bootstrap samples is given.

print.inclass *Print Method for Inclass Object*

Description

Print object of class inclass in nice layout.

Usage

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

Arguments

x	object of class inclass.
...	additional arguments.

Details

An object of class inclass is printed. Information about number and names of the intermediate variables, the used modelling technique and the number of drawn bootstrap samples is given.

prune.classbagg *Pruning for Bagging*

Description

Prune each of the trees returned by bagging.

Usage

```
## S3 method for class 'classbagg'
prune(tree, cp=0.01,...)
```

Arguments

tree	an object returned by bagging (calling this tree is needed by the generic function prune in package rpart).
cp	complexity parameter, see prune.rpart.
...	additional arguments to prune.rpart.

Details

By default, bagging grows classification trees of maximal size. One may want to prune each tree, however, it is not clear whether or not this may decrease prediction error.

Value

An object of the same class as tree with the trees pruned.

Examples

```
data("Glass", package = "mlbench")
library("rpart")

mod <- bagging(Type ~ ., data=Glass, nbagg=10, coob=TRUE)
pmod <- prune(mod)
print(pmod)
```

rsurv *Simulate Survival Data*

Description

Simulation Setup for Survival Data.

Usage

```
rsurv(N, model=c("A", "B", "C", "D", "tree"), gamma=NULL, fact=1, pnon=10,
      gethaz=FALSE)
```

Arguments

N	number of observations.
model	type of model.
gamma	simulate censoring time as runif(N, 0, gamma). Defaults to NULL (no censoring).
fact	scale parameter for model=tree.
pnon	number of additional non-informative variables for the tree model.
gethaz	logical, indicating wheather the hazard rate for each observation should be returned.

Details

Simulation setup similar to configurations used in LeBlanc and Crowley (1992) or Keles and Segal (2002) as well as a tree model used in Hothorn et al. (2004). See Hothorn et al. (2004) for the details.

Value

A data frame with elements time, cens, X1 ... X5. If pnon > 0, additional noninformative covariables are added. If gethaz=TRUE, the hazard attribute returns the hazard rates.

References

M. LeBlanc and J. Crowley (1992), Relative Risk Trees for Censored Survival Data. *Biometrics* **48**, 411–425.

S. Keles and M. R. Segal (2002), Residual-based tree-structured survival analysis. *Statistics in Medicine*, **21**, 313–326.

Torsten Hothorn, Berthold Lausen, Axel Benner and Martin Radespiel-Troeger (2004), Bagging Survival Trees. *Statistics in Medicine*, **23**(1), 77–91.

Examples

```
library("survival")
# 3*X1 + X2
simdat <- rsurv(500, model="C")
coxph(Surv(time, cens) ~ ., data=simdat)
```

`sbrier` *Model Fit for Survival Data*

Description

Model fit for survival data: the integrated Brier score for censored observations.

Usage

```
sbrier(obj, pred, btime= range(obj[,1]))
```

Arguments

`obj`	an object of class `Surv`.
`pred`	predicted values. Either a probability or a list of `survfit` objects.
`btime`	numeric vector of times, the integrated Brier score is computed if this is of `length > 1`. The Brier score at `btime` is returned otherwise.

Details

There is no obvious criterion of model fit for censored data. The Brier score for censoring as well as it's integrated version were suggested by Graf et al (1999).

The integrated Brier score is always computed over a subset of the interval given by the range of the time slot of the survival object `obj`.

Value

The (integrated) Brier score with attribute `time` is returned.

References

Erika Graf, Claudia Schmoor, Willi Sauerbrei and Martin Schumacher (1999), Assessment and comparison of prognostic classification schemes for survival data. *Statistics in Medicine* **18**(17-18), 2529–2545.

See Also

More measures for the validation of predicted surival probabilities are implemented in package `pec`.

Examples

```
library("survival")
data("DLBCL", package = "ipred")
smod <- Surv(DLBCL$time, DLBCL$cens)

KM <- survfit(smod ~ 1)
# integrated Brier score up to max(DLBCL$time)
sbrier(smod, KM)

# integrated Brier score up to time=50
sbrier(smod, KM, btime=c(0, 50))

# Brier score for time=50
sbrier(smod, KM, btime=50)

# a "real" model: one single survival tree with Intern. Prognostic Index
# and mean gene expression in the first cluster as predictors
mod <- bagging(Surv(time, cens) ~ MGEc.1 + IPI, data=DLBCL, nbagg=1)

# this is a list of survfit objects (==KM-curves), one for each observation
# in DLBCL
pred <- predict(mod, newdata=DLBCL)

# integrated Brier score up to max(time)
sbrier(smod, pred)

# Brier score at time=50
sbrier(smod, pred, btime=50)
# artificial examples and illustrations

cleans <- function(x) { attr(x, "time") <- NULL; names(x) <- NULL; x }

n <- 100
time <- rpois(n, 20)
cens <- rep(1, n)

# checks, Graf et al. page 2536, no censoring at all!
# no information: \pi(t) = 0.5

a <- sbrier(Surv(time, cens), rep(0.5, n), time[50])
stopifnot(all.equal(cleans(a),0.25))

# some information: \pi(t) = S(t)

n <- 100
time <- 1:100
mod <- survfit(Surv(time, cens) ~ 1)
a <- sbrier(Surv(time, cens), rep(list(mod), n))
mymin <- mod$surv * (1 - mod$surv)
cleans(a)
sum(mymin)/diff(range(time))
```

```
# independent of ordering
rand <- sample(1:100)
b <- sbrier(Surv(time, cens)[rand], rep(list(mod), n)[rand])
stopifnot(all.equal(cleans(a), cleans(b)))



# 2 groups at different risk

time <- c(1:10, 21:30)
strata <- c(rep(1, 10), rep(2, 10))
cens <- rep(1, length(time))

# no information about the groups

a <- sbrier(Surv(time, cens), survfit(Surv(time, cens) ~ 1))
b <- sbrier(Surv(time, cens), rep(list(survfit(Surv(time, cens) ~1)), 20))
stopifnot(all.equal(a, b))

# risk groups known

mod <- survfit(Surv(time, cens) ~ strata)
b <- sbrier(Surv(time, cens), c(rep(list(mod[1]), 10), rep(list(mod[2]), 10)))
stopifnot(a > b)

### GBSG2 data
data("GBSG2", package = "TH.data")

thsum <- function(x) {
  ret <- c(median(x), quantile(x, 0.25), quantile(x,0.75))
  names(ret)[1] <- "Median"
  ret
}

t(apply(GBSG2[,c("age", "tsize", "pnodes",
                 "progrec", "estrec")], 2, thsum))

table(GBSG2$menostat)
table(GBSG2$tgrade)
table(GBSG2$horTh)

# pooled Kaplan-Meier

mod <- survfit(Surv(time, cens) ~ 1, data=GBSG2)
# integrated Brier score
sbrier(Surv(GBSG2$time, GBSG2$cens), mod)
# Brier score at 5 years
sbrier(Surv(GBSG2$time, GBSG2$cens), mod, btime=1825)

# Nottingham prognostic index

GBSG2 <- GBSG2[order(GBSG2$time),]
```

```
NPI <- 0.2*GBSG2$tsize/10 + 1 + as.integer(GBSG2$tgrade)
NPI[NPI < 3.4] <- 1
NPI[NPI >= 3.4 & NPI <=5.4] <- 2
NPI[NPI > 5.4] <- 3

mod <- survfit(Surv(time, cens) ~ NPI, data=GBSG2)
plot(mod)

pred <- c()
survs <- c()
for (i in sort(unique(NPI)))
    survs <- c(survs, getsurv(mod[i], 1825))

for (i in 1:nrow(GBSG2))
   pred <- c(pred, survs[NPI[i]])

# Brier score of NPI at t=5 years
sbrier(Surv(GBSG2$time, GBSG2$cens), pred, btime=1825)
```

`slda` *Stabilised Linear Discriminant Analysis*

Description

Linear discriminant analysis based on left-spherically distributed linear scores.

Usage

```
## S3 method for class 'formula'
slda(formula, data, subset, na.action=na.rpart, ...)
## S3 method for class 'factor'
slda(y, X, q=NULL, ...)
```

Arguments

`y`	the response variable: a factor vector of class labels.
`X`	a data frame of predictor variables.
`q`	the number of positive eigenvalues the scores are derived from, see below.
`formula`	a formula of the form `lhs ~ rhs` where `lhs` is the response variable and `rhs` a set of predictors.
`data`	optional data frame containing the variables in the model formula.
`subset`	optional vector specifying a subset of observations to be used.
`na.action`	function which indicates what should happen when the data contain `NA`s. Defaults to `na.rpart`.
`...`	additional parameters passed to `lda`.

Details

This function implements the LDA for q-dimensional linear scores of the original p predictors derived from the PC_q rule by Laeuter et al. (1998). Based on the product sum matrix

$$W = (X - \bar{X})^\top (X - \bar{X})$$

the eigenvalue problem $WD = diag(W)DL$ is solved. The first q columns D_q of D are used as a weight matrix for the original p predictors: XD_q. By default, q is the number of eigenvalues greater one. The q-dimensional linear scores are left-spherically distributed and are used as predictors for a classical LDA.

This form of reduction of the dimensionality was developed for discriminant analysis problems by Laeuter (1992) and was used for multivariate tests by Laeuter et al. (1998), Kropf (2000) gives an overview. For details on left-spherically distributions see Fang and Zhang (1990).

Value

An object of class `slda`, a list with components

`scores`	the weight matrix.
`mylda`	an object of class `lda`.

References

Fang Kai-Tai and Zhang Yao-Ting (1990), *Generalized Multivariate Analysis*, Springer, Berlin.

Siegfried Kropf (2000), *Hochdimensionale multivariate Verfahren in der medizinischen Statistik*, Shaker Verlag, Aachen (in german).

Juergen Laeuter (1992), *Stabile multivariate Verfahren*, Akademie Verlag, Berlin (in german).

Juergen Laeuter, Ekkehard Glimm and Siegfried Kropf (1998), Multivariate Tests Based on Left-Spherically Distributed Linear Scores. *The Annals of Statistics*, **26**(5) 1972–1988.

See Also

`predict.slda`

Examples

```
library("mlbench")
library("MASS")
learn <- as.data.frame(mlbench.twonorm(100))
test <- as.data.frame(mlbench.twonorm(1000))

mlda <- lda(classes ~ ., data=learn)
mslda <- slda(classes ~ ., data=learn)

print(mean(predict(mlda, newdata=test)$class != test$classes))
print(mean(predict(mslda, newdata=test)$class != test$classes))
```

`Smoking` *Smoking Styles*

Description

The `Smoking` data frame has 55 rows and 9 columns.

Usage

```
data("Smoking")
```

Format

This data frame contains the following columns:

NR numeric, patient number.

Sex factor, sex of patient.

Age factor, age group of patient, grouping consisting of those in their twenties, those in their thirties and so on.

TarY numeric, tar yields of the cigarettes.

NicY numeric, nicotine yields of the cigarettes.

COY numeric, carbon monoxide (CO) yield of the cigarettes.

TVPS numeric, total volume puffed smoke.

BPNL numeric, blood plasma nicotine level.

COHB numeric, carboxyhaemoglobin level, i.e. amount of CO absorbed by the blood stream.

Details

The data describes different smoking habits of probands.

Source

Hand and Taylor (1987), Study F *Smoking Styles*.

References

D.J. Hand and C.C. Taylor (1987), *Multivariate analysis of variance and repeated measures.* London: Chapman \& Hall, pp. 167–181.

summary.classbagg *Summarising Bagging*

Description

summary method for objects returned by bagging.

Usage

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

Arguments

object	object returned by bagging.
...	further arguments to be passed to or from methods.

Details

A representation of all trees in the object is printed.

Value

none

summary.inbagg *Summarising Inbagg*

Description

Summary of inbagg is returned.

Usage

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

Arguments

object	an object of class inbagg.
...	additional arguments.

Details

A representation of an indirect bagging model (the intermediates variables, the number of bootstrap samples, the trees) is printed.

Value

none

See Also

print.summary.inbagg

`summary.inclass` *Summarising Inclass*

Description

Summary of inclass is returned.

Usage

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

Arguments

`object`	an object of class `inclass`.
`...`	additional arguments.

Details

A representation of an indirect classification model (the intermediates variables, which modelling technique is used and the prediction model) is printed.

Value

none

See Also

print.summary.inclass

`varset` *Simulation Model*

Description

Three sets of variables are calculated: explanatory, intermediate and response variables.

Usage

```
varset(N, sigma=0.1, theta=90, threshold=0, u=1:3)
```

Arguments

`N`	number of simulated observations.
`sigma`	standard deviation of the error term.
`theta`	angle between two u vectors.
`threshold`	cutpoint for classifying to 0 or 1.
`u`	starting values.

Details

For each observation values of two explanatory variables $x = (x_1, x_2)^\top$ and of two responses $y = (y_1, y_2)^\top$ are simulated, following the formula:

$$y = U * x + e = \begin{pmatrix} u_1^\top \\ u_2^\top \end{pmatrix} * x + e$$

where x is the evaluation of as standard normal random variable and e is generated by a normal variable with standard deviation `sigma`. U is a 2*2 Matrix, where

$$u_1 = \begin{pmatrix} u_{1,1} \\ u_{1,2} \end{pmatrix}, u_2 = \begin{pmatrix} u_{2,1} \\ u_{2,2} \end{pmatrix}, ||u_1|| = ||u_2|| = 1,$$

i.e. a matrix of two normalised vectors.

Value

A list containing the following arguments

`explanatory`	N*2 matrix of 2 explanatory variables.
`intermediate`	N*2 matrix of 2 intermediate variables.
`response`	response vectors with values 0 or 1.

References

David J. Hand, Hua Gui Li, Niall M. Adams (2001), Supervised classification with structured class definitions. *Computational Statistics & Data Analysis* **36**, 209–225.

Examples

```
theta90 <- varset(N = 1000, sigma = 0.1, theta = 90, threshold = 0)
theta0 <- varset(N = 1000, sigma = 0.1, theta = 0, threshold = 0)
par(mfrow = c(1, 2))
plot(theta0$intermediate)
plot(theta90$intermediate)
```

Index