

# Package ‘ipred’

January 2, 2012

**Title** Improved Predictors

**Version** 0.8-11

**Date** 2011-02-08

**Author** Andrea Peters and Torsten Hothorn

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

**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), rpart (>= 3.1-8), MASS, mlbench, survival, nnet,class

**Suggests** mvtnorm

**License** GPL

**Repository** CRAN

**Date/Publication** 2011-02-08 22:04:31

## R topics documented:

|                  |    |
|------------------|----|
| bagging          | 2  |
| bootest          | 6  |
| control.errorest | 7  |
| cv               | 8  |
| DLBCL            | 9  |
| dystrophy        | 10 |
| errorest         | 11 |
| GBSG2            | 16 |
| GlaucomaM        | 17 |
| GlaucomaMVF      | 20 |
| inbagg           | 23 |
| inclass          | 25 |
| ipredknn         | 28 |

|                             |           |
|-----------------------------|-----------|
| kfoldcv . . . . .           | 29        |
| mypredict.lm . . . . .      | 30        |
| predict.classbagg . . . . . | 30        |
| predict.inbagg . . . . .    | 32        |
| predict.inclass . . . . .   | 33        |
| predict.ipredknn . . . . .  | 35        |
| predict.slda . . . . .      | 36        |
| print.classbagg . . . . .   | 37        |
| print.cvclass . . . . .     | 38        |
| print.inbagg . . . . .      | 38        |
| print.inclass . . . . .     | 39        |
| prune.classbagg . . . . .   | 40        |
| rsurv . . . . .             | 41        |
| sbrier . . . . .            | 42        |
| slda . . . . .              | 44        |
| Smoking . . . . .           | 46        |
| summary.classbagg . . . . . | 47        |
| summary.inbagg . . . . .    | 47        |
| summary.inclass . . . . .   | 48        |
| varset . . . . .            | 49        |
| <b>Index</b>                | <b>51</b> |

---

|         |  |
|---------|--|
| bagging | <i>Bagging Classification, Regression and Survival Trees</i> |
|---------|--|

---

## 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**

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

**Details**

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.

### Value

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

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

### Author(s)

Torsten.Hothorn <Torsten.Hothorn@rzmail.uni-erlangen.de>

### References

- Leo Breiman (1996a), Bagging Predictors. *Machine Learning* **24**(2), 123–140.
- Leo Breiman (1996b), Out-Of-Bag Estimation. *Technical Report* <ftp://ftp.stat.berkeley.edu/pub/users/breiman/OOBestimation.ps.Z>.
- 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 (2002b), Bundling Classifiers by Bagging Trees. *submitted*. Preprint available from <http://www.mathpreprints.com/math/Preprint/blausen/20021016/1>.
- Torsten Hothorn, Berthold Lausen, Axel Benner and Martin Radespiel-Troeger (2004), Bagging Survival Trees. *Statistics in Medicine*, **23**(1), 77–91.

**Examples**

```
# Classification: Breast Cancer data

data(BreastCancer)

# 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)
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)

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

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

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)
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.  |

|                         |   |
|-------------------------|---|
| <code>list.tindx</code> | list of numeric vectors, indicating which observations are included in each bootstrap sample. |
| <code>both.boot</code>  | logical, return both (bootstrap and 632plus) estimations or only one of them.                 |
| <code>...</code>        | additional arguments to <code>model</code> .  |

**Details**

See [errorest](#).

**Author(s)**

Torsten.Hothorn <Torsten.Hothorn@rzmail.uni-erlangen.de>

---

|                               |                                      |
|-------------------------------|--------------------------------------|
| <code>control.errorest</code> | <i>Control Error Rate Estimators</i> |
|-------------------------------|--------------------------------------|

---

**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**

|                          |   |
|--------------------------|---|
| <code>k</code>           | integer, specify <code>\$k\$</code> for <code>\$k\$</code> -fold cross-validation.  |
| <code>nboot</code>       | integer, number of bootstrap replications.  |
| <code>strat</code>       | logical, if TRUE, cross-validation is performed using stratified sampling (for classification problems).  |
| <code>random</code>      | logical, if TRUE, cross-validation is performed using a random ordering of the data.  |
| <code>predictions</code> | 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 <code>nboot</code> is returned with predicted values of the <code>i</code> th out-of-bootstrap sample in column <code>i</code> and 'NA's for those observations not included in the <code>i</code> th out-of-bootstrap sample. |
| <code>getmodels</code>   | logical, indicates a list of all models should be returned. For cross-validation only.  |
| <code>list.tindx</code>  | 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.

**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](#).

**Author(s)**

Torsten.Hothorn <Torsten.Hothorn@rzmail.uni-erlangen.de>

---

|       |                                      |
|-------|--------------------------------------|
| DLBCL | <i>Diffuse Large B-Cell Lymphoma</i> |
|-------|--------------------------------------|

---

**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**

```
data(DLBCL)
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 dehydrogenase.

**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 dehydrogenase (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 Geoffrey Hinton (1998), Coaching variables for regression and classification. *Statistics and Computing* 8, 25-33.

**Examples**

```
## Not run:
data(dystrophy)
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 $lhs \sim rhs$ . Either describing the model of explanatory and response variables in the usual way (see <a href="#">lm</a> ) or the model between explanatory and intermediate variables in the framework of indirect classification, see <a href="#">inclass</a> . |
| data      | a data frame containing the variables in the model formula and additionally the class membership variable if <code>model = inclass</code> . <code>data</code> is required for indirect classification, otherwise <code>formula</code> 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 <a href="#">na.omit</a> .   |

|                        |   |
|------------------------|---|
| <code>model</code>     | function. Modelling technique whose error rate is to be estimated. The function <code>model</code> can either return an object representing a fitted model or a function with argument <code>newdata</code> which returns predicted values. In this case, the <code>predict</code> argument to <code>errorest</code> is ignored.  |
| <code>predict</code>   | 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, <code>predict</code> 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. |
| <code>estimator</code> | estimator of the misclassification error: <code>cv</code> cross-validation, <code>boot</code> bootstrap or <code>632plus</code> bias corrected bootstrap (classification only).   |
| <code>est.param</code> | a list of additional parameters that control the calculation of the estimator, see <a href="#"><code>control.errorest</code></a> for details.   |
| <code>...</code>       | additional parameters to <code>model</code> .   |

## 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.param` 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.

## Author(s)

Andrea Peters <Peters.Andrea@imbe.imed.uni-erlangen.de> and  
Torsten Hothorn <Torsten.Hothorn@rzmail.uni-erlangen.de>

## 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)

# 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)
## 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.param = 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.param = control.errorest(list.tindx = list.tindx))

## End(Not run)
data(Glass)

# 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.param=control.errorest(strat=TRUE))

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

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

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

# compute sensitivity and specificity for stabilised LDA

data(GlaucomaM)

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

# sensitivity

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

# specificity

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(iffelse(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.param=control.errorest(k=nrow(smoking)))

## End(Not run)

# Regression

data(BostonHousing)

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

# Survival data

data(GBSG2)

# 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.param=control.errorest(k=5))

```

---

GBSG2

*German Breast Cancer Study Group 2*

---

### Description

A data frame containing the observations from the GBSG2 study.

### Usage

```
data(GBSG2)
```

### Format

This data frame contains the observations of 686 women:

**horTh** hormonal therapy, a factor at two levels no and yes.

**age** of the patients in years.

**menostat** menopausal status, a factor at two levels pre (premenopausal) and post (postmenopausal).

**tsize** tumor size (in mm).

**tgrade** tumor grade, a ordered factor at levels I < II < III.

**pnodes** number of positive nodes.

**progrec** progesterone receptor (in fmol).

**estrec** estrogen receptor (in fmol).

**time** recurrence free survival time (in days).

**cens** censoring indicator (0- censored, 1- event).

### Source

<http://www.blackwellpublishers.com/rss/Volumes/A162p1.htm>

### References

M. Schumacher, G. Basert, H. Bojar, K. Huebner, M. Olschewski, W. Sauerbrei, C. Schmoor, C. Beyerle, R.L.A. Neumann and H.F. Rauschecker for the German Breast Cancer Study Group (1994), Randomized  $2 \times 2$  trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. *Journal of Clinical Oncology*, **12**, 2086–2093.

W. Sauerbrei and P. Royston (1999). Building multivariable prognostic and diagnostic models: transformation of the predictors by using fractional polynomials. *Journal of the Royal Statistical Society Series A*, Volume **162**(1), 71–94.

**Examples**

```

data(GBSG2)

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)

```

**Description**

The GlaucomaM data has 196 observations in two classes. 62 variables are derived from a confocal laser scanning image of the optic nerve head, describing its morphology. Observations are from normal and glaucomatous eyes, respectively.

**Usage**

```
data(GlaucomaM)
```

**Format**

This data frame contains the following predictors describing the morphology of the optic nerve head 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.  
**Class** a factor with levels glaucoma and normal.

### Details

All variables are derived from a laser scanning image of the eye background taken by the Heidelberg Retina Tomograph. Most of the 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 observations in both groups are matched by age and sex to prevent any bias.

### Note

GLaucomM overlaps in some parts with [GlaucomaMVF](#).

### Source

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

### Examples

```
data(GlaucomaM)
errorest(Class ~ ., data=GlaucomaM, model=rpart,
         predict=function(obj, newdata)
           predict(obj, newdata, type="class"),
         control=rpart.control(xval=0))
glbagg <- bagging(Class ~ ., data=GlaucomaM, coob=TRUE)
glbagg
```

---

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 ocular 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 ocular 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 ocular 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 (2002), Diagnosis of glaucoma by indirect classifiers, *Methods of Information in Medicine*, to appear.

**Examples**

```
## Not run:
data(GlaucomaMVF)

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 subbagging.

**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  $y \sim w_1 + w_2 + w_3 \sim x_1 + x_2 + x_3$  describes how to model the intermediate variables  $w_1$ ,  $w_2$ ,  $w_3$  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 <code>newdata</code> 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. <code>subset</code> ).  |

### 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  $y \sim w_1 + w_2 + w_3 + x_1 + x_2$  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 <code>nbagg</code> , describing the prediction models corresponding to each bootstrap sample. Each element of <code>mtrees</code> is a list with elements <code>bindx</code> (observations of bag sample), <code>btree</code> (classifying function of bag sample) and <code>bfct</code> (predictive models for intermediates of bag sample). |
|--------|--|

**y**                    vector of response values.  
**W**                    data frame of intermediate variables.  
**X**                    data frame of explanatory variables.

### Author(s)

Andrea Peters <Peters.Andrea@imbe.imed.uni-erlangen.de>

### 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

```

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 $y \sim w_1 + w_2 + w_3 \sim x_1 + x_2 + x_3$ models each intermediate variable $w_1$ , $w_2$ , $w_3$ by $w_i \sim x_1 + x_2 + x_3$ and the response by $y \sim w_1 + w_2 + w_3$ 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  $y \sim w_1 + w_2 + w_3 \sim x_1 + x_2 + x_3$  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.

### Author(s)

Andrea Peters <Peters.Andrea@imbe.imed.uni-erlangen.de>

### 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)
# 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 $lhs \sim rhs$ 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](#).

### Author(s)

Torsten.Hothorn <Torsten.Hothorn@rzmail.uni-erlangen.de>

### Examples

```
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**

|                     |   |
|---------------------|---|
| <code>k</code>      | number of groups.                                 |
| <code>N</code>      | total sample size.                                |
| <code>nlevel</code> | 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  $k$ .

**Author(s)**

Torsten Hothorn <Torsten.Hothorn@rzmail.uni-erlangen.de>

**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](#)).

### Author(s)

Andrea Peters <Peters.Andrea@imbe.imed.uni-erlangen.de>

---

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.

**Author(s)**

Torsten.Hothorn <Torsten.Hothorn@rzmail.uni-erlangen.de>

**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)
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 <a href="#">inbagg</a> .         |
| newdata | data frame to be classified.                                 |
| ...     | additional arguments 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.

## Author(s)

Andrea Peters <Peters.Andrea@imbe.imed.uni-erlangen.de>

## 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(mvtnorm)
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 <a href="#">inclass</a> .   |
| newdata | data frame to be classified.   |
| ...     | additional arguments corresponding to the predictive models specified in <a href="#">inclass</a> . |

**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.

**Author(s)**

Andrea Peters <Peters.Andrea@imbe.imed.uni-erlangen.de>

**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(iffelse((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)

# 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(iffelse(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)
glaucoma <- GlaucomaMVF[(names(GlaucomaMVF) != "tension")]
# explanatory variables are derived by laser scanning image and intra ocular 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)
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.

**Author(s)**

Torsten.Hothorn <Torsten.Hothorn@rzmail.uni-erlangen.de>

---

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 <a href="#">predict.lda</a> . |

**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.

**Author(s)**

Torsten.Hothorn <Torsten.Hothorn@rzmail.uni-erlangen.de>

---

`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

**Author(s)**

Torsten.Hothorn <Torsten.Hothorn@rzmail.uni-erlangen.de>

---

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

**Author(s)**

Torsten.Hothorn <Torsten.Hothorn@rzmail.uni-erlangen.de>

---

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.

**Author(s)**

Andrea Peters <Peters.Andrea@imbe.imed.uni-erlangen.de>

---

`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.

**Author(s)**

Andrea Peters <Peters.Andrea@imbe.imed.uni-erlangen.de>

---

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 <a href="#">bagging</a> (calling this tree is needed by the generic function <code>prune</code> in package <code>rpart</code> ). |
| cp   | complexity parameter, see <a href="#">prune.rpart</a> .  |
| ...  | additional arguments to <a href="#">prune.rpart</a> .  |

## 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.

## Author(s)

Torsten.Hothorn <Torsten.Hothorn@rzmail.uni-erlangen.de>

## Examples

```
data(Glass)  
  
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 <code>runif(N, 0, gamma)</code> . Defaults to NULL (no censoring). |
| fact   | scale parameter for <code>model=tree</code> .   |
| pnon   | number of additional non-informative variables for the tree model.                            |
| gethaz | logical, indicating whether 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

```
# 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=c(0, max(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 its integrated version were suggested by Graf et al (1999).

## 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.

**Examples**

```

data(DLBCL)
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)
stopifnot(all.equal(cleans(a), sum(mymin)/max(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)

```

---

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 NAs. 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  $WD = \text{diag}(W)DL$  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`.

### Author(s)

Torsten.Hothorn <Torsten.Hothorn@rzmail.uni-erlangen.de>

### 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

```
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

**Author(s)**

Torsten.Hothorn <Torsten.Hothorn@rzmail.uni-erlangen.de>

---

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

**Author(s)**

Andrea Peters <Peters.Andrea@imbe.imed.uni-erlangen.de>

**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

**Author(s)**

Andrea Peters <Peters.Andrea@imbe.imed.uni-erlangen.de>

**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  $x = (x_1, x_2)^T$  and of two responses  $y = (y_1, y_2)^T$  are simulated, following the formula:

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

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.    |

**Author(s)**

Andrea Peters <Peters.Andrea@imbe.imed.uni-erlangen.de>

**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

## \*Topic **datasets**

- DLBCL, 9
- dystrophy, 10
- GBSG2, 16
- GlaucomaM, 17
- GlaucomaMVF, 20
- Smoking, 46

## \*Topic **misc**

- bootest, 6
- control.errorest, 7
- cv, 8
- errorest, 11
- inbagg, 23
- inclass, 25
- kfoldcv, 29
- mypredict.lm, 30
- predict.inbagg, 32
- predict.inclass, 33
- print.cvclass, 38
- print.inbagg, 38
- print.inclass, 39
- summary.inbagg, 47
- summary.inclass, 48
- varset, 49

## \*Topic **multivariate**

- ipredknn, 28
- predict.ipredknn, 35
- predict.slda, 36
- slda, 44

## \*Topic **survival**

- rsurv, 41
- sbrier, 42

## \*Topic **tree**

- bagging, 2
- predict.classbagg, 30
- print.classbagg, 37
- prune.classbagg, 40
- summary.classbagg, 47

bagging, 2, 25, 27, 28, 31, 37, 40, 47

bootest, 6, 12

control.errorest, 7, 12

cv, 8, 12

DLBCL, 9

dystrophy, 10

errorest, 6–8, 11, 28, 38

GBSG2, 16

GlaucomaM, 17, 23

GlaucomaMVF, 20, 20

inbagg, 23, 32, 33

inclass, 11, 24, 25, 27, 33, 34

ipredbagg (bagging), 2

ipredknn, 28

kfoldcv, 29

knn, 28, 36

lda, 4, 45

lm, 11, 25

mypredict.lm, 30

na.omit, 11

na.rpart, 3, 45

predict, 4, 36

predict.classbagg, 3, 4, 30

predict.inbagg, 32

predict.inclass, 30, 33

predict.ipredknn, 28, 35

predict.lda, 36

predict.regbagg (predict.classbagg), 30

predict.slda, 36, 45

predict.survbagg (predict.classbagg), 30

print, 4

print (print.classbagg), 37

print.bootestclass (print.cvclass), 38  
print.bootestreg (print.cvclass), 38  
print.bootestsurv (print.cvclass), 38  
print.classbagg, 4, 37  
print.cvclass, 38  
print.cvreg (print.cvclass), 38  
print.cvsurv (print.cvclass), 38  
print.inbagg, 38  
print.inclass, 39  
print.summary.bagging  
    (summary.classbagg), 47  
print.summary.inbagg, 48  
print.summary.inbagg (summary.inbagg),  
    47  
print.summary.inclass, 49  
print.summary.inclass  
    (summary.inclass), 48  
prune.classbagg, 3, 4, 40  
prune.regbagg (prune.classbagg), 40  
prune.rpart, 4, 40  
prune.survbagg (prune.classbagg), 40  
  
rpart, 3, 24, 25  
rpart.control, 3  
rsurv, 41  
  
sbrier, 42  
slda, 4, 44  
Smoking, 46  
summary, 4  
summary.classbagg, 4, 47  
summary.inbagg, 47  
summary.inclass, 48  
summary.regbagg (summary.classbagg), 47  
summary.survbagg (summary.classbagg), 47  
Surv, 3  
  
varset, 49