Package ‘apricom’

November 24, 2015

Title Tools for the a Priori Comparison of Regression Modelling Strategies

Version 1.0.0

Date 2015-11-11

Maintainer Romin Pajouheshnia <R.Pajouheshnia@umcutrecht.nl>

Description Tools to compare several model adjustment and validation methods prior to application in a final analysis.

Depends R (>= 3.1.1)

Imports logistf, penalized, rms, graphics, stats, shrink

Suggests knitr, testthat, rmarkdown

License GPL-2

LazyData TRUE

RoxygenNote 5.0.1

NeedsCompilation no

Author Romin Pajouheshnia [aut, cre],
Wiebe Pestman [aut],
Rolf Groenwold [aut]

Repository CRAN

Date/Publication 2015-11-24 19:05:16

R topics documented:

`bootval` *Bootstrap-derived Shrinkage After Estimation*

Description

Shrink regression coefficients using a bootstrap-derived shrinkage factor.

Usage

```
bootval(dataset, model, N, sdm, int = TRUE, int.adj)
```

Arguments

`dataset`	a dataset for regression analysis. Data should be in the form of a matrix, with the outcome variable as the final column. Application of the `datashape` function beforehand is recommended, especially if categorical predictors are present. For regression with an intercept included a column vector of 1s should be included before the dataset (see examples).
`model`	type of regression model. Either "linear" or "logistic".
`N`	the number of times to replicate the bootstrapping process
`sdm`	a shrinkage design matrix. For examples, see `ols.shrink`
`int`	logical. If TRUE the model will include a regression intercept.
`int.adj`	logical. If TRUE the regression intercept will be re-estimated after shrinkage of the regression coefficients.

Details

This function applies bootstrapping to a dataset in order to derive a shrinkage factor and apply it to the regression coefficients. Regression coefficients are estimated in a bootstrap sample, and then a shrinkage factor is estimated using the input data. The mean of N shrinkage factors is then applied to the original regression coeffients, and the regression intercept may be re-estimated.

This process can currently be applied to linear or logistic regression models.

Value

bootval returns a list containing the following:

raw.coeff	the raw regression model coefficients, pre-shrinkage.
shrunk.coeff	the shrunken regression model coefficients
lambda	the mean shrinkage factor over N bootstrap replicates
N	the number of bootstrap replicates
sdm	the shrinkage design matrix used to apply the shrinkage factor(s) to the regression coefficients

Examples

```
## Example 1: Linear regression using the iris dataset
data(iris)
iris.data <- as.matrix(iris[, 1:4])
iris.data <- cbind(1, iris.data)
sdm1 <- matrix(c(0, 1, 1, 1), nrow = 1)
set.seed(777)
bootval(dataset = iris.data, model = "linear", N = 200, sdm = sdm1,
int = TRUE, int.adj = TRUE)

## Example 2: logistic regression using a subset of the mtcars data
data(mtcars)
mtc.data <- cbind(1,datashape(mtcars, y = 8, x = c(1, 6, 9)))
head(mtc.data)
set.seed(777)
bootval(dataset = mtc.data, model = "logistic", N = 500)
```

---

compare *A Comparison of Regression Modelling Strategies*

Description

Compare two nested modelling strategies and return measures of their relative predictive performance

Usage

```
compare(model, Ntrials, strat1, strat2, data, Nrows, Ncomp, int = TRUE,
  int.adj, trim = FALSE, output = TRUE)
```

Arguments

model	the type of regression model. Either "linear" or "logistic".
Ntrials	number of simulation trials.
strat1	a list containing the strategy name and strategy-specific parameter values. This modelling strategy is taken as the reference for comparison.

`strat2`	a list containing the strategy name and strategy-specific parameter values. This modelling strategy is compared with the reference strategy, strat1.
`data`	a list describing the dataset in which the selected modelling strategies will be compared. If the first object in the list is "norm" or "unif", the user may submit parameters for generating multivariable simulated datasets (see details below. Users may specify their own dataset using the format `list("dataset", user_data)`, where the second object in the list is their own dataset.
`Nrows`	the number of rows of observations in simulated datasets of type "norm" or "unif".
`Ncomp`	the number of rows of observations in the comparison set. This dataset is taken to represent the overall population, from which the training set is sampled. When `data` is of type `"dataset"`, if Ncomp is not specified, the original data will be used as the comparison dataset.
`int`	logical. If `int == TRUE` an intercept will be included in the regression model.
`int.adj`	logical. If `int.adj == TRUE` the intercept will be re-estimated after shrinkage is applied to the regression coefficients.
`trim`	logical. If `trim == TRUE` a "trimmed" comparison distribution will be returned, along with a victory rate and median precision ratio derived using the trimmed distribution. The trimmed distribution only contains precision ratios within a range of plus or minus two times the interquartile range around the median precision ratio.
`output`	logical. If `output == TRUE` the function will return two graphical representations of the comparison distribution.

Details

This is the core function in the **apricom** package. The *compare* function can be used to compare the performance of two prediction model building approaches for either simulated or user-specified data. For further details, see the **apricom** user manual.

The following strategies are currently supported: heuristic shrinkage ("heuristic"), split-sample-derived shrinkage ("split"), cross-validation-derived shrinkage ("kcv"), leave-one-out cross-validation-derived shrinkage ("loocv"), bootstrap-derived shrinkage ("boot") and penalized logistic regression using Firth's penalty ("pml.firth"). Furthermore, models built using these methods may be compared with raw models fitted by ordinary least squares estimation ("lsq") or maximum likelihood estimation ("ml").

Strategies should be specified within the "strat1" and "strat2" arguments in the form of a list, starting with the strategy name (as listed above in parentheses), followed by relevant parameters for each respective method. For further details see individual help files for each strategy, and the examples below. Note that in the *compare* function call, the dataset should not be specified within the "strat1" or "strat2" arguments, and instead should only be called within the "data" argument.

Value

`compare` returns a list containing the following:

`VR`	the victory rate of strategy 2 over strategy 1.

`MPR`	the median precision ratio over Ntrials comparison trials.
`PR.IQR`	the precision ratio interquartile range over `Ntrials` comparison trials.
`VR.trim`	if `trim == TRUE` the "trimmed" victory rate of strategy 2 over strategy 1 is returned.
`MPR.trim`	if `trim == TRUE` the "trimmed" median precision ratio over Ntrials comparison trials is returned.
`distribution`	the comparison distribution of strategy 2 vs. strategy 1. This is the distribution of precision ratios generated from `Ntrials` comparison trials
`distribution.trim`	if `trim == TRUE` the "trimmed" comparison distribution is returned.
`N.rejected`	the number of trials excluded from the comparison distribution by trimming
`strat1`	modelling strategy 1
`strat2`	modelling strategy 2
`shrinkage1`	If strategy 1 is a shrinkage-after-estimation technique, a vector or matrix containing the shrinkage factor estimated in each trial is returned
`shrinkage1`	If strategy 1 is a shrinkage-after-estimation technique, a vector or matrix containing the shrinkage factor estimated in each trial is returned

Note

When using `compare` it is strongly recommended that ideally 10000 comparison trials are used, to give stable estimates. Comparisons with logistic regression modelling model adjustment strategies are *considerably* slower than with linear regression, and 1000-5000 trials may be preferred. The examples provided in this documentation use considerably fewer comparison trials and yield highly unstable estimates.

References

Pestman W., Groenwold R. H. H., Teerenstra. S, *"Comparison of strategies when building linear prediction models."* Numerical Linear Algebra with Applications (2013)

Examples

```
## Example 1: Comparison of heuristic formula-derived shrinkage against
## a raw least squares model. Data is simulated multivariable random
## normally distributed.The comparison set will have 2000 rows. Here only
## 10 trial replicates are used, but at least 1000 should be used in practice

 mu <- c(rep(0, 21))
 rho <- 0.5
 comp1 <- compare(model = "linear", Ntrials = 10, strat1 = list("lsq"),
          strat2 = list("heuristic", DF = 8),
          data = list("norm", mu, rho), Nrows = 200, Ncomp = 2000,
          int = TRUE, int.adj = FALSE, trim = FALSE, output = TRUE)


## Example 2: A truncated comparison of 10-rep, 10-fold
## cross-validation-derived shrinkage against leave-one-out cross-validation.
## Data is simulated multivariable random uniformly distributed
## (50 rows; 5 predictors with mean=0 ; r^2 = 0.7)
## The comparison set will contain 1000 observations.

mu <- c(rep(0, 6))
rho <- 0.7
comp2 <- compare(model = "linear", Ntrials = 10, strat1 = list("loocv"),
          strat2 = list("kcv", k = 10, nreps = 10),data = list("unif", mu, rho),
          Nrows = 50, Ncomp = 1000, trim = TRUE)


## Example 3:  Comparison of penalized logistic regression with
## Firth's penalty against raw logistic regression model using
## maximum likelihood estimation.
## Note that the logistf package is required for pml.firth.

library(shrink)
data(deepvein)
dv.data <- datashape(deepvein, y = 3, x = 4:11)
set.seed(123)
comp4 <- compare(model = "logistic", Ntrials = 10,
         strat1 = list("ml"), strat2 = list("pml.firth"),
         data = list("dataset", dv.data), int = TRUE,
         int.adj = TRUE, trim = FALSE, output = TRUE)
```

`compdist` *Comparison Distribution Descriptives*

Description

calculate a comparison distribution and associated descriptive statisics.

Usage

```
compdist(a, b, model, output, lambda1, lambda2, trim, strat1, strat2)
```

Arguments

`a`	a vector containing the sum of square errors or -2 * log likelihood estimates derived using modelling strategy 1.
`b`	a vector containing the sum of square errors or -2 * log likelihood estimates derived using modelling strategy 2.
`model`	type of regression model. Either "linear" or "logistic".
`output`	logical. If output is TRUE the function will return two graphical representations of the comparison distribution.

`lambda1`	a vector or matrix containing the estimated shrinkage factors derived using strategy 1
`lambda2`	a vector or matrix containing the estimated shrinkage factors derived using strategy 2
`trim`	logical. If trim is TRUE a "trimmed" comparison distribution will be returned, along with a victory rate and median precision ratio derived using the trimmed distribution. The trimmed distribution only contains precision ratios within a range of plus or minus two times the interquartile range around the median precision ratio.
`strat1`	a list containing the strategy name and strategy-specific parameter values.
`strat2`	a list containing the strategy name and strategy-specific parameter values.

Details

This function works within `compare` to calculate the comparison distribution of strategy 2 against strategy 1. If `trim == TRUE`, a truncated distribution consisting of +/- 2 * interquartile range around the median precision ratio.

The victory rate for linear regression is the left tail probability with a cut-off of 1. The victory rate for logistic regression instead taken at a cut-off of 0. For more details, refer to the accompanying PDF document.

If `output == TRUE` the function will plot a histogram of the distribution and a scatterplot of the precision (SSE or -2 log likelihood) of strategy 2 against strategy 1. Points that lie below the red or blue line represent trials in which strategy 2 outperformed strategy 1 (the victory rate).

Note

For details of returned values and examples see `compare`.

`datashape` *Data shaping tools*

Description

Generate a new dataset in a format that is accepted by `compare`. Dummy variables are created for categorical variables.

Usage

```
datashape(dataset, y, x, subs)
```

Arguments

`dataset`	a dataframe or matrix containing user data.
`y`	the column number of the outcome variable in dataset d.
`x`	a vector containing the explanatory or predictor variables of interest. The new data matrix will only contain these variables and the outcome variable.
`subs`	a vector describing a subset of rows from dataset d to include in the returned data matrix.

Details

This function can be used to prepare a dataset before applying the `compare` function. The outcome column number must be specified, and specific predictors and observation subsets may be specified. 2-level categorical variables will be converted to binary, while dummy variables will be created for categorical predictors with greater than two levels.

The "datashaped" dataset should be saved to a new object.

Value

This function returns a matrix conforming to the specifications supplied to the datashape function.

Warning

`datashape` will not function if missing values are present.

@examples ## Preparing the iris dataset data(iris) iris.shaped <- datashape(dataset = iris, y = 4) head(iris.shaped) ## Creating a copy of iris with sepal-related predictors and a subset of observations. iris.sub <- datashape(dataset = iris, y = 4, x = c(1,2), subs = c(1:20, 50:70)) head(iris.sub)

`grandpart` *Accessory Function for Cross-validation*

Description

Randomly partition a dataset into k approximately equally sized partitions.

Usage

```
grandpart(dataset, k)
```

Arguments

`dataset`	a dataset for partitioning.
`k`	the number of partitions; the cross-validation fold number.

Value

Returns a list containing the partitioned datasets.

Note

This function is not designed to be called directly, but acts within `kcrossval`

`kcrossval` *Cross-validation-derived Shrinkage After Estimation*

Description

Shrink regression coefficients using a Cross-validation-derived shrinkage factor.

Usage

```
kcrossval(dataset, model, k, nreps, sdm, int = TRUE, int.adj)
```

Arguments

`dataset`	a dataset for regression analysis. Data should be in the form of a matrix, with the outcome variable as the final column. Application of the `datashape` function beforehand is recommended, especially if categorical predictors are present. For regression with an intercept included a column vector of 1s should be included before the dataset (see examples)
`model`	type of regression model. Either "linear" or "logistic".
`k`	the number of cross-validation folds. This number must be within the range 1 < k <= 0.5 * number of observations
`nreps`	the number of times to replicate the cross-validation process.
`sdm`	a shrinkage design matrix. For examples, see `ols.shrink`
`int`	logical. If TRUE the model will include a regression intercept.
`int.adj`	logical. If TRUE the regression intercept will be re-estimated after shrinkage of the regression coefficients.

Details

This function applies k-fold cross-validation to a dataset in order to derive a shrinkage factor and apply it to the regression coefficients. Data is randomly partitioned into k equally sized sets. One set is used as a validation set, while the remaining data is used as a training set. Regression coefficients are estimated in the training set, and then a shrinkage factor is estimated using the validation set. This process is repeated so that each partitioned set is used as the validation set once, resulting in k folds. The mean of k shrinkage factors is then applied to the original regression coeffients, and the regression intercept may be re-estimated. This process is repeated nreps times and the mean of the regression coefficients is returned.

This process can currently be applied to linear or logistic regression models.

Value

kcrossval returns a list containing the following:

raw.coeff	the raw regression model coefficients, pre-shrinkage.
shrunk.coeff	the shrunken regression model coefficients.
lambda	the mean shrinkage factor over nreps cross-validation replicates.
nFolds	the number of cross-validation folds.
nreps	the number of cross-validation replicates.
sdm	the shrinkage design matrix used to apply the shrinkage factor(s) to the regression coefficients.

Examples

```
## Example 1: Linear regression using the iris dataset
## 2-fold Cross-validation-derived shrinkage with 50 reps
data(iris)
iris.data <- as.matrix(iris[, 1:4])
iris.data <- cbind(1, iris.data)
sdm1 <- matrix(c(0, 1, 1, 1), nrow = 1)
kcrossval(dataset = iris.data, model = "linear", k = 2,
nreps = 50, sdm = sdm1, int = TRUE, int.adj = TRUE)

## Example 2: logistic regression using a subset of the mtcars data
## 10-fold CV-derived shrinkage (uniform shrinkage and intercept re-estimation)
data(mtcars)
mtc.data <- cbind(1,datashape(mtcars, y = 8, x = c(1, 6, 9)))
head(mtc.data)
set.seed(321)
kcrossval(dataset = mtc.data, model = "logistic",
k = 10, nreps = 10)
```

---

loglikelihood *Negative 2 log likelihood*

Description

Calculate the -2 * log likelihood of a dataset given a specified model.

Usage

```
loglikelihood(b, dataset)
```

Arguments

b	intercept and coefficients of a generalized linear model.
dataset	a test dataset used to derive the likelihood.

Value

the function returns the -2 * log likelihood.

Examples

```
## Using the mtcars dataset
## Resample, fit an ordinary least squares model and calculate likelihood
data(mtcars)
mtc.data <- cbind(1,datashape(mtcars, y = 8, x = c(1, 6, 9)))
head(mtc.data)
mtc.boot <- randboot(mtc.data, replace = TRUE)
boot.betas <- ml.rgr(mtc.boot)
loglikelihood(b = boot.betas, dataset = mtc.data)
```

`loocval` *Leave-one-out Cross-validation-derived Shrinkage*

Description

Shrink regression coefficients using a shrinkage factor derived using leave-one-out cross-validation.

Usage

```
loocval(dataset, model, nreps = 1, sdm, int = TRUE, int.adj)
```

Arguments

`dataset`	a dataset for regression analysis. Data should be in the form of a matrix, with the outcome variable as the final column. Application of the `datashape` function beforehand is recommended, especially if categorical predictors are present. For regression with an intercept included a column vector of 1s should be included before the dataset (see examples)
`model`	type of regression model. Either "linear" or "logistic".
`nreps`	the number of times to replicate the cross-validation process.
`sdm`	a shrinkage design matrix.
`int`	logical. If TRUE the model will include a regression intercept.
`int.adj`	logical. If TRUE the regression intercept will be re-estimated after shrinkage of the regression coefficients.

Details

This function applies leave-one-out cross-validation to a dataset in order to derive a shrinkage factor and apply it to the regression coefficients. One row of the data is used as a validation set, while the remaining data is used as a training set. Regression coefficients are estimated in the training set, and then a shrinkage factor is estimated using the validation set. This process is repeated so that each data row is used as the validation set once. The mean of the shrinkage factors is then applied to the original regression coeffients, and the regression intercept may be re-estimated. This process may be repeated nreps times but each rep should yield the same shrunken coefficients.

This process can currently be applied to linear or logistic regression models.

Value

`loocval` returns a list containing the following:

`raw.coeff`	the raw regression model coefficients, pre-shrinkage.
`shrunk.coeff`	the shrunken regression model coefficients.
`lambda`	the mean shrinkage factor over nreps cross-validation replicates.
`nreps`	the number of cross-validation replicates.
`sdm`	the shrinkage design matrix used to apply the shrinkage factor(s) to the regression coefficients.

Note

Warning: this method is not recommended for use in practice. Due to the high variance and inherent instability of leave-one-out methods the value of the shrinkage factor may be extreme.

Examples

```
## Example 1: Linear regression using the iris dataset
## Leave-one-out cross-validation-derived shrinkage
data(iris)
iris.data <- as.matrix(iris[, 1:4])
iris.data <- cbind(1, iris.data)
sdm1 <- matrix(c(0, 1, 1, 1), nrow = 1)
set.seed(123)
loocval(dataset = iris.data, model = "linear", sdm = sdm1,
int = TRUE, int.adj = TRUE)

## Example 2: logistic regression using a subset of the mtcars data
## Leave-one-out cross-validation-derived shrinkage
data(mtcars)
mtc.data <- cbind(1,datashape(mtcars, y = 8, x = c(1, 6, 9)))
head(mtc.data)
set.seed(123)
loocval(dataset = mtc.data, model = "logistic")
```

`ml.rgr` *Logistic Regression with Maximum Likelihood Estimation*

Description

Fit a logistic regression model using maximum likelihood estimation

Usage

```
ml.rgr(dataset)
```

Arguments

`dataset`	a `p x m` data matrix, where the final column is a binary outcome variable. `datashape` may be applied to data so that the dataset is in the correct format for this function (see manual)

Details

This function is a wrapper for `glm.fit`, for convenient application within several functions in the **apricomp** package. This function may be called directly. For regression with an intercept included, the first column in the dataset must be a column of 1s.

Value

The function returns a column-vector containing the logistic regression coefficients and intercept (if specified).

Examples

```
## Logistic regression using a subset of the mtcars data (outcome is "vs")
data(mtcars)
mtc.df <- mtcars[, c(8, 1, 9)]
mtc.shaped <- datashape(dataset = mtc.df, y = 1)
ml.rgr(mtc.shaped)
ml.rgr(cbind(1,mtc.shaped))
```

`ml.shrink` *Estimation of a Shrinkage Factor for Logistic Regression*

Description

Estimate a shrinkage factor for shrinkage-after-estimation techniques, with application to logistic regression models.

Usage

```
ml.shrink(b, dat)
```

Arguments

`b`	`1 x m` matrix of regression coefficients, derived by resampling or sample-splitting
`dat`	a `p x m` data matrix, where the final column is a binary outcome variable. This dataset acts as a "test set" or "validation set".

Details

This function works together with `bootval`, `splitval`, `kcrossval` and `loocval` to estimate a shrinkage factor. For further details, see References. This function should not be used directly, and instead should be called via one of the aforementioned shrinkage-after-estimation functions.

Value

the function returns a single shrinkage factor

Note

Currently, this function can only derive a single shrinkage factor for a given model, and is unable to estimate (weighted) predictor-specific shrinkage factors.

References

Harrell, F. E. *"Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis." Springer*, (2001).

Steyerberg, E. W. *"Clinical Prediction Models", Springer* (2009)

`ols.rgr` *Linear Regression using Ordinary Least Squares*

Description

Fit a linear regression model using Ordinary Least Squares.

Usage

```
ols.rgr(dataset)
```

Arguments

`dataset`	a `p x m` data matrix, where the final column is a continuous outcome variable. `datashape` may be applied to data so that the dataset is in the correct format for this function (see manual)

Details

This function may be called directly. For regression with an intercept included, the first column in the dataset must be a column of 1s.

Value

the function returns a column-vector containing the linear regression coefficients.

Examples

```
## Linear regression using a subset of the mtcars data (outcome is "wt")
data(mtcars)
mtc.df <- mtcars[, c(6, 1, 4)]
mtc.shaped <- datashape(dataset = mtc.df, y = 1)
ols.rgr(mtc.shaped)
ols.rgr(cbind(1,mtc.shaped))
```

`ols.shrink` *Estimation of a Shrinkage Factor for Linear Regression*

Description

Estimate a shrinkage factor for shrinkage-after-estimation techniques, with application to linear regression models.

Usage

```
ols.shrink(b, dat, sdm)
```

Arguments

`b`	1 x `m` matrix of regression coefficients, derived by resampling or sample splitting
`dat`	a `p x m` data matrix, where the final column is a continuous outcome variable. This dataset acts as a "test set" or "validation set".
`sdm`	the shrinkage design matrix. This determines the regression coefficients that will be involved in the shrinkage process.

Details

This is an accessory function that works together with `bootval`, `splitval`, `kcrossval` and `loocval` to estimate a shrinkage factor. For further details, see References. This function should not be used directly, and instead should be called via one of the aforementioned shrinkage-after-estimation functions.

Value

the function returns a shrinkage factor.

Note

Currently, this function can only derive a single shrinkage factor for a given model, and is unable to estimate (weighted) predictor-specific shrinkage factors.

References

Harrell, F. E. *"Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis." Springer*, (2001).

Steyerberg, E. W. *"Clinical Prediction Models", Springer* (2009)

Examples

```
## Shrinkage design matrix examples for a model with an
## intercept and 4 predictors:
## 1. Uniform shrinkage (default design within apricomp).
   sdm1 <- matrix(c(0, rep(1, 4)), nrow = 1)
   print(sdm1)
## 2. Non-uniform shrinkage; 1 shrinkage factor applied only to the
##    first two predictors
   sdm2 <- matrix(c(0, 1, 1, 0, 0), nrow = 1)
   print(sdm2)
```

`randboot` *Bootstrap Resampling*

Description

Generate a dataset from a given dataset using sampling with or without replacement.

Usage

```
randboot(dataset, n, replace = TRUE)
```

Arguments

`dataset`	a p x m data matrix.
`n`	the number of rows of observations to be included in the new dataset.
`replace`	logical. If replace is TRUE sampling will be with replacement, thus returning a bootstrap replicate.

Details

This function is a simple shortcut for generating a bootstrap sample. It was originally designed for use within the comparison framework, but can be used as a stand-alone function.

Value

The function returns a new n x m matrix. The default is a bootstrap replicate of the orginal data.

randnor *Multivariable Random Normal data*

Description

Generate a simulated multivariable random normally distributed dataset using the method of Cholesky Decomposition.

Usage

```
randnor(n, mu, Cov)
```

Arguments

`n`	the number of rows of observations in the dataset
`mu`	a vector of length m containing the column means of the dataset
`Cov`	an m x m covariance matrix

Value

A simulated matrix of values based on the input parameters is returned.

References

Rizzo M. L., *"Statistical Computing with R", Chapman & Hall/CRC* (2007)

Examples

```
## Simulated data based on the iris dataset
mu <- c(rep(0, 4))
covmatr <- matrix(c(0.7, -0.04, 1.3, 0.5, -0.04, 0.2, -0.3, -0.1,
1.3, -0.3, 3.1, 1.3, 0.5, -0.1, 1.3, 0.6), ncol = 4)
sim.dat <- randnor(n = 100, mu = mu, Cov = covmatr)
head(sim.dat)
```

randpart *Accessory Function for Sample Splitting*

Description

Randomly partition a dataset into a training set and a test set.

Usage

```
randpart(dataset, fract)
```

Arguments

`dataset`	a dataset for splitting.
`fract`	the proportion of observations to be designated to the training set.

Value

Returns a list containing the training set and test set.

Note

This function is not designed to be called directly, but acts within `splitval`

`randunif` *Multivariable Random Uniform data*

Description

Generate a simulated multivariable random uniformly distributed dataset using the method of Cholesky Decomposition.

Usage

```
randunif(n, mu, Cov, Q)
```

Arguments

`n`	the number of rows of observations in the dataset
`mu`	a vector containing the column means of the dataset
`Cov`	a covariance matrix
`Q`	an optional orthogonal matrix

Value

A simulated matrix of values based on the input parameters is returned.

References

Rizzo M. L., *"Statistical Computing with R", Chapman & Hall/CRC* (2007)

Examples

```
## Simulated data based on the iris dataset
mu <- c(rep(0, 4))
covmatr <- matrix(c(0.7, -0.04, 1.3, 0.5, -0.04, 0.2, -0.3, -0.1,
1.3, -0.3, 3.1, 1.3, 0.5, -0.1, 1.3, 0.6), ncol = 4)
sim.dat <- randunif(n = 100, mu = mu, Cov = covmatr)
head(sim.dat)
```

`shrink.heur` *Shrinkage After Estimation Using Heuristic Formulae*

Description

Shrink regression coefficients using heuristic formulae, first described by Van Houwelingen and Le Cessie (Stat. Med., 1990)

Usage

```
shrink.heur(dataset, model, DF, int = TRUE, int.adj = FALSE)
```

Arguments

`dataset`	a dataset for regression analysis. Data should be in the form of a matrix, with the outcome variable as the final column. Application of the `datashape` function beforehand is recommended, especially if categorical predictors are present. For regression with an intercept included a column vector of 1s should be included before the dataset (see examples)
`model`	type of regression model. Either "linear" or "logistic".
`DF`	the number of degrees of freedom or number of predictors in the model. If DF is missing the value will be automatically estimated. This may be inaccurate for complex models with non-linear terms.
`int`	logical. If TRUE the model will include a regression intercept.
`int.adj`	logical. If TRUE the regression intercept will be re-estimated after shrinkage of the regression coefficients. If FALSE the regression intercept will be re-estimated as described by Harrell 2001.

Details

This function can be used to estimate shrunken regression coefficients based on heuristic formulae (see References). A linear or logistic regression model with an intercept is fitted to the data, and a shrinkage factor is estimated. The shrinkage factor is then applied to the regression coefficients. If `int.adj == FALSE` the intercept value is estimated as described in Harrell 2001.If `int.adj == TRUE` the intercept value will be re-estimated by refitting the model with the shrunken coefficients.

The heuristic formula work by applying the number of model degrees of freedom (or the number of predictors) as a penalty, and so as the model becomes more complex, the necessary shrinkage increases, and the shrinkage factor becomes closer to zero.

Value

`shrink.heur` returns a list containing the following:

`raw.coeff`	the raw regression model coefficients, pre-shrinkage.
`shrunk.coeff`	the shrunken regression model coefficients

`lambda`	the heuristic shrinkage factor
`DF`	the number of degrees of freedom or number of predictors in the model

Note

Warning: In poorly fitting models that includea large number of predictors the number of degrees of freedom may approch or exceed the model chi square. In such cases the shrinkage factor will be very small or even negative, and a different model building strategy is recommended.

References

Harrell, F. E. *"Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis." Springer*, (2001).

Harrell, F. E., Kerry L. Lee, and Daniel B. Mark. *"Tutorial in biostatistics multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors." Statistics in medicine* (1996) 15:361-387.

Steyerberg, E. *"Clinical Prediction Models" Springer* (2009)

Van Houwelingen, J. C. and Le Cessie, S., *"Predictive value of statistical models." Statistics in medicine* (1990) 9:1303:1325.

Examples

```
## Example 1: Linear regression using the iris dataset
## shrinkage using a heuristic formula
data(iris)
iris.data <- as.matrix(iris[, 1:4])
iris.data <- cbind(1, iris.data)
set.seed(123)
shrink.heur(dataset = iris.data, model = "linear")

## Example 2: logistic regression using a subset of the mtcars data
## shrinkage using a heuristic formula
data(mtcars)
mtc.data <- cbind(1,datashape(mtcars, y = 8, x = c(1,6,9)))
head(mtc.data)
set.seed(321)
shrink.heur(dataset = mtc.data, model = "logistic", DF = 3,
int = TRUE, int.adj = TRUE)
```

`splitval` *Split-sample-derived Shrinkage After Estimation*

Description

Shrink regression coefficients using a split-sample-derived shrinkage factor.

Usage

```
splitval(dataset, model, nrounds, fract, sdm, int = TRUE, int.adj)
```

Arguments

`dataset`	a dataset for regression analysis. Data should be in the form of a matrix, with the outcome variable as the final column. Application of the `datashape` function beforehand is recommended, especially if categorical predictors are present. For regression with an intercept included a column vector of 1s should be included before the dataset (see examples)
`model`	type of regression model. Either "linear" or "logistic".
`nrounds`	the number of times to replicate the sample splitting process.
`fract`	the fraction of observations designated to the training set
`sdm`	a shrinkage design matrix. For examples, see `ols.shrink`
`int`	logical. If TRUE the model will include a regression intercept.
`int.adj`	logical. If TRUE the regression intercept will be re-estimated after shrinkage of the regression coefficients.

Details

This function applies sample-splitting to a dataset in order to derive a shrinkage factor and apply it to the regression coefficients. Data are randomly split into two sets, a training set and a test set. Regression coefficients are estimated using the training sample, and then a shrinkage factor is estimated using the test set. The mean of N shrinkage factors is then applied to the original regression coeffients, and the regression intercept may be re-estimated.

This process can currently be applied to linear or logistic regression models.

Value

`splitval` returns a list containing the following:

`raw.coeff`	the raw regression model coefficients, pre-shrinkage.
`shrunk.coeff`	the shrunken regression model coefficients
`lambda`	the mean shrinkage factor over Nrounds split-sample replicates
`Nrounds`	the number of rounds of sample splitting
`sdm`	the shrinkage design matrix used to apply the shrinkage factor(s) to the regression coefficients

Examples

```
## Example 1: Linear regression using the iris dataset
## Split-sample-derived shrinkage with 100 rounds of sample-splitting
data(iris)
iris.data <- as.matrix(iris[, 1:4])
iris.data <- cbind(1, iris.data)
sdm1 <- matrix(c(0, 1, 1, 1), nrow = 1)
```

```
set.seed(321)
splitval(dataset = iris.data, model = "linear", nrounds = 100,
fract = 0.75, sdm = sdm1, int = TRUE, int.adj = TRUE)

## Example 2: logistic regression using a subset of the mtcars data
## Split-sample-derived shrinkage
data(mtcars)
mtc.data <- cbind(1,datashape(mtcars, y = 8, x = c(1, 6, 9)))
head(mtc.data)
set.seed(123)
splitval(dataset = mtc.data, model = "logistic",
nrounds = 100, fract = 0.5)
```

sse *Sum of Square Errors*

Description

Compute the sum of squared prediction errors (or residual sum of squares) when a linear model is applied to a dataset.

Usage

```
sse(b, dataset)
```

Arguments

`b`	vector or column-matrix of regression coefficients
`dataset`	a matrix or dataframe. The final column is the outcome variable.

Value

The function returns the sum of square errors.

Examples

```
## Using simulated data derived from the iris dataset
mu <- c(rep(0, 4))
covmatr <- matrix(c(0.7, -0.04, 1.3, 0.5, -0.04, 0.2, -0.3, -0.1,
1.3, -0.3, 3.1, 1.3, 0.5, -0.1, 1.3, 0.6), ncol = 4)
sim.dat <- randnor(n = 100, mu = mu, Cov = covmatr)
sim.dat <- cbind(1, sim.dat)
## resample and fit an ordinary least squares model, and then
## calculate the sum of square errors of the model when applied
## to the original data
sim.boot <- randboot(sim.dat, replace = TRUE)
boot.betas <- ols.rgr(sim.boot)
sse(b = boot.betas, dataset = sim.dat)
```

`strategy` *An Accessory Function for Strategy Selection*

Description

This function acts as a switchboard for various regression modelling strategy, and links these strategies to `compare`.

Usage

```
strategy(g, method, int, int.adj, model)
```

Arguments

`g`	a data matrix to which a modelling strategy will be applied.
`method`	a list containing the a regression modelling strategy name and strategy-specific parameter values.
`int`	logical. If int is TRUE an intercept will be included in the regression model.
`int.adj`	logical. If int.adj is TRUE the intercept will be re-estimated after shrinkage is applied to the regression coefficients.
`model`	the type of regression model. Either "linear" or "logistic".

Details

For further details, see `compare`.

Value

The object returned by `strategy` depends on the strategy selected in the call to `compare`. For example, if `method[1] = "lsq"`, `strategy` will return a matrix of regression coefficients, whereas if `method[1] = "boot"`, strategy will return a list containing values returned by `bootval`.

Note

This function is not designed to be called directly, but acts as a workhorse function for `compare`

Index