Package 'perryExamples'

July 23, 2025

Type Package

Title Examples for Integrating Prediction Error Estimation into Regression Models

Version 0.1.1

Date 2021-11-03

Depends R (>= 2.14.1), perry (>= 0.3.0), robustbase

Imports stats, quantreg, lars

Description
Examples for integrating package 'perry' for prediction error estimation into regression models.

License GPL (>= 2)

LazyLoad yes

Author Andreas Alfons [aut, cre]

Maintainer Andreas Alfons <alfons@ese.eur.nl>

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2021-11-03 11:20:02 UTC

Contents

perryExamples-package *Examples for Integrating Prediction Error Estimation into Regression Models*

Description

Examples for integrating package 'perry' for prediction error estimation into regression models.

Details

The DESCRIPTION file:

Package:	perryExamples
Type:	Package
Title:	Examples for Integrating Prediction Error Estimation into Regression Models
Version:	0.1.1
Date:	2021-11-03
Depends:	R (>= 2.14.1), perry (>= 0.3.0), robustbase
Imports:	stats, quantreg, lars
Description:	Examples for integrating package 'perry' for prediction error estimation into regression models.
License:	GPL (>= 2)
LazyLoad:	yes
Authors@R:	person("Andreas", "Alfons", email = "alfons@ese.eur.nl", role = c("aut", "cre"))
Author:	Andreas Alfons [aut, cre]
Maintainer:	Andreas Alfons <alfons@ese.eur.nl>
Encoding:	UTF-8
RoxygenNote:	7.1.2

Index of help topics:

Author(s)

Andreas Alfons [aut, cre]

Maintainer: Andreas Alfons <alfons@ese.eur.nl>

Bundesliga *Austrian Bundesliga football player data*

Description

The data set contains information on the market value of midfielders and forwards in the Austrian *Bundesliga*, together with player information and performance measures. The data are collected for the (still ongoing) 2013/14 season, with performance measures referring to competitions on the Austrian level (Bundesliga, Cup) or the European level (UEFA Champions League, UEFA Europa League). Only players with complete information are included in the data set.

Usage

```
data("Bundesliga")
```

Format

A data frame with 123 observations on the following 20 variables.

`Player` factor; the player's name.

`Team` factor; the player's team.

`MarketValue` numeric; the player's market value (in Euros).

`Age` numeric; the player's age (in years).

`Height` numeric; the player's height (in cm).

`Foreign` a dummy variable indicating whether the player is foreign or Austrian.

`Forward` a dummy variable indicating whether the player is a forward or midfielder.

`BothFeet` a dummy variable indicating whether the player is equally strong with both feet or has one stronger foot.

`AtClub` numeric; the number of seasons the player is with his current club (at the upcoming transfer window).

`Contract` numeric; the remaining number of seasons in the player's contract (at the upcoming transfer window).

`Matches` numeric; the number of matches in which the player was on the field.

`Goals` numeric; the number of goals the player scored.

`OwnGoals` numeric; the number of own goals the player scored.

`Assists` numeric; the number of assists the player gave.

`Yellow` numeric; the number of yellow cards the player received.

`YellowRed` numeric; the number of times the player was sent off with two yellow cards within one game.

`Red` numeric; the number of times the player was sent off with a red card.

`SubOn` numeric; the number of times the player was substituted on.

`SubOff` numeric; the number of times the player was substituted off.

`Minutes` numeric; the total number of minutes the player was on the field.

Source

The data were scraped from `http://www.transfermarkt.com` on 2014-03-02.

Examples

```
data("Bundesliga")
summary(Bundesliga)
```

`ladlasso` *LAD-lasso with penalty parameter selection*

Description

Fit LAD-lasso models and select the penalty parameter by estimating the respective prediction error via (repeated) K-fold cross-validation, (repeated) random splitting (also known as random subsampling or Monte Carlo cross-validation), or the bootstrap.

Usage

```
ladlasso(
  x,
  y,
  lambda,
  standardize = TRUE,
  intercept = TRUE,
  splits = foldControl(),
  cost = mape,
  selectBest = c("hastie", "min"),
  seFactor = 1,
  ncores = 1,
  cl = NULL,
  seed = NULL,
  ...
)

ladlasso.fit(x, y, lambda, standardize = TRUE, intercept = TRUE, ...)
```

Arguments

`x`	a numeric matrix containing the predictor variables.
`y`	a numeric vector containing the response variable.
`lambda`	for `ladlasso`, a numeric vector of non-negative values to be used as penalty parameter. For `ladlasso.fit`, a single non-negative value to be used as penalty parameter.
`standardize`	a logical indicating whether the predictor variables should be standardized to have unit MAD (the default is `TRUE`).

`intercept` a logical indicating whether a constant term should be included in the model (the default is `TRUE`).

`splits` an object giving data splits to be used for prediction error estimation (see `perryTuning`).

`cost` a cost function measuring prediction loss (see `perryTuning` for some requirements). The default is to use the mean absolute prediction error (see `cost`).

`selectBest, seFactor` arguments specifying a criterion for selecting the best model (see `perryTuning`). The default is to use a one-standard-error rule.

`ncores, cl` arguments for parallel computing (see `perryTuning`).

`seed` optional initial seed for the random number generator (see `.Random.seed` and `perryTuning`).

`...` for `ladlasso`, additional arguments to be passed to the prediction loss function `cost`. For `ladlasso.fit`, additional arguments to be passed to `rq.fit.lasso`.

Value

For `ladlasso`, an object of class `"perryTuning"`, see `perryTuning`). It contains information on the prediction error criterion, and includes the final model with the optimal tuning paramter as component `finalModel`.

For `ladlasso.fit`, an object of class `ladlasso` with the following components:

`lambda` numeric; the value of the penalty parameter.

`coefficients` a numeric vector containing the coefficient estimates.

`fitted.values` a numeric vector containing the fitted values.

`residuals` a numeric vector containing the residuals.

`standardize` a logical indicating whether the predictor variables were standardized to have unit MAD.

`intercept` a logical indicating whether the model includes a constant term.

`muX` a numeric vector containing the medians of the predictors.

`sigmaX` a numeric vector containing the MADs of the predictors.

`muY` numeric; the median of the response.

`call` the matched function call.

Author(s)

Andreas Alfons

References

Wang, H., Li, G. and Jiang, G. (2007) Robust regression shrinkage and consistent variable selection through the LAD-lasso. *Journal of Business & Economic Statistics*, **25**(3), 347–355.

See Also

`perryTuning`, `rq.fit.lasso`

Examples

```
## load data
data("Bundesliga")
Bundesliga <- Bundesliga[, -(1:2)]
f <- log(MarketValue) ~ Age + I(Age^2) + .
mf <- model.frame(f, data=Bundesliga)
x <- model.matrix(terms(mf), mf)[, -1]
y <- model.response(mf)

## set up repeated random splits
splits <- splitControl(m = 40, R = 10)

## select optimal penalty parameter
lambda <- seq(40, 0, length.out = 20)
fit <- ladlasso(x, y, lambda = lambda, splits = splits, seed = 2014)
fit

## plot prediction error results
plot(fit, method = "line")
```

`lasso` *Lasso with penalty parameter selection*

Description

Fit lasso models and select the penalty parameter by estimating the respective prediction error via (repeated) K-fold cross-validation, (repeated) random splitting (also known as random subsampling or Monte Carlo cross-validation), or the bootstrap.

Usage

```
lasso(
  x,
  y,
  lambda = seq(1, 0, length.out = 50),
  mode = c("fraction", "lambda"),
  standardize = TRUE,
  intercept = TRUE,
  splits = foldControl(),
  cost = rmspe,
  selectBest = c("hastie", "min"),
  seFactor = 1,
  ncores = 1,
  cl = NULL,
  seed = NULL,
  ...
)
```

```
lasso.fit(
  x,
  y,
  lambda = seq(1, 0, length.out = 50),
  mode = c("fraction", "lambda"),
  standardize = TRUE,
  intercept = TRUE,
  ...
)
```

Arguments

x	a numeric matrix containing the predictor variables.
y	a numeric vector containing the response variable.
lambda	for lasso, a numeric vector of non-negative values to be used as penalty parameter. For lasso.fit, a single non-negative value to be used as penalty parameter.
mode	a character string specifying the type of penalty parameter. If "fraction", lambda gives the fractions of the smallest value of the penalty parameter that sets all coefficients to 0 (hence all values of lambda should be in the interval [0,1] in that case). If "lambda", lambda gives the grid of values for the penalty parameter directly.
standardize	a logical indicating whether the predictor variables should be standardized to have unit variance (the default is TRUE).
intercept	a logical indicating whether a constant term should be included in the model (the default is TRUE).
splits	an object giving data splits to be used for prediction error estimation (see perryTuning).
cost	a cost function measuring prediction loss (see perryTuning for some requirements). The default is to use the root mean squared prediction error (see cost).
selectBest, seFactor	arguments specifying a criterion for selecting the best model (see perryTuning). The default is to use a one-standard-error rule.
ncores, cl	arguments for parallel computing (see perryTuning).
seed	optional initial seed for the random number generator (see .Random.seed and perryTuning).
...	for lasso, additional arguments to be passed to the prediction loss function cost. For lasso.fit, additional arguments to be passed to lars.

Value

For lasso, an object of class "perryTuning", see perryTuning). It contains information on the prediction error criterion, and includes the final model with the optimal tuning paramter as component finalModel.

For lasso.fit, an object of class lasso with the following components:

lambda numeric; the value of the penalty parameter.

`coefficients` a numeric vector containing the coefficient estimates.

`fitted.values` a numeric vector containing the fitted values.

`residuals` a numeric vector containing the residuals.

`standardize` a logical indicating whether the predictor variables were standardized to have unit variance.

`intercept` a logical indicating whether the model includes a constant term.

`muX` a numeric vector containing the means of the predictors.

`sigmaX` a numeric vector containing the standard deviations of the predictors.

`mu` numeric; the mean of the response.

`call` the matched function call.

Author(s)

Andreas Alfons

References

Tibshirani, R. (1996) Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, **58**(1), 267–288.

See Also

`perryTuning`, `lars`

Examples

```
## load data
data("Bundesliga")
Bundesliga <- Bundesliga[, -(1:2)]
f <- log(MarketValue) ~ Age + I(Age^2) + .
mf <- model.frame(f, data=Bundesliga)
x <- model.matrix(terms(mf), mf)[, -1]
y <- model.response(mf)

## set up repeated random splits
splits <- splitControl(m = 40, R = 10)

## select optimal penalty parameter
fit <- lasso(x, y, splits = splits, seed = 2014)
fit

## plot prediction error results
plot(fit, method = "line")
```

perry-methods *Resampling-based prediction error for fitted models*

Description

Estimate the prediction error of a fitted model via (repeated) K-fold cross-validation, (repeated) random splitting (also known as random subsampling or Monte Carlo cross-validation), or the bootstrap. Methods are available for least squares fits computed with `lm` as well as for the following robust alternatives: MM-type models computed with `lmrob` and least trimmed squares fits computed with `ltsReg`.

Usage

```
## S3 method for class 'lm'
perry(
  object,
  splits = foldControl(),
  cost = rmspe,
  ncores = 1,
  cl = NULL,
  seed = NULL,
  ...
)

## S3 method for class 'lmrob'
perry(
  object,
  splits = foldControl(),
  cost = rtmspe,
  ncores = 1,
  cl = NULL,
  seed = NULL,
  ...
)

## S3 method for class 'lts'
perry(
  object,
  splits = foldControl(),
  fit = c("reweighted", "raw", "both"),
  cost = rtmspe,
  ncores = 1,
  cl = NULL,
  seed = NULL,
  ...
)
```

Arguments

object	the fitted model for which to estimate the prediction error.
splits	an object of class "cvFolds" (as returned by cvFolds) or a control object of class "foldControl" (see foldControl) defining the folds of the data for (repeated) K-fold cross-validation, an object of class "randomSplits" (as returned by randomSplits) or a control object of class "splitControl" (see splitControl) defining random data splits, or an object of class "bootSamples" (as returned by bootSamples) or a control object of class "bootControl" (see bootControl) defining bootstrap samples.
cost	a cost function measuring prediction loss. It should expect the observed values of the response to be passed as the first argument and the predicted values as the second argument, and must return either a non-negative scalar value, or a list with the first component containing the prediction error and the second component containing the standard error. The default is to use the root mean squared prediction error for the "lm" method and the root trimmed mean squared prediction error for the "lmrob" and "lts" methods (see cost).
ncores	a positive integer giving the number of processor cores to be used for parallel computing (the default is 1 for no parallelization). If this is set to NA, all available processor cores are used.
cl	a **parallel** cluster for parallel computing as generated by makeCluster. If supplied, this is preferred over ncores.
seed	optional initial seed for the random number generator (see .Random.seed). Note that also in case of parallel computing, resampling is performed on the manager process rather than the worker processes. On the parallel worker processes, random number streams are used and the seed is set via clusterSetRNGStream.
...	additional arguments to be passed to the prediction loss function cost.
fit	a character string specifying for which fit to estimate the prediction error. Possible values are "reweighted" (the default) for the prediction error of the reweighted fit, "raw" for the prediction error of the raw fit, or "both" for the prediction error of both fits.

Value

An object of class "perry" with the following components:

pe a numeric vector containing the estimated prediction errors. For the "lm" and "lmrob" methods, this is a single numeric value. For the "lts" method, this contains one value for each of the requested fits. In case of more than one replication, those are average values over all replications.

se a numeric vector containing the estimated standard errors of the prediction loss. For the "lm" and "lmrob" methods, this is a single numeric value. For the "lts" method, this contains one value for each of the requested fits.

reps a numeric matrix containing the estimated prediction errors from all replications. For the "lm" and "lmrob" methods, this is a matrix with one column. For the "lts" method, this contains one column for each of the requested fits. However, this is only returned in case of more than one replication.

`splits` an object giving the data splits used to estimate the prediction error.

`y` the response.

`yHat` a list containing the predicted values from all replications.

`call` the matched function call.

Note

The `perry` methods extract the data from the fitted model and call `perryFit` to perform resampling-based prediction error estimation.

Author(s)

Andreas Alfons

See Also

`perryFit`

Examples

```
## load data
data("Bundesliga")
n <- nrow(Bundesliga)

## fit linear model
Bundesliga$logMarketValue <- log(Bundesliga$MarketValue)
fit <- lm(logMarketValue ~ Contract + Matches + Goals + Assists,
          data=Bundesliga)

## perform K-fold cross-validation
perry(fit, foldControl(K = 5, R = 10), seed = 1234)

## perform random splitting
perry(fit, splitControl(m = n/3, R = 10), seed = 1234)

## perform bootstrap prediction error estimation
# 0.632 estimator
perry(fit, bootControl(R = 10, type = "0.632"), seed = 1234)
# out-of-bag estimator
perry(fit, bootControl(R = 10, type = "out-of-bag"), seed = 1234)
```

`ridge` *Ridge regression with penalty parameter selection*

Description

Fit ridge regression models and select the penalty parameter by estimating the respective prediction error via (repeated) K-fold cross-validation, (repeated) random splitting (also known as random subsampling or Monte Carlo cross-validation), or the bootstrap.

Usage

```
ridge(
  x,
  y,
  lambda,
  standardize = TRUE,
  intercept = TRUE,
  splits = foldControl(),
  cost = rmspe,
  selectBest = c("hastie", "min"),
  seFactor = 1,
  ncores = 1,
  cl = NULL,
  seed = NULL,
  ...
)

ridge.fit(x, y, lambda, standardize = TRUE, intercept = TRUE, ...)
```

Arguments

x	a numeric matrix containing the predictor variables.
y	a numeric vector containing the response variable.
lambda	a numeric vector of non-negative values to be used as penalty parameter.
standardize	a logical indicating whether the predictor variables should be standardized to have unit variance (the default is TRUE).
intercept	a logical indicating whether a constant term should be included in the model (the default is TRUE).
splits	an object giving data splits to be used for prediction error estimation (see perryTuning).
cost	a cost function measuring prediction loss (see perryTuning for some requirements). The default is to use the root mean squared prediction error (see cost).
selectBest, seFactor	arguments specifying a criterion for selecting the best model (see perryTuning). The default is to use a one-standard-error rule.
ncores, cl	arguments for parallel computing (see perryTuning).
seed	optional initial seed for the random number generator (see .Random.seed and perryTuning).
...	for ridge, additional arguments to be passed to the prediction loss function cost. For ridge.fit, additional arguments are currently ignored.

Value

For ridge, an object of class "perryTuning", see perryTuning). It contains information on the prediction error criterion, and includes the final model with the optimal tuning paramter as component finalModel.

For ridge.fit, an object of class ridge with the following components:

`lambda` a numeric vector containing the values of the penalty parameter.

`coefficients` a numeric vector or matrix containing the coefficient estimates.

`fitted.values` a numeric vector or matrix containing the fitted values.

`residuals` a numeric vector or matrix containing the residuals.

`standardize` a logical indicating whether the predictor variables were standardized to have unit variance.

`intercept` a logical indicating whether the model includes a constant term.

`muX` a numeric vector containing the means of the predictors.

`sigmaX` a numeric vector containing the standard deviations of the predictors.

`muY` numeric; the mean of the response.

`call` the matched function call.

Author(s)

Andreas Alfons

References

Hoerl, A.E. and Kennard, R.W. (1970) Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, **12**(1), 55–67.

See Also

perryTuning

Examples

```
## load data
data("Bundesliga")
Bundesliga <- Bundesliga[, -(1:2)]
f <- log(MarketValue) ~ Age + I(Age^2) + .
mf <- model.frame(f, data=Bundesliga)
x <- model.matrix(terms(mf), mf)[, -1]
y <- model.response(mf)

## set up repeated random splits
splits <- splitControl(m = 40, R = 10)

## select optimal penalty parameter
lambda <- seq(600, 0, length.out = 50)
fit <- ridge(x, y, lambda = lambda, splits = splits, seed = 2014)
fit

## plot prediction error results
plot(fit, method = "line")
```

TopGearMPG *Top Gear fuel consumption data*

Description

The data set contains information on fuel consumption of cars featured on the website of the popular BBC television show *Top Gear*, together with car information and performance measures. Only cars with complete information are included in the data set.

Usage

```
data("TopGearMPG")
```

Format

A data frame with 255 observations on the following 11 variables.

`Maker` factor; the car maker.

`Model` factor; the car model.

`Type` factor; the exact model type.

`MPG` numeric; the combined fuel consuption (urban + extra urban; in miles per gallon).

`Cylinders` numeric; the number of cylinders in the engine.

`Displacement` numeric; the displacement of the engine (in cc).

`BHP` numeric; the power of the engine (in bhp).

`Torque` numeric; the torque of the engine (in lb/ft).

`Acceleration` numeric; the time it takes the car to get from 0 to 62 mph (in seconds).

`TopSpeed` numeric; the car's top speed (in mph).

`Weight` numeric; the car's curb weight (in kg).

Source

The data were scraped from `http://www.topgear.com/uk/` on 2014-02-24.

Examples

```
data("TopGearMPG")
plot(TopGearMPG[, -(1:3)])
```

Index