Package ‘PredictingBlackSwans’

September 27, 2018

Title Replicate the results of my master thesis 'Predicting Black Swans and Analyzing the Symptoms of their preceeding Imbalances via the Lasso'

Version 1.0

Author Pablo Rodriguez Mira [aut, cre]

Maintainer Pablo Rodriguez Mira <rodriguez.mira.pablo@gmail.com>

Description This package replicates the results of my master thesis.

Depends R (>= 3.4.3), data.table, ggplot2

Imports glmnet, Rcpp, pROC, gtable, randomForest, scales, grid, xtable, grDevices, stargazer, MASS, doMC, nleqslv, reshape, tikzDevice, tseries

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1.9000

LinkingTo Rcpp, RcppArmadillo

R topics documented:

ABias_fun 2
AMSPE_fun 3
analyzingBS 3
AVar_fun 4
cv_nodewise_totalerr 4
despLasso 5
despLassoLowComp 5
despLassoPaper 6
diffFun 7
getZresiduals 7
getZresidualsSQRTL 8
growthFun 8
hamiltonFilter 9

inferenceMeanSePlot . . . 9
inferenceSim . . . 10
inferenceSimMain . . . 10
inferenceSimMainPaper . . . 11
inferenceSimPrintResults . . . 12
interactionsFun . . . 12
JST_rawData . . . 13
lagFun . . . 14
logDiffFun . . . 15
MCCV_Analysis . . . 15
missingValuesAnalysis . . . 16
mysd . . . 16
nodewise_cv . . . 17
nodewise_sqrtlasso . . . 17
nodewise_sqrtlasso_low_comp . . . 18
plotCrisProb . . . 18
plotLinEffects . . . 19
predictingBS . . . 19
predictionSim . . . 20
sqrt_lasso . . . 21
testAnalysis . . . 21

Index **22**

`ABias_fun` *Compute the Squared Average Bias (Squared ABias)*

Description

Computes the squared average bias (Squared ABias) of an estimator in a Monte Carlo simulation study.

Usage

```
ABias_fun(modelFit, truth)
```

Arguments

`modelFit`	A matrix of fitted probabilities of dimensions (number of observations x number of simulations).
`truth`	The true probabilities of the data generating process at x_i, i.e., $\Lambda(x_i^T \beta)$.

Value

A numeric scalar value.

`AMSPE_fun` *Compute the Average Mean Squared Predicion Error (AMSPE) of an estimator*

Description

Computes the Average Mean Squared Prediction Error (AMSPE) of an estimator in a Monte Carlo simulation study.

Usage

```
AMSPE_fun(modelFit, truth)
```

Arguments

`modelFit`	A matrix of fitted probabilities of dimensions (number of observations x number of simulations).
`truth`	The true probabilities of the data generating process at x_i, i.e., $\Lambda(x_i^T \beta)$.

Value

A numeric scalar value.

`analyzingBS` *Analyzing Black Swans*

Description

Main function to reproduce the results of our paper regarding the inference analysis.

Usage

```
analyzingBS(path = getwd(), cvfolds = 5, seed = 813, parallel = T,
  ncores = 2L)
```

Arguments

`path`	Path to export the results
`cvfolds`	Number of folds for the cross-validation in the desparsified Lasso for the initial estimator (Lasso).
`seed`	Seed for reproducibility.
`parallel`	Should parallel computing be used? (Only for UNIX computers)
`ncores`	Number of cores for parallel-computing.

Details

A new directory "Inference_Analysis" will be created where the results will be placed in.

AVar_fun *Compute the Average Variance (AVariance) of an estimator*

Description

Computes the Average Variance (AVariance) of an estimator in a Monte Carlo simulations study.

Usage

```
AVar_fun(modelFit)
```

Arguments

modelFit	A matrix of fitted probabilities of dimensions (number of observations x number of simulations).

Value

A numeric scalar value.

cv_nodewise_totalerr *Compute the total cross-validated error for the nodewise regression*

Description

Compute the total cross-validated error for the nodewise regression

Usage

```
cv_nodewise_totalerr(c, dataselects, x, lambdas, K)
```

Arguments

c	Column of the response in the nodewise regression.
dataselects	Fold index for the cross-validation.
x	Matrix of predictors.
lambdas	Sequence of regularization parameters.
K	Number of folds for the cross-validation.

Value

A (Number of lambdas x Number of folds) matrix of cross-validated errors (error on the discarded fold).

`despLasso` *Compute the Desparsified Lasso Estimator*

Description

Compute the Desparsified Lasso Estimator for Logistic Regression

Usage

```
despLasso(x, y, nodewise = c("cv", "sqrtLasso"), lambda = c("BCW", "VdG"),
  cvfolds = 5, parallel = TRUE, ncores = getOption("mc.cores", 2L))
```

Arguments

`x`	Matrix of predictors.
`y`	Response variable.
`nodewise`	Either 'cv' for the nodewise regression using the Lasso with cross-validation or 'sqrtLasso' for the square-root Lasso.
`lambda`	Tuning parameter for the square-root Lasso in the nodewise regressions. Either 'BCW' for the proposal by Belloni, Chernozhukov and Wang (2011) or 'VdG' for the proposal by van de Geer (2014).
`cvfolds`	Number of folds for the cross-validation for both the initial estimator and for the CV-nodewise-Lasso (if this is chosen).
`parallel`	Should parallel computing be used when possible?
`ncores`	Number of cores to be used when parallel is TRUE.

`despLassoLowComp` *Compute the Desparsified Lasso Estimator for a Low Dimensional Component*

Description

Compute the Desparsified Lasso Estimator for Logistic Regression for a low dimensional subset of variables of interest. The square-root Lasso with Belloni, Chernozhukov & Wang (2011) tuning parameter is used.

Usage

```
despLassoLowComp(x, y, cvfolds, lowSel = seq(1, ncol(x)), parallel = TRUE,
  ncores = getOption("mc.cores", 2L))
```

Arguments

`x`	Matrix of predictors.
`y`	Response variable.
`cvfolds`	Number of folds for the cross-validation for the initial estimator.
`lowSel`	Low dimensional selection of variables of interest for which the desparsified Lasso will yield estimates.
`parallel`	Should parallel computing be used when possible?
`ncores`	Number of cores to be used when parallel is TRUE.

`despLassoPaper` *Compute the Desparsified Lasso Estimator as in the original paper by van de Geer (2014)*

Description

Compute the Desparsified Lasso Estimator for Logistic Regression with the scaling parameter sigmahat as in the original paper (van de Geer (2014)) using the "desparsified outer product of the gradient" of the loss function instead of the Hessian used in our preferred implementation.

Usage

```
despLassoPaper(x, y, nodewise = c("cv", "sqrtLasso"), lambda = c("BCW",
  "VdG"), cvfolds = 5, parallel = TRUE, ncores = getOption("mc.cores",
  2L))
```

Arguments

`x`	Matrix of predictors.
`y`	Response variable.
`nodewise`	Method to use for the nodewise Lasso: either 'cv' for Lasso with cross-validation or 'sqrtLasso' for the square root nodewise Lasso or c('cv', 'sqrtLasso') for both estimators of the approximate inverse matrix.
`lambda`	Tuning parameter rule for the square-root nodewise Lasso. Either BCW for the rule after Bernoulli, Chernozhukov & Wang (2011) or VdG for Van de Geer (2014).
`cvfolds`	Number of folds for the cross-validation for both the initial estimator and for the CV-nodewise-Lasso (if this is chosen).
`parallel`	Should parallel computing be used when possible?
`ncores`	Number of cores to be used when parallel is TRUE.

diffFun — *Compute time differences*

Description

Compute time differences y_t - t_t-1.

Usage

```
diffFun(inputData, inputVars)
```

Arguments

`inputData`	A data.table.
`inputVars`	Variables for which the time differences will be calculated.

Value

Additional variables in the dataset. These have the names of the input variables with a "_diff"-ending.

Examples

```
diffFun(data=dat, inputVars=c("gdp", "revenue"))
```

getZresiduals — *Compute the residuals of the nodewise Lasso regressions*

Description

Compute the residuals of the nodewise Lasso regressions

Usage

```
getZresiduals(i, x, lambda)
```

Arguments

`i`	Column index of the response.
`x`	Matrix of predictors.
`lambda`	Regularization parameter.

Value

Vector of residuals.

getZresidualsSQRTL — *Compute the residuals of the square-root Lasso regressions*

Description

Compute the residuals of the square-root Lasso regressions

Usage

```
getZresidualsSQRTL(j, x, lambda)
```

Arguments

`j`	Column index for the response.
`x`	Matrix of predictors.
`lambda`	Either 'BCW' for the simulations method proposed by Belloni, Chernozhukov and Wang (2011) or 'VdG' for the proposed method in van de Geer (2014).

Value

Vector of residuals.

growthFun — *Compute growth rates*

Description

Compute growth rates as (y_t - y_t-1) / y_t-1.

Usage

```
growthFun(data, inputVars)
```

Arguments

`data`	A data.table.
`inputVars`	Variables for which the growth rates will be calculated.

Value

Additional variables in the data.table. These have the names of the input variables with a "_gr"-ending.

Examples

```
growthFun(data=dat, inputVars=c("gdp", "revenue"))
```

`hamiltonFilter` *Implementation of the Hamilton (2017) filter*

Description

Compute the deviations from trend for a time series using the Hamilton filter for each country separately.

Usage

```
hamiltonFilter(inputData, inputVar, h = 3)
```

Arguments

`inputData`	A data.table.
`inputVar`	The variable for detrending.
`h`	Horizon for which we build a prediction.

Value

A new data.table with an additional variable. This variable has the name of the input variable with a '_dt'-ending.

`inferenceMeanSePlot` *Plot the standard errors of the estimate against the estimates*

Description

Plot the Monte-Carlo standard errors of the estimate (y-axis) against the Monte-Carlo estimates (x-axis) to assess the drivers of the inference results regarding the worse coverage with simultaneouly better power properties of the tests.

Usage

```
inferenceMeanSePlot(path)
```

Arguments

`path`	Path to look for the input-files.

inferenceSim *Inference Simulation Study*

Description

Replicate the results of our simulations part regarding inference. IMPORTANT NOTE: This function is thought to be run as a script. See the details.

Usage

```
inferenceSim(path = getwd(), parallel = T, ncores = getOption("mc.cores",
  2L), seed = 912)
```

Arguments

`path`	Path to export the results.
`parallel`	Should parallel computing be used? Note: It only works for UNIX systems.
`ncores`	How many cores should be used for parallel computing?
`seed`	Seed for reproducibility purposes.

Details

This function may take to long to run for computers with few kernels or for Windows-computers. Therefore we suggest to run this function as an script to split the computation of the simulations in several days.

Examples

```
inferenceSim()
```

inferenceSimMain *Main function for the inference simulation study*

Description

Computes all the results for the simulation study with our preferred implementation using the "desparsified Hessian" to estimate the standard error.

Usage

```
inferenceSimMain(path = getwd(), n = 100, p = 150, rho = 0.5,
  nSim = 100, nomSize = 0.05, cvfolds = 5, seed = 182, parallel = T,
  ncores = getOption("mc.cores", 2L))
```

Arguments

`path`	Path to export the results.
`n`	Number of observations.
`p`	Number of predictors.
`rho`	Correlation parameter of the Toeplitz covariance matrix.
`nSim`	Number of simulations.
`nomSize`	Nominal size (type I error a.k.a. alpha).
`cvfolds`	Number of folds for the cross-validations.
`seed`	Seed for replication purposes.
`parallel`	Should parallel computing be used? Note: It only works for UNIX systems.
`ncores`	How many cores should be used for parallel computing?

Examples

```
inferenceSimMain()
```

`inferenceSimMainPaper` *Main function for the inference simulation study using the "desparsified outer product of the gradient"*

Description

Computes all the results for the simulation study using the "desparsified outer product of the gradient" as in the original paper (van de Geer (2014)).

Usage

```
inferenceSimMainPaper(path = getwd(), n = 100, p = 150, rho = 0.5,
  nSim = 100, nomSize = 0.05, cvfolds = 5, seed = 182, parallel = T,
  ncores = getOption("mc.cores", 2L))
```

Arguments

`path`	Path to export the results.
`n`	Number of observations.
`p`	Number of predictors.
`rho`	Correlation parameter of the Toeplitz covariance matrix.
`nSim`	Number of simulations.
`nomSize`	Nominal size (type I error a.k.a. alpha).
`cvfolds`	Number of folds for the cross-validations.
`seed`	Seed for replication purposes.
`parallel`	Should parallel computing be used? Note: It only works for UNIX systems.
`ncores`	How many cores should be used for parallel computing?

inferenceSimPrintResults *Print the inference simulation results in Latex format*

Description

Print the results of our simulation study for inference in Latex format as in the master thesis.

Usage

```
inferenceSimPrintResults(inPath, outPath)
```

Arguments

`inPath`	Path to look for the input-files.
`outPath`	Path to export the results.

interactionsFun *Compute interactions terms*

Description

Compute interactions terms

Usage

```
interactionsFun(inputData, inputVars)
```

Arguments

`inputData`	A data.table.
`inputVars`	Variables for which the interactions will be computed.

Value

Additional variables in the dataset. These have the names of the pairs input variables with an "_i_" in between.

Examples

```
interactionsFun(data=dat, inputVars=c("gdp", "revenue"))
```

JST_rawData *Dataset for the application part: The Jordà-Schularick-Taylor Macrohistory Database*

Description

An extensive data collection containing macroeconomic data for 17 advanced economies since 1870 on an annual basis. This data set captures the near-universe of advanced-country macroeconomic and asset price dynamics, covering on average over 90 percent of advanced-economy output and over 50 percent of world output.

Format

A data frame with 2499 rows and 29 variables:

year Year

country Country

iso ISO 3-letter code

ifs IFS 3-number country-code

pop Population

rgdpmad Real GDP per capita (PPP)

rgdppc Real GDP per capita (index, 2005=100)

rconpc Real consumption per capita (index, 2006=100)

gdp GDP (nominal, local currency)

iy Investment-to-GDP ratio

cpi Consumer prices (index, 1990=100)

ca Current account (nominal, local currency)

imports Imports (nominal, local currency)

exports Exports (nominal, local currency)

narrowm Narrow money (nominal, local currency)

money Broad money (nominal, local currency)

stir Short-term interest rate (nominal, percent per year)

ltrate Long-term interest rate (nominal, percent per year)

stocks Stock prices (nominal index)

debtgdp Public debt-to-GDP ratio

revenue Government revenues (nominal, local currency)

expenditure Government expenditure (nominal, local currency)

xrusd USD exchange rate (local currency / USD)

crisisJST Systemic financial crises indicator (0 = No Crisis; 1 = Crisis)

tloans Total loans to non-financial private sector (nominal, local currency)

tmort Mortgage loans to non-financial private sector (nominal, local currency)

thh Total loans to households (nominal, local currency)

tbus Total loans to business (nominal, local currency)

hpnom House prices (nominal index, 1990=100)

Source

http://www.macrohistory.net/data/. Download date: 30.07.2018.

`lagFun` *Compute lags of variables*

Description

Compute lags of variables

Usage

```
lagFun(inputData, inputVars, maxLag)
```

Arguments

`inputData`	A data.table.
`inputVars`	Variables for which the lags will be computed.
`maxLag`	Number up to which lags will be computed. E.g. if maxLag = 3 then the first, second and third lag will be calculated.

Value

Additional variables in the data.table. These will have the names of the input variables with an "_jL"-ending, with j = 1, ..., maxAve.

Examples

```
lagFun(inputData=dat, inputVars=c("gdp", "revenue"), maxLag=3)
```

`logDiffFun` *Compute log-differences*

Description

Compute log-differences $log(y_t) - log(y_{t-1})$

Usage

```
logDiffFun(inputData, inputVars)
```

Arguments

`inputData`	A data.table.
`inputVars`	Variables for which the log-differences will be calculated.

Value

Additional variables in the data.table. These have the names of the input variables with a "_lDiff"-ending.

Examples

```
logDiff(data=dat, inputVars=c("gdp", "revenue"))
```

`MCCV_Analysis` *Monte Carlo Cross-Validation Analysis for the performance comparison*

Description

Main function to reproduce the results of our paper regarding the performance comparison by means of Monte Carlo Cross-Validation (MCCV).

Usage

```
MCCV_Analysis(mccvnumber = 100, cvfolds = 5, seed = 813, parallel = T,
  ncores = 2L)
```

Arguments

`mccvnumber`	Number of Monte Carlo Cross-Validation runs.
`cvfolds`	Number of folds for the cross-validation with the Lasso.
`seed`	Seed for reproducibility.
`parallel`	Should paralle computing be used? (Note: Only for UNIX computers).
`ncores`	Number of cores.

missingValuesAnalysis *Missing Values Analysis*

Description

Output missing values information.

Usage

```
missingValuesAnalysis(path)
```

Arguments

path	Path to export the results.

mysd *Auxiliary function to compute the standard deviation with factor (1 / n)*

Description

Auxiliary function to compute the standard deviation with factor (1 / n)

Usage

```
mysd(y)
```

Arguments

y	A vector.

Value

The standard deviation of the vector y with factor (1 / n) instead of the default in base R (1 / (n- 1)).

nodewise_cv *Compute the nodewise Lasso using K-fold cross-validation*

Description

Compute the nodewise Lasso using K-fold cross-validation and output the matrix of residuals Z stemming from the nodewise regressions.

Usage

```
nodewise_cv(x, parallel = TRUE, ncores = 2L, lambda = "lambda.min",
  cvfolds = 5)
```

Arguments

x	Predictor matrix.
parallel	Should parallel computing be used?
ncores	Number of cores for parallel computing.
lambda	Either "lambda.1se" or "lambda.min" defined as in the glmnet-Package. See e.g. ?glmnet::cv.glmnet.
cvfolds	Number of folds for the cross-validation.

Value

The matrix of residuals of the nodewise Lasso regressions, i.e. $Z = (Z_1, ..., Z_p)$ with $Z_j \in R^n$.

nodewise_sqrtlasso *Compute the nodewise Lasso using the square-root Lasso*

Description

Compute the nodewise Lasso using the square-root Lasso to calculate the matrix of residuals Z stemming from the nodewise regressions.

Usage

```
nodewise_sqrtlasso(x, parallel = TRUE, ncores = 2L, lambda = "BCW")
```

Arguments

x	Predictor matrix.
parallel	Should parallel computing be used?
ncores	Number of cores for parallel computing.
lambda	Either 'BCW' for the simulations method proposed by Belloni, Chernozhukov and Wang (2011) or 'VdG' for the proposed method in van de Geer (2014).

Value

The matrix of residuals of the nodewise square-root Lasso regressions, i.e. $Z = (Z_1, ..., Z_p)$ with $Z_j \in R^n$.

---

nodewise_sqrtlasso_low_comp

Compute the nodewise Lasso using the square-root Lasso for a low dimensional component

---

Description

Compute the nodewise Lasso using the square-root Lasso to calculate the matrix of residuals Z stemming from the nodewise regressions for a low dimensional component.

Usage

```
nodewise_sqrtlasso_low_comp(x, lowSel = 1:ncol(x), parallel = TRUE,
  ncores = 2L, lambda = "BCW")
```

Arguments

`x`	Predictor matrix.
`lowSel`	Indixes of the low-dimensional selection of variables to desparsify.
`parallel`	Should parallel computing be used?
`ncores`	Number of cores for parallel computing.
`lambda`	Either 'BCW' for the simulations method proposed by Belloni, Chernozhukov and Wang (2011) or 'VdG' for the proposed method in van de Geer (2014).

Value

The matrix of residuals of the nodewise square-root Lasso regressions, i.e. $Z = (Z_1, ..., Z_p)$ with $Z_j \in R^n$.

---

plotCrisProb

Plot Crisis Probabilities for each Country

Description

Plot crisis probabilities for each country in the sample.

Usage

```
plotCrisProb(crisDT, fullDT, countryName)
```

Arguments

`crisDT`	A data table with the countries, years and predicted probability.
`fullDT`	The full data set.
`countryName`	Name of the country to plot. It can also be "All".

Value

A ggplot.

`plotLinEffects` *Plot Linear Effects driving Crisis Probabilities*

Description

Plot of the linear effects driving the crisis probabilities.

Usage

```
plotLinEffects(driverTable, fullDT, countryName)
```

Arguments

`driverTable`	A data.table with the countries, years and linear effects.
`fullDT`	The full data set.
`countryName`	Name of the country to plot.

Value

A ggplot.

`predictingBS` *Predicting Black Swans*

Description

Main function to reproduce the results of our paper regarding the prediction analysis.

Usage

```
predictingBS(path = getwd(), mccvnumber = 100, cvfolds = 5, seed = 813,
  parallel = T, ncores = 2L)
```

Arguments

`path`	Path to export the results
`mccvnumber`	Number of Monte Carlo Cross-Validations.
`cvfolds`	Number of folds for the Lasso.
`seed`	Seed for reproducibility.
`parallel`	Should parallel computing be used? (Only for UNIX computers)
`ncores`	Number of cores for parallel computing.

Details

A new directory "Prediction_Analysis" will be created where the results will be placed in.

`predictionSim` *Prediction Simulation Study*

Description

Replicate the results of our simulations part regarding prediction accuracy.

Usage

```
predictionSim(path = getwd(), n = 100, pList = c(80, 150, 500),
  coefConfig = c("big", "small"), rho = 0.9, nSim = 100, cvfolds = 5,
  parallel = TRUE, ncores = 2L, seed = 182)
```

Arguments

`path`	Path to export the results.
`n`	Number of observations.
`pList`	Vector of number of covariates for each scenario.
`coefConfig`	Either 'big' or 'small' or c('big', 'small') for the scenarios corresponding to big or small coefficients.
`rho`	Correlation strength of the Toeplitz covariance matrix.
`nSim`	Number of simulations.
`cvfolds`	Number of folds for the cross-validation.
`parallel`	Should parallel computing be used when possible?
`ncores`	Number of cores for parallel computing.
`seed`	Seed for replication purposes.

sqrt_lasso *Compute the square-root Lasso*

Description

Compute the square-root Lasso solution and its residuals for the nodewise Lasso.

Usage

```
sqrt_lasso(y, X, lambda)
```

Arguments

`y`	Response variable.
`X`	Matrix of regressors.
`lambda`	Regularization parameter.

Details

The software is adapted from the Matlab-software provided by Belloni, Chernozhukov and Wang (2011) in https://faculty.fuqua.duke.edu/~abn5/belloni-software.html

testAnalysis *Test Analysis*

Description

Main function to reproduce the results of our paper regarding the test analysis in the prediction part.

Usage

```
testAnalysis(mccvnumber = 100, cvfolds = 5, seed = 813, parallel = T,
  ncores = 2L)
```

Arguments

`mccvnumber`	Number of Monte Carlo Cross-Validation runs.
`cvfolds`	Number of folds for the cross-validation with the Lasso.
`seed`	Seed for reproducibility.
`parallel`	Should parallel computing be used? (Note: Only for UNIX computers)
`ncores`	Number of cores for parallel computing

Index

∗Topic **PredictingBlackSwans**
- ABias_fun, 2
- AMSPE_fun, 3
- analyzingBS, 3
- AVar_fun, 4
- diffFun, 7
- growthFun, 8
- hamiltonFilter, 9
- inferenceMeanSePlot, 9
- inferenceSim, 10
- inferenceSimMain, 10
- inferenceSimMainPaper, 11
- inferenceSimPrintResults, 12
- interactionsFun, 12
- lagFun, 14
- logDiffFun, 15
- MCCV_Analysis, 15
- missingValuesAnalysis, 16
- plotCrisProb, 18
- plotLinEffects, 19
- predictingBS, 19
- predictionSim, 20
- testAnalysis, 21

ABias_fun, 2
AMSPE_fun, 3
analyzingBS, 3
AVar_fun, 4

cv_nodewise_totalerr, 4

despLasso, 5
despLassoLowComp, 5
despLassoPaper, 6
diffFun, 7

getZresiduals, 7
getZresidualsSQRTL, 8
growthFun, 8

hamiltonFilter, 9

inferenceMeanSePlot, 9
inferenceSim, 10
inferenceSimMain, 10
inferenceSimMainPaper, 11
inferenceSimPrintResults, 12
interactionsFun, 12

JST_rawData, 13

lagFun, 14
logDiffFun, 15

MCCV_Analysis, 15
missingValuesAnalysis, 16
mysd, 16

nodewise_cv, 17
nodewise_sqrtlasso, 17
nodewise_sqrtlasso_low_comp, 18

plotCrisProb, 18
plotLinEffects, 19
predictingBS, 19
predictionSim, 20

sqrt_lasso, 21

testAnalysis, 21