Package 'FRESA.CAD'

February 2, 2018

Type Package

Title Feature Selection Algorithms for Computer Aided Diagnosis

Version 3.0.1

Date 2018-02-01

Author Jose Gerardo Tamez-Pena, Antonio Martinez-Torteya and Israel Alanis

Maintainer Jose Gerardo Tamez-Pena <jose.tamezpena@itesm.mx>

Description Contains a set of utilities for building and testing formula-based models (linear, logistic or COX) for Computer Aided Diagnosis/Prognosis applications. Utilities include data adjustment, univariate analysis, model building, model-validation, longitudinal analysis, reporting and visualization.

License LGPL (>= 2)

Depends Rcpp (>= 0.10.0),stringr,miscTools,Hmisc,pROC

LinkingTo Rcpp, RcppArmadillo

Suggests nlme,rpart,gplots,RColorBrewer,class,cvTools,glmnet,survival,e1071

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-02-01 23:16:36 UTC

R topics documented:

`FRESA.CAD-package` *FeatuRE Selection Algorithms for Computer-Aided Diagnosis (FRESA.CAD)*

Description

Contains a set of utilities for building and testing formula-based models for Computer Aided Diagnosis/prognosis applications via feature selection. Bootstrapped Stage Wise Model Selection (B:SWiMS) controls the false selection (FS) for linear, logistic, or Cox proportional hazards regression models. Utilities include functions for: univariate/longitudinal analysis, data conditioning (i.e. covariate adjustment and normalization), model validation and visualization.

Details

Package:	FRESA.CAD
Type:	Package
Version:	3.0.1
Date:	2018-02-01
License:	LGPL (>= 2)

Purpose: The design of diagnostic or prognostic multivariate models via the selection of significantly discriminant features. The models are selected via the bootstrapped step-wise selection of model features that offer a significant improvement in subject classification/error. The false selection control is achieved by train-test partitions, where train sets are used to select variables and test sets used to evaluate model performance. Variables that do not improve subject classification/error on the blind test are not included in the models. The main function of this package is the selection and cross-validation of diagnostic/prognostic linear, logistic, or Cox proportional hazards regression model constructed from a large set of candidate features. The variable selection may start by conditioning all variables via a covariate-adjustment and a z-inverse-rank-transformation. In order to integrate features with partial discriminant power, the package can be used to categorize the continuous variables and rank their discriminant power. Once ranked, each feature is bootstrap-tested in a multivariate model, and its blind performance is evaluated. Variables with a statistical significant improvement in classification/error are stored and finally inserted into the final model according to their relative store frequency. A cross-validation procedure may be used to diagnose the amount of model shrinkage produced by the selection scheme.

Author(s)

Jose Gerardo Tamez-Pena, Antonio Martinez-Torteya and Israel Alanis Maintainer: <jose.tamezpena@itesm.mx>

References

Pencina, M. J., D'Agostino, R. B., & Vasan, R. S. (2008). Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond. *Statistics in medicine* **27**(2), 157-172.

Examples

```
## Not run:
library("epiR")
library("FRESA.CAD")
library(network)
library(GGally)
library("e1071")

# Get the stage C prostate cancer data from the rpart package

library(rpart)
data(stagec)
# Split the stages into several columns
dataCancer <- cbind(stagec[,c(1:3,5:6)],
```

```
                        gleason4 = 1*(stagec[,7] == 4),
                        gleason5 = 1*(stagec[,7] == 5),
                        gleason6 = 1*(stagec[,7] == 6),
                        gleason7 = 1*(stagec[,7] == 7),
                        gleason8 = 1*(stagec[,7] == 8),
                        gleason910 = 1*(stagec[,7] >= 9),
                        eet = 1*(stagec[,4] == 2),
                        diploid = 1*(stagec[,8] == "diploid"),
                        tetraploid = 1*(stagec[,8] == "tetraploid"),
                        notAneuploid = 1-1*(stagec[,8] == "aneuploid"))


#Impute missing values
dataCancerImputed <- nearestneighborimpute(dataCancer)

# Remove the incomplete cases
    dataCancer <- dataCancer[complete.cases(dataCancer),]


    # Load a pre-stablished data frame with the names and descriptions of all variables
    data(cancerVarNames)
    # the Heat Map
    hm <- heatMaps(cancerVarNames,varRank=NULL,Outcome="pgstat",
                   data=dataCancer,title="Heat Map",hCluster=FALSE
                   ,prediction=NULL,Scale=TRUE,
                   theFiveColors=c("blue","cyan","black","yellow","red"),
                   outcomeColors =
                   c("blue","lightgreen","yellow","orangered","red"),
                   transpose=FALSE,cexRow=0.50,cexCol=0.80,srtCol=35)

    # The univariate analysis
    UniRankFeaturesRaw <- univariateRankVariables(variableList = cancerVarNames,
                                                  formula = "pgstat ~ 1+pgtime",
                                                  Outcome = "pgstat",
                                                  data = dataCancer,
                                                  categorizationType = "Raw",
                                                  type = "LOGIT",
                                                  rankingTest = "zIDI",
                                                  description = "Description",
                                                  uniType="Binary")

    print(UniRankFeaturesRaw)

    # A simple BSIWMS Model

    BSWiMSModel <- BSWiMS.model(formula = Surv(pgtime, pgstat) ~ 1, dataCancerImputed)

    # The Log-Rank Analysis using survdiff

lrsurvdiff <- survdiff(Surv(pgtime,pgstat)~
                       BSWiMSModel$BSWiMS.model$back.model$linear.predictors > 0,
   data=dataCancerImputed)
```

```
  # The Log-Rank Analysis EmpiricalSurvDiff and permutations of the null Chi distribution
    lrp <- EmpiricalSurvDiff(dataCancerImputed$pgtime,dataCancerImputed$pgstat,
                         BSWiMSModel$BSWiMS.model$back.model$linear.predictors > 0,
                         type="Chi",plots=TRUE,samples=10000)

# The Log-Rank Analysis EmpiricalSurvDiff and permutations of the null SLR distribution
    lrp <- EmpiricalSurvDiff(dataCancerImputed$pgtime,dataCancerImputed$pgstat,
                         BSWiMSModel$BSWiMS.model$back.model$linear.predictors > 0,
                         type="SLR",plots=TRUE,samples=10000)

# The Log-Rank Analysis EmpiricalSurvDiff and bootstrapping the SLR distribution
lrp <- EmpiricalSurvDiff(dataCancerImputed$pgtime,dataCancerImputed$pgstat,
                          BSWiMSModel$BSWiMS.model$back.model$linear.predictors > 0,
                         computeDist=TRUE,plots=TRUE)

   #The performance of the final model using the summary function
   sm <- summary(BSWiMSModel$BSWiMS.model$back.model)
   print(sm$coefficients)
   pv <- plot(sm$bootstrap)

   # The equivalent model
   eq <- reportEquivalentVariables(BSWiMSModel$BSWiMS.model$back.model,data=dataCancer,
                                   variableList=cancerVarNames,Outcome = "pgstat",
                                   timeOutcome="pgtime",
                                   type = "COX");

   print(eq$equivalentMatrix)

   #The list of all models of the bootstrap forward selection
   print(BSWiMSModel$forward.selection.list)

   #With FRESA.CAD we can do a leave-one-out using the list of models
   pm <- ensemblePredict(BSWiMSModel$forward.selection.list,
                       dataCancer,predictType = "linear",type="LOGIT",Outcome="pgstat")

   #Ploting the ROC with 95
   pm <- plotModels.ROC(cbind(dataCancer$pgstat,
                     pm$ensemblePredict),main=("LOO Forward Selection Median Predict"))

   #The plotModels.ROC provides the diagnosis confusion matrix.
   summary(epi.tests(pm$predictionTable))



   #FRESA.CAD can be used to create a bagged model using the forward selection formulas
   bagging <- baggedModel(BSWiMSModel$forward.selection.list,dataCancer,useFreq=32)
   pm <- predict(bagging$bagged.model)
   pm <- plotModels.ROC(cbind(dataCancer$pgstat,pm),main=("Bagged"))

   #Let's check the performance of the model
   sm <- summary(bagging$bagged.model)
   print(sm$coefficients)
```

```
#Using bootstrapping object I can check the Jaccard Index
print(bagging$Jaccard.SM)

#Ploting the evolution of the coefficient value
plot(bagging$coefEvolution$grade,main="Evolution of grade")


gplots::heatmap.2(bagging$formulaNetwork,trace="none",
                mar=c(10,10),main="eB:SWIMS Formula Network")
barplot(bagging$frequencyTable,las = 2,cex.axis=1.0,
        cex.names=0.75,main="Feature Frequency")
n <- network::network(bagging$formulaNetwork, directed = FALSE,
                      ignore.eval = FALSE,names.eval = "weights")
ggnet2(n, label = TRUE, size = "degree",size.cut = 3,size.min = 1,
       mode = "circle",edge.label = "weights",edge.label.size=4)


# Get a Cox proportional hazards model using:
# - The default parameters

mdCOXs <- FRESA.Model(formula = Surv(pgtime, pgstat) ~ 1,data = dataCancer)
sm <- summary(mdCOXs$BSWiMS.model)
print(sm$coefficients)
sm <- summary(mdCOXs$eBSWiMS.model$equivalentModel)
print(sm$coefficients)

# The model with singificant improvement in the residual error
mdCOXs <- FRESA.Model(formula = Surv(pgtime, pgstat) ~ 1,
                      data = dataCancer,OptType = "Residual" )
sm <- summary(mdCOXs$BSWiMS.model)
print(sm$coefficients)

# Get a Cox proportional hazards model using second order models:
mdCOX <- FRESA.Model(formula = Surv(pgtime, pgstat) ~ 1,
                     data = dataCancer,categorizationType="RawRaw")
sm <- summary(mdCOX$BSWiMS.model)
print(sm$coefficients)

namesc <- names(mdCOX$BSWiMS.model$coefficients)[-1]
hm <- heatMaps(mdCOX$univariateAnalysis[namesc,],varRank=NULL,
               Outcome="pgstat",data=dataCancer,
               title="Heat Map",hCluster=FALSE,prediction=NULL,Scale=TRUE,
               theFiveColors=c("blue","cyan","black","yellow","red"),
               outcomeColors = c("blue","lightgreen","yellow","orangered","red"),
               transpose=FALSE,cexRow=0.50,cexCol=0.80,srtCol=35)

# The LOO estimation
pm <- ensemblePredict(mdCOX$BSWiMS.models$formula.list,dataCancer,
                predictType = "linear",type="LOGIT",Outcome="pgstat")
pm <- plotModels.ROC(cbind(dataCancer$pgstat,pm$ensemblePredict),main=("LOO Median Predict"))
#Let us check the diagnosis performance
summary(epi.tests(pm$predictionTable))
```

```
# Get a Logistic model using FRESA.Model
# - The default parameters
dataCancer2 <-dataCancer
dataCancer2$pgtime <-NULL
mdLOGIT <- FRESA.Model(formula = pgstat ~ 1,data = dataCancer2)
if (!is.null(mdLOGIT$bootstrappedModel)) pv <- plot(mdLOGIT$bootstrappedModel)
sm <- summary(mdLOGIT$BSWiMS.model)
print(sm$coefficients)


## FRESA.Model with Cross Validation and Recursive Partitioning and Regression Trees


md <- FRESA.Model(formula = Surv(pgtime, pgstat) ~ 1,data = dataCancer,
              CVfolds = 20,repeats = 5,equivalent = TRUE,usrFitFun=rpart)

pm <- plotModels.ROC(md$cvObject$LASSO.testPredictions,theCVfolds=20,main="CV LASSO",cex=0.90)
pm <- plotModels.ROC(md$cvObject$KNN.testPrediction,theCVfolds=20,main="KNN",cex=0.90)
pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                 predictor="Prediction",main="B:SWiMS Bagging",cex=0.90)
pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
               predictor="Median",main="Forward Selection Median Ensemble",cex=0.90)
pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                 predictor="Bagged",main="Forward Selection Bagging",cex=0.90)
pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                 predictor="Forward",main="Forward Model",cex=0.90)
pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                 predictor="first.B.SWiMS",main="The First B:SWiMS Model",cex=0.90)
pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                 predictor="eB.SWiMS",main="The Equivalent B.SWiMS",cex=0.90)
pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
               predictor="Ensemble.B.SWiMS",main="B:SWiMS Median Ensemble",cex=0.90)

md$cvObject$Models.testPrediction$usrFitFunction <-
   md$cvObject$Models.testPrediction$usrFitFunction -0.5
md$cvObject$Models.testPrediction$usrFitFunction_Sel <-
   md$cvObject$Models.testPrediction$usrFitFunction_Sel -0.5

pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                 predictor="usrFitFunction",
                 main="Recursive Partitioning and Regression Trees",cex=0.90)
pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                 predictor="usrFitFunction_Sel",
                main="Recursive Partitioning and Regression Trees with FS",cex=0.90)


## FRESA.Model with Cross Validation, LOGISTIC and Support Vector Machine


md <- FRESA.Model(formula = pgstat ~ 1,data = dataCancer2,
                  CVfolds = 20,repeats = 5,equivalent = TRUE,usrFitFun=svm)

pm <- plotModels.ROC(md$cvObject$LASSO.testPredictions,theCVfolds=20,main="CV LASSO",cex=0.90)
```

```
pm <- plotModels.ROC(md$cvObject$KNN.testPrediction,theCVfolds=20,main="KNN",cex=0.90)
 pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                  predictor="Prediction",main="B:SWiMS Bagging",cex=0.90)
 pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                predictor="Median",main="Forward Selection Median Ensemble",cex=0.90)
 pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                  predictor="Bagged",main="Forward Selection Bagging",cex=0.90)
 pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                  predictor="Forward",main="Forward Model",cex=0.90)
 pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                  predictor="first.B.SWiMS",main="The First B:SWiMS Model",cex=0.90)
 pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                  predictor="eB.SWiMS",main="The Equivalent B.SWiMS",cex=0.90)
 pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                predictor="Ensemble.B.SWiMS",main="B:SWiMS Median Ensemble",cex=0.90)
 md$cvObject$Models.testPrediction$usrFitFunction <-
    md$cvObject$Models.testPrediction$usrFitFunction -0.5
 md$cvObject$Models.testPrediction$usrFitFunction_Sel <-
    md$cvObject$Models.testPrediction$usrFitFunction_Sel -0.5
 pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                  predictor="usrFitFunction",
                  main="Support Vector Machine",cex=0.90)
 pm <- plotModels.ROC(md$cvObject$Models.testPrediction,theCVfolds=20,
                  predictor="usrFitFunction_Sel",
                  main="Support Vector Machine with FS",cex=0.90)

## End(Not run)
```

---

backVarElimination_Bin

IDI/NRI-based backwards variable elimination

Description

This function removes model terms that do not significantly affect the integrated discrimination improvement (IDI) or the net reclassification improvement (NRI) of the model.

Usage

```
backVarElimination_Bin(object,
                  pvalue = 0.05,
                  Outcome = "Class",
                  data,
                  startOffset = 0,
                  type = c("LOGIT", "LM", "COX"),
                  selectionType = c("zIDI", "zNRI"),
   adjsize= 1)
```

Arguments

`object`	An object of class `lm`, `glm`, or `coxph` containing the model to be analyzed
`pvalue`	The maximum *p*-value, associated to either IDI or NRI, allowed for a term in the model
`Outcome`	The name of the column in `data` that stores the variable to be predicted by the model
`data`	A data frame where all variables are stored in different columns
`startOffset`	Only terms whose position in the model is larger than the `startOffset` are candidates to be removed
`type`	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
`selectionType`	The type of index to be evaluated by the `improveProb` function (`Hmisc` package): z-score of IDI or of NRI
`adjsize`	The expected size of a random model for multiple selection correction

Details

For each model term x_i, the IDI or NRI is computed for the Full model and the reduced model(where the term x_i removed). The term whose removal results in the smallest drop in improvement is selected. The hypothesis: the term adds classification improvement is tested by checking the pvalue of improvement. If $p(IDIorNRI) > pvalue$, then the term is removed. In other words, only model terms that significantly aid in subject classification are kept. The procedure is repeated until no term fulfils the removal criterion.

Value

`back.model`	An object of the same class as `object` containing the reduced model
`loops`	The number of loops it took for the model to stabilize
`reclas.info`	A list with the NRI and IDI statistics of the reduced model, as given by the `getVar.Bin` function
`back.formula`	An object of class `formula` with the formula used to fit the reduced model
`lastRemoved`	The name of the last term that was removed (-1 if all terms were removed)
`at.opt.model`	the model before the BH procedure
`beforeFSC.formula`	the string formula of the model before the BH procedure

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

References

Pencina, M. J., D'Agostino, R. B., & Vasan, R. S. (2008). Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond. *Statistics in medicine* **27**(2), 157-172.

See Also

backVarElimination_Res, bootstrapVarElimination_Bin, bootstrapVarElimination_Res

---

backVarElimination_Res

NeRI-based backwards variable elimination

Description

This function removes model terms that do not significantly improve the "net residual" (NeRI)

Usage

```
backVarElimination_Res(object,
                       pvalue = 0.05,
                       Outcome = "Class",
                       data,
                       startOffset = 0,
                       type = c("LOGIT", "LM", "COX"),
                       testType = c("Binomial", "Wilcox", "tStudent", "Ftest"),
                       setIntersect = 1,
   adjsize= 1)
```

Arguments

object	An object of class lm, glm, or coxph containing the model to be analyzed
pvalue	The maximum *p*-value, associated to the NeRI, allowed for a term in the model
Outcome	The name of the column in data that stores the variable to be predicted by the model
data	A data frame where all variables are stored in different columns
startOffset	Only terms whose position in the model is larger than the startOffset are candidates to be removed
type	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
testType	Type of non-parametric test to be evaluated by the improvedResiduals function: Binomial test ("Binomial"), Wilcoxon rank-sum test ("Wilcox"), Student's *t*-test ("tStudent"), or *F*-test ("Ftest")
setIntersect	The intersect of the model (To force a zero intersect, set this value to 0)
adjsize	The expected size of a random model for multiple selection correction

Details

For each model term x_i, the residuals are computed for the Full model and the reduced model(where the term x_i removed). The term whose removal results in the smallest drop in residuals improvement is selected. The hypothesis: the term improves residuals is tested by checking the pvalue of improvement. If $p(residualsbetterthanreducedresiduals) > pvalue$, then the term is removed. In other words, only model terms that significantly aid in improving residuals are kept. The procedure is repeated until no term fulfils the removal criterion. The p-values of improvement can be computed via a sign-test (Binomial) a paired Wilcoxon test, paired t-test or f-test. The first three tests compare the absolute values of the residuals, while the f-test test if the variance of the residuals is improved significantly.

Value

`back.model`	An object of the same class as `object` containing the reduced model
`loops`	The number of loops it took for the model to stabilize
`reclas.info`	A list with the NeRI statistics of the reduced model, as given by the `getVar.Res` function
`back.formula`	An object of class `formula` with the formula used to fit the reduced model
`lastRemoved`	The name of the last term that was removed (-1 if all terms were removed)
`at.opt.model`	the model with before the FSR procedure.
`beforeFSC.formula`	the string formula of the the FSR procedure

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

`backVarElimination_Bin`, `bootstrapVarElimination_Bin` `bootstrapVarElimination_Res`

`baggedModel` *Get the bagged model from a list of models*

Description

This function will take the frequency-ranked of variables and the list of models to create a single bagged model

Usage

```
baggedModel(modelFormulas,
            data,
            type=c("LM","LOGIT","COX"),
            Outcome=NULL,
            timeOutcome=NULL,
            frequencyThreshold=0.025,
            univariate=NULL,
useFreq=TRUE,
n_bootstrap=1
            )
```

Arguments

modelFormulas	The name of the column in data that stores the variable to be predicted by the model
data	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
type	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
Outcome	The name of the column in data that stores the time to outcome
timeOutcome	The name of the column in data that stores the time to event (needed only for a Cox proportional hazards regression model fitting)
frequencyThreshold	set the frequency the threshold of the frequency of features to be included in the model)
univariate	The FFRESA.CAD univariate analysis matrix
useFreq	Use the feature frequency to order the formula terms. If set to a positive value is the number of estimation loops
n_bootstrap	if greater than 1, defines the number of bootstraps samples to be used

Value

bagged.model	the bagged model
formula	the formula of the model
frequencyTable	the table of variables ranked by their model frequency
faverageSize	the average size of the models
formulaNetwork	The matrix of interaction between formulas
Jaccard.SM	The Jaccard Stability Measure of the formulas
coefEvolution	The evolution of the coefficients
avgZvalues	The average Z value of each coefficient

Author(s)

Jose G. Tamez-Pena

See Also

ensemblePredict

`bootstrapValidation_Bin` *Bootstrap validation of binary classification models*

Description

This function bootstraps the model *n* times to estimate for each variable the empirical distribution of model coefficients, area under ROC curve (AUC), integrated discrimination improvement (IDI) and net reclassification improvement (NRI). At each bootstrap the non-observed data is predicted by the trained model, and statistics of the test prediction are stored and reported. The method keeps track of predictions and plots the bootstrap-validated ROC. It may plots the blind test accuracy, sensitivity, and specificity, contrasted with the bootstrapped trained distributions.

Usage

```
bootstrapValidation_Bin(fraction = 1,
                    loops = 200,
                    model.formula,
                    Outcome,
                    data,
                    type = c("LM", "LOGIT", "COX"),
                    plots = FALSE,
best.model.formula=NULL)
```

Arguments

`fraction`	The fraction of data (sampled with replacement) to be used as train
`loops`	The number of bootstrap loops
`model.formula`	An object of class `formula` with the formula to be used
`Outcome`	The name of the column in `data` that stores the variable to be predicted by the model
`data`	A data frame where all variables are stored in different columns
`type`	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
`plots`	Logical. If `TRUE`, density distribution plots are displayed
`best.model.formula`	An object of class `formula` with the formula to be used for the best model

Details

The bootstrap validation will estimate the confidence interval of the model coefficients and the NRI and IDI. The non-sampled values will be used to estimate the blind accuracy, sensitivity, and specificity. A plot to monitor the evolution of the bootstrap procedure will be displayed if `plots` is set to TRUE. The plot shows the train and blind test ROC. The density distribution of the train accuracy, sensitivity, and specificity are also shown, with the blind test results drawn along the y-axis.

Value

`data`	The data frame used to bootstrap and validate the model
`outcome`	A vector with the predictions made by the model
`blind.accuracy`	The accuracy of the model in the blind test set
`blind.sensitivity`	The sensitivity of the model in the blind test set
`blind.specificity`	The specificity of the model in the blind test set
`train.ROCAUC`	A vector with the AUC in the bootstrap train sets
`blind.ROCAUC`	An object of class `roc` containing the AUC in the bootstrap blind test set
`boot.ROCAUC`	An object of class `roc` containing the AUC using the mean of the bootstrapped coefficients
`fraction`	The fraction of data that was sampled with replacement
`loops`	The number of loops it took for the model to stabilize
`base.Accuracy`	The accuracy of the original model
`base.sensitivity`	The sensitivity of the original model
`base.specificity`	The specificity of the original model
`accuracy`	A vector with the accuracies in the bootstrap test sets
`sensitivities`	A vector with the sensitivities in the bootstrap test sets
`specificities`	A vector with the specificities in the bootstrap test sets
`train.accuracy`	A vector with the accuracies in the bootstrap train sets
`train.sensitivity`	A vector with the sensitivities in the bootstrap train sets
`train.specificity`	A vector with the specificities in the bootstrap train sets
`s.coef`	A matrix with the coefficients in the bootstrap train sets
`boot.model`	An object of class `lm`, `glm`, or `coxph` containing a model whose coefficients are the median of the coefficients of the bootstrapped models
`boot.accuracy`	The accuracy of the `mboot.model` model
`boot.sensitivity`	The sensitivity of the `mboot.model` model

`boot.specificity`
: The specificity of the `mboot.model` model

`z.NRIs`
: A matrix with the z-score of the NRI for each model term, estimated using the bootstrap train sets

`z.IDIs`
: A matrix with the z-score of the IDI for each model term, estimated using the bootstrap train sets

`test.z.NRIs`
: A matrix with the z-score of the NRI for each model term, estimated using the bootstrap test sets

`test.z.IDIs`
: A matrix with the z-score of the IDI for each model term, estimated using the bootstrap test sets

`NRIs`
: A matrix with the NRI for each model term, estimated using the bootstrap test sets

`IDIs`
: A matrix with the IDI for each model term, estimated using the bootstrap test sets

`testOutcome`
: A vector that contains all the individual outcomes used to validate the model in the bootstrap test sets

`testPrediction`
: A vector that contains all the individual predictions used to validate the model in the bootstrap test sets

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

`bootstrapValidation_Res`, `plot.bootstrapValidation_Bin`, `summary.bootstrapValidation_Bin`

---

bootstrapValidation_Res

Bootstrap validation of regression models

Description

This function bootstraps the model n times to estimate for each variable the empirical bootstrapped distribution of model coefficients, and net residual improvement (NeRI). At each bootstrap the non-observed data is predicted by the trained model, and statistics of the test prediction are stores and reported.

Usage

```
bootstrapValidation_Res(fraction = 1,
                        loops = 200,
                        model.formula,
                        Outcome,
                        data,
```

```
                        type = c("LM", "LOGIT", "COX"),
                        plots = FALSE,
bestmodel.formula=NULL)
```

Arguments

`fraction`	The fraction of data (sampled with replacement) to be used as train
`loops`	The number of bootstrap loops
`model.formula`	An object of class `formula` with the formula to be used
`Outcome`	The name of the column in `data` that stores the variable to be predicted by the model
`data`	A data frame where all variables are stored in different columns
`type`	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
`plots`	Logical. If `TRUE`, density distribution plots are displayed
`bestmodel.formula`	An object of class `formula` with the best formula to be compared

Details

The bootstrap validation will estimate the confidence interval of the model coefficients and the NeRI. It will also compute the train and blind test root-mean-square error (RMSE), as well as the distribution of the NeRI *p*-values.

Value

`data`	The data frame used to bootstrap and validate the model
`outcome`	A vector with the predictions made by the model
`boot.model`	An object of class `lm`, `glm`, or `coxph` containing a model whose coefficients are the median of the coefficients of the bootstrapped models
`NeRIs`	A matrix with the NeRI for each model term, estimated using the bootstrap test sets
`tStudent.pvalues`	A matrix with the *t*-test *p*-value of the NeRI for each model term, estimated using the bootstrap train sets
`wilcox.pvalues`	A matrix with the Wilcoxon rank-sum test *p*-value of the NeRI for each model term, estimated using the bootstrap train sets
`bin.pvalues`	A matrix with the binomial test *p*-value of the NeRI for each model term, estimated using the bootstrap train sets
`F.pvalues`	A matrix with the *F*-test *p*-value of the NeRI for each model term, estimated using the bootstrap train sets
`test.tStudent.pvalues`	A matrix with the *t*-test *p*-value of the NeRI for each model term, estimated using the bootstrap test sets
`test.wilcox.pvalues`	A matrix with the Wilcoxon rank-sum test *p*-value of the NeRI for each model term, estimated using the bootstrap test sets

`test.bin.pvalues`
: A matrix with the binomial test *p*-value of the NeRI for each model term, estimated using the bootstrap test sets

`test.F.pvalues`
: A matrix with the *F*-test *p*-value of the NeRI for each model term, estimated using the bootstrap test sets

`testPrediction`
: A vector that contains all the individual predictions used to validate the model in the bootstrap test sets

`testOutcome`
: A vector that contains all the individual outcomes used to validate the model in the bootstrap test sets

`testResiduals`
: A vector that contains all the residuals used to validate the model in the bootstrap test sets

`trainPrediction`
: A vector that contains all the individual predictions used to validate the model in the bootstrap train sets

`trainOutcome`
: A vector that contains all the individual outcomes used to validate the model in the bootstrap train sets

`trainResiduals`
: A vector that contains all the residuals used to validate the model in the bootstrap train sets

`testRMSE`
: The global RMSE, estimated using the bootstrap test sets

`trainRMSE`
: The global RMSE, estimated using the bootstrap train sets

`trainSampleRMSE`
: A vector with the RMSEs in the bootstrap train sets

`testSampledRMSE`
: A vector with the RMSEs in the bootstrap test sets

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

`bootstrapValidation_Bin`, `plot.bootstrapValidation_Res`

---

bootstrapVarElimination_Bin

IDI/NRI-based backwards variable elimination with bootstrapping

---

Description

This function removes model terms that do not improve the bootstrapped integrated discrimination improvement (IDI) or net reclassification improvement (NRI) significantly.

Usage

```
bootstrapVarElimination_Bin(object,
                        pvalue = 0.05,
                        Outcome = "Class",
                        data,
                        startOffset = 0,
                        type = c("LOGIT", "LM", "COX"),
                        selectionType = c("zIDI", "zNRI"),
                        loops = 250,
                        fraction = 1.0,
                        print=TRUE,
                        plots=TRUE,
                        adjsize=1,
                        uniAdjPvalues=NULL
                        )
```

Arguments

`object`	An object of class `lm`, `glm`, or `coxph` containing the model to be analyzed
`pvalue`	The maximum *p*-value, associated to either IDI or NRI, allowed for a term in the model
`Outcome`	The name of the column in `data` that stores the variable to be predicted by the model
`data`	A data frame where all variables are stored in different columns
`startOffset`	Only terms whose position in the model is larger than the `startOffset` are candidates to be removed
`type`	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
`selectionType`	The type of index to be evaluated by the `improveProb` function (`Hmisc` package): z-score of IDI or of NRI
`loops`	The number of bootstrap loops
`fraction`	The fraction of data (sampled with replacement) to be used as train
`print`	Logical. If `TRUE`, information will be displayed
`plots`	Logical. If `TRUE`, plots are displayed
`adjsize`	The expected size of a random model for mutiple selection correction
`uniAdjPvalues`	The univariate p-values

Details

For each model term x_i, the IDI or NRI is computed for the Full model and the reduced model(where the term x_i removed). The term whose removal results in the smallest drop in bootstrapped improvement is selected. The hypothesis: the term adds classification improvement is tested by checking the p value of average improvement. If $p(IDIorNRI) > pvalue$, then the term is removed. In other words, only model terms that significantly aid in subject classification are kept. The procedure is repeated until no term fulfils the removal criterion.

Value

back.model	An object of the same class as object containing the reduced model
loops	The number of loops it took for the model to stabilize
reclas.info	A list with the NRI and IDI statistics of the reduced model, as given by the getVar.Bin function
bootCV	An object of class bootstrapValidation_Bin containing the results of the bootstrap validation in the reduced model
back.formula	An object of class formula with the formula used to fit the reduced model
lastRemoved	The name of the last term that was removed (-1 if all terms were removed)
at.opt.model	The model will have the fitted model that had close to maximum bootstrapped test accuracy
beforeFSC.formula	The formula of the model before False Selection Correction
at.Accuracy.formula	the string formula of the model that had the best or close to tbe best test accuracy

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

References

Pencina, M. J., D'Agostino, R. B., & Vasan, R. S. (2008). Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond. *Statistics in medicine* **27**(2), 157-172.

See Also

bootstrapVarElimination_Res, backVarElimination_Bin, backVarElimination_Res

---

bootstrapVarElimination_Res

NeRI-based backwards variable elimination with bootstrapping

Description

This function removes model terms that do not improve the bootstrapped net residual improvement (NeRI) significantly.

Usage

```
bootstrapVarElimination_Res(object,
                            pvalue = 0.05,
                            Outcome = "Class",
                            data,
                            startOffset = 0,
                            type = c("LOGIT", "LM", "COX"),
                            testType = c("Binomial",
                                         "Wilcox",
                                         "tStudent",
                                         "Ftest"),
                            loops = 250,
                            fraction = 1.0,
                            setIntersect = 1,
                            print=TRUE,
                            plots=TRUE,
                            adjsize= 1,
                            uniAdjPvalues=NULL
                                )
```

Arguments

`object`	An object of class `lm`, `glm`, or `coxph` containing the model to be analysed
`pvalue`	The maximum *p*-value, associated to the NeRI, allowed for a term in the model
`Outcome`	The name of the column in `data` that stores the variable to be predicted by the model
`data`	A data frame where all variables are stored in different columns
`startOffset`	Only terms whose position in the model is larger than the `startOffset` are candidates to be removed
`type`	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
`testType`	Type of non-parametric test to be evaluated by the `improvedResiduals` function: Binomial test ("Binomial"), Wilcoxon rank-sum test ("Wilcox"), Student's *t*-test ("tStudent"), or *F*-test ("Ftest")
`loops`	The number of bootstrap loops
`fraction`	The fraction of data (sampled with replacement) to be used as train
`setIntersect`	The intersect of the model (To force a zero intersect, set this value to 0)
`print`	Logical. If `TRUE`, information will be displayed
`plots`	Logical. If `TRUE`, plots are displayed
`adjsize`	The expected size of a random model for multiple selection correction
`uniAdjPvalues`	The univariate p-values

Details

For each model term x_i, the residuals are computed for the Full model and the reduced model(where the term x_i removed). The term whose removal results in the smallest drop in bootstrapped test residuals improvement is selected. The hypothesis: the term improves residuals is tested by checking the p-value of average improvement. If $p(residualsbetterthanreducedresiduals) > pvalue$, then the term is removed. In other words, only model terms that significantly aid in improving residuals are kept. The procedure is repeated until no term fulfils the removal criterion. The p-values of improvement can be computed via a sign-test (Binomial) a paired Wilcoxon test, paired t-test or f-test. The first three tests compare the absolute values of the residuals, while the f-test test if the variance of the residuals is improved significantly.

Value

`back.model`	An object of the same class as `object` containing the reduced model
`loops`	The number of loops it took for the model to stabilize
`reclas.info`	A list with the NeRI statistics of the reduced model, as given by the `getVar.Res` function
`bootCV`	An object of class `bootstrapValidation_Res` containing the results of the bootstrap validation in the reduced model
`back.formula`	An object of class `formula` with the formula used to fit the reduced model
`lastRemoved`	The name of the last term that was removed (-1 if all terms were removed)
`at.opt.model`	The model with close to minimum bootstrapped RMSE
`beforeFSC.formula`	The formula of the model before the FSC stage
`at.RMSE.formula`	the string formula of the model that had the minimum or close to minimum RMSE

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

`bootstrapVarElimination_Bin`, `backVarElimination_Res`, `bootstrapValidation_Res`

`BSWiMS.model` *BSWiMS model selection*

Description

This function returns a set of models that best predict the outcome. Based on a Bootstrap Stage Wise Model Selection algorithm.

Usage

```
BSWiMS.model(formula,
             data,
             type = c("Auto","LM","LOGIT","COX"),
             testType = c("Auto","zIDI",
                          "zNRI",
                          "Binomial",
                          "Wilcox",
                          "tStudent",
                          "Ftest"),
             pvalue=0.05,
             elimination.pValue=0.05,
             update.pvalue=c(0.05,0.05),
             variableList=NULL,
             size=0,
             loops=32,
             elimination.bootstrap.steps = 200,
             unitPvalues=NULL,
             adjsize=0.0,
             fraction=1.0,
             maxTrainModelSize=20,
             maxCycles=10,
             print=FALSE,
             plots=FALSE
             )
```

Arguments

formula	An object of class formula with the formula to be fitted
data	A data frame where all variables are stored in different columns
type	The fit type. Auto will determine the fitting based on the formula
testType	For an Binary-based optimization, the type of index to be evaluated by the improveProb function (Hmisc package): z-value of Binary or of NRI. For a NeRI-based optimization, the type of non-parametric test to be evaluated by the improvedResiduals function: Binomial test ("Binomial"), Wilcoxon rank-sum test ("Wilcox"), Student's t-test ("tStudent"), or F-test ("Ftest")
pvalue	The maximum p-value, associated to the testType, allowed for a term in the model (it will control the false selection rate)
elimination.pValue	The p-value for back elimination
update.pvalue	The p-value for update forward selection
variableList	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
size	The number of candidate variables to be tested (the first size variables from variableList)

loops	The number of bootstrap loops for the forward selection procedure
elimination.bootstrap.steps	The number of bootstrap loops for the backwards elimination procedure
unitPvalues	The univariate pvalue of the association of each feature to the outcome
adjsize	The expected size of a random model for mutiple selection correction
fraction	The fraction of data (sampled with replacement) to be used as train
maxTrainModelSize	Maximum number of terms that can be included in the each forward selection model
maxCycles	The maximum number of model generation cycles
print	Logical. If TRUE, information will be displayed
plots	Logical. If TRUE, plots are displayed

Details

This is a core function of FRESA.CAD. The function will generate a set of B:SWiMS models from the data based on the provided baseline formula. The function will loop extracting a models whose all terms are statistical significant. After each loop it will remove the significant terms, and it will repeat the model generation until no mode significant models are found or the maximum number of cycles is reached.

Value

BSWiMS.model	the output of the bootstrap backwards elimination step
forward.model	The output of the forward selection step
update.model	The output of the forward selection step
univariate	The univariate ranking of variables if no list of features was provided
bagging	The model after bagging the set of models
formula.list	The formulas extracted at each cycle
forward.selection.list	All formulas generated by the forward selection procedure

Author(s)

Jose G. Tamez-Pena

References

Pencina, M. J., D'Agostino, R. B., & Vasan, R. S. (2008). Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond. *Statistics in medicine* **27**(2), 157-172.

Examples

```
## Not run:
library(rpart)
data(stagec)
# Split the gleason into several columns

dataCancer <- cbind(stagec[,c(1:3,5:6)],
gleason4 = 1*(stagec[,7] == 4),
gleason5 = 1*(stagec[,7] == 5),
gleason6 = 1*(stagec[,7] == 6),
gleason7 = 1*(stagec[,7] == 7),
gleason8 = 1*(stagec[,7] == 8),
gleason910 = 1*(stagec[,7] >= 9),
eet = 1*(stagec[,4] == 2),
diploid = 1*(stagec[,8] == "diploid"),
tetraploid = 1*(stagec[,8] == "tetraploid"),
notAneuploid = 1-1*(stagec[,8] == "aneuploid"))

#Impute missing values
dataCancerImputed <- nearestneighborimpute(dataCancer)
# A simple B:SWiMS Model

BSWiMSModel <- BSWiMS.model(formula = Surv(pgtime, pgstat) ~ 1, dataCancerImputed)

#The performance of the final model
sm <- summary(BSWiMSModel$BSWiMS.model$back.model)
print(sm$coefficients)
#The ROC plot and Survival Analysis
pv <- plot(sm$bootstrap)
#The plot provides the diagnosis confusion matrix.
library("epiR")
summary(epi.tests(pv$diagnosticMatrix))


## End(Not run)
```

`cancerVarNames` *Data frame used in several examples of this package*

Description

This data frame contains two columns, one with names of variables, and the other with descriptions of such variables. It is used in several examples of this package. Specifically, it is used in examples working with the stage C prostate cancer data from the `rpart` package

Usage

```
data(cancerVarNames)
```

Format

A data frame with names and descriptions of the variables used in several examples

`Var` A column with the names of the variables

`Description` A column with a short description of the variables

Examples

```
data(cancerVarNames)
```

`crossValidationFeatureSelection_Bin`

IDI/NRI-based selection of a linear, logistic, or Cox proportional hazards regression model from a set of candidate variables

---

Description

This function performs a cross-validation analysis of a feature selection algorithm based on the integrated discrimination improvement (IDI) or the net reclassification improvement (NRI) to return a predictive model. It is composed of an IDI/NRI-based feature selection followed by an update procedure, ending with a bootstrapping backwards feature elimination. The user can control how many train and blind test sets will be evaluated.

Usage

```
crossValidationFeatureSelection_Bin(size = 10,
                                    fraction = 1.0,
                                    pvalue = 0.05,
                                    loops = 100,
                                    covariates = "1",
                                    Outcome,
                                    timeOutcome = "Time",
                                    variableList,
                                    data,
                                    maxTrainModelSize = 20,
                                    type = c("LM", "LOGIT", "COX"),
                                    selectionType = c("zIDI", "zNRI"),
                                    startOffset = 0,
                                    elimination.bootstrap.steps = 25,
                                    trainFraction = 0.67,
                                    trainRepetition = 9,
                                    elimination.pValue = 0.05,
                                    bootstrap.steps = 25,
                                    nk = 0,
                                    unirank = NULL,
                                    print=TRUE,
                                    plots=TRUE,
```

```
                                    lambda="lambda.1se",
                                    adjsize=1.0,
                                    equivalent=FALSE,
                                    bswimsCycles=10,
                                    usrFitFun=NULL
)
```

Arguments

`size`	The number of candidate variables to be tested (the first `size` variables from `variableList`)
`fraction`	The fraction of data (sampled with replacement) to be used as train
`pvalue`	The maximum *p*-value, associated to either IDI or NRI, allowed for a term in the model
`loops`	The number of bootstrap loops
`covariates`	A string of the type "1 + var1 + var2" that defines which variables will always be included in the models (as covariates)
`Outcome`	The name of the column in `data` that stores the variable to be predicted by the model
`timeOutcome`	The name of the column in `data` that stores the time to event (needed only for a Cox proportional hazards regression model fitting)
`variableList`	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
`data`	A data frame where all variables are stored in different columns
`maxTrainModelSize`	Maximum number of terms that can be included in the model
`type`	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
`selectionType`	The type of index to be evaluated by the `improveProb` function (`Hmisc` package): *z*-score of IDI or of NRI
`startOffset`	Only terms whose position in the model is larger than the `startOffset` are candidates to be removed
`elimination.bootstrap.steps`	The number of bootstrap loops for the backwards elimination procedure
`trainFraction`	The fraction of data (sampled with replacement) to be used as train for the cross-validation procedure
`trainRepetition`	The number of cross-validation folds (it should be at least equal to $1/$`trainFraction` for a complete cross-validation)
`elimination.pValue`	The maximum *p*-value, associated to either IDI or NRI, allowed for a term in the model by the backward elimination procedure
`bootstrap.steps`	The number of bootstrap loops for the confidence intervals estimation

nk	The number of neighbours used to generate a *k*-nearest neighbours (KNN) classification. If zero, *k* is set to the square root of the number of cases. If less than zero, it will not perform the KNN classification
unirank	A list with the results yielded by the uniRankVar function, required only if the rank needs to be updated during the cross-validation procedure
print	Logical. If TRUE, information will be displayed
plots	Logical. If TRUE, plots are displayed
lambda	The passed value to the s parameter of the glmnet cross validation coefficient
adjsize	The expected size of a random model for multiple selection correction
equivalent	Is set to TRUE CV will compute the equivalent model
bswimsCycles	The maximum number of models to be returned by BSWiMS.model
usrFitFun	A user fitting function to be evaluated by the cross validation procedure

Details

This function produces a set of data and plots that can be used to inspect the degree of over-fitting or shrinkage of a model. It uses bootstrapped data, cross-validation data, and, if possible, retrain data. During each cycle, a train and a test ROC will be generated using bootstrapped data. At the end of the cross-validation feature selection procedure, a set of three plots may be produced depending on the specifications of the analysis. The first plot shows the ROC for each cross-validation blind test. The second plot, if enough samples are given, shows the ROC of each model trained and tested in the blind test partition. The final plot shows ROC curves generated with the train, the bootstrapped blind test, and the cross-validation test data. Additionally, this plot will also contain the ROC of the cross-validation mean test data, and of the cross-validation coherence. These set of plots may be used to get an overall perspective of the expected model shrinkage. Along with the plots, the function provides the overall performance of the system (accuracy, sensitivity, and specificity). The function also produces a report of the expected performance of a KNN algorithm trained with the selected features of the model, and an elastic net algorithm. The test predictions obtained with these algorithms can then be compared to the predictions generated by the logistic, linear, or Cox proportional hazards regression model.

Value

formula.list	A list containing objects of class formula with the formulas used to fit the models found at each cycle
Models.testPrediction	A data frame with the blind test set predictions (Full B:SWiMS,Median,Bagged,Forward,Backwards Eliminations) made at each fold of the cross validation, where the models used to generate such predictions (formula.list) were generated via a feature selection process which included only the train set. It also includes a column with the Outcome of each prediction, and a column with the number of the fold at which the prediction was made.
FullBSWiMS.testPrediction	A data frame similar to Models.testPrediction, but where the model used to generate the predictions was the Full model, generated via a feature selection process which included all data.

`TestRetrained.blindPredictions`
: A data frame similar to `Models.testPrediction`, but where the models were retrained on an independent set of data (only if enough samples are given at each fold)

`LastTrainBSWiMS.bootstrapped`
: An object of class `bootstrapValidation_Bin` containing the results of the bootstrap validation in the last trained model

`Test.accuracy`
: The global blind test accuracy of the cross-validation procedure

`Test.sensitivity`
: The global blind test sensitivity of the cross-validation procedure

`Test.specificity`
: The global blind test specificity of the cross-validation procedure

`Train.correlationsToFull`
: The Spearman ρ rank correlation coefficient between the predictions made with each model from `formula.list` and the Full model in the train set

`Blind.correlationsToFull`
: The Spearman ρ rank correlation coefficient between the predictions made with each model from `formula.list` and the Full model in the test set

`FullModelAtFoldAccuracies`
: The blind test accuracy for the Full model at each cross-validation fold

`FullModelAtFoldSpecificties`
: The blind test specificity for the Full model at each cross-validation fold

`FullModelAtFoldSensitivities`
: The blind test sensitivity for the Full model at each cross-validation fold

`FullModelAtFoldAUC`
: The blind test ROC AUC for the Full model at each cross-validation fold

`AtCVFoldModelBlindAccuracies`
: The blind test accuracy for the Full model at each final cross-validation fold

`AtCVFoldModelBlindSpecificities`
: The blind test specificity for the Full model at each final cross-validation fold

`AtCVFoldModelBlindSensitivities`
: The blind test sensitivity for the Full model at each final cross-validation fold

`CVTrain.Accuracies`
: The train accuracies at each fold

`CVTrain.Sensitivity`
: The train sensitivity at each fold

`CVTrain.Specificity`
: The train specificity at each fold

`CVTrain.AUCs`
: The train ROC AUC for each fold

`Models.CVblindMeanSensitivites`
: The mean ROC sensitivities at certain specificities for all test final cross-validation folds (i.e. 1.00, 0.95, 0.90, 0.80, 0.70, 0.60, 0.50, 0.40, 0.30, 0.20, 0.10, 0.05, and 0.00)

`forwardSelection`
: A list containing the values returned by `ForwardSelection.Model.Bin` using all data

`updateforwardSelection`
: A list containing the values returned by `updateModel.Bin` using all data and the model from `forwardSelection`

`BSWiMS`
: A list containing the values returned by `bootstrapVarElimination_Bin` using all data and the model from `updateforwardSelection`

`FullBSWiMS.bootstrapped`
: An object of class `bootstrapValidation_Bin` containing the results of the bootstrap validation in the Full model

`Models.testSensitivities`
: A matrix with the mean ROC sensitivities at certain specificities for each train and all test cross-validation folds using the cross-validation models (i.e. 0.95, 0.90, 0.80, 0.70, 0.60, 0.50, 0.40, 0.30, 0.20, 0.10, and 0.05)

`FullKNN.testPrediction`
: A data frame similar to `Models.testPrediction`, but where a KNN classifier with the same features as the Full model was used to generate the predictions

`KNN.testPrediction`
: A data frame similar to `Models.testPrediction`, but where KNN classifiers with the same features as the cross-validation models were used to generate the predictions at each cross-validation fold

`Fullenet`
: An object of class `cv.glmnet` containing the results of an elastic net cross-validation fit

`LASSO.testPredictions`
: A data frame similar to `Models.testPrediction`, but where the predictions were made by the elastic net model

`LASSOVariables`
: A list with the elastic net Full model and the models found at each cross-validation fold

`uniTrain.Accuracies`
: The list of accuracies of an univariate analysis on each one of the model variables in the train sets

`uniTest.Accuracies`
: The list of accuracies of an univariate analysis on each one of the model variables in the test sets

`uniTest.TopCoherence`
: The accuracy coherence of the top ranked variable on the test set

`uniTrain.TopCoherence`
: The accuracy coherence of the top ranked variable on the train set

`Models.trainPrediction`
: A data frame with the outcome and the train prediction of every model

`FullBSWiMS.trainPrediction`
: A data frame with the outcome and the train prediction at each CV fold for the main model

`LASSO.trainPredictions`
: A data frame with the outcome and the prediction of each enet lasso model

`BSWiMS.ensemble.prediction`
: The ensemble prediction by all models on the test data

`AtOptFormulas.list`
: The list of formulas with "optimal" performance

`ForwardFormulas.list`
: The list of formulas produced by the forward procedure

`baggFormulas.list`
: The list of the bagged models

`LassoFilterVarList`
: The list of variables used by LASSO fitting

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

References

Pencina, M. J., D'Agostino, R. B., & Vasan, R. S. (2008). Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond. *Statistics in medicine* **27**(2), 157-172.

See Also

`crossValidationFeatureSelection_Res`, `ForwardSelection.Model.Bin`, `ForwardSelection.Model.Res`

---

crossValidationFeatureSelection_Res

NeRI-based selection of a linear, logistic, or Cox proportional hazards regression model from a set of candidate variables

---

Description

This function performs a cross-validation analysis of a feature selection algorithm based on net residual improvement (NeRI) to return a predictive model. It is composed of a NeRI-based feature selection followed by an update procedure, ending with a bootstrapping backwards feature elimination. The user can control how many train and blind test sets will be evaluated.

Usage

```
crossValidationFeatureSelection_Res(size = 10,
                                    fraction = 1.0,
                                    pvalue = 0.05,
                                    loops = 100,
                                    covariates = "1",
                                    Outcome,
                                    timeOutcome = "Time",
                                    variableList,
                                    data,
                                    maxTrainModelSize = 20,
```

```
                                type = c("LM", "LOGIT", "COX"),
                                testType = c("Binomial",
                                             "Wilcox",
                                             "tStudent",
                                             "Ftest"),
                                startOffset = 0,
                                elimination.bootstrap.steps = 25,
                                trainFraction = 0.67,
                                trainRepetition = 9,
                                elimination.pValue = 0.05,
                                setIntersect = 1,
                                update.pvalue = c(0.05,0.05),
                                unirank = NULL,
                                print=TRUE,
                                plots=TRUE,
                                lambda="lambda.1se",
                                adjsize=1.0,
                                equivalent=FALSE,
                                bswimsCycles=10,
                                usrFitFun=NULL
                                )
```

Arguments

`size`	The number of candidate variables to be tested (the first `size` variables from `variableList`)
`fraction`	The fraction of data (sampled with replacement) to be used as train
`pvalue`	The maximum *p*-value, associated to the NeRI, allowed for a term in the model
`loops`	The number of bootstrap loops
`covariates`	A string of the type "1 + var1 + var2" that defines which variables will always be included in the models (as covariates)
`Outcome`	The name of the column in `data` that stores the variable to be predicted by the model
`timeOutcome`	The name of the column in `data` that stores the time to event (needed only for a Cox proportional hazards regression model fitting)
`variableList`	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
`data`	A data frame where all variables are stored in different columns
`maxTrainModelSize`	Maximum number of terms that can be included in the model
`type`	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
`testType`	Type of non-parametric test to be evaluated by the `improvedResiduals` function: Binomial test ("Binomial"), Wilcoxon rank-sum test ("Wilcox"), Student's *t*-test ("tStudent"), or *F*-test ("Ftest")
`startOffset`	Only terms whose position in the model is larger than the `startOffset` are candidates to be removed

`elimination.bootstrap.steps`
: The number of bootstrap loops for the backwards elimination procedure

`trainFraction`
: The fraction of data (sampled with replacement) to be used as train for the cross-validation procedure

`setIntersect`
: The intersect of the model (To force a zero intersect, set this value to 0)

`trainRepetition`
: The number of cross-validation folds (it should be at least equal to $1/trainFraction$ for a complete cross-validation)

`elimination.pValue`
: The maximum *p*-value, associated to the NeRI, allowed for a term in the model by the backward elimination procedure

`update.pvalue`
: The maximum *p*-value, associated to the NeRI, allowed for a term in the model by the update procedure

`unirank`
: A list with the results yielded by the `uniRankVar` function, required only if the rank needs to be updated during the cross-validation procedure

`print`
: Logical. If `TRUE`, information will be displayed

`plots`
: Logical. If `TRUE`, plots are displayed

`lambda`
: The passed value to the s parameter of the glmnet cross validation coefficient

`adjsize`
: The expected size of a random model for multiple selection correction

`equivalent`
: Is set to TRUE CV will compute the equivalent model

`bswimsCycles`
: The maximum number of models to be returned by `BSWiMS.model`

`usrFitFun`
: A user fitting function to be evaluated by the cross validation procedure

Details

This function produces a set of data and plots that can be used to inspect the degree of over-fitting or shrinkage of a model. It uses bootstrapped data, cross-validation data, and, if possible, retrain data.

Value

`formula.list`
: A list containing objects of class `formula` with the formulas used to fit the models found at each cycle

`Models.testPrediction`
: A data frame with the blind test set predictions made at each fold of the cross validation (Full B:SWiMS,Median,Bagged,Forward,Backward Elimination), where the models used to generate such predictions (`formula.list`) were generated via a feature selection process which included only the train set. It also includes a column with the `Outcome` of each prediction, and a column with the number of the fold at which the prediction was made.

`FullBSWiMS.testPrediction`
: A data frame similar to `Models.testPrediction`, but where the model used to generate the predictions was the Full model, generated via a feature selection process which included all data.

`BSWiMS` A list containing the values returned by `bootstrapVarElimination_Res` using all data and the model from `updatedforwardModel`

`forwardSelection`
A list containing the values returned by `ForwardSelection.Model.Res` using all data

`updatedforwardModel`
A list containing the values returned by `updateModel.Res` using all data and the model from `forwardSelection`

`testRMSE` The global blind test root-mean-square error (RMSE) of the cross-validation procedure

`testPearson` The global blind test Pearson r product-moment correlation coefficient of the cross-validation procedure

`testSpearman` The global blind test Spearman ρ rank correlation coefficient of the cross-validation procedure

`FulltestRMSE` The global blind test RMSE of the Full model

`FullTestPearson`
The global blind test Pearson r product-moment correlation coefficient of the Full model

`FullTestSpearman`
The global blind test Spearman ρ rank correlation coefficient of the Full model

`trainRMSE` The train RMSE at each fold of the cross-validation procedure

`trainPearson` The train Pearson r product-moment correlation coefficient at each fold of the cross-validation procedure

`trainSpearman` The train Spearman ρ rank correlation coefficient at each fold of the cross-validation procedure

`FullTrainRMSE` The train RMSE of the Full model at each fold of the cross-validation procedure

`FullTrainPearson`
The train Pearson r product-moment correlation coefficient of the Full model at each fold of the cross-validation procedure

`FullTrainSpearman`
The train Spearman ρ rank correlation coefficient of the Full model at each fold of the cross-validation procedure

`testRMSEAtFold` The blind test RMSE at each fold of the cross-validation procedure

`FullTestRMSEAtFold`
The blind test RMSE of the Full model at each fold of the cross-validation procedure

`Fullenet` An object of class `cv.glmnet` containing the results of an elastic net cross-validation fit

`LASSO.testPredictions`
A data frame similar to `Models.testPrediction`, but where the predictions were made by the elastic net model

`LASSOVariables` A list with the elastic net Full model and the models found at each cross-validation fold

`byFoldTestMS` A vector with the Mean Square error for each blind fold

`byFoldTestSpearman`
: A vector with the Spearman correlation between prediction and outcome for each blind fold

`byFoldTestPearson`
: A vector with the Pearson correlation between prediction and outcome for each blind fold

`byFoldCstat`
: A vector with the C-index (Somers' Dxy rank correlation :`rcorr.cens`) between prediction and outcome for each blind fold

`CVBlindPearson`
: A vector with the Pearson correlation between the outcome and prediction for each repeated experiment

`CVBlindSpearman`
: A vector with the Spearm correlation between the outcome and prediction for each repeated experiment

`CVBlindRMS`
: A vector with the RMS between the outcome and prediction for each repeated experiment

`Models.trainPrediction`
: A data frame with the outcome and the train prediction of every model

`FullBSWiMS.trainPrediction`
: A data frame with the outcome and the train prediction at each CV fold for the main model

`LASSO.trainPredictions`
: A data frame with the outcome and the prediction of each enet lasso model

`uniTrainMSS`
: A data frame with mean square of the train residuals from the univariate models of the model terms

`uniTestMSS`
: A data frame with mean square of the test residuals of the univariate models of the model terms

`BSWiMS.ensemble.prediction`
: The ensemble prediction by all models on the test data

`AtOptFormulas.list`
: The list of formulas with "optimal" performance

`ForwardFormulas.list`
: The list of formulas produced by the forward procedure

`baggFormulas.list`
: The list of the bagged models

`LassoFilterVarList`
: The list of variables used by LASSO fitting

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

`crossValidationFeatureSelection_Bin`, `improvedResiduals`, `bootstrapVarElimination_Res`

`EmpiricalSurvDiff` *Estimate the LR value and its associated p-values*

Description

Permutations or Bootstrapping computation of the standardized log-rank (SLR) or the Chi=SLR^2 p-values for differences in survival times

Usage

```
EmpiricalSurvDiff(times=times,
                  status=status,
                  groups=groups,
                  samples=1000,
                  type=c("SLR","Chi"),
                  plots=FALSE,
                  minAproxSamples=100,
                  computeDist=FALSE,
                  ...
                  )
```

Arguments

`times`	A numeric vector with he observed times to event
`status`	A numeric vector indicating if the time to event is censored
`groups`	A numeric vector indicating the label of the two survival groups
`samples`	The number of bootstrap samples
`type`	The type of log-rank statistics. SLR or Chi
`plots`	If TRUE, the Kaplan-Meier plot will be plotted
`minAproxSamples`	The number of tail samples used for the normal-distribution approximation
`computeDist`	If TRUE, it will compute the bootstrapped distribution of the SLR
`...`	Additional parameters for the plot

Details

It will compute the null distribution of the SRL or the square SLR (Chi) via permutations, and it will return the p-value of differences between survival times between two groups. It may also be used to compute the empirical distribution of the difference in SLR using bootstrapping. (computeDist=TRUE) The p-values will be estimated based on the sampled distribution, or normal-approximated along the tails.

Value

pvalue	the minimum one-tailed p-value : min[p(SRL < 0),p(SRL > 0)] for type="SLR" or the two tailed p-value: 1-p(|SRL| > 0) for type="Chi"
LR	A list of LR statistics: LR=Expected, VR=Variance, SLR=Standardized LR.
p.equal	The two tailed p-value: 1-p(|SRL| > 0)
p.sup	The one tailed p-value: p(SRL < 0), return NA for type="Chi"
p.inf	The one tailed p-value: p(SRL > 0), return NA for type="Chi"
nullDist	permutation derived probability density function of the null distribution
LRDist	bootrapped derived probability density function of the SLR (computeDist=TRUE)

Author(s)

Jose G. Tamez-Pena

Examples

```
## Not run:

      library(rpart)
      data(stagec)

      # The Log-Rank Analysis using survdiff

      lrsurvdiff <- survdiff(Surv(pgtime,pgstat)~grade>2,data=stagec)
      print(lrsurvdiff)

      # The Log-Rank Analysis: permutations of the null Chi distribution
      lrp <- EmpiricalSurvDiff(stagec$pgtime,stagec$pgstat,stagec$grade>2,
                        type="Chi",plots=TRUE,samples=10000,
                        main="Chi Null Distribution")
      print(list(unlist(c(lrp$LR,lrp$pvalue))))

      # The Log-Rank Analysis: permutations of the null SLR distribution
      lrp <- EmpiricalSurvDiff(stagec$pgtime,stagec$pgstat,stagec$grade>2,
                        type="SLR",plots=TRUE,samples=10000,
                        main="SLR Null Distribution")
      print(list(unlist(c(lrp$LR,lrp$pvalue))))

      # The Log-Rank Analysis: Bootstraping the SLR distribution
      lrp <- EmpiricalSurvDiff(stagec$pgtime,stagec$pgstat,stagec$grade>2,
                        computeDist=TRUE,plots=TRUE,samples=100000,
                        main="SLR Null and SLR bootrapped")
      print(list(unlist(c(lrp$LR,lrp$pvalue))))


## End(Not run)
```

ensemblePredict *The median prediction from a list of models*

Description

Given a list of model formulas, this function will train such models and return the a single(ensemble) prediction from the list of formulas on a test data set. It may also provides a *k*-nearest neighbours (KNN) prediction using the features listed in such models.

Usage

```
ensemblePredict(formulaList,
                trainData,
                testData = NULL,
                predictType = c("prob", "linear"),
                type = c("LOGIT", "LM", "COX","SVM"),
                Outcome = NULL,
                nk = 0
  )
```

Arguments

formulaList	A list made of objects of class formula, each representing a model formula to be fitted and predicted with
trainData	A data frame with the data to train the model, where all variables are stored in different columns
testData	A data frame similar to trainData, but with the data set to be predicted. If NULL, trainData will be used
predictType	Prediction type: Probability ("prob") or linear predictor ("linear")
type	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
Outcome	The name of the column in data that stores the variable to be predicted by the model
nk	The number of neighbours used to generate the KNN classification. If zero, *k* is set to the square root of the number of cases. If less than zero, it will not perform the KNN classification

Value

ensemblePredict	A vector with the median prediction for the testData data set, using the models from formulaList
medianKNNPredict	A vector with the median prediction for the testData data set, using the KNN models
predictions	A matrix, where each column represents the predictions made with each model from formulaList

KNNpredictions	A matrix, where each column represents the predictions made with a different KNN model
wPredict	A vector with the weighted mean ensemble

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

`featureAdjustment` *Adjust each listed variable to the provided set of covariates*

Description

This function fits the candidate variables to the provided model,for each strata, on a control population. If the variance of the residual (the fitted observation minus the real observation) is reduced significantly, then, such residual is used in the resulting data frame. Otherwise, the control mean is subtracted to the observation.

Usage

```
featureAdjustment(variableList,
                  baseModel,
                  strata = NA,
                  data,
                  referenceframe,
                  type = c("LM", "GLS"),
                  pvalue = 0.05,
                  correlationGroup = "ID")
```

Arguments

variableList	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
baseModel	A string of the type "1 + var1 + var2" that defines the model to which variables will be fitted
strata	The name of the column in data that stores the variable that will be used to stratify the model
data	A data frame where all variables are stored in different columns
referenceframe	A data frame similar to data, but with only the control population
type	Fit type: linear fitting ("LM"), or generalized least squares fitting ("GLS")
pvalue	The maximum p-value, associated to the F-test, for the model to be allowed to reduce variability
correlationGroup	The name of the column in data that stores the variable to be used to group the data (only needed if type defined as "GLS")

Value

A data frame, where each input observation has been adjusted from `data` at each `strata`

Note

This function prints the residuals and the F-statistic for all candidate variables

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

`ForwardSelection.Model.Bin`

IDI/NRI-based feature selection procedure for linear, logistic, and Cox proportional hazards regression models

---

Description

This function performs a bootstrap sampling to rank the variables that statistically improve prediction. After the frequency rank, the function uses a forward selection procedure to create a final model, whose terms all have a significant contribution to the integrated discrimination improvement (IDI) or the net reclassification improvement (NRI). For each bootstrap, the IDI/NRI is computed and the variable with the largest statically significant IDI/NRI is added to the model. The procedure is repeated at each bootstrap until no more variables can be inserted. The variables that enter the model are then counted, and the same procedure is repeated for the rest of the bootstrap loops. The frequency of variable-inclusion in the model is returned as well as a model that uses the frequency of inclusion.

Usage

```
ForwardSelection.Model.Bin(size = 100,
                           fraction = 1,
                           pvalue = 0.05,
                           loops = 100,
                           covariates = "1",
                           Outcome,
                           variableList,
                           data,
                           maxTrainModelSize = 20,
                           type = c("LM", "LOGIT", "COX"),
                           timeOutcome = "Time",
                           selectionType=c("zIDI", "zNRI"),
                           cores = 4,
                           randsize = 0)
```

Arguments

size	The number of candidate variables to be tested (the first size variables from variableList)
fraction	The fraction of data (sampled with replacement) to be used as train
pvalue	The maximum *p*-value, associated to either IDI or NRI, allowed for a term in the model
loops	The number of bootstrap loops
covariates	A string of the type "1 + var1 + var2" that defines which variables will always be included in the models (as covariates)
Outcome	The name of the column in data that stores the variable to be predicted by the model
variableList	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
data	A data frame where all variables are stored in different columns
maxTrainModelSize	Maximum number of terms that can be included in the model
type	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
timeOutcome	The name of the column in data that stores the time to event (needed only for a Cox proportional hazards regression model fitting)
selectionType	The type of index to be evaluated by the improveProb function (Hmisc package): *z*-score of IDI or of NRI
cores	Cores to be used for parallel processing
randsize	the model size of a random outcome. If randsize is less than zero. It will estimate the size

Value

final.model	An object of class lm, glm, or coxph containing the final model
var.names	A vector with the names of the features that were included in the final model
formula	An object of class formula with the formula used to fit the final model
ranked.var	An array with the ranked frequencies of the features
z.selection	A vector in which each term represents the *z*-score of the index defined in selectionType obtained with the Full model and the model without one term
formula.list	A list containing objects of class formula with the formulas used to fit the models found at each cycle
variableList	A list of variables used in the forward selection

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

References

Pencina, M. J., D'Agostino, R. B., & Vasan, R. S. (2008). Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond. *Statistics in medicine* **27**(2), 157-172.

See Also

ForwardSelection.Model.Res

`ForwardSelection.Model.Res` *NeRI-based feature selection procedure for linear, logistic, or Cox proportional hazards regression models*

Description

This function performs a bootstrap sampling to rank the most frequent variables that statistically aid the models by minimizing the residuals. After the frequency rank, the function uses a forward selection procedure to create a final model, whose terms all have a significant contribution to the net residual improvement (NeRI).

Usage

```
ForwardSelection.Model.Res(size = 100,
                     fraction = 1,
                     pvalue = 0.05,
                     loops = 100,
                     covariates = "1",
                     Outcome,
                     variableList,
                     data,
                     maxTrainModelSize = 20,
                     type = c("LM", "LOGIT", "COX"),
                     testType=c("Binomial", "Wilcox", "tStudent", "Ftest"),
                     timeOutcome = "Time",
                     cores = 4,
                     randsize = 0)
```

Arguments

`size`	The number of candidate variables to be tested (the first `size` variables from `variableList`)
`fraction`	The fraction of data (sampled with replacement) to be used as train
`pvalue`	The maximum *p*-value, associated to the NeRI, allowed for a term in the model (controls the false selection rate)
`loops`	The number of bootstrap loops

covariates	A string of the type "1 + var1 + var2" that defines which variables will always be included in the models (as covariates)
Outcome	The name of the column in data that stores the variable to be predicted by the model
variableList	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
data	A data frame where all variables are stored in different columns
maxTrainModelSize	Maximum number of terms that can be included in the model
type	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
testType	Type of non-parametric test to be evaluated by the improvedResiduals function: Binomial test ("Binomial"), Wilcoxon rank-sum test ("Wilcox"), Student's *t*-test ("tStudent"), or *F*-test ("Ftest")
timeOutcome	The name of the column in data that stores the time to event (needed only for a Cox proportional hazards regression model fitting)
cores	Cores to be used for parallel processing
randsize	the model size of a random outcome. If randsize is less than zero. It will estimate the size

Value

final.model	An object of class lm, glm, or coxph containing the final model
var.names	A vector with the names of the features that were included in the final model
formula	An object of class formula with the formula used to fit the final model
ranked.var	An array with the ranked frequencies of the features
z.NeRIs	A vector in which each element represents the *z*-score of the NeRI, associated to the testType, for each feature found in the final model
formula.list	A list containing objects of class formula with the formulas used to fit the models found at each cycle
variableList	A list of variables used in the forward selection

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

ForwardSelection.Model.Bin

FRESA.Model *Automated model selection*

Description

This function uses a wrapper procedure to select the best features of a non-penalized linear model that best predict the outcome, given the formula of an initial model template (linear, logistic, or Cox proportional hazards), an optimization procedure, and a data frame. A filter scheme may be enabled to reduce the search space of the wrapper procedure. The false selection rate may be empirically controlled by enabling bootstrapping, and model shrinkage can be evaluated by cross-validation.

Usage

```
FRESA.Model(formula,
            data,
            OptType = c("Binary", "Residual"),
            pvalue = 0.05,
            filter.p.value = 0.10,
            loops = 32,
            maxTrainModelSize = 20,
            elimination.bootstrap.steps = 100,
            bootstrap.steps = 100,
            print = FALSE,
            plots = FALSE,
            CVfolds = 1,
            repeats = 1,
            nk = 0,
            categorizationType = c("Raw",
                                   "Categorical",
                                   "ZCategorical",
                                   "RawZCategorical",
                                   "RawTail",
                                   "RawZTail",
                                   "Tail",
                                   "RawRaw"),
            cateGroups = c(0.1, 0.9),
            raw.dataFrame = NULL,
            var.description = NULL,
            testType = c("zIDI",
                         "zNRI",
                         "Binomial",
                         "Wilcox",
                         "tStudent",
                         "Ftest"),
            lambda="lambda.1se",
            equivalent=FALSE,
            bswimsCycles=10,
```

```
usrFitFun=NULL
)
```

Arguments

formula	An object of class formula with the formula to be fitted
data	A data frame where all variables are stored in different columns
OptType	Optimization type: Based on the integrated discrimination improvement (Binary) index for binary classification ("Binary"), or based on the net residual improvement (NeRI) index for linear regression ("Residual")
pvalue	The maximum *p*-value, associated to the testType, allowed for a term in the model (it will control the false selection rate)
filter.p.value	The maximum *p*-value, for a variable to be included to the feature selection procedure
loops	The number of bootstrap loops for the forward selection procedure
maxTrainModelSize	Maximum number of terms that can be included in the model
elimination.bootstrap.steps	The number of bootstrap loops for the backwards elimination procedure
bootstrap.steps	The number of bootstrap loops for the bootstrap validation procedure
print	Logical. If TRUE, information will be displayed
plots	Logical. If TRUE, plots are displayed
CVfolds	The number of folds for the final cross-validation
repeats	The number of times that the cross-validation procedure will be repeated
nk	The number of neighbours used to generate a *k*-nearest neighbours (KNN) classification. If zero, *k* is set to the square root of the number of cases. If less than zero, it will not perform the KNN classification
categorizationType	How variables will be analyzed: As given in data ("Raw"); broken into the *p*-value categories given by cateGroups ("Categorical"); broken into the *p*-value categories given by cateGroups, and weighted by the *z*-score ("ZCategorical"); broken into the *p*-value categories given by cateGroups, weighted by the *z*-score, plus the raw values ("RawZCategorical"); raw values, plus the tails ("RawTail"); or raw values, weighted by the *z*-score, plus the tails ("RawZTail")
cateGroups	A vector of percentiles to be used for the categorization procedure
raw.dataFrame	A data frame similar to data, but with unadjusted data, used to get the means and variances of the unadjusted data
var.description	A vector of the same length as the number of columns of *data*, containing a description of the variables

testType	For an Binary-based optimization, the type of index to be evaluated by the improveProb function (Hmisc package): z-value of Binary or of NRI. For a NeRI-based optimization, the type of non-parametric test to be evaluated by the improvedResiduals function: Binomial test ("Binomial"), Wilcoxon rank-sum test ("Wilcox"), Student's t-test ("tStudent"), or F-test ("Ftest")
lambda	The passed value to the s parameter of the glmnet cross validation coefficient
equivalent	Is set to TRUE CV will compute the equivalent model
bswimsCycles	The maximum number of models to be returned by BSWiMS.model
usrFitFun	An optional user provided fitting function to be evaluated by the cross validation procedure: fitting: usrFitFun(formula,data), with a predict function

Details

This is the main function of FRESA.CAD given an outcome formula, and a data.frame this function will do an univariate analysis of the data (univariateRankVariables), then it will select the top ranked variables; after that it will select the model that best describes the outcome. At output it will return the bootstrapped performance of the model (bootstrapValidation_Bin or bootstrapValidation_Res). It can be set to report the cross-validation performance of the selection process which will return either a crossValidationFeatureSelection_Bin or a crossValidationFeatureSelectior object.

Value

BSWiMS.model	An object of class lm, glm, or coxph containing the final model
reducedModel	The resulting object of the backward elimination procedure
univariateAnalysis	A data frame with the results from the univariate analysis
forwardModel	The resulting object of the feature selection function.
updatedforwardModel	The resulting object of the the update procedure
bootstrappedModel	The resulting object of the bootstrap procedure on final.model
cvObject	The resulting object of the cross-validation procedure
used.variables	The number of terms that passed the filter procedure
call	the function call

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

References

Pencina, M. J., D'Agostino, R. B., & Vasan, R. S. (2008). Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond. *Statistics in medicine* **27**(2), 157-172.

Examples

```
## Not run:
# Start the graphics device driver to save all plots in a pdf format
pdf(file = "Example.pdf")
# Get the stage C prostate cancer data from the rpart package
library(rpart)
data(stagec)
# Split the stages into several columns
dataCancer <- cbind(stagec[,c(1:3,5:6)],
                    gleason4 = 1*(stagec[,7] == 4),
                    gleason5 = 1*(stagec[,7] == 5),
                    gleason6 = 1*(stagec[,7] == 6),
                    gleason7 = 1*(stagec[,7] == 7),
                    gleason8 = 1*(stagec[,7] == 8),
                    gleason910 = 1*(stagec[,7] >= 9),
                    eet = 1*(stagec[,4] == 2),
                    diploid = 1*(stagec[,8] == "diploid"),
                    tetraploid = 1*(stagec[,8] == "tetraploid"),
                    notAneuploid = 1-1*(stagec[,8] == "aneuploid"))
# Remove the incomplete cases
dataCancer <- dataCancer[complete.cases(dataCancer),]
# Load a pre-stablished data frame with the names and descriptions of all variables
data(cancerVarNames)

# Get a Cox proportional hazards model using:
# - The default parameters
md <- FRESA.Model(formula = Surv(pgtime, pgstat) ~ 1,
                  data = dataCancer,
  var.description = cancerVarNames[,2])
# Get a logistic regression model using
# - The default parameters
md <- FRESA.Model(formula = pgstat ~ 1,
                  data = dataCancer,
  var.description = cancerVarNames[,2])
# Get a logistic regression model using:
# - redidual-based optimization
md <- FRESA.Model(formula = pgstat ~ 1,
                  data = dataCancer,
                  OptType = "Residual",
  var.description = cancerVarNames[,2])
# Shut down the graphics device driver
dev.off()
## End(Not run)
```

`getKNNpredictionFromFormula`

Predict classification using KNN

Description

This function will return the classification of the samples of a test set using a *k*-nearest neighbors (KNN) algorithm with euclidean distances, given a formula and a train set.

Usage

```
getKNNpredictionFromFormula(model.formula,
                            trainData,
                            testData,
                            Outcome = "CLASS",
                            nk = 3)
```

Arguments

`model.formula`	An object of class `formula` with the formula to be used
`trainData`	A data frame with the data to train the model, where all variables are stored in different columns
`testData`	A data frame similar to `trainData`, but with the data set to be predicted
`Outcome`	The name of the column in `trainData` that stores the variable to be predicted by the model
`nk`	The number of neighbors used to generate the KNN classification

Value

`prediction`	A vector with the predicted outcome for the `testData` data set
`prob`	The proportion of *k* neighbours that predicted the class to be the one being reported in `prediction`
`binProb`	The proportion of *k* neighbours that predicted the class of the outcome to be equal to 1
`featureList`	A vector with the names of the features used by the KNN procedure

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

`predict.fitFRESA`, `knn`

`getSignature` *Returns a CV signature template*

Description

This function returns the matrix template [mean,sd,IQR] that maximizes the ROC AUC between cases of controls.

Usage

```
getSignature(
              data,
              varlist=NULL,
              Outcome=NULL,
              target=c("All","Control","Case"),
              CVFolds=3,
              repeats=9,
              distanceFunction=signatureDistance,
              ...
)
```

Arguments

`data`	A data frame whose rows contains the sampled "subject" data, and each column is a feature.
`varlist`	The varlist is a character vector that list all the features to be searched by the Backward elimination forward selection procedure.
`Outcome`	The name of the column that has the binary outcome. 1 for cases, 0 for controls
`target`	The target template that will be used to maximize the AUC.
`CVFolds`	The number of folds to be used
`repeats`	how many times the CV procedure will be repeated
`distanceFunction`	The function to be used to compute the distance between the template and each sample
`...`	the parameters to be passed to the distance function

Details

The function repeats full cycles of a Cross Validation (RCV) procedure. At each CV cycle the algorithm estimate the mean template and the distance between the template and the test samples. The ROC AUC is computed after the RCV is completed. A forward selection scheme. The set of features that maximize the AUC during the Forward loop is returned.

Value

`controlTemplate`
: the control matrix with quantile probs[0.025,0.25,0.5,0.75,0.975] that maximized the AUC (template of controls subjects)

`caseTamplate`
: the case matrix with quantile probs[0.025,0.25,0.5,0.75,0.975] that maximized the AUC (template of case subjects)

`AUCevolution`
: The AUC value at each cycle

`featureSizeEvolution`
: The number of features at each cycle

`featureList`
: The final list of features

`CVOutput`
: A data frame with four columns: ID, Outcome, Case Distances, Control Distances. Each row contains the CV test results

`MaxAUC`
: The maximum ROC AUC

Author(s)

Jose G. Tamez-Pena

---

getVar.Bin *Analysis of the effect of each term of a binary classification model by analysing its reclassification performance*

Description

This function provides an analysis of the effect of each model term by comparing the binary classification performance between the Full model and the model without each term. The model is fitted using the train data set, but probabilities are predicted for the train and test data sets. Reclassification improvement is evaluated using the `improveProb` function (`Hmisc` package). Additionally, the integrated discrimination improvement (IDI) and the net reclassification improvement (NRI) of each model term are reported.

Usage

```
getVar.Bin(object,
                      data,
                      Outcome = "Class",
                      type = c("LOGIT", "LM", "COX"),
                      testData = NULL,
                      callCpp=TRUE)
```

Arguments

object	An object of class lm, glm, or coxph containing the model to be analysed
data	A data frame where all variables are stored in different columns
Outcome	The name of the column in data that stores the variable to be predicted by the model
type	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
testData	A data frame similar to data, but with a data set to be independently tested. If NULL, data will be used.
callCpp	is set to true it will use the c++ implementation of improvement.

Value

z.IDIs	A vector in which each term represents the z-score of the IDI obtained with the Full model and the model without one term
z.NRIs	A vector in which each term represents the z-score of the NRI obtained with the Full model and the model without one term
IDIs	A vector in which each term represents the IDI obtained with the Full model and the model without one term
NRIs	A vector in which each term represents the NRI obtained with the Full model and the model without one term
testData.z.IDIs	A vector similar to z.IDIs, where values were estimated in testdata
testData.z.NRIs	A vector similar to z.NRIs, where values were estimated in testdata
testData.IDIs	A vector similar to IDIs, where values were estimated in testdata
testData.NRIs	A vector similar to NRIs, where values were estimated in testdata
uniTrainAccuracy	A vector with the univariate train accuracy of each model variable
uniTestAccuracy	A vector with the univariate test accuracy of each model variable

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

References

Pencina, M. J., D'Agostino, R. B., & Vasan, R. S. (2008). Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond. *Statistics in medicine* **27**(2), 157-172.

See Also

getVar.Res

getVar.Res *Analysis of the effect of each term of a linear regression model by analysing its residuals*

Description

This function provides an analysis of the effect of each model term by comparing the residuals of the Full model and the model without each term. The model is fitted using the train data set, but analysis of residual improvement is done on the train and test data sets. Residuals are compared by a paired *t*-test, a paired Wilcoxon rank-sum test, a binomial sign test and the *F*-test on residual variance. Additionally, the net residual improvement (NeRI) of each model term is reported.

Usage

```
getVar.Res(object,
           data,
           Outcome = "Class",
           type = c("LM", "LOGIT", "COX"),
           testData = NULL,
           callCpp=TRUE)
```

Arguments

object	An object of class lm, glm, or coxph containing the model to be analyzed
data	A data frame where all variables are stored in different columns
Outcome	The name of the column in data that stores the variable to be predicted by the model
type	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
testData	A data frame similar to data, but with a data set to be independently tested. If NULL, data will be used.
callCpp	is set to true it will use the c++ implementation of residual improvement.

Value

tP.value	A vector in which each element represents the single sided *p*-value of the paired *t*-test comparing the absolute values of the residuals obtained with the Full model and the model without one term
BinP.value	A vector in which each element represents the *p*-value associated with a significant improvement in residuals according to the binomial sign test
WilcoxP.value	A vector in which each element represents the single sided *p*-value of the Wilcoxon rank-sum test comparing the absolute values of the residuals obtained with the Full model and the model without one term
FP.value	A vector in which each element represents the single sided *p*-value of the *F*-test comparing the residual variances of the residuals obtained with the Full model and the model without one term

NeRIs A vector in which each element represents the net residual improvement between the Full model and the model without one term

testData.tP.value
A vector similar to tP.value, where values were estimated in testdata

testData.BinP.value
A vector similar to BinP.value, where values were estimated in testdata

testData.WilcoxP.value
A vector similar to WilcoxP.value, where values were estimated in testdata

testData.FP.value
A vector similar to FP.value, where values were estimated in testdata

testData.NeRIs A vector similar to NeRIs, where values were estimated in testdata

unitestMSE A vector with the univariate residual mean sum of squares of each model variable on the test data

unitrainMSE A vector with the univariate residual mean sum of squares of each model variable on the train data

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

getVar.Bin

---

heatMaps *Plot a heat map of selected variables*

Description

This function creates a heat map for a data set based on a univariate or frequency ranking

Usage

```
heatMaps(variableList=NULL,
         varRank = NULL,
         Outcome,
         data,
         title = "Heat Map",
         hCluster = FALSE,
         prediction = NULL,
         Scale = FALSE,
         theFiveColors=c("blue","cyan","black","yellow","red"),
         outcomeColors = c("blue","lightgreen","yellow","orangered","red"),
         transpose=FALSE,
         ...)
```

Arguments

variableList	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
varRank	A data frame with the name of the variables in variableList, ranked according to a certain metric
Outcome	The name of the column in data that stores the variable to be predicted by the model
data	A data frame where all variables are stored in different columns
title	The title of the plot
hCluster	Logical. If TRUE, variables will be clustered
prediction	A vector with a prediction for each subject, which will be used to rank the heat map
Scale	An optional value to force the data normalization outcome
theFiveColors	the colors of the heatmap
outcomeColors	the colors of the outcome bar
transpose	transpose the heatmap
...	additional parameters for the heatmap.2 function

Value

dataMatrix	A matrix with all the terms in data described by variableList
orderMatrix	A matrix similar to dataMatrix, where rows are ordered according to the outcome
heatMap	A list with the values returned by the heatmap.2 function (gplots package)

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

Examples

```
## Not run:

library(rpart)
data(stagec)

# Set the options to keep the na
options(na.action='na.pass')
# create a model matrix with all the NA values imputed
stagecImputed <- as.data.frame(nearestneighborimpute(model.matrix(~.,stagec)[,-1]))

# the simple heat map
hm <- heatMaps(Outcome="pgstat",data=stagecImputed,title="Heat Map",Scale=TRUE)

# transposing the heat-map with clustered colums
hm <- heatMaps(Outcome="pgstat",data=stagecImputed,title="Heat Map",Scale=TRUE,
   transpose= TRUE,hCluster = TRUE,
```

```
   cexRow=0.80,cexCol=0.50,srtCol=35)

# transposing the heat-map with reds and time to event as outcome
hm <- heatMaps(Outcome="pgtime",data=stagecImputed,title="Heat Map",Scale=TRUE,
   theFiveColors=c("black","red","orange","yellow","white"),
   cexRow=0.50,cexCol=0.80,srtCol=35)

## End(Not run)
```

`improvedResiduals` *Estimate the significance of the reduction of predicted residuals*

Description

This function will test the hypothesis that, given a set of two residuals (new vs. old), the new ones are better than the old ones as measured with non-parametric tests. Four *p*-values are provided: one for the binomial sign test, one for the paired Wilcoxon rank-sum test, one for the paired *t*-test, and one for the `F`-test. The proportion of subjects that improved their residuals, the proportion that worsen their residuals, and the net residual improvement (NeRI) will be returned.

Usage

```
improvedResiduals(oldResiduals,
                  newResiduals,
                  testType = c("Binomial", "Wilcox", "tStudent", "Ftest"))
```

Arguments

`oldResiduals`	A vector with the residuals of the original model
`newResiduals`	A vector with the residuals of the new model
`testType`	Type of non-parametric test to be evaluated: Binomial test ("Binomial"), Wilcoxon rank-sum test ("Wilcox"), Student's *t*-test ("tStudent"), or *F*-test ("Ftest")

Details

This function will test the hypothesis that the new residuals are "better" than the old residuals. To test this hypothesis, four types of tests are performed:

1. The paired *t*-test, which compares the absolute value of the residuals
2. The paired Wilcoxon rank-sum test, which compares the absolute value of residuals
3. The binomial sign test, which evaluates whether the number of subjects with improved residuals is greater than the number of subjects with worsened residuals
4. The *F*-test, which is the standard test for evaluating whether the residual variance is "better" in the new residuals.

The proportions of subjects that improved and worsen their residuals are returned, and so is the NeRI.

Value

p1	Proportion of subjects that improved their residuals to the total number of subjects
p2	Proportion of subjects that worsen their residuals to the total number of subjects
NeRI	The net residual improvement (p1-p2)
p.value	The one tail *p*-value of the test specified in *testType*
BinP.value	The *p*-value associated with a significant improvement in residuals
WilcoxP.value	The single sided *p*-value of the Wilcoxon rank-sum test comparing the absolute values of the new and old residuals
tP.value	The single sided *p*-value of the paired t-test comparing the absolute values of the new and old residuals
FP.value	The single sided *p*-value of the F-test comparing the residual variances of the new and old residuals

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

---

listTopCorrelatedVariables

List the variables that are highly correlated with each other

---

Description

This function computes the Pearson, Spearman, or Kendall correlation for each specified variable in the data set and returns a list of the variables that are correlated to them. It also provides a short variable list without the highly correlated variables.

Usage

```
listTopCorrelatedVariables(variableList,
                           data,
                           pvalue = 0.001,
                           corthreshold = 0.9,
                           method = c("pearson", "kendall", "spearman"))
```

Arguments

variableList	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
data	A data frame where all variables are stored in different columns
pvalue	The maximum *p*-value, associated to method, allowed for a pair of variables to be defined as significantly correlated

`corthreshold` The minimum correlation score, associated to `method`, allowed for a pair of variables to be defined as significantly correlated

`method` Correlation method: Pearson product-moment ("pearson"), Spearman's rank ("spearman"), or Kendall rank ("kendall")

Value

`correlated.variables`

A data frame with two columns:

1. cor.var.names: The variables that are correlated
2. cor.var.value: The correlation value

`short.list` A vector with a list of variables that are not correlated to each other. For every correlated pair, only the variable that first entered the correlation analysis was kept

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

Examples

```
## Not run:
# Start the graphics device driver to save all plots in a pdf format
pdf(file = "Example.pdf")
# Get the stage C prostate cancer data from the rpart package
library(rpart)
data(stagec)
# Split the stages into several columns
dataCancer <- cbind(stagec[,c(1:3,5:6)],
                    gleason4 = 1*(stagec[,7] == 4),
                    gleason5 = 1*(stagec[,7] == 5),
                    gleason6 = 1*(stagec[,7] == 6),
                    gleason7 = 1*(stagec[,7] == 7),
                    gleason8 = 1*(stagec[,7] == 8),
                    gleason910 = 1*(stagec[,7] >= 9),
                    eet = 1*(stagec[,4] == 2),
                    diploid = 1*(stagec[,8] == "diploid"),
                    tetraploid = 1*(stagec[,8] == "tetraploid"),
                    notAneuploid = 1-1*(stagec[,8] == "aneuploid"))
# Remove the incomplete cases
dataCancer <- dataCancer[complete.cases(dataCancer),]
# Load a pre-stablished data frame with the names and descriptions of all variables
data(cancerVarNames)
# Get the variables that have a correlation coefficient larger
# than 0.65 at a p-value of 0.05
cor <- listTopCorrelatedVariables(variableList = cancerVarNames,
                                  data = dataCancer,
                                  pvalue = 0.05,
                                  corthreshold = 0.65,
                                  method = "pearson")
# Shut down the graphics device driver
```

```
dev.off()
## End(Not run)
```

`modelFitting` *Fit a model to the data*

Description

This function fits a linear, logistic, or Cox proportional hazards regression model to given data

Usage

```
modelFitting(model.formula,
             data,
             type = c("LOGIT", "LM", "COX","SVM"),
             fitFRESA=TRUE,
              ...)
```

Arguments

`model.formula`	An object of class `formula` with the formula to be used
`data`	A data frame where all variables are stored in different columns
`type`	Fit type: Logistic ("LOGIT"), linear ("LM"), Cox proportional hazards ("COX") or "SVM"
`fitFRESA`	if true it will perform use the FRESA cpp code for fitting
`...`	Additional parameters for fitting a default `glm` object

Value

A fitted model of the type defined in `type`

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

`nearestneighborimpute` *nearest neighbor NA imputation*

Description

The function will replace any NA present in the data-frame with the median values of the nearest neighbours.

Usage

```
nearestneighborimpute(tobeimputed,
                      referenceSet=NULL,
                      distol=1.05
                     )
```

Arguments

`tobeimputed`	a data frame with missing values (NA values)
`referenceSet`	An optional data frame with a set of complete observations. This data frame will be added to the search set
`distol`	The tolerance used to define if a particular set of row observations is similar to the minimum distance

Details

This function will find any NA present in the data set and it will search for the row set of complete observations that have the closest IQR normalized Manhattan distance to the row with missing values. If a set of rows have similar minimum distances (toldis*(minimum distance) > row set distance) the median value will be used.

Value

A data frame, where each NA has been replaced with the value of the nearest neighbours

Author(s)

Jose G. Tamez-Pena

Examples

```
## Not run:
# Get the stage C prostate cancer data from the rpart package
library(rpart)
data(stagec)
# Set the options to keep the na
options(na.action='na.pass')
# create a model matrix with all the NA values imputed
```

```
stagecImputed <- nearestneighborimpute(model.matrix(~.,stagec)[,-1])

## End(Not run)
```

---

plot.bootstrapValidation_Bin

Plot ROC curves of bootstrap results

Description

This function plots ROC curves and a Kaplan-Meier curve (when fitting a Cox proportional hazards regression model) of a bootstrapped model.

Usage

```
## S3 method for class 'bootstrapValidation_Bin'
plot(x,
     xlab = "Years",
     ylab = "Survival",
 strata.levels=c(0),
     main = "ROC",
 cex=1.0,
 ...)
```

Arguments

x	A bootstrapValidation_Bin object
xlab	The label of the *x*-axis
ylab	The label of the *y*-axis
strata.levels	stratification level for the Kaplan-Meier plots
main	Main Plot title
cex	The text cex
...	Additional parameters for the generic plot function

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

plot.bootstrapValidation_Res

plot.bootstrapValidation_Res *Plot ROC curves of bootstrap results*

Description

This function plots ROC curves and a Kaplan-Meier curve (when fitting a Cox proportional hazards regression model) of a bootstrapped model.

Usage

```
## S3 method for class 'bootstrapValidation_Res'
plot(x,
     xlab = "Years",
     ylab = "Survival",
     ...)
```

Arguments

x	A bootstrapValidation_Res object
xlab	The label of the *x*-axis
ylab	The label of the *y*-axis
...	Additional parameters for the plot

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

plot.bootstrapValidation_Bin

plotModels.ROC *Plot test ROC curves of each cross-validation model*

Description

This function plots test ROC curves of each model found in the cross validation process. It will also aggregate the models into a single prediction performance, plotting the resulting ROC curve (models coherence). Furthermore, it will plot the mean sensitivity for a given set of specificities.

Usage

```
plotModels.ROC(modelPredictions,
 number.of.models=0,
specificities=c(0.975,0.95,0.90,0.80,0.70,0.60,0.50,0.40,0.30,0.20,0.10,0.05),
 theCVfolds=1,
 predictor="Prediction",
cex=1.0,
 ...)
```

Arguments

modelPredictions	A data frame returned by the crossValidationFeatureSelection_Bin function, either the Models.testPrediction, the FullBSWiMS.testPrediction, the Models.CVtestPredictions, the TestRetrained.blindPredictions, the KNN.testPrediction, or the LASSO.testPredictions value
number.of.models	The maximum number of models to plot
specificities	Vector containing the specificities at which the ROC sensitivities will be calculated
theCVfolds	The number of folds performed in a Cross-validation experiment
predictor	The name of the column to be plotted
cex	Controlling the font size of the text inside the plots
...	Additional parameters for the roc function (pROC package)

Value

ROC.AUCs	A vector with the AUC of each ROC
mean.sensitivities	A vector with the mean sensitivity at the specificities given by specificities
model.sensitivities	A matrix where each row represents the sensitivity at the specificity given by specificities for a different ROC
specificities	The specificities used to calculate the sensitivities
senAUC	The AUC of the ROC curve that resulted from using mean.sensitivities
predictionTable	The confusion matrix between the outcome and the ensemble prediction
ensemblePrediction	The ensemble (median prediction) of the repeated predictions

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

predict.fitFRESA *Linear or probabilistic prediction*

Description

This function returns the predicted outcome of a specific model. The model is used to generate linear predictions. The probabilistic values are generated using the logistic transformation on the linear predictors.

Usage

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

Arguments

object	An object of class fitFRESA containing the model to be analyzed
...	A list with: testdata=testdata;predictType=c("linear","prob") and impute=FALSE. If impute is set to TRUE it will use the object model to impute missing data

Value

A vector with the predicted values

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

nearestneighborimpute

---

rankInverseNormalDataFrame *Perform a z-transformation of the data using the rank-based inverse normal transformation*

Description

This function takes a data frame and a reference control population to return a z-transformed data set conditioned to the reference population. Each sample data for each feature column in the data frame is conditionally z-transformed using a rank-based inverse normal transformation, based on the rank of the sample in the reference frame.

Usage

```
rankInverseNormalDataFrame(variableList,
                           data,
                           referenceframe,
                           strata=NA)
```

Arguments

variableList	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
data	A data frame where all variables are stored in different columns
referenceframe	A data frame similar to data, but with only the control population
strata	The name of the column in data that stores the variable that will be used to stratify the model

Value

A data frame where each observation has been conditionally z-transformed, given control data

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

Examples

```
## Not run:
# Start the graphics device driver to save all plots in a pdf format
pdf(file = "Example.pdf")
# Get the stage C prostate cancer data from the rpart package
library(rpart)
data(stagec)
# Split the stages into several columns
dataCancer <- cbind(stagec[,c(1:3,5:6)],
                    gleason4 = 1*(stagec[,7] == 4),
                    gleason5 = 1*(stagec[,7] == 5),
                    gleason6 = 1*(stagec[,7] == 6),
                    gleason7 = 1*(stagec[,7] == 7),
                    gleason8 = 1*(stagec[,7] == 8),
                    gleason910 = 1*(stagec[,7] >= 9),
                    eet = 1*(stagec[,4] == 2),
                    diploid = 1*(stagec[,8] == "diploid"),
                    tetraploid = 1*(stagec[,8] == "tetraploid"),
                    notAneuploid = 1-1*(stagec[,8] == "aneuploid"))
# Remove the incomplete cases
dataCancer <- dataCancer[complete.cases(dataCancer),]
# Load a pre-stablished data frame with the names and descriptions of all variables
data(cancerVarNames)
# Set the group of no progression
noProgress <- subset(dataCancer,pgstat==0)
# z-transform g2 values using the no-progression group as reference
```

```
dataCancerZTransform <- rankInverseNormalDataFrame(variableList = cancerVarNames[2,],
                                                   data = dataCancer,
                                                   referenceframe = noProgress)
# Shut down the graphics device driver
dev.off()
## End(Not run)
```

---

reportEquivalentVariables

Report the set of variables that will perform an equivalent IDI discriminant function

---

Description

Given a model, this function will report a data frame with all the variables that may be interchanged in the model without affecting its classification performance. For each variable in the model, this function will loop all candidate variables and report all of which result in an equivalent or better zIDI than the original model.

Usage

```
reportEquivalentVariables(object,
                          pvalue = 0.05,
                          data,
                          variableList,
                          Outcome = "Class",
                          timeOutcome=NULL,
                          type = c("LOGIT", "LM", "COX"),
                          eqPGain = 1.0,
                          description = ".",
                          method="BH",
                          osize=0,
                          fitFRESA=TRUE)
```

Arguments

object	An object of class lm, glm, or coxph containing the model to be analyzed
pvalue	The maximum *p*-value, associated to the IDI , allowed for a pair of variables to be considered equivalent
data	A data frame where all variables are stored in different columns
variableList	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
Outcome	The name of the column in data that stores the variable to be predicted by the model
timeOutcome	The name of the column in data that stores the time to event
type	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")

eqPGain	A fraction to which the z-score will be relaxed, for a pair of variables to be considered equivalent
description	The name of the column in variableList that stores the variable description
method	The method used by the p-value adjustment algorithm
osize	The number of features used for p-value adjustment
fitFRESA	if TRUE it will use the cpp based fitting method

Value

pvalueList	A list with all the unadjusted p-values of the equivalent features per model variable
equivalentMatrix	A data frame with three columns. The first column is the original variable of the model. The second column lists all variables that, if interchanged, will not statistically affect the performance of the model. The third column lists the corresponding z-scores of the IDI for each equivalent variable.
formulaList	a character vector with all the equivalent formulas
equivalentModel	a bagged model that used all the equivalent formulas. The model size is limited by the number of observations

Author(s)

Jose G. Tamez-Pena

---

residualForFRESA *Return residuals from prediction*

Description

Given a model and a new data set, this function will return the residuals of the predicted values. When dealing with a Cox proportional hazards regression model, the function will return the Martingale residuals.

Usage

```
residualForFRESA(object,
                 testData,
                 Outcome,
                 eta = 0.05)
```

Arguments

object	An object of class lm, glm, or coxph containing the model to be analysed
testData	A data frame where all variables are stored in different columns, with the data set to be predicted
Outcome	The name of the column in data that stores the variable to be predicted by the model
eta	The weight of the contribution of the Martingale residuals, or 1 - the weight of the contribution of the classification residuals (only needed if object is of class coxph)

Value

A vector with the residuals (i.e. the differences between the predicted and the real outcome)

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

---

signatureDistance *Distance to the signature template*

Description

This function returns a normalized distance to the signature template

Usage

```
signatureDistance(
              template,
              data=NULL,
              method = c("pearson","spearman","kendall","RMS","MAN")
)
```

Arguments

template	A named template with quantile probs[0.025,0.25,0.5,0.75,0.975] of the signature
data	A data frame that will be used to compute the distance
method	The distance method.

Details

The distance to the template: "pearson","spearman" and "kendall" distances are computed using the correlation function i.e. 1-r. "RMS" distance is the root mean square distance "MAN" is the mean standardized L^1 distance

Value

result	the distance to the template

Author(s)

Jose G. Tamez-Pena

---

summary.bootstrapValidation_Bin

Generate a report of the results obtained using the bootstrapValidation_Bin function

---

Description

This function prints two tables describing the results of the bootstrap-based validation of binary classification models. The first table reports the accuracy, sensitivity, specificity and area under the ROC curve (AUC) of the train and test data set, along with their confidence intervals. The second table reports the model coefficients and their corresponding integrated discrimination improvement (IDI) and net reclassification improvement (NRI) values.

Usage

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

Arguments

object	An object of class bootstrapValidation_Bin
...	Additional parameters for the generic summary function

Value

performance	A vector describing the results of the bootstrapping procedure
summary	An object of class summary.lm, summary.glm, or summary.coxph containing a summary of the analyzed model
coef	A matrix with the coefficients, IDI, NRI, and the 95% confidence intervals obtained via bootstrapping
performance.table	A matrix with the tabulated results of the blind test accuracy, sensitivity, specificities, and area under the ROC curve

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

summaryReport

---

summary.fitFRESA *Returns the summary of the fit*

Description

Returns a summary of fitted model created by the modelFitting function with the fitFRESA parameter set to TRUE

Usage

```
## S3 method for class 'fitFRESA'
summary(object,
type=c("Improvement","Residual"),
ci=c(0.025,0.975),
data=NULL,
...)
```

Arguments

object	fitted model with the modelFitting function
type	the type of coefficient estimation
ci	lower and upper limit of the ci estimation
data	the data to be used for 95
...	parameters of the boostrap method

Value

a list with the analysis results.

Author(s)

Jose G. Tamez-Pena

See Also

modelFitting,bootstrapValidation_Bin,bootstrapValidation_Res

summaryReport *Report the univariate analysis, the cross-validation analysis and the correlation analysis*

Description

This function takes the variables of the cross-validation analysis and extracts the results from the univariate and correlation analyses. Then, it prints the cross-validation results, the univariate analysis results, and the correlated variables. As output, it returns a list of each one of these results.

Usage

```
summaryReport(univariateObject,
              summaryBootstrap,
              listOfCorrelatedVariables = NULL,
              digits = 2)
```

Arguments

`univariateObject`	A data frame that contains the results of the `univariateRankVariables` function
`summaryBootstrap`	A list that contains the results of the `summary.bootstrapValidation_Bin` function
`listOfCorrelatedVariables`	A matrix that contains the `correlated.variables` value from the results obtained with the `listTopCorrelatedVariables` function
`digits`	The number of significant digits to be used in the print function

Value

`performance.table`	A matrix with the tabulated results of the blind test accuracy, sensitivity, specificities, and area under the ROC curve
`coefStats`	A data frame that lists all the model features along with its univariate statistics and bootstrapped coefficients
`cor.varibles`	A matrix that lists all the features that are correlated to the model variables

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

`summary.bootstrapValidation_Bin`

`timeSerieAnalysis` *Fit the listed time series variables to a given model*

Description

This function plots the time evolution and does a longitudinal analysis of time dependent features. Features listed are fitted to the provided time model (mixed effect model) with a generalized least squares (GLS) procedure. As output, it returns the coefficients, standard errors, *t*-values, and corresponding *p*-values.

Usage

```
timeSerieAnalysis(variableList,
                  baseModel,
                  data,
                  timevar = "time",
                  contime = ".",
                  Outcome = ".",
                  ...,
                  description = ".",
                  Ptoshow = c(1),
                  plegend = c("p"),
                  timesign = "-",
                  catgo.names = c("Control", "Case")
                  )
```

Arguments

`variableList`	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
`baseModel`	A string of the type "1 + var1 + var2" that defines the model to which variables will be fitted
`data`	A data frame where all variables are stored in different columns
`timevar`	The name of the column in `data` that stores the visit ID
`contime`	The name of the column in `data` that stores the continuous time (e.g. days or months) that has elapsed since the baseline visit
`Outcome`	The name of the column in `data` that stores an optional binary outcome that may be used to show the stratified analysis
`description`	The name of the column in `variableList` that stores the variable description
`Ptoshow`	Index of the *p*-values to be shown in the plot
`plegend`	Legend of the *p*-values to be shown in the plot
`timesign`	The direction of the arrow of time
`catgo.names`	The legends of the binary categories
`...`	Additional parameters to be passed to the `gls` function

Details

This function will plot the evolution of the mean value of the listed variables with its corresponding error bars. Then, it will fit the data to the provided time model with a GLS procedure and it will plot the fitted values. If a binary variable was provided, the plots will contain the case and control data. As output, the function will return the model coefficients and their corresponding *t*-values, and the standard errors and their associated *p*-values.

Value

`coef`	A matrix with the coefficients of the GLS fitting
`std.Errors`	A matrix with the standardized error of each coefficient
`t.values`	A matrix with the *t*-value of each coefficient
`p.values`	A matrix with the *p*-value of each coefficient
`sigmas`	The root-mean-square error of the fitting

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

`uniRankVar` *Univariate analysis of features (additional values returned)*

Description

This function reports the mean and standard deviation for each feature in a model, and ranks them according to a user-specified score. Additionally, it does a Kolmogorov-Smirnov (KS) test on the raw and *z*-standardized data. It also reports the raw and *z*-standardized *t*-test score, the *p*-value of the Wilcoxon rank-sum test, the integrated discrimination improvement (IDI), the net reclassification improvement (NRI), the net residual improvement (NeRI), and the area under the ROC curve (AUC). Furthermore, it reports the *z*-value of the variable significance on the fitted model. Besides reporting an ordered data frame, this function returns all arguments as values, so that the results can be updates with the `update.uniRankVar` if needed.

Usage

```
uniRankVar(variableList,
           formula,
           Outcome,
           data,
           categorizationType = c("Raw",
                                  "Categorical",
                                  "ZCategorical",
                                  "RawZCategorical",
                                  "RawTail",
                                  "RawZTail",
                                  "Tail",
```

```
                          "RawRaw"),
            type = c("LOGIT", "LM", "COX"),
            rankingTest = c("zIDI",
                            "zNRI",
                            "IDI",
                            "NRI",
                            "NeRI",
                            "Ztest",
                            "AUC",
                            "CStat",
                            "Kendall"),
             cateGroups = c(0.1, 0.9),
             raw.dataFrame = NULL,
             description = ".",
             uniType = c("Binary", "Regression"),
             FullAnalysis=TRUE,
             acovariates = NULL,
             timeOutcome = NULL)
```

Arguments

variableList	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
formula	An object of class formula with the formula to be fitted
Outcome	The name of the column in data that stores an optional binary outcome that may be used to show the stratified analysis
data	A data frame where all variables are stored in different columns
categorizationType	How variables will be analysed : As given in data ("Raw"); broken into the p-value categories given by cateGroups ("Categorical"); broken into the p-value categories given by cateGroups, and weighted by the z-score ("ZCategorical"); broken into the p-value categories given by cateGroups, weighted by the z-score, plus the raw values ("RawZCategorical"); raw values, plus the tails ("RawTail"); or raw values, weighted by the z-score, plus the tails ("RawZTail")
type	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
rankingTest	Variables will be ranked based on: The z-score of the IDI ("zIDI"), the z-score of the NRI ("zNRI"), the IDI ("IDI"), the NRI ("NRI"), the NeRI ("NeRI"), the z-score of the model fit ("Ztest"), the AUC ("AUC"), the Somers' rank correlation ("Cstat"), or the Kendall rank correlation ("Kendall")
cateGroups	A vector of percentiles to be used for the categorization procedure
raw.dataFrame	A data frame similar to data, but with unadjusted data, used to get the means and variances of the unadjusted data
description	The name of the column in variableList that stores the variable description
uniType	Type of univariate analysis: Binary classification ("Binary") or regression ("Regression")

FullAnalysis	If FALSE it will only order the features according to its z-statistics of the linear model
acovariates	the list of covariates
timeOutcome	the name of the Time to event feature

Details

This function will create valid dummy categorical variables if, and only if, data has been z-standardized. The p-values provided in cateGroups will be converted to its corresponding z-score, which will then be used to create the categories. If non z-standardized data were to be used, the categorization analysis would return wrong results.

Value

orderframe	A sorted list of model variables stored in a data frame
variableList	The argument variableList
formula	The argument formula
Outcome	The argument Outcome
data	The argument data
categorizationType	The argument categorizationType
type	The argument type
rankingTest	The argument rankingTest
cateGroups	The argument cateGroups
raw.dataFrame	The argument raw.dataFrame
description	The argument description
uniType	The argument uniType

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

References

Pencina, M. J., D'Agostino, R. B., & Vasan, R. S. (2008). Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond. *Statistics in medicine* **27**(2), 157-172.

See Also

update.uniRankVar, univariateRankVariables

univariateRankVariables *Univariate analysis of features*

Description

This function reports the mean and standard deviation for each feature in a model, and ranks them according to a user-specified score. Additionally, it does a Kolmogorov-Smirnov (KS) test on the raw and z-standardized data. It also reports the raw and z-standardized t-test score, the p-value of the Wilcoxon rank-sum test, the integrated discrimination improvement (IDI), the net reclassification improvement (NRI), the net residual improvement (NeRI), and the area under the ROC curve (AUC). Furthermore, it reports the z-value of the variable significance on the fitted model.

Usage

```
univariateRankVariables(variableList,
                        formula,
                        Outcome,
                        data,
                        categorizationType = c("Raw",
                                               "Categorical",
                                               "ZCategorical",
                                               "RawZCategorical",
                                               "RawTail",
                                               "RawZTail",
                                               "Tail",
                                               "RawRaw"),
                        type = c("LOGIT", "LM", "COX"),
                        rankingTest = c("zIDI",
                                        "zNRI",
                                        "IDI",
                                        "NRI",
                                        "NeRI",
                                        "Ztest",
                                        "AUC",
                                        "CStat",
                                        "Kendall"),
                        cateGroups = c(0.1, 0.9),
                        raw.dataFrame = NULL,
                        description = ".",
                        uniType = c("Binary","Regression"),
                        FullAnalysis=TRUE,
                        acovariates = NULL,
                        timeOutcome = NULL
)
```

Arguments

variableList	A data frame with the candidate variables to be ranked
formula	An object of class formula with the formula to be fitted
Outcome	The name of the column in data that stores the variable to be predicted by the model
data	A data frame where all variables are stored in different columns
categorizationType	How variables will be analysed: As given in data ("Raw"); broken into the *p*-value categories given by cateGroups ("Categorical"); broken into the *p*-value categories given by cateGroups, and weighted by the *z*-score ("ZCategorical"); broken into the *p*-value categories given by cateGroups, weighted by the *z*-score, plus the raw values ("RawZCategorical"); raw values, plus the tails ("RawTail"); or raw values, weighted by the *z*-score, plus the tails ("RawZTail")
type	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
rankingTest	Variables will be ranked based on: The *z*-score of the IDI ("zIDI"), the *z*-score of the NRI ("zNRI"), the IDI ("IDI"), the NRI ("NRI"), the NeRI ("NeRI"), the *z*-score of the model fit ("Ztest"), the AUC ("AUC"), the Somers' rank correlation ("Cstat"), or the Kendall rank correlation ("Kendall")
cateGroups	A vector of percentiles to be used for the categorization procedure
raw.dataFrame	A data frame similar to data, but with unadjusted data, used to get the means and variances of the unadjusted data
description	The name of the column in variableList that stores the variable description
uniType	Type of univariate analysis: Binary classification ("Binary") or regression ("Regression")
FullAnalysis	If FALSE it will only order the features according to its z-statistics of the linear model
acovariates	the list of covariates
timeOutcome	the name of the Time to event feature

Details

This function will create valid dummy categorical variables if, and only if, data has been *z*-standardized. The *p*-values provided in cateGroups will be converted to its corresponding *z*-score, which will then be used to create the categories. If non *z*-standardized data were to be used, the categorization analysis would return wrong results.

Value

A sorted data frame. In the case of a binary classification analysis, the data frame will have the following columns:

Name	Name of the raw variable or of the dummy variable if the data has been categorized
parent	Name of the raw variable from which the dummy variable was created

descrip	Description of the parent variable, as defined in description
cohortMean	Mean value of the variable
cohortStd	Standard deviation of the variable
cohortKSD	D statistic of the KS test when comparing a normal distribution and the distribution of the variable
cohortKSP	Associated p-value to the cohortKSD
caseMean	Mean value of cases (subjects with Outcome equal to 1)
caseStd	Standard deviation of cases
caseKSD	D statistic of the KS test when comparing a normal distribution and the distribution of the variable only for cases
caseKSP	Associated p-value to the caseKSD
caseZKSD	D statistic of the KS test when comparing a normal distribution and the distribution of the z-standardized variable only for cases
caseZKSP	Associated p-value to the caseZKSD
controlMean	Mean value of controls (subjects with Outcome equal to 0)
controlStd	Standard deviation of controls
controlKSD	D statistic of the KS test when comparing a normal distribution and the distribution of the variable only for controls
controlKSP	Associated p-value to the controlsKSD
controlZKSD	D statistic of the KS test when comparing a normal distribution and the distribution of the z-standardized variable only for controls
controlZKSP	Associated p-value to the controlsZKSD
t.Rawvalue	Normal inverse p-value (z-value) of the t-test performed on raw.dataFrame
t.Zvalue	z-value of the t-test performed on data
wilcox.Zvalue	z-value of the Wilcoxon rank-sum test performed on data
ZGLM	z-value returned by the lm, glm, or coxph functions for the z-standardized variable
zNRI	z-value returned by the improveProb function (Hmisc package) when evaluating the NRI
zIDI	z-value returned by the improveProb function (Hmisc package) when evaluating the IDI
zNeRI	z-value returned by the improvedResiduals function when evaluating the NeRI
ROCAUC	Area under the ROC curve returned by the roc function (pROC package)
cStatCorr	c index of Somers' rank correlation returned by the rcorr.cens function (Hmisc package)
NRI	NRI returned by the improveProb function (Hmisc package)
IDI	IDI returned by the improveProb function (Hmisc package)
NeRI	NeRI returned by the improvedResiduals function
kendall.r	Kendall τ rank correlation coefficient between the variable and the binary outcome

kendall.p	Associated *p*-value to the kendall.r
TstudentRes.p	*p*-value of the improvement in residuals, as evaluated by the paired *t*-test
WilcoxRes.p	*p*-value of the improvement in residuals, as evaluated by the paired Wilcoxon rank-sum test
FRes.p	*p*-value of the improvement in residual variance, as evaluated by the *F*-test
caseN_Z_Low_Tail	Number of cases in the low tail
caseN_Z_Hi_Tail	Number of cases in the top tail
controlN_Z_Low_Tail	Number of controls in the low tail
controlN_Z_Hi_Tail	Number of controls in the top tail

In the case of regression analysis, the data frame will have the following columns:

Name	Name of the raw variable or of the dummy variable if the data has been categorized
parent	Name of the raw variable from which the dummy variable was created
descrip	Description of the parent variable, as defined in description
cohortMean	Mean value of the variable
cohortStd	Standard deviation of the variable
cohortKSD	D statistic of the KS test when comparing a normal distribution and the distribution of the variable
cohortKSP	Associated *p*-value to the cohortKSP
cohortZKSD	D statistic of the KS test when comparing a normal distribution and the distribution of the *z*-standardized variable
cohortZKSP	Associated *p*-value to the cohortZKSD
ZGLM	*z*-value returned by the glm or Cox procedure for the z-standardized variable
zNRI	*z*-value returned by the improveProb function (Hmisc package) when evaluating the NRI
NeRI	NeRI returned by the improvedResiduals function
cStatCorr	*c* index of Somers' rank correlation returned by the rcorr.cens function (Hmisc package)
spearman.r	Spearman ρ rank correlation coefficient between the variable and the outcome
pearson.r	Pearson *r* product-moment correlation coefficient between the variable and the outcome
kendall.r	Kendall τ rank correlation coefficient between the variable and the outcome
kendall.p	Associated *p*-value to the kendall.r
TstudentRes.p	*p*-value of the improvement in residuals, as evaluated by the paired *t*-test
WilcoxRes.p	*p*-value of the improvement in residuals, as evaluated by the paired Wilcoxon rank-sum test
FRes.p	*p*-value of the improvement in residual variance, as evaluated by the *F*-test

Author(s)

Jose G. Tamez-Pena

References

Pencina, M. J., D'Agostino, R. B., & Vasan, R. S. (2008). Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond. *Statistics in medicine* **27**(2), 157-172.

`update.uniRankVar` *Update the univariate analysis using new data*

Description

This function updates the results from an univariate analysis using a new data set

Usage

```
## S3 method for class 'uniRankVar'
update(object,
           ...)
```

Arguments

`object`	A list with the results from the `uniRankVar` function
`...`	Additional parameters to be passed to the `uniRankVar` function, used to update the univariate analysis

Value

A list with the same format as the one yielded by the `uniRankVar` function

Author(s)

Jose G. Tamez-Pena

See Also

`uniRankVar`

`updateModel.Bin` *Update the IDI/NRI-based model using new data or new threshold values*

Description

This function will take the frequency-ranked set of variables and will generate a new model with terms that meet either the integrated discrimination improvement (IDI), or the net reclassification improvement (NRI), threshold criteria.

Usage

```
updateModel.Bin(Outcome,
            covariates = "1",
            pvalue = c(0.025, 0.05),
            VarFrequencyTable,
            variableList,
            data,
            type = c("LM", "LOGIT", "COX"),
            lastTopVariable = 0,
            timeOutcome = "Time",
            selectionType = c("zIDI","zNRI"),
            maxTrainModelSize = 0
            )
```

Arguments

`Outcome`	The name of the column in `data` that stores the variable to be predicted by the model
`covariates`	A string of the type "1 + var1 + var2" that defines which variables will always be included in the models (as covariates)
`pvalue`	The maximum *p*-value, associated to either IDI or NRI, allowed for a term in the model
`VarFrequencyTable`	An array with the ranked frequencies of the features, (e.g. the `ranked.var` value returned by the `ForwardSelection.Model.Bin` function)
`variableList`	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
`data`	A data frame where all variables are stored in different columns
`type`	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
`lastTopVariable`	The maximum number of variables to be tested
`timeOutcome`	The name of the column in `data` that stores the time to event (needed only for a Cox proportional hazards regression model fitting)

`selectionType` The type of index to be evaluated by the `improveProb` function (`Hmisc` package): z-score of IDI or of NRI

`maxTrainModelSize` Maximum number of terms that can be included in the model

Value

`final.model` An object of class `lm`, `glm`, or `coxph` containing the final model

`var.names` A vector with the names of the features that were included in the final model

`formula` An object of class `formula` with the formula used to fit the final model

`z.selectionType` A vector in which each term represents the z-score of the index defined in `selectionType` obtained with the Full model and the model without one term

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

`updateModel.Res`

---

updateModel.Res *Update the NeRI-based model using new data or new threshold values*

Description

This function will take the frequency-ranked set of variables and will generate a new model with terms that meet the net residual improvement (NeRI) threshold criteria.

Usage

```
updateModel.Res(Outcome,
                covariates = "1",
                pvalue = c(0.025, 0.05),
                VarFrequencyTable,
                variableList,
                data,
                type = c("LM", "LOGIT", "COX"),
                testType=c("Binomial", "Wilcox", "tStudent"),
                lastTopVariable = 0,
                timeOutcome = "Time",
                maxTrainModelSize = -1
)
```

Arguments

`Outcome`	The name of the column in `data` that stores the variable to be predicted by the model
`covariates`	A string of the type "1 + var1 + var2" that defines which variables will always be included in the models (as covariates)
`pvalue`	The maximum p-value, associated to the NeRI, allowed for a term in the model
`VarFrequencyTable`	An array with the ranked frequencies of the features, (e.g. the `ranked.var` value returned by the `ForwardSelection.Model.Res` function)
`variableList`	A data frame with two columns. The first one must have the names of the candidate variables and the other one the description of such variables
`data`	A data frame where all variables are stored in different columns
`type`	Fit type: Logistic ("LOGIT"), linear ("LM"), or Cox proportional hazards ("COX")
`testType`	Type of non-parametric test to be evaluated by the `improvedResiduals` function: Binomial test ("Binomial"), Wilcoxon rank-sum test ("Wilcox"), Student's t-test ("tStudent"), or F-test ("Ftest")
`lastTopVariable`	The maximum number of variables to be tested
`timeOutcome`	The name of the column in `data` that stores the time to event (needed only for a Cox proportional hazards regression model fitting)
`maxTrainModelSize`	Maximum number of terms that can be included in the model

Value

`final.model`	An object of class `lm`, `glm`, or `coxph` containing the final model
`var.names`	A vector with the names of the features that were included in the final model
`formula`	An object of class `formula` with the formula used to fit the final model
`z.NeRI`	A vector in which each element represents the z-score of the NeRI, associated to the `testType`, for each feature found in the final model

Author(s)

Jose G. Tamez-Pena and Antonio Martinez-Torteya

See Also

`updateModel.Bin`

Index