Package: MUVR2 (via r-universe)

February 15, 2025

Title Multivariate Methods with Unbiased Variable Selection

Version 0.1.0

Author Carl Brunius [aut], Yingxiao Yan [aut, cre]

Maintainer Yingxiao Yan <yingxiao@chalmers.se>

Description Predictive multivariate modelling for metabolomics. Types: Classification and regression. Methods: Partial Least Squares, Random Forest ans Elastic Net Data structures: Paired and unpaired Validation: repeated double cross-validation (Westerhuis et al. (2008)<doi:10.1007/s11306-007-0099-6>, Filzmoser et al. (2009)<doi:10.1002/cem.1225>) Variable selection: Performed internally, through tuning in the inner cross-validation loop.

URL https://github.com/MetaboComp/MUVR2

BugReports https://github.com/MetaboComp/MUVR2/issues

Depends R (>= 3.2.2)

Imports stats, graphics, randomForest, ranger, pROC, doParallel, foreach, caret, glmnet, splines, dplyr, psych, magrittr, mgcv, grDevices, parallel

License GPL-3

LazyData true

RoxygenNote 7.3.2

Encoding UTF-8

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/pak/sysreqs libicu-dev

Repository https://metabocomp.r-universe.dev

RemoteUrl https://github.com/metabocomp/muvr2

RemoteRef HEAD

RemoteSha 861ac7bdd77996a4b611fbffc621be0d51ca873d

Contents

`biplotPLS` *PLS biplot*

Description

Makes a biplot of a fitted object (e.g. from a MUVR with PLS core).

Usage

```
biplotPLS(
  fit,
  comps = 1:2,
  xCol,
  labPlSc = TRUE,
  labs,
  vars,
  labPlLo = TRUE,
  pchSc = 16,
  colSc,
  colLo = 2,
  supLeg = FALSE
)
```

Arguments

`fit`	A PLS fit (e.g. from MUVRclassObject$Fit[[2]])
`comps`	Which components to plot
`xCol`	(Optional) Continuous vector for grey scale gradient of observation (sample) color (e.g. Y vector in regression analysis)
`labPlSc`	Boolean to plot observation (sample) names (defaults to TRUE)
`labs`	(Optional) Label names
`vars`	Which variables to plot (names in rownames(loadings))
`labPlLo`	Boolean to plot variable names (defaults to TRUE)
`pchSc`	Plotting character for observation scores
`colSc`	Colors for observation scores (only if xCol omitted)
`colLo`	Colors for variable loadings (defaults to red)
`supLeg`	Boolean for whether to suppress legends

Value

A PLS biplot

Examples

```
data("freelive2")
nRep <- 2 # Number of MUVR2 repetitions
nOuter <- 3 # Number of outer cross-validation segments
varRatio <- 0.75 # Proportion of variables kept per iteration
method <- 'PLS' # Selected core modeling algorithm
regrModel <- MUVR2(X = XRVIP2,
                   Y = YR2,
                   nRep = nRep,
                   nOuter = nOuter,
                   method = method,
                   modReturn = TRUE)
biplotPLS(regrModel$Fit[[2]],
          comps = 1:2,
          xCol = YR2,
          labPlSc = FALSE,
          labPlLo = FALSE)
```

---

checkinput *Check input*

Description

This can be run to test if the command input of parameters contradict each other and check the structure of the data. If something goes wrong, warning messages are given.

Usage

```
checkinput(
  X,
  Y,
  ML,
  DA,
  method,
  fitness,
  nInner,
  nOuter,
  varRatio,
  scale,
  modReturn,
  logg,
  parallel
)
```

Arguments

X	The original data of X, not the result after onehotencoding
Y	The original data of Y
ML	ML in MUVR2
DA	DA in MUVR2
method	RF or PLS so far in MUVR2
fitness	fitness in MUVR2
nInner	nInnerin MUVR2
nOuter	nOuter in MUVR2
varRatio	varRatio in MUVR2
scale	scale
modReturn	modReturn in MUVR2
logg	logg in MUVR2
parallel	parallel in MUVR2

Value

correct_input: the original input(call) and the real input used in MUVR2 when you enter your input

Examples

```
data("freelive2")
checkinput(X = XRVIP2,
           Y = YR2,  ## YR2 a numeric variable
           DA = FALSE,
           fitness="RMSEP")
```

---

confusionMatrix *Confusion matrix*

Description

Make a confusion matrix from a MUVR object.

Usage

```
confusionMatrix(MVObj, model = "mid")
```

Arguments

MVObj	A MUVR object (classification analysis)
model	min, mid or max model

Value

A confusion matrix of actual vs predicted class

Examples

```
data("mosquito")
data("crisp")
nRep <- 2 # Number of MUVR2 repetitions
nOuter <- 4 # Number of outer cross-validation segments
varRatio <- 0.6 # Proportion of variables kept per iteration
classModel <- MUVR2_EN(X = Xotu,
                       Y = Yotu,
                       nRep = nRep,
                       nOuter = nOuter,
                       DA = TRUE,
                       modReturn = TRUE)
confusionMatrix(classModel)
MLModel <- MUVR2(X = crispEM,
                 ML = TRUE,
                 nRep = nRep,
                 nOuter = nOuter,
                 varRatio = varRatio,
                 method = "RF",
                 modReturn = TRUE)
 confusionMatrix(MLModel)
```

---

crispEM *Effect matrix for the crisp multilevel tutorial*

Description

Effect matrix for the crisp multilevel tutorial

Usage

```
data(crisp)
```

---

customParams *Make custom parameters for internal modelling*

Description

Make custom parameters for MUVR internal modelling, not rdCV. Please note that, at present, there is no mtryMax for the outer (consensus) loop in effect.

Usage

```
customParams(
  method = c("RF", "PLS", "SVM", "ANN"),
  robust = 0.05,
  ntreeIn = 150,
  ntreeOut = 300,
  mtryMaxIn = 150,
  compMax = 5,
  nodes = 200,
  threshold = 0.1,
  stepmax = 1e+08,
  neuralMaxIn = 10,
  kernel = "notkernel",
  nu = 0.1,
  gamma = 1,
  degree = 1,
  oneHot,
  NZV,
  rfMethod = c("randomForest", "ranger"),
  svmMethod = c("svm", "ksvm", "svmlight"),
  annMethod = c("nnet", "neuralnet")
)
```

Arguments

method	PLS or RF (default)
robust	Robustness (slack) criterion for determining min and max knees (defaults to 0.05)
ntreeIn	RF parameter: Number of trees in inner cross-validation loop models (defaults to 150)
ntreeOut	RF parameter: Number of trees in outer (consensus) cross-validation loop models (defaults to 300)
mtryMaxIn	RF parameter: Max number of variables to sample from at each node in the inner CV loop (defaults to 150). Will be further limited by standard RF rules (see randomForest documentation)
compMax	PLS parameter: Maximum number of PLS components (defaults to 5)
nodes	ann parameter:
threshold	ann parameter:
stepmax	ann parameter:
neuralMaxIn	ann parameter: Maximum number of ANN (defaults to 20)
kernel	svm parameter: kernal function to use, which includes sigmoid, radical, polynomial
nu	svm parameter: ratios of errors allowed in the training set range from 0-1
gamma	svm parameters: needed for "vanilladot","polydot","rbfdot" kernel in svm

degree	svm parameter: needed for polynomial kernel in svm
oneHot	TRUE or FALSE using onehot endcoding or not
NZV	TRUE or FALSE using non-zero variance or not
rfMethod	randomforest method, which includes randomForest and ranger
svmMethod	support vector machine method, which includes svm, ksvm, s
annMethod	artificial neural network method which includes 2 different ann methods

Value

a ‘methParam‘ object

Examples

```
# Standard parameters for random forest
methParam <- customParams() # or
methParam <- customParams('RF')
# Custom ntreeOut parameters for random forest
methParam <- customParams('RF',ntreeOut=50) # or
methParam <- customParams('RF')
methParam$ntreeOut <- 50
methParam
```

---

getBER *Get BER*

Description

Get Balanced Error Rate (BER) in classification.

Usage

```
getBER(actual, predicted, weigh_added = FALSE, weighing_matrix)
```

Arguments

actual	Vector of actual classifications of samples
predicted	Vector of predicted classifications of samples
weigh_added	To add a weighing matrix when it is classification
weighing_matrix	The matrix used to get a misclassification score

Value

BER

Examples

```
data("mosquito")
actual <- Yotu
predicted <- sampling_from_distribution(actual)
getBER(actual, predicted)
```

`getMISS` *Get number of misclassifications*

Description

Get number of misclassifications from classification analysis.

Usage

```
getMISS(actual, predicted, weigh_added = FALSE, weighing_matrix)
```

Arguments

`actual`	Vector of actual classifications of samples
`predicted`	Vector of predicted classifications of samples
`weigh_added`	Boolean, add a weighing matrix when it is classification
`weighing_matrix`	The matrix used to get a misclassification score

Value

number of misclassifications

Examples

```
data("mosquito")
actual <- Yotu
predicted <- sampling_from_distribution(actual)
getMISS(actual, predicted)
```

getVar *Get min, mid or max model from Elastic Net modelling*

Description

Obtain the min, mid, or max number of variables for an object generated from the rdCVnet() function.

Usage

```
getVar(
  rdCVnetObject,
  option = c("quantile", "fitness"),
  fit_curve = c("loess", "gam"),
  span = 1,
  k = 5,
  outlier = c("none", "IQR", "residual"),
  robust = 0.05,
  quantile = 0.25
)
```

Arguments

rdCVnetObject	an object obtained from the rdCVnet() function
option	quantile or fitness: which way to perform variable selection
fit_curve	gam or loess method for fitting the curve in the fitness option
span	parameter for using loess to fit curve in the fitness option: how smooth the curve needs to be
k	parameter for using gam to fit curve in the fitness option
outlier	if remove outlier variables or not. There are 3 options: "none","IRQ", "residual"
robust	if the option is fitness, robust parameter decides how much deviation it is allowed from the optimal perdiction performance for min and max variabel selection
quantile	if the option is quantile, this value decides the cut for the first quantile, ranging from 0 to 0.5

Value

a rdCVnet object

Examples

```
data("mosquito")
nRep <- 2
nOuter <- 4
```

```
varRatio <-0.6
classModel <- MUVR2_EN(X = Xotu,
                       Y = Yotu,
                       nRep = nRep,
                       nOuter = nOuter,
                       DA = TRUE,
                       modReturn = TRUE)
classModel<-getVar(classModel)
```

---

getVIRank *Get variable importance*

Description

Extract autoselected variables from MUVR model object.

Usage

```
getVIRank(MUVRclassObject, model = "mid", n, all = FALSE)
```

Arguments

MUVRclassObject	an object of MUVR class
model	which model to use ("min", "mid" (default), or "max")
n	customize values
all	logical, to get the ranks of all variable or not

Value

data frame with order, name and average rank of variables (‘order‘, ‘name‘ & ‘rank‘)

Examples

```
data("freelive2")
nRep <- 2
nOuter <- 4
varRatio <-0.6
regrModel <- MUVR2(X = XRVIP2,
                   Y = YR2,
                   nRep = nRep,
                   nOuter = nOuter,
                   varRatio = varRatio,
                   method = "PLS",
                   modReturn = TRUE)
getVIRank(regrModel, model="min")
```

get_rmsep *Get RMSEP*

Description

Get Root Mean Square Error of Prediction (RMSEP) in classification.

Usage

```
get_rmsep(actual, predicted)
```

Arguments

actual	Vector of actual classifications of samples
predicted	Vector of predicted classifications of samples

Value

RMSEP

Examples

```
data("mosquito")
actual <- YR2
predicted <- sampling_from_distribution(actual)
get_rmsep(actual, predicted)
```

H0_reference *Get reference distribution for resampling tests*

Description

Make reference distribution for resampling tests to assess overfitting.

Usage

```
H0_reference(Y, n = 1000, fitness = c("Q2", "BER", "MISS", "AUROC"), ...)
```

Arguments

Y	the target variable
n	number of permutations to run
fitness	number of repetitions for each permutation (defaults to value of actual model)
...	additional arguments for sampling from distribution

Value

a histogram of reference distribution

Examples

```
data("freelive2")
H0_reference(YR2)
```

`H0_test` *Perform permutation or resampling tests*

Description

This function will extract data and parameter settings from a MUVR object and run standard permutation or resampling test. This will fit a standard case of multivariate predictive modelling in either a regression, classification or multilevel case. However, if an analysis has a complex sample dependency which requires constrained permutation of your response vector or if a variable pre-selection is performed for decreased computational burden, then permutaion/resampling loops should be constructed manually. In those cases, View(H0_test) can be a first start from which to build custom solutions for permutation analysis.

Usage

```
H0_test(
  MUVRclassObject,
  n = 50,
  nRep,
  nOuter,
  varRatio,
  parallel,
  type = c("resampling", "permutation")
)
```

Arguments

`MUVRclassObject`	a 'MUVR' class object
`n`	number of permutations to run
`nRep`	number of repetitions for each permutation (defaults to value of actual model)
`nOuter`	number of outer validation segments for each permutation (defaults to value of actual model)
`varRatio`	varRatio for each permutation (defaults to value of actual model)
`parallel`	whether to run calculations using parallel processing which requires registered backend (defaults to parallelization for the actual model)
`type`	either permutation or resampling, to decide whether the permutation sampling is performed on original Y values or the probability(If Y categorical)/distributions(If Y continuous) of Y values

Value

permutation_output: A permutation matrix with permuted fitness statistics (nrow=n and ncol=3 for min/mid/max)

Examples

```
data("freelive2")
nRep <- 2
nOuter <- 4
varRatio <-0.6
regrModel <- MUVR2(X = XRVIP2,
                   Y = YR2,
                   nRep = nRep,
                  nOuter = nOuter,
                   varRatio = varRatio,
                   method = "PLS",
                   modReturn = TRUE)
H0_test(regrModel)
```

IDR *Subject identifiers for the rye metabolomics regression tutorial*

Description

Subject identifiers for the rye metabolomics regression tutorial

Usage

```
data(freelive)
```

IDR2 *Subject identifiers for the rye metabolomics regression tutorial, using unique individuals*

Description

Subject identifiers for the rye metabolomics regression tutorial, using unique individuals

Usage

```
data(freelive2)
```

mergeModels *Merge two MUVR class objects*

Description

Merge two MUVR class objects that use regression for PLS or RF methods. The resultant MUVR class object has the same indata except that nRep is different.

Usage

```
mergeModels(MV1, MV2)
```

Arguments

MV1	a MUVR class Object
MV2	a MUVR class Object

Value

A merged MURV class object

Examples

```
data("freelive2")
nRep <- 2
nOuter <- 4
varRatio <-0.6
regrModel <- MUVR2(X = XRVIP2,
                   Y = YR2,
                   nRep = nRep,
                  nOuter = nOuter,
                   varRatio = varRatio,
                   method = "PLS",
                   modReturn = TRUE)
mergedModel<-mergeModels(regrModel,regrModel)
```

MUVR2 *MUVR2 with PLS and RF*

Description

"Multivariate modelling with Unbiased Variable selection" using PLS and RF. Repeated double cross validation with tuning of variables in the inner loop.

Usage

```
MUVR2(
  X,
  Y,
  ID,
  scale = TRUE,
  nRep = 5,
  nOuter = 6,
  nInner,
  varRatio = 0.75,
  DA = FALSE,
  fitness = c("AUROC", "MISS", "BER", "RMSEP", "wBER", "wMISS"),
  method = c("PLS", " RF", "ANN", "SVM"),
  methParam,
  ML = FALSE,
  modReturn = FALSE,
  logg = FALSE,
  parallel = TRUE,
  weigh_added = FALSE,
  weighing_matrix = NULL,
  keep,
  ...
)
```

Arguments

X	Predictor variables. NB: Variables (columns) must have names/unique identifiers. NAs not allowed in data. For multilevel, only the positive half of the difference matrix is specified.
Y	Response vector (Dependent variable). For classification, a factor (or character) variable should be used. For multilevel, Y is calculated automatically.
ID	Subject identifier (for sampling by subject; Assumption of independence if not specified)
scale	If TRUE, the predictor variable matrix is scaled to unit variance for PLS modeling.
nRep	Number of repetitions of double CV. (Defaults to 5)
nOuter	Number of outer CV loop segments. (Defaults to 6)
nInner	Number of inner CV loop segments. (Defaults to nOuter - 1)
varRatio	Ratio of variables to include in subsequent inner loop iteration. (Defaults to 0.75)
DA	Boolean for Classification (discriminant analysis) (By default, if Y is numeric -> DA = FALSE. If Y is factor (or character) -> DA = TRUE)
fitness	Fitness function for model tuning (choose either 'AUROC' or 'MISS' (default) for classification; or 'RMSEP' (default) for regression.)
method	Multivariate method. Supports 'PLS' and 'RF' (default)

methParam	List with parameter settings for specified MV method (see function code for details)
ML	Boolean for multilevel analysis (defaults to FALSE)
modReturn	Boolean for returning outer segment models (defaults to FALSE). Setting modReturn = TRUE is required for making MUVR predictions using predMV().
logg	Boolean for whether to sink model progressions to 'log.txt'
parallel	Boolean for whether to perform 'foreach' parallel processing (Requires a registered parallel backend; Defaults to 'TRUE')
weigh_added	To add a weighing matrix when it is classfication
weighing_matrix	The matrix used for get a miss classfication score
keep	Confounder variables can be added. NB: Variables (columns) must match column names.
...	additional argument

Value

A 'MUVR' object

Examples

```
data(freelive2)
nRep <- 2 # Number of MUVR2 repetitions
nOuter <- 3 # Number of outer cross-validation segments
varRatio <- 0.6 # Proportion of variables kept per iteration
method <- 'PLS' # Selected core modeling algorithm
regrModel <- MUVR2(X = XRVIP2,
                   Y = YR2,
                   nRep = nRep,
                   nOuter = nOuter,
                   varRatio = varRatio,
                   method = method,
                   modReturn = TRUE)
```

MUVR2_EN *MUVR2 with EN*

Description

"Multivariate modelling with Unbiased Variable selection" using Elastic Net (EN). Repeated double cross validation with tuning of variables using Elastic Net.

Usage

```
MUVR2_EN(
  X,
  Y,
  ID,
  alow = 1e-05,
  ahigh = 1,
  astep = 11,
  alog = TRUE,
  nRep = 5,
  nOuter = 6,
  nInner,
  NZV = TRUE,
  DA = FALSE,
  fitness = c("AUROC", "MISS", "BER", "RMSEP", "wBER", "wMISS"),
  methParam,
  ML = FALSE,
  modReturn = FALSE,
  parallel = TRUE,
  keep = NULL,
  weigh_added = FALSE,
  weighing_matrix = NULL,
  ...
)
```

Arguments

X	Predictor variables. NB: Variables (columns) must have names/unique identifiers. NAs not allowed in data. For multilevel, only the positive half of the difference matrix is specified.
Y	Response vector (Dependent variable). For classification, a factor (or character) variable should be used. For multilevel, Y is calculated automatically.
ID	Subject identifier (for sampling by subject; Assumption of independence if not specified)
alow	alpha tuning: lowest value of alpha
ahigh	alpha tuning: highest value of alpha
astep	alpha tuning: number of alphas to try from low to high
alog	alpha tuning: Whether to space tuning of alpha in logarithmic scale (TRUE; default) or normal/arithmetic scale (FALSE)
nRep	Number of repetitions of double CV. (Defaults to 5)
nOuter	Number of outer CV loop segments. (Defaults to 6)
nInner	Number of inner CV loop segments. (Defaults to nOuter-1)
NZV	Boolean for whether to filter out near zero variance variables (defaults to TRUE)
DA	Boolean for Classification (discriminant analysis) (By default, if Y is numeric -> DA=FALSE. If Y is factor (or character) -> DA=TRUE)

fitness	Fitness function for model tuning (choose either 'AUROC' or 'MISS' (default) for classification; or 'RMSEP' (default) for regression.)
methParam	List with parameter settings for specified MV method (see function code for details)
ML	Boolean for multilevel analysis (defaults to FALSE)
modReturn	Boolean for returning outer segment models (defaults to FALSE). Setting modReturn=TRUE is required for making MUVR predictions using predMV().
parallel	Boolean for whether to perform 'foreach' parallel processing (Requires a registered parallel backend; Defaults to 'TRUE')
keep	A group of confounders that you want to manually set as non-zero
weigh_added	weigh_added
weighing_matrix	weighing_matrix
...	Pass additional arguments

Value

A MUVR object

Examples

```
data("freelive2")
nRep <- 2 # Number of MUVR2 repetitions
nOuter <- 4 # Number of outer cross-validation segments
regrModel <- MUVR2_EN(X = XRVIP2,
                      Y = YR2,
                      nRep = nRep,
                      nOuter = nOuter,
                      modReturn = TRUE)
```

---

nearZeroVar *Identify variables with near zero variance*

Description

Adapted and stripped down from mixOmics v 5.2.0 (https://cran.r-project.org/web/packages/mixOmics/).

Usage

```
nearZeroVar(x, freqCut = 95/5, uniqueCut = 10)
```

Arguments

x	a numeric vector or matrix, or a data frame with all numeric data.
freqCut	the cutoff for the ratio of the most common value to the second most common value.
uniqueCut	the cutoff for the percentage of distinct values out of the number of total samples.

Value

nzv object

Examples

```
data("freelive2")
nearZeroVar(XRVIP2)
data("mosquito")
nearZeroVar(Xotu)
```

---

onehotencoding *One hot encoding*

Description

Each factor and character variable with n categories(>2) will be transformed to n variables. Each factor and character variable with 2 categories will be transformed to one 01 numeric dummy variable. Each factor and character variable with 1 categories will be transformed to one numeric variable that only has value 1. Each factor and character variable with 0 categories will be transformed to one numeric variable that only has value -999. Each logical variable will be transformed to one 01 numeric dummy variable.

Usage

```
onehotencoding(X)
```

Arguments

X	data frame data with numeric, factor, character and/or logical variables

Value

matrix with all variables transformed to numeric variables

Examples

```
#To test the scenario when X has factor and character when using PLS
#add one factor and one character variable(freelive data X,
# which originally has 112 numeric samples and 1147 observations)
# factor variable has 3,6,5factors(nearzero variance), character variable has 7,4 categories
factor_variable1<-as.factor(c(rep("33",105),rep("44",3),rep("55",4)))
factor_variable2<-as.factor(c(rep("AB",20),rep("CD",10),rep("EF",30),
                          rep("GH",15),rep("IJ",25),rep("KL",12)))
factor_variable3<-as.factor(c(rep("Tessa",25),rep("Olle",30),rep("Yan",12),
                           rep("Calle",25),rep("Elisa",20)))
factor_variable4<-as.factor(c(rep(NA,112)))
character_variable1<-c(rep("one",16),rep("two",16),rep("three",16),
                       rep("four",16),rep("five",16),rep("six",16),rep("seven",16))
character_variable2<-c(rep("yes",28),rep("no",28),
```

```
                          rep("yes",28),rep("no",28))
character_variable3<-c(rep("Hahahah",112))
character_variable4<-as.character(c(rep(NA,112)))
logical_variable1<-c(rep(TRUE,16),rep(FALSE,16),rep(TRUE,16),
rep(FALSE,16),rep(TRUE,16),rep(FALSE,32))
logical_variable2<-c(rep(TRUE,28),rep(FALSE,28),rep(TRUE,28),rep(FALSE,28))

 X<-data.frame(row.names<-1:112)
 X<-cbind(X,XRVIP,
       factor_variable1,factor_variable2,factor_variable3,factor_variable4,
       character_variable1,character_variable2,character_variable3,character_variable4,
        logical_variable1,logical_variable2)
  onehotencoding(X)
```

`permutationPlot` *Plot permutation analysis*

Description

Plot permutation analysis using actual model and permutation result. This is basically a wrapper for the MUVR2::plotPerm() function using model objects to make coding nicer and cleaner.

Usage

```
permutationPlot(
  MUVRclassObject,
  permutation_result,
  model = "Mid",
  type = "t",
  side = c("greater", "smaller"),
  pos,
  xlab = NULL,
  xlim,
  ylim = NULL,
  breaks = "Sturges",
  main = NULL
)
```

Arguments

`MUVRclassObject`	A 'MUVR' class object
`permutation_result`	A permutation result. It is a list of 1 items: permutation_output
`model`	'Min', 'Mid', or 'Max'
`type`	't' (default; for Student's t) or 'non' for "non-parametric" (i.e. rank) studen'ts
`side`	'smaller' for actual lower than H0 or 'greater' for actual larger than H0 (automatically selected if not specified)

pos	which side of actual to put p-value on
xlab	optional xlabel
xlim	optional x-range
ylim	otional y-range
breaks	optional custom histogram breaks (defaults to 'sturges')
main	optional plot title (or TRUE for autoname)

Value

A permutation plot

Examples

```
data("freelive2")
nRep <- 2
nOuter <- 4
varRatio <-0.6
regrModel <- MUVR2(X = XRVIP2,
                   Y = YR2,
                   nRep = nRep,
                  nOuter = nOuter,
                   varRatio = varRatio,
                   method = "PLS",
                   modReturn = TRUE)
permutation_result<-H0_test(regrModel,n=10)
permutationPlot(regrModel,permutation_result)
```

---

plotMV *Plot predictions*

Description

Plot predicted and actual target variables, with different plots depending on modelling approach.

Usage

```
plotMV(MUVRclassObject, model = "min", factCols, sampLabels, ylim = NULL)
```

Arguments

MUVRclassObject	An MUVR class object
model	What type of model to plot ('min', 'mid' or 'max'). Defaults to 'mid'.
factCols	An optional vector with colors for the factor levels (in the same order as the levels)
sampLabels	Sample labels (optional; implemented for classification)
ylim	Optional for imposing y-limits for regression and classification analysis

Value

A plot of results from multivariate predictions

Examples

```
data("freelive2")
nRep <- 2
nOuter <- 4
varRatio <-0.6
regrModel <- MUVR2(X = XRVIP2,
                   Y = YR2,
                   nRep = nRep,
                  nOuter = nOuter,
                   varRatio = varRatio,
                   method = "PLS",
                   modReturn = TRUE)
plotMV(regrModel, model="min")
```

---

plotPCA *PCA score plot*

Description

Customised PCA score plots with the possibility to choose PCs, exporting to png and the possibility to add color or different plotting symbols according to variable.

Usage

```
plotPCA(pca, PC1 = 1, PC2 = 2, file, colVar, symbVar, main = "")
```

Arguments

pca	A ‘prcomp‘ object
PC1	Principal component on x-axis
PC2	Principal component on y-axis
file	If specified provides the name of a png export file. Otherwise normal plot.
colVar	Continuous variable for coloring observations (40 cuts)
symbVar	Categorical/discrete variable for multiple plot symbols
main	If provided provides a main title of the plot

Value

A PCA score plot. Exported as png if ‘file‘ specified in function call.

Examples

```
data("freelive2")
pca_object<-prcomp(XRVIP2)
plotPCA(pca_object)
```

`plotPerm` *Plot for comparison of actual model fitness vs permutation/resampling*

Description

Plots histogram of null hypothesis (permutation/resampling) distribution, actual model fitness and cumulative p-value. Plot defaults to "greater than" or "smaller than" tests and cumulative probability in Student's t-distribution.

Usage

```
plotPerm(
  actual,
  distribution,
  xlab = NULL,
  side = c("greater", "smaller"),
  type = "t",
  ylab = NULL,
  xlim,
  ylim = NULL,
  breaks = "Sturges",
  pos,
  main = NULL,
  permutation_visual = "none",
  curve = TRUE,
  extend = 0.1,
  multiple_p_shown = NULL,
  show_actual_value = TRUE,
  show_p = TRUE,
  round_number = 4
)
```

Arguments

`actual`	Actual model fitness (e.g. Q2, AUROC or number of misclassifications)
`distribution`	Null hypothesis (permutation) distribution of similar metric as 'actual'
`xlab`	Label for x-axis (e.g. 'Q2 using real value',"Q2 using distributions","BER" 'AUROC', or 'Misclassifications')
`side`	Cumulative p either "greater" or "smaller" than H0 distribution (defaults to side of median(H0))
`type`	c('t','non',"smooth","rank","ecdf")

ylab	label for y-axis
xlim	Choice of user-specified x-limits (if default is not adequate)
ylim	Choice of user-specified y-limits (if default is not adequate)
breaks	Choice of user-specified histogram breaks (if default is not adequate)
pos	Choice of position of p-value label (if default is not adequate)
main	Choice of user-specified plot title
permutation_visual	choice of showing median or mean or none
curve	if add curve or not base on the mid
extend	how many percenrtage of the orignical range do we start
multiple_p_shown	show many p values
show_actual_value	show the actual value on the vertical line or not
show_p	if p value is added to the figure
round_number	How many digits does it keep

Value

Plot

Examples

```
data("freelive2")
actual <- sample(YR2, 1)
distribution <- YR2
plotPerm (actual, distribution)
```

---

plotPred *Plot predictions for PLS regression*

Description

At present, this function only supports predictions for PLS regression type problems.

Usage

```
plotPred(Ytrue, Ypreds)
```

Arguments

Ytrue	True value of Y, should be a vector
Ypreds	Predicted value of Y can be a vector or data frame with the same number of rows

Value

A plot, plot the prediction

Examples

```
data("freelive2")
Ytrue<-YR2
Ypreds<-sampling_from_distribution(YR2)
plotPred(Ytrue,Ypreds)
Ytrue<-YR2
nRep <- 2
nOuter <- 4
varRatio <-0.6
regrModel <- MUVR2(X = XRVIP2,
                   Y = YR2,
                   nRep = nRep,
                   nOuter = nOuter,
                   varRatio = varRatio,
                   method = "PLS",
                   modReturn = TRUE)
Ypreds<-regrModel$yPred
plotPred(Ytrue,Ypreds)
```

---

plotStability *Plot stability*

Description

Plot stability of selected variables and prediction fitness as a function of number of repetitions.

Usage

```
plotStability(MUVRrdCVclassObject, model = "min", VAll, nVarLim, missLim)
```

Arguments

MUVRrdCVclassObject	MUVR class object or rdCV object
model	'min' (default), 'mid' or 'max'
VAll	Option of specifying which variables (i.e. names) to consider as reference set. Defaults to variables selected from the 'model' of the 'MUVRrdCVclassObject'
nVarLim	Option of specifying upper limit for number of variables
missLim	Option of specifying upper limit for number of misclassifications

Value

Plot of number of variables, proportion of variables overlapping with reference and prediction accuracy (Q2 for regression; MISS otherwise) as a function of number of repetitions.

Examples

```
data("freelive2")
nRep <- 2
nOuter <- 4
varRatio <-0.6
regrModel <- MUVR2(X = XRVIP2,
                   Y = YR2,
                   nRep = nRep,
                  nOuter = nOuter,
                   varRatio = varRatio,
                   method = "PLS",
                   modReturn = TRUE)
plotStability(regrModel, model = "min")
```

---

plotVAL *Plot validation metric*

Description

Produces a plot of validation metric vs number of variables in model (inner segment).

Usage

```
plotVAL(MUVRclassObject, show_outlier = TRUE)
```

Arguments

MUVRclassObject	An object of class 'MUVR'
show_outlier	Boolean, show outliers

Value

A plot

Examples

```
data("freelive2")
nRep <- 2
nOuter <- 4
varRatio <-0.6
regrModel <- MUVR2(X = XRVIP2,
                   Y = YR2,
                   nRep = nRep,
                  nOuter = nOuter,
                   varRatio = varRatio,
                   method = "PLS",
                   modReturn = TRUE)
```

```
plotVAL(regrModel)
```

`plotVIRank` *Plot variable importance ranking*

Description

Plot variable importance ranking in MUVR object. Regardless of MV core method, variables are sorted by rank, where lower is better. ‘plotVIRank‘ produces boxplots of variable rankings for all model repetitions.

Usage

```
plotVIRank(
  MUVRclassObject,
  n,
  model = "min",
  cut,
  maptype = c("heatmap", "dotplot"),
  add_blank = 4,
  cextext = 1
)
```

Arguments

`MUVRclassObject`	An MUVR class object only applied to PLS, RF not rdCVnet
`n`	Number of top ranking variables to plot (defaults to those selected by MUVR2)
`model`	Which model to choose (’min’, ’mid’ (default) or ’max’)
`cut`	Optional value to cut length of variable names to ‘cut‘ number of characters
`maptype`	for rdCvnet dot plot or heat map
`add_blank`	put more blank when the rownames is too long,
`cextext`	the cex of the text

Value

Barplot of variable rankings (lower is better)

Examples

```
data("freelive2")
nRep <- 2
nOuter <- 4
varRatio <-0.6
regrModel <- MUVR2(X = XRVIP2,
```

```
                    Y = YR2,
                    nRep = nRep,
                   nOuter = nOuter,
                    varRatio = varRatio,
                    method = "PLS",
                    modReturn = TRUE)
plotVIRank(regrModel, n=20)
```

---

pPerm *Calculate permutation p-value Calculate perutation p-value of actual model performance vs null hypothesis distribution. 'pPerm' will calculate the cumulative (1-tailed) probability of 'actual' belonging to 'permutation_distribution'. 'side' is guessed by actual value compared to median(permutation_distribution). Test is performed on original data OR ranked for non-parametric statistics.*

Description

Calculate permutation p-value Calculate perutation p-value of actual model performance vs null hypothesis distribution. 'pPerm' will calculate the cumulative (1-tailed) probability of 'actual' belonging to 'permutation_distribution'. 'side' is guessed by actual value compared to median(permutation_distribution). Test is performed on original data OR ranked for non-parametric statistics.

Usage

```
pPerm(
  actual,
  permutation_distribution,
  side = c("smaller", "greater"),
  type = "t",
  extend = 0.1
)
```

Arguments

actual	Actual model performance (e.g. misclassifications or Q2)
permutation_distribution	Null hypothesis distribution from permutation test (same metric as 'actual')
side	Smaller or greater than (automatically guessed if omitted) (Q2 and AUC is a "greater than" test, whereas misclassifications is "smaller than")
type	one of ('t','non',"smooth","ecdf","rank")
extend	extend how much it extend

Value

p-value

Examples

```
data("freelive2")
actual <- sample(YR2, 1)
permutation_distribution <- YR2
pPerm(actual, permutation_distribution)
```

---

predMV *Predict outcomes Predict MV object using a MUVR class object and a X testing set. At present, this function only supports predictions for PLS regression type problems.*

Description

Predict outcomes Predict MV object using a MUVR class object and a X testing set. At present, this function only supports predictions for PLS regression type problems.

Usage

```
predMV(MUVRclassobject, newdata, model = "min")
```

Arguments

MUVRclassobject	An 'MUVR' class object
newdata	New data for which to predict outcomes
model	What type of model to plot ('min', 'mid' or 'max'). Defaults to 'mid'.

Value

The predicted result based on the MUVR model and the newdata

Examples

```
data("freelive2")
nRep <- 2
nOuter <- 4
varRatio <-0.6
regrModel <- MUVR2(X = XRVIP2,
                   Y = YR2,
                   nRep = nRep,
                   nOuter = nOuter,
                   varRatio = varRatio,
                   method = "PLS",
                   modReturn=TRUE)
predMV(regrModel,XRVIP2)
```

preProcess *Perform matrix pre-processing*

Description

Perform matrix pre-processing

Usage

```
preProcess(
  X,
  offset = 0,
  zeroOffset = 0,
  trans = "none",
  center = "none",
  scale = "none"
)
```

Arguments

X	Data matrix with samples in rows and variables in columns
offset	Add offset to all data points (defaults to 0)
zeroOffset	Add offset to zero data (defaults to 0)
trans	Either 'log', 'sqrt' or 'none' (default is 'none')
center	Either 'mean', 'none' or a numeric vector of length equal to the number of columns of X (defaults to 'none').
scale	Either 'UV', 'Pareto', 'none' or a numeric vector of length equal to the number of columns of X (defaults to 'none').

Value

A pre-processed data matrix

Examples

```
data("freelive2")
preProcess(XRVIP2)
```

Q2_calculation *Q2 calculation*

Description

Q2 calculation

Usage

```
Q2_calculation(yhat, y)
```

Arguments

yhat	prediction values
y	real values

Value

Q2

Examples

```
data("freelive2")
actual <- YR2
predicted <- MUVR2::sampling_from_distribution(actual)
Q2_calculation(actual, predicted)
```

qMUVR2 *Wrapper for speedy access to MUVR2 (autosetup of parallelization)*

Description

Wrapper for speedy access to MUVR2 (autosetup of parallelization)

Usage

```
qMUVR2(
  X,
  Y,
  ML = FALSE,
  method = "RF",
  varRatio = 0.65,
  nCore,
  repMult = 1,
  nOuter = 5,
  ...
)
```

Arguments

X	X-data
Y	Y-data
ML	Boolean for multilevel
method	'RF' (default) or 'PLS'
varRatio	proportion of variables to keep in each loop of the recursive feature elimination
nCore	Number of threads to use for calculation (defaults to detectCores()-1)
repMult	Multiplier of cores -> nRep = repMult * nCore
nOuter	Number of outer segments
...	Additional arguments(see MUVR)

Value

MUVR object

Examples

```
data("freelive2")
regrModel <- qMUVR2(X = XRVIP2,
                    Y = YR2,
                    nCore = 1)
```

---

rdCV *Wrapper for repeated double cross-validation without variable selection*

Description

Wrapper for repeated double cross-validation without variable selection

Usage

```
rdCV(
  X,
  Y,
  ID,
  nRep = 5,
  nOuter = 6,
  nInner,
  DA = FALSE,
  fitness = c("AUROC", "MISS", "RMSEP", "BER"),
  method = c("PLS", "RF"),
  methParam,
  ML = FALSE,
```

```
  modReturn = FALSE,
  logg = FALSE
)
```

Arguments

X	Independent variables. NB: Variables (columns) must have names/unique identifiers. NAs not allowed in data. For ML, X is upper half only (X1-X2)
Y	Response vector (Dependent variable). For DA (classification), Y should be factor or character. For ML, Y is omitted. For regression, Y is numeric.
ID	Subject identifier (for sampling by subject; Assumption of independence if not specified)
nRep	Number of repetitions of double CV.
nOuter	Number of outer CV loop segments.
nInner	Number of inner CV loop segments.
DA	Logical for Classification (discriminant analysis) (Defaults do FALSE, i.e. regression). PLS is limited to two-class problems (see 'Y' above).
fitness	Fitness function for model tuning (choose either 'AUROC' or 'MISS'or 'BER' for classification; or 'RMSEP' (default) for regression.)
method	Multivariate method. Supports 'PLS' and 'RF' (default)
methParam	List with parameter settings for specified MV method (defaults to ???)
ML	Logical for multilevel analysis (defaults to FALSE)
modReturn	Logical for returning outer segment models (defaults to FALSE)
logg	Logical for whether to sink model progressions to 'log.txt'

Value

An object containing stuff...

Examples

```
data("freelive2")
nRep <- 2 # Number of MUVR2 repetitions
nOuter <- 4 # Number of outer cross-validation segments
varRatio <- 0.75 # Proportion of variables kept per iteration
method <- 'RF' # Selected core modeling algorithm
regrModel <- rdCV(X = XRVIP2,
                  Y = YR2,
                  nRep = nRep,
                  nOuter = nOuter,
                  method = method,
                  modReturn = TRUE)
```

rdcvNetParams *Make custom parameters for rdcvNet internal modelling*

Description

Custom parameters can be set in the function call or by manually setting "slots" in the resulting methParam object.

Usage

```
rdcvNetParams(
  robust = 0.05,
  family = "gaussian",
  nRepInner = 1,
  NZV = TRUE,
  oneHot = TRUE
)
```

Arguments

robust	Robustness (slack) criterion for determining min and max knees (defaults to 0.05)
family	the options could be "gaussian", "binomial", "poisson", "multinomial", "cox", "mgaussian"
nRepInner	how many nRepInner
NZV	NZV
oneHot	TRUE or FALSE using onehot endcoding or not

Value

a ‘methParam‘ object

Examples

```
# Standard parameters for rdcvNet
methParam <- rdcvNetParams()
```

sampling_from_distribution *Sampling from the distribution of something*

Description

Sampling from the distribution of something

Usage

```
sampling_from_distribution(X, upperlimit, lowerlimit, extend, n)
```

Arguments

X	a vector (numeric or factor) where the distribution/probility will be generated
upperlimit	if X is numeric, set upper limit
lowerlimit	if X is numeric, set lower limit
extend	If X is numeric, how much you want to extend from the lower and upper existing X.
n	How many you want to sample

Value

a resampled thing

Examples

```
data("mosquito")
sampling_from_distribution(Yotu)
data("freelive2")
sampling_from_distribution(YR2,
                           upperlimit=200,
                           lowerlimit=0,
                           n=length(YR2)
                           )
```

varClass *Report variables belonging to different classes*

Description

Reports names and numbers of variables: all as well as optimal (min model), redundant (from min up to max) and noisy (the rest).

Usage

```
varClass(MUVRclassObject)
```

Arguments

MUVRclassObject	A MUVR class object

Value

A list with names and numbers of variables: all as well as optimal (Corresponding to 'min' or minial-optimal model), redundant (from 'min' up to 'max' or all-relevant) and noisy (the rest)

Examples

```
data("mosquito")
nRep <- 2
nOuter <- 4
classModel <- MUVR2_EN(X = Xotu,
                       Y = Yotu,
                       nRep = nRep,
                       nOuter = nOuter,
                       DA = TRUE,
                       modReturn = TRUE)
classModel<-getVar(classModel,option="quantile")
varClass(classModel)
```

Xotu *Microbiota composition in mosquitos for the classification tutorial*

Description

Microbiota composition in mosquitos for the classification tutorial

Usage

```
data(mosquito)
```

XRVIP *Metabolomics data for the rye metabolomics regression tutorial*

Description

Metabolomics data for the rye metabolomics regression tutorial

Usage

```
data(freelive)
```

XRVIP2 *Metabolomics data for the rye metabolomics regression tutorial, using unique individuals*

Description

Metabolomics data for the rye metabolomics regression tutorial, using unique individuals

Usage

```
data(freelive2)
```

Yotu *Village of capture of mosquitos for the classification tutorial*

Description

Village of capture of mosquitos for the classification tutorial

Usage

```
data(mosquito)
```

YR *Rye consumption for the rye metabolomics regression tutorial*

Description

Rye consumption for the rye metabolomics regression tutorial

Usage

```
data(freelive)
```

YR2 *Rye consumption for the rye metabolomics regression tutorial, using unique individuals*

Description

Rye consumption for the rye metabolomics regression tutorial, using unique individuals

Usage

```
data(freelive2)
```

Index