Package 'BioPred'

January 20, 2025

Type Package

Title An R Package for Biomarkers Analysis in Precision Medicine

Version 1.0.2

Date 2024-11-03

Maintainer Zihuan Liu <zihuan.liu@abbvie.com>

Description
Provides functions for training extreme gradient boosting model using propensity score A-learning and weight-
learning methods. For further details, see Liu et al. (2024) <doi:10.1093/bioinformatics/btae592>.

Encoding UTF-8

Language en

License GPL-3

Imports xgboost, pROC, ggplot2, PropCIs, survival, survminer, mgcv, onewaytests, car

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, kableExtra

VignetteBuilder knitr

NeedsCompilation no

Depends R (>= 4.0.0)

LazyData true

Author Zihuan Liu [aut, cre],
Yan Sun [aut],
Xin Huang [aut]

Repository CRAN

Date/Publication 2024-11-04 08:30:13 UTC

Contents

`cat_summary` *Summarize Categorical Variables in Subgroup*

Description

This function provides a summary of categorical variables in a dataset.

Usage

```
cat_summary(
  yvar,
  yname,
  xvars,
  xname.list,
  data,
  yvar.display = yvar,
  xvars.display = xvars
)
```

Arguments

`yvar`	Name of the variable for summary.
`yname`	A vector of ordered y values.
`xvars`	Names of the variables for grouping.

xname.list	A list (same order as xvars) of ordered x values for each xvar.
data	The dataset.
yvar.display	Display name for yvar.
xvars.display	Display name for xvars.

Value

A list containing the contingency table, frequency table, and percentage table.

Examples

```
# Load a sample dataset
data <- data.frame(
  outcome = sample(c("A", "B", "C"), 100, replace = TRUE),  # categorical outcome
  group1 = sample(c("Male", "Female"), 100, replace = TRUE),  # group variable 1
  group2 = sample(c("Young", "Old"), 100, replace = TRUE)  # group variable 2
)

# Summarize categorical outcome by two grouping variables
cat_summary(
  yvar = "outcome",
  yname = c("A", "B", "C"),  # ordered categories for outcome
  xvars = c("group1", "group2"),
  xname.list = list(c("Male", "Female"), c("Young", "Old")),
  data = data,
  yvar.display = "Outcome Category",
  xvars.display = c("Gender", "Age Group")
)
```

---

cdf_plot *CDF Plot for a biomarker*

Description

Cumulative Distribution Function (CDF) plot for a biomarker.

Usage

```
cdf_plot(xvar, data, y.int = 5, xlim = NULL, xvar.display = xvar, group = NULL)
```

Arguments

xvar	The biomarker name.
data	The dataset.
y.int	Increasement interval on the y.
xlim	cdf plot range for xvar, when NULL, c(min(x), max(x)) will be used.
xvar.display	Display name of the biomarker.
group	A separate CDF line will be plotted for each group.

Value

A ggplot object representing the CDF inverse plot.

Examples

```
# Load a sample dataset
  data <- data.frame(
  biomarker = rnorm(100, mean = 50, sd = 10),
  group = sample(c("Group A", "Group B"), 100, replace = TRUE)
)

# Basic CDF plot for a single biomarker without groups
  cdf_plot(
  xvar = "biomarker",
  data = data,
  y.int = 10,
  xlim = c(30, 70),
  xvar.display = "Biomarker Level"
)
# CDF plot for a biomarker with groups
cdf_plot(
  xvar = "biomarker",
  data = data,
  y.int = 10,
  xlim = c(30, 70),
  xvar.display = "Biomarker Level",
  group = "group"
)
```

---

cut_perf *Cutoff Performance Evaluation*

Description

This function evaluates the performance of a predictive model at a selected cutoff point.

Usage

```
cut_perf(
  yvar,
  censorvar = NULL,
  xvar,
  cutoff,
  dir,
  xvars.adj = NULL,
  data,
  type,
  yvar.display = yvar,
  xvar.display = xvar
)
```

Arguments

yvar	Response variable name.
censorvar	Censoring variable name (0-censored, 1-event).
xvar	Biomarker name.
cutoff	Selected cutoff value.
dir	Direction for desired subgroup (">", ">=", "<", "<=").
xvars.adj	Other covariates to adjust when evaluating the performance.
data	Data frame containing the variables.
type	Type of analysis: "c" for continuous, "s" for survival, and "b" for binary.
yvar.display	Display name of response variable.
xvar.display	Display name of biomarker variable.

Value

A list containing various performance metrics and optionally, plots.

Examples

```
# Load a sample dataset
data <- data.frame(
  survival_time = rexp(100, rate = 0.1),  # survival time
  status = sample(c(0, 1), 100, replace = TRUE),  # censoring status
  biomarker = rnorm(100, mean = 0, sd = 1),  # biomarker levels
  covariate1 = rnorm(100, mean = 50, sd = 10)  # an additional covariate
)
# Perform cutoff performance evaluation for continuous outcome
data$continuous_outcome <- rnorm(100, mean = 10, sd = 5)
cut_perf(
  yvar = "continuous_outcome",
  xvar = "biomarker",
  cutoff = 0.5,
  dir = ">=",
  data = data,
  type = "c",
  yvar.display = "Continuous Outcome",
  xvar.display = "Biomarker Level"
)

# Perform cutoff performance evaluation for binary outcome
data$binary_outcome <- sample(c(0, 1), 100, replace = TRUE)
cut_perf(
  yvar = "binary_outcome",
  xvar = "biomarker",
  cutoff = 0,
  dir = "<=",
  data = data,
  type = "b",
  yvar.display = "Binary Outcome",
```

```
  xvar.display = "Biomarker Level"
)
```

`eval_metric_bin` *Evaluation Metrics for XGBoostSub_bin Model*

Description

Function for evaluating XGBoostSub_bin model performance.

Usage

```
eval_metric_bin(model, X_feature, y_label, pi, trt, Loss_type = "A_learning")
```

Arguments

`model`	The trained XGBoostSub_bin model object.
`X_feature`	The input features matrix.
`y_label`	The input y matrix.
`pi`	The propensity scores vector, which should range from 0 to 1, representing the probability of assignment to treatment.
`trt`	The treatment indicator vector. Should take values of 1 or -1, where 1 represents the treatment group and -1 represents the control group.
`Loss_type`	Type of loss function to use: "A_learning" or "Weight_learning".

Details

eval_metric: Function for Evaluating XGBoostSub_bin Model Performance

This function evaluates the performance of an XGBoostSub_bin model using a A-learning or weight-learning function.

Value

Evaluation result of the XGBoostSub_bin model.

eval_metric_con *Evaluation Metrics for XGBoostSub_con Model*

Description

Function for evaluating XGBoostSub_con model performance.

Usage

```
eval_metric_con(model, X_feature, y_label, pi, trt, Loss_type = "A_learning")
```

Arguments

model	The trained XGBoostSub_con model object.
X_feature	The input features matrix.
y_label	The input y matrix.
pi	The propensity scores vector, which should range from 0 to 1, representing the probability of assignment to treatment.
trt	The treatment indicator vector. Should take values of 1 or -1, where 1 represents the treatment group and -1 represents the control group.
Loss_type	Type of loss function to use: "A_learning" or "Weight_learning".

Details

eval_metric: Function for Evaluating XGBoostSub_con Model Performance

This function evaluates the performance of an XGBoostSub_con model using a A-learning or weight-learning function.

Value

Evaluation result of the XGBoostSub_con model.

eval_metric_sur *Evaluation Metrics for XGBoostSub_sur Model*

Description

Function for evaluating XGBoostSub_sur model performance.

Usage

```
eval_metric_sur(
  model,
  X_feature,
  y_label,
  pi,
  trt,
  censor,
  Loss_type = "A_learning"
)
```

Arguments

model	The trained XGBoostSub_sur model object.
X_feature	The input features matrix.
y_label	The input y matrix.
pi	The propensity scores vector, which should range from 0 to 1, representing the probability of assignment to treatment.
trt	The treatment indicator vector. Should take values of 1 or -1, where 1 represents the treatment group and -1 represents the control group.
censor	The censor status vector. Should take values of 1 or 0, where 1 represents censoring and 0 represents an observed event.
Loss_type	Type of loss function to use: "A_learning" or "Weight_learning".

Details

eval_metric: Function for Evaluating XGBoostSub_con Model Performance

This function evaluates the performance of an XGBoostSub_con model using a A-learning or weight-learning function.

Value

Evaluation result of the XGBoostSub_sur model.

---

fixcut_bin *Fixed Cutoff Analysis for Individual Biomarker Associated with Binary Outcome Variables*

Description

This function conducts fixed cutoff analysis for individual biomarker associated with binary outcome variables.

Usage

```
fixcut_bin(
  yvar,
  xvar,
  dir,
  cutoffs,
  data,
  method = "Fisher",
  yvar.display = yvar,
  xvar.display = xvar,
  vert.x = FALSE
)
```

Arguments

yvar	Binary response variable name. 0 represents controls and 1 represents cases.
xvar	Biomarker name.
dir	Cutoff direction for the desired subgroup. Options are ">", ">=", "<", or "<=".
cutoffs	A vector of candidate cutoffs.
data	The dataset containing the variables.
method	Method for cutoff selection. Options are "Fisher", "Youden", "Conc.Prob", "Accuracy", or "Kappa". - "Fisher": Minimizes the Fisher test p-value. - "Youden": Maximizes the Youden index. - "Conc.Prob": Maximizes sensitivity * specificity. - "Accuracy": Maximizes accuracy. - "Kappa": Maximizes Kappa coefficient.
yvar.display	Display name of the response variable.
xvar.display	Display name of the predictor variable.
vert.x	Whether to display the cutoff in a 90-degree angle when plotting (saves space).

Value

A list containing statistical summaries, selected cutoff statistics, selected cutoff value, confusion matrix, and a ggplot object for visualization.

Examples

```
# Load a sample dataset
data <- data.frame(
  outcome = sample(c(0, 1), 100, replace = TRUE),
  biomarker = rnorm(100, mean = 0, sd = 1)
)

# Perform fixed cutoff analysis using the "Fisher" method for a biomarker
fixcut_bin(
  yvar = "outcome",
  xvar = "biomarker",
  dir = ">",
```

```
  cutoffs = seq(-2, 2, by = 0.5),
  data = data,
  method = "Fisher",
  yvar.display = "Binary Outcome",
  xvar.display = "Biomarker Level",
  vert.x = TRUE
)

# Perform fixed cutoff analysis using the "Youden" method
fixcut_bin(
  yvar = "outcome",
  xvar = "biomarker",
  dir = "<",
  cutoffs = seq(-2, 2, by = 0.5),
  data = data,
  method = "Youden",
  yvar.display = "Binary Outcome",
  xvar.display = "Biomarker Level",
  vert.x = FALSE
)

# Perform fixed cutoff analysis using "Accuracy" method with different direction
fixcut_bin(
  yvar = "outcome",
  xvar = "biomarker",
  dir = ">=",
  cutoffs = c(-1, 0, 1),
  data = data,
  method = "Accuracy",
  yvar.display = "Binary Outcome",
  xvar.display = "Biomarker Level",
  vert.x = TRUE
)
```

`fixcut_con` *Fixed Cutoff Analysis for Individual Biomarker Associated with Continuous Outcome*

Description

This function conducts fixed cutoff analysis for individual biomarker associated with continuous outcome variables.

Usage

```
fixcut_con(
  yvar,
  xvar,
  dir,
  cutoffs,
```

```
  data,
  method = "t.test",
  yvar.display = yvar,
  xvar.display = xvar,
  vert.x = FALSE
)
```

Arguments

yvar	Continuous response variable name.
xvar	Biomarker name.
dir	Cutoff direction for the desired subgroup. Options are ">", ">=", "<", or "<=".
cutoffs	A vector of candidate cutoffs.
data	The dataset containing the variables.
method	Method for cutoff selection. Currently only supports "t.test". - "t.test": Minimizes the t-test p-value.
yvar.display	Display name of the response variable.
xvar.display	Display name of the predictor variable.
vert.x	Whether to display the cutoff in a 90-degree angle when plotting (saves space).

Value

A list containing statistical summaries, selected cutoff statistics, selected cutoff value, group statistics, and a ggplot object for visualization.

Examples

```
# Load a sample dataset
data <- data.frame(
  outcome = rnorm(100, mean = 10, sd = 5),
  biomarker = rnorm(100, mean = 0, sd = 1)
)

# Perform fixed cutoff analysis using the "t.test" method with '>' direction
fixcut_con(
  yvar = "outcome",
  xvar = "biomarker",
  dir = ">",
  cutoffs = seq(-2, 2, by = 0.5),
  data = data,
  method = "t.test",
  yvar.display = "Continuous Outcome",
  xvar.display = "Biomarker Level",
  vert.x = TRUE
)

# Perform fixed cutoff analysis with '<=' direction
fixcut_con(
  yvar = "outcome",
```

```
  xvar = "biomarker",
  dir = "<=",
  cutoffs = c(-1, 0, 1),
  data = data,
  method = "t.test",
  yvar.display = "Continuous Outcome",
  xvar.display = "Biomarker Level",
  vert.x = FALSE
)
```

`fixcut_sur` *Fixed Cutoff Analysis for Individual Biomarker Associated with Survival Outcome*

Description

This function conducts fixed cutoff analysis for Individual Biomarker Associated with survival outcome variables.

Usage

```
fixcut_sur(
  yvar,
  censorvar,
  xvar,
  dir,
  cutoffs,
  data,
  method = "logrank",
  yvar.display = yvar,
  xvar.display = xvar,
  vert.x = FALSE
)
```

Arguments

`yvar`	Survival response variable name.
`censorvar`	Censoring variable. 0 indicates censored, 1 indicates an event.
`xvar`	Biomarker name.
`dir`	Cutoff direction for the desired subgroup. Options are ">", ">=", "<", or "<=".
`cutoffs`	A vector of candidate cutoffs.
`data`	The dataset containing the variables.
`method`	Method for cutoff selection. Currently only supports "logrank". - "logrank": Minimizes the logrank test p-value.
`yvar.display`	Display name of the response variable.
`xvar.display`	Display name of the predictor variable.
`vert.x`	Whether to display the cutoff in a 90-degree angle when plotting (saves space).

Value

A list containing statistical summaries, selected cutoff statistics, selected cutoff value, group statistics, and a ggplot object for visualization.

Examples

```
# Load a sample dataset
data <- data.frame(
  time = rexp(100, rate = 0.1),  # survival time
  status = sample(c(0, 1), 100, replace = TRUE),  # censoring status
  biomarker = rnorm(100, mean = 0, sd = 1)  # biomarker levels
)

fixcut_sur(
  yvar = "time",
  censorvar = "status",
  xvar = "biomarker",
  dir = "<=",
  cutoffs = c(-1, 0, 1),
  data = data,
  method = "logrank",
  yvar.display = "Survival Time",
  xvar.display = "Biomarker Level",
  vert.x = FALSE
)
```

---

gam_ctr_plot *GAM Contrast Plot*

Description

Computes and plots the contrasts between treatment and control group based on a GAM for exploring the relationship be-tween treatment benefit and biomarker.

Usage

```
gam_ctr_plot(
  yvar,
  censorvar = NULL,
  xvar,
  xvars.adj = NULL,
  sxvars.adj = NULL,
  trtvar = NULL,
  type,
  data,
  k,
  title = "Group Contrast",
  ybreaks = NULL,
```

```
  xbreaks = NULL,
  rugcol.var = NULL,
  link.scale = TRUE,
  prt.sum = TRUE,
  prt.chk = FALSE,
  outlier.rm = FALSE
)
```

Arguments

yvar	Response variable name.
censorvar	Censoring variable name (0-censored, 1-event). Required if type is "s" (survival).
xvar	Biomarker name.
xvars.adj	Potential confounding variables to adjust for using linear terms.
sxvars.adj	Potential confounding variables to adjust for using curves.
trtvar	Treatment variable that the contrast will build upon (treatment-control).
type	Type of response variable. Options are "c" for continuous, "s" for survival, and "b" for binary response.
data	The dataset containing the variables.
k	Upper limit on the degrees of freedom associated with an s smooth.When this k is too large, program will report error saying
title	Title of the plot.
ybreaks	Breaks on the y-axis.
xbreaks	Breaks on the x-axis.
rugcol.var	Variable name that defines the color of the rug.
link.scale	Whether to show the plot (y-axis) in the scale of the link function (linear predictor).
prt.sum	Whether to print summary or not.
prt.chk	Whether to print model diagnosis.
outlier.rm	Whether to remove outliers based on 1.5IQR.

Value

A list containing the p-value table, summarized p-value table, s-value table, summarized s-value table, and the plot.

Examples

```
# Load a sample dataset
data <- data.frame(
  response = rnorm(100),
  biomarker = rnorm(100, mean = 50, sd = 10),
  censor = sample(c(0, 1), 100, replace = TRUE),
  treatment = sample(c(0, 1), 100, replace = TRUE),
```

```
  age = rnorm(100, mean = 60, sd = 10),
  group = sample(c("Group A", "Group B"), 100, replace = TRUE)
)

# Generate a GAM contrast plot for a continuous response variable
gam_ctr_plot(
  yvar = "response",
  xvar = "biomarker",
  trtvar = "treatment",
  type = "c",
  data = data,
  xvars.adj = "age",
  k = 5,
  title = "GAM Contrast Plot for Treatment vs. Control"
)

# Generate a GAM contrast plot for survival analysis
gam_ctr_plot(
  yvar = "response",
  censorvar = "censor",
  xvar = "biomarker",
  trtvar = "treatment",
  type = "s",
  data = data,
  k = 5,
  title = "GAM Contrast Plot for Survival Data"
)

# Generate a GAM contrast plot for a binary response variable
data$binary_response <- as.numeric(data$response > 0)
gam_ctr_plot(
  yvar = "binary_response",
  xvar = "biomarker",
  trtvar = "treatment",
  type = "b",
  data = data,
  k = 5,
  title = "GAM Contrast Plot for Binary Outcome"
)
```

---

gam_plot *GAM Plot*

Description

Generates a generalized additive model (GAM) plot for exploring the relationship between a response variable and a biomarker.

Usage

```
gam_plot(
  yvar,
  censorvar = NULL,
  xvar,
  xvars.adj = NULL,
  sxvars.adj = NULL,
  type,
  data,
  k,
  pred.type = "iterms",
  link.scale = TRUE,
  title = "Trend Plot",
  ybreaks = NULL,
  xbreaks = NULL,
  rugcol.var = NULL,
  add.points = FALSE,
  prt.sum = TRUE,
  prt.chk = FALSE,
  outlier.rm = FALSE,
  newdat = NULL
)
```

Arguments

yvar	Response variable name.
censorvar	Censoring variable name for survival analysis (0-censored, 1-event).
xvar	Biomarker name.
xvars.adj	Potential confounding variables to adjust for using linear terms.
sxvars.adj	Potential confounding variables to adjust for using curve terms.
type	"c" for continuous, "s" for survival, and "b" for binary response.
data	The dataset containing the variables.
k	Upper limit on the degrees of freedom associated with an s smooth.
pred.type	"iterms" for trend of xvar, "response" for Y at the original scale.
link.scale	Whether to show the plot in the scale of the link function.
title	Title of the plot.
ybreaks	Breaks on the y-axis.
xbreaks	Breaks on the x-axis.
rugcol.var	Variable name defining the color of the rug and points.
add.points	Whether to add data points to the plot.
prt.sum	Whether to print summary or not.
prt.chk	Whether to print model diagnosis.
outlier.rm	Whether to remove outliers based on 1.5IQR.
newdat	User-supplied customized data for prediction and plotting.

Value

A list containing p-table, s-table, GAM summary, GAM check, and the plot.

Examples

```
# Load a sample dataset
data <- data.frame(
  response = rnorm(100),
  biomarker = rnorm(100, mean = 50, sd = 10),
  censor = sample(c(0, 1), 100, replace = TRUE),
  age = rnorm(100, mean = 60, sd = 10),
  group = sample(c("Group A", "Group B"), 100, replace = TRUE)
)

# Generate a GAM plot for a continuous response variable
gam_plot(
  yvar = "response",
  xvar = "biomarker",
  type = "c",
  data = data,
  xvars.adj = "age",
  sxvars.adj = NULL,
  k = 5,
  pred.type = "iterms",
  title = "GAM Plot of Biomarker and Response"
)

# Generate a GAM plot for survival analysis
gam_plot(
  yvar = "response",
  censorvar = "censor",
  xvar = "biomarker",
  type = "s",
  data = data,
  k = 5,
  title = "GAM Survival Plot for Biomarker"
)

# Generate a GAM plot for a binary response variable
data$binary_response <- as.numeric(data$response > 0)
gam_plot(
  yvar = "binary_response",
  xvar = "biomarker",
  type = "b",
  data = data,
  k = 5,
  pred.type = "response",
  title = "GAM Plot for Binary Response"
)
```

`get_subgroup_results` *Get Subgroup Results*

Description

This function predicts the treatment assignment for each patient based on a cutoff value.

Usage

```
get_subgroup_results(model, X_feature, subgroup_label = NULL, cutoff = 0.5)
```

Arguments

`model`	The trained XGBoost-based subgroup model.
`X_feature`	The data matrix containing patient features.
`subgroup_label`	(Optional) The subgroup labels. In real-world data, this information is typically unknown and only available in simulated data. If provided, the prediction accuracy will also be returned.
`cutoff`	The cutoff value for treatment assignment, defaulted to 0.5.

Value

A data frame containing each subject and assigned treatment (1 for treatment, 0 for control). If subgroup labels are provided, it also returns the prediction accuracy of the subgroup labels.

Examples

```
X_data <- matrix(rnorm(100 * 10), ncol = 10)  # 100 samples with 10 features
y_data <- rnorm(100)  # continuous outcome variable
trt <- sample(c(1, -1), 100, replace = TRUE)  # treatment indicator (1 or -1)
pi <- runif(100, min = 0.3, max = 0.7)  # propensity scores between 0 and 1

# Define XGBoost parameters
params <- list(
  max_depth = 3,
  eta = 0.1,
  subsample = 0.8,
  colsample_bytree = 0.8
)

# Train the model using A-learning loss
model_A <- XGBoostSub_con(
  X_data = X_data,
  y_data = y_data,
  trt = trt,
  pi = pi,
  Loss_type = "A_learning",
  params = params,
```

```
  nrounds = 5,
  disable_default_eval_metric = 1,
  verbose = TRUE
)
subgroup_results=get_subgroup_results(model_A, X_data, subgroup_label=NULL, cutoff = 0.5)
```

`predictive_biomarker_imp` *Plot Predictive Biomarker Importance based on XGBoost-based Subgroup Model*

Description

This function calculates and plots the importance of biomarkers in a trained XGBoostSub_con, XGBoostSub_bin or XGBoostSub_sur model.

Usage

```
predictive_biomarker_imp(model)
```

Arguments

`model`	The trained XGBoost-based model.

Value

A barplot showing the biomarker importance.

Examples

```
X_data <- matrix(rnorm(100 * 10), ncol = 10)  # 100 samples with 10 features
y_data <- rnorm(100)  # continuous outcome variable
trt <- sample(c(1, -1), 100, replace = TRUE)  # treatment indicator (1 or -1)
pi <- runif(100, min = 0.3, max = 0.7)  # propensity scores between 0 and 1

# Define XGBoost parameters
params <- list(
  max_depth = 3,
  eta = 0.1,
  subsample = 0.8,
  colsample_bytree = 0.8
)

# Train the model using A-learning loss
model_A <- XGBoostSub_con(
  X_data = X_data,
  y_data = y_data,
  trt = trt,
  pi = pi,
```

```
  Loss_type = "A_learning",
  params = params,
  nrounds = 5,
  disable_default_eval_metric = 1,
  verbose = TRUE
)
biomarker_imp=predictive_biomarker_imp(model_A)
```

---

roc_bin *AUC ROC Table for Biomarkers Associated with Binary Outcomes*

Description

Computes the area under the receiver operating characteristic (ROC) curve for Biomarkers Associated with Binary Outcomes, and returns the results as a table.

Usage

```
roc_bin(yvar, xvars, dirs, data, yvar.display = yvar, xvars.display = xvars)
```

Arguments

yvar	Binary response variable name, where 0 represents controls and 1 represents cases.
xvars	A vector of biomarker names.
dirs	A vector of directions for the biomarkers. Options are "auto", ">", or "<". - "auto" (default): automatically determines in which group the median is higher and takes the direction accordingly. - ">": indicates that the biomarkers for the control group are higher than those for the case group (controls > t >= cases). - "<": indicates that the biomarkers for the control group are lower or equal to those for the case group (controls < t <= cases).
data	The dataset containing the variables.
yvar.display	Display name for the binary response variable.
xvars.display	Display names for the biomarkers.

Value

A table containing the AUC values for each biomarker.

Examples

```
# Load a sample dataset
data <- data.frame(
  outcome = sample(c(0, 1), 100, replace = TRUE),
  biomarker1 = rnorm(100, mean = 0, sd = 1),
  biomarker2 = rnorm(100, mean = 5, sd = 2)
)
```

```
# Compute AUC for a single biomarker with auto direction
roc_bin(
  yvar = "outcome",
  xvars = "biomarker1",
  dirs = "auto",
  data = data,
  yvar.display = "Binary Outcome",
  xvars.display = "Biomarker 1"
)

# Compute AUC for multiple biomarkers with specified directions
roc_bin(
  yvar = "outcome",
  xvars = c("biomarker1", "biomarker2"),
  dirs = c("auto", "<"),
  data = data,
  yvar.display = "Binary Outcome",
  xvars.display = c("Biomarker 1", "Biomarker 2")
)
```

`roc_bin_plot` *ROC Plot Biomarkers Associated with Binary Outcomes*

Description

Generates ROC plots for different biomarkers associated with binary outcomes.

Usage

```
roc_bin_plot(
  yvar,
  xvars,
  dirs,
  data,
  yvar.display = yvar,
  xvars.display = xvars
)
```

Arguments

`yvar`	Binary response variable name, where 0 represents controls and 1 represents cases.
`xvars`	A vector of biomarker names.
`dirs`	A vector of directions for the biomarkers. Options are "auto", ">", or "<". - "auto" (default): automatically determines in which group the median is higher and takes the direction accordingly. - ">" indicates that the biomarkers for the control group are higher than those for the case group (controls > t >= cases).

	- "<" indicates that the biomarkers for the control group are lower or equal to those for the case group (controls < t <= cases).
data	The dataset containing the variables.
yvar.display	Display name for the binary response variable.
xvars.display	Display names for the biomarkers.

Value

ROC plots for different biomarkers associated with binary outcomes.

Examples

```
# Load a sample dataset
data <- data.frame(
  outcome = sample(c(0, 1), 100, replace = TRUE),
  biomarker1 = rnorm(100, mean = 0, sd = 1),
  biomarker2 = rnorm(100, mean = 5, sd = 2)
)

# Generate ROC plot for a single biomarker with auto direction
roc_bin_plot(
  yvar = "outcome",
  xvars = "biomarker1",
  dirs = "auto",
  data = data,
  yvar.display = "Binary Outcome",
  xvars.display = "Biomarker 1"
)

# Generate ROC plots for multiple biomarkers with specified directions
roc_bin_plot(
  yvar = "outcome",
  xvars = c("biomarker1", "biomarker2"),
  dirs = c("auto", "<"),
  data = data,
  yvar.display = "Binary Outcome",
  xvars.display = c("Biomarker 1", "Biomarker 2")
)
```

---

scat_cont_plot *Scatter Plot for a Biomarker Associated with Continuous Outcome*

Description

Generates a scatter plot for exploring the relationship between a continuous response variable and a biomarker variable.

Usage

```
scat_cont_plot(
  yvar,
  xvar,
  data,
  ybreaks = NULL,
  xbreaks = NULL,
  yvar.display = yvar,
  xvar.display = xvar
)
```

Arguments

yvar	Continuous response variable name.
xvar	biomarker name.
data	The dataset containing the variables.
ybreaks	Breaks on the y-axis.
xbreaks	Breaks on the x-axis.
yvar.display	Display name for the response variable.
xvar.display	Display name for the biomarker variable.

Value

A list containing correlation coefficients, scatter plot, slope, and intercept.

Examples

```
data <- data.frame(
  outcome = rnorm(100, mean = 10, sd = 2),
  biomarker = rnorm(100, mean = 0, sd = 1)
)

# Generate a scatter plot with default axis breaks
scat_cont_plot(
  yvar = "outcome",
  xvar = "biomarker",
  data = data,
  yvar.display = "Continuous Outcome",
  xvar.display = "Biomarker Level"
)

# Generate a scatter plot with specified axis breaks
scat_cont_plot(
  yvar = "outcome",
  xvar = "biomarker",
  data = data,
  ybreaks = seq(5, 15, by = 1),
  xbreaks = seq(-2, 2, by = 0.5),
  yvar.display = "Continuous Outcome",
```

```
  xvar.display = "Biomarker Level"
)
```

---

subgrp_perf *Subgroup Performance Evaluation for Prognostic Cases*

Description

This function evaluates subgroup performance based on different types of response variables.

Usage

```
subgrp_perf(
  yvar,
  censorvar = NULL,
  grpvar,
  grpname,
  xvars.adj = NULL,
  data,
  type,
  yvar.display = yvar,
  grpvar.display = grpvar
)
```

Arguments

yvar	The response variable name.
censorvar	(Optional) The censoring variable name (0-censored, 1-event).
grpvar	The subgroup variable name.
grpname	A vector of ordered subgroup names (values in the column of grpvar).
xvars.adj	(Optional) Other covariates to adjust when evaluating the performance.
data	The dataset containing the variables.
type	The type of response variable: "c" for continuous, "s" for survival, and "b" for binary.
yvar.display	Display name of the response variable.
grpvar.display	Display name of the group variable.

Value

A list containing subgroup performance results including logrank p-value, median and mean survival, Cox model p-value, ANOVA p-value, and more based on the specified response variable type.

Examples

```
# Load a sample dataset
data <- data.frame(
  survival_time = rexp(100, rate = 0.1),  # survival time
  status = sample(c(0, 1), 100, replace = TRUE),  # censoring status
  group = sample(c("Low", "Medium", "High"), 100, replace = TRUE),  # subgroup variable
  covariate = rnorm(100, mean = 50, sd = 10)  # an additional covariate
)

# Perform subgroup performance evaluation for survival analysis
subgrp_perf(
  yvar = "survival_time",
  censorvar = "status",
  grpvar = "group",
  grpname = c("Low", "Medium", "High"),
  data = data,
  type = "s",
  yvar.display = "Survival Time",
  grpvar.display = "Risk Group"
)

# Perform subgroup performance evaluation for continuous outcome
data$continuous_outcome <- rnorm(100, mean = 10, sd = 5)
subgrp_perf(
  yvar = "continuous_outcome",
  grpvar = "group",
  grpname = c("Low", "Medium", "High"),
  data = data,
  type = "c",
  yvar.display = "Continuous Outcome",
  grpvar.display = "Risk Group"
)

# Perform subgroup performance evaluation for binary outcome
data$binary_outcome <- sample(c(0, 1), 100, replace = TRUE)
subgrp_perf(
  yvar = "binary_outcome",
  grpvar = "group",
  grpname = c("Low", "Medium", "High"),
  data = data,
  type = "b",
  yvar.display = "Binary Outcome",
  grpvar.display = "Risk Group"
)
```

`subgrp_perf_pred` *Subgroup Performance Evaluation for Predictive Cases*

Description

This function evaluates the performance of subgroups based on different types of response variables in predictive cases.

Usage

```
subgrp_perf_pred(
  yvar,
  censorvar = NULL,
  grpvar,
  grpname,
  trtvar,
  trtname,
  xvars.adj = NULL,
  data,
  type,
  yvar.display = yvar,
  grpvar.display = grpvar,
  trtvar.display = trtvar
)
```

Arguments

yvar	Response variable name.
censorvar	Censoring variable name (0-censored, 1-event).
grpvar	Subgroup variable name.
grpname	A vector of ordered subgroup names (values in the column of grpvar).
trtvar	Treatment variable name.
trtname	A vector of ordered treatment names (values in the column of trtvar).
xvars.adj	Other covariates to adjust when evaluating the performance.
data	The dataset.
type	"c" for continuous; "s" for "survival", and "b" for binary.
yvar.display	Display name of the response variable.
grpvar.display	Display name of the group variable.
trtvar.display	Display name of the treatment variable.

Value

A list containing the comparison results, group results, and possibly a plot.

Examples

```
# Load a sample dataset
data <- data.frame(
  response = rnorm(100, mean = 10, sd = 5),  # continuous response
  survival_time = rexp(100, rate = 0.1),  # survival time
```

```
  status = sample(c(0, 1), 100, replace = TRUE),  # censoring status
  group = sample(c("Low", "Medium", "High"), 100, replace = TRUE),  # subgroup variable
  treatment = sample(c("A", "B"), 100, replace = TRUE)  # treatment variable
)

# Subgroup performance evaluation for predictive cases - survival analysis
subgrp_perf_pred(
  yvar = "survival_time",
  censorvar = "status",
  grpvar = "group",
  grpname = c("Low", "Medium", "High"),
  trtvar = "treatment",
  trtname = c("A", "B"),
  data = data,
  type = "s",
  yvar.display = "Survival Time",
  grpvar.display = "Risk Group",
  trtvar.display = "Treatment"
)

# Subgroup performance evaluation for predictive cases - continuous outcome
subgrp_perf_pred(
  yvar = "response",
  grpvar = "group",
  grpname = c("Low", "Medium", "High"),
  trtvar = "treatment",
  trtname = c("A", "B"),
  data = data,
  type = "c",
  yvar.display = "Response",
  grpvar.display = "Risk Group",
  trtvar.display = "Treatment"
)

# Subgroup performance evaluation for predictive cases - binary outcome
data$binary_response <- sample(c(0, 1), 100, replace = TRUE)
subgrp_perf_pred(
  yvar = "binary_response",
  grpvar = "group",
  grpname = c("Low", "Medium", "High"),
  trtvar = "treatment",
  trtname = c("A", "B"),
  data = data,
  type = "b",
  yvar.display = "Binary Response",
  grpvar.display = "Risk Group",
  trtvar.display = "Treatment"
)
```

`tutorial_data` *Tutorial Data*

Description

A dataset containing sample data for demonstrating the functionalities of the BioPred package.

Usage

```
data(tutorial_data)
```

Format

A data frame with the following columns:

x1 Numeric. A biomarker variable.

x2 Numeric. A biomarker variable.

x3 Numeric. A biomarker variable.

x4 Numeric. A biomarker variable.

x5 Numeric. A biomarker variable.

x6 Numeric. A biomarker variable.

x7 Numeric. A biomarker variable.

x8 Numeric. A biomarker variable.

x9 Numeric. A biomarker variable.

x10 Numeric. A biomarker variable.

y.con Numeric. A continuous outcome variable.

y.bin Binary. A binary outcome variable, where 0 represents one class and 1 represents another class.

y.time Numeric. The time in months, used for survival analysis.

y.event Binary. Event indicator variable, where 0 indicates censoring and 1 indicates the event of interest occurred.

subgroup_label Binary. Ground truth of subgroup label. In real-world scenarios, this information is typically unavailable.

treatment Binary. Treatment indicator variable, where 0 represents control and 1 represents treatment.

treatment_categorical Factor. A categorical version of the treatment variable, with levels "Placebo" and "Treatment".

risk_category Factor.

Details

This dataset is used to illustrate various functions within the BioPred package, including predictive modeling and subgroup analysis. The columns represent different types of data typically encountered in clinical studies.

Examples

```
data(tutorial_data)
head(tutorial_data)
```

XGBoostSub_bin *XGBoost Model with Modified Loss Function for Subgroup Identification with Binary Outcomes*

Description

Function for training XGBoost model with customized loss function for binary outcomes

Usage

```
XGBoostSub_bin(
  X_data,
  y_data,
  trt,
  pi,
  Loss_type = "A_learning",
  params = list(),
  nrounds = 50,
  disable_default_eval_metric = 1,
  verbose = TRUE
)
```

Arguments

X_data	The input features matrix.
y_data	The input y matrix.
trt	The treatment indicator vector. Should take values of 1 or -1, where 1 represents the treatment group and -1 represents the control group.
pi	The propensity scores vector, which should range from 0 to 1, representing the probability of assignment to treatment.
Loss_type	Type of loss function to use: "A_learning" or "Weight_learning".
params	A list of additional parameters for the xgb.train function.
nrounds	Number of boosting rounds. Default is 50.
disable_default_eval_metric	If 1, default evaluation metric will be disabled.
verbose	Logical. If TRUE, training progress will be printed; if FALSE, no progress will be printed.

Details

XGBoostSub_bin: Function for Training XGBoost Model with Customized Loss Function for binary outcomes

This function trains an XGBoost model using a customized loss function based on the A-learning and weight-learning.

This function requires the 'xgboost' library. Make sure to install and load the 'xgboost' library before using this function.

After running this function, the returned model can be used like a regular xgboost model.

Value

Trained XGBoostSub_bin model.

Examples

```
X_data <- matrix(rnorm(100 * 10), ncol = 10)  # 100 samples with 10 features
y_data <- rbinom(100, 1, 0.5)  # binary outcomes (0 or 1)
trt <- sample(c(1, -1), 100, replace = TRUE)  # treatment indicator (1 or -1)
pi <- runif(100, min = 0.3, max = 0.7)  # propensity scores between 0 and 1

# Define XGBoost parameters
params <- list(
  max_depth = 3,
  eta = 0.1,
  subsample = 0.8,
  colsample_bytree = 0.8
)

# Train the model using A-learning loss
model_A <- XGBoostSub_bin(
  X_data = X_data,
  y_data = y_data,
  trt = trt,
  pi = pi,
  Loss_type = "A_learning",
  params = params,
  nrounds = 5,
  disable_default_eval_metric = 1,
  verbose = TRUE
)

# Train the model using Weight-learning loss
model_W <- XGBoostSub_bin(
  X_data = X_data,
  y_data = y_data,
  trt = trt,
  pi = pi,
  Loss_type = "Weight_learning",
  params = params,
  nrounds = 5,
  disable_default_eval_metric = 1,
  verbose = TRUE
)
```

XGBoostSub_con *XGBoost Model with Modified Loss Function for Subgroup Identification with Continuous Outcomes*

Description

Function for training XGBoost model with customized loss function for continuous outcomes

Usage

```
XGBoostSub_con(
  X_data,
  y_data,
  trt,
  pi,
  Loss_type = "A_learning",
  params = list(),
  nrounds = 50,
  disable_default_eval_metric = 1,
  verbose = TRUE
)
```

Arguments

X_data	The input features matrix.
y_data	The input y matrix.
trt	The treatment indicator vector. Should take values of 1 or -1, where 1 represents the treatment group and -1 represents the control group.
pi	The propensity scores vector, which should range from 0 to 1, representing the probability of assignment to treatment.
Loss_type	Type of loss function to use: "A_learning" or "Weight_learning".
params	A list of additional parameters for the xgb.train function.
nrounds	Number of boosting rounds. Default is 50.
disable_default_eval_metric	If 1, default evaluation metric will be disabled.
verbose	Logical. If TRUE, training progress will be printed; if FALSE, no progress will be printed.

Details

XGBoostSub_con: Function for Training XGBoost Model with Customized Loss Function for continuous outcomes

This function trains an XGBoost model using a customized loss function based on the A-learning and weight-learning.

This function requires the 'xgboost' library. Make sure to install and load the 'xgboost' library before using this function.

After running this function, the returned model can be used like a regular xgboost model.

Value

Trained XGBoostSub_con model.

Examples

```
X_data <- matrix(rnorm(100 * 10), ncol = 10)  # 100 samples with 10 features
y_data <- rnorm(100)  # continuous outcome variable
trt <- sample(c(1, -1), 100, replace = TRUE)  # treatment indicator (1 or -1)
pi <- runif(100, min = 0.3, max = 0.7)  # propensity scores between 0 and 1

# Define XGBoost parameters
params <- list(
  max_depth = 3,
  eta = 0.1,
  subsample = 0.8,
  colsample_bytree = 0.8
)

# Train the model using A-learning loss
model_A <- XGBoostSub_con(
  X_data = X_data,
  y_data = y_data,
  trt = trt,
  pi = pi,
  Loss_type = "A_learning",
  params = params,
  nrounds = 5,
  disable_default_eval_metric = 1,
  verbose = TRUE
)

# Train the model using Weight-learning loss
model_W <- XGBoostSub_con(
  X_data = X_data,
  y_data = y_data,
  trt = trt,
  pi = pi,
  Loss_type = "Weight_learning",
  params = params,
  nrounds = 5,
  disable_default_eval_metric = 1,
  verbose = TRUE
)
```

XGBoostSub_sur *XGBoost Model with Modified Loss Function for Subgroup Identification with Survival Outcomes*

Description

Function for training XGBoost model with customized loss function for survival outcomes

Usage

```
XGBoostSub_sur(
  X_data,
  y_data,
  trt,
  pi,
  censor,
  Loss_type = "Weight_learning",
  params = list(),
  nrounds = 50,
  disable_default_eval_metric = 1,
  verbose = TRUE
)
```

Arguments

X_data	The input features matrix.
y_data	The input y matrix.
trt	The treatment indicator vector. Should take values of 1 or -1, where 1 represents the treatment group and -1 represents the control group.
pi	The propensity scores vector, which should range from 0 to 1, representing the probability of assignment to treatment.
censor	The censor status vector. Should take values of 1 or 0, where 1 represents censoring and 0 represents an observed event.
Loss_type	Type of loss function to use: "A_learning" or "Weight_learning".
params	A list of additional parameters for the xgb.train function.
nrounds	Number of boosting rounds. Default is 50.
disable_default_eval_metric	If 1, default evaluation metric will be disabled.
verbose	Logical. If TRUE, training progress will be printed; if FALSE, no progress will be printed.

Details

XGBoostSub_sur: Function for Training XGBoost Model with Customized Loss Function for survival outcomes

This function trains an XGBoost model using a customized loss function based on the A-learning and weight-learning.

This function requires the 'xgboost' library. Make sure to install and load the 'xgboost' library before using this function.

Value

Trained XGBoostSub_sur model.

Examples

```
X_data <- matrix(rnorm(100 * 10), ncol = 10)  # 100 samples with 10 features
y_data <- rexp(100, rate = 0.1)  # survival times, simulated as exponential
trt <- sample(c(1, -1), 100, replace = TRUE)  # treatment indicator (1 or -1)
pi <- runif(100, min = 0.3, max = 0.7)  # propensity scores between 0 and 1
censor <- rbinom(100, 1, 0.7)  # censoring indicator (1 = censored, 0 = observed)

# Define XGBoost parameters
params <- list(
  max_depth = 3,
  eta = 0.1,
  subsample = 0.8,
  colsample_bytree = 0.8
)

# Train the model using A-learning loss
model_A <- XGBoostSub_sur(
  X_data = X_data,
  y_data = y_data,
  trt = trt,
  pi = pi,
  censor = censor,
  Loss_type = "A_learning",
  params = params,
  nrounds = 5,
  disable_default_eval_metric = 1,
  verbose = TRUE
)

# Train the model using Weight-learning loss
model_W <- XGBoostSub_sur(
  X_data = X_data,
  y_data = y_data,
  trt = trt,
  pi = pi,
  censor = censor,
  Loss_type = "Weight_learning",
  params = params,
```

```
  nrounds = 5,
  disable_default_eval_metric = 1,
  verbose = TRUE
)
```

Index