Package 'BuyseTest'

March 20, 2023

Type Package

Title Generalized Pairwise Comparisons

Version 2.4.0

Date 2023-03-20

Description Implementation of the Generalized Pairwise Comparisons (GPC)
as defined in Buyse (2010) <doi:10.1002/sim.3923> for complete observations,
and extended in Peron (2018) <doi:10.1177/0962280216658320> to deal with right-censoring.
GPC compare two groups of observations (intervention vs. control group)
regarding several prioritized endpoints to estimate the probability that a random observation drawn from
one group performs better than a random observation drawn from the other group (Mann-Whitney parameter).
The net benefit and win ratio statistics,
i.e. the difference and ratio between the probabilities relative to the intervention and control groups,
can then also be estimated. Confidence intervals and p-values are obtained based on asymptotic results (Ozenne 2021 <doi:10.1177/09622802211037067>),
non-parametric bootstrap, or permutations.
The software enables the use of thresholds of minimal importance difference,
stratification, non-prioritized endpoints (O Brien test), and can handle right-censoring and competing-risks.

License GPL-3

Encoding UTF-8

URL https://github.com/bozenne/BuyseTest

BugReports https://github.com/bozenne/BuyseTest/issues

Depends R (>= 2.10), Rcpp

Imports data.table, doParallel, foreach, ggplot2, methods, lava,
parallel, prodlim, riskRegression, rlang, stats, stats4, utils

Suggests cvAUC, mvtnorm, pbapply, pROC, R.rsp, survival, testthat

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder R.rsp

NeedsCompilation yes

RoxygenNote 7.2.1

Collate '0-onLoad.R' '1-setGeneric.R' 'BuyseMultComp.R' 'BuyseTTEM.R'
'BuyseTest-Peron.R' 'BuyseTest-check.R' 'BuyseTest-inference.R'
'BuyseTest-initialization.R' 'BuyseTest-package.R'
'BuyseTest-print.R' 'BuyseTest.R' 'BuyseTest.options.R'
'PairScore.R' 'RcppExports.R' 'S4-BuysePower.R'
'S4-BuysePower-summary.R' 'S4-BuysePower-show.R'
'S4-BuyseTest.R' 'S4-BuyseTest-coef.R' 'S4-BuyseTest-confint.R'
'S4-BuyseTest-get.R' 'S4-BuyseTest-sensitivity.R'
'S4-BuyseTest-summary.R' 'S4-BuyseTest-show.R'
'S4-BuyseTest.options.R' 'as.data.table.performance.R' 'auc.R'
'autoplot.sensitivity.R' 'brier.R' 'constStrata.R'
'discreteRoot.R' 'iid.prodlim.R' 'normexp.R' 'performance.R'
'performanceResample.R' 'powerBuyseTest.R' 'predict.logit.R'
'rbind.performanceResample.R' 'simBuyseTest.R'
'simCompetingRisks.R' 'summary.performance.R' 'valid.R'

Author Brice Ozenne [aut, cre] (<https://orcid.org/0000-0001-9694-2956>),
Julien Peron [ctb],
Eva Cantagallo [aut]

Maintainer Brice Ozenne <brice.mh.ozenne@gmail.com>

Repository CRAN

Date/Publication 2023-03-20 22:30:02 UTC

R topics documented:

`BuyseTest-package` *BuyseTest package: Generalized Pairwise Comparisons*

Description

Implementation of the Generalized Pairwise Comparisons. `BuyseTest` is the main function of the package. See the vignette of an overview of the functionalities of the package. Run `citation("BuyseTest")` in R for how to cite this package in scientific publications. See the section reference below for examples of application in clinical studies.

Details

The Generalized Pairwise Comparisons form all possible pairs of observations, one observation being taken from the intervention group and the other is taken from the control group, and compare the difference in endpoints ($Y - X$) to the threshold of clinical relevance (τ).

For a single endpoint, if the difference is greater or equal than the threshold of clinical relevance ($Y \geq X + \tau$), the pair is classified as favorable (i.e. win). If the difference is lower or equal than minus the threshold of clinical relevance ($X \geq Y + \tau$), the pair is classified as unfavorable (i.e. loss). Otherwise the pair is classified as neutral. In presence of censoring, it might not be possible to compare the difference to the threshold. In such cases the pair is classified as uninformative.

Simultaneously analysis of several endpoints is performed by prioritizing the endpoints, assigning the highest priority to the endpoint considered the most clinically relevant. The endpoint with highest priority is analyzed first, and neutral and uninformative pair are analyzed regarding endpoint of lower priority.

References

Method papers on the GPC procedure and its extensions: On the GPC procedure: Marc Buyse (2010). **Generalized pairwise comparisons of prioritized endpoints in the two-sample problem**. *Statistics in Medicine* 29:3245-3257
On the win ratio: D. Wang, S. Pocock (2016). **A win ratio approach to comparing continuous non-normal outcomes in clinical trials**. *Pharmaceutical Statistics* 15:238-245
On the Peron's scoring rule: J. Peron, M. Buyse, B. Ozenne, L. Roche and P. Roy (2018). **An extension of generalized pairwise comparisons for prioritized outcomes in the presence of censoring**. *Statistical Methods in Medical Research* 27: 1230-1239.
On the Gehan's scoring rule: Gehan EA (1965). **A generalized two-sample Wilcoxon test for doubly censored data**. *Biometrika* 52(3):650-653
On inference in GPC using the U-statistic theory: Ozenne B, Budtz-Jorgensen E, Peron J (2021). **The asymptotic distribution of the Net Benefit estimator in presence of right-censoring**. *Statistical Methods in Medical Research* 2021 doi:10.1177/09622802211037067
On how to handle right-censoring: J. Peron, M. Idlhaj, D. Maucort-Boulch, et al. (2021) **Correcting the bias of the net benefit estimator due to right-censored observations**. *Biometrical Journal* 63: 893–906.

Examples of application in clinical studies:
J. Peron, P. Roy, K. Ding, W. R. Parulekar, L. Roche, M. Buyse (2015). **Assessing the benefit-risk of new treatments using generalized pairwise comparisons: the case of erlotinib in pancreatic cancer**. *British journal of cancer* 112:(6)971-976.
J. Peron, P. Roy, T. Conroy, F. Desseigne, M. Ychou, S. Gourgou-Bourgade, T. Stanbury, L. Roche, B. Ozenne, M. Buyse (2016). **An assessment of the benefit-risk balance of FOLFORINOX in metastatic pancreatic adenocarcinoma**. *Oncotarget* 7:82953-60, 2016.

Comparison between the net benefit and alternative measures of treatment effect:
J. Peron, P. Roy, B. Ozenne, L. Roche, M. Buyse (2016). **The net chance of a longer survival as a patient-oriented measure of benefit in randomized clinical trials**. *JAMA Oncology* 2:901-5.
E. D. Saad , J. R. Zalcberg, J. Peron, E. Coart, T. Burzykowski, M. Buyse (2018). **Understanding and communicating measures of treatment effect on survival: can we do better?**. *J Natl Cancer Inst*.

.colCenter_cpp *Substract a vector of values in each column*

Description

Fast computation of sweep(X, FUN = "-", STATS = center, MARGIN = 1)

Usage

```
.colCenter_cpp(X, center)
```

Arguments

X	A matrix.
center	A vector with length the number of rows of X .

Value

A matrix of same size as x.

.colCumSum_cpp *Column-wise cumulative sum*

Description

Fast computation of apply(x,2,cumsum)

Usage

```
.colCumSum_cpp(X)
```

Arguments

X	A matrix.

Value

A matrix of same size as x.

.colMultiply_cpp *Multiply by a vector of values in each column*

Description

Fast computation of sweep(X, FUN = "*", STATS = scale, MARGIN = 1)

Usage

```
.colMultiply_cpp(X, scale)
```

Arguments

X	A matrix.
scale	A vector with length the number of rows of X .

Value

A matrix of same size as x.

.colScale_cpp *Divide by a vector of values in each column*

Description

Fast computation of sweep(X, FUN = "/", STATS = scale, MARGIN = 1)

Usage

```
.colScale_cpp(X, scale)
```

Arguments

X	A matrix.
scale	A vector with length the number of rows of X .

Value

A matrix of same size as x.

.rowCenter_cpp *Substract a vector of values in each row*

Description

Fast computation of sweep(X, FUN = "-", STATS = center, MARGIN = 2)

Usage

```
.rowCenter_cpp(X, center)
```

Arguments

X	A matrix.
center	A vector with length the number of columns of X.

Value

A matrix of same size as x.

.rowCumProd_cpp *Apply cumprod in each row*

Description

Fast computation of t(apply(x,1,cumprod))

Usage

```
.rowCumProd_cpp(X)
```

Arguments

X	A matrix.

Value

A matrix of same size as x.

.rowCumSum_cpp *Row-wise cumulative sum*

Description

Fast computation of apply(x,1,cumsum)

Usage

```
.rowCumSum_cpp(X)
```

Arguments

X	A matrix.

Value

A matrix of same size as x.

.rowMultiply_cpp *Multiply by a vector of values in each row*

Description

Fast computation of sweep(X, FUN = "*", STATS = center, MARGIN = 2)

Usage

```
.rowMultiply_cpp(X, scale)
```

Arguments

X	A matrix.
scale	A vector with length the number of columns of X.

Value

A matrix of same size as x.

.rowScale_cpp *Dividy by a vector of values in each row*

Description

Fast computation of sweep(X, FUN = "/", STATS = center, MARGIN = 2)

Usage

```
.rowScale_cpp(X, scale)
```

Arguments

X	A matrix.
scale	A vector with length the number of columns of X.

Value

A matrix of same size as x.

as.data.table.performance *Convert Performance Objet to data.table*

Description

Extract the AUC/brier score values or the prediction into a data.table format.

Usage

```
## S3 method for class 'performance'
as.data.table(
  x,
  keep.rownames = FALSE,
  type = "performance",
  format = NULL,
  ...
)
```

Arguments

x	object of class "performance".
keep.rownames	Not used. For compatibility with the generic method.
type	[character] either "metric" to extract AUC/brier score or "prediction" to extract predictions.
format	[character] should the result be outcome in the long format ("long") or in the wide format ("wide"). Note relevant when using type="metric".
...	Not used. For compatibility with the generic method.

---

auc *Estimation of the Area Under the ROC Curve (EXPERIMENTAL)*

Description

Estimation of the Area Under the ROC curve, possibly after cross validation, to assess the discriminant ability of a biomarker regarding a disease status.

Usage

```
auc(
  labels,
  predictions,
  fold = NULL,
  observation = NULL,
  direction = ">",
  add.halfNeutral = TRUE,
  null = 0.5,
  conf.level = 0.95,
  transformation = TRUE,
  order.Hprojection = 2,
  pooling = "mean"
)
```

Arguments

labels	[integer/character vector] the disease status (should only take two different values).
predictions	[numeric vector] A vector with the same length as labels containing the biomarker values.
fold	[character/integer vector] If using cross validation, the index of the fold. Should have the same length as labels.
observation	[integer vector] If using cross validation, the index of the corresponding observation in the original dataset. Necessary to compute the standard error when using cross validation.

direction [character] ">" lead to estimate P[Y>X], "<" to estimate P[Y<X], and "auto" to estimate max(P[Y>X],P[Y<X]).

add.halfNeutral [logical] should half of the neutral score be added to the favorable and unfavorable scores? Useful to match the usual definition of the AUC in presence of ties.

null [numeric, 0-1] the value against which the AUC should be compared when computing the p-value.

conf.level [numeric, 0-1] the confidence level of the confidence intervals.

transformation [logical] should a log-log transformation be used when computing the confidence intervals and the p-value.

order.Hprojection [1,2] the order of the H-projection used to linear the statistic when computing the standard error. 2 is involves more calculations but is more accurate in small samples. Only active when the fold argument is NULL.

pooling [character] method used to compute the global AUC from the fold-specific AUC: either an empirical average "mean" or a weighted average with weights proportional to the number of pairs of observations in each fold "pairs".

Details

The iid decomposition of the AUC is based on a first order decomposition. So its squared value will not exactly match the square of the standard error estimated with a second order H-projection.

Value

A *data.frame* containing for each fold the AUC value with its standard error (when computed). The last line of the data.frame contains the global AUC value with its standard error.

References

Erin LeDell, Maya Petersen, and Mark van der Laan (2015). **Computationally efficient confidence intervals for cross-validated area under the ROC curve estimates**. *Electron J Stat.* 9(1):1583–1607.

Examples

```
library(data.table)

n <- 200
set.seed(10)
X <- rnorm(n)
dt <- data.table(Y = as.factor(rbinom(n, size = 1, prob = 1/(1+exp(1/2-X)))),
                 X = X,
                 fold = unlist(lapply(1:10,function(iL){rep(iL,n/10)})))

## compute auc
auc(labels = dt$Y, predictions = dt$X, direction = ">")
```

```
## compute auc after 10-fold cross-validation
auc(labels = dt$Y, prediction = dt$X, fold = dt$fold, observation = 1:NROW(dt))
```

`autoplot.sensitivity` *Graphical Display for Sensitivity Analysis*

Description

Display the statistic of interest across various threshold values, possibly with confidence intervals. Currently only works when varying thresholds relative to one or two variables.

Usage

```
## S3 method for class 'sensitivity'
autoplot(
  object,
  plot = TRUE,
  col = NULL,
  ci = TRUE,
  band = TRUE,
  label = "Threshold for",
  position = NULL,
  size.line = 1,
  size.point = 1.75,
  size.ci = 0.5,
  alpha = 0.1,
  ...
)
```

Arguments

`object`	output of the sensitivity method
`plot`	[logical] should the graph be displayed in a graphical window
`col`	[character vector] color used to identify the thresholds relative to a second variable.
`ci`	[logical] should the confidence intervals be displayed?
`band`	[logical] should the simulatenous confidence intervals be displayed?
`label`	[character] text used before the name of the variables in the legend.
`position`	relative position of the error bars for a given x value. Can for instance be `position_dodge(width = 5)`.
`size.line`	[numeric] width of the line connecting the point estimates.
`size.point`	[numeric] size of the point representing the point estimates.
`size.ci`	[numeric] width of the lines representing the confidence intervals.

alpha	[numeric] transparency for the area representing the simultaneous confidence intervals.
...	not used. For compatibility with the generic method.

`boot2pvalue` *Compute the p.value from the distribution under H1*

Description

Compute the p.value associated with the estimated statistic using a bootstrap sample of its distribution under H1.

Usage

```
boot2pvalue(
  x,
  null,
  estimate = NULL,
  alternative = "two.sided",
  FUN.ci = quantileCI,
  checkSign = TRUE,
  tol = .Machine$double.eps^0.5
)
```

Arguments

x	[numeric vector] a vector of bootstrap estimates of the statistic.
null	[numeric] value of the statistic under the null hypothesis.
estimate	[numeric] the estimated statistic.
alternative	[character] a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
FUN.ci	[function] the function used to compute the confidence interval. Must take `x`, `alternative`, `conf.level` and `sign.estimate` as arguments and only return the relevant limit (either upper or lower) of the confidence interval.
checkSign	[logical] should a warning be output if the sign of the estimate differs from the sign of the mean bootstrap value?
tol	[numeric] the absolute convergence tolerance.

Details

For test statistic close to 0, this function returns 1.

For positive test statistic, this function search the quantile alpha such that:

- quantile(x, probs = alpha)=0 when the argument alternative is set to "greater".
- quantile(x, probs = 0.5*alpha)=0 when the argument alternative is set to "two.sided".

If the argument alternative is set to "less", it returns 1.

For negative test statistic, this function search the quantile alpha such that:

- quantile(x, probs = 1-alpha=0 when the argument alternative is set to "less".
- quantile(x, probs = 1-0.5*alpha=0 when the argument alternative is set to "two.sided".

If the argument alternative is set to "greater", it returns 1.

Examples

```
set.seed(10)

#### no effect ####
x <- rnorm(1e3)
boot2pvalue(x, null = 0, estimate = mean(x), alternative = "two.sided")
## expected value of 1
boot2pvalue(x, null = 0, estimate = mean(x), alternative = "greater")
## expected value of 0.5
boot2pvalue(x, null = 0, estimate = mean(x), alternative = "less")
## expected value of 0.5

#### positive effect ####
x <- rnorm(1e3, mean = 1)
boot2pvalue(x, null = 0, estimate = 1, alternative = "two.sided")
## expected value of 0.32 = 2*pnorm(q = 0, mean = -1) = 2*mean(x<=0)
boot2pvalue(x, null = 0, estimate = 1, alternative = "greater")
## expected value of 0.16 = pnorm(q = 0, mean = 1) = mean(x<=0)
boot2pvalue(x, null = 0, estimate = 1, alternative = "less")
## expected value of 0.84 = 1-pnorm(q = 0, mean = 1) = mean(x>=0)

#### negative effect ####
x <- rnorm(1e3, mean = -1)
boot2pvalue(x, null = 0, estimate = -1, alternative = "two.sided")
## expected value of 0.32 = 2*(1-pnorm(q = 0, mean = -1)) = 2*mean(x>=0)
boot2pvalue(x, null = 0, estimate = -1, alternative = "greater")
## expected value of 0.84 = pnorm(q = 0, mean = -1) = mean(x<=0)
boot2pvalue(x, null = 0, estimate = -1, alternative = "less") # pnorm(q = 0, mean = -1)
## expected value of 0.16 = 1-pnorm(q = 0, mean = -1) = mean(x>=0)
```

BuyseMultComp *Adjustment for Multiple Comparisons*

Description

Adjust p-values and confidence intervals estimated via GPC for multiple comparisons.

Usage

```
BuyseMultComp(
  object,
  cluster = NULL,
  linfct = NULL,
  rhs = NULL,
  endpoint = NULL,
  statistic = NULL,
  cumulative = TRUE,
  conf.level = NULL,
  band = TRUE,
  global = FALSE,
  alternative = NULL,
  transformation = NULL,
  ...
)
```

Arguments

object	A BuyseTest object or a list of BuyseTest objects. All objects should contain the same endpoints.
cluster	[character] name of the variable identifying the observations in the dataset used by each BuyseTest model. Only relevant when using a list of BuyseTest objects to correctly combine the influence functions. If NULL, then it is assumed that the BuyseTest objects correspond to different groups of individuals.
linfct	[numeric matrix] a contrast matrix of size the number of endpoints times the number of BuyseTest models.
rhs	[numeric vector] the values for which the test statistic should be tested against. Should have the same number of rows as `linfct`.
endpoint	[character or numeric vector] the endpoint(s) to be considered.
statistic	[character] the statistic summarizing the pairwise comparison: `"netBenefit"` displays the net benefit, as described in Buyse (2010) and Peron et al. (2016)), `"winRatio"` displays the win ratio, as described in Wang et al. (2016), `"favorable"` displays the proportion in favor of the treatment (also called Mann-Whitney parameter), as described in Fay et al. (2018). `"unfavorable"` displays the proportion in favor of the control. Default value read from `BuyseTest.options()`.
cumulative	[logical] should the summary statistic be cumulated over endpoints? Otherwise display the contribution of each endpoint.
conf.level	[numeric] confidence level for the confidence intervals. Default value read from `BuyseTest.options()`.
band	[logical] Should confidence intervals and p-values adjusted for multiple comparisons be computed.
global	[logical] Should global test (intersection of all null hypotheses) be made?
alternative	[character] the type of alternative hypothesis: `"two.sided"`, `"greater"`, or `"less"`. Default value read from `BuyseTest.options()`.

`transformation` [logical] should the CI be computed on the logit scale / log scale for the net benefit / win ratio and backtransformed. Otherwise they are computed without any transformation. Default value read from `BuyseTest.options()`. Not relevant when using permutations or percentile bootstrap.

`...` argument passsed to the function `transformCIBP` of the riskRegression package.

Details

Simulateneous confidence intervals and adjusted p-values are computed using a single-step max-test approach via the function `transformCIBP` of the riskRegression package. This corresponds to the single-step Dunnett described in Dmitrienko et al (2013) in table 2 and section 7.

References

Dmitrienko, A. and D'Agostino, R., Sr (2013), Traditional multiplicity adjustment methods in clinical trials. Statist. Med., 32: 5172-5218. https://doi.org/10.1002/sim.5990

Examples

```
#### simulate data ####
set.seed(10)
df.data <- simBuyseTest(1e2, n.strata = 3)

#### adjustment for all univariate analyses ####
ff1 <- treatment ~ TTE(eventtime, status = status, threshold = 0.1)
ff2 <- update(ff1, .~. + cont(score, threshold = 1))
BT2 <- BuyseTest(ff2, data= df.data, trace = FALSE)

## (require riskRegression >= 2021.10.04 to match)
confint(BT2, cumulative = FALSE) ## not adjusted
confintAdj <- BuyseMultComp(BT2, cumulative = FALSE, endpoint = 1:2) ## adjusted
confintAdj
cor(confintAdj$iid) ## correlation between test-statistic

#### 2- adjustment for multi-arm trial ####
## case where we have more than two treatment groups
## here strata will represent the treatment groups
df.data$strata <- as.character(df.data$strata)
df.data$id <- paste0("Id",1:NROW(df.data)) ## define id variable

BT1ba <- BuyseTest(strata ~ TTE(eventtime, status = status, threshold = 1),
                   data= df.data[strata %in% c("a","b"),], trace = FALSE)
BT1ca <- BuyseTest(strata ~ TTE(eventtime, status = status, threshold = 0.1),
                   data= df.data[strata %in% c("a","c"),], trace = FALSE)
BT1cb <- BuyseTest(strata ~ TTE(eventtime, status = status, threshold = 0.1),
                   data= df.data[strata %in% c("b","c"),], trace = FALSE)
rbind("b-a" = confint(BT1ba),
      "c-a" = confint(BT1ca),
      "c-b" = confint(BT1cb)) ## not adjusted
confintAdj <- BuyseMultComp(list("b-a" = BT1ba, "c-a" = BT1ca, "c-b" = BT1cb),
```

```
                              cluster = "id", global = TRUE)
confintAdj
dim(confintAdj$iid) ## number of subjects x number of analyses
cor(confintAdj$iid)
```

BuyseTest *Generalized Pairwise Comparisons (GPC)*

Description

Performs Generalized Pairwise Comparisons for binary, continuous and time-to-event endpoints.

Usage

```
BuyseTest(
  formula,
  data,
  scoring.rule = NULL,
  pool.strata = NULL,
  correction.uninf = NULL,
  model.tte = NULL,
  method.inference = NULL,
  n.resampling = NULL,
  strata.resampling = NULL,
  hierarchical = NULL,
  weightEndpoint = NULL,
  weightObs = NULL,
  neutral.as.uninf = NULL,
  add.halfNeutral = NULL,
  keep.pairScore = NULL,
  seed = NULL,
  cpus = NULL,
  trace = NULL,
  treatment = NULL,
  endpoint = NULL,
  type = NULL,
  threshold = NULL,
  status = NULL,
  operator = NULL,
  censoring = NULL,
  restriction = NULL,
  strata = NULL
)
```

Arguments

formula	[formula] a symbolic description of the GPC model, typically `treatment ~ type1(endpoint1) + type2(endpoint2, threshold2) + strata`. See Details, section "Specification of the GPC model".

data [data.frame] dataset.

scoring.rule [character] method used to compare the observations of a pair in presence of right censoring (i.e. "timeToEvent" endpoints). Can be "Gehan" or "Peron". See Details, section "Handling missing values".

pool.strata [character] weights used to combine estimates across strata. Can be "Buyse" to weight proportionnally to the number of pairs in the strata, "CMH" to weight proportionnally to the ratio between the number of pairs in the strata and the number of observations in the strata. "equal" to weight equally each strata, or "var-netBenefit" to weight each strata proportionally to the precision of its estimated net benefit (similar syntax for the win ratio: "var-winRatio")

correction.uninf [integer] should a correction be applied to remove the bias due to the presence of uninformative pairs? 0 indicates no correction, 1 impute the average score of the informative pairs, and 2 performs IPCW. See Details, section "Handling missing values".

model.tte [list] optional survival models relative to each time to each time to event endpoint. Models must prodlim objects and stratified on the treatment and strata variable. When used, the uncertainty from the estimates of these survival models is ignored.

method.inference [character] method used to compute confidence intervals and p-values. Can be "none", "u-statistic", "permutation", "studentized permutation", "bootstrap", "studentized bootstrap". See Details, section "Statistical inference".

n.resampling [integer] the number of permutations/samples used for computing the confidence intervals and the p.values. See Details, section "Statistical inference".

strata.resampling [character] the variable on which the permutation/sampling should be stratified. See Details, section "Statistical inference".

hierarchical [logical] should only the uninformative pairs be analyzed at the lower priority endpoints (hierarchical GPC)? Otherwise all pairs will be compaired for all endpoint (full GPC).

weightEndpoint [numeric vector] weights used to cumulating the pairwise scores over the endpoints. Only used when hierarchical=FALSE. Disregarded if the argument formula is defined.

weightObs [character or numeric vector] weights or variable in the dataset containing the weight associated to each observation. These weights are only considered when performing GPC (but not when fitting surival models).

neutral.as.uninf [logical vector] should paired classified as neutral be re-analyzed using endpoints of lower priority (as it is done for uninformative pairs). See Details, section "Handling missing values".

add.halfNeutral [logical] should half of the neutral score be added to the favorable and unfavorable scores?

keep.pairScore [logical] should the result of each pairwise comparison be kept?

seed [integer, >0] the seed to consider when performing resampling. If NULL no seed is set.

cpus [integer, >0] the number of CPU to use. Only the permutation test can use parallel computation. See Details, section "Statistical inference".

trace [integer] should the execution of the function be traced ? 0 remains silent and 1-3 correspond to a more and more verbose output in the console.

treatment, endpoint, type, threshold, status, operator, censoring, restriction, strata Alternative to formula for describing the GPC model. See Details, section "Specification of the GPC model".

Details

Specification of the GPC model:
There are two way to specify the GPC model in BuyseTest. A *Formula interface* via the argument formula where the response variable should be a binary variable defining the treatment arms. The rest of the formula should indicate the endpoints by order of priority and the strata variables (if any). A *Vector interface* using the following arguments

- treatment: [character] name of the treatment variable identifying the control and the experimental group. Must have only two levels (e.g. 0 and 1).
- endpoint: [character vector] the name of the endpoint variable(s).
- threshold: [numeric vector] critical values used to compare the pairs (threshold of minimal important difference). A pair will be classified as neutral if the difference in endpoint is strictly below this threshold. There must be one threshold for each endpoint variable; it must be NA for binary endpoints and positive for continuous or time to event endpoints.
- status: [character vector] the name of the binary variable(s) indicating whether the endpoint was observed or censored. Must value NA when the endpoint is not a time to event.
- operator: [character vector] the sign defining a favorable endpoint. ">0" indicates that higher values are favorable while "<0" indicates the opposite.
- type: [character vector] indicates whether it is a binary outcome ("b", "bin", or "binary"), a continuous outcome ("c", "cont", or "continuous"), or a time to event outcome ("t", "tte", "time", or "timetoevent")
- censoring: [character vector] is the endpoint subject to right or left censoring ("left" or "right"). The default is right-censoring.
- restriction: [numeric vector] value above which any difference is classified as neutral.
- strata: [character vector] if not NULL, the GPC will be applied within each group of patient defined by the strata variable(s).

The formula interface can be more concise, especially when considering few outcomes, but may be more difficult to apprehend for new users. Note that arguments endpoint, threshold, status, operator, type, and censoring must have the same length.

GPC procedure

The GPC procedure form all pairs of observations, one belonging to the experimental group and the other to the control group, and class them in 4 categories:

- *Favorable pair*: the endpoint is better for the observation in the experimental group.
- *Unfavorable pair*: the endpoint is better for the observation in the control group.
- *Neutral pair*: the difference between the endpoints of the two observations is (in absolute value) below the threshold. When `threshold=0`, neutral pairs correspond to pairs with equal endpoint. Lower-priority outcomes (if any) are then used to classified the pair into favorable/unfavorable.
- *Uninformative pair*: censoring/missingness prevents from classifying into favorable, unfavorable or neutral.

With complete data, pairs can be decidely classified as favorable/unfavorable/neutral. In presence of missing values, the GPC procedure uses the scoring rule (argument `scoring.rule`) and the correction for uninformative pairs (argument `correction.uninf`) to classify the pairs. The classification may not be 0,1, e.g. the probability that the pair is favorable/unfavorable/neutral with the Peron's scoring rule. To export the classification of each pair set the argument codekeep.pairScore to `TRUE` and call the function `getPairScore` on the result of the `BuyseTest` function.

Handling missing values

- `scoring.rule`: indicates how to handle right-censoring in time to event endpoints using information from the survival curves. The Gehan's scoring rule (argument `scoring.rule="Gehan"`) only scores pairs that can be decidedly classified as favorable, unfavorable, or neutral while the "Peron"'s scoring rule (argument `scoring.rule="Peron"`) uses the empirical survival curves of each group to also score the pairs that cannot be decidedly classified. The Peron's scoring rule is the recommanded scoring rule but only handles right-censoring.
- `correction.uninf`: indicates how to handle missing values that could not be classified by the scoring rule.

  `correction.uninf=0` treat them as uninformative: this is an equivalent to complete case analysis when `neutral.as.uninf=FALSE`, while when `neutral.as.uninf=TRUE`, uninformative pairs are treated as neutral, i.e., analyzed at the following endpoint (if any). This approach will (generally) lead to biased estimates for the proportion of favorable, unfavorable, or neutral pairs.

  `correction.uninf=1` imputes to the uninformative pairs the average score of the informative pairs, i.e. assumes that uninformative pairs would on average behave like informative pairs. This is therefore the recommanded approach when this assumption is resonnable, typically when the the tail of the survival function estimated by the Kaplan–Meier method is close to 0.

  `correction.uninf=2` uses inverse probability of censoring weights (IPCW), i.e. up-weight informative pairs to represent uninformative pairs. It also assumes that uninformative pairs would on average behave like informative pairs and is only recommanded when the analysis is stopped after the first endpoint with uninformative pairs.

Note that both corrections will convert the whole proportion of uninformative pairs of a given endpoint into favorable, unfavorable, or neutral pairs. See Peron et al (2021) for further details and recommandations

Statistical inference

The argument `method.inference` defines how to approximate the distribution of the GPC estimators and so how standard errors, confidence intervals, and p-values are computed. Available methods are:

- argument `method.inference="none"`: only the point estimate is computed which makes the execution of the `BuyseTest` faster than with the other methods.
- argument `method.inference="u-statistic"`: uses a Gaussian approximation to obtain the distribution of the GPC estimators. The U-statistic theory indicates that this approximation is asymptotically exact. The variance is computed using a H-projection of order 1 (default option), which is a consistent but downward biased estimator. An unbiased estimator can be obtained using a H-projection of order 2 (only available for the uncorrected Gehan's scoring rule, see `BuyseTest.options`). **WARNING**: the current implementation of the H-projection is not valid when using corrections for uninformative pairs (`correction.uninf=1`, or `correction.uninf=2`).
- argument `method.inference="permutation"`: perform a permutation test, estimating in each sample the summary statistics (net benefit, win ratio).
- argument `method.inference="studentized permutation"`: perform a permutation test, estimating in each sample the summary statistics (net benefit, win ratio) and the variance-covariance matrix of the estimate.
- argument `method.inference="bootstrap"`: perform a non-parametric boostrap, estimating in each sample the summary statistics (net benefit, win ratio).
- argument `method.inference=" studentized bootstrap"`: perform a non-parametric boostrap, estimating in each sample the summary statistics (net benefit, win ratio) and the variance-covariance matrix of the estimator.

Additional arguments for permutation and bootstrap resampling:

- `strata.resampling` If `NA` or of length 0, the permutation/non-parametric boostrap will be performed by resampling in the whole sample. Otherwise, the permutation/non-parametric boostrap will be performed separately for each level that the variable defined in `strata.resampling` take.
- `n.resampling` set the number of permutations/samples used. A large number of permutations (e.g. `n.resampling=10000`) are needed to obtain accurate CI and p.value. See (Buyse et al., 2010) for more details.
- `cpus` indicates whether the resampling procedure can be splitted on several cpus to save time. Can be set to `"all"` to use all available cpus. The detection of the number of cpus relies on the `detectCores` function from the *parallel* package.

Pooling results across strata the relative contribution of each strata to the global estimator is decided by

- `"CMH"` weights: Cochran-Mantel-Haenszel type weights which are optimal if the odds ratios are constant across strata.
- `"Buyse"` weights: optimal if the risk difference is constant across strata.

Default values
The default of the arguments `scoring.rule`, `correction.uninf`, `method.inference`, `n.resampling`, `hierarchical`, `neutral.as.uninf`, `keep.pairScore`, `strata.resampling`, `cpus`, `trace` is read from `BuyseTest.options()`.
Additional (hidden) arguments are

- `alternative` [character] the alternative hypothesis. Must be one of "two.sided", "greater" or "less" (used by `confint`).
- `conf.level` [numeric] level for the confidence intervals (used by `confint`).
- `keep.survival` [logical] export the survival values used by the Peron's scoring rule.
- `order.Hprojection` [1 or 2] the order of the H-projection used to compute the variance when `method.inference="u-statistic"`.

Value

An R object of class `S4BuyseTest`.

Author(s)

Brice Ozenne

References

On the GPC procedure: Marc Buyse (2010). **Generalized pairwise comparisons of prioritized endpoints in the two-sample problem**. *Statistics in Medicine* 29:3245-3257
On the win ratio: D. Wang, S. Pocock (2016). **A win ratio approach to comparing continuous non-normal outcomes in clinical trials**. *Pharmaceutical Statistics* 15:238-245
On the Peron's scoring rule: J. Peron, M. Buyse, B. Ozenne, L. Roche and P. Roy (2018). **An extension of generalized pairwise comparisons for prioritized outcomes in the presence of censoring**. *Statistical Methods in Medical Research* 27: 1230-1239.
On the Gehan's scoring rule: Gehan EA (1965). **A generalized two-sample Wilcoxon test for doubly censored data**. *Biometrika* 52(3):650-653
On inference in GPC using the U-statistic theory: Ozenne B, Budtz-Jorgensen E, Peron J (2021). **The asymptotic distribution of the Net Benefit estimator in presence of right-censoring**. *Statistical Methods in Medical Research* 2021 doi:10.1177/09622802211037067
On how to handle right-censoring: J. Peron, M. Idlhaj, D. Maucort-Boulch, et al. (2021) **Correcting the bias of the net benefit estimator due to right-censored observations**. *Biometrical Journal* 63: 893–906.

See Also

S4BuyseTest-summary for a summary of the results of generalized pairwise comparison.
S4BuyseTest-confint for exporting estimates with confidence intervals and p-values.
S4BuyseTest-class for a presentation of the S4BuyseTest object.
S4BuyseTest-sensitivity for performing a sensitivity analysis on the choice of the threshold(s).
constStrata to create a strata variable from several clinical variables.

Examples

```
library(data.table)

#### simulate some data ####
set.seed(10)
df.data <- simBuyseTest(1e2, n.strata = 2)

## display
if(require(prodlim)){
   resKM_tempo <- prodlim(Hist(eventtime,status)~treatment, data = df.data)
   plot(resKM_tempo)
}

#### one time to event endpoint ####
BT <- BuyseTest(treatment ~ TTE(eventtime, status = status), data= df.data)

summary(BT) # net benefit
summary(BT, percentage = FALSE)
summary(BT, statistic = "winRatio") # win Ratio

## permutation instead of asymptotics to compute the p-value
## Not run:
    BT <- BuyseTest(treatment ~ TTE(eventtime, status = status), data=df.data,
                    method.inference = "permutation", n.resampling = 1e3)

## End(Not run)

summary(BT, statistic = "netBenefit") ## default
summary(BT, statistic = "winRatio")

## parallel permutation
## Not run:
    BT <- BuyseTest(treatment ~ TTE(eventtime, status = status), data=df.data,
                    method.inference = "permutation", n.resampling = 1e3, cpus = 2)
    summary(BT)

## End(Not run)

## method Gehan is much faster but does not optimally handle censored observations
BT <- BuyseTest(treatment ~ TTE(eventtime, status = status), data=df.data,
                scoring.rule = "Gehan", trace = 0)
summary(BT)
```

```
#### one time to event endpoint: only differences in survival over 1 unit ####
BT <- BuyseTest(treatment ~ TTE(eventtime, threshold = 1, status = status), data=df.data)
summary(BT)

#### one time to event endpoint with a strata variable
BT <- BuyseTest(treatment ~ strata + TTE(eventtime, status = status), data=df.data)
summary(BT)

#### several endpoints with a strata variable
f <- treatment ~ strata + T(eventtime, status, 1) + B(toxicity)
f <- update(f,
            ~. + T(eventtime, status, 0.5) + C(score, 1) + T(eventtime, status, 0.25))

BT <- BuyseTest(f, data=df.data)
summary(BT)

#### real example : veteran dataset of the survival package ####
## Only one endpoint. Type = Time-to-event. Thresold = 0. Stratfication by histological subtype
## scoring.rule = "Gehan"

if(require(survival)){
## Not run:
  library(survival) ## import veteran

  ## scoring.rule = "Gehan"
  BT_Gehan <- BuyseTest(trt ~ celltype + TTE(time,threshold=0,status=status),
                        data=veteran, scoring.rule="Gehan")

  summary_Gehan <- summary(BT_Gehan)
  summary_Gehan <- summary(BT_Gehan, statistic = "winRatio")

  ## scoring.rule = "Peron"
  BT_Peron <- BuyseTest(trt ~ celltype + TTE(time,threshold=0,status=status),
                        data=veteran, scoring.rule="Peron")

  class(BT_Peron)
  summary(BT_Peron)

## End(Not run)
}
```

`BuyseTest.options` *Global options for BuyseTest package*

Description

Update or select global options for the BuyseTest package.

Usage

```
BuyseTest.options(..., reinitialise = FALSE)
```

Arguments

...	options to be selected or updated
reinitialise	should all the global parameters be set to their default value

Examples

```
library(data.table)

## see all global parameters
BuyseTest.options()

## see some of the global parameters
BuyseTest.options("n.resampling", "trace")

## update some of the global parameters
BuyseTest.options(n.resampling = 10, trace = 1)
BuyseTest.options("n.resampling", "trace")

## reinitialise all global parameters
BuyseTest.options(reinitialise = TRUE)
```

BuyseTest.options-class

Class "BuyseTest.options" (global setting for the BuyseTest package)

Description

Class defining the global settings for the BuyseTest package.

Author(s)

Brice Ozenne

See Also

BuyseTest.options to select or update global settings.

BuyseTest.options-methods *Methods for the class "BuyseTest.options"*

Description

Methods to update or select global settings

Usage

```
## S4 method for signature 'BuyseTest.options'
alloc(object, field)

## S4 method for signature 'BuyseTest.options'
select(object, name.field)
```

Arguments

object	an object of class BuyseTest.options.
field	a list named with the name of the fields to update and containing the values to assign to these fields
name.field	a character vector containing the names of the field to be selected.

BuyseTTEM *Time to Event Model*

Description

Pre-compute quantities of a time to event model useful for predictions. Only does something for prodlim objects.

Usage

```
BuyseTTEM(object, ...)

## S3 method for class 'formula'
BuyseTTEM(object, treatment, iid, iid.surv = "exp", ...)

## S3 method for class 'prodlim'
BuyseTTEM(object, treatment, iid, iid.surv = "exp", ...)

## S3 method for class 'survreg'
BuyseTTEM(object, treatment, n.grid = 1000, iid, ...)

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

Arguments

object	time to event model.
...	additional arguments passed to lower lever methods.
treatment	[character] Name of the treatment variable.
iid	[logical] Should the iid decomposition of the predictions be output.
iid.surv	[character] Estimator of the survival used when computing the influence function. Can be the product limit estimator ("prodlim") or an exponential approximation ("exp", same as in riskRegression::predictCoxPL).
n.grid	[integer, >0] Number of timepoints used to discretize the time scale. Not relevant for prodlim objects.

Examples

```
library(prodlim)
library(data.table)

tau <- seq(0,3,length.out=10)

#### survival case ####
set.seed(10)
df.data <- simBuyseTest(1e2, n.strata = 2)

e.prodlim <- prodlim(Hist(eventtime,status)~treatment+strata, data = df.data)
## plot(e.prodlim)

e.prodlim2 <- BuyseTTEM(e.prodlim, treatment = "treatment", iid = TRUE)

predict(e.prodlim2, time = tau, treatment = "T", strata = "a")
predict(e.prodlim, times = tau, newdata = data.frame(treatment = "T", strata = "a"))

predict(e.prodlim2, time = tau, treatment = "C", strata = "a")
predict(e.prodlim, times = tau, newdata = data.frame(treatment = "C", strata = "a"))

#### competing risk case ####
df.dataCR <- copy(df.data)
df.dataCR$status <- rbinom(NROW(df.dataCR), prob = 0.5, size = 2)

e.prodlimCR <- prodlim(Hist(eventtime,status)~treatment+strata, data = df.dataCR)
## plot(e.prodlimCR)

e.prodlimCR2 <- BuyseTTEM(e.prodlimCR, treatment = "treatment", iid = TRUE)

predict(e.prodlimCR2, time = tau, treatment = "T", strata = "a")
predict(e.prodlimCR, times = tau, newdata = data.frame(treatment = "T", strata = "a"), cause = 1)

predict(e.prodlimCR2, time = tau, treatment = "C", strata = "a")
predict(e.prodlimCR, times = tau, newdata = data.frame(treatment = "C", strata = "a"), cause = 1)
```

coef.BuyseTestAuc *Extract the AUC Value*

Description

Extract the AUC value.

Usage

```
## S3 method for class 'BuyseTestAuc'
coef(object, ...)
```

Arguments

object	object of class BuyseTestAUC (output of the auc function).
...	not used. For compatibility with the generic function.

Value

Estimated value for the AUC (numeric).

coef.BuyseTestBrier *Extract the Brier Score*

Description

Extract the Brier score.

Usage

```
## S3 method for class 'BuyseTestBrier'
coef(object, ...)
```

Arguments

object	object of class BuyseTestBrier (output of the brier function).
...	not used. For compatibility with the generic function.

Value

Estimated value for Brier score (numeric).

confint.BuyseTestAuc *Extract the AUC value with its Confidence Interval*

Description

Extract the AUC value with its Confidence Interval and p-value testing whether the AUC equals 0.5.

Usage

```
## S3 method for class 'BuyseTestAuc'
confint(object, ...)
```

Arguments

object	object of class BuyseTestAUC (output of the auc function).
...	not used. For compatibility with the generic function.

Value

Estimated value for the AUC, its standard error, the lower and upper bound of the confidence interval and the p-value.

confint.BuyseTestBrier *Extract the Brier Score with its Confidence Interval*

Description

Extract the Brier score with its Confidence Interval and possibly a p-value.

Usage

```
## S3 method for class 'BuyseTestBrier'
confint(object, ...)
```

Arguments

object	object of class BuyseTestBrier (output of the brier function).
...	not used. For compatibility with the generic function.

Value

Estimated value for the brier score, its standard error, the lower and upper bound of the confidence interval and the p-value.

`constStrata` *Strata creation*

Description

Create strata from several variables.

Usage

```
constStrata(
  data,
  strata,
  sep = ".",
  lex.order = FALSE,
  trace = TRUE,
  as.numeric = FALSE
)
```

Arguments

`data`	[data.frame] dataset.
`strata`	[character vector] A vector of the variables capturing the stratification factors.
`sep`	[character] string to construct the new level labels by joining the constituent ones.
`lex.order`	[logical] Should the order of factor concatenation be lexically ordered ?
`trace`	[logical] Should the execution of the function be traced ?
`as.numeric`	[logical] Should the strata be converted from factors to numeric?

Details

This function uses the `interaction` function from the *base* package to form the strata.

Value

A *factor vector* or a *numeric vector*.

Author(s)

Brice Ozenne

Examples

```
library(data.table)

library(survival) ## import veteran

# strata with two variables : celltype and karno
```

```
veteran$strata1 <- constStrata(veteran,c("celltype","karno"))
table(veteran$strata1)

# strata with three variables : celltype, karno and age dichotomized at 60 years
veteran$age60 <- veteran$age>60
veteran$age60 <- factor(veteran$age60,labels=c("<=60",">60")) # convert to factor with labels
veteran$strata2 <- constStrata(veteran,c("celltype","karno","age60"))
table(veteran$strata2) # factor strata variable

veteran$strata2 <- constStrata(veteran,c("celltype","karno","age60"), as.numeric=TRUE)
table(veteran$strata2) # numeric strata variable
```

`discreteRoot` *Dichotomic search for monotone function*

Description

Find the root of a monotone function on a discrete grid of value using dichotomic search

Usage

```
discreteRoot(
  fn,
  grid,
  increasing = TRUE,
  check = TRUE,
  tol = .Machine$double.eps^0.5
)
```

Arguments

`fn`	[function] objective function to minimize in absolute value.
`grid`	[vector] possible minimizers.
`increasing`	[logical] is the function fn increasing?
`check`	[logical] should the program check that fn takes a different sign for the first vs. the last value of the grid?
`tol`	[numeric] the absolute convergence tolerance.

Author(s)

Brice Ozenne

getCount *Extract the Number of Favorable, Unfavorable, Neutral, Uninformative pairs*

Description

Extract the number of favorable, unfavorable, neutral, uninformative pairs.

Usage

```
getCount(object, type)

## S4 method for signature 'S4BuyseTest'
getCount(object, type)
```

Arguments

object	an R object of class S4BuyseTest, i.e., output of BuyseTest
type	the type of pairs to be counted. Can be "favorable", "unfavorable", neutral, or uninf. Can also be "all" to select all of them.

Value

A "vector" containing the number of pairs

Author(s)

Brice Ozenne

getIid *Extract the H-decomposition of the Estimator*

Description

Extract the H-decomposition of the GPC estimator.

Usage

```
getIid(
  object,
  endpoint = NULL,
  statistic = NULL,
  stratified = FALSE,
  cumulative = TRUE,
  center = TRUE,
```

```
  scale = TRUE,
  type = "all",
  cluster = NULL
)

## S4 method for signature 'S4BuyseTest'
getIid(
  object,
  endpoint = NULL,
  statistic = NULL,
  stratified = FALSE,
  cumulative = TRUE,
  center = TRUE,
  scale = TRUE,
  type = "all",
  cluster = NULL
)
```

Arguments

object	an R object of class S4BuyseTest, i.e., output of BuyseTest
endpoint	[character] for which endpoint(s) the H-decomposition should be output? If NULL returns the sum of the H-decomposition over all endpoints.
statistic	[character] statistic relative to which the H-decomposition should be output.
stratified	[logical] should the H-decomposition relative to the strata-specific statistics be output? Otherwise display the influence function of the global statistic (i.e. over all strata).
cumulative	[logical] should the H-decomposition be cumulated over endpoints? Otherwise display the contribution of each endpoint.
center	[logical] if TRUE the H-decomposition is centered around 0 (estimated statistic is substracted).
scale	[logical] if TRUE the H-decomposition is rescaled (by the sample size in the corresponding arm) such that its sums of squares approximate the variance of the estimator.
type	[character] type of H-decomposition to be output. Can be only for the nuisance parameters ("nuisance"), or for the u-statistic given the nuisance parameters ("u-statistic"), or both.
cluster	[numeric vector] return the H-decomposition aggregated by cluster.

Details

WARNING: argument scale and center should be used with care as when set to FALSE they may not lead to a meaningful decomposition.

Author(s)

Brice Ozenne

See Also

BuyseTest for performing a generalized pairwise comparison.
S4BuyseTest-summary for a more detailed presentation of the S4BuyseTest object.

---

getPairScore *Extract the Score of Each Pair*

Description

Extract the score of each pair.

Usage

```
getPairScore(
  object,
  endpoint = NULL,
  strata = NULL,
  sum = FALSE,
  rm.withinStrata = TRUE,
  rm.strata = is.na(object@strata),
  rm.indexPair = TRUE,
  rm.weight = FALSE,
  rm.corrected = (object@correction.uninf == 0),
  unlist = TRUE,
  trace = 1
)

## S4 method for signature 'S4BuyseTest'
getPairScore(
  object,
  endpoint = NULL,
  strata = NULL,
  sum = FALSE,
  rm.withinStrata = TRUE,
  rm.strata = is.na(object@strata),
  rm.indexPair = TRUE,
  rm.weight = FALSE,
  rm.corrected = (object@correction.uninf == 0),
  unlist = TRUE,
  trace = 1
)
```

Arguments

object	an R object of class S4BuyseTest, i.e., output of BuyseTest
endpoint	[integer/character vector] the endpoint for which the scores should be output.

strata	[integer/character vector] the strata for which the scores should be output.
sum	[logical] should the scores be cumulated over endpoints?
rm.withinStrata	[logical] should the columns indicating the position of each member of the pair within each treatment group be removed?
rm.strata	[logical] should the column containing the level of the strata variable be removed from the output?
rm.indexPair	[logical] should the column containing the number associated to each pair be removed from the output?
rm.weight	[logical] should the column weight be removed from the output?
rm.corrected	[logical] should the columns corresponding to the scores after weighting be removed from the output?
unlist	[logical] should the structure of the output be simplified when possible?
trace	[logical] should a message be printed to explain what happened when the function returned NULL?

Details

The maximal output (i.e. with all columns) contains for each endpoint, a data.table with:

- "strata": the name of the strata to which the pair belongs.
- "index.T": the index of the treatment observation in the pair relative to the original dataset.
- "index.C": the index of the control observation in the pair relative to the original dataset.
- "indexWithinStrata.T": the index of the treatment observation in the pair relative to the treatment group and the strata.
- "indexWithinStrata.C": the index of the control observation in the pair relative to the control group and the strata.
- "favorable": the probability that the endpoint is better in the treatment arm vs. in the control arm.
- "unfavorable": the probability that the endpoint is worse in the treatment arm vs. in the control arm.
- "neutral": the probability that the endpoint is no different in the treatment arm vs. in the control arm.
- "uninformative": the weight of the pair that cannot be attributed to favorable/unfavorable/neutral.
- "weight": the residual weight of the pair to be analyzed at the current outcome. Each pair starts with a weight of 1.
- "favorable.corrected": same as "favorable" after weighting.
- "unfavorable.corrected": same as "favorable" after weighting.
- "neutral.corrected": same as "favorable" after weighting.
- "uninformative.corrected": same as "favorable" after weighting.

Note that the .T and .C may change since they correspond of the label of the treatment and control arms. The first weighting consists in multiplying the probability by the residual weight of the pair (i.e. the weight of the pair that was not informative at the previous endpoint). This is always performed. For time to event endpoint an additional weighting may be performed to avoid a possible bias in presence of censoring.

Author(s)

Brice Ozenne

Examples

```
library(data.table)
library(prodlim)

## run BuyseTest
library(survival) ## import veteran

BT.keep <- BuyseTest(trt ~ tte(time, threshold = 20, status = "status") + cont(karno),
                     data = veteran, keep.pairScore = TRUE,
                     trace = 0, method.inference = "none")

## Extract scores
pScore <- getPairScore(BT.keep, endpoint = 1)

## look at one pair
indexPair <- intersect(which(pScore$index.1 == 22),
                       which(pScore$index.2 == 71))
pScore[indexPair]

## retrive pair in the original dataset
pVeteran <- veteran[pScore[indexPair,c(index.1,index.2)],]
pVeteran

## the observation from the control group is censored at 97
## the observation from the treatment group has an event at 112
## since the threshold is 20, and (112-20)<97
## we know that the pair is not in favor of the treatment

## the formula for probability in favor of the control is
## Sc(97)/Sc(112+20)
## where Sc(t) is the survival at time t in the control arm.

## we first estimate the survival in each arm
e.KM <- prodlim(Hist(time,status)~trt, data = veteran)

## and compute the survival
iSurv <- predict(e.KM, times =  c(97,112+20),
                 newdata = data.frame(trt = 1, stringsAsFactors = FALSE))[[1]]

## the probability in favor of the control is then
pUF <- iSurv[2]/iSurv[1]
pUF
## and the complement to one of that is the probability of being neutral
pN <- 1 - pUF
pN

if(require(testthat)){
   testthat::expect_equal(pUF, pScore[indexPair, unfavorable])
```

```
   testthat::expect_equal(pN, pScore[indexPair, neutral])
}
```

---

getPseudovalue *Extract the pseudovalues of the Estimator*

Description

Extract the pseudovalues of the estimator. The average of the pseudovalues is the estimate and their standard deviation the standard error of the estimate times a factor n (i.e. a t-test on their mean will give asymptotically valid confidence intervals and p-values).

Usage

```
getPseudovalue(object, statistic = NULL, endpoint = NULL)

## S4 method for signature 'S4BuyseTest'
getPseudovalue(object, statistic = NULL, endpoint = NULL)
```

Arguments

object	an R object of class S4BuyseTest, i.e., output of BuyseTest
statistic	[character] the type of statistic relative to which the pseudovalues should be computed. Can be "netBenefit", "winRatio", "favorable", or "unfavorable".
endpoint	[character] for which endpoint(s) the pseudovalues should be output? If NULL returns the sum of the H-decomposition over all endpoints.

Author(s)

Brice Ozenne

See Also

BuyseTest for performing a generalized pairwise comparison.
S4BuyseTest-summary for a more detailed presentation of the S4BuyseTest object.

Examples

```
set.seed(10)
n <- 250
d <- simBuyseTest(n)

e.BT <- BuyseTest(treatment ~ tte(eventtime,status,2) + bin(toxicity),
                  data = d, trace = 0)

#### net Benefit
pseudo <- getPseudovalue(e.BT)
summary(lm(pseudo~1))$coef
```

```
## asymptotically equivalent to
confint(e.BT, transformation = TRUE)
## (small differences: small sample corrections)

summary(lm(getPseudovalue(e.BT, endpoint = 1)~1))$coef

#### win Ratio
pseudo <- getPseudovalue(e.BT, statistic = "winRatio")
summary(lm(pseudo~1))$coef ## wrong p-value (should compare to 1 instead of 0)
## asymptotically equivalent to
confint(e.BT, statistic = "winRatio", transformation = TRUE)

#### favorable
pseudo <- getPseudovalue(e.BT, statistic = "favorable")
summary(lm(pseudo~1))$coef ## wrong p-value (should compare to 1/2 instead of 0)
## asymptotically equivalent to
confint(e.BT, statistic = "favorable", transformation = TRUE)

#### unfavorable
pseudo <- getPseudovalue(e.BT, statistic = "unfavorable")
summary(lm(pseudo~1))$coef ## wrong p-value (should compare to 1/2 instead of 0)
## asymptotically equivalent to
confint(e.BT, statistic = "unfavorable", transformation = TRUE)
```

`getSurvival` *Extract the Survival and Survival Jumps*

Description

Extract the survival and survival jumps.

Usage

```
getSurvival(
  object,
  type = NULL,
  endpoint = NULL,
  strata = NULL,
  unlist = TRUE,
  trace = TRUE
)

## S4 method for signature 'S4BuyseTest'
getSurvival(
  object,
  type = NULL,
  endpoint = NULL,
  strata = NULL,
  unlist = TRUE,
```

```
  trace = TRUE
)
```

Arguments

object	an R object of class S4BuyseTest, i.e., output of BuyseTest
type	[character vector] the type of survival to be output. See details.
endpoint	[integer/character vector] the endpoint for which the survival should be output.
strata	[integer/character vector] the strata for which the survival should be output.
unlist	[logical] should the structure of the output be simplified when possible.
trace	[logical] should a message be printed to explain what happened when the function returned NULL.

Details

The argument type can take any of the following values:

- "survTimeC": survival at the event times for the observations of the control arm.
- "survTimeT": survival at the event times for the observations of the treatment arm.
- "survJumpC": survival at the jump times for the survival model in the control arm.
- "survJumpT": survival at the time times for the survival model in the treatment arm.
- "lastSurv": survival at the last event time.

Author(s)

Brice Ozenne

GPC_cpp *C++ function performing the pairwise comparison over several endpoints.*

Description

GPC_cpp call for each endpoint and each strata the pairwise comparison function suited to the type of endpoint and store the results.

Usage

```
GPC_cpp(
  endpoint,
  status,
  indexC,
  posC,
  indexT,
  posT,
```

```
  threshold,
  restriction,
  weightEndpoint,
  weightObs,
  method,
  pool,
  op,
  D,
  D_UTTE,
  n_strata,
  nUTTE_analyzedPeron_M1,
  index_endpoint,
  index_status,
  index_UTTE,
  list_survTimeC,
  list_survTimeT,
  list_survJumpC,
  list_survJumpT,
  list_lastSurv,
  p_C,
  p_T,
  iid_survJumpC,
  iid_survJumpT,
  zeroPlus,
  correctionUninf,
  hierarchical,
  hprojection,
  neutralAsUninf,
  addHalfNeutral,
  keepScore,
  precompute,
  returnIID,
  debug
)

GPC2_cpp(
  endpoint,
  status,
  indexC,
  posC,
  indexT,
  posT,
  threshold,
  restriction,
  weightEndpoint,
  weightObs,
  method,
  pool,
```

```
  op,
  D,
  D_UTTE,
  n_strata,
  nUTTE_analyzedPeron_M1,
  index_endpoint,
  index_status,
  index_UTTE,
  list_survTimeC,
  list_survTimeT,
  list_survJumpC,
  list_survJumpT,
  list_lastSurv,
  p_C,
  p_T,
  iid_survJumpC,
  iid_survJumpT,
  zeroPlus,
  correctionUninf,
  hierarchical,
  hprojection,
  neutralAsUninf,
  addHalfNeutral,
  keepScore,
  precompute,
  returnIID,
  debug
)
```

Arguments

endpoint	A matrix containing the values of each endpoint (in columns) for each observation (in rows).
status	A matrix containing the values of the status variables relative to each endpoint (in columns) for each observation (in rows).
indexC	A list containing, for each strata, which rows of the endpoint and status matrices corresponds to the control observations. Not unique when bootstraping.
posC	A list containing, for each strata, the unique identifier of each control observations.
indexT	A list containing, for each strata, which rows of the endpoint and status matrices corresponds to the treatment observations. Not unique when bootstraping.
posT	A list containing, for each strata, the unique identifier of each treatment observations.
threshold	Store the thresholds associated to each endpoint. Must have length D. The threshold is ignored for binary endpoints.
restriction	Store the restriction time associated to each endpoint. Must have length D.

weightEndpoint	Store the weight associated to each endpoint. Must have length D.
weightObs	A vector containing the weight associated to each observation.
method	The index of the method used to score the pairs. Must have length D. 1 for binary/continuous, 2 for Gaussian, 3/4 for Gehan (left or right-censoring), and 5/6 for Peron (right-censoring survival or competing risks).
pool	The index of the method used to pool results across strata. Can be 0 (weight inversely proportional to the sample size), 1 (Mantel Haenszel weights), 2 (equal weights), 3 (precision weights)
op	The index of the operator used to score the pairs. Must have length D. 1 for larger is beter, -1 for smaller is better.
D	The number of endpoints.
D_UTTE	The number of distinct time to event endpoints.
n_strata	The number of strata.
nUTTE_analyzedPeron_M1	The number of unique time-to-event endpoints that have been analyzed the Peron scoring rule before the current endpoint. Must have length D.
index_endpoint	The position of the endpoint at each priority in the argument endpoint. Must have length D.
index_status	The position of the status at each priority in the argument status. Must have length D.
index_UTTE	The position, among all the unique tte endpoints, of the TTE endpoints. Equals -1 for non tte endpoints. Must have length n_TTE.
list_survTimeC	A list of matrix containing the survival estimates (-threshold, 0, +threshold ...) for each event of the control group (in rows).
list_survTimeT	A list of matrix containing the survival estimates (-threshold, 0, +threshold ...) for each event of the treatment group (in rows).
list_survJumpC	A list of matrix containing the survival estimates and survival jumps when the survival for the control arm jumps.
list_survJumpT	A list of matrix containing the survival estimates and survival jumps when the survival for the treatment arm jumps.
list_lastSurv	A list of matrix containing the last survival estimate in each strata (rows) and treatment group (columns).
p_C	Number of nuisance parameter in the survival model for the control group, for each endpoint and strata
p_T	Number of nuisance parameter in the survival model for the treatment group, for each endpoint and strata
iid_survJumpC	A list of matrix containing the iid of the survival estimates in the control group.
iid_survJumpT	A list of matrix containing the iid of the survival estimates in the treatment group.
zeroPlus	Value under which doubles are considered 0?
correctionUninf	Should the uninformative weight be re-distributed to favorable and unfavorable?

hierarchical	Should only the uninformative pairs be analyzed at the lower priority endpoints (hierarchical GPC)? Otherwise all pairs will be compaired for all endpoint (full GPC).
hprojection	Order of the H-projection used to compute the variance.
neutralAsUninf	Should paired classified as neutral be re-analyzed using endpoints of lower priority?
addHalfNeutral	Should half of the neutral score be added to the favorable and unfavorable scores?
keepScore	Should the result of each pairwise comparison be kept?
precompute	Have the integrals relative to the survival be already computed and stored in list_survTimeC/list_survTimeT and list_survJumpC/list_survJumpT (derivatives)
returnIID	Should the iid be computed? Second element: is there any nuisance parameter?
debug	Print messages tracing the execution of the function to help debugging. The amount of messages increase with the value of debug (0-5).

Details

GPC_cpp implements GPC looping first over endpoints and then over pairs. To handle multiple endpoints, it stores some of the results which can be memory demanding when considering large sample - especially when computing the iid decomposition. GPC2_cpp implements GPC looping first over pairs and then over endpoints. It has rather minimal memory requirement but does not handle correction for uninformative pairs.

Author(s)

Brice Ozenne

---

iid.BuyseTestAuc *Extract the idd Decomposition for the AUC*

Description

Extract the iid decompotion relative to AUC estimate.

Usage

```
## S3 method for class 'BuyseTestAuc'
iid(x, ...)
```

Arguments

x	object of class BuyseTestAUC (output of the auc function).
...	not used. For compatibility with the generic function.

Value

A column vector.

iid.BuyseTestBrier *Extract the idd Decomposition for the Brier Score*

Description

Extract the iid decompotion relative to Brier score estimate.

Usage

```
## S3 method for class 'BuyseTestBrier'
iid(x, ...)
```

Arguments

x	object of class BuyseTestBrier (output of the brier function).
...	not used. For compatibility with the generic function.

Value

A column vector.

iid.prodlim *Extract i.i.d. decomposition from a prodlim model*

Description

Compute the influence function for each observation used to estimate the model

Usage

```
## S3 method for class 'prodlim'
iid(x, add0 = FALSE, ...)
```

Arguments

x	A prodlim object.
add0	[logical] add the 0 to vector of relevant times.
...	not used. For compatibility with the generic method.

Details

This function is a simplified version of the iidCox function of the riskRegression package. Formula for the influence function can be found in (Ozenne et al., 2017).

Author(s)

Brice Ozenne

References

Brice Ozenne, Anne Lyngholm Sorensen, Thomas Scheike, Christian Torp-Pedersen and Thomas Alexander Gerds. riskRegression: Predicting the Risk of an Event using Cox Regression Models. The R Journal (2017) 9:2, pages 440-460.

Examples

```
library(data.table)
library(prodlim)

set.seed(10)
dt <- simBuyseTest(10)
setkeyv(dt, "treatment")

e.KM <- prodlim(Hist(eventtime,status)~treatment, data = dt)
lava::iid(e.KM)
```

---

performance *Assess Performance of a Classifier*

Description

Assess the performance in term of AUC and brier score of one or several binary classifiers. Currently limited to logistic regressions and random forest.

Usage

```
performance(
  object,
  data = NULL,
  newdata = NA,
  individual.fit = FALSE,
  impute = "none",
  name.response = NULL,
  fold.size = 1/10,
  fold.repetition = 0,
  fold.balance = FALSE,
  null = c(brier = NA, AUC = 0.5),
  conf.level = 0.95,
  se = TRUE,
  transformation = TRUE,
  auc.type = "classical",
  simplify = TRUE,
```

```
  trace = TRUE,
  seed = NULL
)
```

Arguments

object	a glm or range object, or a list of such object.
data	[data.frame] the training data.
newdata	[data.frame] an external data used to assess the performance.
individual.fit	[logical] if TRUE the predictive model is refit for each individual using only the predictors with non missing values.
impute	[character] in presence of missing value in the regressors of the training dataset, should a complete case analysis be performed ("none") or should the median/mean ("median"/"mean") value be imputed. For categorical variables, the most frequent value is imputed.
name.response	[character] the name of the response variable (i.e. the one containing the categories).
fold.size	[double, >0] either the size of the test dataset (when >1) or the fraction of the dataset (when <1) to be used for testing when using cross-validation.
fold.repetition	[integer] when strictly positive, the number of folds used in the cross-validation. If 0 then no cross validation is performed.
fold.balance	[logical] should the outcome distribution in the folds of the cross-validation be similar to the one of the original dataset?
null	[numeric vector of length 2] the right-hand side of the null hypothesis relative to each metric.
conf.level	[numeric] confidence level for the confidence intervals.
se	[logical] should the uncertainty about AUC/brier be computed? When TRUE adapt the method of LeDell et al. (2015) to repeated cross-validation for the AUC and the brier score.
transformation	[logical] should the CI be computed on the logit scale / log scale for the net benefit / win ratio and backtransformed. Otherwise they are computed without any transformation.
auc.type	[character] should the auc be computed approximating the predicted probability by a dirac ("classical", usual AUC formula) or approximating the predicted probability by a normal distribution.
simplify	[logical] should the number of fold and the size of the fold used for the cross validation be removed from the output?
trace	[logical] Should the execution of the function be traced.
seed	[integer, >0] seed used to ensure reproducibility.

References

LeDell E, Petersen M, van der Laan M. Computationally efficient confidence intervals for cross-validated area under the ROC curve estimates. Electron J Stat. 2015;9(1):1583-1607. doi:10.1214/15-EJS1035

Examples

```
## Simulate data
set.seed(10)
n <- 100
df.train <- data.frame(Y = rbinom(n, prob = 0.5, size = 1), X1 = rnorm(n), X2 = rnorm(n))
df.test <- data.frame(Y = rbinom(n, prob = 0.5, size = 1), X1 = rnorm(n), X2 = rnorm(n))

## fit logistic model
e.null <- glm(Y~1, data = df.train, family = binomial(link="logit"))
e.logit1 <- glm(Y~X1, data = df.train, family = binomial(link="logit"))
e.logit2 <- glm(Y~X1+X2, data = df.train, family = binomial(link="logit"))

## assess performance on the training set (biased)
## and external dataset
performance(e.logit1, newdata = df.test)
e.perf <- performance(list(null = e.null, p1 = e.logit1, p2 = e.logit2),
                      newdata = df.test)
e.perf
summary(e.perf, order.model = c("null","p2","p1"))

## assess performance using cross validation
## Not run:
set.seed(10)
performance(e.logit1, fold.repetition = 10, se = FALSE)
set.seed(10)
performance(list(null = e.null, prop = e.logit1), fold.repetition = 10)
performance(e.logit1, fold.repetition = c(50,20,10))

## End(Not run)
```

`performanceResample` *Uncertainty About Performance of a Classifier (EXPERIMENTAL)*

Description

Use resampling to quantify uncertainties about the performance of one or several binary classifiers evaluated via cross-validation.

Usage

```
performanceResample(
  object,
  data = NULL,
  name.response = NULL,
  type.resampling = "permutation",
  n.resampling = 1000,
  fold.repetition = 0,
  conf.level = 0.95,
  cpus = 1,
```

```
  seed = NULL,
  trace = TRUE,
  filename = NULL,
  ...
)
```

Arguments

object	a glm or range object, or a list of such object.
data	[data.frame] the training data.
name.response	[character] The name of the response variable (i.e. the one containing the categories).
type.resampling	[character] Should non-parametric bootstrap ("bootstrap") or permutation of the outcome ("permutation") be used.
n.resampling	[integer,>0] Nnumber of bootstrap samples or permutations.
fold.repetition	[integer,>0] Nnumber of folds used in the cross-validation. Should be strictly positive.
conf.level	[numeric, 0-1] confidence level for the confidence intervals.
cpus	[integer, >0] the number of CPU to use. If strictly greater than 1, resampling is perform in parallel.
seed	[integer, >0] seed used to ensure reproducibility.
trace	[logical] Should the execution of the function be traced.
filename	[character] Prefix for the files containing each result.
...	arguments passed to performance.

Details

WARNING: using bootstrap after cross-validation may not provide valid variance/CI/p-value estimates.

---

powerBuyseTest *Performing simulation studies with BuyseTest*

Description

Performs a simulation studies for several sample sizes. Returns estimates, their standard deviation, the average estimated standard error, and the rejection rate.

Usage

```
powerBuyseTest(
  sim,
  sample.size,
  sample.sizeC = NULL,
  sample.sizeT = NULL,
  n.rep,
  null = c(netBenefit = 0),
  cpus = 1,
  seed = NULL,
  conf.level = NULL,
  power = NULL,
  max.sample.size = 5000,
  alternative = NULL,
  order.Hprojection = NULL,
  transformation = NULL,
  trace = 1,
  ...
)
```

Arguments

sim	[function] take two arguments: the sample size in the control group (n.C) and the sample size in the treatment group (n.C) and generate datasets. The datasets must be data.table objects.
sample.size	[integer vector, >0] the various sample sizes at which the simulation should be perform. Disregarded if any of the arguments sample.sizeC or sample.sizeT are specified.
sample.sizeC	[integer vector, >0] the various sample sizes in the control group.
sample.sizeT	[integer vector, >0] the various sample sizes in the treatment group.
n.rep	[integer, >0] the number of simulations.
null	[numeric vector] For each statistic of interest, the null hypothesis to be tested. The vector should be named with the names of the statistics.
cpus	[integer, >0] the number of CPU to use. Default value is 1.
seed	[integer, >0] the seed to consider for the simulation study.
conf.level	[numeric, 0-1] type 1 error level. Default value read from BuyseTest.options().
power	[numeric, 0-1] type 2 error level used to determine the sample size. Only relevant when sample.size, sample.sizeC, and sample.sizeT are not given. See details.
max.sample.size	[interger, 0-1] sample size used to approximate the sample size achieving the requested type 1 and type 2 error (see details). Can have length 2 to indicate the sample in each group when the groups have unequal sample size.
alternative	[character] the type of alternative hypothesis: "two.sided", "greater", or "less". Default value read from BuyseTest.options().

`order.Hprojection` [integer 1,2] the order of the H-project to be used to compute the variance of the net benefit/win ratio. Default value read from `BuyseTest.options()`.

`transformation` [logical] should the CI be computed on the logit scale / log scale for the net benefit / win ratio and backtransformed. Otherwise they are computed without any transformation. Default value read from `BuyseTest.options()`.

`trace` [integer] should the execution of the function be traced?

`...` other arguments (e.g. `scoring.rule`, `method.inference`) to be passed to `initializeArgs`.

Details

Sample size calculation: to approximate the sample size achieving the requested type 1 (α) and type 2 error (β), GPC are applied on a large sample (as defined by the argument `max.sample.size`): $N^* = m^* + n^*$ where m^* is the sample size in the control group and n^* is the sample size in the active group. Then the effect (δ) and the asymptotic variance of the estimator (σ^2) are estimated. The total sample size is then deduced as (two-sided case):

$$\hat{N} = \hat{\sigma}^2 \frac{(u_{1-\alpha/2} + u_{1-\beta})^2}{\hat{\delta}^2}$$

from which the group specific sample sizes are deduced: $\hat{m} = \hat{N}\frac{m^*}{N^*}$ and $\hat{n} = \hat{N}\frac{n^*}{N^*}$. Here \(u_x\) denotes the x-quantile of the normal distribution. Since this is an approximation, a simulation study is performed with this sample size to provide the estimated power. It may not be exactly the requested power but should provide a reasonnable guess which can be refined with further simulation studies. Note that the maximumal sample size should be very high for the estimation to be accurate (ideally 10000 or more).

Author(s)

Brice Ozenne

Examples

```
library(data.table)

#### Using simBuyseTest ####
## only point estimate
## Not run:
powerBuyseTest(sim = simBuyseTest, sample.size = c(10, 25, 50, 75, 100),
               formula = treatment ~ bin(toxicity), seed = 10, n.rep = 1000,
               method.inference = "none", trace = 2, keep.pairScore = FALSE)

## End(Not run)

## point estimate with rejection rate

## Not run:
powerBuyseTest(sim = simBuyseTest, sample.size = c(10, 50, 100),
               formula = treatment ~ bin(toxicity), seed = 10, n.rep = 1000,
               method.inference = "u-statistic", trace = 4)
```

```
## End(Not run)

#### Using user defined simulation function ####
## power calculation for Wilcoxon test
simFCT <- function(n.C, n.T){
    out <- rbind(cbind(Y=stats::rt(n.C, df = 5), group=0),
                 cbind(Y=stats::rt(n.T, df = 5), group=1) + 1)
    return(data.table::as.data.table(out))
}
simFCT2 <- function(n.C, n.T){
    out <- rbind(cbind(Y=stats::rt(n.C, df = 5), group=0),
                 cbind(Y=stats::rt(n.T, df = 5), group=1) + 0.25)
    return(data.table::as.data.table(out))
}


## Not run:
powerW <- powerBuyseTest(sim = simFCT, sample.size = c(5,10,20,30,50,100),
                         n.rep = 1000, formula = group ~ cont(Y), cpus = "all")
summary(powerW)

## End(Not run)

## sample size needed to reach (approximately) a power
## based on summary statistics obtained on a large sample
## Not run:
sampleW <- powerBuyseTest(sim = simFCT, power = 0.8, max.sample.size = 10000,
                         n.rep = 1000, formula = group ~ cont(Y), cpus = "all")
summary(sampleW) ## not very accurate but gives an order of magnitude

sampleW2 <- powerBuyseTest(sim = simFCT2,
                           power = 0.8, max.sample.size = 10000,
                           n.rep = 1000, formula = group ~ cont(Y), cpus = "all")
summary(sampleW2) ## more accurate with larger samples

## End(Not run)
```

`predict.BuyseTTEM` *Prediction with Time to Event Model*

Description

Evaluate the cumulative incidence function (cif) / survival in one of the treatment groups.

Usage

```
## S3 method for class 'BuyseTTEM'
predict(object, time, treatment, strata, cause = 1, iid = FALSE, ...)
```

Arguments

object	time to event model.
time	[numeric vector] time at which to evaluate the cif/survival.
treatment	[character/integer] Treatment or index of the treatment group.
strata	[character/integer] Strata or index of the strata.
cause	[integer] The cause relative to which the cif will be evaluated.
iid	[logical] Should the influence function associated with the cif/survival be output?
...	not used, for compatibility with the generic method.

---

rbind.performance *Combine Resampling Results For Performance Objects*

Description

Combine permutation or bootstrap samples. Useful to run parallel calculations (see example below).

Usage

```
## S3 method for class 'performance'
rbind(..., tolerance = 1e-05)
```

Arguments

...	performance objects.
tolerance	[numeric] maximum acceptable difference between the point estimates. Can be NA to skip this sanity check.

Examples

```
if(FALSE){

#### simulate data ####
set.seed(10)
n <- 100
df.train <- data.frame(Y = rbinom(n, prob = 0.5, size = 1),
                       X1 = rnorm(n), X2 = rnorm(n), X3 = rnorm(n), X4 = rnorm(n),
                       X5 = rnorm(n), X6 = rnorm(n), X7 = rnorm(n), X8 = rnorm(n),
                       X9 = rnorm(n), X10 = rnorm(n))
df.train$Y <- rbinom(n, size = 1,
                     prob = 1/(1+exp(-df.train$X5 - df.train$X6 - df.train$X7)))

#### fit models ####
e.null <- glm(Y~1, data = df.train, family = binomial(link="logit"))
e.logit <- glm(Y~X1+X2, data = df.train, family = binomial(link="logit"))
e.logit2 <- glm(Y~X1+X2+X3+X4+X5+X6+X7+X8+X9+X10, data = df.train,
```

```
               family = binomial(link="logit"))

#### evaluate model (same seed) ####
fold.repetition <- 5 ## 0: internal perf (fast)
                     ## >0: 10 fold CV repeated (slow)
test <- performanceResample(list(e.logit,e.logit2), seed = 10,
                            fold.repetition = fold.repetition, n.resampling = 100)
test.1 <- performanceResample(list(e.logit,e.logit2), seed = 10,
                            fold.repetition = fold.repetition, n.resampling = 1:50)
test.2 <- performanceResample(list(e.logit,e.logit2), seed = 10,
                            fold.repetition = fold.repetition, n.resampling = 51:100)
rbind(test.1,test.2)
test

## Note: when the prediction model call RNG then test.1 and test.2 may not give test

#### evaluate model (different seed) ####
test.3 <- performanceResample(list(e.logit,e.logit2), seed = 11,
                            fold.repetition = fold.repetition, n.resampling = 1:50)
test.4 <- performanceResample(list(e.logit,e.logit2), seed = 12,
                            fold.repetition = fold.repetition, n.resampling = 51:100)
rbind(test.3,test.4, tolerance = NA) ## does not check equality of the point estimate
                                     ## between test.3 and test.4
test
}
```

S4BuysePower-class *Class "S4BuysePower" (output of BuyseTest)*

Description

A powerBuyseTest output is reported in a S4BuysePower object.

Author(s)

Brice Ozenne

See Also

powerBuyseTest for the function computing generalized pairwise comparisons.
S4BuysePower-summary for the summary of the BuyseTest function results

S4BuysePower-show *Show Method for Class "S4BuysePower"*

Description

Display the main results stored in a S4BuysePower object.

Usage

```
## S4 method for signature 'S4BuysePower'
show(object)
```

Arguments

object	an R object of class S4BuysePower, i.e., output of BuyseTest

Author(s)

Brice Ozenne

See Also

BuyseTest for performing a generalized pairwise comparison.
S4BuysePower-summary for a more detailed presentation of the S4BuysePower object.

S4BuysePower-summary *Summary Method for Class "S4BuysePower"*

Description

Summarize the results from the powerBuyseTest function.

Usage

```
## S4 method for signature 'S4BuysePower'
summary(
  object,
  print = TRUE,
  statistic = NULL,
  endpoint = NULL,
  order.Hprojection = NULL,
  transformation = NULL,
  legend = TRUE,
  col.rep = FALSE,
  digit = 4
)
```

Arguments

object	output of powerBuyseTest
print	[logical] Should the table be displayed?.
statistic	[character] statistic relative to which the power should be computed: "netBenefit" displays the net benefit, as described in Buyse (2010) and Peron et al. (2016)), "winRatio" displays the win ratio, as described in Wang et al. (2016), "mannWhitney" displays the proportion in favor of the treatment (also called Mann-Whitney parameter), as described in Fay et al. (2018). Default value read from BuyseTest.options().
endpoint	[character vector] the endpoints to be displayed: must be the name of the endpoint followed by an underscore and then by the threshold.
order.Hprojection	[integer 1,2] the order of the H-project to be used to compute the variance of the net benefit/win ratio.
transformation	[logical] should the CI be computed on the logit scale / log scale for the net benefit / win ratio and backtransformed.
legend	[logical] should explainations about the content of each column be displayed?
col.rep	[logical] should the number of successful simulations be displayed?
digit	[integer vector] the number of digit to use for printing the counts and the delta.

Author(s)

Brice Ozenne

References

On the GPC procedure: Marc Buyse (2010). **Generalized pairwise comparisons of prioritized endpoints in the two-sample problem**. *Statistics in Medicine* 29:3245-3257
On the win ratio: D. Wang, S. Pocock (2016). **A win ratio approach to comparing continuous non-normal outcomes in clinical trials**. *Pharmaceutical Statistics* 15:238-245
On the Mann-Whitney parameter: Fay, Michael P. et al (2018). **Causal estimands and confidence intervals asscoaited with Wilcoxon-Mann-Whitney tests in randomized experiments**. *Statistics in Medicine* 37:2923-2937 \

See Also

powerBuyseTest for performing a simulation study for generalized pairwise comparison.

S4BuyseTest-class *Class "S4BuyseTest" (output of BuyseTest)*

Description

A BuyseTest output is reported in a S4BuyseTest object.

Author(s)

Brice Ozenne

See Also

BuyseTest for the function computing generalized pairwise comparisons.
S4BuyseTest-summary for the summary of the BuyseTest function results

S4BuyseTest-coef *Coef Method for Class "S4BuyseTest"*

Description

Extract summary statistics from the result of a BuyseTest function.

Usage

```
## S4 method for signature 'S4BuyseTest'
coef(
  object,
  endpoint = NULL,
  statistic = NULL,
  stratified = FALSE,
  cumulative = NULL,
  resampling = FALSE,
  ...
)
```

Arguments

object	output of BuyseTest
endpoint	[character] for which endpoint(s) the summary statistic should be output? If NULL returns the summary statistic for all endpoints.
statistic	[character] the type of summary statistic. See the detail section.
stratified	[logical] should the summary statistic be strata-specific? Otherwise a summary statistic over all strata is returned.

cumulative	[logical] should the summary statistic be cumulated over endpoints? Otherwise display the contribution of each endpoint.
resampling	[logical] should the summary statistic obtained by resampling be output?
...	ignored.

Details

One of the following statistic can be specified:

- "netBenefit": returns the net benefit.
- "winRatio": returns the win ratio.
- "favorable": returns the proportion in favor of the treatment (also called Mann-Whitney parameter).
- "unfavorable": returns the proportion in favor of the control.
- "unfavorable": returns the proportion of neutral pairs.
- "unfavorable": returns the proportion of uninformative pairs.
- "count.favorable": returns the number of pairs in favor of the treatment.
- "count.unfavorable": returns the number of pairs in favor of the control.
- "count.neutral": returns the number of neutral pairs.
- "count.uninf": returns the number of uninformative pairs.

Value

When resampling=FALSE and stratified=FALSE, return a numeric vector. When resampling=FALSE and stratified=TRUE, return a matrix. When resampling=TRUE and stratified=FALSE, return a matrix. When resampling=TRUE and stratified=TRUE, return an 3-dimensional array.

Author(s)

Brice Ozenne

S4BuyseTest-confint *Confidence Intervals for Model Parameters*

Description

Computes confidence intervals for net benefit statistic or the win ratio statistic.

Usage

```
## S4 method for signature 'S4BuyseTest'
confint(
  object,
  endpoint = NULL,
  statistic = NULL,
  stratified = FALSE,
  cumulative = TRUE,
  null = NULL,
  conf.level = NULL,
  alternative = NULL,
  method.ci.resampling = NULL,
  order.Hprojection = NULL,
  transformation = NULL,
  cluster = NULL,
  sep = "."
)
```

Arguments

object	an R object of class S4BuyseTest, i.e., output of BuyseTest
endpoint	[character] for which endpoint(s) the confidence intervals should be output? If NULL returns the confidence intervals for all endpoints.
statistic	[character] the statistic summarizing the pairwise comparison: "netBenefit" displays the net benefit, as described in Buyse (2010) and Peron et al. (2016)), "winRatio" displays the win ratio, as described in Wang et al. (2016), "favorable" displays the proportion in favor of the treatment (also called Mann-Whitney parameter), as described in Fay et al. (2018). "unfavorable" displays the proportion in favor of the control. Default value read from BuyseTest.options().
stratified	[logical] should strata-specific statistic be output? Otherwise output a global statistics obtained after pooling the strata-specific ones.
cumulative	[logical] should the summary statistic be cumulated over endpoints? Otherwise display the contribution of each endpoint.
null	[numeric] right hand side of the null hypothesis (used for the computation of the p-value).
conf.level	[numeric] confidence level for the confidence intervals. Default value read from BuyseTest.options().
alternative	[character] the type of alternative hypothesis: "two.sided", "greater", or "less". Default value read from BuyseTest.options().
method.ci.resampling	[character] the method used to compute the confidence intervals and p-values when using bootstrap or permutation ("percentile", "gaussian", "student"). See the details section.
order.Hprojection	[integer, 1-2] order of the H-decomposition used to compute the variance.

`transformation`	[logical] should the CI be computed on the inverse hyperbolic tangent scale / log scale for the net benefit / win ratio and backtransformed. Otherwise they are computed without any transformation. Default value read from `BuyseTest.options()`. Not relevant when using permutations or percentile bootstrap.
`cluster`	[numeric vector] Group of observations for which the iid assumption holds .
`sep`	[character] character string used to separate the endpoint and the strata when naming the statistics.

Details

statistic: when considering a single endpoint and denoting Y the endpoint in the treatment group, X the endpoint in the control group, and τ the threshold of clinical relevance, the net benefit is $P[Y \geq X + \tau] - P[X \geq Y + \tau]$, the win ratio is $\frac{P[Y \geq X+\tau]}{P[X \geq Y+\tau]}$, the proportion in favor of treatment is $P[Y \geq X + \tau]$, the proportion in favor of control is $P[X \geq Y + \tau]$.

method.ci.resampling: when using bootstrap/permutation, p-values and confidence intervals are computing as follow:

- `percentile` (bootstrap): compute the confidence interval using the quantiles of the bootstrap estimates. Compute the p-value by finding the confidence level at which a bound of the confidence interval equals the null hypothesis.
- `percentile` (permutation): apply the selected transformation to the estimate and permutation estimates. Compute the confidence interval by (i) shfiting the estimate by the quantiles of the centered permutation estimates and (ii) back-transforming . Compute the p-value as the relative frequency at which the estimate are less extreme than the permutation estimates.
- `gaussian` (bootstrap and permutation): apply the selected transformation to the estimate and bootstrap/permutation estimates. Estimate the variance of the estimator using the empirical variance of the transformed boostrap/permutation estimates. Compute confidence intervals and p-values under the normality assumption and back-transform the confidence intervals.
- `student` (bootstrap): apply the selected transformation to the estimate, its standard error, the bootstrap estimates, and their standard error. Compute the studentized bootstrap estimates by dividing the centered bootstrap estimates by their standard error. Compute the confidence interval based on the standard error of the estimate and the quantiles of the studentized bootstrap estimates, and back-transform. Compute the p-value by finding the confidence level at which a bound of the confidence interval equals the null hypothesis.
- `student` (permutation): apply the selected transformation to the estimate, its standard error, the permutation estimates, and their standard error. Compute the studentized permutation estimates by dividing the centered permutation estimates by their standard error. Compute the confidence interval based on the standard error of the estimate and the quantiles of the studentized permutation estimates, and back-transform. Compute the p-value as the relative frequency at which the studentized estimate are less extreme than the permutation studentized estimates.

WARNING: when using a permutation test, the uncertainty associated with the estimator is computed under the null hypothesis. Thus the confidence interval may not be valid if the null hypothesis is false.

Value

A matrix containing a column for the estimated statistic (over all strata), the lower bound and upper bound of the confidence intervals, and the associated p-values. When using resampling methods:

- an attribute `n.resampling` specified how many samples have been used to compute the confidence intervals and the p-values.
- an attribute `method.ci.resampling` method used to compute the confidence intervals and p-values.

Author(s)

Brice Ozenne

References

On the GPC procedure: Marc Buyse (2010). **Generalized pairwise comparisons of prioritized endpoints in the two-sample problem**. *Statistics in Medicine* 29:3245-3257
On the win ratio: D. Wang, S. Pocock (2016). **A win ratio approach to comparing continuous non-normal outcomes in clinical trials**. *Pharmaceutical Statistics* 15:238-245
On the Mann-Whitney parameter: Fay, Michael P. et al (2018). **Causal estimands and confidence intervals asscoaited with Wilcoxon-Mann-Whitney tests in randomized experiments**. *Statistics in Medicine* 37:2923-2937

See Also

`BuyseTest` for performing a generalized pairwise comparison.
`S4BuyseTest-summary` for a more detailed presentation of the `S4BuyseTest` object.

S4BuyseTest-show *Show Method for Class "S4BuyseTest"*

Description

Display the main results stored in a `S4BuyseTest` object.

Usage

```
## S4 method for signature 'S4BuyseTest'
show(object)
```

Arguments

`object`	an R object of class `S4BuyseTest`, i.e., output of `BuyseTest`

Author(s)

Brice Ozenne

See Also

BuyseTest for performing a generalized pairwise comparison.
S4BuyseTest-summary for a more detailed presentation of the S4BuyseTest object.

S4BuyseTest-summary *Summary Method for Class "S4BuyseTest"*

Description

Summarize the results from the BuyseTest function.

Usage

```
## S4 method for signature 'S4BuyseTest'
summary(
  object,
  print = TRUE,
  percentage = TRUE,
  statistic = NULL,
  conf.level = NULL,
  strata = if (length(object@level.strata) == 1) {
     "global"
 } else {
     NULL
 },
  type.display = 1,
  digit = c(2, 4, 5),
  ...
)
```

Arguments

object	output of BuyseTest
print	[logical] Should the table be displayed?.
percentage	[logical] Should the percentage of pairs of each type be displayed ? Otherwise the number of pairs is displayed.
statistic	[character] the statistic summarizing the pairwise comparison: "netBenefit" displays the net benefit, as described in Buyse (2010) and Peron et al. (2016)), "winRatio" displays the win ratio, as described in Wang et al. (2016), "favorable" displays the proportion in favor of the treatment (also called Mann-Whitney parameter), as described in Fay et al. (2018). "unfavorable" displays the proportion in favor of the control. Default value read from BuyseTest.options().
conf.level	[numeric] confidence level for the confidence intervals. Default value read from BuyseTest.options().

strata	[character vector] the name of the strata to be displayed. Can also be "global" to display the average over all strata.
type.display	[numeric or character] the results/summary statistics to be displayed. Either an integer indicating refering to a type of display in BuyseTest.options() or the name of the column to be output (e.g. c("strata","Delta","p.value")).
digit	[integer vector] the number of digit to use for printing the counts and the delta.
...	arguments to be passed to S4BuyseTest-confint

Details

Content of the output

The "results" table in the output show the result of the GPC at each endpoint, as well as its contribution to the global statistics. More precisely, the column:

- endpoint lists the endpoints, by order of priority.
- threshold lists the threshold associated to each endpoint.
- **weight:** lists the weight of each priority.
- **strata:** list the strata relative to which the results of the priority are displayed. If "global", then the results are over all strata at a given priority.
- total lists the number of pairs to be analyzed at the current priority (or strata).
- total(%) lists the percentage of pairs to be analyzed at the current priority (or strata).
- favorable lists the number of pairs classified in favor of the treatment group at the current priority (or strata).
- favorable(%) lists the number of pairs classified in favor of the treatment group at the current priority (or strata).
- unfavorable lists the number of pairs classified in favor of the control group at the current priority (or strata).
- unfavorable(%) lists the percentage of pairs classified in favor of the control group at the current priority (or strata).
- neutral lists the number of pairs classified as neither in favor of the treatment group nor in favor of the control group at the current priority (or strata).
- neutral(%) lists the percentage of pairs classified as neither in favor of the treatment group nor in favor of the control group at the current priority (or strata).
- uninf lists the number of pairs that could not be classified at the current priority (or strata) due to missing values/censoring.
- uninf(%) lists the percentage of pairs that could not be classified at the current priority (or strata) due to missing values/censoring.
- delta lists the value of the priority-specific statistic (e.g. net benefit or win ratio), i.e. computed on the pairs analyzed at the current priority only.
- Delta lists the value of the cumulative statistic (e.g. net benefit or win ratio), i.e. computed on all the pairs analyzed up to the current priority.
- Delta(%) lists the relative statistic (i.e. statistic up to the current priority divided by the final statistic).

- `information(%)` lists the information fraction (i.e. number of favorable and unfavorable pairs up to the current priority divided by the final number of favorable and unfavorable pairs).
- `CI` lists the confidence intervals for `Delta` (not adjusted for multiple comparison).
- `null` lists the null hypothesis (`Delta=null`).
- `p.value` p-value relative to the null hypothesis (no adjustment for multiple comparison).
- `resampling` number of samples used to compute the confidence intervals or p-values from permutations or bootstrap samples. Only displayed if some bootstrap samples have been discarded, for example, they did not lead to sample any case or control.

Note: when using the Peron scoring rule or a correction for uninformative pairs, the columns `total`, `favorable`, `unfavorable`, `neutral`, and `uninf` are computing by summing the contribution of the pairs. This may lead to a decimal value.

statistic: when considering a single endpoint and denoting Y the endpoint in the treatment group, X the endpoint in the control group, and τ the threshold of clinical relevance, the net benefit is $P[Y \geq X + \tau] - P[X \geq Y + \tau]$, the win ratio is $\frac{P[Y \geq X+\tau]}{P[X \geq Y+\tau]}$, the proportion in favor of treatment is $P[Y \geq X + \tau]$, the proportion in favor of control is $P[X \geq Y + \tau]$.

Statistical inference
When the interest is in obtaining p-values, we recommand the use of a permutation test. However, when using a permutation test confidence intervals are not displayed in the summary. This is because there is no (to the best of our knowledge) straightforward way to obtain good confidence intervals with permutations. An easy way consist in using the quantiles of the permutation distribution and then shift by the point estimate of the statistic. This is what is output by `S4BuyseTest-confint`. However this approach leads to a much too high coverage when the null hypothesis is false. The limits of the confidence interval can also end up being outside of the interval of definition of the statistic (e.g. outside [-1,1] for the proportion in favor of treatment). Therefore, for obtaining confidence intervals, we recommand the boostrap method or the u-statistic method.

Win ratio
For the win ratio, the proposed implementation enables the use of thresholds and endpoints that are not time to events as well as the correction proposed in Peron et al. (2016) to account for censoring. These development have not been examined by Wang et al. (2016), or in other papers (to the best of our knowledge). They are only provided here by implementation convenience.

Competing risks
In presence of competing risks, looking at the net benefit/win ratio computed with respect to the event of interest will likely not give a full picture of the difference between the two groups. For instance a treatment may decrease the risk of the event of interest (i.e. increase the net benefit for this event) by increasing the risk of the competing event. If the competing event is death, this is not desirable. It is therefore advised to taking into consideration the risk of the competing event, e.g. by re-running BuyseTest where cause 1 and 2 have been inverted.

Author(s)

Brice Ozenne

References

On the GPC procedure: Marc Buyse (2010). **Generalized pairwise comparisons of prioritized endpoints in the two-sample problem**. *Statistics in Medicine* 29:3245-3257

On the win ratio: D. Wang, S. Pocock (2016). **A win ratio approach to comparing continuous non-normal outcomes in clinical trials**. *Pharmaceutical Statistics* 15:238-245
On the Mann-Whitney parameter: Fay, Michael P. et al (2018). **Causal estimands and confidence intervals asscoaited with Wilcoxon-Mann-Whitney tests in randomized experiments**. *Statistics in Medicine* 37:2923-2937 \

See Also

`BuyseTest` for performing a generalized pairwise comparison.
`S4BuyseTest-class` for a presentation of the `S4BuyseTest` object.
`S4BuyseTest-confint` to output confidence interval and p-values in a matrix format.

Examples

```
library(data.table)

dt <- simBuyseTest(1e2, n.strata = 3)

 ## Not run:
 BT <- BuyseTest(treatment ~ TTE(eventtime, status = status) + Bin(toxicity), data=dt)

## End(Not run)

 summary(BT)
 summary(BT, percentage = FALSE)
 summary(BT, statistic = "winRatio")
```

`sensitivity` *Sensitivity Analysis for the Choice of the Thresholds*

Description

Evaluate the statistic of interest along various thresholds of clinical relevance.

Usage

```
sensitivity(object, ...)

## S4 method for signature 'S4BuyseTest'
sensitivity(
  object,
  threshold,
  statistic = NULL,
  band = FALSE,
  conf.level = NULL,
  null = NULL,
  transformation = NULL,
```

```
  alternative = NULL,
  adj.p.value = FALSE,
  trace = TRUE,
  cpus = 1,
  ...
)
```

Arguments

object	an R object of class S4BuyseTest, i.e., output of BuyseTest
...	argument passsed to the function transformCIBP of the riskRegression package.
threshold	[list] a list containing for each endpoint the thresholds to be considered.
statistic	[character] the statistic summarizing the pairwise comparison: "netBenefit" displays the net benefit, as described in Buyse (2010) and Peron et al. (2016)), "winRatio" displays the win ratio, as described in Wang et al. (2016), "favorable" displays the proportion in favor of the treatment (also called Mann-Whitney parameter), as described in Fay et al. (2018). "unfavorable" displays the proportion in favor of the control. Default value read from BuyseTest.options().
band	[logical] should simulateneous confidence intervals be computed?
conf.level	[numeric] confidence level for the confidence intervals. Default value read from BuyseTest.options().
null	[numeric] right hand side of the null hypothesis (used for the computation of the p-value).
transformation	[logical] should the CI be computed on the logit scale / log scale for the net benefit / win ratio and backtransformed. Otherwise they are computed without any transformation. Default value read from BuyseTest.options(). Not relevant when using permutations or percentile bootstrap.
alternative	[character] the type of alternative hypothesis: "two.sided", "greater", or "less". Default value read from BuyseTest.options().
adj.p.value	[logical] should p-value adjusted for multiple comparisons be computed?
trace	[logical] Should the execution of the function be traced?
cpus	[integer, >0] the number of CPU to use. Default value is 1.

Details

Simulateneous confidence intervals and adjusted p-values are computed using a single-step max-test approach via the function transformCIBP of the riskRegression package.

Examples

```
## Not run:
require(ggplot2)

## simulate data
```

```
set.seed(10)
df.data <- simBuyseTest(1e2, n.strata = 2)

## with one endpoint
ff1 <- treatment ~ TTE(eventtime, status = status, threshold = 0.1)
BT1 <- BuyseTest(ff1, data= df.data)
se.BT1 <- sensitivity(BT1, threshold = seq(0,2,0.25), band = TRUE)
autoplot(se.BT1)

## with two endpoints
ff2 <- update(ff1, .~. + cont(score, threshold = 1))
BT2 <- BuyseTest(ff2, data= df.data)
se.BT2 <- sensitivity(BT2, threshold = list(eventtime = seq(0,2,0.25), score = 0:2),
                      band = TRUE)
autoplot(se.BT2)
autoplot(se.BT2, col = NA)

## End(Not run)
```

`simCompetingRisks` *Simulation of Gompertz competing risks data for the BuyseTest*

Description

Simulate Gompertz competing risks data with proportional (via prespecified sub-distribution hazard ratio) or non-proportional sub-distribution hazards. A treatment variable with two groups (treatment and control) is created.

Usage

```
simCompetingRisks(
  n.T,
  n.C,
  p.1C = NULL,
  v.1C,
  v.1T,
  v.2C,
  v.2T,
  sHR = NULL,
  b.1T = NULL,
  b.1C = NULL,
  b.2T = NULL,
  b.2C = NULL,
  cens.distrib = NULL,
  param.cens = NULL,
  latent = NULL
)
```

Arguments

n.T	[integer, >0] number of patients in the treatment arm
n.C	[integer, >0] number of patients in the control arm
p.1C	[integer, >0] proportion of events of interest in the control group. Can be NULL if and only if (b.1T, b.1C, b.2T, b.2C) are provided.
v.1C, v.1T, v.2C, v.2T	[double, <0] shape parameters for Gompertz distribution of time to event of interest in control/treatment (C/T) group and of time to competing event in control/treatment (C/T) group respectively
sHR	[double, >0] pre-specified sub-distribution hazard ratio for event of interest. Can be NULL if and only if (b.1T, b.1C, b.2T, b.2C) are provided.
b.1C, b.1T, b.2C, b.2T	[double, >0] rate parameters for Gompertz distribution of time to event of interest in control/treatment (C/T) group and of time to competing event in control/treatment (C/T) group respectively. Can be NULL if and only if (p.1C, sHR) are provided.
cens.distrib	[character] censoring distribution. Can be "exponential" for exponential censoring or "uniform" for uniform censoring. NULL means no censoring.
param.cens	[>0] parameter for censoring distribution. Should be a double for rate parameter of exponential censoring distribution or a vector of doubles for lower and upper bounds of uniform censoring distribution. NULL means no censoring
latent	[logical] If TRUE, also export the latent variables (e.g. true event times, true event types and censoring times). NULL sets this parameter to FALSE.

Details

The times to the event of interest and to the competing event in each group follow an improper Gompertz distribution (see Jeong and Fine, 2006), whose cumulative distribution function is

$F(t; b, v) = 1 - \exp(b (1 - \exp (v t)) / v)$

and hazard functions is

$h(t; b, v) = b \exp(v t)$

The shape parameters must be negative to have improper distributions for the times to the two events in each group. Note however that in each group, the overall cumulative incidence function must be proper (i.e. the maximum values of the cumulative incidence of each event type sum up to 1 in each group). When only providing the shape parameters, the rate parameters are computed to fulfill this condition. In case you whish to provide the rate parameters too, make sure that the condition is met.

Author(s)

Eva Cantagallo

References

Jeong J-H. and Fine J. (2006) **Direct parametric inference for the cumulative incidence function**. *Journal of the Royal Statistical Society* 55: 187-200

Examples

```
#### Providing p.1C and sHR ####
d <- simCompetingRisks(n.T = 100, n.C = 100, p.1C = 0.55, v.1C = -0.30,
v.1T = -0.30, v.2C = -0.30, v.2T = -0.30, sHR = 0.5, b.1T = NULL,
b.1C = NULL, b.2T = NULL, b.2C = NULL)

#### Providing the rate parameters ####
d <- simCompetingRisks(n.T = 100, n.C = 100, p.1C = NULL, v.1C = -0.30,
v.1T = -0.30, v.2C = -0.30, v.2T = -0.30, sHR = NULL, b.1T = 0.12,
b.1C = 0.24, b.2T = 0.33, b.2C = 0.18)

#### With exponential censoring ####
d <- simCompetingRisks(n.T = 100, n.C = 100, p.1C = 0.55, v.1C = -0.30,
v.1T = -0.30, v.2C = -0.30, v.2T = -0.30, sHR = 0.5, b.1T = NULL,
b.1C = NULL, b.2T = NULL, b.2C = NULL, cens.distrib = "exponential",
param.cens = 0.8, latent = TRUE)

### With uniform censoring ####
d <- simCompetingRisks(n.T = 100, n.C = 100, p.1C = 0.55, v.1C = -0.30,
v.1T = -0.30, v.2C = -0.30, v.2T = -0.30, sHR = 0.5, b.1T = NULL,
b.1C = NULL, b.2T = NULL, b.2C = NULL, cens.distrib = "uniform",
param.cens = c(0, 7), latent=TRUE)
```

Simulate endpoints for GPC

Simulation of data for the BuyseTest

Description

Simulate categorical, continuous or time to event endpoints, possibly along with a strata variable. Categorical endpoints are simulated by thresholding a latent Gaussian variable (tobit model), continuous endpoints are simulated using a Gaussian distribution, and time to event endpoints are simulated using Weibull distributions for the event of interest, competing events, and censoring. This function is built upon the `lvm` and `sim` functions from the lava package.

Usage

```
simBuyseTest(
  n.T,
  n.C = NULL,
```

```
  argsBin = list(),
  argsCont = list(),
  argsTTE = list(),
  n.strata = NULL,
  names.strata = NULL,
  format = "data.table",
  latent = FALSE
)
```

Arguments

`n.T`	[integer, >0] number of patients in the treatment arm
`n.C`	[integer, >0] number of patients in the control arm
`argsBin`	[list] arguments to be passed to `simBuyseTest_bin`. They specify the distribution parameters of the categorical endpoints.
`argsCont`	[list] arguments to be passed to `simBuyseTest_continuous`. They specify the distribution parameters of the continuous endpoints.
`argsTTE`	[list] arguments to be passed to `simBuyseTest_TTE`. They specify the distribution parameters of the time to event endpoints.
`n.strata`	[integer, >0] number of strata. `NULL` indicates no strata.
`names.strata`	[character vector] name of the strata variables. Must have same length as `n.strata`.
`format`	[character] the format of the output. Can be `"data.table"`, `"data.frame"` or `"matrix"`.
`latent`	[logical] If `TRUE` also export the latent variables (e.g. censoring times or event times).

Details

Endpoints are simulated independently of the strata variable and independently of each other, with the exception of categorical endpoint and the time to event endpoints that can be correlated by specifying a non-0 value for the `rho.T` and `rho.C` elements of the argument `argsBin`.

Arguments in the list `argsBin`:

- `p.T` list of probabilities for the values taken by each endpoint (categorical endpoint, treatment group).
- `p.C` same as `p.T` but for the control group.
- `rho.T` value of the regression coefficient between the underlying latent variable and the survival time. Only implemented for weibull distributed survival times.
- `rho.C` same as `rho.T` but for the control group.
- `name` names of the binary variables.

Arguments in the list `argsCont`:

- `mu.T` expected value of each endpoint (continuous endpoint, treatment group).
- `mu.C` same as `mu.C` but for the control group.

- sigma.T standard deviation of the values of each endpoint (continuous endpoint, treatment group).
- sigma.C same as sigma.T but for the control group.
- name names of the continuous variables.

Arguments in the list argsTTE:

- CR should competing risks be simulated?
- scale.T,scale.C,scale.CR,scale.censoring.T,scale.censoring.C scale parameter of the Weibull distribution for, respectively, the event of interest in the treatment group, the event of interest in the control group, the competing event in both groups, the censoring mechanism in the treatment group, the censoring mechanism in the control group
- shape.T,shape.C,shape.CR,shape.censoring.T,shape.censoring.C shape parameter of the Weibull distribution for, respectively, the event of interest in the treatment group, the event of interest in the control group, the competing event in both groups, the censoring mechanism in the treatment group, the censoring mechanism in the control group
- dist.T,dist.C,dist.CR,dist.censoring.T,dist.censoring.C type of distribution ("weibull", "uniform", "piecewiseExp") for, respectively, the event of interest in the treatment group, the event of interest in the control group, the competing event in both groups, the censoring mechanism in the treatment group, the censoring mechanism in the control group. For uniform distirbutions the (scale,shape) parameters becomes the support (min, max) of the censoring distribution. For piecewise exponential distributions the (scale,shape) should be lists of numeric (see example) and the shape parameters becomes the time parameters (first element should be 0).
- name names of the time to event variables.
- name.censoring names of the event type indicators. #'

Author(s)

Brice Ozenne

Examples

```
library(data.table)

n <- 1e2

#### by default ####
simBuyseTest(n)

## with a strata variable having 5 levels
simBuyseTest(n, n.strata = 5)
## with a strata variable named grade
simBuyseTest(n, n.strata = 5, names.strata = "grade")
## several strata variables
simBuyseTest(1e3, n.strata = c(2,4), names.strata = c("Gender","AgeCategory"))

#### only categorical endpoints ####
args <- list(p.T = list(c(low=0.1,moderate=0.5,high=0.4)))
```

```
dt.bin <- simBuyseTest(n, argsBin = args, argsCont = NULL, argsTTE = NULL)
table(dt.bin$toxicity)/NROW(dt.bin)

args <- list(p.T = list(c(low=0.1,moderate=0.5,high=0.4), c(0.1,0.9)))
dt.bin <- simBuyseTest(n, argsBin = args, argsCont = NULL, argsTTE = NULL)
table(dt.bin$toxicity1)/NROW(dt.bin)
table(dt.bin$toxicity2)/NROW(dt.bin)

#### only continuous endpoints ####
args <- list(mu.T = c(3:5/10), sigma.T = rep(1,3))
dt.cont <- simBuyseTest(n, argsBin = NULL, argsCont = args, argsTTE = NULL)
c(mean(dt.cont$score1), mean(dt.cont$score2), mean(dt.cont$score3))
c(sd(dt.cont$score1), sd(dt.cont$score2), sd(dt.cont$score3))

#### only TTE endpoints ####
## weibull distributed
args <- list(scale.T = c(3:5/10), scale.censoring.T = rep(1,3))
dt.tte <- simBuyseTest(n, argsBin = NULL, argsCont = NULL, argsTTE = args)
1/c(sum(dt.tte$eventtime1)/sum(dt.tte$status1),
  sum(dt.tte$eventtime2)/sum(dt.tte$status2),
  sum(dt.tte$eventtime3)/sum(dt.tte$status3))

1/c(sum(dt.tte$eventtime1)/sum(dt.tte$status1==0),
  sum(dt.tte$eventtime2)/sum(dt.tte$status2==0),
  sum(dt.tte$eventtime3)/sum(dt.tte$status3==0))

hist(dt.tte$eventtime1)

## uniform distributed
args <- list(scale.T = 0, shape.T = 1, dist.T = "uniform", scale.censoring.T = 1e5,
             scale.C = 0, shape.C = 2, dist.C = "uniform", scale.censoring.C = 1e5)
dt.tte <- simBuyseTest(n, argsBin = NULL, argsCont = NULL, argsTTE = args)

par(mfrow=c(1,2))
hist(dt.tte$eventtime[dt.tte$treatment=="C"])
hist(dt.tte$eventtime[dt.tte$treatment=="T"])

## piecewise constant exponential distributed
## time [0;4]: scale parameter 10
## time [4;12]: scale parameter 13
## time [12;18.]: scale parameter 18
## time [18.5;36]: scale parameter 31
## after that: scale parameter 37
vec.scale <- list(c(10,13,18,31,100))
vec.time <- list(c(0,4,12,18.5,36))
args <- list(scale.T = vec.scale, shape.T = vec.time, dist.T = "piecewiseExp",
             scale.C = 10, shape.C = 1, dist.C = "weibull",
             scale.censoring.T = 1e5)
dt.tte <- simBuyseTest(n, argsBin = NULL, argsCont = NULL, argsTTE = args)

if(require(prodlim)){
plot(prodlim(Hist(eventtime,status)~treatment, data = dt.tte))
}
```

```
#### correlated categorical / time to event endpoint ####
## WARNING: only for weibull distributed time to event endpoint
args.bin <- list(p.T = list(c(low=0.1,moderate=0.5,high=0.4)), rho.T = 1)
args.tte <- list(scale.T = 2, scale.censoring.T = 1)
dt.corr <- simBuyseTest(n, argsBin = args.bin, argsCont = NULL, argsTTE = args.tte)

1/(sum(dt.corr$eventtime)/sum(dt.corr$status))
1/(sum(dt.corr$eventtime)/sum(dt.corr$status==0))
table(dt.corr$toxicity)/NROW(dt.corr)

boxplot(eventtime ~ toxicity, data = dt.corr)
```

`summary.performance` *Summary Method for Performance Objects*

Description

Summary of the performance of binary classifiers

Usage

```
## S3 method for class 'performance'
summary(object, order.model = NULL, digits = c(3, 3), print = TRUE, ...)
```

Arguments

`object`	output of performance.
`order.model`	[character vector] ordering of the models.
`digits`	[numeric vector of length 2] number of digits used for the estimates and p-values.
`print`	[logical] should the performance be printed in the console.
`...`	not used.

Index