Package 'BuyseTest'

January 16, 2019

Type Package

Title Generalized Pairwise Comparisons

Version 1.7

Date 2019-01-15

Description Implementation of the Generalized Pairwise Comparisons (GPC). GPC compare two groups of observations (intervention vs. control group) regarding several prioritized endpoints.
The net benefit and win ratio statistics can then be estimated and corresponding confidence intervals and p-values can be estimated using resampling methods or the asymptotic U-statistic theory. The software enables the use of thresholds of minimal importance difference, stratification, and corrections to deal with right-censored endpoints or missing values.

License GPL-3

Encoding UTF-8

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

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

Depends R (>= 2.10), prodlim, Rcpp, data.table

Imports doParallel, foreach, methods, lava, parallel, stats, stats4, utils

Suggests pbapply, R.rsp, survival, testthat

LinkingTo Rcpp, RcppArmadillo

SystemRequirements C++11

VignetteBuilder R.rsp

NeedsCompilation yes

RoxygenNote 6.1.0.9000

Collate '0-onLoad.R' '1-setGeneric.R' 'BuyseRes-object.R' 'BuyseRes-confint.R' 'BuyseRes-iid.R' 'BuyseRes-summary.R' 'BuyseRes_get.R' 'BuyseRes_show.R' 'BuyseSim-object.R' 'BuyseSim-summary.R' 'BuyseSim_show.R' 'BuyseTest-check.R'

'BuyseTest-inference.R' 'BuyseTest-initialization.R'
'BuyseTest-package.R' 'BuyseTest-print.R' 'BuyseTest.R'
'BuyseTest.options-object.R' 'BuyseTest.options.R'
'PairScore.R' 'RcppExports.R' 'constStrata.R' 'discreteRoot.R'
'powerBuyseTest.R' 'seBuyseTest.R' 'simBuyseTest.R' 'valid.R'

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

Maintainer Brice Ozenne <brice.ozenne@orange.fr>

Repository CRAN

Date/Publication 2019-01-16 09:00:07 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 value of their endpoints.

If the difference in endpoint value between the two observations of the pair is greater than the threshold of clinical relevance, the pair is classified as favorable (i.e. win). If the difference is lower than minus the threshold of clinical relevance 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

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*.

`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, 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.
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)
```

`BuyseRes-class` *Class "BuyseRes" (output of BuyseTest)*

Description

A `BuyseTest` output is reported in a `BuyseRes` object.

See Also

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

`BuyseRes-confint` *Confidence Intervals for Model Parameters*

Description

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

Usage

```
## S4 method for signature 'BuyseRes'
confint(object, statistic = NULL,
  conf.level = NULL, alternative = NULL, method.boot = "percentile",
  transformation = NULL)
```

Arguments

object	an R object of class BuyseRes, i.e., output of BuyseTest
statistic	[character] the statistic summarizing the pairwise comparison: "netBenefit" displays the net benefit, as described in Buyse (2010) and Peron et al. (2016)), whereas "winRatio" displays the win ratio, as described in Wang et al. (2016). Default value read from BuyseTest.options().
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.boot	[character] the method used to compute the boostrap confidence intervals and p-values. Can be "percentile" for computing the CI using the quantiles of the boostrap distribution or "gaussian" for using a Gaussian approximation to compute the CI where the standard error is computed using the bootstrap samples.
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().

Details

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.
More precisely, the quantiles of the distribution of the statistic are computed under the null hypothesis and then shifted by the point estimate of the statistic. Therefore it is possible that the limits of the confidence interval are estimated outside of the interval of definition of the statistic (e.g. outside [-1,1] for the proportion in favor of treatment).

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.

See Also

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

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

Description

Display the main results stored in a BuyseRes object.

Usage

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

Arguments

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

See Also

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

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

Description

Summarize the results from the BuyseTest function.

Usage

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

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)), whereas "winRatio" displays the win ratio, as described in Wang et al. (2016). 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.
digit	[integer vector] the number of digit to use for printing the counts and the delta.
...	arguments to be passed to confint

Details

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.
More precisely, the quantiles of the distribution of the statistic are computed under the null hypothesis and then shifted by the point estimate of the statistic. Therefore it is possible that the limits of the confidence interval are estimated outside of the interval of definition of the statistic (e.g. outside [-1,1] for the proportion in favor of treatment).

Note: 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 (at out knowledge). They are only provided here by implementation convenience.

See Also

BuyseTest for performing a generalized pairwise comparison.
BuyseRes-class for a presentation of the BuyseRes object.

Examples

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

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

## End(Not run)

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

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

Description

A powerBuyseTest output is reported in a BuyseSim object.

See Also

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

---

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

Description

Display the main results stored in a BuyseSim object.

Usage

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

Arguments

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

See Also

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

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

Description

Summarize the results from the powerBuyseTest function.

Usage

```
## S4 method for signature 'BuyseSim'
summary(object, print = TRUE, statistic = NULL,
  legend = TRUE, method.inference = NULL, col.rep = TRUE,
  digit = 4)
```

Arguments

object	output of powerBuyseTest
print	[logical] Should the table be displayed?.
statistic	[character] the statistic summarizing the pairwise comparison: "netBenefit" displays the net benefit whereas "winRatio" displays the win ratio.
legend	[logical] should explainations about the content of each column be displayed?
method.inference	[character vector] for which inference method the rejection rate should 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.
...	arguments to be passed from the generic method to the class specific method [not relevant to the user]

See Also

powerBuyseTest for performing a simulation study for generalized pairwise comparison.

BuyseTest *Generalized Pairwise Comparisons (GPC)*

Description

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

Usage

```
BuyseTest(formula, data, method.tte = NULL, correction.uninf = NULL,
  model.tte = NULL, method.inference = NULL, n.resampling = NULL,
  neutral.as.uninf = NULL, keep.pairScore = NULL, treatment = NULL,
  endpoint = NULL, type = NULL, threshold = NULL, censoring = NULL,
  operator = NULL, strata = NULL, alternative = NULL, seed = 10,
  cpus = NULL, trace = NULL, keep.comparison)
```

Arguments

formula	[formula] a symbolic description of the model to be fitted. The response variable should be a binary variable defining the treatment arms. The rest of the formula should indicate the strata variables (if any) and the endpoints by order of priority.
data	[data.frame] dataset.

method.tte [character] defines the method used to compare the observations of a pair in presence of right censoring (i.e. "timeToEvent" endpoints). Can be "Gehan" or "Peron". "Gehan" only scores pairs that can be decidedly classified as favorable, unfavorable, or neutral. "Peron" uses the empirical survival curves of each group to also score the pairs that cannot be decidedly classified (see Peron et al. for more details). Default value read from BuyseTest.options().

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 pair, and 2 performs inverse probability of censoring weights. Default value read from BuyseTest.options().

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.

method.inference [character] should the asymptotic theory ("asymptotic"), or a permutation test ("permutation" or "stratified permutation"), or bootstrap resampling ("bootstrap" or "stratified boostrap") be used to compute p-values and confidence intervals.

n.resampling [integer] the number of simulations used for computing the confidence interval and the p.values. See details. Default value read from BuyseTest.options().

neutral.as.uninf [logical] should paired classified as neutral be re-analyzed using endpoints of lower priority. Default value read from BuyseTest.options().

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

treatment [character] the name of the treatment variable identifying the control and the experimental group. Disregarded if the argument formula is defined.

endpoint [character vector] the name of the endpoint variable(s). Disregarded if the argument formula is defined.

type [character vector] the type of each endpoint: "binary", "continuous" or "timeToEvent".

threshold [numeric vector] critical values used to compare the pairs (threshold of minimal important difference). There must be one threshold for each endpoint variable. Disregarded if the argument formula is defined.

censoring [character vector] the name of the binary variable(s) indicating whether the endpoint was observed or censored. There must be one threshold for each endpoint variable. Must value NA when the endpoint is not a time to event. Disregarded if the argument formula is defined.

operator [character vector] the sign defining a favorable endpoint: ">0" indicates that higher values are favorable while "<0" indicates the opposite. Disregarded if the argument formula is defined.

strata [numeric vector] if not NULL, the GPC will be applied within each group of patient defined by the strata variable(s). Disregarded if the argument formula is defined.

alternative [character] the alternative hypothesis. Must be one of "two.sided", "greater" or "less". Default value read from BuyseTest.options().

seed [integer, >0] the seed to consider for the permutation test. If NULL no seed is set.

cpus [integer, >0] the number of CPU to use. Only the permutation test can use parallel computation. Default value read from BuyseTest.options().

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. Default value read from BuyseTest.options().

keep.comparison
Obsolete. Alias for 'keep.pairScore'.

Details

treatment: The variable corresponding to treatment in data must have only two levels (e.g. 0 and 1).
endpoint, threshold, censoring, operator, and type: they must have the same length.
threshold must be NA for binary endpoints and positive for continuous or time to event endpoints. censoring must be NA for binary or continuous endpoints and indicate a variable in data for time to event endpoints. Short forms for endpoint type are "bin" (binary endpoint), "cont" (continuous endpoint), \ code"TTE" (time-to-event endpoint). **operator:** when the operator is set to "<0" the corresponding column in the dataset is multiplied by -1.

n.resampling: The number of permutation replications must be specified to enable the computation of the confidence intervals and the p.value. 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 parallelization: Argument cpus 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 .

Dealing with neutral or uninformative pairs: Neutral pairs correspond to pairs for which the difference between the endpoint of the control observation and the endpoint of the treatment observation is (in absolute value) below the threshold. When threshold=0, neutral pairs correspond to pairs with equal endpoint.
Uninformative pairs correspond to pairs for which the censoring prevent from classifying them into favorable, unfavorable or neutral. Neutral or uninformative pairs for an endpoint with priority l are, when available, analyzed on the endpoint with priority l-1.

method.tte: the method.tte="Peron" is recommended in presence of right censored observations since it gives a more efficient estimator than method.tte="Gehan".

method.inference: the method.inference="asymptotic" estimate the distribution of the net benefit or win ratio statistics based on the H-decomposition of U-statistics. method.inference="asymptotic-bebu" is an equivalent approach based on Bebu et al. 2015 (formula 2.2). The current implementation of "asymptotic" and "asymptotic-bebu" is not valid in small sample or when using method.tte="Peron", or correction.uninf=1, or correction.uninf=2.

correction.uninf: in presence of uninformative pairs, the proportion of favorable, unfavorable, or neutral pairs is underestimated. Inverse probability of censoring weights (correction.uninf=2) is only recommanded when the analysis is stopped after the first endpoint with uninformative pairs. Imputing the average score of the informative pairs (correction.uninf=1) gives equivalent results at the first endpoint but better behaves at latter endpoints. Note that both corrections will convert the whole proportion of uninformative pairs of a given endpoint into favorable, unfavorable, or neutral pairs.

Value

An R object of class BuyseRes.

References

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
D. Wang, S. Pocock (2016). **A win ratio approach to comparing continuous non-normal outcomes in clinical trials**. *Pharmaceutical Statistics* 15:238-245
I. Bebu, J. M. Lachin (2015). **Large sample inference for a win ratio analysis of a composite outcome based on prioritized components**. *Biostatistics* 17(1):178-187
Marc Buyse (2010). **Generalized pairwise comparisons of prioritized endpoints in the two-sample problem**. *Statistics in Medicine* 29:3245-3257
Efron B (1967). **The two sample problem with censored data**. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* 4:831-583
Gehan EA (1965). **A generalized two-sample Wilcoxon test for doubly censored data**. *Biometrika* 52(3):650-653

See Also

BuyseRes-summary for a summary of the results of generalized pairwise comparison.
BuyseRes-class for a presentation of the BuyseRes object.
constStrata to create a strata variable from several clinical variables.

Examples

```
# reset the default value of the number of permuation sample
BuyseTest.options(method.inference = "none") # no permutation test

#### 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, censoring = status), data= df.data)

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

## bootstrap to compute the CI
```

```
## Not run:
    BT <- BuyseTest(Treatment ~ TTE(eventtime, censoring = status), data=df.data,
                    method.inference = "permutation", n.resampling = 1e3)

## End(Not run)

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

## parallel boostrap
## Not run:
    BT <- BuyseTest(Treatment ~ TTE(eventtime, censoring = 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, censoring = status), data=df.data,
                method.tte = "Gehan", trace = 0)
summary(BT)

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

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

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

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
#### method.tte = "Gehan"

if(require(survival)){
## Not run:
  data(veteran,package="survival")

  ## method.tte = "Gehan"
  BT_Gehan <- BuyseTest(trt ~ celltype + TTE(time,threshold=0,censoring=status),
                        data=veteran, method.tte="Gehan",
                        method.inference = "permutation", n.resampling = 1e3)

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

```
  ## method.tte = "Peron"
  BT_Peron <- BuyseTest(trt ~ celltype + TTE(time,threshold=0,censoring=status),
                        data=veteran, method.tte="Peron",
                        method.inference = "permutation", n.resampling = 1e3)

  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

Details

It only affects the `BuyseTest` function

Examples

```
## 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.

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.

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*.

Examples

```
data(veteran,package="survival")

# 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.

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 'BuyseRes'
getCount(object, type)
```

Arguments

object	an R object of class BuyseRes, 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

getPairScore *Extract the Score of Each Pair*

Description

Extract the score of each pair.

Usage

```
getPairScore(object, endpoint = NULL, strata = NULL,
  rm.withinStrata = TRUE, rm.weight = FALSE, unlist = TRUE,
  trace = 1)

## S4 method for signature 'BuyseRes'
getPairScore(object, endpoint = NULL,
  strata = NULL, rm.withinStrata = TRUE, rm.weight = FALSE,
  unlist = TRUE, trace = 1)
```

Arguments

object	an R object of class BuyseRes, 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.
rm.withinStrata	[logical] should the columns indicating the position of each member of the pair within each treatment group be removed?
rm.weight	[logical] should the column weight be remove 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?
...	not used. For compatibility with the generic method.

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.

Examples

```
## run BuyseTest
data(veteran,package="survival")

BT.keep <- BuyseTest(trt ~ tte(time, threshold = 20, censoring = "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
pScore[91]

## retrive pair in the original dataset
pVeteran <- veteran[pScore[91,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))[[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[91, unfavorable])
   testthat::expect_equal(pN, pScore[91, neutral])
}
```

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

Arguments

`object`	an R object of class `BuyseRes`, 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`.
`...`	not used. For compatibility with the generic method.

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.

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, censoring, indexC, indexT, threshold, method, D,
  n_strata, n_TTE, n_UTTE, Wscheme, index_endpoint, index_censoring,
  index_UTTE, reanalyzed, list_survTimeC, list_survTimeT, list_survJumpC,
  list_survJumpT, list_lastSurv, correctionUninf, neutralAsUninf,
  keepScore, reserve, returnOnlyDelta)
```

Arguments

endpoint	A matrix containing the values of each endpoint (in columns) for each observation (in rows).
censoring	A matrix containing the values of the censoring variables relative to each endpoint (in columns) for each observation (in rows).
indexC	A list containing the indexes of control observations belonging for each strata.
indexT	A list containing the indexes of treatment observations belonging for each strata.
threshold	Store the thresholds associated to each endpoint. Must have length D. The threshold is ignored for binary endpoints. Must have D columns.
method	The index of the method used to score the pairs. Must have length D. 1 for continuous, 2 for Gehan, and 3 for Peron.
D	The number of endpoints.
n_strata	The number of strata.
n_TTE	The number of time-to-event endpoints.
n_UTTE	The number of unique time-to-event endpoints.

Wscheme	The matrix describing the weighting strategy. For each endpoint (except the first) in column, weights of each pair are initialized at 1 and multiplied by the weight of the endpoints in rows where there is a 1. Must have D lines and D columns.
index_endpoint	The position of the endpoint at each priority in the argument endpoint. Must have length D.
index_censoring	The position of the censoring at each priority in the argument censoring. 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.
reanalyzed	Will this endpoint be re-analyzed latter with a different threshold.
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).
correctionUninf	Should the uninformative weight be re-distributed to favorable and unfavorable?
neutralAsUninf	Should paired classified as neutral be re-analyzed using endpoints of lower priority?
keepScore	Should the result of each pairwise comparison be kept?
reserve	Should vector storing neutral pairs and uninformative pairs be initialized at their maximum possible length?
returnOnlyDelta	Should only the net benefit and win ratio be output?.

---

iid *Extract the H-decomposition of the Estimator*

Description

Extract the H-decomposition of the GPC estimator.

Usage

```
iid(object, endpoint = NULL, order = 1)

## S4 method for signature 'BuyseRes'
iid(object, endpoint = NULL, order = 1)
```

Arguments

object	an R object of class BuyseRes, 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.
order	[integer 1,2] which order of the H-decomposition should be output? Can be the first or second order. NULL output a list containing all terms.

See Also

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

---

powerBuyseTest *Performing simulation studies with BuyseTest*

Description

Performs a simulation studies for several sample sizes. Returns estimates, standard errors, confidence intervals and p.values.

Usage

```
powerBuyseTest(sim, sample.size, sample.sizeC = NULL,
  sample.sizeT = NULL, n.rep, null = c(0, 1), cpus = 1,
  alternative = NULL, seed = 10, conf.level = 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] the null hypothesis to be tested for the net benefit (first element) and the win ratio (second element).
cpus	[integer, >0] the number of CPU to use. Only the permutation test can use parallel computation. Default value read from BuyseTest.options().
alternative	[character] the alternative hypothesis. Must be one of "two.sided", "greater" or "less". Default value read from BuyseTest.options().

`seed`	[integer, >0] the seed to consider for the simulation study.
`conf.level`	[numeric] confidence level for the confidence intervals. Default value read from `BuyseTest.options()`.
`order.Hprojection`	[integer 1,2] the order of the H-project to be used to compute the asymptotic variance.
`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?
`...`	parameters from `BuyseTest`.

Examples

```
## using simBuyseTest
powerBuyseTest(sim = simBuyseTest, sample.size = c(100), n.rep = 2,
               formula = Treatment ~ tte(eventtime, censoring = status),
               method.inference = "asymptotic", trace = 4)

## using user defined simulation function
simFCT <- function(n.C, n.T){
    out <- data.table(Y=rnorm(n.C+n.T),
                      T=c(rep(1,n.C),rep(0,n.T))
                     )
return(out)
}

powerBuyseTest(sim = simFCT, sample.size = c(100), n.rep = 2,
               formula = T ~ cont(Y), method.inference = "asymptotic", trace = 4)
```

`Simulation function` *Simulation of data for the BuyseTest*

Description

Simulate binary, continuous or time to event data, possibly with strata. Outcomes are simulated independently of each other and independently of the strata variable.

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 binary 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

This function is built upon the lvm and sim functions from the lava package.

Arguments in the list argsBin:

- p.T probability of event of each endpoint (binary endpoint, treatment group).
- p.C same as p.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:

- rates.T hazard corresponding to each endpoint (time to event endpoint, treatment group).

- `rates.C` same as `rates.T` but for the control group.
- `rates.Censoring` Censoring same as `rates.T` but for the censoring.
- `name` names of the time to event variables.
- `nameCensoring` names of the event type indicators.

Examples

```
n <- 1e2

#### default option ####
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 binary endpoints ####
args <- list(p.T = c(3:5/10))
simBuyseTest(n, argsBin = args, argsCont = NULL, argsTTE = NULL)

#### only continuous endpoints ####
args <- list(mu.T = c(3:5/10), sigma.T = rep(1,3))
simBuyseTest(n, argsBin = NULL, argsCont = args, argsTTE = NULL)

#### only TTE endpoints ####
args <- list(rates.T = c(3:5/10), rates.Censoring = rep(1,3))
simBuyseTest(n, argsBin = NULL, argsCont = NULL, argsTTE = args)
```

Index