Package: BuyseTest (via r-universe)

October 24, 2026

Type Package

Title Generalized Pairwise Comparisons

Version 3.3.9

Date 2026-06-01

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/worse/equivalently than a random observation drawn from the other group. Summary statistics such as the net treatment benefit, win ratio, or win odds are then deduced from these probabilities. 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 (>= 3.5.0), Rcpp

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

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

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder R.rsp

NeedsCompilation yes

RoxygenNote 7.3.3

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' 'CasinoTest.R' 'PairScore.R' 'RcppExports.R' 'S4-BuysePower.R' 'S4-BuysePower-model.tables.R' 'S4-BuysePower-nobs.R' 'S4-BuysePower-summary.R' 'S4-BuysePower-print.R' 'S4-BuysePower-show.R' 'S4-BuyseTest.R' 'S4-BuyseTest-coef.R' 'S4-BuyseTest-confint.R' 'S4-BuyseTest-get.R' 'S4-BuyseTest-model.tables.R' 'S4-BuyseTest-nobs.R' 'S4-BuyseTest-plot.R' 'S4-BuyseTest-summary.R' 'S4-BuyseTest-print.R' 'S4-BuyseTest-sensitivity.R' 'S4-BuyseTest-show.R' 'S4-BuyseTest-update.R' 'S4-BuyseTest.options.R' 'S4-BuyseTest.vcov.R' 'as.data.table.performance.R' 'auc.R' 'autoplot.S4BuyseTest.R' 'brier.R' 'constStrata.R' 'discreteRoot.R' 'doc-data.R' 'efronlim.R' 'iid.S3sensitivity.R' 'iid.prodlim.R' 'mover.R' 'normexp.R' 'performance.R' 'performanceResample.R' 'plot.S3sensitivity.R' 'powerBuyseTest.R' 'predict.logit.R' 'rbind.performanceResample.R' 'sim.simBuyseTest.R' 'simBuyseTest.R' 'simCompetingRisks.R' 'summary.performance.R' 'valid.R'

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

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

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

Date/Publication 2026-06-01 10:50:02 UTC

RemoteUrl https://github.com/cran/BuyseTest

RemoteRef HEAD

RemoteSha 6d52fd25177f025ae54bb589b64e9a903bb65d4c

Contents

BuyseTest-package 4
.colCenter_cpp 6
.colCumSum_cpp 7
.colMultiply_cpp 7
.colScale_cpp 8
.rowCenter_cpp 8
.rowCumProd_cpp 9
.rowCumSum_cpp 9

.rowMultiply_cpp . . . 10
.rowScale_cpp . . . 10
as.data.table.performance . . . 11
auc . . . 11
autoplot.S4BuyseTest . . . 13
brier . . . 14
BuyseMultComp . . . 15
BuyseTest . . . 17
BuyseTest.options . . . 26
BuyseTest.options-class . . . 27
BuyseTest.options-methods . . . 27
BuyseTTEM . . . 28
CasinoTest . . . 31
CHARM . . . 33
coef.BuyseTestAuc . . . 34
coef.BuyseTestBrier . . . 34
confint.BuyseTestAuc . . . 35
confint.BuyseTestBrier . . . 35
constStrata . . . 36
EB . . . 37
efronlim . . . 38
getCount . . . 39
getIid . . . 39
getMatrixScore . . . 41
getPairScore . . . 43
getPseudovalue . . . 46
getSurvival . . . 47
iid.BuyseTestAuc . . . 48
iid.BuyseTestBrier . . . 49
iid.prodlim . . . 49
mover . . . 50
performance . . . 51
performanceResample . . . 54
plot.S3sensitivity . . . 55
powerBuyseTest . . . 56
predict.BuyseTTEM . . . 60
prodige . . . 61
profil . . . 62
rbind.performance . . . 63
S4BuysePower-class . . . 64
S4BuysePower-model.tables . . . 64
S4BuysePower-nobs . . . 65
S4BuysePower-print . . . 66
S4BuysePower-show . . . 67
S4BuysePower-summary . . . 67
S4BuyseTest-class . . . 69
S4BuyseTest-coef . . . 69
S4BuyseTest-confint . . . 71

S4BuyseTest-model.tables . . . 73
S4BuyseTest-nobs . . . 75
S4BuyseTest-plot . . . 75
S4BuyseTest-print . . . 77
S4BuyseTest-summary . . . 77
S4BuyseTest-update . . . 80
S4BuyseTest-vcov . . . 81
sensitivity . . . 82
simBuyseTest . . . 84
simCompetingRisks . . . 88
summary.performance . . . 90

Index **92**

`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.

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.

Keywords: documented methods/functions are classified according to the following keywords

- models: function fitting a statistical model/method based on a dataset (e.g. `auc`, `brier`, `BuyseTest`, `BuyseTTEM`, `CasinoTest`, `performance`)
- htest: methods performing statistical inference based on an existing model (e.g. `BuyseMultComp`, `performanceResample`, `powerBuyseTest`, `sensitivity`)
- methods: extractors (e.g. `getCount`, `getPairScore`, `getPseudovalue`, `getSurvival`, `getIid`)
- print: concise display of an object in the console (e.g. `print`, `summary`)
- utilities: function used to facilitate user interactions (e.g. `BuyseTest.options`, `constStrata`)
- hplot: graphical display (e.g. `autoplot.S3sensitivity`)

- internal: function used internally but that need to be exported for parallel calculations (e.g. GPC_cpp)
- datagen: function for generating data sets (e.g. simBuyseTest, simCompetingRisks)
- classes: definition of S4 classes

Author(s)

Maintainer: Brice Ozenne <brice.mh.ozenne@gmail.com> (ORCID)

Authors:

- Eva Cantagallo
- William Anderson

Other contributors:

- Julien Peron [contributor]
- Johan Verbeeck [contributor]

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 stratified win ratio: G. Dong et al. (2018). **The stratified win ratio**. *Journal of biopharmaceutical statistics*. 28(4):778-796
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.
On how using a restriction time: Piffoux M, Ozenne B, De Backer M, Buyse M, Chiem JC, Péron J (2024). **Restricted Net Treatment Benefit in oncology**. *Journal of Clinical Epidemiology*. Jun;170:111340.

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.

Discussion about the relevance of GPC based 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*. Ezimamaka Ajufo, Aditi Nayak, Mandeep R. Mehra (2023). **Fallacies of Using the Win Ratio in Cardiovascular Trials: Challenges and Solutions**. *JACC: Basic to Translational Science*. 8(6):720-727.

See Also

Useful links:

- https://github.com/bozenne/BuyseTest
- Report bugs at https://github.com/bozenne/BuyseTest/issues

.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.

Value

A data.table object

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

An S3 object of class `BuyseTestAUC` that inherits from data.frame. The last line of the object 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.S4BuyseTest *Graphical Display for GPC*

Description

Graphical display of the percentage of favorable, unfavorable, neutral, and uninformative pairs per endpoint.

Usage

```
## S3 method for class 'S4BuyseTest'
autoplot(
  object,
  type = "hist",
```

```
  strata = "global",
  endpoint = NULL,
  label.strata = NULL,
  label.endpoint = NULL,
  color = c("#7CAE00", "#F8766D", "#C77CFF", "#00BFC4"),
  ...
)
```

Arguments

`object`	an R object of class `S4BuyseTest`, i.e., output of `BuyseTest`
`type`	[character] type of plot: histogram (`"hist"`), pie chart (`"pie"`), or nested pie charts (`"racetrack"`).
`strata`	[character vector] strata(s) relative to which the percentage should be displayed.
`endpoint`	[character vector] endpoint(s) relative to which the percentage should be displayed.
`label.strata`	[character vector] new labels for the strata levels. Should match the length of argument `strata`.
`label.endpoint`	[character vector] new labels for the endpoints. Should match the length of argument `endpoint`.
`color`	[character vector] colors used to display the percentages for each type of pair.
`...`	not used, for compatibility with the generic function.

Value

a ggplot object.

---

brier *Estimation of the Brier Score (EXPERIMENTAL)*

Description

Estimation of the brier score, possibly after cross validation, to assess the discriminant ability and calibration of a biomarker regarding a disease status.

Usage

```
brier(
  labels,
  predictions,
  iid = NULL,
  fold = NULL,
  observation = NULL,
  null = NA,
  conf.level = 0.95,
  transformation = TRUE
)
```

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.
`iid`	[array, optional] influence function of the prediction. For cross validation (CV) should be a 3 dimensional array (one slice per CV fold). Otherwise a matrix with as many column as observations and rows as predictions.
`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.
`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.

Value

An S3 object of class `BuyseTestBrier` that inherits from data.frame.

`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"`, `"winRatio"`, `"favorable"`, `"unfavorable"`. See the documentation of the `coef` method for further details. 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.

Value

An S3 object of class `BuyseMultComp`.

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
if(require(lava)){
cor(lava::iid(confintAdj)) ## 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
if(require(lava)){
cor(lava::iid(confintAdj))
}
```

`BuyseTest` *Two-group GPC*

Description

Performs Generalized Pairwise Comparisons (GPC) between two groups. Can handle one or several 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 `type1(endpoint1) + type2(endpoint2, threshold2) ~ treatment + strata(gender)`. 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"`, `"Peron"`, `"Efron"`, or `"Latta"`. See Details, section "Handling missing values".
pool.strata	[character] weights used to combine estimates across strata. Can be `"Buyse"` to weight proportionally to the number of pairs in the strata, `"CMH"` to weight proportionally 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, `"standardisation"` to recover a marginal comparison, 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", "varExact permutation". 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 pairs 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] Random number generator (RNG) state used when starting resampling. If NULL no state 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` with:

- on one side of the ~ symbol the endpoints by order of priority, surrounded by parentheses and a character string indicating the type of variable (binary,continuous,timetoevent). Additional argument may be included in the parenthesis: threshold of clinical relevance (`threshold`), restriction time for time to event endpoints (`restriction`), ...
- in the other side a binary variable defining the two treatment arms.
- the right-hand side of the formula may also contain a strata variable(s) surrounded by parenthese and the character strata. The argument `match` should be set to `TRUE` when strata are independent but not observations (e.g. cross-over trial with strata as patients).

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 and type of threshold `">0"` indicates that higher values are favorable while `"<0"` indicates that lower values are favorable. `"+"` indicates an additive threshold while `"*"` a multiplicative threshold. Can be combined as `"*<0"` to use a multiplicative threshold and lower values as favorable.
- `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 and left-censoring is only implemented with the Gehan's scoring rule.
- `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/Efron's/Latta's scoring rule. To export the classification of each pair set the argument `keep.pairScore` to `TRUE` and call the function `getPairScore` on the result of the `BuyseTest` function.

Handling of right-censoring: the recommended default approach to handle administrative censoring is to use a restriction time. Alternatively one can use the argument `correction.uninf` to 1 or 2 to re-classify the uninformative score obtained by the scoring rule at given endpoint:

- `correction.uninf=0`: if there is any lower priority endpoint and the argument `neutral.as.uninf` is `TRUE`, the uninformative score is re-analyzed using the lower priority endpoint. Otherwise it counts as 0, i.e. neither favorable nor unfavorable. The former makes the estimand dependent on the censoring distribution while the later typically bias estimates toward the null.
- `correction.uninf=1`: update the favorable, unfavorable, and neutral score of each pair with uninformative score by redistributing this uninformative score in favorable, unfavorable, and neutral score according to the average score among the informative pairs. This can be seen as imputing the uninformative into an informative score under a proportional hazard assumption and as an extension of Harrell's scoring rule compatible with multiple endpoints.
- `correction.uninf=2`: This is equivalent to Harrell's scoring rule in that it re-weight informative pairs to represent uninformative pairs, essentially disregarding the uninformative pairs. Can also be seen as handling the uniformative under a proportional hazard assumption but is not compatible with lower priority endpoints as the neutral score will be re-analysed only based on the informative pairs with neutral score (which may not be representative of the uninformative one for lower priority endpoints).

Several scoring rules are available to handle drop-out via the argument `scoring.rule`, the recommended one being the Peron's scoring rule:

- `scoring.rule="Gehan"`: When no observations is censored or only the pair with the largest timepoint is censored, the pair is decidedly classified as favorable, unfavorable, or neutral. Otherwise pairs are classified as uninformative.
- `scoring.rule="Peron"` score pairs involving censored observations using Kaplan Meier survival curves stratified on treatment (and strata variables if any). It may still lead to uninformative pairs when the survival curve is only partially known.

- scoring.rule="Efron": same as the Peron's scoring rule except that the survival curve is extrapolated to 0 when its tail is unknown. Only relevant when using a (stratified) Kaplan-Meier estimator and no competing risks.
- scoring.rule="Latta": same as the Peron's scoring rule except that Kaplan Meier survival curves are common for both treatment groups (but are still stratified on strata variable if any).

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:

- method.inference="none": only the point estimate is computed which makes the execution of the BuyseTest faster than with the other methods.
- method.inference="u-statistic": H-projection of order 1 or 2 to compute the variance (1 is the default ,see BuyseTest.options). The first order is downward biased but consistent. When considering the Gehan scoring rule, no transformation nor correction, the second order is unbiased and equivalent to the variance of the (stratified) bootstrap distribution. P-values and confidence intervals are then evaluated assuming that the estimates follow a Gaussian distribution. In presence of strata, the weights used to combine estimates across strata are assumed known. This is the case the type of weights used is pre-defined and the covariate distribution is fixed by design as in a blocked randomized trial. **WARNING**: the current implementation of the H-projection is not valid when using corrections for uninformative pairs (correction.uninf=1, or correction.uninf=2).
- method.inference="permutation": permutation test, estimating in each sample the summary statistics (e.g. net treatment benefit).
- method.inference="studentized permutation": permutation test, estimating in each sample the summary statistics (e.g. net treatment benefit) and the variance-covariance matrix of the estimate.
- method.inference="varExact permutation": closed-form variance formula under the permutation distribution (Anderson and Verbeeck 2023). P-values and confidence intervals are then evaluated assuming that the estimates follow a Gaussian distribution. **WARNING**: the current implementation of the variance estimator for the permutation distribution is not valid when using the Peron scoring rule or corrections for uninformative pairs.
- method.inference="bootstrap": non-parametric boostrap, estimating in each sample the summary statistics (e.g. net treatment benefit).
- method.inference="studentized bootstrap": non-parametric boostrap, estimating in each sample the summary statistics (e.g. net treatment benefit) 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.

- seed: the seed is used to generate one seed per sample. These seeds are the same whether one or several CPUs are used.
- 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
Consider K strata and denote by m_k and n_k the sample size in the control and active arm (respectively) for strata k. Let σ_k be the standard error of the strata-specific summary statistic (e.g. net benefit). The strata specific weights, w_k, are given by:

- "CMH": $w_k = \frac{\frac{m_k \times n_k}{m_k + n_k}}{\sum_{l=1}^{K} \frac{m_l \times n_l}{m_l + n_l}}$. Optimal if the if the odds ratios are constant across strata.
- "equal": $w_k = \frac{1}{K}$
- "Buyse": $w_k = \frac{m_k \times n_k}{\sum_{l=1}^{K} m_l \times n_l}$. Optimal if the risk difference is constant across strata
- "var-*" (e.g. "var-netBenefit"): . $w_k = \frac{1/\sigma_k^2}{\sum_{l=1}^{K} 1/\sigma_k^2}$

Only when using "var-winRatio", the pooled Win Ratio is computed by pooling the strata-specific win-ratios. Otherwise the pooled Win Ratio is obtained by dividing the pooled number of favorable pairs divided by the pooled number of unfavorable pairs, possibly adding half the pooled neutral pairs, according to formula (1) in Dong et al. (2018).

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 stratified win ratio: G. Dong et al. (2018). **The stratified win ratio**. *Journal of biopharmaceutical statistics*. 28(4):778-796
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 Peron's scoring rule with competing risks: E. Cantagallo, M. De Backer, M. Kicinski, B. Ozenne, L Collette, C. Legrand, M. Buyse, J. Péron. **A new measure of treatment effect in clinical trials involving competing risks based on generalized pairwise comparisons**. *Biometrical Journal* 2021 Feb;63(2):272-288. doi: 10.1002/bimj.201900354. 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 using a restriction time: Piffoux M, Ozenne B, De Backer M, Buyse M, Chiem JC, Péron J (2024). **Restricted Net Treatment Benefit in oncology**. *Journal of Clinical Epidemiology*. Jun;170:111340.
On the argument `correction.uninf`: 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.
On closed-form formula for permutation variance: W.N. Anderson and J. Verbeeck (2023). **Exact Permutation and Bootstrap Distribution of Generalized Pairwise Comparisons Statistics**. *Mathematics* , 11, 1502. doi:10.3390/math11061502.

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-model.tables` for exporting the number or percentage of favorable/unfavorable/neutral/uninformative pairs.
`sensitivity` for performing a sensitivity analysis on the choice of the threshold(s).
`S4BuyseTest-plot` for graphical display of the pairs across endpoints.
`getIid` for exporting the first order H-decomposition.
`getPairScore` and `getMatrixScore` for exporting the score of each pair in a data.table or matrix format.

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
model.tables(BT) ## export the table at the end of summary
summary(BT, percentage = FALSE)
summary(BT, statistic = "winRatio") ## win Ratio

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

## End(Not run)

summary(BTperm)
summary(BTperm, statistic = "winRatio")

## same with parallel calculations
## Not run:
    BTperm <- BuyseTest(treatment ~ TTE(eventtime, status = status), data=df.data,
                    method.inference = "permutation", n.resampling = 1e3, cpus = 8)
    summary(BTperm)

## 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
BTS <- BuyseTest(treatment ~ strata + TTE(eventtime, status = status), data=df.data)
summary(BTS)

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

BTM <- BuyseTest(ff, data=df.data)
summary(BTM)
plot(BTM)
```

```
#### 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:
  data(cancer, package = "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")

  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 = NULL, iid, ...)

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

Arguments

object	time to event model.
...	For compatibility with the generic method.
treatment	[list or character] List containing the value of the treatment variable for each individual or name of the treatment variable. The former is necessary when the survival model do not dependent on treatment.
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.

Value

An S3 object of class BuyseTTEM.

Examples

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

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

########################
#### survival case ####
########################

set.seed(10)
df.data <- simBuyseTest(1e2, n.strata = 2)
df.Ta <- data.frame(treatment = "T", strata = "a")
df.Ca <- data.frame(treatment = "C", strata = "a")

#### Non-parametric model (stratified Kapan-Meier) ####

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

## convert to TTEM object
eKM.TTEM <- BuyseTTEM(eKM.prodlim, treatment = "treatment", iid = TRUE)

## extract jump times
jumpKM.Ta <- predict(eKM.TTEM, type = "jump", treatment = "T", strata = "a")
jumpKM.Ta
## extract last observation time
last.timeKM.Ta <- predict(eKM.TTEM, type = "last.time", treatment = "T", strata = "a")
last.timeKM.Ta
## extract survival at last observation time
last.survKM.Ta <- predict(eKM.TTEM, type = "last", treatment = "T", strata = "a")
last.survKM.Ta
## extract jump size
changeKM.Ta <- predict(eKM.TTEM, type = "change", treatment = "T", strata = "a")
last.survKM.Ta-sum(changeKM.Ta$survival) ## sum to 1
## prediction
predict(eKM.TTEM, time = tau, treatment = "T", strata = "a")
## same as prodlim: predict(eKM.prodlim, times = tau, newdata = df.Ta)
predict(eKM.TTEM, time = tau, treatment = "C", strata = "a")
## fix bug in prodlim: predict(eKM.prodlim, times = tau, newdata = df.Ca)
## survival should be 0 if the last observation is an event

#### Parametric model ####
dist <- "lognormal"

## estimate
eWei.survreg <- survreg(Surv(eventtime,status)~treatment+strata, data = df.data,
                        dist = dist)
## gridCDF <- seq(.01,.99,by=.01)
## plot(x = predict(eWei.survreg, newdata = df.Ta, type = "quantile", p = gridCDF,
```

```
##                  y = rev(gridCDF), col = "red", type = "l")
## points(x = predict(eWei.survreg, newdata = df.Ca, type = "quantile", p = gridCDF,
##                  y = rev(gridCDF), col = "blue", type = "l")

## conver to TTEM object
eWei.TTEM <- BuyseTTEM(eWei.survreg, treatment = "treatment", iid = TRUE, n.grid = 1e3)

## extract jump times
jumpWei.Ta <- predict(eWei.TTEM, type = "jump", treatment = "T", strata = "a")
jumpWei.Ta
## extract last observation time
last.timeWei.Ta <- predict(eWei.TTEM, type = "last.time", treatment = "T", strata = "a")
last.timeWei.Ta
## extract survival at last observation time
last.survWei.Ta <- predict(eWei.TTEM, type = "last", treatment = "T", strata = "a")
last.survWei.Ta
## extract jump size
changeWei.Ta <- predict(eWei.TTEM, type = "change", treatment = "T", strata = "a")
last.survWei.Ta-sum(changeWei.Ta$survival) ## sum to 1
## prediction
predWei.Ta <- predict(eWei.TTEM, time = tau, treatment = "T", strata = "a")
predWei.Ta
## survreg: predict(eWei.survreg, newdata = df.Ta, type = "quantile", p = 1-predWei.Ta$survival)-tau
## approximate retrieve tau, up to the grid resolution

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

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

eAJ.TTEM <- BuyseTTEM(eAJ.prodlim, treatment = "treatment", iid = TRUE)

## extract jump times
jumpAJ.Ta <- predict(eAJ.TTEM, type = "jump", treatment = "T", strata = "a")
jumpAJ.Ta
## extract last observation time
last.timeAJ.Ta <- predict(eAJ.TTEM, type = "last.time", treatment = "T", strata = "a")
last.timeAJ.Ta
## extract survival at last observation time
last.cifAJ.Ta <- predict(eAJ.TTEM, type = "last", treatment = "T", strata = "a")
last.cifAJ.Ta
## extract jump size
changeAJ.Ta <- predict(eAJ.TTEM, type = "change", treatment = "T", strata = "a")
last.cifAJ.Ta-sum(changeAJ.Ta$cif) ## sum to 0
## prediction
predict(eAJ.TTEM, time = tau, treatment = "T", strata = "a")
## same as prodlim: predict(eAJ.prodlim, times = tau, newdata = df.Ta, cause = 1)

predict(eAJ.TTEM, time = tau, treatment = "C", strata = "a")
## same as prodlim: predict(eAJ.prodlim, times = tau, newdata = df.Ca, cause = 1)
```

CasinoTest *Multi-group GPC (EXPERIMENTAL)*

Description

Perform Generalized Pairwise Comparisons (GPC) for two or more groups. Can handle one or several binary, continuous and time-to-event endpoints.

Usage

```
CasinoTest(
  formula,
  data,
  statistic = NULL,
  method.inference = NULL,
  type = "unweighted",
  conf.level = NULL,
  transformation = NULL,
  alternative = NULL,
  method.multcomp = "none",
  seed = NA,
  ...
)
```

Arguments

formula	[formula] a symbolic description of the GPC model, see the BuyseTest function
data	[data.frame] dataset.
statistic	[character] the statistic summarizing the pairwise comparison: "netBenefit" (Net Treatment Benefit) or "favorable" (Mann-Withney parameter). Default value read from BuyseTest.options().
method.inference	[character] method used to compute confidence intervals and p-values. Can be "none", "u-statistic", or "rank". Default value read from BuyseTest.options().
type	[character] Type of estimator: can be "unweighted" or "weighted".
conf.level	[numeric] confidence level for the confidence intervals. Default value read from BuyseTest.options().
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.
alternative	[character] the type of alternative hypothesis: "two.sided", "greater", or "less". Default value read from BuyseTest.options().

method.multcomp
: [character] method used to adjust for multiple comparisons. Can be any element of 'p.adjust.methods' (e.g. "holm"), "maxT-integration", or "maxT-simulation".

seed
: [integer, >0] Random number generator (RNG) state used when adjusting for multiple comparisons. If NULL no state is set.

...
: arguments passed to BuyseTest

Details

Require to have installed the package riskRegression and BuyseTest

Setting argument method.inference to "rank" uses a U-statistic approach with a small sample correction to match the variance estimator derived in Result 4.16 page 228 of Brunner (2018).

Value

An S3 object of class CasinoTest that inherits from data.frame.

References

Edgar Brunner, Arne C Bathke, and Frank Konietschke (2018). **Rank and pseudo-rank procedures for independent observations in factorial designs**. Springer.

Examples

```
library(data.table)
library(BuyseTest)

#### simulate data ####
set.seed(11)
n <- 10
dt <- rbind(data.table(score = rnorm(n), group = "A"),
            data.table(score = rnorm(2*n), group = "B"),
            data.table(score = rnorm(3*n), group = "C"))
dt$index <- 1:NROW(dt)

#### estimation ####
score.casino <- dt$score

## naive casino (by hand)
M.score <- outer(dt[group=="A",score],score.casino,function(x,y){(x>y)+0.5*(x==y)})
mean(M.score) ## 0.4766667

## naive casino (via BuyseTest)
CasinoTest(group ~ cont(score), data = dt, type = "weighted", statistic = "favorable")
CasinoTest(group ~ cont(score), data = dt, type = "weighted", statistic = "netBenefit")

## harmonic casino (by hand)
hweight <- unlist(tapply(dt$group, dt$group, function(x){rep(1/length(x),length(x))}))
M.scoreW <- sweep(M.score, MARGIN = 2, FUN = "*", STATS = NROW(dt)*hweight/3)
mean(M.scoreW) ## 0.3680556
```

```
## harmonic casino (via BuyseTest)
CasinoTest(group ~ cont(score), data = dt, type = "unweighted", statistic = "favorable")

#### Relative liver weights data (Brunner 2018, table 4.1, page 183) ####
liverW <- rbind(
  data.frame(value = c(3.78, 3.40, 3.29, 3.14, 3.55, 3.76, 3.23, 3.31),
             group = "Placebo"),
  data.frame(value = c(3.46,3.98,3.09,3.49,3.31,3.73,3.23),
             group = "Dose 1"),
  data.frame(value = c(3.71, 3.36, 3.38, 3.64, 3.41, 3.29, 3.61, 3.87),
             group = "Dose 2"),
  data.frame(value = c(3.86,3.80,4.14,3.62,3.95,4.12,4.54),
             group = "Dose 3"),
  data.frame(value = c(4.14,4.11,3.89,4.21,4.81,3.91,4.19, 5.05),
             group = "Dose 4")
)
liverW$valueU <- liverW$value + (1:NROW(liverW))/1e6

## same as table 4.1, page 183 in Brunner et al (2018)
CasinoTest(group ~ cont(value), data = liverW, type = "weighted", statistic = "favorable")
CasinoTest(group ~ cont(valueU), data = liverW, type = "unweighted", statistic = "favorable")
```

CHARM *RCT In Chronic Heart Failure Assessing an Inhibitor of the Renin-Angiotensin System.*

Description

Simulated data inspired from the CHARM-Preserved Trial comparing cardiovascular death or admission to hospital for chronic heart failure (CHF) among patients with CHF treated with candesartan or placebo.

- `id`: study participant.
- `treatment`: treatment arm: candesartan (T) or placebo (C).
- `Mortality`: time from inclusion (say diagnosis) to death or loss to follow-up.
- `statusMortality`: event triggering the end of follow-up: death (1), drop-out (0).
- `Hospitalization`: time from inclusion (say diagnosis) to hospitalization or end of follow-up.
- `statusHospitalization`: hospitalization (1) or end of follow-up (0).
- `Composite`: time to hospitalization, death, or loss to follow-up, whichever comes first.
- `statusComposite`: hospitalization or death (1), drop-out (0).

Usage

```
data(CHARM, package = "BuyseTest")
```

Format

An object of class `data.frame` with 3023 rows and 8 columns.

Author(s)

Johan Verbeeck

References

Yusuf Salim, et al. "Effects of candesartan in patients with chronic heart failure and preserved left-ventricular ejection fraction: the CHARM-Preserved Trial". The Lancet (2003) 9386(362):777-781.

---

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
```

EB *Rare disease trial*

Description

16 pediatric subjects suffering from a aare skin disease (epidermolysis bullosa simplex) treated with placebo or diacerin cream in a longitudinal cross-over trial (14 paired).

- `Id`: study participant.
- `Time`: end of the 4 weeks treatment period (t4) or 12 weeks after treatment ended (t12)
- `treatment`: treatment arm: diacerin cream (V) or placebo (P).
- `StdDiffCount`: blister uncertainty
- `Bin`: indicator of 40
- `DiffQoL`: Change in quality of life since baseline.
- `period`: same as time but coded as numeric: 1 for t4 and 2 for t12

Usage

```
data(CHARM, package = "BuyseTest")
```

Format

An object of class `data.frame` with 30 rows and 7 columns.

References

Wally et al. "Diacerein orphan drug development for epidermolysis bullosa simplex: A phase 2/3 randomized, placebo-controlled, double-blind clinical trial". Journal of American Academy of Dermatology (2018) 78(5):892-901. https://doi.org/10.1016/j.jaad.2018.01.019.

efronlim *Constrained Kaplan-Meier Estimator*

Description

Kaplan-Meier estimator, possibly stratified, constrained to 0 just after end of follow-up in each strata.

Usage

```
efronlim(formula, data, ...)
```

Arguments

formula	formula with on the right-hand side Hist(time,event) or Surv(time,event) and 1 or stratification variable(s) on the right-hand side.
data	A data.frame containing the variables of present in argument formula.
...	additional arguments passed to prodlim::prodlim

Details

If in any strata there is censoring at the observed time, then the dataset is updated by setting one of the censored observations to an infinitesimal later timepoint with an event instead of censoring. Then the possibly stratified Kaplan-Meier estimator is run on this updated dataset.

Value

A prodlim object.

Examples

```
library(data.table)

set.seed(1)
dt.data <- simBuyseTest(1e2, n.strata = 2)
dt.data$time1 <- pmin(dt.data$eventtime, 1)
dt.data$status1 <- dt.data$status * (dt.data$eventtime<=1)

## KM
if(require(prodlim)){
   e.KM <- prodlim(Hist(time1,status1)~1, data = dt.data)
   plot(e.KM)
}

e.KM0 <- efronlim(Hist(time1,status1)~1, data = dt.data)
plot(e.KM0)

## stratfied KM
```

```
if(require(prodlim)){
   e.KMS <- prodlim(Hist(time1,status1)~strata, data = dt.data)
   plot(e.KMS)
}

e.KMS <- efronlim(Hist(time1,status1)~strata, data = dt.data)
plot(e.KMS)
```

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

Description

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

Usage

```
## 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

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

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] the statistic relative to which the H-decomposition should be output: "netBenefit", "winRatio", "favorable", "unfavorable". See the documentation of the coef method for further details. Default value read from BuyseTest.options().
strata	[character vector] the strata relative to which the H-decomposition of the statistic should be output. Can also be "global" or FALSE to output the H-decompositon of the pooled statistic. or TRUE to output the H-decompositon of each strata-specific statistic.
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.
simplify	[logical] should the result be coerced to the lowest possible dimension?

Details

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

Value

A list of matrices, each element of the list correspond to a statistic (global or strata-specific) and each matrix has as many columns as endpoints and rows as observations.

Author(s)

Brice Ozenne

See Also

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

---

getMatrixScore *Extract the Score Matrix*

Description

Extract all pairwise comparisons in a matrix format.

Usage

```
## S4 method for signature 'S4BuyseTest'
getMatrixScore(
  object,
  endpoint = NULL,
  statistic = NULL,
  strata = NULL,
  cumulative = TRUE,
  within = FALSE,
  unlist = TRUE
)
```

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.
statistic	[character] the statistic summarizing the pairwise comparison: "netBenefit", "favorable", "unfavorable", "neutral", "uninf".
strata	[character vector] the strata relative to which the score should be output.
cumulative	[logical] should the scores be cumulated over endpoints?
within	[logical] should within group pairwise comparisons be exported? This requires to re-run GPC which can be time and memory consuming.
unlist	[logical] should the structure of the output be simplified when possible?

Author(s)

Brice Ozenne

Examples

```
library(data.table)
library(prodlim)
library(survival) ## import veteran

#### Gehan scoring rule ####
## non stratified GPC
e.Gehan <- BuyseTest(trt ~ tte(time, threshold = 20, status = "status"),
                     data = veteran, scoring.rule = "Gehan", keep.pairScore = TRUE)
Mscore.Gehan <- getMatrixScore(e.Gehan)
mean(Mscore.Gehan) - coef(e.Gehan) ## same
m <- NROW(Mscore.Gehan)
n <- NCOL(Mscore.Gehan)

var1.Gehan <- sum((colSums(Mscore.Gehan)/m - mean(Mscore.Gehan))^2)
var2.Gehan <- sum((rowSums(Mscore.Gehan)/n - mean(Mscore.Gehan))^2)
var1.Gehan/n^2 + var2.Gehan/m^2 - confint(e.Gehan)[,"se"]^2 ## same

MscoreFull.Gehan <- getMatrixScore(e.Gehan, within = TRUE)
table(MscoreFull.Gehan[1:m,m+(1:n)] - Mscore.Gehan) ## same
all(MscoreFull.Gehan[1:m,m+(1:n)] + t(MscoreFull.Gehan[m+(1:n),1:m])==0)

## stratified GPC
e.GehanS <- BuyseTest(trt ~ tte(time, threshold = 20, status = "status") + celltype,
                     data = veteran, scoring.rule = "Gehan", keep.pairScore = TRUE)
Mscore.GehanS <- getMatrixScore(e.GehanS)
Mscore.GehanS

Mscore.GehanS1 <- getMatrixScore(e.GehanS, strata = "adeno")
Mscore.GehanS1

MscoreFull.GehanS <- getMatrixScore(e.GehanS, within = TRUE)

#### Peron scoring rule ####
## non stratified GPC
e.Peron <- BuyseTest(trt ~ tte(time, threshold = 20, status = "status"),
                    data = veteran[veteran$celltype == "adeno",], keep.pairScore = TRUE)
Mscore.Peron <- getMatrixScore(e.Peron)
mean(Mscore.Peron) - coef(e.Peron) ## same
m <- NROW(Mscore.Peron)
n <- NCOL(Mscore.Peron)

var1.Peron <- sum((colSums(Mscore.Peron)/m - mean(Mscore.Peron))^2)
var2.Peron <- sum((rowSums(Mscore.Peron)/n - mean(Mscore.Peron))^2)
var1.Peron/n^2 + var2.Peron/m^2 - confint(e.Peron)[,"se"]^2 ## too low
## as ignore the uncertainty about the survival model

MscoreFull.Peron <- getMatrixScore(e.Peron, within = TRUE)
```

```
table(MscoreFull.Peron[1:m,m+(1:n)] - Mscore.Peron) ## same
all(MscoreFull.Peron[1:m,m+(1:n)] + t(MscoreFull.Peron[m+(1:n),1:m])==0)

MscoreFull.Peron[2,m+(1:n)]
Mscore.Peron[2,]

## stratified GPC
e.PeronS <- BuyseTest(trt ~ tte(time, threshold = 20, status = "status") + celltype,
                      data = veteran, keep.pairScore = TRUE)
Mscore.PeronS <- getMatrixScore(e.PeronS)
Mscore.PeronS

Mscore.PeronS1 <- getMatrixScore(e.PeronS, strata = "adeno")
Mscore.PeronS1

MscoreFull.PeronS <- getMatrixScore(e.PeronS, within = TRUE)
```

`getPairScore` *Extract the Score of Each Pair*

Description

Extract the score of each pair.

Usage

```
## S4 method for signature 'S4BuyseTest'
getPairScore(
  object,
  endpoint = NULL,
  strata = NULL,
  cumulative = FALSE,
  rm.withinStrata = TRUE,
  rm.strata = all(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`	[character vector] the strata relative to which the score should be output.
`cumulative`	[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?

Value

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

```
## 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 statistic relative to which the pseudovalues should be computed: "netBenefit", "winRatio", "favorable", "unfavorable". See the documentation of the coef method for further details. Default value read from BuyseTest.options().
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

```
## 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 relative to 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

---

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).

Value

A list containing:

- IFbeta: Influence function for the regression coefficient.
- IFhazard: Time differential of the influence function of the hazard.
- IFcumhazard: Influence function of the cumulative hazard.
- time: Times at which the influence function has been evaluated.
- etime.max: Last observation time (i.e. jump or censoring) in each strata.
- label.strata: Strata to which each observation belong.
- X: Design matrix.
- table: Hazard at each time for each strata.

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)
```

`mover` *Inference using MOVER in match designs (EXPERIMENTAL)*

Description

Implementation of the method of variance estimates recovery (MOVER), an alternative method for statistic inference in a matched design proposed by Matsouaka et al. (2022), for the Net Treatment Benefit.

Usage

```
mover(
  object,
  endpoint = NULL,
  conf.level = 0.95,
  p.value = TRUE,
  type = "Wilson",
  tol = 1e-06
)
```

Arguments

object	R object of class S4BuyseTest, i.e., output of BuyseTest
endpoint	[character] for which endpoint the confidence intervals should be output?
conf.level	[numeric] confidence level for the confidence intervals.
p.value	[logical] should the confidence interval be inverted to obtain a p-value.
type	[character] method used to compute confidence intervals for a proportion: "Wilson" or "Agresti-Coull".
tol	[numeric, >0] numeric tolerance for what is considered neglectable (i.e. 0).

Details

The p-value calculation is performed by finding the confidence level such that the upper or lower bound of the corresponding confidence interval is 0. This was not part of the methodology presented in the original paper.

References

R A Matsouaka, A Coles (2022). **Robust statistical inference for the matched net benefit and the matched win ratio using prioritized composite endpoints**. *Stat Methods Med Res* 31(8):1423-1438. doi: 10.1177/09622802221090761.

---

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] Random number generator (RNG) state used when starting data spliting. If NULL no state is set.

Value

An S3 object of class performance.

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] Number 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] Random number generator (RNG) state used when starting resampling. If NULL no state is set.
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.

Value

An S3 object of class performance.

---

plot.S3sensitivity *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 'S3sensitivity'
plot(x, plot = TRUE, ...)

## S3 method for class 'S3sensitivity'
autoplot(
  object,
  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

plot	[logical] should the graph be displayed in a graphical window
...	not used. For compatibility with the generic method.
object, x	output of the sensitivity method
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.

Details

The `autoplot` and `plot` methods are very similar. The main difference is that the former returns a ggplot2 object whereas the later automatically display the figure in a graphical window and returns an (invible) list with the plot and the data.

Value

a ggplot2 object

---

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. Can also be use for power calculation or to approximate the sample size needed to reach a specific power.

Usage

```
powerBuyseTest(
  formula,
  sim,
  sample.size,
  n.rep = c(1000, 10),
```

```
  null = c(netBenefit = 0),
  cpus = 1,
  export.cpus = NULL,
  seed = NULL,
  conf.level = NULL,
  power = NULL,
  max.sample.size = 2000,
  alternative = NULL,
  order.Hprojection = NULL,
  transformation = NULL,
  trace = 1,
  ...
)
```

Arguments

formula	[formula] a symbolic description of the GPC model, typically `treatment ~ type1(endpoint1) + type2(endpoint2, threshold2) + strata`. See Details in the documentation of the BuyseTest function, section "Specification of the GPC model".
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.frame objects or inherits from data.frame.
sample.size	[integer vector or matrix, >0] the group specific sample sizes relative to which the simulations should be perform. When a vector, the same sample size is used for each group. Alternatively can be a matrix with two columns, one for each group (respectively T and C).
n.rep	[integer, >0] the number of simulations. When specifying the power instead of the sample size, should be a vector of length 2 where the second element indicates the number of simulations used to identify the sample size.
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.
export.cpus	[character vector] name of the variables to export to each cluster.
seed	[integer, >0] Random number generator (RNG) state used when starting the simulation study. If `NULL` no state is set.
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` is 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 (respectively T and C) 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.
This approximation can be improved by increasing the sample size (argument `max.sample.size`) and/or by performing it multiple times based on a different dataset and average estimated sample size per group (second element of argument `n.rep`).
To evaluate the approximation, a simulation study is then performed with the estimated sample size. It will not exactly match the requested power but should provide a reasonnable guess which can be refined with further simulation studies. The larger the sample size (and/or number of CPUs) the more accurate the approximation.

seed: the seed is used to generate one seed per simulation. These simulation seeds are the same whether one or several CPUs are used.

Value

An S4 object of class `S4BuysePower`.

Author(s)

Brice Ozenne

Examples

```
library(data.table)

#### Using simBuyseTest ####
## save time by not generating TTE outcomes
simBuyseTest2 <- function(...){simBuyseTest(..., argsCont = NULL, argsTTE = NULL)}

## only point estimate
```

```
## Not run:
pBT <- powerBuyseTest(sim = simBuyseTest2, sample.size = c(10, 25, 50, 75, 100),
                  formula = treatment ~ bin(toxicity), seed = 10, n.rep = 1000,
                  method.inference = "none", keep.pairScore = FALSE, cpus = 5)
summary(pBT)
model.tables(pBT)

## End(Not run)

## point estimate with rejection rate

## Not run:
powerBuyseTest(sim = simBuyseTest2, 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, formula = group ~ cont(Y),
                         n.rep = c(1000,10), max.sample.size = 2000, cpus = 5,
                         seed = 10)
nobs(sampleW)
summary(sampleW) ## not very accurate but gives an order of magnitude

sampleW2 <- powerBuyseTest(sim = simFCT2, power = 0.8, formula = group ~ cont(Y),
                         n.rep = c(1000,10), max.sample.size = 2000, cpus = 5,
                         seed = 10)
summary(sampleW2) ## more accurate when the sample size needed is not too small
```

```
## 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,
  type = NULL,
  se = NULL,
  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.
`type`	[character] by default (`NULL`), output the cif (competing risks) or survival (no competing risks) at the timepoints specified by argument times. Otherwise, when argument times is missing, output the discretisation times for the survival (`"jump"`), or the survival at the last observation time (`"last"`), or the last observation time (`"last.time"`), or the change in cif or survival between discretisation times (`"change"`).
`se`	[logical] Should the standard errors associated with the cif/survival be output?
`iid`	[logical] Should the influence function associated with the cif/survival be output?
`...`	not used, for compatibility with the generic method.

Value

a list containing the survival (element `survival`) or the cumulative incidence function (element `cif`), and possible standard errors (element `.se`) and influence function (element `.iid`).

prodige *RCT In Metastatic Pancreatic Cancer Comparing Two Chemoterapy.*

Description

Simulated data inspired from the PRODIGE trial comparing the survival of patients with metastatic pancreatic cancer treated with FOLFIRINOX or gemcitabine .

- `id`: study participant.
- `treatment`: treatment arm: FOLFIRINOX (T) or gemcitabine (C).
- `OS`: time from inclusion (say diagnosis) to end of follow-up.
- `statusOS`: event triggering the end of follow-up: death (1), drop-out (0).
- `PFS`: time from inclusion (say diagnosis) to progression of the disease or end of follow-up.
- `statusPFS`: progression (1) or end of follow-up (0).
- `toxicity`: most serious side effect observed during the follow-up (1 mild, 6 severe).
- `sex`: male (M) or female (F)

Usage

```
data(prodige, package = "BuyseTest")
```

Format

An object of class `data.table` (inherits from `data.frame`) with 823 rows and 8 columns.

Author(s)

Brice Ozenne

References

Conroy, Thierry, et al. "FOLFIRINOX versus gemcitabine for metastatic pancreatic cancer" New England Journal of Medicine (2011) 364(19):1817-25. doi: 10.1056/NEJMoa1011923.

`profil` *N-of-1 trials with On-demand Sildenafil as a Treatment for Raynaud Phenomenon*

Description

Data from the Profil trial, a multiple crossover study where 38 patients had repeat blocks. Each block lasted 3 weeks and corresponded to a sequence of all three treatments (placebo, sildenafil 40mg, or sildenafil 80mg), each treatment being allocated during 1 week. The principal outcome were Raynaud Condition Score (RCS) and the frequency and cumulative duration of attacks over 24 hours.

- `id`: study participant.
- `age`: age in years, between 18 and 74 years.
- `male`: indicator of whether the participant was a male.
- `period`: number of the block for a given patient. From 1 to 5.
- `time`: number of days since inclusion.
- `treatment`: treatment allocated during the period (placebo, lowDose, highDose)
- `rcs`: RCS score, ranging from 0 to 10.
- `number`: daily number of attack, ranging from 0 to 9.
- `duration`: cumulative daily duration of attacks in minutes, between 0 and 755 minutes.

Usage

```
data(profil, package = "BuyseTest")
```

Format

An object of class `data.frame` with 2306 rows and 9 columns.

Author(s)

Brice Ozenne took a subset of the columns from the original dataset (available at https://datadryad.org/dataset/doi:10.5061/dryad.c670tq2) and translated their names into english.

References

Matthieu Roustit, Joris Giai, Olivier Gaget, et al. On-Demand Sildenafil as a Treatment for Raynaud Phenomenon: A Series of n-of-1 Trials. Ann Intern Med.2018;169:694-703. [Epub 30 October 2018]. doi:10.7326/M18-0517

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-model.tables`
Extract Summary for Class "S4BuysePower"

Description

Extract a summary of the results from the `powerBuyseTest` function.

Usage

```
## S4 method for signature 'S4BuysePower'
model.tables(
  x,
  type = "summary",
  statistic = NULL,
  endpoint = NULL,
  order.Hprojection = NULL,
  transformation = NULL
)
```

Arguments

x	output of powerBuyseTest
type	[character] should a summary of the results ("summary") or the raw results ("raw") be output?
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.

Value

data.frame

Author(s)

Brice Ozenne

See Also

powerBuyseTest for performing a simulation study for generalized pairwise comparison.

S4BuysePower-nobs *Sample Size for Class "S4BuysePower"*

Description

Display the sample size in each treatmnet arm as well as the number of pairs.

Usage

```
## S4 method for signature 'S4BuysePower'
nobs(object, ...)
```

Arguments

object	an R object of class S4BuysePower, i.e., output of powerBuyseTest
...	no used, for compatibility with the generic method.

Value

A data.frame with two colunms, one for each treatment group, and as many rows as sample sizes used for the simulation.

Author(s)

Brice Ozenne

`S4BuysePower-print` *Print Method for Class "S4BuysePower"*

Description

Display the main results stored in a `S4BuysePower` object.

Usage

```
## S4 method for signature 'S4BuysePower'
print(x, ...)
```

Arguments

`x`	an R object of class `S4BuysePower`, i.e., output of `powerBuyseTest`
`...`	additional arguments passed to the summary method.

Value

invisible table

Author(s)

Brice Ozenne

See Also

`powerBuyseTest` for performing power calculation based on GPC.
`S4BuysePower-summary` for a more detailed presentation of the `S4BuysePower` object.

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

Description

Display the main results stored in a S4BuysePower object.

Display the main results stored in a S4BuyseTest object.

Usage

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

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

Arguments

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

Value

invisible NULL

Author(s)

Brice Ozenne

See Also

powerBuyseTest for performing power calculation based on GPC.
S4BuysePower-summary for a more detailed presentation of the S4BuysePower object.

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

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

Description

Summarize the results from the powerBuyseTest function.

Usage

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

Arguments

object	output of powerBuyseTest
statistic	[character] the statistic summarizing the pairwise comparison: "netBenefit", "winRatio", "favorable", "unfavorable". See the documentation of the coef method for further details. 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.
print	[logical] Should the table be displayed?.
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.
...	Not used. For compatibility with the generic method.

Value

data.frame

Author(s)

Brice Ozenne

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 *Extract Summary Statistics from GPC*

Description

Extract summary statistics (net benefit, win ratio, ...) from GPC.

Usage

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

Arguments

object	a S4BuyseTest 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 statistic summarizing the pairwise comparison: • "netBenefit": displays the net benefit, as described in Buyse (2010) and Peron et al. (2016)),

- "winRatio": win ratio or win odds.
- "favorable": proportion strictly in favor of the treatment or Mann-Whitney parameter.
- "unfavorable": proportion in favor of the control.
- "neutral": proportion of neutral pairs.
- "uninf": proportion of uninformative pairs.
- "count.favorable": number of pairs in favor of the treatment.
- "count.unfavorable": number of pairs in favor of the control.
- "count.neutral": number of neutral pairs.
- "count.uninf": number of uninformative pairs.

Default value read from BuyseTest.options().

strata	[character vector] the strata relative to which the statistic should be output. Can also be "global" or FALSE to output the statistic pooled over all strata, or TRUE to output each strata-specific statistic.
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?
simplify	[logical] should the result be coerced to the lowest possible dimension?
...	ignored.

Details

statistic: with a single endpoint denoted Y and X in the treatment and control group and a threshold of clinical relevance τ:

- "netBenefit": $P[Y \geq X + \tau] - P[X \geq Y + \tau]$. See Buyse (2010).
- "winRatio": the win ratio $\frac{P[Y \geq X+\tau]}{P[X \geq Y+\tau]}$ or the win odds $\frac{P[Y \geq X+\tau]+0.5P[|Y-X|<\tau]}{P[X \geq Y+\tau]+0.5P[|Y-X|<\tau]}$. see Wang (2016) and Dong (2019).
- "favorable": $P[Y \geq X + \tau]$ or the Mann-Whitney parameter $P[Y \geq X + \tau] + 0.5P[|Y - X| < \tau]$. See Fay (2018).
- "unfavorable": $P[Y \leq X + \tau]$ or $P[Y \leq X + \tau] + 0.5P[|Y - X| < \tau]$.

The value of the argument add.halfNeutral used when running BuyseTest decides whether $0.5P[|Y - X| < \tau]$ is considered, e.g. whether the win ratio or win odds is output.

Value

When resampling=FALSE and simplify=FALSE, a matrix (strata, endpoint). When resampling=FALSE and simplify=FALSE, an array (sample, strata, endpoint).

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 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
On the win odds: Dong, G., Hoaglin, D. C., Qiu, J., Matsouaka, R. A., Chang, Y. W., Wang, J., & Vandemeulebroecke, M. (2019). **The Win Ratio: On Interpretation and Handling of Ties**. *Statistics in Biopharmaceutical Research*, 12(1), 99–106. https://doi.org/10.1080/19466315.2019.1575279
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

`S4BuyseTest-confint` *Extract Confidence Interval from GPC*

Description

Extract confidence intervals for summary statistics (net benefit, win ratio, ...) estimated by GPC.

Usage

```
## S4 method for signature 'S4BuyseTest'
confint(
  object,
  endpoint = NULL,
  statistic = NULL,
  strata = 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"`, `"winRatio"`, `"favorable"`, `"unfavorable"`. See the documentation of the `coef` method for further details. Default value read from `BuyseTest.options()`.

strata	[character] the strata relative to which the statistic should be output. Can also be "global" or FALSE to output the statistic pooled over all strata, or TRUE to output each strata-specific statistic.
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

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

See Also

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

S4BuyseTest-model.tables

Extract Summary for Class "S4BuyseTest"

Description

Extract a summary of the results from the `BuyseTest` function.

Usage

```
## S4 method for signature 'S4BuyseTest'
model.tables(
  x,
  percentage = TRUE,
  statistic = NULL,
  conf.level = NULL,
```

```
  strata = NULL,
  columns = "summary",
  ...
)
```

Arguments

x	output of BuyseTest
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", "winRatio", "favorable", "unfavorable". See the documentation of the coef method for further details. Default value read from BuyseTest.options().
conf.level	[numeric] confidence level for the confidence intervals. Default value read from BuyseTest.options().
strata	[logical] should the strata-specific results be displayed or the results pooled across strata? Can also be NULL to display both.
columns	[character vector] subset of columns to be output (e.g. "endpoint", "favorable", ...). Can also be "summary" or "print" to only select columns displayed in the summary or print. NULL will select all columns.
...	arguments to be passed to S4BuyseTest-confint

Author(s)

Brice Ozenne

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)

 model.tables(BT)
 model.tables(BT, percentage = FALSE)
 model.tables(BT, statistic = "winRatio")
```

S4BuyseTest-nobs *Sample Size for Class "S4BuyseTest"*

Description

Display the sample size in each treatment arm as well as the number of pairs.

Usage

```
## S4 method for signature 'S4BuyseTest'
nobs(object, strata = FALSE, resampling = FALSE, simplify = TRUE, ...)
```

Arguments

object	an R object of class S4BuyseTest, i.e., output of BuyseTest
strata	[character vector] the strata relative to which the number of pairs should be output. Can also be "global" or FALSE to output the total number of pairs (i.e. across all strata), or TRUE to output each strata-specific number of pairs.
resampling	[logical] should the sample size of each bootstrap or permutation sample be output?
simplify	[logical] should the result be coerced to the lowest possible dimension?
...	no used, for compatibility with the generic method.

Value

A vector (when argument strata is FALSE) or a matrix (when argument strata is TRUE). In the latter case each line correspond to a strata.

Author(s)

Brice Ozenne

S4BuyseTest-plot *Graphical Display for GPC*

Description

Graphical display of the percentage of favorable, unfavorable, neutral, and uninformative pairs per endpoint.

Usage

```
## S4 method for signature 'S4BuyseTest,ANY'
plot(
  x,
  type = "hist",
  strata = "global",
  endpoint = NULL,
  label.strata = NULL,
  label.endpoint = NULL,
  plot = TRUE,
  color = c("#7CAE00", "#F8766D", "#C77CFF", "#00BFC4"),
  ...
)
```

Arguments

x	an R object of class S4BuyseTest, i.e., output of BuyseTest
type	[character] type of plot: histogram ("hist"), pie chart ("pie"), or nested pie charts ("racetrack").
strata	[character vector] strata(s) relative to which the percentage should be displayed.
endpoint	[character vector] endpoint(s) relative to which the percentage should be displayed.
label.strata	[character vector] new labels for the strata levels. Should match the length of argument strata.
label.endpoint	[character vector] new labels for the endpoints. Should match the length of argument endpoint.
plot	[logical] should the graphic be displayed in a graphical window.
color	[character vector] colors used to display the percentages for each type of pair.
...	not used, for compatibility with the generic function.

Value

an invisible list containing the data and the ggplot object used for graphical display.

Examples

```
if(require(ggplot2)){

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

ff1 <- treatment ~ bin(toxicity) + TTE(eventtime, status = status,
                                       restriction = 1, threshold = 0.5)
BT1 <- BuyseTest(ff1, data= df.data)
plot(BT1, type = "hist")
plot(BT1, type = "pie")
```

```
plot(BT1, type = "racetrack")

ff2 <- update(ff1, ~.+cont(score))
BT2 <- BuyseTest(ff2, data= df.data)
plot(BT2, type = "hist")
plot(BT2, type = "pie")
plot(BT2, type = "racetrack")

}
```

S4BuyseTest-print *Print Method for Class "S4BuyseTest"*

Description

Display the main results stored in a S4BuyseTest object.

Usage

```
## S4 method for signature 'S4BuyseTest'
print(x, ...)
```

Arguments

x	an R object of class S4BuyseTest, i.e., output of BuyseTest
...	additional arguments passed to the summary method.

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 = NULL,
  type.display = 1,
  digit = c(2, 4, 5),
  ...
)
```

Arguments

object	output of BuyseTest
print	[logical] Should the results be displayed in the console?
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", "winRatio", "favorable", "unfavorable". See the documentation of the coef method for further details. Default value read from BuyseTest.options().
conf.level	[numeric] confidence level for the confidence intervals. Default value read from BuyseTest.options().
strata	[logical] should the strata-specific results be displayed or the results pooled across strata? Can also be NULL to display both.
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` (or `total(%)`) lists the number (or percentage) of pairs to be analyzed at the current priority (or strata).
- `favorable` (or `favorable(%)`) lists the number (or percentage) of pairs classified in favor of the treatment group at the current priority (or strata).
- `unfavorable` (or `unfavorable(%)`) lists the number (or percentage) of pairs classified in favor of the control group at the current priority (or strata).
- `neutral` (or `neutral(%)`) lists the number (or 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` (or `uninf(%)`) lists the number (or 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 (not adjusted 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

See Also

`BuyseTest` for performing a generalized pairwise comparison.
`S4BuyseTest-model.tables` to obtain the table displayed at the end of the summary method in a `data.frame` format. `S4BuyseTest-confint` to output estimate, standard errors, confidence interval and p-values. `S4BuyseTest-plot` for a graphical display of the scoring of the pairs. `BuyseMultComp` for efficient adjustment for multiple comparisons.

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")
```

S4BuyseTest-update *Re-run two-group GPC*

Description

Re-run Generalized Pairwise Comparisons (GPC) between two groups.

Usage

```
## S4 method for signature 'S4BuyseTest'
update(object, ...)
```

Arguments

object	an R object of class S4BuyseTest, i.e., output of BuyseTest
...	additional arguments to the call, or arguments with changed values.

Value

an R object of class S4BuyseTest

Author(s)

Brice Ozenne

S4BuyseTest-vcov *Extract Uncertainty from GPC*

Description

Extract uncertainty about the summary statistics (net benefit, win ratio, ...) from GPC.

Usage

```
## S4 method for signature 'S4BuyseTest'
vcov(
  object,
  endpoint = NULL,
  statistic = NULL,
  strata = FALSE,
  cumulative = TRUE,
  ...
)
```

Arguments

object	a S4BuyseTest object, output of BuyseTest.
endpoint	[character] for which endpoint(s) the variance-covariance matrix should be output? If NULL consider all endpoints.
statistic	[character] the statistic summarizing the pairwise comparison relative to which the variance is to be output: "netBenefit", "winRatio", "favorable", "unfavorable". See the documentation of the coef method for further details. Default value read from BuyseTest.options().

strata	[character vector] the strata relative to which the variance-covariance matrix should be output. Can also be "global" or FALSE to output the statistic pooled over all strata.
cumulative	[logical] should the summary statistic be cumulated over endpoints? Otherwise display the contribution of each endpoint.
...	ignored.

Details

When consider a second order H-decomposition, it is only used to estimate variance: the correlation between endpoints is evaluated using the first order H-decomposition.

Value

A numeric matrix.

Author(s)

Brice Ozenne

---

sensitivity *Sensitivity Analysis for the Choice of the Thresholds*

Description

Evaluate a summary statistic (net benefit, win ratio, ...) using GPC along various thresholds of clinical relevance.

Usage

```
## 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
threshold	[list] a list containing for each endpoint the thresholds to be considered.
statistic	[character] the statistic summarizing the pairwise comparison: "netBenefit", "winRatio", "favorable", "unfavorable". See the documentation of the coef method for further details. 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.
...	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.

Value

An S3 object of class S3sensitivity.

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)
plot(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)
plot(se.BT2)
plot(se.BT2, col = NA)

## End(Not run)
```

---

simBuyseTest *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(),
  names.strata = NULL,
  level.strata = NULL,
  n.strata = NULL,
  name.cluster = "id",
  prefix.cluster = NULL,
  name.treatment = "treatment",
  level.treatment = c("C", "T"),
  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.
names.strata	[character vector] name of the strata variables. Must have same length as n.strata.
level.strata	[list of character vector] value associated to each strata. Must have same length as n.strata.
n.strata	[integer, >0] number of strata. NULL indicates no strata.
name.cluster	[character] name of the cluster variable. If NULL no cluster variable is created.
prefix.cluster	[character] character string to be added to the cluster index.
name.treatment	[character] name of the treatment variable.
level.treatment	[character vector of length 2] levels of the treatment variable.
format	[character] the format of the output. Can be "data.table", "data.frame", "matrix" or "function".
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. #'

Value

Depends on the argument format: either a data.frame, data.table, matrix containing the simulated data, or a function that can be used to simulate data.

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)

#### pre-computation ####
library(lava)
mySimulator <- simBuyseTest(n, format = "function") ## creates lvm object once for all
set.seed(1)
sim(mySimulator)
set.seed(2)
sim(mySimulator)
```

`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.

Value

A data.frame

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)
```

---

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

∗ **S4BuyseTest-method**
getCount, 39
getIid, 39
getMatrixScore, 41
getPairScore, 43
getSurvival, 47
∗ **classes**
BuyseTest.options-class, 27
S4BuysePower-class, 64
S4BuyseTest-class, 69
∗ **datagen**
simBuyseTest, 84
simCompetingRisks, 88
∗ **datasets**
CHARM, 33
EB, 37
prodige, 61
profil, 62
∗ **function**
constStrata, 36
∗ **get**
getCount, 39
getMatrixScore, 41
getPairScore, 43
getSurvival, 47
∗ **hplot**
autoplot.S4BuyseTest, 13
plot.S3sensitivity, 55
S4BuyseTest-plot, 75
∗ **htest**
BuyseMultComp, 15
performanceResample, 54
powerBuyseTest, 56
sensitivity, 82
∗ **methods**
as.data.table.performance, 11
coef.BuyseTestBrier, 34
confint.BuyseTestBrier, 35
iid.BuyseTestBrier, 49
iid.prodlim, 49
predict.BuyseTTEM, 60
S4BuysePower-model.tables, 64
S4BuysePower-nobs, 65
S4BuyseTest-model.tables, 73
S4BuyseTest-nobs, 75
S4BuyseTest-update, 80
∗ **method**
getPseudovalue, 46
S4BuyseTest-coef, 69
S4BuyseTest-confint, 71
S4BuyseTest-vcov, 81
∗ **models**
auc, 11
brier, 14
BuyseTest, 17
BuyseTTEM, 28
CasinoTest, 31
∗ **model**
performance, 51
∗ **print**
S4BuysePower-print, 66
S4BuysePower-show, 67
S4BuysePower-summary, 67
S4BuyseTest-print, 77
S4BuyseTest-summary, 77
summary.performance, 90
∗ **utilities**
.colCenter_cpp, 6
.colCumSum_cpp, 7
.colMultiply_cpp, 7
.colScale_cpp, 8
.rowCenter_cpp, 8
.rowCumProd_cpp, 9
.rowCumSum_cpp, 9
.rowMultiply_cpp, 10
.rowScale_cpp, 10
BuyseTest.options, 26
constStrata, 36

.colCenter_cpp, 6
.colCumSum_cpp, 7
.colMultiply_cpp, 7
.colScale_cpp, 8
.rowCenter_cpp, 8
.rowCumProd_cpp, 9
.rowCumSum_cpp, 9
.rowMultiply_cpp, 10
.rowScale_cpp, 10

alloc,BuyseTest.options-method
(BuyseTest.options-methods), 27
as.data.table.performance, 11
auc, *4*, 11
autoplot.S3sensitivity, *4*
autoplot.S3sensitivity
(plot.S3sensitivity), 55
autoplot.S4BuyseTest, 13

brier, *4*, 14
BuyseMultComp, *4*, 15, *80*
BuyseTest, *4*, *14*, 17, *32*, *39–41*, *43*, *46, 47*, *51*, *57*, *67*, *69–71*, *73–78*, *80, 81*, *83*
BuyseTest-package, 4
BuyseTest.options, *4*, 26, *27*
BuyseTest.options-class, 27
BuyseTest.options-methods, 27
BuyseTTEM, *4*, 28

CasinoTest, *4*, 31
CHARM, 33
coef,S4BuyseTest-method
(S4BuyseTest-coef), 69
coef.BuyseTestAuc, 34
coef.BuyseTestBrier, 34
confint,S4BuyseTest-method
(S4BuyseTest-confint), 71
confint.BuyseTestAuc, 35
confint.BuyseTestBrier, 35
constStrata, *4*, 36

EB, 37
efronlim, 38

getCount, *4*, 39
getCount,S4BuyseTest-method (getCount), 39
getIid, *4*, *24*, 39
getIid,S4BuyseTest-method (getIid), 39
getMatrixScore, *24*, 41
getMatrixScore,S4BuyseTest-method
(getMatrixScore), 41
getPairScore, *4*, *24*, 43
getPairScore,S4BuyseTest-method
(getPairScore), 43
getPseudovalue, *4*, 46
getPseudovalue,S4BuyseTest-method
(getPseudovalue), 46
getSurvival, *4*, 47
getSurvival,S4BuyseTest-method
(getSurvival), 47
GPC_cpp, *5*

iid.BuyseTestAuc, 48
iid.BuyseTestBrier, 49
iid.prodlim, 49

model.tables,S4BuysePower-method
(S4BuysePower-model.tables), 64
model.tables,S4BuyseTest-method
(S4BuyseTest-model.tables), 73
mover, 50

nobs,S4BuysePower-method
(S4BuysePower-nobs), 65
nobs,S4BuyseTest-method
(S4BuyseTest-nobs), 75

performance, *4*, 51, *55*
performanceResample, *4*, 54
plot,S4BuyseTest,ANY-method
(S4BuyseTest-plot), 75
plot.S3sensitivity, 55
powerBuyseTest, *4*, 56, *64–68*
predict.BuyseTTEM, 60
print,S4BuysePower-method
(S4BuysePower-print), 66
print,S4BuyseTest-method
(S4BuyseTest-print), 77
prodige, 61
profil, 62

rbind.performance, 63

S4BuysePower, *58*
S4BuysePower-class, 64
S4BuysePower-model.tables, 64
S4BuysePower-nobs, 65
S4BuysePower-print, 66

S4BuysePower-show, 67
S4BuysePower-summary, 67
S4BuyseTest, *14*, *23*, *39–41*, *43*, *46*, *47*, *71*, *76*, *83*
S4BuyseTest-class, 69
S4BuyseTest-coef, 69
S4BuyseTest-confint, 71
S4BuyseTest-model.tables, 73
S4BuyseTest-nobs, 75
S4BuyseTest-plot, 75
S4BuyseTest-print, 77
S4BuyseTest-show *(*S4BuysePower-show*)*, 67
S4BuyseTest-summary, 77
S4BuyseTest-update, 80
S4BuyseTest-vcov, 81
select,BuyseTest.options-method *(*BuyseTest.options-methods*)*, 27
sensitivity, *4*, *24*, 82
sensitivity,S4BuyseTest-method *(*sensitivity*)*, 82
show,S4BuysePower-method *(*S4BuysePower-show*)*, 67
show,S4BuyseTest-method *(*S4BuysePower-show*)*, 67
simBuyseTest, *5*, 84
simCompetingRisks, *5*, 88
summary,S4BuysePower-method *(*S4BuysePower-summary*)*, 67
summary,S4BuyseTest-method *(*S4BuyseTest-summary*)*, 77
summary.performance, 90

update,S4BuyseTest-method *(*S4BuyseTest-update*)*, 80

vcov,S4BuyseTest-method *(*S4BuyseTest-vcov*)*, 81