Overview of the package BuyseTest

Brice Ozenne

July 7, 2025

This vignette describes the main functionalities of the **BuyseTest** package, focusing on software (and not statistical) aspects, and assume that the reader is familar with the GPC framework [1].

The BuyseTest package implements the Generalized Pairwise Comparisons (GPC) as defined in Buyse (2010) for complete observations, and extended in Péron et al. (2018) to deal with right-censoring and Piffoux et al. (2024) to incorporate a restriction time. When considering a single endpoint, the GPC procedure can be summarized as follow. Denote the endpoint by Y in the treatment group and by X in the control group. Given a threshold of clinical relevance τ, the aim of GPC is to estimate the proportion in favor of treatment[2] $\mathbb{P}\left[Y \geq X + \tau\right]$ and the proportion in favor of control $\mathbb{P}\left[X \geq Y + \tau\right]$. Their difference $\mathbb{P}\left[Y \geq X + \tau\right] - \mathbb{P}\left[X \geq Y + \tau\right]$ leads to the net treatment benefit and their ratio $\frac{\mathbb{P}[Y \geq X+\tau]}{\mathbb{P}[X \geq Y+\tau]}$ to the win ratio. The software also evaluate the proportion of neutral pairs $\mathbb{P}\left[|X - Y| < \tau\right]$ and which can be included to obtain the probabilistic index $\mathbb{P}\left[Y \geq X + \tau\right] + 0.5\mathbb{P}\left[|X - Y| < \tau\right]$ or win odds $\frac{\mathbb{P}[Y \geq X+\tau]+0.5\mathbb{P}[|X-Y|<\tau]}{\mathbb{P}[X \geq Y+\tau]+0.5\mathbb{P}[|X-Y|<\tau]}$.

- the function `BuyseTest` performs the GPC procedure and is the main function of the package. The user can interact with its output via various methods:
  - `summary` to obtain an overview of the results, including the estimated net treatment benefit. The result table at the end of the output can be directly access using `model.tables`.
  - `coef` to extract the estimates.
  - `confint` or `model.tables` to extract estimates, confidence intervals, and p.values.
  - `plot` for a graphical display of the scoring of the pair per endpoint.
  - `sensitivity` to perform a sensitivity analysis on the choice of the threshold(s).
  - `nobs` to extract the number of observations and pairs.
  - `getIid` to extract the iid decomposition of the estimator.
  - `getPairScore` to extract the contribution of each pair to the net treatment benefit.
  - `getSurvival` to extract the estimates of the survival used for right-censored endpoints.
  - `BuyseMultComp` to adjust p-values and confidence intervals for multiple comparisons.
- the `powerBuyseTest` function performs simulation studies, e.g. to estimate the statistical power or assess the bias / type 1 error rate of a test for a specific design. The `simBuyseTest` function can facilitate the definition of the data generating mechanism.

[1] if not, Buyse (2010) is a good place to start.

[2] in absence of ties this equals the Wilcoxon-Mann-Whitney parameter

- the `BuyseTest.options` function enables the user to access the default values used in the **BuyseTest** package. The function can also change the default values to better match the user needs.

Another vignette, "Wilcoxon test via GPC", details connexions between GPC and the Wilcoxon rank sum test. Before going further we need to load the **BuyseTest** package in the R session:

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

To illustrate the functionalities of the package, we will used the `veteran` dataset from the **survival** package:

```
data(cancer, package = "survival")
veteran <- cbind(id = 1:NROW(veteran), veteran)
veteran$trt <- factor(veteran$trt,1:2,c("Pl","Exp"))
head(veteran)
```

```
  id trt celltype time status karno diagtime age prior
1  1  Pl squamous   72      1    60        7  69     0
2  2  Pl squamous  411      1    70        5  64    10
3  3  Pl squamous  228      1    60        3  38     0
4  4  Pl squamous  126      1    60        9  63    10
5  5  Pl squamous  118      1    70       11  65    10
6  6  Pl squamous   10      1    20        5  49     0
```

See `?veteran` for a presentation of the database.

Note: the **BuyseTest** package is under active development. Newer package versions may include additional functionalities and fix previous bugs. The version of the package that is being is:

```
utils::packageVersion("BuyseTest")
```

```
[1] '3.3.1'
```

For completness, the details of the R session used to generate this document are:

```
sessionInfo()
```

```
R version 4.3.3 (2024-02-29)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 22.04.5 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
```

```
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: Europe/Copenhagen
tzcode source: system (glibc)

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] data.table_1.17.2  BuyseTest_3.3.1    Rcpp_1.0.14        prodlim_2025.04.28
[5] ggplot2_3.5.2      survival_3.5-8

loaded via a namespace (and not attached):
 [1] Matrix_1.6-5        gtable_0.3.6        future.apply_1.11.3 dplyr_1.1.4
 [5] compiler_4.3.3      tidyselect_1.2.1    MatrixModels_0.5-3  parallel_4.3.3
 [9] globals_0.18.0      splines_4.3.3       scales_1.4.0        lattice_0.22-5
[13] R6_2.6.1            generics_0.1.3      future_1.49.0       tibble_3.2.1
[17] pillar_1.10.2       RColorBrewer_1.1-3  rlang_1.1.6         cli_3.6.5
[21] withr_3.0.2         magrittr_2.0.3      digest_0.6.37       grid_4.3.3
[25] lifecycle_1.0.4     lava_1.8.1          vctrs_0.6.5         SparseM_1.81
[29] glue_1.8.0          farver_2.1.2        listenv_0.9.1       codetools_0.2-19
[33] stats4_4.3.3        colorspace_2.1-1    parallelly_1.44.0   tools_4.3.3
[37] pkgconfig_2.0.3
```

1 Performing generalized pairwise comparisons (GPC)

To perform generalized pairwise comparisons, the `BuyseTest` function needs:

- where the data are stored - argument `data`
- the name of the endpoints - argument `endpoint`
- the type of each endpoint - argument `type`
- the variable defining the two treatment groups - argument `treatment`

The `BuyseTest` function has many optional arguments. For example:

- the threshold of clinical relevance associated to each endpoint - argument `threshold`
- the censoring associated to each endpoint (for time to event endpoints) - argument `status`

There are two equivalent ways to define the GPC:

- using a separate argument for each element:

```
BT <- BuyseTest(data = veteran,
                endpoint = "time",
                type = "timeToEvent",
                treatment = "trt",
                status = "status",
                threshold = 20)
```

```
        Generalized Pairwise Comparisons

Settings
   - 2 groups  : Control = Pl and Treatment = Exp
   - 1 endpoint:
      priority endpoint type           operator             threshold event
      1        time     time to event  higher is favorable  20        status (0 1)
   - right-censored pairs: probabilistic score based on the survival curves

Point estimation and calculation of the iid decomposition

Estimation of the estimator's distribution
   - method: moments of the U-statistic

Gather the results in a S4BuyseTest object
```

- or via a formula interface. In the formula interface, endpoints are wrapped by parentheses, preceded by a character string indicated their type:
  - binary (`b`, `bin`, or `binary`)
  - continuous (`c`, `cont`, or `continuous`)
  - time to event (`t`, `tte`, or `timetoevent`)

For instance:

```
BT.f <- BuyseTest(trt ~ tte(time, threshold = 20, status = "status"),
                  data = veteran, trace = FALSE)
```

where we also set the argument `trace` to `FALSE` to execute silently the function. We can check that the two approaches are equivalent:

```
BT.f@call <- list(); BT@call <- list();
testthat::expect_equal(BT.f,BT)
```

1.1 Displaying the results

The results of the GPC can be displayed using the `summary` method:

```
summary(BT)
```

```
      Generalized pairwise comparisons with 1 endpoint

 - statistic       : net treatment benefit  (delta: endpoint specific, Delta: global)
 - null hypothesis : Delta == 0
 - confidence level: 0.95
 - inference       : H-projection of order 1 after atanh transformation
 - treatment groups: Exp (treatment) vs. Pl (control)
 - censored pairs  : probabilistic score based on the survival curves
 - results
 endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   Delta
     time        20      100        37.78          46.54      15.68        0 -0.0877
 CI [2.5% ; 97.5%] p.value
  [-0.2735;0.1045] 0.37162
```

It displays information about each endpoint, percentage of pairs classified as favorable, unfavorable, neutral, and uninformative, as well as the estimated net treatment benefit (column `Delta`), its confidence interval, and the corresponding p-value testing the absence of a group difference. To display the number of pairs instead of the percentage of pairs that are favorable/unfavorable/neutral/uniformative, one can set the argument `percentage` to `FALSE`. See `help(S4BuyseTest-summary)` for more details about the `summary` method, its input and output.

The `print` method provides a more concise display of the results:

```
print(BT, percentage = FALSE)
```

```
endpoint threshold total favorable unfavorable neutral uninf   Delta CI [2.5% ; 97.5%]
    time        20  4692   1772.59     2183.89  735.52     0 -0.0877  [-0.2735;0.1045]
p.value
0.37162
```

To access these values, we recommand using the `model.tables` method that outputs the information from the previous table in a `data.frame` format:

```
model.tables(BT, percentage = FALSE)
```

```
  endpoint threshold total favorable unfavorable  neutral uninf       Delta   lower.ci
1     time        20  4692  1772.593    2183.886 735.5205     0 -0.08765836 -0.2735301
   upper.ci  p.value
1 0.1045245 0.371617
```

An even more concise output can be obtained via the `confint` method:

```
confint(BT)
```

```
            estimate         se   lower.ci  upper.ci null  p.value
time_t20 -0.08765836 0.09760901 -0.2735301 0.1045245    0 0.371617
```

or `coef` method:

```
coef(BT)
```

```
   time_t20
-0.08765836
```

1.2 What about other summary statistics?

Results for other summary statistics are also accessible: - argument `statistic`

- proportion in favor of treatment (`favorable`): $\mathbb{P}[Y \geq X + \tau]$
- proportion in favor of control (`unfavorable`): $\mathbb{P}[X \geq Y + \tau]$
- win ratio (`winRatio`): $\frac{\mathbb{P}[Y \geq X+\tau]}{\mathbb{P}[X \geq Y+\tau]}$

For instance, to display the estimated win ratio instead of the estimated net treatment benefit, use:

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

```
      Generalized pairwise comparisons with 1 endpoint

 - statistic       : win ratio  (delta: endpoint specific, Delta: global)
 - null hypothesis : Delta == 1
 - confidence level: 0.95
 - inference       : H-projection of order 1 after log transformation
 - treatment groups: Exp (treatment) vs. Pl (control)
 - censored pairs  : probabilistic score based on the survival curves
 - results
 endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)  Delta
     time        20      100        37.78          46.54      15.68        0 0.8117
 CI [2.5% ; 97.5%] p.value
   [0.5134;1.2833] 0.37195
```

⚠ In presence of ties, the null distribution of the proportion in favor of treatment or control depends on the data generative mechanism and the threshold of clinical relevance. This is why, unless the argument `null` is provided by the user, the `confint` method will not produce any `p.value`:

```
confint(BT, statistic = "favorable")
```

```
          estimate         se  lower.ci upper.ci null p.value
time_t20 0.3777905 0.04902199 0.2874747 0.477467   NA      NA
```

A permutation test may be used to empirically estimate a value for the null hypothesis:

```
BT.perm <- BuyseTest(trt ~ tte(time, threshold = 20, status = "status"),
                     data = veteran, trace = FALSE,
                     method.inference = "permutation", seed = 10)
confint(BT.perm, statistic = "favorable")
```

```
          estimate         se lower.ci upper.ci      null   p.value
time_t20 0.3777905 0.04770182       NA       NA 0.4205855 0.3636364
```

which, in this example, is around 0.42. It worth noting that testing an inadequate null hypothesis can have dramatic consequences on the p-value:

```
rbind(confint(BT, statistic = "favorable", null = 0.42),
      confint(BT, statistic = "favorable", null = 0.5))
```

```
           estimate         se  lower.ci upper.ci null    p.value
time_t20  0.3777905 0.04902199 0.2874747 0.477467 0.42 0.39826735
time_t201 0.3777905 0.04902199 0.2874747 0.477467 0.50 0.01673643
```

Considering the proportion of neutral pairs in the summary statistics: - argument `add.halfNeutral`

- Wilcoxon-Mann-Whitney parameter or probabilistic index: $\mathbb{P}[Y \geq X + \tau] + 0.5\mathbb{P}[|Y - X| < \tau]$.
- win odds: $\frac{\mathbb{P}[Y \geq X+\tau]+0.5\mathbb{P}[|Y-X|<\tau]}{\mathbb{P}[X \geq Y+\tau]+0.5\mathbb{P}[|Y-X|<\tau]}$.

have been recommended (e.g. Ajufo et al. (2023)) over the win ratio. These summary statistics can be output by specifying the argument `add.halfNeutral` to `TRUE` when calling `BuyseTest`:

```
BT.half <- BuyseTest(trt ~ tte(time, threshold = 20, status = "status"),
                     data = veteran, trace = FALSE, add.halfNeutral = TRUE)
confint(BT.half, statistic = "favorable")
```

```
          estimate         se  lower.ci  upper.ci null   p.value
time_t20 0.4561708 0.04880921 0.3632263 0.5522714  0.5 0.3716632
```

```
confint(BT.half, statistic = "winRatio")
```

```
          estimate        se  lower.ci upper.ci null   p.value
time_t20 0.8388127 0.1650208 0.5704361 1.233454    1 0.3716211
```

Testing a net treatment benefit of 0, a win odds of 1, or a Wilcoxon-Mann-Whitney parameter of 0.5 corresponds to the same hypothesis and therefore the same p-value should be obtained. The (small) discrepancy in p-values observed in this example (0.371617 vs. 0.3716211 vs. 0.3716632) are due to small sample approximation. Such discrepancies will not arise when using non-parametric bootstrap or permutation tests using quantiles of the bootstrap or permutation distribution, e.g.:

```
BT.halfperm <- BuyseTest(trt ~ tte(time, threshold = 20, status = "status"),
                         data = veteran, trace = FALSE, add.halfNeutral = TRUE,
                         method.inference = "bootstrap", seed = 10)
Mstat <- rbind(netBenefit = confint(BT.halfperm, statistic = "netBenefit"),
               winRatio = confint(BT.halfperm, statistic = "winRatio"),
               favorable = confint(BT.halfperm, statistic = "favorable"))
Mstat
```

```
              estimate         se   lower.ci  upper.ci null p.value
netBenefit -0.08765836 0.10021632 -0.2720510 0.1033974  0.0   0.383
winRatio    0.83881270 0.17440155  0.5722640 1.2306429  1.0   0.383
favorable   0.45617082 0.05010816  0.3639745 0.5516987  0.5   0.383
```

1.3 Stratified GPC

GPC can be performed for subgroups of a categorical variable - argument `strata`
For instance, the celltype may have huge influence on the survival time and the investigator would like to only compare patients that have the same celltype. In the formula interface this is achieved by adding this variable wrapped by parenthesis and preceded by the character string strata in the formula:

```
ffstrata <- trt ~ tte(time, threshold = 20, status = "status") + strata(celltype)
BTstrata <- BuyseTest(ffstrata, data = veteran, trace = 0)
```

When doing a stratified analysis, the summary method displays strata-specific and global results[3]:

```
keep.colStrata <- c("endpoint","strata", "total",
                    "favorable","unfavorable","neutral","uninf","delta","Delta")
summary(BTstrata, type.display = keep.colStrata)
```

```
      Generalized pairwise comparisons with 1 endpoint and 4 strata

 - statistic       : net treatment benefit  (delta: endpoint specific, Delta: global)
 - null hypothesis : Delta == 0
 - confidence level: 0.95
 - inference       : H-projection of order 1 after atanh transformation
 - treatment groups: Exp (treatment) vs. Pl (control)
 - strata weights  : 26.38%, 34.63%, 18.47%, 20.52%
 - uninformative pairs: no contribution
 - results
endpoint    strata total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   delta   Delta
    time    global   100.00        36.06          45.77      17.33     0.85 -0.0997 -0.0997
          squamous    25.38        14.33           8.77       2.28     0.00  0.2193
         smallcell    45.69        12.69          20.88      11.27     0.85 -0.1792
             adeno    13.71         4.74           6.15       2.81     0.00 -0.1034
             large    15.23         4.30           9.97       0.96     0.00 -0.3722
```

The percentage of pairs in the total/favorable/unfavorable/neutral/uninf columns are relative to the overall number of pairs whereas the column `delta` presents the endpoint and strata-specific net treatment benefits (in the last 4 lines). The last column (`Delta`) displays the global (i.e. pooled over strata), conditional, net treatment benefit.
⚠ With the default way of pooling results across strata, the proportion of favorable pairs minus the proportion of unfavorable pairs (`36.06%-45.77%=9.71%`) does not equal the global net treatment benefit (`9.97%`). To retrieve this value of the Net Treatment Benefit, one should first extract the number of pairs per strata using the method `nobs`:

```
strata.obs <- as.data.frame(nobs(BTstrata, strata = TRUE))
strata.obs
```

```
          Pl Exp pairs
squamous  15  20   300
```

[3] the strata-specific results can be removed by setting the argument `strata` to `"global"` when calling `summary`.

```
smallcell 30  18   540
adeno      9  18   162
large     15  12   180
```

and use the method `model.tables` to extract the number of favorable and unfavorable pairs per strata:

```
dfStrata <- model.tables(BTstrata, percentage = FALSE,
                         strata = c("squamous","smallcell","adeno","large"),
                         columns = c("strata","total","favorable","unfavorable"))
dfStrata
```

```
     strata total favorable unfavorable
2  squamous   300 169.40260    103.6104
3 smallcell   540 150.00000    246.7778
4     adeno   162  56.00000     72.7500
5     large   180  50.83333    117.8333
```

We retrieve the strata-specific net treatment benefits by comparing, in each strata, the number of favorable and unfavorable pairs relative to the number of pairs[4]:

```
delta <- (dfStrata$favorable - dfStrata$unfavorable)/strata.obs$pairs
delta
```

```
[1]  0.2193074 -0.1792181 -0.1033951 -0.3722222
```

The global net treatment benefit is then the sum of the strata-specific net treatment benefits weighted by the strata weights:

```
weightCMH <- strata.obs$pairs/(strata.obs$Pl + strata.obs$Exp)

list(estimate = sum(delta * weightCMH/sum(weightCMH)),
     weight = 100*weightCMH/sum(weightCMH))
```

```
$estimate
[1] -0.09967584

$weight
[1] 26.38329 34.62807 18.46830 20.52034
```

⚠ The approach is true for the probabilistic index but not for the win ratio/odds: the ratio between the global proportions is taken, i.e., pooling is performed at the numerator and at the denominator instead of pooling fractions - see Dong et al. (2018), equation 1.

[4] Alernatively one could compute, from the `summary`, the difference between the percentage of favorable and unfavorable pairs relative to the percentage of pairs in the strata, e.g. $(14.33\% - 8.77\%)/25.38\% \approx 21.93\%$

The default weighting scheme is `CMH`, standing for Cochran-Mantel-Haenszel, which has been recommaned in the litterature (Dong et al., 2018). It is efficient under the assumption of a common multiplicative effect (across strata) on the odds ratio scale.

Other weighting schemes can be used. - argument `pool.strata`. When considering additive effect, one should instead weight proportionnaly to the number of pairs:

```
BTstrata2 <- BuyseTest(ffstrata, data = veteran, trace = 0, pool.strata = "buyse")
summary(BTstrata2, type.display = keep.colStrata)
```

```
      Generalized pairwise comparisons with 1 endpoint and 4 strata

 - statistic       : net treatment benefit  (delta: endpoint specific, Delta: global)
 - null hypothesis : Delta == 0
 - confidence level: 0.95
 - inference       : H-projection of order 1 after atanh transformation
 - treatment groups: Exp (treatment) vs. Pl (control)
 - strata weights  : 25.38%, 45.69%, 13.71%, 15.23%
 - uninformative pairs: no contribution
 - results
endpoint    strata total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   delta   Delta
    time    global   100.00        36.06          45.77      17.33     0.85 -0.0971 -0.0971
          squamous    25.38        14.33           8.77       2.28     0.00  0.2193
         smallcell    45.69        12.69          20.88      11.27     0.85 -0.1792
             adeno    13.71         4.74           6.15       2.81     0.00 -0.1034
             large    15.23         4.30           9.97       0.96     0.00 -0.3722
```

The strata-specifc net treatment benefits are unchanged: the weighting scheme only affects the evaluation of the overall net treatment benefit. With this weighting scheme it now equals the difference between the overall proportion of favorable vs. unfavorable pairs (`36.06%-45.77%`). While extractors will by default output global estimates (i.e. after pooling the results over strata)

```
confint(BTstrata2)
```

```
           estimate        se   lower.ci   upper.ci null  p.value
time_t20 -0.09706901 0.0977929 -0.2829348 0.09582321    0 0.323961
```

one can specify the argument `strata` to extract strata-specific estimates:

```
confint(BTstrata, strata = TRUE)
```

```
                     estimate        se   lower.ci  upper.ci null   p.value
time_t20.squamous   0.2193074 0.1911515 -0.1690137 0.5486919    0 0.2669352
time_t20.smallcell -0.1792181 0.1540933 -0.4567640 0.1301230    0 0.2551275
time_t20.adeno     -0.1033951 0.2465197 -0.5314450 0.3667172    0 0.6771002
time_t20.large     -0.3722222 0.2190018 -0.7110335 0.1068610    0 0.1240457
```

⚠ The pooled estimator presented in this section has a conditional interpretation, as they summarize comparisons made between observations from the same strata. They will generally differ from the marginal (i.e. non-adjusted) Net Treatment Benefit and tend to be more extreme (i.e. away from 0) in presence of group difference.

1.4 Standardization

When the interest lies in a marginal effect but one wish to adjust on baseline covariates to obtain more precise estimate, one should *not* restrict the comparisons between pairs of observations from the same strata. Instead one should estimate a Net Treatment Nenefit for each possible combinations of strata and pool the results (Buyse et al. (2025), chapter 9). This is what is being done when setting the argument `pool.strata` to `"standardization"`:

```
BTstd <- BuyseTest(ffstrata, data = veteran, trace = 0, pool.strata = "standardization")
model.tables(BTstd)[,c("strata","total","delta","Delta","lower.ci","upper.ci","p.value")]
```

```
               strata      total       delta     Delta   lower.ci  upper.ci   p.value
1              global 100.000000 -0.11874500 -0.118745 -0.2857534 0.0552638 0.1805479
2            squamous   6.393862  0.21930736        NA         NA        NA        NA
3  smallcell.squamous  12.787724  0.35699653        NA         NA        NA        NA
4      adeno.squamous   3.836317  0.41018519        NA         NA        NA        NA
5      large.squamous   6.393862  0.03622106        NA         NA        NA        NA
6  squamous.smallcell   5.754476 -0.50654161        NA         NA        NA        NA
7           smallcell  11.508951 -0.17921811        NA         NA        NA        NA
8     adeno.smallcell   3.452685 -0.25308642        NA         NA        NA        NA
9     large.smallcell   5.754476 -0.80740741        NA         NA        NA        NA
10     squamous.adeno   5.754476 -0.41165224        NA         NA        NA        NA
11    smallcell.adeno  11.508951 -0.02906379        NA         NA        NA        NA
12              adeno   3.452685 -0.10339506        NA         NA        NA        NA
13        large.adeno   5.754476 -0.76311728        NA         NA        NA        NA
14     squamous.large   3.836317 -0.04494949        NA         NA        NA        NA
15    smallcell.large   7.672634  0.25946502        NA         NA        NA        NA
16        adeno.large   2.301790  0.21296296        NA         NA        NA        NA
17              large   3.836317 -0.37222222        NA         NA        NA        NA
```

Here `strata` equal to `squamous` means that the comparison betwen the active and control group was made using only patients whose lung cancer cell type were `squamous`. We retrive the same results as when setting `pool.strata` to `"buyse"` or `"CMH"`. However now additional strata have been added like `"smallcell.squamous"` where control patients whose lung cancer cell type were `smallcell` are being compared to active patients whose lung cancer cell type were `squamous`. Indeed:

```
BuyseTest(trt ~ tte(time, threshold = 20, status = "status"),
          data = rbind(veteran[veteran$celltype == "smallcell" & veteran$trt == "Pl",],
                       veteran[veteran$celltype == "squamous" & veteran$trt == "Exp",]),
          trace = 0)
```

```
endpoint threshold Delta
    time        20 0.357
```

leads, up to rounding, to the same result.

Note: it is possible to extract the strata-specific estimate (e.g. `coef(BTstd, strata = TRUE)`) but the software does not keep track of the strata-specific uncertainty via the H-decomposition and thus not able to output confidence intervals. A resampling method could be used if those are of interest:

```
BTstd.boot <- BuyseTest(ffstrata, data = veteran, trace = 0, pool.strata = "standardization",
                        method.inference = "bootstrap", n.resampling = 100)
confint(BTstd.boot, strata = TRUE)[1:6,]
```

```
Estimated p-value of 1 - consider increasing the number of boostrap samples

                               estimate        se    lower.ci    upper.ci null    p.value
time_t20.squamous             0.21930736 0.1928818 -0.23620851  0.52173313    0 0.26000000
time_t20.smallcell.squamous   0.35699653 0.1672720 -0.00620671  0.62983611    0 0.05000000
time_t20.adeno.squamous       0.41018519 0.1853448  0.02059982  0.72023944    0 0.04000000
time_t20.large.squamous       0.03622106 0.2231223 -0.41553367  0.46671491    0 0.88000000
time_t20.squamous.smallcell -0.50654161 0.1655352 -0.79085529 -0.19423414    0 0.00990099
time_t20.smallcell          -0.17921811 0.1535131 -0.47130077  0.09038033    0 0.25000000
```

Here `n.resampling` was set to a low value only to save computation time but this may lead to unreliable confidence intervals/p-values: larger values are recommended (e.g. `10000`).

1.5 Using multiple endpoints

More than one endpoint can be considered by indicating a vector of endpoints, types, and thresholds. In the formula interface, the different endpoints must be separated with a "+" on the right hand side of the formula:

```
ff2 <- trt ~ tte(time, threshold = 20, status = "status") + cont(karno, threshold = 0)
BT.H <- BuyseTest(ff2, data = veteran, trace = 0)
summary(BT.H)
```

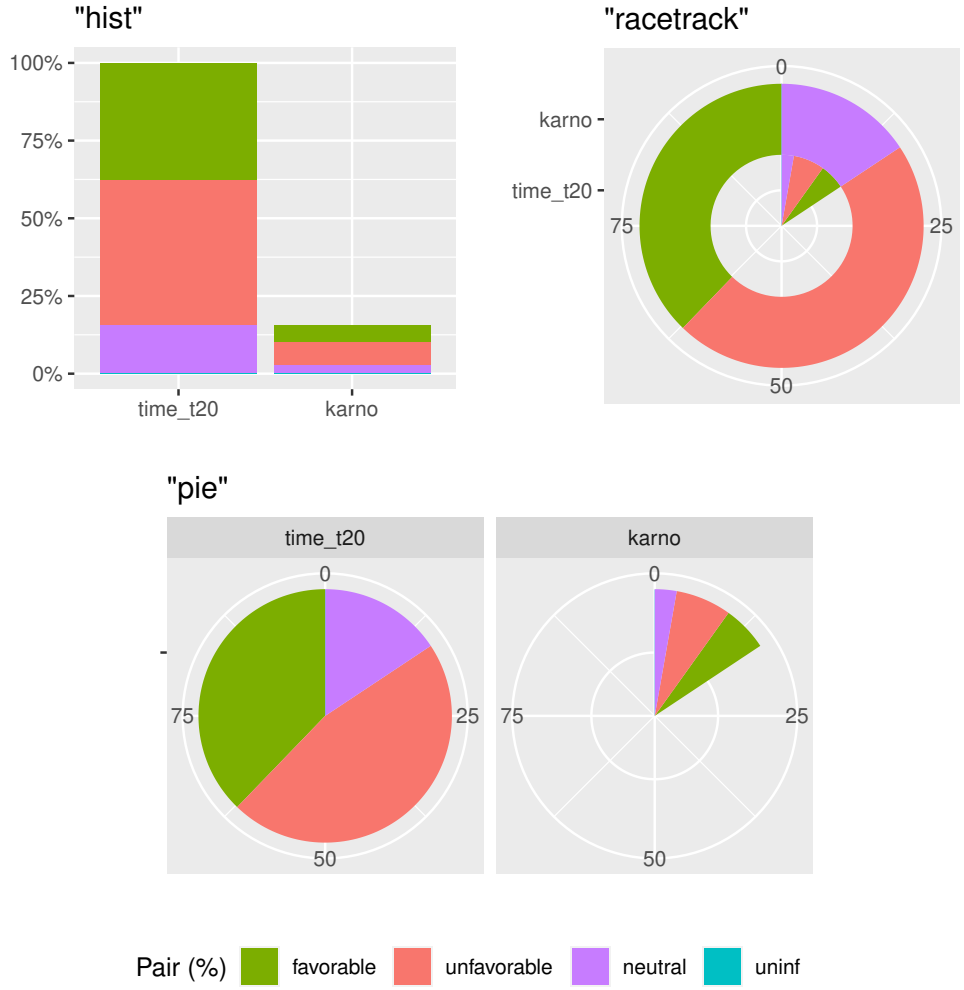
```
      Generalized pairwise comparisons with 2 prioritized endpoints

 - statistic       : net treatment benefit  (delta: endpoint specific, Delta: global)
 - null hypothesis : Delta == 0
 - confidence level: 0.95
 - inference       : H-projection of order 1 after atanh transformation
 - treatment groups: Exp (treatment) vs. Pl (control)
 - censored pairs  : probabilistic score based on the survival curves
 - neutral pairs   : re-analyzed using lower priority endpoints
 - results
 endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   delta   Delta
     time        20   100.00        37.78          46.54      15.68        0 -0.0877 -0.0877
    karno             15.68         5.78           7.11       2.78        0 -0.0133 -0.1009
 CI [2.5% ; 97.5%] p.value
  [-0.2735;0.1045] 0.37162
  [-0.2901;0.0959] 0.31478
```

The hierarchy of the endpoint is defined from left (most important endpoint, here `time`) to right (least important endpoint, here `karno`). In the `summary` output, the confidence intervals and p.values are computed for the column `Delta`, i.e. here -8.77% is the net treatment benefit for the first endpoint (line 1) and -10.09% is the net treatment benefit for the first and second endpoint (line 2). In other words, the last confidence interval and p-value is the one for the analysis over all endpoints (generally the one to report).

A graphical representation of the GPC procedure can be obtained by the `plot` method. It will display the percentage of favorable, unfavorable, neutral, and uninformative pairs per endpoint. Three (equivalent) graphical display are possible, the first one (`"hist"`) being the recommanded one:

```
plot(BT.H, type = "hist")
plot(BT.H, type = "pie")
plot(BT.H, type = "racetrack")
```



It is also possible to perform the comparisons on all pairs for all endpoints by setting the argument `hierarchical` to `FALSE`:

```
BT.nH <- BuyseTest(ff2, hierarchical = FALSE, data = veteran, trace = 0)
summary(BT.nH)
```

```
      Generalized pairwise comparisons with 2 endpoints

 - statistic       : net treatment benefit  (delta: endpoint specific, Delta: global)
 - null hypothesis : Delta == 0
 - confidence level: 0.95
 - inference       : H-projection of order 1 after atanh transformation
 - treatment groups: Exp (treatment) vs. Pl (control)
 - censored pairs  : probabilistic score based on the survival curves
 - results
 endpoint threshold weight total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   delta
     time        20    0.5      100        37.78          46.54      15.68        0 -0.0877
    karno              0.5      100        41.82          44.95      13.24        0 -0.0313
   Delta CI [2.5% ; 97.5%] p.value
 -0.0438  [-0.1388;0.0519] 0.36977
 -0.0595  [-0.2267;0.1111] 0.49514
```

In that case the score of a pair is the weighted sum of the score relative to each endpoint. By default, the weights are all set to the same value but this behavior can be changed by setting the argument `weight` when calling `BuyseTest`, e.g.:

```
ff2w <- trt ~ tte(time, threshold = 20, status = "status", weight = 0.8)
ff2w <- update.formula(ff2w, . ~ . + cont(karno, threshold = 0, weight = 0.2))
BT.nHw <- BuyseTest(ff2w, hierarchical = FALSE, data = veteran, trace = 0)
model.tables(BT.nHw)
```

```
  endpoint threshold weight total favorable unfavorable  neutral uninf       delta
1     time     2e+01    0.8   100  37.77905    46.54489 15.67606     0 -0.08765836
3    karno     1e-12    0.2   100  41.81586    44.94885 13.23529     0 -0.03132992
        Delta   lower.ci   upper.ci   p.value
1 -0.07012668 -0.2203714 0.08336855 0.3707289
3 -0.07639267 -0.2503756 0.10237001 0.4026905
```

This has been refered as the O'Brien test in the litterature (Verbeeck et al. (2019), section 3.2). Alternatively, one may be interested in the endpoint specific results. This can be performed by applying the `BuyseTest` function separately to each endpoint, e.g.:

```
confint(BuyseTest(trt ~ cont(karno, threshold = 0), data = veteran, trace = 0))
```

```
         estimate         se   lower.ci  upper.ci null   p.value
karno -0.03132992 0.09787113 -0.2197111 0.1593037    0 0.7490407
```

or setting the argument `cumulative` to `FALSE` when calling the `confint` function:

```
confint(BT.nHw, cumulative = FALSE)
```

```
            estimate         se   lower.ci  upper.ci null   p.value
time_t20 -0.08765836 0.09760901 -0.2735301 0.1045245    0 0.3716170
karno    -0.03132992 0.09787113 -0.2197111 0.1593037    0 0.7490407
```

Note: the apparent discrepency in p-value between the hierarchical and non-hierarchical GPC at the first priority (0.3762 vs 0.3698 vs 0-3707) is due to the use of a transformation that makes the p-value dependent on the estimate. Otherwise the p-value would be the same at the first priority, e.g.:

```
confint(BT.nHw, transform = FALSE)
```

```
            estimate         se   lower.ci   upper.ci null   p.value
time_t20 -0.07012668 0.07808721 -0.2231748 0.08292143    0 0.3691557
karno    -0.07639267 0.09093303 -0.2546181 0.10183280    0 0.4008534
```

1.6 Statistical inference

Uncertainty about the estimates can be quantified using: - argument `method.inference`

- **permutation test** (`"permutation"`). Assuming exchangeability under the null hypothesis, this approach gives valid p-values (regardless to the sample size) for testing the absence of a difference between the groups.

```
BT.perm <- BuyseTest(trt ~ tte(time, threshold = 20, status = "status"),
                     data = veteran, trace = 0, method.inference = "permutation",
                     seed = 10)
summary(BT.perm)
```

```
      Generalized pairwise comparisons with 1 endpoint

 - statistic       : net treatment benefit  (delta: endpoint specific, Delta: global)
 - null hypothesis : Delta == 0
 - confidence level: 0.95
 - inference       : permutation test with 1000 samples
                     p-value computed using the permutation distribution
 - treatment groups: Exp (treatment) vs. Pl (control)
 - censored pairs  : probabilistic score based on the survival curves
 - results
 endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   Delta p.value
     time        20      100        37.78          46.54      15.68        0 -0.0877 0.35265
```

- **bootstrap resampling** (`"bootstrap"`). In large enough samples, this approach gives valid p-values and confidence intervals.

```
BT.boot <- BuyseTest(trt ~ tte(time, threshold = 20, status = "status"),
                     data = veteran, trace = 0, method.inference = "bootstrap",
                     seed = 10)
summary(BT.boot)
```

```
      Generalized pairwise comparisons with 1 endpoint

 - statistic       : net treatment benefit  (delta: endpoint specific, Delta: global)
 - null hypothesis : Delta == 0
 - confidence level: 0.95
 - inference       : bootstrap resampling with 1000 samples
                     CI computed using the percentile method; p-value by test inversion
 - treatment groups: Exp (treatment) vs. Pl (control)
 - censored pairs  : probabilistic score based on the survival curves
 - results
 endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   Delta
     time        20      100        37.78          46.54      15.68        0 -0.0877
 CI [2.5% ; 97.5%] p.value
  [-0.2721;0.1034]   0.383
```

- **asymptotic distribution** (`"u-statistic"`). In large enough samples, this approach gives valid p-values and confidence intervals (Ozenne et al., 2021).

```
BT.ustat <- BuyseTest(trt ~ tte(time, threshold = 20, status = "status"),
                      data = veteran, trace = 0, method.inference = "u-statistic")
summary(BT.ustat)
```

```
      Generalized pairwise comparisons with 1 endpoint

 - statistic       : net treatment benefit  (delta: endpoint specific, Delta: global)
 - null hypothesis : Delta == 0
 - confidence level: 0.95
 - inference       : H-projection of order 1 after atanh transformation
 - treatment groups: Exp (treatment) vs. Pl (control)
 - censored pairs  : probabilistic score based on the survival curves
 - results
 endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   Delta
     time        20      100        37.78          46.54      15.68        0 -0.0877
 CI [2.5% ; 97.5%] p.value
  [-0.2735;0.1045] 0.37162
```

The first two approaches require simulating a large number of samples and applying the GPC to each of these samples. The `seed` argument is used to generate a seed for each sample. The number of samples is set using the arugment `n.resampling` and it should large enough to limit the Monte Carlo error when estimating the p-value. Typically should be at least 10000 to get, roughtly, 2-digit precision, as examplified below:

```
set.seed(10)
sapply(1:10, function(i){mean(rbinom(1e4, size = 1, prob = 0.05))})
```

```
[1] 0.0511 0.0491 0.0489 0.0454 0.0516 0.0522 0.0468 0.0483 0.0491 0.0508
```

Indeed, here we get a reasonnable approximation of `0.05` (if we round and only keep 2 digits). Note that to get 3 digits precision we would need more samples. The last method does not rely on resampling but on the computation of the influence function of the estimator. Fortunately, when using the Gehan's scoring rule, this does not really involve any extra-calculations and this is therefore very fast to perform. When using the Peron's scoring rule, more serious extra-calculations are involved so the computation time is expected to increase by a factor 5 to 10 compared to the point estimate alone (i.e. `method.inference` equal to `"none"`).

It is possible to relax the exchangeability assumption using a studentized permutation. A studentized bootstrap is also possible to improve on the better small samples properties of the bootstrap confidence intervals. Both rely on the asymptotic approach to estimate standard errors and are more numerically intensive.

1.7 What if smaller is better?

By default `BuyseTest` will always assume that higher values of an endpoint are favorable. This behavior can be changed by specifying `operator = "<0"` for an endpoint:

```
ffop <- trt ~ tte(time, status = "status", threshold = 20, operator = "<0")
BTinv <- BuyseTest(ffop, data = veteran, trace = 0)
summary(BTinv)
```

```
      Generalized pairwise comparisons with 1 endpoint

 - statistic       : net treatment benefit  (delta: endpoint specific, Delta: global)
 - null hypothesis : Delta == 0
 - confidence level: 0.95
 - inference       : H-projection of order 1 after atanh transformation
 - treatment groups: Exp (treatment) vs. Pl (control)
 - censored pairs  : probabilistic score based on the survival curves
 - results
 endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)  Delta
     time        20      100        46.54          37.78      15.68        0 0.0877
 CI [2.5% ; 97.5%] p.value
  [-0.1045;0.2735] 0.37162
```

Internally `BuyseTest` will compute the favorable and unfavorable score as usual and then switch them around if the operator equals `"<0"`.

1.8 Stopping comparison for neutral pairs

In presence of neutral pairs, `BuyseTest` will, by default, continue the comparison on the endpoints with lower priority. For instance let consider a dataset with one observation in each treatment arm:

```
dt.sim <- data.table(Id = 1:2,
                     treatment = c("Yes","No"),
                     tumor = c("Yes","Yes"),
                     size = c(15,20))
dt.sim
```

```
      Id treatment  tumor  size
   <int>    <char> <char> <num>
1:     1       Yes    Yes    15
2:     2        No    Yes    20
```

If we use the GPC with tumor as the first endpoint and size as the second endpoint:

```
BT.pair <- BuyseTest(treatment ~ bin(tumor) + cont(size, operator = "<0"), data = dt.sim,
                     trace = 0, method.inference = "none")
summary(BT.pair)
```

```
      Generalized pairwise comparisons with 2 prioritized endpoints

- statistic       : net treatment benefit  (delta: endpoint specific, Delta: global)
- treatment groups: Yes (treatment) vs. No (control)
- neutral pairs   : re-analyzed using lower priority endpoints
- results
endpoint total(%) favorable(%) unfavorable(%) neutral(%) uninf(%) delta Delta
   tumor      100            0              0        100        0     0     0
    size      100          100              0          0        0     1     1
```

the outcome of the comparison is neutral for the first priority, but favorable for the second. Setting the argument `neutral.as.uninf` to `FALSE` will stop the comparison when a pair is classified as neutral:

```
BT.pair2 <- BuyseTest(treatment ~ bin(tumor) + cont(size, operator = "<0"), data = dt.sim,
                     trace = 0, method.inference = "none", neutral.as.uninf = FALSE)
summary(BT.pair2)
```

```
      Generalized pairwise comparisons with 2 prioritized endpoints

- statistic       : net treatment benefit  (delta: endpoint specific, Delta: global)
- treatment groups: Yes (treatment) vs. No (control)
- neutral pairs   : ignored at lower priority endpoints
- results
endpoint total(%) favorable(%) unfavorable(%) neutral(%) uninf(%) delta Delta
   tumor      100            0              0        100        0     0     0
    size        0            0              0          0        0     0     0
```

So in this case no pair is analyzed at second priority.

1.9 Is multiple testing a concern with GPC?

Yes, as with any other statistical method. Having a pre-defined statistical plan (i.e. written before looking at the data) specifying the hierarchy of endpoints, their threshold of clinical relevance is recommanded. When planning multiple GPC, summarize the results can be done via one of two principles:

- **intersection union principle**: one rejects the (global) null hypothesis if there is evidence for an effect in all the GPC analyses. This is typically a sensitivity analysis: checking that the results are not too sensitive to the choice of an hyperparameter. No multiplicity adjustment is needed other than considering the largest p-value among all tests. For instance, when checking whether the estimated net treatment benefit is similar across a range of threshold of clincial relevance, we would obtain a p-value of 0.76

```
BTse <- sensitivity(BT.ustat, threshold = seq(0,500, length.out=10),
                            trace = FALSE)
BTse
```

```
        time    estimate         se    lower.ci   upper.ci null   p.value
1    0.00000 -0.08752774 0.10041203 -0.27851884 0.11012263    0 0.3858177
2   55.55556 -0.08095829 0.08957699 -0.25229456 0.09530004    0 0.3682107
3  111.11111 -0.03170177 0.07463991 -0.17629003 0.11422560    0 0.6712414
4  166.66667  0.01896964 0.06452954 -0.10713643 0.14447503    0 0.7688360
5  222.22222  0.03315614 0.05523512 -0.07506821 0.14060850    0 0.5486177
6  277.77778  0.04217485 0.04654025 -0.04914025 0.13279075    0 0.3653982
7  333.33333  0.04112991 0.03946828 -0.03631838 0.11808708    0 0.2979105
8  388.88889  0.04075638 0.03300933 -0.02402114 0.10519310    0 0.2174545
9  444.44444  0.04097871 0.03027888 -0.01844156 0.10011054    0 0.1764199
10 500.00000  0.03517173 0.02769280 -0.01915553 0.08929191    0 0.2044340
```

- **union intersection principle**: one rejects the (global) null hypothesis if there is evidence for an effect for at least on of the GPC analyses. This is a typical exploratory analysis where one look for the most promising outcome. Adjustment for multiplicity is needed. Since estimates from GPC procedure are typically highly correlated, one can improve on bonferroni adjustment using a max-test adjustment. This is what is performed via the `BuyseMultComp` function:

```
BuyseMultComp(BT.H, endpoint = 1:2)
```

```
  - Univariate tests:
           estimate         se   lower.ci   upper.ci null  p.value lower.band upper.band
time_t20 -0.08765836 0.09760901 -0.2735301 0.10452446    0 0.371617 -0.2798817  0.1113226
karno    -0.10092285 0.09971277 -0.2901336 0.09588144    0 0.314777 -0.2965716  0.1028561
         adj.p.value
time_t20   0.4117239
karno      0.3508339
```

Here we look at whether there is a benefit in survival alone (first priority `time_t20`) or a benefit over both endpoint (second priority `karno`). Setting the argument `cumulative` to `FALSE` when considering non-hierarchical GPC analyses enables to efficiently adjust endpoint-specific GPC for multiple comparisons:

```
BuyseMultComp(BT.nH, cumulative = FALSE, endpoint = 1:2)
```

```
  - Univariate tests:
           estimate         se   lower.ci  upper.ci null   p.value lower.band upper.band
time_t20 -0.08765836 0.09760901 -0.2735301 0.1045245    0 0.3716170 -0.2953329  0.1279261
karno    -0.03132992 0.09787113 -0.2197111 0.1593037    0 0.7490407 -0.2420777  0.1822409
         adj.p.value
time_t20   0.5597555
karno      0.9236602
```

One can also consider the global endpoint of two different GPC analyses:

```
BuyseMultComp(list(hierarchical = BT.H, Obrien = BT.nH), cluster = "id")
```

```
  - Univariate tests:
               estimate         se   lower.ci   upper.ci null   p.value lower.band
hierarchical -0.10092285 0.09971277 -0.2901336 0.09588144    0 0.3147770 -0.3014645
Obrien       -0.05949414 0.08700807 -0.2266953 0.11111326    0 0.4951361 -0.2368800
             upper.band adj.p.value
hierarchical  0.1081696   0.3831444
Obrien        0.1217304   0.5851872
```

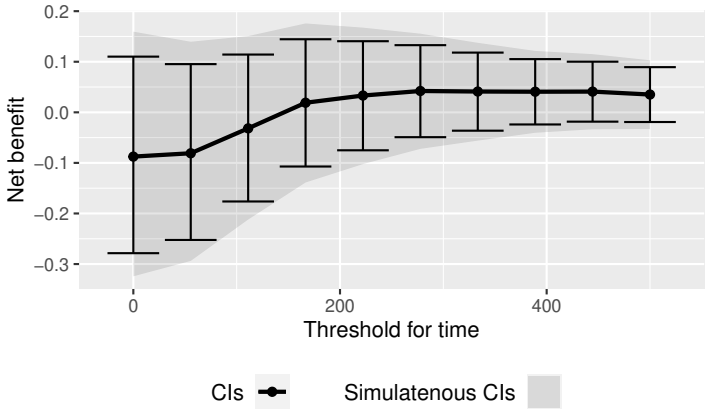
Finally the `sensitivity` method can also be used to adjust for multiple comparisons over multiple thresholds:

```
BTse.ustat <- sensitivity(BT.ustat, threshold = seq(0,500, length.out=10),
                          band = TRUE, adj.p.value = TRUE, seed = 10, trace = FALSE)
BTse.ustat[,c("time","estimate",
              "lower.ci","upper.ci","p.value",
              "lower.band","upper.band","adj.p.value")]
```

```
        time    estimate    lower.ci   upper.ci   p.value  lower.band upper.band adj.p.value
1    0.00000 -0.08752774 -0.27851884 0.11012263 0.3858177 -0.32450860  0.1597923   0.7746620
2   55.55556 -0.08095829 -0.25229456 0.09530004 0.3682107 -0.29401340  0.1397613   0.7528122
3  111.11111 -0.03170177 -0.17629003 0.11422560 0.6712414 -0.21223939  0.1509285   0.9810295
4  166.66667  0.01896964 -0.10713643 0.14447503 0.7688360 -0.13892698  0.1759257   0.9969925
5  222.22222  0.03315614 -0.07506821 0.14060850 0.5486177 -0.10250127  0.1676028   0.9257172
6  277.77778  0.04217485 -0.04914025 0.13279075 0.3653982 -0.07236883  0.1556205   0.7492675
7  333.33333  0.04112991 -0.03631838 0.11808708 0.2979105 -0.05604663  0.1375345   0.6544816
8  388.88889  0.04075638 -0.02402114 0.10519310 0.2174545 -0.04053858  0.1215153   0.5206881
9  444.44444  0.04097871 -0.01844156 0.10011054 0.1764199 -0.03359858  0.1151022   0.4429140
10 500.00000  0.03517173 -0.01915553 0.08929191 0.2044340 -0.03301187  0.1030295   0.4967546
```

Here by setting the argument `band` to `TRUE` (and `adj.p.value` to `TRUE`), we obtain confidence intervals (and p-values) adjusted for multiple comparisons. Said otherwise, the columns `lower.ci` and `upper.ci` provide a (pointwise) confidence interval with 95% coverage for a given threshold while the columns `lower.band` and `upper.band` provide a (simutaneous) confidence interval with 95% coverage across all given thresholds. The difference can be visualized using the `autoplot` method:

```
library(ggplot2)
autoplot(BTse.ustat)
```



Simultaneous and pointwise confidence intervals are here of similar width due to the very high correlation between estimates across thresholds:

```
BTse.cor <- cor(lava::iid(BTse.ustat))
range(BTse.cor[lower.tri(BTse.cor)])
```

```
[1] 0.3716902 0.9848999
```

Note that with multiple endpoints, the thresholds can be specified using a list:

```
BTse.H <- sensitivity(BT.H, trace = FALSE,
                      threshold = list(time = seq(0,500,length = 10), karno = c(0,40,80)))
head(BTse.H)
```

```
       time karno    estimate         se   lower.ci   upper.ci null   p.value
1   0.00000     0 -0.08754474 0.10044847 -0.2786016 0.11017738    0 0.3858987
2  55.55556     0 -0.11177487 0.09915501 -0.2995661 0.08435417    0 0.2636263
3 111.11111     0 -0.08618872 0.09822940 -0.2732475 0.10715096    0 0.3826244
4 166.66667     0 -0.05180121 0.09818252 -0.2400240 0.14017526    0 0.5984319
5 222.22222     0 -0.03668720 0.09810141 -0.2253052 0.15458146    0 0.7086747
6 277.77778     0 -0.02906324 0.09773146 -0.2172647 0.16122161    0 0.7663054
```

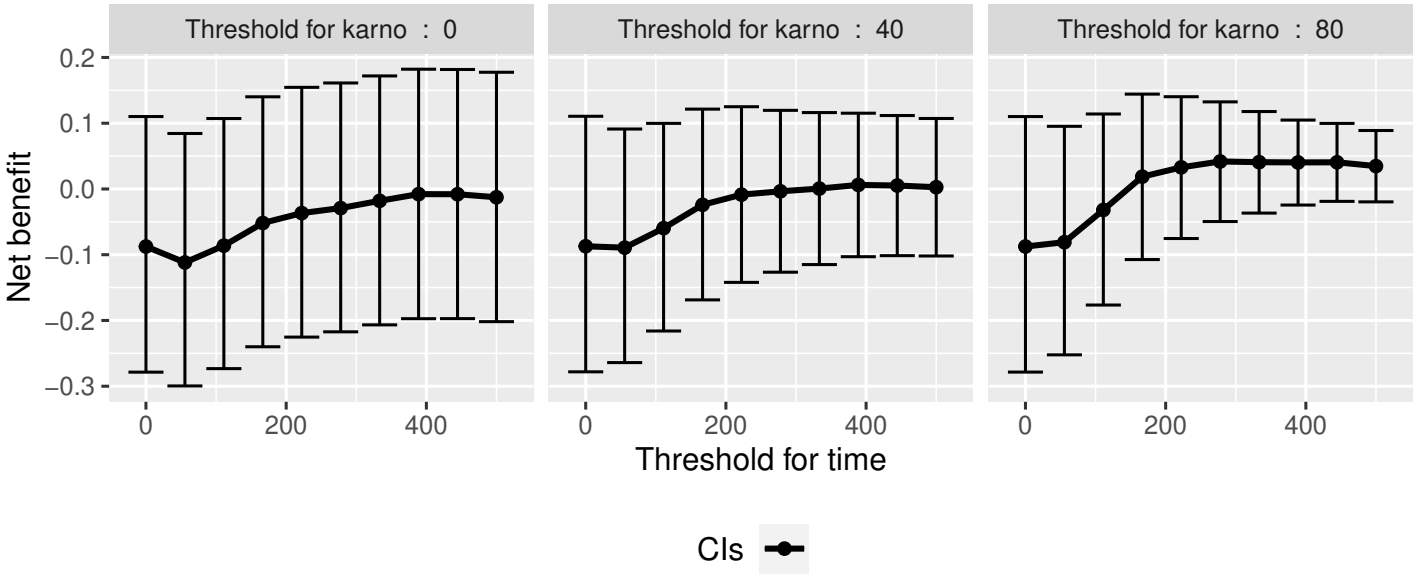
or a matrix:

```
grid <- expand.grid(list("time_t20" = seq(0,500,length = 10), "karno" = c(0,40,80)))
cbind(head(grid)," " = "  ...   ",tail(grid))
BTse.H2 <-sensitivity(BT.H, threshold = grid, trace = FALSE)
range(BTse.H-BTse.H2)
```

```
   time_t20 karno          time_t20 karno
1   0.00000     0   ...    222.2222    80
2  55.55556     0   ...    277.7778    80
3 111.11111     0   ...    333.3333    80
4 166.66667     0   ...    388.8889    80
5 222.22222     0   ...    444.4444    80
6 277.77778     0   ...    500.0000    80
[1] 0 0
```

The latter should be used when the same endpoint is used at different priorities (each column correspond to the threshold that should be used at a priority). As before we can display the results using the autoplot function:

```
autoplot(BTse.H, col = NA)
##  alternative display:
## autoplot(BTse.H, position  = position_dodge(width = 15))
```



The autoplot function can only be used when 1 or 2 thresholds are varied at the same time.

2 Getting additional inside: looking at the pair level

So far we have looked at the overall score and probabilities. But it is also possible to extract the score relative to each pair, as well as to "manually" compute this score. This can give further inside on what the software is actually doing and what is the contribution of each individual on the evaluation of the treatment.

2.1 Extracting the contribution of each pair to the statistic

The net treatment benefit or the win ratio statistics can be expressed as a sum of a score over all pairs of patients. The argument `keep.pairScore` enables to export the score relative to each pair in the output of BuyseTest:

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

The method `getPairScore` can then be used to extract the contribution of each pair. For instance the following code extracts the contribution for the first endpoint:

```
getPairScore(BT.keep, endpoint = 1)
```

```
Key: <index.Exp, index.Pl>
      index.Pl index.Exp favorable unfavorable neutral uninf weight
         <num>     <num>     <num>       <num>   <num> <num>  <num>
   1:        1        70         1           0       0     0      1
   2:        2        70         1           0       0     0      1
   3:        3        70         1           0       0     0      1
   4:        4        70         1           0       0     0      1
   5:        5        70         1           0       0     0      1
  ---
4688:       65       137         0           1       0     0      1
4689:       66       137         0           1       0     0      1
4690:       67       137         0           1       0     0      1
4691:       68       137         0           1       0     0      1
4692:       69       137         0           1       0     0      1
```

Each line corresponds to different comparison between a pair from the control arm and the treatment arm. The column `strata` store to which strata the pair belongs (first, second, ...). The columns favorable, unfavorable, neutral, uninformative contains the result of the comparison, e.g. the first pair was classified as favorable while the last was classified as favorable with a weight of 1. The second and third columns indicates the rows in the original dataset corresponding to the pair:

```
veteran[c(70,1),]
```

```
   id trt celltype time status karno diagtime age prior
70 70 Exp squamous  999      1    90       12  54    10
1   1  Pl squamous   72      1    60        7  69     0
```

For the first pair, the event was observed for both observations and since $999 > 72 + 20$ the pair is rated favorable. Substracting the average probability of the pair being favorable minus the average probability of the pair being unfavorable:

```
getPairScore(BT.keep, endpoint = 1)[, mean(favorable) - mean(unfavorable)]
```

```
[1] -0.08765836
```

gives the net treatment benefit in favor of the treatment for the first endpoint:

```
BT.keep
```

```
endpoint threshold   delta   Delta
    time        20 -0.0877 -0.0877
   karno           -0.0133 -0.1009
```

More examples and explanation can be found in the documentation of the method `getPairScore`.

2.2 Extracting the survival probabilities

When using `scoring.rule` equals `"Peron"`, survival probabilities at event time, and event times +/- threshold in the control and treatment arms are used to score the pair. Setting `keep.survival` to `TRUE` and `precompute` to `FALSE` in BuyseTest.options enables to export the survival probabilities in the output of BuyseTest:

```
BuyseTest.options(keep.survival = TRUE, precompute = FALSE)
BT.keep2 <- BuyseTest(trt ~ tte(time, threshold = 20, status = "status") + cont(karno),
                      data = veteran, keep.pairScore = TRUE, scoring.rule = "Peron",
                      trace = 0, method.inference = "none")
```

The method `getSurvival` can then be used to extract these survival probabilities. For instance the following code extracts the survival for the first endpoint:

```
outSurv <- getSurvival(BT.keep2, endpoint = 1, strata = 1)
str(outSurv)
```

```
List of 5
 $ survTimeC: num [1:69, 1:13] 72 411 228 126 118 10 82 110 314 100 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:13] "time" "survivalC-threshold" "survivalC_0" "survivalC+threshold" ...
 $ survTimeT: num [1:68, 1:13] 999 112 87 231 242 991 111 1 587 389 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:13] "time" "survivalC-threshold" "survivalC_0" "survivalC+threshold" ...
 $ survJumpC: num [1:57, 1:6] 3 4 7 8 10 11 12 13 16 18 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:6] "time" "survival" "dSurvival" "index.survival" ...
```

```
 $ survJumpT: num [1:51, 1:6] 1 2 7 8 13 15 18 19 20 21 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:6] "time" "survival" "dSurvival" "index.survival" ...
 $ lastSurv : num [1:2] 0 0
```

2.2.1 Computation of the score with only one censored event

Let's look at pair 91:

```
getPairScore(BT.keep2, endpoint = 1, rm.withinStrata = FALSE)[91]
```

```
Key: <index.Exp, index.Pl>
   index.Pl index.Exp indexWithinStrata.Pl indexWithinStrata.Exp favorable unfavorable
      <num>     <num>                <num>                 <num>     <num>       <num>
1:       22        71                   22                     2         0   0.6950827
     neutral uninf weight
       <num> <num>  <num>
1: 0.3049173     0      1
```

In the dataset this corresponds to:

```
veteran[c(22,71),]
```

```
   id trt  celltype time status karno diagtime age prior
22 22  Pl smallcell   97      0    60        5  67     0
71 71 Exp  squamous  112      1    80        6  60     0
```

The observation from the control group is censored at 97 while 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 $\frac{S_c(97)}{S_c(112+20)}$. The survival at the event time in the censoring group is stored in survTimeC. Since observation 22 is the 22th observation in the control group:

```
iSurv <- outSurv$survTimeC[22,]
iSurv
```

```
                      time       survivalC-threshold               survivalC_0
                97.0000000                 0.5615232                 0.5171924
      survivalC+threshold       survivalT-threshold               survivalT_0
                0.4235463                 0.4558824                 0.3643277
      survivalT+threshold index.survivalC-threshold        index.survivalC_0
                0.2827500                25.0000000                28.0000000
index.survivalC+threshold index.survivalT-threshold        index.survivalT_0
                33.0000000                27.0000000                32.0000000
index.survivalT+threshold
                35.0000000
```

Since we are interested in the survival in the control arm exactly at the event time:

```
Sc97 <- iSurv["survivalC_0"]
Sc97
```

```
survivalC_0
  0.5171924
```

The survival at the event time in the treatment group is stored in survTimeC. Since observation 71 is the 2nd observation in the treatment group:

```
iSurv <- outSurv$survTimeT[2,] ## survival at time 112+20
iSurv
```

```
                    time       survivalC-threshold               survivalC_0
            112.0000000                 0.5319693                 0.4549201
      survivalC+threshold       survivalT-threshold               survivalT_0
              0.3594915                 0.3801681                 0.2827500
      survivalT+threshold index.survivalC-threshold        index.survivalC_0
              0.2827500                27.0000000                32.0000000
index.survivalC+threshold index.survivalT-threshold        index.survivalT_0
             37.0000000                31.0000000                35.0000000
index.survivalT+threshold
             35.0000000
```

Since we are interested in the survival in the control arm at the event time plus threshold:

```
Sc132 <- iSurv["survivalC+threshold"]
Sc132
```

```
survivalC+threshold
          0.3594915
```

The probability in favor of the control is then:

```
Sc132/Sc97
```

```
survivalC+threshold
          0.6950827
```

2.2.2 Computation of the score with two censored events

When both observations are censored, the formula for computing the probability in favor of treatment or control involves an integral. This integral can be computed using the function `calcIntegralSurv\_cpp` that takes as argument a matrix containing the survival and the jumps in survival, e.g.:

```
head(outSurv$survJumpT)
```

```
     time  survival   dSurvival index.survival index.dsurvival1 index.dsurvival2
[1,]    1 0.7681159 -0.02941176             12                 0                 1
[2,]    2 0.7536232 -0.01470588             13                 1                 2
[3,]    7 0.7388463 -0.02941176             14                 2                 3
[4,]    8 0.7388463 -0.02941176             14                 3                 4
[5,]   13 0.7092924 -0.01470588             16                 4                 5
[6,]   15 0.6945155 -0.02941176             17                 5                 6
```

and the starting time of the integration time. For instance, let's look at pair 148:

```
getPairScore(BT.keep2, endpoint = 1, rm.withinStrata = FALSE)[148]
```

```
Key: <index.Exp, index.Pl>
   index.Pl index.Exp indexWithinStrata.Pl indexWithinStrata.Exp favorable unfavorable
      <num>     <num>                <num>                 <num>     <num>       <num>
1:       10        72                   10                     3 0.5058685   0.3770426
     neutral uninf weight
       <num> <num>  <num>
1: 0.1170889     0      1
```

which corresponds to the observations:

```
veteran[c(10,72),]
```

```
   id trt celltype time status karno diagtime age prior
10 10  Pl squamous  100      0    70        6  70     0
72 72 Exp squamous   87      0    80        3  48     0
```

The probability in favor of the treatment (p_F) and control (p_{UF}) can be computed as:

$$p_F = -\frac{1}{S_T(x)S_C(y)} \int_{t>y} S_T(t+\tau) dS_C(t)$$
$$p_{UF} = -\frac{1}{S_T(x)S_C(y)} \int_{t>x} S_C(t+\tau) dS_T(t)$$

where $x = 87$ and $y = 100$. To ease the call of `calcIntegralScore_cpp` we create a warper:

```
calcInt <- function(...){ ## no need for the functionnal derivative of the score
    BuyseTest:::.calcIntegralSurv_cpp(...,
                                      returnDeriv = FALSE,
                                      derivSurv = matrix(0),
                                      derivSurvD = matrix(0))
}
```

and then call it to compute the probabilities:

```
denom <- as.double(outSurv$survTimeT[3,"survivalT_0"] * outSurv$survTimeC[10,"survivalC_0"])
M <- cbind("favorable" = -calcInt(outSurv$survJumpC, start = 100,
                                  lastSurv = outSurv$lastSurv[2],
                                  lastdSurv = outSurv$lastSurv[1])/denom,
           "unfavorable" = -calcInt(outSurv$survJumpT, start = 87,
                                    lastSurv = outSurv$lastSurv[1],
                                    lastdSurv = outSurv$lastSurv[2])/denom)
rownames(M) <- c("lowerBound","upperBound")
M
```

```
           favorable unfavorable
lowerBound 0.5058685   0.3770426
upperBound 0.5058685   0.3770426
```

Note: the lower bound is identical to the upper bound as we could estimate the full survival curve:

```
outSurv$lastSurv
```

```
[1] 0 0
```

3 Dealing with missing values or/and right censoring

In presence of censoring or missing values, it is often not be possible to classify all pairs without a model for the censoring mechanism. The unclassified pairs, called uninformative, have a score of 0 which will typically bias the estimate of the net net treatment benefit towards 0 [5]. Consider the following dataset:

```
set.seed(10)
dt <- simBuyseTest(1e2, latent = TRUE, argsCont = NULL,
                   argsTTE = list(scale.T = 1/2, scale.C = 1,
                                  scale.censoring.C = 1, scale.censoring.T = 1))
dt[, eventtimeCensoring := NULL]
dt[, status1 := 1]
head(dt)
```

```
      id treatment eventtimeUncensored eventtime status toxicity eta_toxicity status1
   <num>    <fctr>               <num>     <num>  <num>   <fctr>        <num>   <num>
1:     1         C           0.2135567 0.2135567      1      yes  -0.07945702       1
2:     2         C           0.3422379 0.3422379      1       no   1.18175155       1
3:     3         C           1.3933222 1.3933222      1       no   2.18614406       1
4:     4         C           0.6737702 0.1961599      0       no   0.40617493       1
5:     5         C           0.5642992 0.5642992      1      yes  -0.73835910       1
6:     6         C           1.1039218 0.1764950      0      yes  -1.95648670       1
```

where we have the uncensored event times (`eventtimeUncensored`) as well as the censored event times (`eventtime`). The percentage of censored observations is:

```
100*dt[,mean(status==0)]
```

```
[1] 44
```

We would like to be able to recover the net treatment benefit estimated with the uncensored event times:

```
BuyseTest(treatment ~ tte(eventtimeUncensored, status1, threshold = 0.5),
          data = dt,
          scoring.rule = "Gehan", method.inference = "none", trace = 0)
```

```
           endpoint threshold  Delta
eventtimeUncensored       0.5 -0.271
```

using the censored survival times.

[5] While the power is typically reduced, the type 1 error will still be controled if censoring is at random

The `BuyseTest` function handles missing values via two arguments:

- `scoring.rule` indicates how pairs involving missing data are compared.
  - **the Gehan's scoring rule** compares the observed values. If it is not possible to decide whether one observation has a better endpoint than the other (e.g. because both are right-censoring) then the paired is scored uninformative.
  - **the Peron's scoring rule** compares the probabilty of one observation having a better endpoint than the other given the observed values. This require a model for the censoring distribution. If the full survival curve can be identified then all pairs can be fully classified otherwise some of the pair will be partially uninformative.
  - **the Efron's scoring rule** 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.
- `correction.uninf` indicates what to do with the uninformative scores. For instance setting this argument to `TRUE` will re-distribute this score to favorable/unfavorable/neutral scores.

The Peron's scoring rule is the default (and recommanded) approach. It uses a Kaplan Meier estimator stratified on treatment and GPC `strata` variable (if any) as survival model. When the last observation is censored, then part of the survival curve is unknown which can be necessary to score some of the pairs (especially in presence of a threshold of clinical relevance). One can:

- use a restriction time within the time interval where the survival curve can be estimated for each group.
- still use the default Peron's scoring rule: this will lead to uninformative pairs which can be re-classified based on a lower priority endpoint.
- use the Peron's scoring rule with another survival model, using parametric assumptions to inform about the unknown part of the survival curve. This can be achieved via the `model.tte` argument or using the Efron's scoring rule.
- use an add-hoc correction for the uninformative pairs (`correction.uninf`)

The first two solutions lead to a change of estimand, the first being much more clearly defined than the second. The last two solutions correspond to make statistical assumptions, the former assumptions being more explicit than with the later solution.

3.1 Gehan's scoring rule

In the example, Gehan's scoring rule:

```
e.G <- BuyseTest(treatment ~ tte(eventtime, status, threshold = 0.5),
          data = dt, scoring.rule = "Gehan", trace = 0)
model.tables(e.G)
```

```
   endpoint threshold total favorable unfavorable neutral uninf   Delta   lower.ci
1 eventtime       0.5   100      4.67       14.39   20.44  60.5 -0.0972 -0.1593869
     upper.ci     p.value
1 -0.03424474 0.002514882
```

leads to many uninformative pairs (about 60%) and an estimate much closer to 0 than the truth.

3.2 Peron's scoring rule

In the example, Peron's scoring rule:

```
e.P <- BuyseTest(treatment ~ tte(eventtime, status, threshold = 0.5),
          data = dt, scoring.rule = "Peron", trace = 0)
model.tables(e.P)
```

```
   endpoint threshold total favorable unfavorable  neutral    uninf      Delta   lower.ci
1 eventtime       0.5   100   11.1737    43.33707 44.12373 1.365504 -0.3216337 -0.4584262
    upper.ci      p.value
1 -0.1699543 5.385074e-05
```

leads to no uninformative pairs. Indeed the last observation in each group is an (uncensored) event:

```
dt[,.SD[which.max(eventtime)],by="treatment"]
```

```
   treatment    id eventtimeUncensored eventtime status toxicity eta_toxicity status1
      <fctr> <num>               <num>     <num>  <num>   <fctr>        <num>   <num>
1:         C    72            2.668629  2.668629      1      yes   -1.9256436       1
2:         T   154            1.674053  1.588657      0      yes   -0.8647272       1
```

so the full survival curve could be identified. As a result the estimate is very close to the truth.

Note 1: the censoring model can be specified by first fitting a survival model (`prodlim` or `survreg`) for the survival time:

```
library(prodlim)
e.prodlim <- prodlim(Hist(eventtime, status) ~ treatment, data = dt)
```

Then passing the model to the `BuyseTest` via the `model.tte` argument:

```
e.P1 <- BuyseTest(treatment ~ tte(eventtime, status, threshold = 0.5),
                  model.tte = e.prodlim,
                  data = dt, scoring.rule = "Peron", trace = 0)
model.tables(e.P1)
```

```
   endpoint threshold total favorable unfavorable  neutral    uninf      Delta   lower.ci
1 eventtime       0.5   100   11.1737    43.33707 44.12373 1.365504 -0.3216337 -0.4584262
    upper.ci      p.value
1 -0.1699543 5.385074e-05
```

When the dataset used to fit the survival model match the one used to run the GPC procedure, the overall uncertainty will be computed. Otherwise:

```
dt2 <- dt[order(dt$eventtime)]
e.P2 <- BuyseTest(treatment ~ tte(eventtime, status, threshold = 0.5),
                  model.tte = prodlim(Hist(eventtime, status) ~ treatment, data = dt2),
                  data = dt, scoring.rule = "Peron", trace = 0)
model.tables(e.P2)
```

```
Uncertainty related to the estimation of the survival probabilities is ignored.
Consider adding an attribute "iidNuisance" to the argument 'model.tte' taking value TRUE to change
   endpoint threshold total favorable unfavorable  neutral    uninf      Delta   lower.ci
1 eventtime       0.5   100   11.1737    43.33707 44.12373 1.365504 -0.3216337 -0.4187087
    upper.ci      p.value
1 -0.2172912 6.570106e-09
```

the survival probabilities will assumed to be known with infinite precision and only the uncertainty of the GPC procedure will be considered. Add-hoc modification of the data can be used to obtain 'conservative' estimates when considering a single endpoint, e.g.:

```
dt2[, last := (max(eventtime)==eventtime), by = "treatment"]
## survival stays constant after end of follow-up
dt2[treatment=="C" & last == TRUE, c("eventtime","status") := .(max(dt2$eventtime)+1,1)]
## survival drop to 0 after end of follow-up
dt2[treatment=="T" & last == TRUE, status := 1]
## modified Kaplan Meier estimator
e.prodlim2 <- prodlim(Hist(eventtime, status) ~ treatment, data = dt2)
attr(e.prodlim2, "iidNuisance") <- TRUE
## run GPC
e.P3 <- BuyseTest(treatment ~ tte(eventtime, status, threshold = 0.5),
                  model.tte = e.prodlim2,
                  data = dt, scoring.rule = "Peron", trace = 0)
model.tables(e.P3) ## even more unfavorable to treatment
```

```
   endpoint threshold total favorable unfavorable  neutral uninf      Delta   lower.ci
1 eventtime       0.5   100   11.1737    43.97856 44.84774     0 -0.3280486 -0.4378751
    upper.ci     p.value
1 -0.2085751 2.25273e-07
```

Note 2: it is possible to use a parametric model via the `survreg` function:

```
library(survival)
e.survreg <- survreg(Surv(eventtime, status) ~ treatment, data = dt,
                     dist = "weibull")
```

Then passing the model to the `BuyseTest` via the `model.tte` argument:

```
e.P3 <- BuyseTest(treatment ~ tte(eventtime, status, threshold = 0.5),
                  model.tte = e.survreg,
                  data = dt, scoring.rule = "Peron", trace = 0)
model.tables(e.P3)
```

```
   endpoint threshold total favorable unfavorable  neutral      uninf      Delta   lower.ci
1 eventtime       0.5   100  11.65444    34.18937 54.14472 0.01147085 -0.2253494 -0.3476693
     upper.ci      p.value
1 -0.09548719 0.0007624659
```

Internally the survival curve is discretized using 1000 points starting from survival = 1 to survival = 0.001 (this is why there is a non-0 but small percentage of uninformative pairs). This is performed internally by applying the `BuyseTTEM` method. Another discretisation can be obtained by calling `BuyseTTEM` with another value for the `n.grid` argument:

```
e.TTEM <- BuyseTTEM(e.survreg, treatment = "treatment", iid = TRUE, n.grid = 2500)
str(e.TTEM$peron$jumpSurvHaz[[1]][[1]])
```

```
'data.frame':        2500 obs. of  3 variables:
 $ index.jump: logi  NA NA NA NA NA NA ...
 $ time.jump : num  0 0.000307 0.000632 0.000964 0.001301 ...
 $ survival  : num  1 1 0.999 0.999 0.998 ...
```

and then passing to `BuyseTest`:

```
e.P4 <- BuyseTest(treatment ~ tte(eventtime, status, threshold = 0.5),
                  model.tte = e.TTEM,
                  data = dt, scoring.rule = "Peron", trace = 0)
model.tables(e.P4)
```

```
   endpoint threshold total favorable unfavorable  neutral       uninf      Delta   lower.ci
1 eventtime       0.5   100  11.64894    34.18631 54.16019 0.004558007 -0.2253737 -0.3476861
     upper.ci      p.value
1 -0.09551899 0.0007609754
```

It is therefore possible to extend the approach to other model by defining an appropriate `BuyseTTEM` method. Looking at the code use for defining `BuyseTTEM.survreg` can be helpful.

3.3 Correction via re-weighting

The weights of the non-informative pairs is redistributed to the informative pairs. This is only a good strategy when there are no neutral pairs or there are no lower priority endpoints. This gives an estimate much closer to the true net treatment benefit:

```
BT <- BuyseTest(treatment ~ tte(eventtime, status, threshold = 0.5),
                data = dt, keep.pairScore = TRUE, trace = 0,
                scoring.rule = "Gehan", method.inference = "none", correction.uninf = 2)
summary(BT)
```

```
      Generalized pairwise comparisons with 1 endpoint

- statistic       : net treatment benefit  (delta: endpoint specific, Delta: global)
- treatment groups: T (treatment) vs. C (control)
- censored pairs  : deterministic score or uninformative
- uninformative pairs: no contribution, their weight is passed to the informative pairs using IPCW
- results
 endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   Delta
eventtime       0.5      100        11.82          36.43      51.75        0 -0.2461
```

We can also see that no pair is finally classified as non informative. To get some inside about the correction we can look at the scores of the pairs:

```
iScore <- getPairScore(BT, endpoint = 1)
```

To get a synthetic view, we only look at the unique favorable/unfavorable/neutral/uniformative results:

```
iScore[,.SD[1],
       .SDcols = c("favorableC","unfavorableC","neutralC","uninfC"),
       by = c("favorable","unfavorable","neutral","uninf")]
```

```
   favorable unfavorable neutral uninf favorableC unfavorableC neutralC uninfC
       <num>       <num>   <num> <num>      <num>        <num>    <num>  <num>
1:         0           0       1     0   0.000000     0.000000 2.531646      0
2:         0           1       0     0   0.000000     2.531646 0.000000      0
3:         0           0       0     1   0.000000     0.000000 0.000000      0
4:         1           0       0     0   2.531646     0.000000 0.000000      0
```

We can see that the favorable/unfavorable/neutral pairs have seen their contribution multiplied by:

```
iScore[,1/mean(favorable + unfavorable + neutral)]
```

```
[1] 2.531646
```

i.e. the inverse probability of being informative.

3.4 Correction at the pair level

Another possible correction is to distribute the non-informative weight of a pair to the average favorable/unfavorable/neutral probability observed on the sample:

```
BT <- BuyseTest(treatment ~ tte(eventtime, status, threshold = 0.5),
                data = dt, keep.pairScore = TRUE, trace = 0,
                scoring.rule = "Gehan", method.inference = "none", correction.uninf = TRUE)
summary(BT)
```

```
      Generalized pairwise comparisons with 1 endpoint

- statistic       : net treatment benefit  (delta: endpoint specific, Delta: global)
- treatment groups: T (treatment) vs. C (control)
- censored pairs  : deterministic score or uninformative
- uninformative pairs: score equals the averaged score of all informative pairs
- results
 endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   Delta
eventtime       0.5      100        11.82          36.43      51.75        0 -0.2461
```

Looking at the scores of the pairs:

```
iScore <- getPairScore(BT, endpoint = 1)
iScore[,.SD[1],
       .SDcols = c("favorableC","unfavorableC","neutralC","uninfC"),
       by = c("favorable","unfavorable","neutral","uninf")]
```

```
   favorable unfavorable neutral uninf favorableC unfavorableC  neutralC uninfC
       <num>       <num>   <num> <num>      <num>        <num>     <num>  <num>
1:         0           0       1     0  0.0000000    0.0000000 1.0000000      0
2:         0           1       0     0  0.0000000    1.0000000 0.0000000      0
3:         0           0       0     1  0.1182278    0.3643038 0.5174684      0
4:         1           0       0     0  1.0000000    0.0000000 0.0000000      0
```

we can see that the corrected probability have not changed for the informative pairs, but for the non-informative they have been set to:

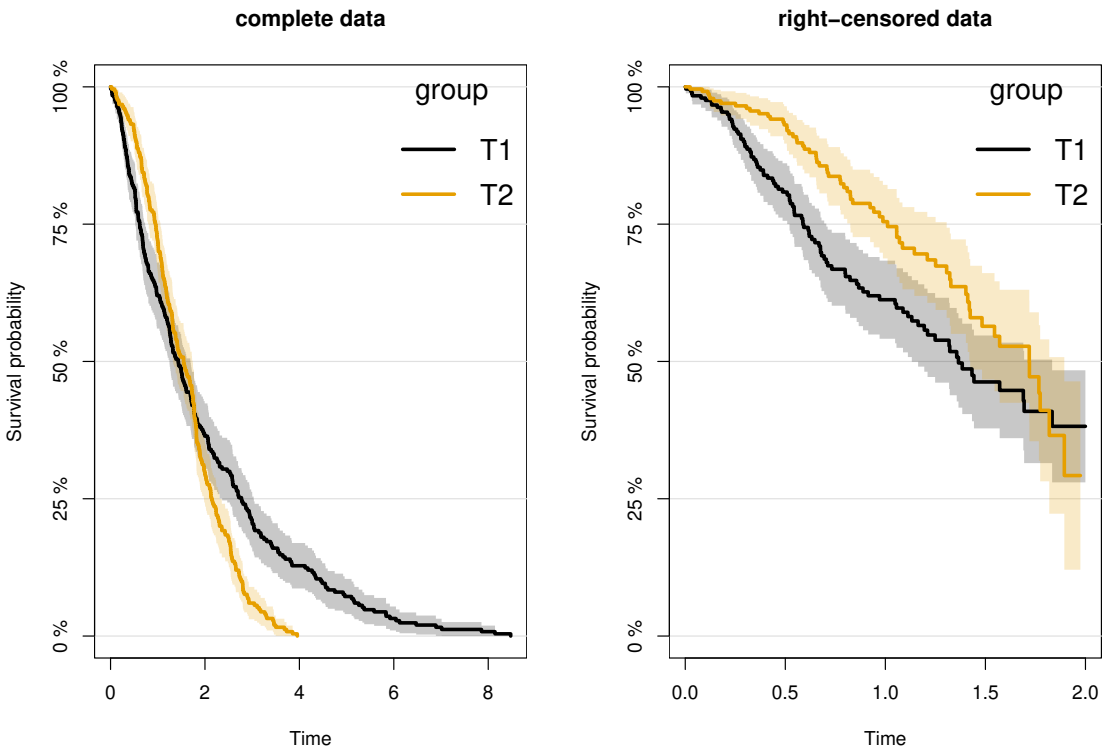
```
iScore[, .(favorable = weighted.mean(favorable, w = 1-uninf),
           unfavorable = weighted.mean(unfavorable, w = 1-uninf),
           neutral = weighted.mean(neutral, w = 1-uninf))]
```

```
   favorable unfavorable   neutral
       <num>       <num>     <num>
1: 0.1182278   0.3643038 0.5174684
```

3.5 Note on the use of the corrections

As mentioned in Péron et al. (2021), the corrections (at the pair level or IPCW) are assumes that uninformative pairs would on average behave like informative pairs. This is typically the case under the proportional hazard assumption. However that may not be the case with other distributions, e.g.:

```
set.seed(10);n <- 250;
df <- rbind(data.frame(group = "T1", time = rweibull(n, shape = 1, scale = 2), status = 1),
            data.frame(group = "T2", time = rweibull(n, shape = 2, scale = 1.8), status = 1))
df$censoring <- runif(NROW(df),0,2)
df$timeC <- pmin(df$time,df$censoring)
df$statusC <- as.numeric(df$time<=df$censoring)
plot(prodlim(Hist(time,status)~group, data = df)); title("complete data");
plot(prodlim(Hist(timeC,statusC)~group, data = df)); title("right-censored data");
```



Here the net treatment benefit that we would have estimated with complete data:

```
BuyseTest.options(method.inference = "none")
e.ref <- BuyseTest(group ~ tte(time,status), data = df, trace = FALSE)
s.ref <- model.tables(e.ref, column = c("favorable","unfavorable","neutral","uninf","Delta"))
s.ref
```

```
  favorable unfavorable neutral uninf    Delta
1   50.2048     49.7952       0     0 0.004096
```

can be taken as a reference. Violation of the assumption will in this example have a substantial impact and lead to a worse estimate with the correction:

```
e.correction <- BuyseTest(group ~ tte(timeC,statusC), data = df, trace = FALSE, correction.
    uninf = TRUE)
s.correction <- model.tables(e.correction, column = c("favorable","unfavorable","neutral","
    uninf","Delta"))
```

```
Warning message:
In .BuyseTest(envir = envirBT, iid = outArgs$iid, method.inference = "none",  :
  Some of the survival curves for endpoint(s) "timeC" are unknown beyond a survival of 0.25.
The correction of uninformative pairs assume that uninformative pairs would on average behave like
This can be a strong assumption and have substantial impact when the tail of the survival curve is
```

than without:

```
e.Peron <- BuyseTest(group ~ tte(timeC,statusC), data = df, trace = FALSE)
s.Peron <- model.tables(e.Peron, column = c("favorable","unfavorable","neutral","uninf","Delta
    "))
rbind("reference" = s.ref,
      "no correction" = s.Peron,
      "correction" = s.correction)
```

```
              favorable unfavorable neutral    uninf      Delta
reference      50.20480    49.79520       0  0.00000 0.00409600
no correction  49.09253    39.74775       0 11.15972 0.09344778
correction     55.25931    44.74069       0  0.00000 0.10518628
```

4 Simulating data using `simBuyseTest`

You can simulate data with the `simBuyseTest` function. For instance the following code simulates data for 5 individuals in the treatment arm and 5 individuals in the control arm:

```
set.seed(10)
simBuyseTest(n.T = 5, n.C = 5)
```

```
       id treatment  eventtime status toxicity       score
    <int>    <fctr>      <num>  <num>   <fctr>       <num>
 1:     1         C 0.60539304      0      yes -1.85374045
 2:     2         C 0.31328027      1      yes -0.07794607
 3:     3         C 0.03946623      0      yes  0.96856634
 4:     4         C 0.32147489      1      yes  0.18492596
 5:     5         C 1.57044952      0      yes -1.37994358
 6:     6         T 0.29069131      0       no  1.10177950
 7:     7         T 0.19522131      0      yes  0.75578151
 8:     8         T 0.04640668      0      yes -0.23823356
 9:     9         T 0.05277335      1      yes  0.98744470
10:    10         T 0.43062009      1      yes  0.74139013
```

By default a categorical, continuous and time to event outcome are generated independently. You can modify their distribution via the arguments `argsBin`, `argsCont`, `argsTTE`. For instance the following code simulates two continuous variables with mean 5 in the treatment arm and 10 in the control arm all with variance 1:

```
set.seed(10)
argsCont <- list(mu.T = c(5,5), mu.C = c(10,10),
                 sigma.T = c(1,1), sigma.C = c(1,1),
                 name = c("tumorSize","score"))
dt <- simBuyseTest(n.T = 5, n.C = 5,
                   argsCont = argsCont)
dt
```

```
       id treatment eventtime status toxicity tumorSize     score
    <int>    <fctr>     <num>  <num>   <fctr>     <num>     <num>
 1:     1         C 0.1805891      0      yes 11.086551  8.564486
 2:     2         C 0.1702538      1      yes  9.237455 10.362087
 3:     3         C 0.2621793      1       no  9.171337  8.240913
 4:     4         C 0.2959301      0       no 10.834474  9.675456
 5:     5         C 0.4816549      1      yes  9.032348  9.348437
 6:     6         T 0.6446131      1       no  5.089347  6.101780
 7:     7         T 0.7372264      1      yes  4.045056  5.755782
 8:     8         T 0.7213402      0      yes  4.804850  4.761766
 9:     9         T 0.1580651      1      yes  5.925521  5.987445
10:    10         T 0.2212117      0      yes  5.482979  5.741390
```

This functionality is based on the `sim` function of the **lava** package.

5 Power calculation using `powerBuyseTest`

The function `powerBuyseTest` can be used to perform power calculation, i.e., estimate the probability of rejecting a null hypothesis under a specific generative mechanism. The user therefore need to specify:

- the generative mechanism via a function - argument `sim`
- the null hypothesis - argument `null`
- the sample size(s) for the which the power should be computed - argument `sample.size`

Consider the following generative mechanism where the outcome follows a Student's t-distribution in the treatment and control group, with same variance and degrees of freedom but different mean:

```
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) + 1/2, group=1))
     return(data.table::as.data.table(out))
}
set.seed(10)
simFCT(101,101)
```

```
              Y group
          <num> <num>
  1:  0.02241932     0
  2: -1.07273566     0
  3:  0.76072274     0
  4: -0.25812356     0
  5:  0.97207866     0
 ---
198:  1.82349375     1
199: -0.98560076     1
200:  1.48143637     1
201:  3.69314316     1
202:  0.96244416     1
```

We then define the null hypothesis:

```
null <- c("netBenefit" = 0)
```

Naming the value is important since that will indicate which statistic should be used (here the net treatment benefit). We can assess the power of a test based on the net treatment benefit using the following syntax:

```
powerW <- powerBuyseTest(sim = simFCT, method.inference = "u-statistic", null = null,
                         sample.size = c(5,10,20,30,50,100),
                         formula = group ~ cont(Y),
                         n.rep = 1000, seed = 10, cpus = 6, trace = 0)
```

And use the summary method to display the power (column `rejection.rate`):

```
summary(powerW)
```

```
        Simulation study with Generalized pairwise comparison
        with 1000 samples

 - net benefit statistic (null hypothesis Delta=0)
 endpoint threshold n.T n.C mean.estimate sd.estimate mean.se rejection.rate
        Y     1e-12   5   5        0.2484       0.359  0.3395          0.069
                     10  10        0.2471      0.2551  0.2464          0.137
                     20  20        0.2444      0.1746  0.1757          0.221
                     30  30         0.243      0.1436  0.1437          0.365
                     50  50        0.2438      0.1114  0.1113          0.557
                    100 100        0.2458      0.0804  0.0787          0.865

 n.T          : number of observations in the treatment group
 n.C          : number of observations in the control group
 mean.estimate: average estimate over simulations
 sd.estimate  : standard deviation of the estimate over simulations
 mean.se      : average estimated standard error of the estimate over simulations
 rejection    : frequency of the rejection of the null hypothesis over simulations
(standard error: H-projection of order 1| p-value: after transformation)
```

It is also possibly to use an asymptotic approximation to derive a approximate sample size satisfying a specific type 1 and type 2 error rate:

```
nW <- powerBuyseTest(sim = simFCT, method.inference = "u-statistic",
                     power = 0.8, max.sample.size = 1000,
                     formula = group ~ cont(Y), null = c("netBenefit" = 0),
                     n.rep = c(1000,10), seed = 10, cpus = 5, trace = 0)
```

This procedure is inspired from the procedure presented by Brunner et al. (2018) in section 3.8.2.2. In short, several 'large' datasets are generated and analyzed using GPC to approximate the statistic of interest (Δ) and its asymptotic variance (σ^2). The sample size needed to achieve the requested power ($1-\beta$) and the requested type 1 error (α) is then deduce, give a dataset, according to the equation $N = \sigma^2 \frac{\left(u_{1-\alpha/2}+u_{1-\beta}\right)^2}{\Delta^2}$ where u_x denotes the x-quantile of the normal distribution. The estimated sample size is then the average calculated sample size across dataset. The argument `max.sample.size` specifies the number of observation per group in the 'large' dataset (here 1000 per group) and the second element of the argument `n.rep` specifies the number of datasets (here 10). The quality of the approximation, as well as the computation time, thus improves when increasing `max.sample.size` and `n.rep[2]`. The achieved power with the estimated sample size can be output as usual using the `summary` method:

```
summary(nW)
```

```
        Sample size calculation with Generalized pairwise comparison
        for a power of 0.8 and type 1 error rate of 0.05
```

```
 - estimated sample size (mean [min;max]): 89 [60;145] controls
                                           89 [60;145] treated

 - net benefit statistic (null hypothesis Delta=0)
 endpoint threshold n.T n.C mean.estimate sd.estimate mean.se rejection.rate
        Y     1e-12  89  89        0.2452      0.0854  0.0834          0.806

 n.T           : number of observations in the treatment group
 n.C           : number of observations in the control group
 mean.estimate: average estimate over simulations
 sd.estimate   : standard deviation of the estimate over simulations
 mean.se       : average estimated standard error of the estimate over simulations
 rejection     : frequency of the rejection of the null hypothesis over simulations
(standard error: H-projection of order 1| p-value: after transformation)
```

6 Modifying default options

The `BuyseTest.options` method enable to get and set the default options of the `BuyseTest` function. For instance, the default option for trace is:

```
BuyseTest.options("trace")
```

```
$trace
[1] 2
```

To change the default option to 0 (i.e. no output) use:

```
BuyseTest.options(trace = 0)
```

To change what the results output by the summary function use:

```
BuyseTest.options(summary.display = list(c("endpoint","threshold","delta","Delta","information
    (%)")))
summary(BT)
```

```
      Generalized pairwise comparisons with 1 endpoint

 - statistic       : net benefit (delta: endpoint specific, Delta: global)
 - null hypothesis : Delta == 0
 - treatment groups: T (treatment) vs. C (control)
 - censored pairs  : deterministic score or uninformative
 - uninformative pairs: score equals the averaged score of all informative pairs
 - results
  endpoint threshold   Delta information(%)
 eventtime       0.5 -0.2461            100
```

To restore the original default options do:

```
BuyseTest.options(reinitialise = TRUE)
```

References

Ajufo, E., Nayak, A., and Mehra, M. R. (2023). Fallacies of using the win ratio in cardiovascular trials: challenges and solutions. *Basic to Translational Science*, 8(6):720–727.

Brunner, E., Bathke, A. C., and Konietschke, F. (2018). *Rank and pseudo-rank procedures for independent observations in factorial designs*. Springer.

Buyse, M. (2010). Generalized pairwise comparisons of prioritized outcomes in the two-sample problem. *Statistics in medicine*, 29(30):3245–3257.

Buyse, M., Verbeeck, J., Saad, E. S., De Backer, M., Deltuvaite-Thomas, V., and Molenberghs, G. (2025). *Handbook of Generalized Pairwise Comparisons Methods for Patient-Centric Analysis*. Chapman & Hall.

Dong, G., Qiu, J., Wang, D., and Vandemeulebroecke, M. (2018). The stratified win ratio. *Journal of biopharmaceutical statistics*, 28(4):778–796.

Ozenne, B., Budtz-Jørgensen, E., and Péron, J. (2021). The asymptotic distribution of the net benefit estimator in presence of right-censoring. *Statistical methods in medical research*, 30(11):2399–2412.

Péron, J., Buyse, M., Ozenne, B., Roche, L., and Roy, P. (2018). An extension of generalized pairwise comparisons for prioritized outcomes in the presence of censoring. *Statistical methods in medical research*, 27(4):1230–1239.

Péron, J., Idlhaj, M., Maucort-Boulch, D., Giai, J., Roy, P., Collette, L., Buyse, M., and Ozenne, B. (2021). Correcting the bias of the net benefit estimator due to right-censored observations. *Biometrical Journal*, 63(4):893–906.

Piffoux, M., Ozenne, B., De Backer, M., Buyse, M., Chiem, J.-C., and Péron, J. (2024). Restricted net treatment benefit in oncology. *Journal of Clinical Epidemiology*, 170:111340.

Verbeeck, J., Spitzer, E., de Vries, T., van Es, G., Anderson, W., Van Mieghem, N., Leon, M., Molenberghs, G., and Tijssen, J. (2019). Generalized pairwise comparison methods to analyze (non) prioritized composite endpoints. *Statistics in medicine*.