Overview of the package BuyseTest

Brice Ozenne

May 27, 2020

This vignette describes the main functionalities of the **BuyseTest** package. This 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. 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 $\mathbb{P}[Y \geq X + \tau]$ and the proportion in favor of control $\mathbb{P}[X \geq Y + \tau]$. Other statistics such as the net benefit $\mathbb{P}[Y \geq X + \tau] - \mathbb{P}[X \geq Y + \tau]$ or the win ratio $\frac{\mathbb{P}[Y \geq X+\tau]}{\mathbb{P}[X \geq Y+\tau]}$ can then be deduced. It is assumed that the reader is familar with the GPC terminology [1], e.g. prioritized endpoints, pair, net benefit, win ratio, threshold of clinical relevance, ..., since this vignette focuses on the software aspect of the **BuyseTest** package (not on the underlying statistical model).

The **BuyseTest** package contains three main functions:

- the function `BuyseTest` is the main function of the package. It performs the GPC, estimates the net benefit/win ratio, and output a *BuyseRes* object. The user can interact with *BuyseRes* objects using:
  - `summary` to obtain a nice display of the results
  - `coef` to extract the estimates.
  - `confint` to extract estimates, confidence intervals, and p.values.
  - `getIid` to extract the iid decomposition of the estimator.
  - `getPairScore` to extract the contribution of each pair to the net benefit/win ratio.
  - `getSurvival` to extract the estimates of the survival (only relevant for right-censored endpoints).
- 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 `BuyseTest.options` function enables the user to display the default values used in the **BuyseTest** package (essentially used by the `BuyseTest` function). function. The function can also change the default values to better match the user needs.

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

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

```
  trt celltype time status karno diagtime age prior
1   1 squamous   72      1    60        7  69     0
2   1 squamous  411      1    70        5  64    10
3   1 squamous  228      1    60        3  38     0
4   1 squamous  126      1    60        9  63    10
5   1 squamous  118      1    70       11  65    10
6   1 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] '2.1.5'
```

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

```
sessionInfo()
```

```
R version 3.5.1 (2018-07-02)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 7 x64 (build 7601) Service Pack 1

Matrix products: default

locale:
[1] LC_COLLATE=Danish_Denmark.1252  LC_CTYPE=Danish_Denmark.1252
[3] LC_MONETARY=Danish_Denmark.1252 LC_NUMERIC=C
[5] LC_TIME=Danish_Denmark.1252

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

other attached packages:
[1] BuyseTest_2.1.5   spelling_1.2      butils.base_1.2   Rcpp_1.0.4.6      data.table_1.12.8
[6] usethis_1.4.0     devtools_2.0.1    roxygen2_7.1.0
```

```
loaded via a namespace (and not attached):
 [1] compiler_3.5.1            iterators_1.0.12          prettyunits_1.0.2
 [4] base64enc_0.1-3           remotes_2.0.2             testthat_2.0.0
 [7] digest_0.6.17             pkgbuild_1.0.2            pkgload_1.0.2
[10] lattice_0.20-35           memoise_1.1.0             rlang_0.3.1
[13] Matrix_1.2-14             foreach_1.5.0             cli_1.0.1
[16] parallel_3.5.1            prodlim_2019.11.13        RcppArmadillo_0.9.850.1.0
[19] withr_2.1.2               stringr_1.3.1             xml2_1.2.0
[22] knitr_1.20                desc_1.2.0                fs_1.2.6
[25] stats4_3.5.1              grid_3.5.1                rprojroot_1.3-2
[28] glue_1.3.0                R6_2.3.0                  processx_3.2.0
[31] survival_2.44-1.1         sessioninfo_1.1.1         lava_1.6.7
[34] callr_3.0.0               purrr_0.3.4               magrittr_1.5
[37] splines_3.5.1             codetools_0.2-15          backports_1.1.5
[40] ps_1.1.0                  assertthat_0.2.1          stringi_1.2.4
[43] doParallel_1.0.15         crayon_1.3.4
```

1 Performing generalized pairwise comparisons (GPC) using the `BuyseTest` function

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 to specify 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[2]:

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

```
        Generalized Pairwise Comparisons

Settings
   - 2 groups  : Control = 1 and Treatment = 2
   - 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
```

[2] the argument `method.inference` is set to "none" to diseable the computation of p-values and confidence intervals. This makes the execution of `BuyseTest` much faster.

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

```
BT.f <- BuyseTest(trt ~ tte(time, threshold = 20, status = "status"),
                  data = veteran, trace = 0)
```

Here we set in addition the argument `trace` to 0 to force the function to be silent (i.e. no display in the terminal). We can check that the two approaches are equivalent:

```
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 benefit (delta: endpoint specific, Delta: global)
> null hypothesis : Delta == 0
> confidence level: 0.95
> inference       : H-projection of order 1
> treatment groups: 1 (control) vs. 2 (treatment)
> right-censored pairs: probabilistic score based on the survival curves
> results
endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   delta   Delta
    time        20      100        37.78          46.54      15.68        0 -0.0877 -0.0877
CI [2.5% ; 97.5%] p.value
 [-0.2735;0.1045] 0.37162
```

To display the number of pairs instead of the percentage of pairs that are favorable/unfavorable/neutral/uniformative, set the argument `percentage` to `FALSE`:

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

```
      Generalized pairwise comparisons with 1 endpoint

> statistic       : net benefit (delta: endpoint specific, Delta: global)
> null hypothesis : Delta == 0
> confidence level: 0.95
> inference       : H-projection of order 1
> treatment groups: 1 (control) vs. 2 (treatment)
> right-censored pairs: probabilistic score based on the survival curves
> results
endpoint threshold total favorable unfavorable neutral uninf   delta   Delta
    time        20  4692   1772.59     2183.89  735.52     0 -0.0877 -0.0877
CI [2.5% ; 97.5%] p.value
 [-0.2735;0.1045] 0.37162
```

By default `summary` displays results relative to the net benefit. To get results for the win ratio set the argument `statistic` to "winRatio":

```
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
> treatment groups: 1 (control) vs. 2 (treatment)
> right-censored pairs: probabilistic score based on the survival curves
> results
endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)  delta  Delta
    time        20      100        37.78          46.54      15.68        0 0.8117 0.8117
CI [2.5% ; 97.5%] p.value
  [0.5134;1.2833] 0.37195
```

See `help(BuyseRes-summary)` for more detailed explanations about the `summary` method and its output.

1.2 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 benefit (delta: endpoint specific, Delta: global)
> null hypothesis : Delta == 0
> confidence level: 0.95
> inference       : H-projection of order 1
> treatment groups: 1 (control) vs. 2 (treatment)
> right-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     1e-12    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`). It is also possible to perform the comparisons on all endpoints 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 benefit (delta: endpoint specific, Delta: global)
> null hypothesis : Delta == 0
> confidence level: 0.95
> inference       : H-projection of order 1
> treatment groups: 1 (control) vs. 2 (treatment)
> right-censored pairs: probabilistic score based on the survival curves
> neutral pairs   : re-analyzed using lower priority endpoints
> results
endpoint threshold weight total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   delta
    time        20      1      100        37.78          46.54      15.68        0 -0.0877
   karno     1e-12      1      100        41.82          44.95      13.24        0 -0.0313
  Delta CI [2.5% ; 97.5%] p.value
-0.0877  [-0.2735;0.1045] 0.37162
-0.1190  [-0.4346;0.2226] 0.49821
```

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 1 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)
summary(BT.nHw)
```

```
      Generalized pairwise comparisons with 2 endpoints

> statistic       : net benefit (delta: endpoint specific, Delta: global)
> null hypothesis : Delta == 0
> confidence level: 0.95
> inference       : H-projection of order 1
> treatment groups: 1 (control) vs. 2 (treatment)
> right-censored pairs: probabilistic score based on the survival curves
> neutral pairs   : re-analyzed using lower priority endpoints
> results
endpoint threshold weight total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   delta
    time        20    0.8      100        37.78          46.54      15.68        0 -0.0877
   karno     1e-12    0.2      100        41.82          44.95      13.24        0 -0.0313
  Delta CI [2.5% ; 97.5%] p.value
-0.0701  [-0.2204;0.0834] 0.37073
-0.0764  [-0.2504;0.1024] 0.40269
```

This has been refered as the O'Brien test in the litterature (Verbeeck et al. (2019), section 3.2).

1.3 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,
                   method.inference = "none", trace = 0)
BTinv
```

```
endpoint threshold  delta  Delta
    time        20 0.0844 0.0844
```

Internally `BuyseTest` will multiply by -1 the values of the endpoint to ensure that lower values are considered as favorable. A direct consequence is that `BuyseTest` will not accept an endpoint with different operators:

```
ffop2 <- update(ffop, . ~ . + tte(time, "status", 10, ">0"))
try(BuyseTest(ffop2, data = veteran,
              method.inference = "none", trace = 0))
```

```
Error in (function (name.call, status, correction.uninf, cpus, data, endpoint,  :
  Cannot have different operator for the same endpoint used at different priorities.
```

1.4 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 a single variable in the right hand side of the formula:

```
ff2strata <- update(ff2, . ~ . + celltype)
BT2 <- BuyseTest(ff2strata, data = veteran,
                 trace = 0, method.inference = "none")
```

The fact the it is not wrapped by `bin`, `cont` or `tte` indicates differentiate it from endpoint variables.

When doing a stratified analysis, the summary method displays the global results as well as the results within each strata[3]:

```
summary(BT2, type.display = c("endpoint","threshold","strata",
                              "total","favorable","unfavorable","delta","Delta"))
```

```
     Generalized pairwise comparisons with 2 prioritized endpoints and 4 strata

> statistic       : net benefit (delta: endpoint specific, Delta: global)
> null hypothesis : Delta == 0
> treatment groups: 1 (control) vs. 2 (treatment)
> right-censored pairs: probabilistic score based on the survival curves
> neutral pairs   : re-analyzed using lower priority endpoints
> uninformative pairs: no contribution at the current endpoint, analyzed at later endpoints
> results
endpoint threshold    strata total(%) favorable(%) unfavorable(%)   delta   Delta
    time        20    global   100.00        36.06          45.77 -0.0971 -0.0971
                     squamous    25.38        14.33           8.77  0.2193
                    smallcell    45.69        12.69          20.88 -0.1792
                        adeno    13.71         4.74           6.15 -0.1034
                        large    15.23         4.30           9.97 -0.3722
   karno     1e-12    global    18.17         6.72           8.07 -0.0135 -0.1106
                     squamous     2.28         0.76           0.94 -0.0071
                    smallcell    12.12         4.33           5.75 -0.0311
                        adeno     2.81         1.46           0.85  0.0448
                        large     0.96         0.17           0.54 -0.0241
```

Note that here the numbers in the total/favorable/unfavorable/ columns are relative to the overall sample while the delta is only relative to the strata. The global delta is a sum of the strata specific delta weighted by the empirical proportion of pairs for each strata.

1.5 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
1:  1       Yes   Yes   15
2:  2        No   Yes   20
```

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

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 benefit (delta: endpoint specific, Delta: global)
> null hypothesis : Delta == 0
> treatment groups: No (control) vs. Yes (treatment)
> neutral pairs   : re-analyzed using lower priority endpoints
> results
endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%) delta Delta
   tumor       0.5      100            0              0        100        0     0     0
    size     1e-12      100          100              0          0        0     1     1
```

the outcome of the comparison is neutral for the first priority, but favorable for the second priority. If we set the argument `neutral.as.uninf` to `FALSE`, `BuyseTest` 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 benefit (delta: endpoint specific, Delta: global)
> null hypothesis : Delta == 0
> treatment groups: No (control) vs. Yes (treatment)
> neutral pairs   : ignored at lower priority endpoints
> results
endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%) delta Delta
   tumor       0.5      100            0              0        100        0     0     0
    size     1e-12        0            0              0          0        0     0     0
```

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

1.6 What about p-value and confidence intervals?

Several methods are available in `BuyseTest` to perform statistical inference:

- **permutation test** setting the argument `method.inference` to `"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 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: 1 (control) vs. 2 (treatment)
> right-censored pairs: probabilistic score based on the survival curves
> results
endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   delta   Delta
    time        20      100        37.78          46.54      15.68        0 -0.0877 -0.0877
p.value
  0.355
```

- **bootstrap resampling** setting the argument `method.inference` to `"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 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: 1 (control) vs. 2 (treatment)
> right-censored pairs: probabilistic score based on the survival curves
> results
```

```
endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   delta   Delta
    time        20      100        37.78          46.54      15.68        0 -0.0877 -0.0877
CI [2.5% ; 97.5%] p.value
 [-0.2922;0.1013]   0.393
```

- **asymptotic distribution** setting the argument `method.inference` to `"u-statistic"`. In large enough samples, this approach gives valid p-values and confidence intervals.

```
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 benefit (delta: endpoint specific, Delta: global)
> null hypothesis : Delta == 0
> confidence level: 0.95
> inference       : H-projection of order 1
> treatment groups: 1 (control) vs. 2 (treatment)
> right-censored pairs: probabilistic score based on the survival curves
> results
endpoint threshold total(%) favorable(%) unfavorable(%) neutral(%) uninf(%)   delta   Delta
    time        20      100        37.78          46.54      15.68        0 -0.0877 -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 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"`).

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 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)
```

```
      index.1 index.2 favorable unfavorable neutral uninf weight
   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),]
```

```
   trt celltype time status karno diagtime age prior
70   2 squamous  999      1    90       12  54    10
1    1 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 benefit in favor of the treatment for the first endpoint:

```
BT.keep
```

```
endpoint threshold   delta   Delta
    time        20 -0.0877 -0.0877
   karno     1e-12 -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 [1:4] 0 0 NA NA
```

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]
```

```
   index.1 index.2 indexWithinStrata.1 indexWithinStrata.2 favorable unfavorable   neutral
1:      22      71                  22                  2         0   0.6950827 0.3049173
   uninf weight
1:     0      1
```

In the dataset this corresponds to:

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

```
   trt  celltype time status karno diagtime age prior
22   1 smallcell   97      0    60        5  67     0
71   2  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]
```

```
   index.1 index.2 indexWithinStrata.1 indexWithinStrata.2 favorable unfavorable   neutral
1:      10      72                  10                  3 0.5058685   0.3770426 0.1170889
   uninf weight
1:     0      1
```

which corresponds to the observations:

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

```
   trt celltype time status karno diagtime age prior
10   1 squamous  100      0    70        6  70     0
72   2 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(...){ ## here we don't need to return 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
```

3 Dealing with missing values or/and right censoring

In presence of censoring or missing values, some pairs may be classified as uninformative. This may bias the estimate of the net net benefit. Two corrections are currently proposed to correct this bias.

To illustrate the effect of these correction, we will use the following dataset:

```
set.seed(10)
dt <- simBuyseTest(5e2, latent = TRUE, argsCont = NULL,
                   argsTTE = list(scale.T = 1/2, scale.C = 1,
                                  scale.Censoring.C = 1, scale.Censoring.T = 1))
dt[, status1 := 1]
head(dt)
```

```
   treatment eventtimeUncensored eventtimeCensoring eventtime status toxicity eta_toxicity
1:         C           1.3499793          0.4546612 0.4546612      0      yes  -0.30786498
2:         C           1.3022440          0.8234702 0.8234702      0       no   0.75808558
3:         C           0.9800451          0.3656312 0.3656312      0      yes  -0.57386341
4:         C           0.1809881          0.6066301 0.1809881      1      yes  -0.93874446
5:         C           0.2747980          0.5944344 0.2747980      1      yes  -0.02769932
6:         C           0.1351895          0.7215782 0.1351895      1      yes  -1.06624865
   status1
1:       1
2:       1
3:       1
4:       1
5:       1
6:       1
```

where we have the uncensored event times as well as the censored event times. The percentage of censored observations is:

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

```
[1] 46
```

We would like to be able to recover the net 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   Delta
eventtimeUncensored       0.5 -0.2314 -0.2314
```

using the censored survival times:

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

```
 endpoint threshold   delta   Delta
eventtime       0.5 -0.0881 -0.0881
```

As we can see on this example, the net benefit is shrunk toward 0.

3.0.1 Inverse probability-of-censoring weights (IPCW)

With IPCW 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 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 benefit (delta: endpoint specific, Delta: global)
> null hypothesis : Delta == 0
> treatment groups: C (control) vs. T (treatment)
> right-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   Delta
eventtime       0.5      100        11.31          34.64      54.05        0 -0.2333 -0.2333
```

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
1:         0           0       0     1   0.000000     0.000000 0.000000      0
2:         0           1       0     0   0.000000     2.647043 0.000000      0
3:         0           0       1     0   0.000000     0.000000 2.647043      0
4:         1           0       0     0   2.647043     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] 2.647043
```

i.e. the inverse probability of being informative.

3.0.2 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 benefit (delta: endpoint specific, Delta: global)
> null hypothesis : Delta == 0
> treatment groups: C (control) vs. T (treatment)
> right-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   Delta
eventtime       0.5      100        11.31          34.64      54.05        0 -0.2333 -0.2333
```

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
1:         0           0       0     1  0.1131029    0.3464344 0.5404627      0
2:         0           1       0     0  0.0000000    1.0000000 0.0000000      0
3:         0           0       1     0  0.0000000    0.0000000 1.0000000      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
1: 0.1131029   0.3464344 0.5404627
```

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)
```

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

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

This functionality is based on the `sim` function of the **lava** package (https://github.com/kkholst/lava)

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))
}
simFCT(101,101)
```

```
            Y group
  1:  0.9243314     0
  2: -1.9571691     0
  3:  4.7270966     0
  4:  1.5312298     0
  5:  0.1236634     0
 ---
198:  0.6567732     1
199: -0.5096230     1
200:  0.5129740     1
201:  0.9709603     1
202:  0.1356299     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 benefit). We can assess the power of the Wilcoxon's test using the following syntax:

```
powerW <- powerBuyseTest(sim = simFCT,
                         sample.size = c(5,10,20,30,50,100),
                         null = null,
                         formula = group ~ cont(Y),
                         n.rep = 1000, seed = 10,
                         cpus = 3, 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

 > statistic   : net benefit (null hypothesis Delta=0)
 endpoint threshold n.T n.C mean.estimate sd.estimate mean.se rejection.rate
        Y     1e-12   5   5        0.2517      0.3672   0.336          0.078
                     10  10        0.2535      0.2538  0.2459           0.14
                     20  20        0.2492      0.1783  0.1754          0.249
                     30  30        0.2463      0.1439  0.1436          0.357
                     50  50        0.2428       0.113  0.1113           0.56
                    100 100        0.2448      0.0799  0.0787          0.862

 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: C (control) vs. T (treatment)
> right-censored pairs: deterministic score or uninformative
> uninformative pairs: score equals the averaged score of all informative pairs
> results
 endpoint threshold   delta   Delta information(%)
eventtime         1 -0.0202 -0.0202            100
```

To restore the original default options do:

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

References

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

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.

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*.