Overview of the package BuyseTest

Brice Ozenne

September 10, 2019

The **BuyseTest** package contains five functions that are relevant for the user:

- the `BuyseTest` function to compute the net benefit/win ratio. It is the main function of the package.
- a `summary` function to display the results computed by the `BuyseTest` function.
- a `confint` function to extract estimates, confidence intervals, and p.values.
- the `getPairScore` method to extract the contribution of each pair to the net benefit/win ratio.
- the `BuyseTest.options` function the contain the default values for the argument of the `BuyseTest` function. These default values can be changed to better match the user needs.

Two additional function are presented in this document: `simBuyseTest` that can be used to quickly simulate data and `getSurvival` that is useful to reproduce the results output by `BuyseTest`.

Before going further we need to load the **BuyseTest** package in the R session:

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

The **BuyseTest** package is under active development. Newer package versions may include additional functionalities and fix previous bugs. To get the current version of the package use `utils::packageVersion`:

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

```
[1] '1.7.8'
```

1 Example data

For this overview 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.

2 Performing generalized pairwise comparisons (GPC) using the BuyseTest function

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

- an 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 associated to each endpoint (default= 10^{-12}) - argument `threshold`
- the censoring associated to each endpoint (for time to event endpoint) - argument `censoring`
- how to compute the distribution of the statistic of interest - argument `method.inference`

There are two equivalent ways to define the GPC:

- using a separate argument for each element[1]:

```
BT <- BuyseTest(data = veteran,
                endpoint = "time",
                type = "timeToEvent",
                treatment = "trt",
                censoring = "status",
                threshold = 20,
                method.inference = "none")
```

```
         Generalized Pairwise Comparisons

Settings
   - treatment groups: Control = 1 and Treatment = 2
   - 1 endpoint:
       priority endpoint type          operator              threshold censoring
       1        time     time to event higher is favorable  20        status
   - management of neutral pairs: re-analyzed using endpoints of lower priority (if any)
   - management of censored survival pairs: use Kaplan Meier survival curves to compute the score

Point estimation

Gather the results in a BuyseRes object
```

[1] we set the argument `method.inference` 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, censoring = "status"),
                  data = veteran, trace = 0, method.inference = "none")
```

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)
```

2.1 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
> treatment groups: 1 (control) vs. 2 (treatment)
> censored pairs  : use Kaplan Meier survival curves to compute the score
> results
endpoint threshold total favorable unfavorable neutral uninf   delta   Delta
    time        20   100     37.78       46.54   15.68     0 -0.0877 -0.0877
```

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
> treatment groups: 1 (control) vs. 2 (treatment)
> censored pairs  : use Kaplan Meier survival curves to compute the score
> results
endpoint threshold total favorable unfavorable neutral uninf   delta   Delta
    time        20  4692   1772.59     2183.89  735.52     0 -0.0877 -0.0877
```

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

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

```
     Generalized pairwise comparisons with 1 endpoint

> statistic       : win ratio (delta: endpoint specific, Delta: global)
> null hypothesis : Delta == 1
> treatment groups: 1 (control) vs. 2 (treatment)
> censored pairs  : use Kaplan Meier survival curves to compute the score
> results
endpoint threshold total favorable unfavorable neutral uninf  delta  Delta
    time        20   100     37.78       46.54   15.68     0 0.8117 0.8117
```

Since we have set the argument `n.permutation` to 0 (i.e. no permutation test) in the stratified analysis, we do not get confidence intervals or p.values when calling the `summary` method. See `help(BuyseRes-summary)` for more detailed explanations about the `summary` method.

2.2 Using multiple endpoints

More than one endpoint can be considered by indicating a vector of endpoints, types, and thresholds. In the formula interface, just add another endpoint at then end of the formula:

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

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`:

```r
BT.multi2 <- BuyseTest(trt ~ tte(time, threshold = 20, censoring = "status",) + cont(
   karno, threshold = 0),
                      hierarchical = FALSE,
                      data = veteran, method.inference = "none", trace = 0)
```

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.:

```r
BT.multi3 <- BuyseTest(trt ~ tte(time, threshold = 20, censoring = "status", weight =
    0.8) + cont(karno, threshold = 0, weight = 0.2),
                      hierarchical = FALSE,
                      data = veteran, method.inference = "none", trace = 0)
```

This has been refered as the O'Brien test in the litterature.

2.3 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:

```
BTinv <- BuyseTest(trt ~ tte(time, threshold = 20, censoring = "status", operator = "
    <0"),
                   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:

```
try(BuyseTest(trt ~ tte(time, threshold = 20, censoring = "status", operator = "<0")
    + tte(time, 10, "status", ">0"),
              data = veteran, method.inference = "none", trace = 0))
```

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

2.4 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:

```
BT2 <- BuyseTest(trt ~ tte(time, threshold = 20, censoring = "status") + cont(karno,
    threshold = 0) + celltype,
                 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:

```
summary(BT2)
```

```
      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)
> censored pairs  : use Kaplan Meier survival curves to compute the score
> uninformative pairs: no contribution at the current endpoint, analyzed at later endpoints (if a
> results
endpoint threshold    strata  total favorable unfavorable neutral uninf    delta   Delta
    time        20    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
 karno     1e-12    global  18.17       6.72         8.07     3.38  0.00 -0.0135 -0.1106
                  squamous   2.28       0.76         0.94     0.59  0.00 -0.0071
                 smallcell  12.12       4.33         5.75     2.03  0.00 -0.0311
                     adeno   2.81       1.46         0.85     0.51  0.00  0.0448
                     large   0.96       0.17         0.54     0.25  0.00 -0.0241
```

Note that here the numbers in the favorable/unfavorable/neutral/uniformative 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.

2.5 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
```

If we perform we 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: 0 (control) vs. 1 (treatment)
> 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: 0 (control) vs. 1 (treatment)
> 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.

2.6 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"`. This approach gives valid p-values.

```
BT.perm <- BuyseTest(trt ~ tte(time, threshold = 20, censoring = "status"),
                      data = veteran, trace = 0, method.inference = "permutation",
                      scoring.rule = "Gehan")
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
                    confidence intervals/p-values computed using the quantiles of the empirical c
> treatment groups: 1 (control) vs. 2 (treatment)
> censored pairs  : uninformative pairs
> uninformative pairs: no contribution at the current endpoint, analyzed at later endpoints (if a
> results
endpoint threshold total favorable unfavorable neutral uninf   delta   Delta p.value
    time        20   100     34.93        44.1      15  5.97 -0.0916 -0.0916   0.322
```

- bootstrap resampling setting the argument `method.inference` to `"bootstrap"`. In large enough samples, this approach gives valid p-values and confidence intervals.

```
BT.perm <- BuyseTest(trt ~ tte(time, threshold = 20, censoring = "status"),
                     data = veteran, trace = 0, method.inference = "bootstrap",
                     scoring.rule = "Gehan")
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       : bootstrap resampling with 1000 samples
                    confidence intervals/p-values computed using the quantiles of the empirical c
> treatment groups: 1 (control) vs. 2 (treatment)
> censored pairs  : uninformative pairs
> uninformative pairs: no contribution at the current endpoint, analyzed at later endpoints (if a
> results
endpoint threshold total favorable unfavorable neutral uninf   delta   Delta
    time        20   100     34.93        44.1      15  5.97 -0.0916 -0.0916
CI [2.5 ; 97.5] p.value
[-0.283;0.0888]   0.349
```

- normal approximation setting the argument `method.inference` to `"u-statistic"`. In large enough samples and for the Gehan scoring rule, this approach gives valid p-values and confidence intervals.

```
BT.ustat <- BuyseTest(trt ~ tte(time, threshold = 20, censoring = "status"),
                      data = veteran, trace = 0, method.inference = "u-statistic",
                      scoring.rule = "Gehan")
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)
> censored pairs  : uninformative pairs
> uninformative pairs: no contribution at the current endpoint, analyzed at later endpoints (if a
> results
endpoint threshold total favorable unfavorable neutral uninf   delta   Delta
    time        20   100     34.93        44.1      15  5.97 -0.0916 -0.0916
 CI [2.5 ; 97.5] p.value
[-0.2708;0.0936]  0.3323
```

The last method is the fastest since it requires very little extra computation compared to the classical GPC. 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 typically be 10000.

3 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.

3.1 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, censoring = "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 favorableC unfavorableC
   1:       1      70         1           0       0     0      1          1            0
   2:       2      70         1           0       0     0      1          1            0
   3:       3      70         1           0       0     0      1          1            0
   4:       4      70         1           0       0     0      1          1            0
   5:       5      70         1           0       0     0      1          1            0
  ---
4688:      65     137         0           1       0     0      1          0            1
4689:      66     137         0           1       0     0      1          0            1
4690:      67     137         0           1       0     0      1          0            1
4691:      68     137         0           1       0     0      1          0            1
4692:      69     137         0           1       0     0      1          0            1
      neutralC uninfC
   1:        0      0
   2:        0      0
   3:        0      0
   4:        0      0
   5:        0      0
  ---
4688:        0      0
4689:        0      0
4690:        0      0
4691:        0      0
4692:        0      0
```

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`.

3.2 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` in BuyseTest.options enables to export the survival probabilities in the output of BuyseTest:

```
BuyseTest.options(keep.survival = TRUE)
BT.keep2 <- BuyseTest(trt ~ tte(time, threshold = 20, censoring = "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, 1:7] 72 411 228 126 118 10 82 110 314 100 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:7] "time" "SurvivalC-threshold" "SurvivalC_0" "SurvivalC+threshold" ...
 $ survTimeT: num [1:68, 1:7] 999 112 87 231 242 991 111 1 587 389 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:7] "time" "SurvivalC-threshold" "SurvivalC_0" "SurvivalC+threshold" ...
 $ survJumpC: num [1:57, 1:3] 3 4 7 8 10 11 12 13 16 18 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:3] "time" "survival" "dSurvival"
 $ survJumpT: num [1:51, 1:3] 1 2 7 8 13 15 18 19 20 21 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:3] "time" "survival" "dSurvival"
 $ lastSurv : num [1:4] 0 0 NA NA
```

3.2.1 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
1:      22      71                  22                   2         0   0.6950827
     neutral uninf weight favorableC unfavorableC  neutralC uninfC
1: 0.3049173     0      1          0    0.6950827 0.3049173      0
```

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 SurvivalC+threshold 
           97.0000000           0.5615232            0.5171924           0.4235463 
SurvivalT-threshold         SurvivalT_0 SurvivalT+threshold 
            0.4558824           0.3643277            0.2827500 
```

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 SurvivalC+threshold 
          112.0000000           0.5319693            0.4549201           0.3594915 
SurvivalT-threshold         SurvivalT_0 SurvivalT+threshold 
            0.3801681           0.2827500            0.2827500 
```

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 
```

3.2.2 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 `calcIntegralScore\_cpp` that takes as argument a matrix containing the survival and the jumps in survival, e.g.:

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

```
     time  survival   dSurvival
[1,]    1 0.7681159 -0.02941176
[2,]    2 0.7536232 -0.01470588
[3,]    7 0.7388463 -0.02941176
[4,]    8 0.7388463 -0.02941176
[5,]   13 0.7092924 -0.01470588
[6,]   15 0.6945155 -0.02941176
```

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
1:      10      72                  10                   3 0.5058685   0.3770426
     neutral uninf weight favorableC unfavorableC  neutralC uninfC
1: 0.1170889     0      1  0.5058685    0.3770426 0.1170889      0
```

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:

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

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
    calcIntegralScore_cpp(...,
                          returnDeriv = FALSE, column = 0,
                          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
```

4 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(rates.T = 2, rates.C = 1,
                                  rates.Censoring.C = 3, rates.Censoring.T = 3))
dt[, status1 := 1]
head(dt)
```

```
   Treatment toxicity eventtimeUncensored eventtimeCensoring eventtime status status1
1:         C        0           0.1588268          2.6268101 0.1588268      1       1
2:         C        1           1.7204676          0.2000192 0.2000192      0       1
3:         C        1           0.4900490          0.5747995 0.4900490      1       1
4:         C        0           0.1138545          1.5188001 0.1138545      1       1
5:         C        1           0.5191035          3.8340048 0.5191035      1       1
6:         C        0           0.9405830          1.9078657 0.9405830      1       1
```

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

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

```
[1] 0.317
```

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

```
BuyseTest(Treatment ~ tte(eventtimeUncensored, status1, threshold = 1),
          data = dt,
          scoring.rule = "Gehan", method.inference = "none", trace = 0)
```

```
           endpoint threshold  delta  Delta
eventtimeUncensored         1 0.2401 0.2401
```

using the censored survival times:

```
BuyseTest(Treatment ~ tte(eventtime, status, threshold = 1),
          data = dt,
          scoring.rule = "Gehan", method.inference = "none", trace = 0)
```

```
 endpoint threshold  delta  Delta
eventtime         1 0.1363 0.1363
```

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

4.0.1 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 = 1),
                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)
> censored pairs  : uninformative pairs
> uninformative pairs: no contribution, their weight is passed to the informative pairs using IPC
> results
 endpoint threshold total favorable unfavorable neutral uninf  delta  Delta
eventtime         1   100     37.11       12.34   50.54     0 0.2477 0.2477
```

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       1     0    0.00000      0.00000  1.81657      0
2:         0           0       0     1    0.00000      0.00000  0.00000      0
3:         0           1       0     0    0.00000      1.81657  0.00000      0
4:         1           0       0     0    1.81657      0.00000  0.00000      0
```

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

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

```
[1] 1.81657
```

i.e. the inverse probability of being informative.

4.0.2 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 = 1),
                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)
> censored pairs  : uninformative pairs
> uninformative pairs: score equals the averaged score of all informative pairs
> results
 endpoint threshold total favorable unfavorable neutral uninf  delta  Delta
eventtime         1   100     37.11       12.34   50.54     0 0.2477 0.2477
```

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       1     0   0.000000    0.0000000 1.0000000      0
2:         0           0       0     1   0.371118    0.1234396 0.5054424      0
3:         0           1       0     0   0.000000    1.0000000 0.0000000      0
4:         1           0       0     0   1.000000    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:  0.371118   0.1234396 0.5054424
```

5 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 toxicity       score eventtime status
 1:         C        1  0.54361539 1.8252132      0
 2:         C        1 -0.70762484 2.9489056      1
 3:         C        1 -0.36944577 0.7213402      0
 4:         C        1 -1.32197565 0.6322603      1
 5:         C        1  1.28059746 0.2212117      0
 6:         T        1  0.01874617 0.1453481      0
 7:         T        1 -0.18425254 0.4855601      0
 8:         T        0 -1.37133055 0.2547505      0
 9:         T        1 -0.59916772 1.0340368      0
10:         T        0  0.29454513 0.3579324      1
```

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 toxicity tumorSize     score eventtime status
 1:         C        1  9.010394 10.667415 0.2729620      0
 2:         C        0  9.965152 11.691755 0.5562477      0
 3:         C        0 10.847160 10.001261 0.8040608      0
 4:         C        0 11.525498  9.257539 1.8477048      1
 5:         C        1  9.932625 10.609684 0.3639572      1
 6:         T        1  5.389794  5.018746 0.6243732      0
 7:         T        1  3.791924  4.815747 0.3527879      1
 8:         T        1  4.636324  3.628669 1.7731161      0
 9:         T        0  3.373327  4.400832 0.1055467      0
10:         T        0  4.743522  5.294545 0.8612402      0
```

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

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 restore the original default options do:

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

7 Information about the R session used for this document

```
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] stats4    parallel  stats     graphics  grDevices utils     datasets  methods
[9] base

other attached packages:
[1] lava_1.6.5         doParallel_1.0.14  iterators_1.0.10   foreach_1.4.4
[5] data.table_1.12.2  BuyseTest_1.7.8    Rcpp_1.0.1         prodlim_2018.04.18

loaded via a namespace (and not attached):
 [1] compiler_3.5.1            prettyunits_1.0.2         base64enc_0.1-3
 [4] remotes_2.0.2             tools_3.5.1               testthat_2.0.0
 [7] digest_0.6.17             pkgbuild_1.0.2            pkgload_1.0.2
[10] memoise_1.1.0             butils.base_1.2           lattice_0.20-35
[13] rlang_0.3.1               Matrix_1.2-14             cli_1.0.1
[16] RcppArmadillo_0.9.400.3.0 withr_2.1.2               desc_1.2.0
[19] fs_1.2.6                  devtools_2.0.1            rprojroot_1.3-2
[22] grid_3.5.1                glue_1.3.0                R6_2.3.0
[25] processx_3.2.0            survival_2.44-1.1         sessioninfo_1.1.1
[28] callr_3.0.0               magrittr_1.5              codetools_0.2-15
[31] backports_1.1.2           ps_1.1.0                  splines_3.5.1
[34] usethis_1.4.0             assertthat_0.2.0          KernSmooth_2.23-15
[37] crayon_1.3.4
```