Overview of the package BuyseTest

Brice Ozenne

November 14, 2019

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. It is assumed that the reader is familiar 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.
  - `iid` to extract the iid decomposition of the estimator.
  - `getCount` to extract the number of favorable, unfavorable, neutral, and uninformative pairs.
  - `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). 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 \texttt{BuyseTest} package in the R session:

\begin{verbatim}
library(BuyseTest)
library(data.table)
\end{verbatim}

To illustrate the functionalities of the package, we will used the \texttt{veteran} dataset from the \texttt{survival} package:

\begin{verbatim}
data(veteran,package="survival")
head(veteran)
\end{verbatim}

\begin{verbatim}
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
\end{verbatim}

See \texttt{?veteran} for a presentation of the database.

Note: the \texttt{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:

\begin{verbatim}
utils::packageVersion("BuyseTest")
\end{verbatim}

\begin{verbatim}
[1] '1.8'
\end{verbatim}

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

\begin{verbatim}
sessionInfo()
\end{verbatim}

\begin{verbatim}
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] stats graphics grDevices utils datasets methods base

other attached packages:
[1] data.table_1.12.2 BuyseTest_1.8 Rcpp_1.0.1 prodlim_2018.04.18
\end{verbatim}
loaded via a namespace (and not attached):

[1] compiler_3.5.1  Matrix_1.2-14  tools_3.5.1  parallel_3.5.1
[5] survival_2.44-1.1  KernSmooth_2.23-15  splines_3.5.1  grid_3.5.1
[9] lava_1.6.5  stats4_3.5.1  lattice_0.20-35
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 `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:

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

**Generalized Pairwise Comparisons**

**Settings**
- 2 groups: Control = 1 and Treatment = 2
- 1 endpoint:

<table>
<thead>
<tr>
<th>priority</th>
<th>endpoint type</th>
<th>operator</th>
<th>threshold</th>
<th>censoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>time</td>
<td>time to event higher is favorable</td>
<td>20</td>
<td>status</td>
</tr>
</tbody>
</table>

- right-censored pairs: probabilistic score based on the survival curves

**Point estimation**

Gather the results in a BuyseRes object

---

2 the argument `method.inference` is set to 'none' to disable 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)

```r
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:

```r
testthat::expect_equal(BT.f, BT)
```

### 1.1 Displaying the results

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

```r
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)
> right-censored pairs: probabilistic score based on the survival curves
> results

<table>
<thead>
<tr>
<th>endpoint</th>
<th>threshold</th>
<th>total(%)</th>
<th>favorable(%)</th>
<th>unfavorable(%)</th>
<th>neutral(%)</th>
<th>uninf(%)</th>
<th>delta</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>20</td>
<td>100</td>
<td>37.78</td>
<td>46.54</td>
<td>15.68</td>
<td>0</td>
<td>-0.0877</td>
<td>-0.0877</td>
</tr>
</tbody>
</table>

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

```r
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)
> right-censored pairs: probabilistic score based on the survival curves
> results

<table>
<thead>
<tr>
<th>endpoint</th>
<th>threshold</th>
<th>total</th>
<th>favorable</th>
<th>unfavorable</th>
<th>neutral</th>
<th>uninf</th>
<th>delta</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>20</td>
<td>4692</td>
<td>1772.59</td>
<td>2183.89</td>
<td>735.52</td>
<td>0</td>
<td>-0.0877</td>
<td>-0.0877</td>
</tr>
</tbody>
</table>
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
- treatment groups: 1 (control) vs. 2 (treatment)
- right-censored pairs: probabilistic score based on the survival curves
- results

<table>
<thead>
<tr>
<th>endpoint</th>
<th>threshold</th>
<th>total(%)</th>
<th>favorable(%)</th>
<th>unfavorable(%)</th>
<th>neutral(%)</th>
<th>uninf(%)</th>
<th>delta</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>20</td>
<td>100</td>
<td>37.78</td>
<td>46.54</td>
<td>15.68</td>
<td>0</td>
<td>0.8117</td>
<td>0.8117</td>
</tr>
</tbody>
</table>

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, censoring = "status") + cont(karno, threshold = 0)
BT.H <- BuyseTest(ff2, data = veteran, method.inference = "none", trace = 0)
summary(BT.H)
```

Generalized pairwise comparisons with 2 prioritized endpoints

- 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 : ignored at lower priority endpoints
- results

<table>
<thead>
<tr>
<th>endpoint</th>
<th>threshold</th>
<th>total(%)</th>
<th>favorable(%)</th>
<th>unfavorable(%)</th>
<th>neutral(%)</th>
<th>uninf(%)</th>
<th>delta</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>20</td>
<td>100.00</td>
<td>37.78</td>
<td>46.54</td>
<td>15.68</td>
<td>0</td>
<td>-0.0877</td>
<td>-0.0877</td>
</tr>
<tr>
<td>karno</td>
<td>1e-12</td>
<td>15.68</td>
<td>5.78</td>
<td>7.11</td>
<td>2.78</td>
<td>0</td>
<td>-0.0133</td>
<td>-0.1009</td>
</tr>
</tbody>
</table>

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, method.inference = "none", trace = 0)
summary(BT.nH)
```
Generalized pairwise comparisons with 2 endpoints

- 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: ignored at lower priority endpoints

results

<table>
<thead>
<tr>
<th>endpoint</th>
<th>threshold</th>
<th>weight</th>
<th>total(%)</th>
<th>favorable(%)</th>
<th>unfavorable(%)</th>
<th>neutral(%)</th>
<th>uninf(%)</th>
<th>delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>20</td>
<td>1</td>
<td>100</td>
<td>37.78</td>
<td>46.54</td>
<td>15.68</td>
<td>0</td>
<td>-0.0877</td>
</tr>
<tr>
<td>karno</td>
<td>1e-12</td>
<td>1</td>
<td>100</td>
<td>41.82</td>
<td>44.95</td>
<td>13.24</td>
<td>0</td>
<td>-0.0313</td>
</tr>
</tbody>
</table>

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
ff2w <- trt ∼ tte(time, threshold = 20, censoring = "status", weight = 0.8)
ff2w <- update.formula(ff2w, . ~ . + cont(karno, threshold = 0, weight = 0.2))
BT.nHw <- BuyseTest(ff2w, hierarchical = FALSE, data = veteran,
                    method.inference = "none", trace = 0)
summary(BT.nHw)
```

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:

```r
ffop <- trt ~ tte(time, censoring = "status", threshold = 20, operator = "<0")
BTinv <- BuyseTest(ffop, data = veteran, 
                   method.inference = "none", trace = 0)
BTinv
```

<table>
<thead>
<tr>
<th>endpoint</th>
<th>threshold</th>
<th>delta</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>20</td>
<td>0.0844</td>
<td>0.0844</td>
</tr>
</tbody>
</table>

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:

```r
ffop2 <- update(ffop, . ~ . + tte(time, "status", 10, ">0"))
try(BuyseTest(ffop2, 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.

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:

```r
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)
```

**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**: ignored at lower priority endpoints
- **uninformative pairs**: no contribution at the current endpoint, analyzed at later endpoints
- **results**

<table>
<thead>
<tr>
<th>endpoint</th>
<th>threshold</th>
<th>total(%)</th>
<th>favorable(%)</th>
<th>unfavorable(%)</th>
<th>neutral(%)</th>
<th>uninf(%)</th>
<th>delta</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>20</td>
<td>100.00</td>
<td>36.06</td>
<td>45.77</td>
<td>17.33</td>
<td>0.85</td>
<td>-0.0971</td>
<td>-0.0971</td>
</tr>
<tr>
<td></td>
<td>25.38</td>
<td>14.33</td>
<td>8.77</td>
<td>2.28</td>
<td>0.00</td>
<td>0.2193</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>45.69</td>
<td>12.69</td>
<td>20.88</td>
<td>11.27</td>
<td>0.85</td>
<td>-0.1792</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>13.71</td>
<td>4.74</td>
<td>6.15</td>
<td>2.81</td>
<td>0.00</td>
<td>-0.1034</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.23</td>
<td>4.30</td>
<td>9.97</td>
<td>0.96</td>
<td>0.00</td>
<td>-0.3722</td>
<td></td>
<td></td>
</tr>
<tr>
<td>karno</td>
<td>1e-12</td>
<td>18.17</td>
<td>6.72</td>
<td>8.07</td>
<td>3.38</td>
<td>0.00</td>
<td>-0.0135</td>
<td>-0.1106</td>
</tr>
<tr>
<td></td>
<td>2.28</td>
<td>0.76</td>
<td>0.94</td>
<td>0.59</td>
<td>0.00</td>
<td>-0.0071</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.12</td>
<td>4.33</td>
<td>5.75</td>
<td>2.03</td>
<td>0.00</td>
<td>-0.0311</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.81</td>
<td>1.46</td>
<td>0.85</td>
<td>0.51</td>
<td>0.00</td>
<td>0.0448</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.96</td>
<td>0.17</td>
<td>0.54</td>
<td>0.25</td>
<td>0.00</td>
<td>-0.0241</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that here the numbers in the favorable/unfavorable/neutral/uninformative 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:

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

<table>
<thead>
<tr>
<th>Id</th>
<th>treatment</th>
<th>tumor</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>Yes</td>
<td>Yes</td>
<td>15</td>
</tr>
<tr>
<td>2:</td>
<td>No</td>
<td>Yes</td>
<td>20</td>
</tr>
</tbody>
</table>

\(^3\)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:

```r
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)
> neutral pairs : ignored at lower priority endpoints
> results

<table>
<thead>
<tr>
<th>endpoint</th>
<th>threshold</th>
<th>total(%)</th>
<th>favorable(%)</th>
<th>unfavorable(%)</th>
<th>neutral(%)</th>
<th>uninf(%)</th>
<th>delta</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>tumor</td>
<td>0.5</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>size</td>
<td>1e-12</td>
<td>100</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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:

```r
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)
> neutral pairs : re-analyzed using lower priority endpoints
> results

<table>
<thead>
<tr>
<th>endpoint</th>
<th>threshold</th>
<th>total(%)</th>
<th>favorable(%)</th>
<th>unfavorable(%)</th>
<th>neutral(%)</th>
<th>uninf(%)</th>
<th>delta</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>tumor</td>
<td>0.5</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>size</td>
<td>1e-12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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". This approach gives valid p-values, regardless to the sample size, for testing the absence of a difference between the groups.

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

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

```r
BT.boot <- BuyseTest(trt ~ tte(time, threshold = 20, censoring = "status"),
                      data = veteran, trace = 0, method.inference = "bootstrap")
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
  confidence intervals/p-values computed using the quantiles of the empirical 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
```

```r
p.value CI [2.5 ; 97.5]
0.393 [-0.2922;0.1013]
```

0.355

0.393
• normal approximation setting the argument `method.inference` to "u-statistic". In large enough samples, this approach gives valid p-values and confidence intervals.

```r
BT.ustat <- BuyseTest(trt ~ tte(time, threshold = 20, censoring = "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

<table>
<thead>
<tr>
<th>endpoint</th>
<th>threshold</th>
<th>total(%)</th>
<th>favorable(%)</th>
<th>unfavorable(%)</th>
<th>neutral(%)</th>
<th>uninf(%)</th>
<th>delta</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>20</td>
<td>100</td>
<td>37.78</td>
<td>46.54</td>
<td>15.68</td>
<td>0</td>
<td>-0.0877</td>
<td>-0.0877</td>
</tr>
</tbody>
</table>

p.value CI [2.5 ; 97.5]
0.37161 [-0.2735;0.1045]

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 argument `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, roughly, 2-digit precision, as examplified below:

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

[1] 0.0466 0.0466 0.0526 0.0544 0.0476 0.0512 0.0496 0.0488 0.0502 0.0475

Here we get a reasonable 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. In particular, possibly large matrices (number of neutral/uninformative pairs times the number of jumps of the survival function) may be involved and can cause the procedure to be slow or to crash due to lack of free memory.
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`:

```r
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:

```r
getAddress(BT.keep, endpoint = 1)
```

<table>
<thead>
<tr>
<th>index.1</th>
<th>index.2</th>
<th>favorable</th>
<th>unfavorable</th>
<th>neutral</th>
<th>uninf</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2:</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3:</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4:</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5:</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

---

| 4688:   | 65      | 137       | 0           | 1       | 0     | 0      |
| 4689:   | 66      | 137       | 0           | 1       | 0     | 1      |
| 4690:   | 67      | 137       | 0           | 1       | 0     | 1      |
| 4691:   | 68      | 137       | 0           | 1       | 0     | 1      |
| 4692:   | 69      | 137       | 0           | 1       | 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:

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

<table>
<thead>
<tr>
<th>trt</th>
<th>celltype</th>
<th>time</th>
<th>status</th>
<th>karno</th>
<th>diagtime</th>
<th>age</th>
<th>prior</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>squamous</td>
<td>999</td>
<td>1</td>
<td>90</td>
<td>12</td>
<td>54</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>squamous</td>
<td>72</td>
<td>1</td>
<td>60</td>
<td>7</td>
<td>69</td>
<td>0</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>endpoint threshold</th>
<th>delta</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>20</td>
<td>-0.0877</td>
</tr>
<tr>
<td>karno</td>
<td>1e-12</td>
<td>-0.0133</td>
</tr>
</tbody>
</table>

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

```r
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:

```r
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 ...
 ..\$ : 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 ...
 ..\$ : 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 ...
 ..\$ : NULL
 ..\$ : chr [1:3] "time" "survival" "dSurvival"
$ survJumpT: num [1:51, 1:3] 1 2 7 8 13 15 18 19 20 21 ...
 ..\$ : NULL
 ..\$ : chr [1:3] "time" "survival" "dSurvival"
$ 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:

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

<table>
<thead>
<tr>
<th>index.1</th>
<th>index.2</th>
<th>indexWithinStrata.1</th>
<th>indexWithinStrata.2</th>
<th>favorable</th>
<th>unfavorable</th>
<th>neutral</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>71</td>
<td>22</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0.6950827 0.3049173</td>
</tr>
</tbody>
</table>

uninf  weight

1: 0 1

In the dataset this corresponds to:

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

<table>
<thead>
<tr>
<th>trt</th>
<th>celltype</th>
<th>time</th>
<th>status</th>
<th>karno</th>
<th>diagtime</th>
<th>age</th>
<th>prior</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>smallcell</td>
<td>97</td>
<td>0</td>
<td>60</td>
<td>5</td>
<td>67</td>
<td>0</td>
</tr>
<tr>
<td>71</td>
<td>squamous</td>
<td>112</td>
<td>1</td>
<td>80</td>
<td>6</td>
<td>60</td>
<td>0</td>
</tr>
</tbody>
</table>

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:

```r
tSurv <- outSurv$survTimeC[22,]
tSurv
```

<table>
<thead>
<tr>
<th>time</th>
<th>SurvivalC-threshold</th>
<th>SurvivalC_0</th>
<th>SurvivalC+threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>97.0000000</td>
<td>0.5615232</td>
<td>0.5171924</td>
<td>0.4235463</td>
</tr>
</tbody>
</table>

SurvivalT-threshold  SurvivalT_0  SurvivalT+threshold

<table>
<thead>
<tr>
<th>time</th>
<th>SurvivalT-threshold</th>
<th>SurvivalT_0</th>
<th>SurvivalT+threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4558824</td>
<td>0.3643277</td>
<td>0.2827500</td>
<td></td>
</tr>
</tbody>
</table>

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

```r
Sc97 <- tSurv["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:

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

<table>
<thead>
<tr>
<th>time</th>
<th>SurvivalC-threshold</th>
<th>SurvivalC_0</th>
<th>SurvivalC+threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>112.0000000</td>
<td>0.5319693</td>
<td>0.4549201</td>
<td>0.3594915</td>
</tr>
</tbody>
</table>

SurvivalT-threshold  SurvivalT_0  SurvivalT+threshold

<table>
<thead>
<tr>
<th>time</th>
<th>SurvivalT-threshold</th>
<th>SurvivalT_0</th>
<th>SurvivalT+threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3801681</td>
<td>0.2827500</td>
<td>0.2827500</td>
<td></td>
</tr>
</tbody>
</table>

15
Since we are interested in the survival in the control arm at the event time plus threshold:

\[
\text{Sc132} \leftarrow \text{iSurv["SurvivalC+threshold"]}
\]

\[
\text{Sc132}
\]

**SurvivalC+threshold**

0.3594915

The probability in favor of the control is then:

\[
\frac{\text{Sc132}}{\text{Sc97}}
\]

**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 `calcIntegralScore_cpp` that takes as argument a matrix containing the survival and the jumps in survival, e.g.:

\[
\text{head(outSurv$survJumpT)}
\]

\[
\begin{array}{ccc}
\text{time} & \text{survival} & \text{dSurvival} \\
[1,] & 1 & 0.7681159 & -0.02941176 \\
[2,] & 2 & 0.7536232 & -0.01470588 \\
[3,] & 7 & 0.7388463 & -0.02941176 \\
[4,] & 8 & 0.7388463 & -0.02941176 \\
[5,] & 13 & 0.7092924 & -0.01470588 \\
[6,] & 15 & 0.6945155 & -0.02941176
\end{array}
\]

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

\[
\text{getPairScore(BT.keep2, endpoint = 1, rm.withinStrata = FALSE)[148]}
\]

\[
\begin{array}{cccccc}
\text{index.1} & \text{index.2} & \text{indexWithinStrata.1} & \text{indexWithinStrata.2} & \text{favorable} & \text{unfavorable} & \text{neutral} \\
1: & 10 & 72 & 10 & 3 & 0.5058685 & 0.3770426 & 0.1170889 \\
\text{uninf weight} \\
1: & 0 & 1
\end{array}
\]

which corresponds to the observations:

\[
\text{veteran[c(10,72),]}
\]

\[
\begin{array}{cccccccccc}
\text{trt} & \text{celltype} & \text{time} & \text{status} & \text{karno} & \text{diagtime} & \text{age} & \text{prior} \\
10 & 1 & \text{squamous} & 100 & 0 & 70 & 6 & 70 & 0 \\
72 & 2 & \text{squamous} & 87 & 0 & 80 & 3 & 48 & 0
\end{array}
\]
The probability in favor of the treatment \((p_F)\) and control \((p_{UF})\) can be computed as:

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

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

```r
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:

```r
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
```

<table>
<thead>
<tr>
<th>favorable</th>
<th>unfavorable</th>
</tr>
</thead>
<tbody>
<tr>
<td>lowerBound</td>
<td>0.5058685</td>
</tr>
<tr>
<td>upperBound</td>
<td>0.3770426</td>
</tr>
</tbody>
</table>

```
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:

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

<table>
<thead>
<tr>
<th>treatment</th>
<th>toxicity</th>
<th>eventtimeUncensored</th>
<th>eventtimeCensoring</th>
<th>eventtime</th>
<th>status</th>
<th>status1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>C</td>
<td>0</td>
<td>0.1588268</td>
<td>2.6268101</td>
<td>0.1588268</td>
<td>1</td>
</tr>
<tr>
<td>2:</td>
<td>C</td>
<td>1</td>
<td>1.7204676</td>
<td>0.2000192</td>
<td>0.2000192</td>
<td>0</td>
</tr>
<tr>
<td>3:</td>
<td>C</td>
<td>1</td>
<td>0.4900490</td>
<td>0.5747995</td>
<td>0.4900490</td>
<td>1</td>
</tr>
<tr>
<td>4:</td>
<td>C</td>
<td>0</td>
<td>0.1138545</td>
<td>1.5188001</td>
<td>0.1138545</td>
<td>1</td>
</tr>
<tr>
<td>5:</td>
<td>C</td>
<td>1</td>
<td>0.5191035</td>
<td>3.8340048</td>
<td>0.5191035</td>
<td>1</td>
</tr>
<tr>
<td>6:</td>
<td>C</td>
<td>0</td>
<td>0.9405830</td>
<td>1.9078657</td>
<td>0.9405830</td>
<td>1</td>
</tr>
</tbody>
</table>

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

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

[1] 0.317

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

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

<table>
<thead>
<tr>
<th>endpoint</th>
<th>threshold</th>
<th>delta</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventtimeUncensored</td>
<td>1</td>
<td>0.2401</td>
<td>0.2401</td>
</tr>
</tbody>
</table>

using the censored survival times:

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

<table>
<thead>
<tr>
<th>endpoint</th>
<th>threshold</th>
<th>delta</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventtime</td>
<td>1</td>
<td>0.1363</td>
<td>0.1363</td>
</tr>
</tbody>
</table>

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:

```r
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)
> 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 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:

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

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

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

<table>
<thead>
<tr>
<th>favorable</th>
<th>unfavorable</th>
<th>neutral</th>
<th>uninf</th>
<th>favorableC</th>
<th>unfavorableC</th>
<th>neutralC</th>
<th>uninfC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.00000</td>
<td>0.00000</td>
<td>1.81657</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.00000</td>
<td>1.81657</td>
<td>0.00000</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1.81657</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0</td>
</tr>
</tbody>
</table>

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

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

[1] 1.81657

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:

```r
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)
> right-censored pairs: deterministic score or uninformative
> uninformative pairs: score equals the averaged score of all informative pairs
> results

<table>
<thead>
<tr>
<th>endpoint</th>
<th>threshold total(%)</th>
<th>favorable(%)</th>
<th>unfavorable(%)</th>
<th>neutral(%)</th>
<th>uninf(%)</th>
<th>delta</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventtime</td>
<td>1</td>
<td>100</td>
<td>37.11</td>
<td>12.34</td>
<td>50.54</td>
<td>0</td>
<td>0.2477 0.2477</td>
</tr>
</tbody>
</table>

Looking at the scores of the pairs:

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

<table>
<thead>
<tr>
<th></th>
<th>favorable</th>
<th>unfavorable</th>
<th>neutral</th>
<th>uninf</th>
<th>favorableC</th>
<th>unfavorableC</th>
<th>neutralC</th>
<th>uninfC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>1.000000</td>
<td>0</td>
</tr>
<tr>
<td>2:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.371118</td>
<td>0.1234396</td>
<td>0.5054424</td>
<td>0</td>
</tr>
<tr>
<td>3:</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>4:</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th></th>
<th>favorable</th>
<th>unfavorable</th>
<th>neutral</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>0.371118</td>
<td>0.1234396</td>
<td>0.5054424</td>
</tr>
</tbody>
</table>
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:

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

<table>
<thead>
<tr>
<th>treatment</th>
<th>toxicity</th>
<th>score</th>
<th>eventtime</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>C</td>
<td>0.54361539</td>
<td>1.8252132</td>
<td>0</td>
</tr>
<tr>
<td>2:</td>
<td>C</td>
<td>-0.70762484</td>
<td>2.9489056</td>
<td>1</td>
</tr>
<tr>
<td>3:</td>
<td>C</td>
<td>-0.36944577</td>
<td>0.7213402</td>
<td>0</td>
</tr>
<tr>
<td>4:</td>
<td>C</td>
<td>-1.32197565</td>
<td>0.6322603</td>
<td>1</td>
</tr>
<tr>
<td>5:</td>
<td>C</td>
<td>1.28059746</td>
<td>0.2212117</td>
<td>0</td>
</tr>
<tr>
<td>6:</td>
<td>T</td>
<td>0.01874617</td>
<td>1.453481</td>
<td>0</td>
</tr>
<tr>
<td>7:</td>
<td>T</td>
<td>-0.18425254</td>
<td>0.4855601</td>
<td>0</td>
</tr>
<tr>
<td>8:</td>
<td>T</td>
<td>-1.37133055</td>
<td>0.2547505</td>
<td>0</td>
</tr>
<tr>
<td>9:</td>
<td>T</td>
<td>-0.59916772</td>
<td>1.0340368</td>
<td>0</td>
</tr>
<tr>
<td>10:</td>
<td>T</td>
<td>0.29454513</td>
<td>0.3579324</td>
<td>1</td>
</tr>
</tbody>
</table>

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:

```r
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
```

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

This functionality is based on the `sim` function of the `lava` package (https://github.com/kkholst/lava)
5 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:

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

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

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

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

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

```r
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 0.2477 0.2477 100
```

To restore the original default options do:

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


