Wilcoxon test via GPC

Brice Ozenne

June 14, 2024

Generalized Pairwise comparisons (GPC) include the Wilcoxon rank sum test as a specific case. This vignette explores the connections between the `BuyseTest` output and the more standard implementation of the wilcoxon-test.

1 Single Wilcoxon test

1.1 Exact test

Consider the first 'two sample test' example from the help page section of `stats::wilcox.test`:

```
x <- c(0.80, 0.83, 1.89, 1.04, 1.45, 1.38, 1.91, 1.64, 0.73, 1.46)
y <- c(1.15, 0.88, 0.90, 0.74, 1.21)
df <- rbind(data.frame(value = x, group="x"),
            data.frame(value = y, group="y"))
```

We can perform a Wilcoxon test using the `wilcox.test` function:

```
wilcox.test(value ~ group, data = df)
```

```

        Wilcoxon rank sum exact test

data:  value by group
W = 35, p-value = 0.2544
alternative hypothesis: true location shift is not equal to 0
```

which, with such a small sample, can perform an exact test, i.e., consider all possible permutation of the group variable. It unfortunately does not ouput any effect size (just the test statistic and corresponding p-value). The package *asht* contains an alternative implementation:

```
asht::wmwTest(value ~ group, data = df, method = "exact.ce")
```

```

        exact Wilcoxon-Man-Whitney test (confidence interval requires proportional odds
        assumption, but test does not)

data:  value by group
```

```
Mann-Whitney estimate = 0.3, p-value = 0.2544
alternative hypothesis: two distributions are not equal
95 percent confidence interval:
 0.08292978 0.63269022
sample estimates:
Mann-Whitney estimate
                  0.3
```

which output an estimate, the probability that a randomly chosen observation from one group has higher value than a randomly chosen observation from the other group, which is refered to as Mann-Whitney parameter or probabilistic index. To match those results with GPC we can use a permutation test:
⚠ the argument `add.halfNeutral` should be set to `TRUE` to adequatly handle ties

```
eperm.BT <- BuyseTest(group ~ cont(value), data = df, add.halfNeutral = TRUE,
                      method.inference = "permutation", n.resampling = 1e4,
                      trace = FALSE, cpus = 5, seed = 10)
confint(eperm.BT, statistic = "favorable")
```

```
      estimate        se lower.ci upper.ci null   p.value
value      0.3 0.1632342       NA       NA  0.5 0.2566743
```

Up to the Monte Carlo error for the p-value calculation, which can be made arbitrarily small by increasing the number of permutations (argument `n.resampling`), the resuls are identical. Note that the 'default' statistical inference method based on asymptotic U-statistic theory:

```
eU.BT <- BuyseTest(group ~ cont(value), data = df,
                   add.halfNeutral = TRUE)
confint(eU.BT, statistic = "favorable")
```

```
      estimate        se  lower.ci  upper.ci null  p.value
value      0.3 0.1334166 0.1098282 0.5981833  0.5 0.182315
```

leads to a different p-value as a different null hypothesis is being tested here: probabilistic index equal 0.5 instead of equality in distribution. This p-value corresponds, up to some small sample approximation and the Monte Carlo error, to the one obtain with a studentized permutation:

```
etperm.BT <- BuyseTest(group ~ cont(value), data = df, add.halfNeutral = TRUE,
                      method.inference = "studentized permutation", n.resampling = 1e4,
                      trace = FALSE, seed = 10)
confint(etperm.BT, statistic = "favorable")
```

```
      estimate        se lower.ci upper.ci null   p.value
value      0.3 0.1334166       NA       NA  0.5 0.1916808
```

1.2 Approximate test

Consider now a bigger (artificial) dataset:

```
set.seed(10)
df2 <- rbind(data.frame(value = round(rnorm(50),2), group="x"),
             data.frame(value = round(rnorm(50),2), group="y"))
any(duplicated(df2$value)) ## test whether there are any ties
```

```
[1] TRUE
```

We can again perform a Wilcoxon test using the `wilcox.test` function:

```
wilcox.test(value ~ group, data = df2)
```

```

        Wilcoxon rank sum test with continuity correction

data:  value by group
W = 967.5, p-value = 0.05188
alternative hypothesis: true location shift is not equal to 0
```

or, equivalenty, with the `wmwTest` function:

```
wmwTest(value ~ group, data = df2)
```

```

        Wilcoxon-Mann-Whitney test with continuity correction (confidence
        interval requires proportional odds assumption, but test does not)

data:  value by group
Mann-Whitney estimate = 0.613, tie factor = 0.99995, p-value = 0.05188
alternative hypothesis: two distributions are not equal
95 percent confidence interval:
 0.4990803 0.7138973
sample estimates:
Mann-Whitney estimate
                0.613
```

In either case, an exact test would be too computationally demanding and an approximate test is performed instead, which assumes a normaly distributed test statistic. The BuyseTest package will not be able to match these results due to the continuity correction. Without continuity correction, e.g.:

```
wmwTest(value ~ group, data = df2, correct = FALSE)
```

```

        Wilcoxon-Mann-Whitney test (confidence interval requires proportional odds
        assumption, but test does not)
```

```
data:  value by group
Mann-Whitney estimate = 0.613, tie factor = 0.99995, p-value = 0.05147
alternative hypothesis: two distributions are not equal
95 percent confidence interval:
 0.4992803 0.7137196
sample estimates:
Mann-Whitney estimate
                0.613
```

it is possible to retrieve the exact same p-value by evaluating the variance of the permutation distribution and assuming a normally distributed test statistic. In this simple example this can be done using an analytic formula (Anderson and Verbeeck, 2023):
⚠ the code, kindly provided by the authors of the paper, has been ported to the package with minimal change. It is therefore meant to be used in the context of the original publication and not in the more general setting covered by the package (strata, right-censoring, ...)

```
eperm.BT2 <- BuyseTest(group ~ cont(value), data = df2, add.halfNeutral = TRUE,
                       method.inference = "varexact-permutation")
confint(eperm.BT2, statistic = "favorable")
```

```
      estimate         se lower.ci upper.ci null    p.value
value    0.613 0.05802219       NA       NA  0.5 0.05147115
```

or, more generally, using a resampling method:

```
eperm.BT2 <- BuyseTest(group ~ cont(value), data = df2, add.halfNeutral = TRUE,
                       method.inference = "permutation", n.resampling = 1e4,
                       trace = FALSE, cpus = 5, seed = 10)
confint(eperm.BT2, statistic = "favorable", method.ci.resampling = "gaussian")
```

```
      estimate         se lower.ci upper.ci null    p.value
value    0.613 0.05814569       NA       NA  0.5 0.05118099
```

2 Multiple Wilcoxon tests

Consider now the case where we would like to compare one reference group (here strata `a`) to multiple treatment groups (here strata `b,c,d,e`). We will consider the following dataset:

```
set.seed(35)
dt <- simBuyseTest(n.T=25, n.strata = 5)
dt$id <- paste0("id",1:NROW(dt))
dt$strata <- as.character(dt$strata)
head(dt)
```

```
       id treatment  eventtime status toxicity      score strata
   <char>    <fctr>      <num>  <num>   <fctr>      <num> <char>
1:    id1         C 0.03384999      1      yes  0.4777913      b
2:    id2         C 0.65039474      0       no -1.1048190      d
3:    id3         C 1.00647502      1       no -0.1407630      b
4:    id4         C 0.01129603      1      yes -0.5512507      a
5:    id5         C 0.22249748      1       no  1.0465250      d
6:    id6         C 0.07400412      0       no -2.0053855      d
```

We can apply the GPC procedure to each pair of group:

```
BuyseTest.options(order.Hprojection=1);BuyseTest.options(trace=0)

ls.BT <- list("b-a=0" = BuyseTest(strata ~ cont(score), add.halfNeutral = TRUE,
                                  data = dt[dt$strata %in% c("a","b"),],
                                  method.inference = "u-statistic"),
              "c-a=0" = BuyseTest(strata ~ cont(score), add.halfNeutral = TRUE,
                                  data = dt[dt$strata %in% c("a","c"),],
                                  method.inference = "u-statistic"),
              "d-a=0" = BuyseTest(strata ~ cont(score), add.halfNeutral = TRUE,
                                  data = dt[dt$strata %in% c("a","d"),],
                                  method.inference = "u-statistic"),
              "e-a=0" = BuyseTest(strata ~ cont(score), add.halfNeutral = TRUE,
                                  data = dt[dt$strata %in% c("a","e"),],
                                  method.inference = "u-statistic")
              )

M.confint <- do.call(rbind,lapply(ls.BT,confint, statistic = "favorable"))
cbind(M.confint,adj.p.value = p.adjust(M.confint[,"p.value"], method = "bonferroni"))
```

```
       estimate        se  lower.ci  upper.ci null    p.value adj.p.value
b-a=0 0.4090909 0.1542200 0.1654639 0.7073759  0.5 0.56434599   1.0000000
c-a=0 0.4375000 0.1465755 0.1948678 0.7142379  0.5 0.67306460   1.0000000
d-a=0 0.2500000 0.1010153 0.1039078 0.4893302  0.5 0.04143057   0.1657223
e-a=0 0.3333333 0.1360828 0.1308601 0.6241219  0.5 0.25767454   1.0000000
```

Because we compare the treatment groups to the same reference, the test statistics are correlated and a Bonferroni adjustment would not be optimal. A better (but still not optimal adjustment) is the max-test adjustment which can be obtained via the `BuyseMultComp` function:

```
e.mc <- BuyseMultComp(ls.BT, statistic = "favorable", cluster = "id", global = TRUE)
print(e.mc, cols = c("estimate","se","p.value","adj.p.value"))
```

```
  - Multivariate test: p.value = 0.2645493 (df = 4)
  - Univariate tests:
       estimate        se    p.value adj.p.value
b-a=0 0.4090909 0.1542200 0.56434599   0.9289219
c-a=0 0.4375000 0.1465755 0.67306460   0.9752151
d-a=0 0.2500000 0.1010153 0.04143057   0.1223430
e-a=0 0.3333333 0.1360828 0.25767454   0.5831344
```

Here the smallest p-value has been multiplied by a factor 2.64 instead of 4. This is thanks to the rather strong correlation between the test statistics:

```
M.cor <- cor(lava::iid(e.mc))
dimnames(M.cor) <- list(names(ls.BT),names(ls.BT))
M.cor
```

```
          b-a=0     c-a=0     d-a=0     e-a=0
b-a=0 1.0000000 0.6519486 0.5601058 0.7520401
c-a=0 0.6519486 1.0000000 0.4240003 0.5439927
d-a=0 0.5601058 0.4240003 1.0000000 0.5051815
e-a=0 0.7520401 0.5439927 0.5051815 1.0000000
```

References

Anderson, W. N. and Verbeeck, J. (2023). Exact permutation and bootstrap distribution of generalized pairwise comparisons statistics. *Mathematics*, 11(6):1502.