Wilcoxon test via GPC

Brice Ozenne

December 14, 2021

1 Single Wilcoxon test

Generalized Pairwise comparisons include the Wilcoxon rank sum test as a specific case.
Consider the following dataset (from the example section of `stats::wilcox.test`):

```
x <- c(1.83,  0.50,  1.62,  2.48, 1.68, 1.88, 1.55, 3.06, 1.30)
y <- c(0.878, 0.647, 0.598, 2.05, 1.06, 1.29, 1.06, 3.14, 1.29)
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 test with continuity correction

data:  value by group
W = 58, p-value = 0.1329
alternative hypothesis: true location shift is not equal to 0

Warning message:
In wilcox.test.default(x = c(1.83, 0.5, 1.62, 2.48, 1.68, 1.88,  :
  cannot compute exact p-value with ties
```

It unfortunately does not ouput any effect size (just the test statistic and corresponding p-value). The package *asht* contains an alternative implementation:

```
library(asht)
wmwTest(value ~ group, data = df, method = "asymptotic")
```

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

data:  value by group
Mann-Whitney estimate = 0.28395, tie factor = 0.99794, p-value = 0.1329
alternative hypothesis: two distributions are not equal
```

```
95 percent confidence interval:
 0.1142978 0.5614097
sample estimates:
Mann-Whitney estimate
            0.2839506
```

which does output an estimate[1]. It matches exactly the p.value and is based on an asymptotic result. It is also possible to get an exact p-value [2]:

```
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 = 0.28395, p-value = 0.1299
alternative hypothesis: two distributions are not equal
95 percent confidence interval:
 0.09721823 0.56323417
sample estimates:
Mann-Whitney estimate
            0.2839506
```

To match those results with GPC we can use a permutation test (⚠ remember to add the argument `add.halfNeutral` to `TRUE` to handle ties the same way as the previous tests):

```
eperm.BT <- BuyseTest(group ~ cont(value), data = df, add.halfNeutral = TRUE,
        method.inference = "permutation", n.resampling = 10000,
        trace = FALSE, seed = 10)
confint(eperm.BT, statistic = "favorable")
```

```
             estimate       se  lower.ci  upper.ci null p.value
value_1e-12 0.2839506 0.140185 0.1050267 0.5725376  0.5  0.1242
Warning message:
In .local(object, ...) :
  Confidence intervals are computed under the null hypothesis and therefore may not be valid.
```

The estimate is precisely the same and the p-value approximately the same. Instead of permutation, we could use the asymptotic theory to obtain p-values and (valid) confidence intervals:

```
BuyseTest.options(order.Hprojection=2)
eU.BT <- BuyseTest(group ~ cont(value), data = df,
    method.inference = "u-statistic",
    add.halfNeutral = TRUE, trace = FALSE)
confint(eU.BT, statistic = "favorable")
```

[1] Mann-Whitney parameter, i.e. probability that a randomly chosen observation from one group has higher value than a randomly chosen observation from the other group

[2] this is only feasible in small samples - otherwise the procedure becomes computationnally challenging

```
             estimate        se   lower.ci  upper.ci null   p.value
value_1e-12 0.2839506 0.1401461 0.09313769 0.6049215  0.5 0.1796262
```

Unsuprisingly, we get the same estimate. However the p-value seems quite a bit different. This might be explained by the fact that this approach does not assume iid[3] observations but only iid observations within each group. A studentised permutation, which is exactly (instead of asymptotically) valid under the same assumption, gives a somewhat similar p-value:

```
etperm.BT <- BuyseTest(group ~ cont(value), data = df, add.halfNeutral = TRUE,
        method.inference = "studentized permutation", n.resampling = 10000,
        trace = FALSE, seed = 10)
confint(etperm.BT, statistic = "favorable")
```

```
             estimate        se  lower.ci  upper.ci null p.value
value_1e-12 0.2839506 0.1401461 0.1006681 0.5809142  0.5   0.163
Warning message:
In .local(object, ...) :
  Confidence intervals are computed under the null hypothesis and therefore may not be valid.
```

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)
```

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

[3] iid=independent and identically distributed

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$strata %in% c("a","b"),]),
       "c-a=0" = BuyseTest(strata ~ cont(score), add.halfNeutral = TRUE,
      data = dt[dt$strata %in% c("a","c"),]),
       "d-a=0" = BuyseTest(strata ~ cont(score), add.halfNeutral = TRUE,
      data = dt[dt$strata %in% c("a","d"),]),
       "e-a=0" = BuyseTest(strata ~ cont(score), add.halfNeutral = TRUE,
      data = dt[dt$strata %in% c("a","e"),])
       )

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 <- cov2cor(crossprod(attr(e.mc,"iid")))
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
```