Validation of choplump R package

by Michael P. Fay

November 1, 2024

Summary

This document outlines several ways we have tested the choplump R package.

- In Section 1 we calculate the exact choplump tests in two ways. One method is a crude and slower method that is easier to program, and the other method is the faster method which is used for the final `choplump` function. We test both methods with small sample sizes and get the same answers.
- In Section 2 we show that when we use the methods outlined in the `computation` vignette to calculate the usual Wilcoxon Rank sum test (but with many tied zeros), we obtain the same answer as from the previously developed `coin` package (Hothorn, Hornik, van de Wiel and Zeileis, 2006).
- In Section 3 we show that the asymptotic approximation to the p-value closely matches the exact p-value for sample sizes as small as 10 total non-zero responses in both groups.

1 Calculate Exact Test Two Ways

In this section, we introduce the `choplumpGeneral` function which is a slow version of the `choplump` test. It's structure closely matches the general description of the choplump test. The key to this function is the `chooseMatrix` function. The call `chooseMatrix(N,M)` produces a `choose(N,M)` by N matrix with each row a different permutation of M ones and $N - M$ zeros.

For example:

```
> chooseMatrix(5,2)

      [,1] [,2] [,3] [,4] [,5]
 [1,]    1    1    0    0    0
 [2,]    1    0    1    0    0
 [3,]    1    0    0    1    0
 [4,]    1    0    0    0    1
 [5,]    0    1    1    0    0
 [6,]    0    1    0    1    0
 [7,]    0    1    0    0    1
 [8,]    0    0    1    1    0
 [9,]    0    0    1    0    1
[10,]    0    0    0    1    1
```

Here is the general chopping function and the general choplump function:

```
> chopGeneral

function (d, W = "W", Z = "Z")
{
    ord <- order(d[, W], d[, Z])
    w <- d[ord, W]
    z <- d[ord, Z]
    M <- length(w[w != 0])
```

```
    m0 <- length(z[w != 0 & z == 0])
    m1 <- M - m0
    n1 <- length(z[z == 1])
    N <- length(w)
    n0 <- N - n1
    k0 <- n0 - m0
    k1 <- n1 - m1
    if (m0/n0 >= m1/n1) {
        a <- 0
        b <- k1 - floor((n1 * k0)/n0)
    }
    else {
        a <- k0 - floor((n0 * k1)/n1)
        b <- 0
    }
    wout <- w[(N + 1 - (M + a + b)):N]
    zout <- c(rep(0, a), rep(1, b), z[(N + 1 - M):N])
    dout <- data.frame(W = wout, Z = zout)
    dout
}
<bytecode: 0x55eb4616fe40>
<environment: namespace:choplump>
```

```
> choplumpGeneral
```

```
function (W, Z, testfunc = testfunc.wilcox.ties.general)
{
    w <- W
    z <- Z
    M <- length(w[w != 0])
    m0 <- length(z[w != 0 & z == 0])
    m1 <- M - m0
    n1 <- length(z[z == 1])
    N <- length(w)
    n0 <- N - n1
    k0 <- n0 - m0
    k1 <- n1 - m1
    d <- data.frame(W = w, Z = z)
    T0 <- testfunc(chopGeneral(d))
    cm <- chooseMatrix(N, n1)
    Nperm <- dim(cm)[1]
    Ti <- rep(NA, Nperm)
    for (i in 1:Nperm) {
        d$Z <- cm[i, ]
        Ti[i] <- testfunc(chopGeneral(d))
    }
    p.lower <- length(Ti[Ti <= T0])
    p.upper <- length(Ti[Ti >= T0])
    out <- c(p.lower = p.lower/Nperm, p.upper = p.upper/Nperm,
        p.2sided = min(1, 2 * min(p.lower, p.upper)/Nperm))
```

```
    out
}
<bytecode: 0x55eb461a2010>
<environment: namespace:choplump>
```

Now we calculate some p-values from some small data sets using both the `choplumpGeneral` function and the final `choplump` function. We show that both give the same answers.

```
> make.data<-function(N,M,SEED){
+     set.seed(SEED)
+     Z<-rep(0,N)
+     Z[sample(1:N,N/2,replace=FALSE)]<-1
+     test<-data.frame(W=c(rep(0,N-M),abs(rnorm(M))),Z=Z)
+     return(test)
+ }
> test<-make.data(10,6,SEED[1])
> test

             W Z
1  0.000000000 1
2  0.000000000 1
3  0.000000000 0
4  0.000000000 1
5  1.272429321 0
6  0.414641434 0
7  1.539950042 1
8  0.928567035 0
9  0.294720447 1
10 0.005767173 0

> choplumpGeneral(test$W,test$Z,testfunc=testfunc.wilcox.ties.general)

  p.lower   p.upper  p.2sided
0.7380952 0.2936508 0.5873016

> cout<-choplump(W~Z,data=test,use.ranks=TRUE,exact=TRUE)

calculating exact test...
requires  62  evaluations of test statistic

> cout

        Exact Choplump Rank Test

data:  W by Z
p-value = 0.5873
alternative hypothesis: two.sided

> cout$p.values

  p.lower   p.upper  p.2sided
0.7380952 0.2936508 0.5873016
```

Now we show the equivalence of the two functions for the difference in means test. Note to match directions we use the negative of the `TDiM` function.

```
> testfunc.DiM.general<-function(d){
+     -TDiM(d$W,d$Z)
+ }
> choplumpGeneral(test$W,test$Z,testfunc=testfunc.DiM.general)

  p.lower   p.upper  p.2sided
0.6349206 0.3809524 0.7619048

> choplump(W~Z,data=test,use.ranks=FALSE,exact=TRUE)$p.values

calculating exact test...
requires  62  evaluations of test statistic
  p.lower   p.upper  p.2sided
0.6349206 0.3809524 0.7619048
```

2 Calculate Exact Wilcoxon Rank sum test using Similar methods

In this section we show how we can use similar methods to those used in the choplump package to calculate either exact or approximate Wilcoxon rank sum test p-values. Then we can check out that the exact p-values match those output from `wilcox_test` in the `coin` package. In the process, we show how our algorithm can be faster in some cases when there are very many zeros.

```
> library(coin)
> test<-make.data(20,12,SEED[1])
> test

            W Z
1  0.00000000 1
2  0.00000000 1
3  0.00000000 1
4  0.00000000 1
5  0.00000000 0
6  0.00000000 0
7  0.00000000 1
8  0.00000000 0
9  0.48742905 0
10 0.73832471 0
11 0.57578135 1
12 0.30538839 0
13 1.51178117 1
14 0.38984324 1
15 0.62124058 0
16 2.21469989 0
17 1.12493092 1
18 0.04493361 0
19 0.01619026 1
20 0.94383621 0
```

```
> wilcox.manyzeros.exact(W=test$W,Z=test$Z)

  p.lower   p.upper  p.2sided
0.7977224 0.2171296 0.4342592

> test2<-data.frame(W=test$W,Z=as.factor(test$Z))
> wilcox_test(W~Z,data=test2,distribution="exact",alternative="less")

        Exact Wilcoxon-Mann-Whitney Test

data:  W by Z (0, 1)
Z = 0.82004, p-value = 0.7977
alternative hypothesis: true mu is less than 0

> wilcox_test(W~Z,data=test2,distribution="exact",alternative="greater")

        Exact Wilcoxon-Mann-Whitney Test

data:  W by Z (0, 1)
Z = 0.82004, p-value = 0.2171
alternative hypothesis: true mu is greater than 0

> wilcox_test(W~Z,data=test2,distribution="exact",alternative="two.sided")

        Exact Wilcoxon-Mann-Whitney Test

data:  W by Z (0, 1)
Z = 0.82004, p-value = 0.4343
alternative hypothesis: true mu is not equal to 0

> test<-make.data(1000,12,SEED[2])
> t0<-proc.time()
> wilcox.manyzeros.exact(W=test$W,Z=test$Z)

  p.lower   p.upper  p.2sided
0.5341409 0.4793681 0.9587362

> ## time for our algorithm
> proc.time()-t0

   user  system elapsed
  0.039   0.000   0.039

> test2<-data.frame(W=test$W,Z=as.factor(test$Z))
> t1<-proc.time()
> wilcox_test(W~Z,data=test2,distribution="exact",alternative="two.sided")

        Exact Wilcoxon-Mann-Whitney Test

data:  W by Z (0, 1)
Z = 0.0023222, p-value = 0.9587
alternative hypothesis: true mu is not equal to 0

> ## time for coin algorithm
> proc.time()-t1

   user  system elapsed
  0.298   0.008   0.305
```

3 Compare Asymptotic Approximation to Exact Calculation

To see how well the approximation performs, we simulate 200 data sets with $N = 100$ and $M = 10$. For this small sample size we can calculate the exact p-values. We randomly assign $N/2$ of the Z_i values to 1 and the others are 0. We take pseudo-random numbers for the X_i, where $X_i = |X_i^\dagger|$ and $X_i^\dagger \sim N(0, 1)$. We plot the bias (approximate p-value minus exact p-value) by the exact p-values in Figure 1, together with the 95% interquantile ranges of the bias. We see that even when M is as small as 10, the approximation does fairly well, with the 95% interquantile range of the bias for the rank tests equal to (-0.0179,0.003), and the similar statistic for the difference in means tests equal to (-0.0192,0.0197).

References

Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel and Achim Zeileis (2006). A Lego System for Conditional Inference. *The American Statistician* **60:** 257-263.

Figure 1: Comparison of Asymptotic Approximation and Exact P-values, Circles (solid lines) are Rank Tests and Triangles (dotted lines) are Difference in Means Tests. Lines enclose middle 95% of bias (asymptotic-exact).