Principles Statistical Data Analysis: HW3

Jan Alexander, Brecht Dewilde, Arthur Leloup

2019 - 2020

Contents

1 R function: `median.test(x,y)`

The function `median.test(x,y)` calculates a permutation p-value associated with $H_0 : F_x = F_y$ versus $H_A : median_x \neq median_y$. In case, the total amount of combinations is sufficiently small (i.e. less than 5000) a full permutation test will be performed otherwise the null-distribution is generated from 5000 random samples. The code with included comments is listed below.

```
# FUNCTION: calc.median.diff: --------------
# Function: calculate the difference in median between two groups.
#
# param: ind = the indices of the first group
# param: vec = the complete list with data from group 1 and group 2
#
calc.median.diff <- function(ind, vec){
  # make both groups
  group1 <- vec[ind]
  group2 <- vec[!(1:length(vec)) %in% ind]
  # calculate the difference in means
  return(median(group1) - median(group2))
}

# FUNCTION: median.test: --------------
# Function: calculate the p-value of the median difference between
# vector x and vector y, based on the null hypothesis F1(x) = F2(x)
#
# When the number of combinations is sufficiently small to do a full
```

```
# permutation test (limit at 5000), the full permutation test will
# be performed.
#
# param: x = vector values for group 1
# param: y = vector values group 2
#
median.test <- function(x, y){
  N <- 5000 # maximum number of permutations
  realization <- median(x) - median(y)
  len_x <- length(x)
  len_y <- length(y)
  vec <- c(x, y)
  len_vec <- len_x + len_y
  if(choose(n = len_vec, len_x) < N){
    #A limited number of combinations: full permutation test
    median.diff <- combn(len_vec,
                         len_x,
                         function(ind){
                           return(calc.median.diff(ind, vec))
                         })
  } else {
    #Too many combinations - A sample test will be performed.
    median.diff <- replicate(N,
                             calc.median.diff(
                               sample(c(1:len_vec),len_x),
                               vec)
    )
  }
  # The distribution will be symmetrical. The p-value can be calculated by
  # the absolute value.
  return(mean(abs(realization) <= abs(median.diff)))
}
```

2 Comparison to other tests

To evaluate the performance of the median test, it is compared with the permutation t-test and the Wilcoxon-Mann-Whitney test as these are commonly used for comparing independent samples.

The three tests where compared on the basis of their **power** and **type-I error** for a given sample size and delta. These statistics where acquired through Monte Carlo simulation (1000 simulations) with two different sample sizes ($n_1 = 20$ and $n_2 = 40$).

2.1 Power

To evaluate the power of the median test with the other tests, multiple samples were randomly drawn from different distributions under H_a (i.e. for a given $\delta = \frac{\sqrt{varY_1}}{2}$). Where $Var(Y_1)$ is based on the known variances of the respective distributions:

- T: $Var(Y_1) = \frac{df}{df-2}$. For $df = 3$, $df = 5$: $Var(Y_1) = 3$ and $\frac{5}{2}$, respectively.
- Standard normal: $Var(Y_1) = 1$
- Exponential: $Var(Y_1) = \frac{1}{\lambda^2}$. For rate $\lambda = 1$: $Var(Y_1) = 1$
- Uniform: $Var(Y_1) = \frac{(b-a)^2}{12}$. For distribution limits $a = 0$, $b = 1$: $Var(Y_1) = \frac{1}{12}$
- Laplace: $Var(Y_1) = 2b^2$. For scale parameter $b = 1$: $Var(Y_1) = 2$
- Logistic: $Var(Y_1) = s^2\pi^2\frac{2}{6}$. For scale parameter $s = 1$: $Var(Y_1) = \pi^2\frac{2}{6}$

A graphical evaluation of the power performance for the different tests:

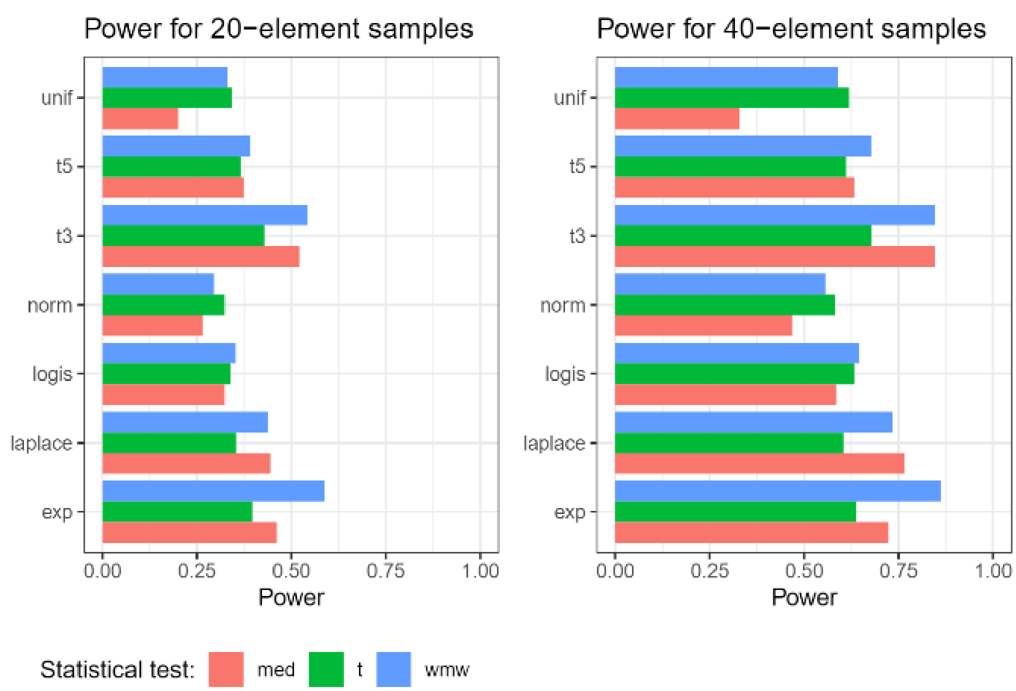

Figure 1: Relative power for different distributions and sample sizes n = 20 and n = 40.

2.2 Type-I error

Next, the simulations were repeated under H_0 (i.e. with $\delta = 0$) to compare the type-I error rate of the median test with the permutation t-test and the Wilcoxon-Mann-Whitney test.

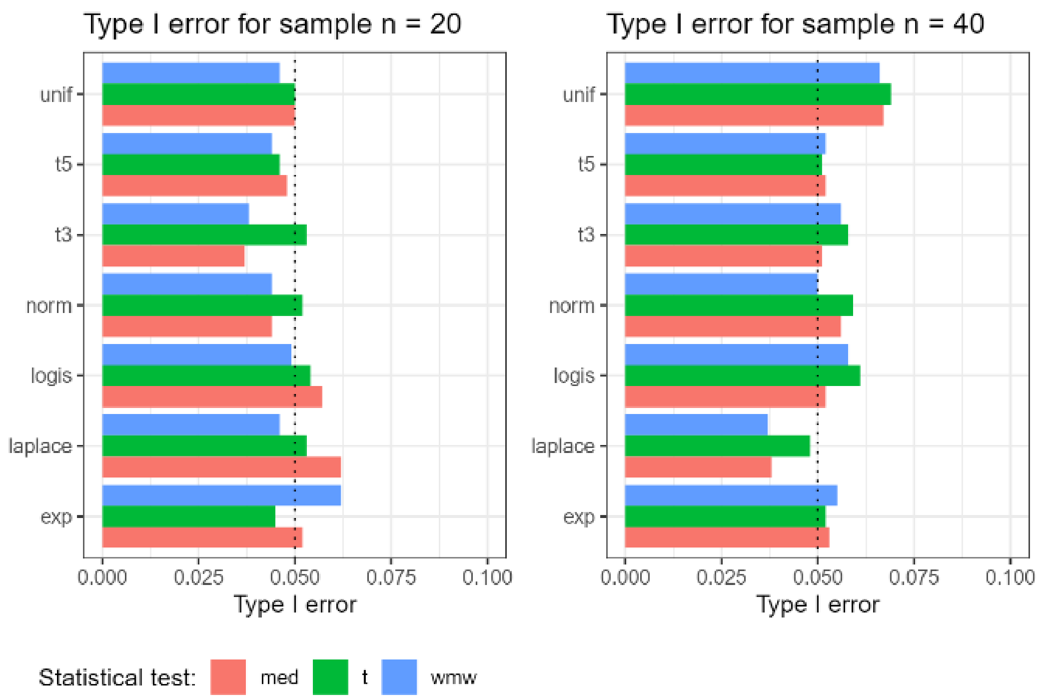

Figure 2: Type I error for different distributions and sample sizes n = 20 and n = 40. The dotted line represents the significance level alpha

3 Conclusion and recommendations

After a non-parametric analysis of the performance of the median test when comparing two independent small samples, we observed that this test is equivalent to the t-test and Wilcoxon-Mann-Whitney test on the basis of power and type-I error. Based on the results of our simulations, we conclude that the median test is more robust for outliers in comparison with the t-test but but more sensitive for outliers in comparison with the Wilcoxon-Mann-Whitney test. Therefore, we recommend to use the median test if it is beneficial to use a hypothesis expressed in terms of a median. In any other case, we advise the Wilcoxon-Mann-Whitney test due to the fact that its performance is slightly better.

4 Addendum

4.1 Full code

Additionally to the function `median.test`, we use a function `calc.rejection` to evaluate the proportion of tests where H_0 is rejected. If $\delta = 0$ (then H_0 is true), this represents the type I error proportion.

```
calc.rejection <- function(N, dist, dist_arg, shift, p.val = 0.05){
  # perform N tests on randomly drawn samples Y1 and Y2
  p.med <- p.wmw <- p.t <- c()
  for(j in 1:N) {
    Y1 <- do.call(what = dist,
                  args = dist_arg)
    Y2 <- do.call(what = dist,
                  args = dist_arg) + shift
    df <- data.frame(rep(c('A', 'B'), each = n), c(Y1, Y2))
    colnames(df) <- c("group", "Y")
    p.med[j] <- median.test(x = Y1, y = Y2)
    p.wmw[j] <- wilcox.test(Y1, Y2, exact = TRUE)$p.value
    p.t[j] <- pvalue(oneway_test(Y~group, data = df,
                                 distribution =
                                   approximate(nresample = 10000)))
  }
  # calculate the proportion of H0 rejections
  return(c( mean(p.med < p.val),
            mean(p.wmw < p.val),
            mean(p.t < p.val)
  )
  )
}

calc.df_power <- function(N, n, delta, p.val = 0.05){
  distributions <- c('rt',
                     'rt',
                     'rexp',
                     'rlogis',
                     'rnorm',
                     'runif',
                     'rlaplace')
  dist_names <- c('t3', 't5', 'exp', 'logis', 'norm', 'unif', 'laplace')
  dist_var <- c(3, 5/3, 1, pi^2/6, 1, 1/12, 2)
  dist_args <- list(list(n, 3),
                    list(n, 5),
                    list(n),
                    list(n),
                    list(n),
                    list(n),
                    list(n))
```

```r
  for(i in 1:length(distributions)) {
    cat(paste('power calculation for distribution : ',
              distributions[i], '\n'))
    res <- calc.rejection(N,
                          distributions[i],
                          dist_arg = dist_args[[i]],
                          delta * (dist_var[i])^(1/2) / 2,
                          p.val = 0.05)
    power.med[i] <- res[1]
    power.wmw[i] <- res[2]
    power.t[i] <- res[3]
  }
  df_power <- data.frame(Distribution = dist_names,
                         med = rep(x = 0.0, length(distributions)),
                         wmw = rep(x = 0.0, length(distributions)),
                         t = rep(x = 0.0, length(distributions)))

  df_power$med <- power.med
  df_power$wmw <- power.wmw
  df_power$t <- power.t
  df_power <- melt(df_power, id.vars = 'Distribution')
  return(df_power)
}
```

Using the above defined formulas, the simulation can be performed with the following code:

```r
# Simulations
N <- 1000
power.med <- power.wmw <- power.t <- c()

for(n in c(20, 40)){
# For simulations under H0 (delta = 0) and Ha (delta = 1)
  for(delta in 0:1) {
    if(delta) {
      title <- paste0('Power_', n, '_', N) }
    else{
      title <- paste0('TypeI_error_', n, '_', N) }

    df_power <- calc.df_power(N, n, delta)

    write.csv(df_power, paste(title, '.csv', sep=''))
  }
}
```

4.2 Influence of the number of simulations

In this report, the number of random permutations to construct the null-distribution was set at 10000 and, in order to keep the computational time somewhat manageable, the number of Monte Carlo simulations was set at 1000. This is - however - an arbitrary choice. In order to have some idea on the effect of the number of Monte Carlo simulations on the stability of the relative statistical power, we ran simulations for a different number of tests under H_a. The samples (n = 40) were drawn from a standard normal distribution. The result (see below) indicate that - for the given distribution, amount of permutations and sample size - the result of the Monte Carlo simulation is relatively stable at 500 simulations or more, suggesting that the increased computational demand that is associated with an additional increase in the number of Monte Carlo simulations does not substantially increase the robustness of the result.

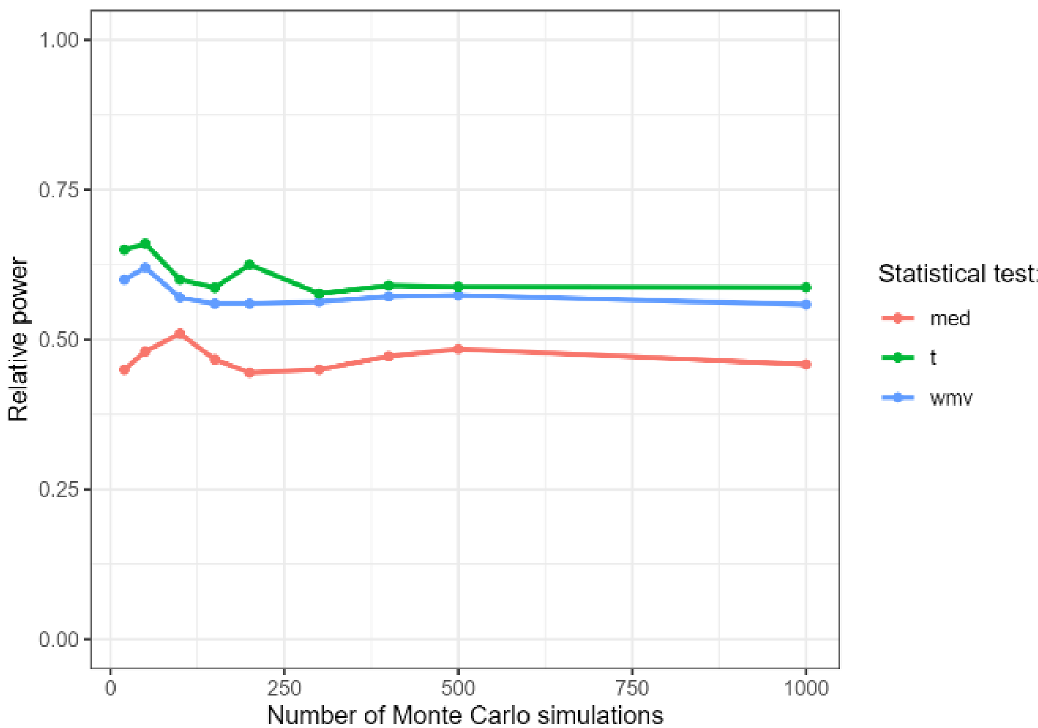

Figure 3: The relative power for an increasing number of Monte Carlo simulations

4.3 Effort distribution

Name	effort
Alexander Jan	33%
Dewilde Brecht	33%
Leloup Arthur	33%