Principles of 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 $H_A : median_x \neq median_y$. If the total amount of combinations exceeds 10000, the null-distribution is generated from 10000 random samples. The code is listed below, the included R comments can guide the reader through the code.

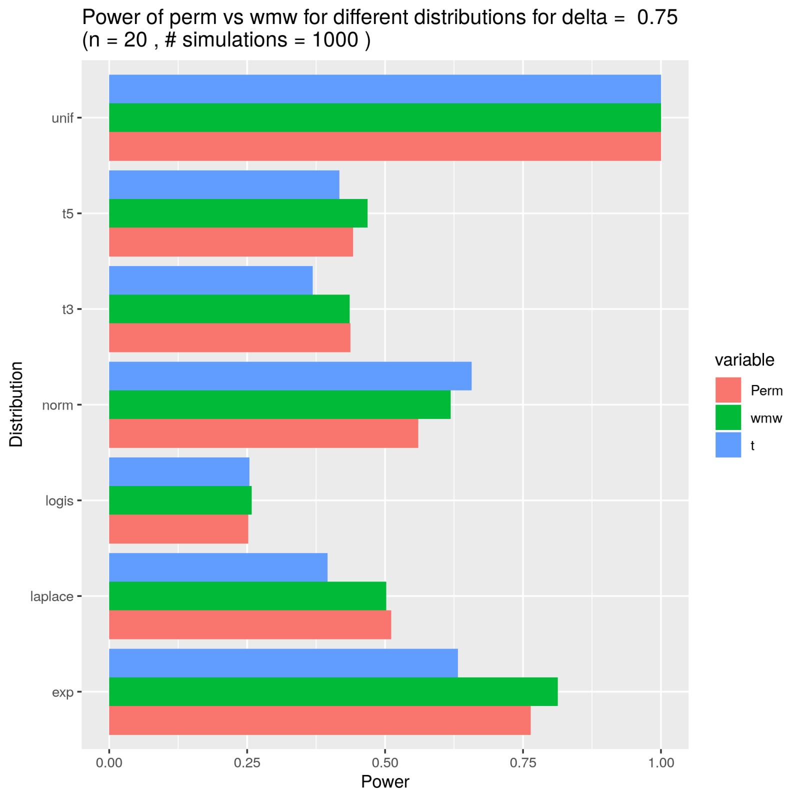
```r
# FUNCTION: calc.median.diff: --------------
# Function: calculate the difference in median between two groups.
#
# param: ind = the indices of the first group
# param: vec = the complete list with data from group 1 and group 2
#
calc.median.diff <- function(ind, vec){
  # make both groups
  group1 <- vec[ind]
  group2 <- vec[!(1:length(vec)) %in% ind]
  # calculate the difference in medians
  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 15000), 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 <- 10000
  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_x+len_y, len_x) < N){
    #cat('A limited number of combinations: full permutation test')
    median.diff <- combn(len_vec,
                         len_x,
                         function(ind){
                           return(calc.median.diff(ind, vec))
                         })
  } else {
    #cat('Too many combinations - A sample test will be performed.\n')
    median.diff <- replicate(N,
                             calc.median.diff(
                               sample(c(1:len_vec),len_x),
                               vec)
    )
  }
  if(FALSE){
    hist(abs(median.diff), main = paste('p = ',
                                        mean(abs(realization) <= abs(median.diff))))
    abline(v = abs(realization), col = "red",
           lty = 1, lwd = 2)
  }
  # The distribution will be symmetrical. The p-value can be calculated by taking
  # the absolute value.
  return(mean(abs(realization) <= abs(median.diff)))
}
```

2 Comparison to other tests

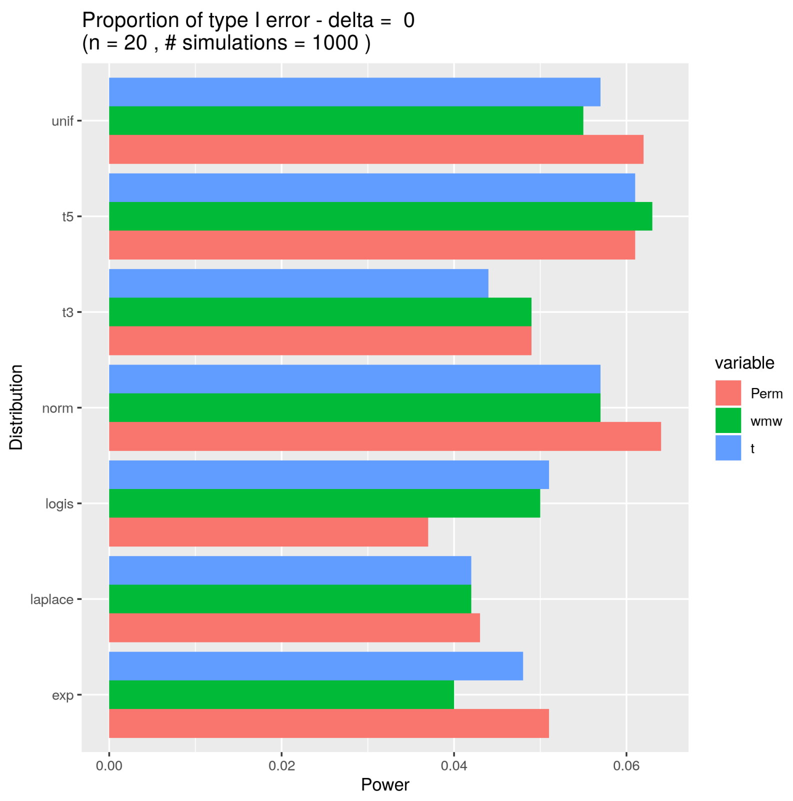
2.1 Power

To evaluate the power of the median test as compared to other statistical tests, the proportions of true positive outcomes of the median test, the permutation t-test and the Wilcoxon–Mann–Whitney test were acquired through Monte Carlo simulation. Samples (n = 20 and n = 40) were randomly drawn from different distributions under H_a (i.e. for a given delta $\neq 0$).



2.2 Type-I error rate

Next, the simulations were repeated under H_0 (i.e. with delta set equal to zero) to compare the type-I error rate of the median test with the permutation t-test and the Wilcoxon–Mann–Whitney test.



3 Conclusion

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc rhoncus orci vitae nisl mollis dictum. Donec non nulla augue. Nullam turpis ligula, vehicula at consectetur in, finibus eget urna. Proin eget massa pellentesque, efficitur orci gravida, ullamcorper massa. Aliquam erat volutpat. Morbi lobortis enim eu mauris commodo posuere. Duis mauris sem, convallis non auctor sit amet, aliquet interdum lorem. Morbi id ultricies diam. Sed eu lacus ut massa elementum tempus. Nulla facilisi. Ut ac nulla vel diam ultrices gravida at sed neque. Aliquam bibendum orci eu felis ultricies, a vestibulum mauris pulvinar.

4 Addendum: full code

```r
# Install and load packages
packages <- c('coin','ggplot2', 'reshape2', 'rmutil')
install.packages(packages)
lapply(packages, library, character.only = TRUE)
#
# 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 medians
  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 15000), 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 <- 10000
  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_x+len_y, len_x) < N){
    #cat('A limited number of combinations: full permutation test')
    median.diff <- combn(len_vec,
                         len_x,
                         function(ind){
                           return(calc.median.diff(ind, vec))
```

```r
                            })
  } else {
    #cat('Too many combinations - A sample test will be performed.\n')
    median.diff <- replicate(N,
                                calc.median.diff(
                                  sample(c(1:len_vec),len_x),
                                  vec)
    )
  }
  if(FALSE){
    hist(abs(median.diff), main = paste('p = ',
                                        mean(abs(realization) <= abs(median.diff))))
    abline(v = abs(realization), col = "red",
           lty = 1, lwd = 2)
  }
  # The distribution will be symmetrical. The p-value can be calculated by taking
  # the absolute value.
  return(mean(abs(realization) <= abs(median.diff)))
}

# FUNCTION: calculate.power: --------------
# Function: calculate the proportion of tests that yield a p-value
# smaller than 0.05 for the median.test, the Wilcoxon exact
# test and the permutation t-test after repetitive sampling.
#
# param: delta = effect size
# param: d = random distribution function
# dist_arg = arguments or the distribution function
# N_tests = the amount of repetitions
# p_value = significance level
#
calculate.power <- function(delta, d, dist_arg, N_tests = 1000, p_value = 0.05){
  cat('power calculation for distribution : ')
  cat(d)
  cat('\n')
  p.perm <- p.wmw <- p.t <- c()
  for(i in 1:N_tests){
    Y1 <- do.call(what = d,
                  args = dist_arg)
    Y2 <- do.call(what = d,
                  args = dist_arg) + delta
    df <- data.frame(rep(c('A', 'B'), each = n), c(Y1, Y2))
    colnames(df) <- c("group", "Y")
    p.perm[i] <- median.test(x = Y1, y = Y2)
    p.wmw[i] <- wilcox.test(Y1, Y2, exact = TRUE)$p.value
    p.t[i] <- pvalue(oneway_test(Y~group, data = df,
                                  distribution = approximate(nresample = 10000)))
  }
  if(FALSE){
    hist(p.perm)
    hist(p.wmw)
  }
  return(c(mean(p.perm < p_value),
           mean(p.wmw < p_value),
           mean(p.t < p_value)
  )
  )
}

# Simulations
n <- 20
distributions <- c('rt', 'rt', 'rexp', 'rlogis', 'rnorm', 'runif', 'rlaplace')
dist_names <- c('t3', 't5', 'exp', 'logis', 'norm', 'unif', 'laplace')
dist_args <- list(list(n, 3), list(n, 5), list(n), list(n), list(n), list(n), list(n))

power.perm <- power.wmw <- power.t <- c()

for(delta in c(0, 0.5, 0.75)){
  for(i in 1:7){
    N_tests <- 1000
    d <- distributions[i]
    res <- calculate.power(delta, d, dist_args[[i]], N_tests = N_tests)
    power.perm[i] <- res[1]
    power.wmw[i] <- res[2]
    power.t[i] <- res[3]
  }
  df_power <- data.frame(Distribution = dist_names,
                         Perm = rep(x = 0.0, length(distributions)),
                         wmw = rep(x = 0.0, length(distributions)),
                         t = rep(x = 0.0, length(distributions)))
  df_power$Perm <- power.perm
  df_power$wmw <- power.wmw
  df_power$t <- power.t
  df_power <- melt(df_power, id.vars = 'Distribution')

  write.csv(df_power, paste('delta_', delta*100, '.csv', sep=''))

  if(delta != 0){
    title <- paste('Power of perm vs wmw for different distributions for delta = ', delta,
                   '\n(n =', n, ', # simulations =', N_tests, ')')
  }else{
    title <- paste('Proportion of type I error - delta = ', delta,
                   '\n(n =', n, ', # simulations =', N_tests, ')')
  }

  plot1 <- ggplot(data = df_power, aes(x = Distribution, y = value, fill = variable)) +
    geom_bar(position = 'dodge', stat = 'identity') +
    ggtitle(title) +
    xlab('Distribution') +
    ylab('Power') +
    coord_flip()

  ggsave( paste('delta_', delta*100, '.png', sep = ''),
          plot = plot1,
          device = 'png'
          )
}
```