Principles of Statistical Data Analysis: HW3

Jan Alexander, Brecht Dewilde, Arthur Leloup

2019 - 2020

Contents

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

2 Comparison to other tests **2**
2.1 Power . . . 2
2.2 Type-I error rate . . . 4

3 Conclusion **5**

4 Addendum: full code **7**

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 5000, the null-distribution is generated from 5000 random samples. The code is listed below, the include comments can guide the reader through the code.

```
# 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.
```

```r
#
# 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

The median test was compared, via simmulations, to two other commenly used tests: + the student t-test; + the Wilcoxon-Mann-Whitney test.

The three tests are compared on the basis of the Type I error and on their power. This comparision is made for two sample sizes:

- 20 observations in each sample
- 40 observations in each sample

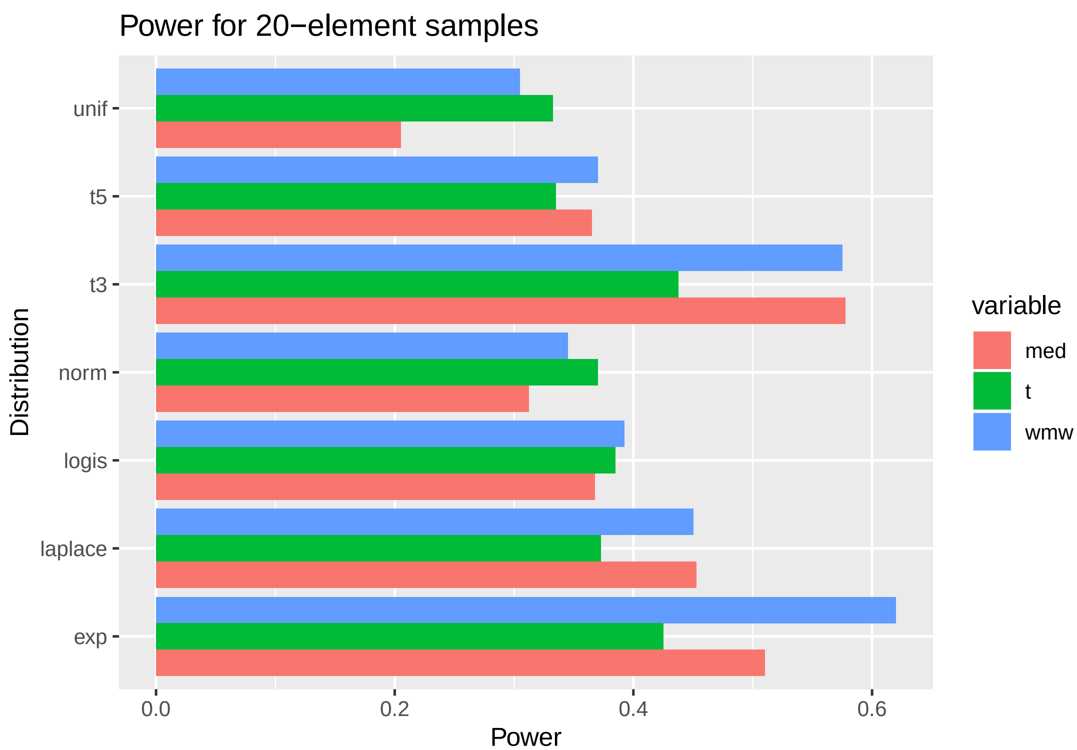
2.1 Power

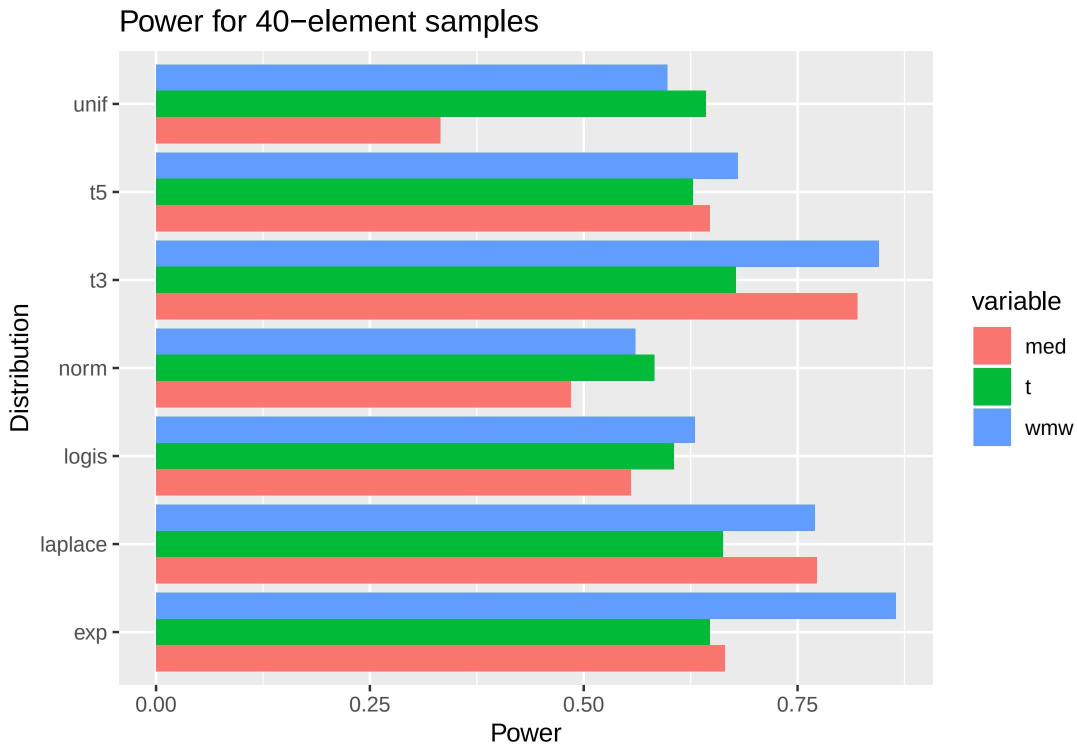
To evaluate the power of the median test as compared to other statistical tests, the proportions of true positive outcomes of the median test, the permutation t-test and the Wilcoxon–Mann–Whitney test were acquired through Monte Carlo simulation (400 simulations). Samples (n =

20 and n = 40) 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:

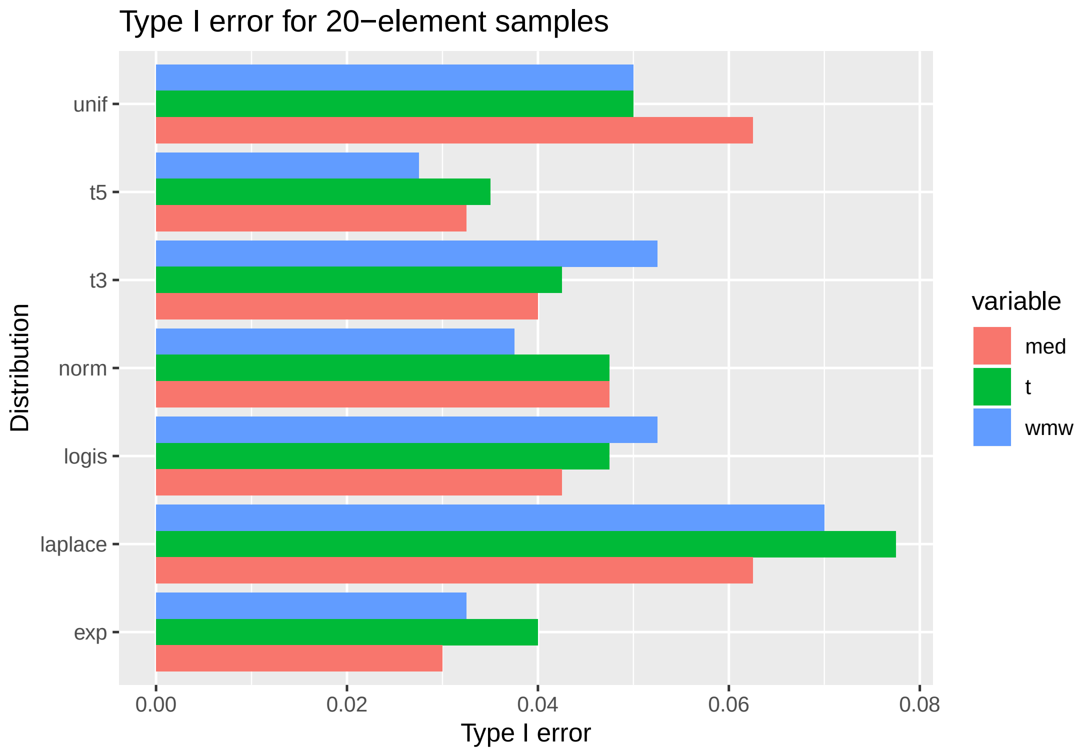
- T-distribution: $var(X) = \frac{\nu}{\nu-2}$. Thus: 3 for $\nu = 3$ and $\frac{5}{2}$ for $\nu = 5$
- Exponential distribution: $var(X) = \frac{1}{\lambda^2}$ or 1, when taking $\lambda = 1$
- Uniform distribution: $var(X) = \frac{(b-a)^2}{12}$, or $\frac{1}{12}$ in this case.
- Laplace distribution: $var(X) = 2b^2$. This is 2 in this case.
- Logistic distribution: $var(X) = s^2\pi^2\frac{2}{6}$. In this case, $s = 1$.





2.2 Type-I error rate

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.



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.

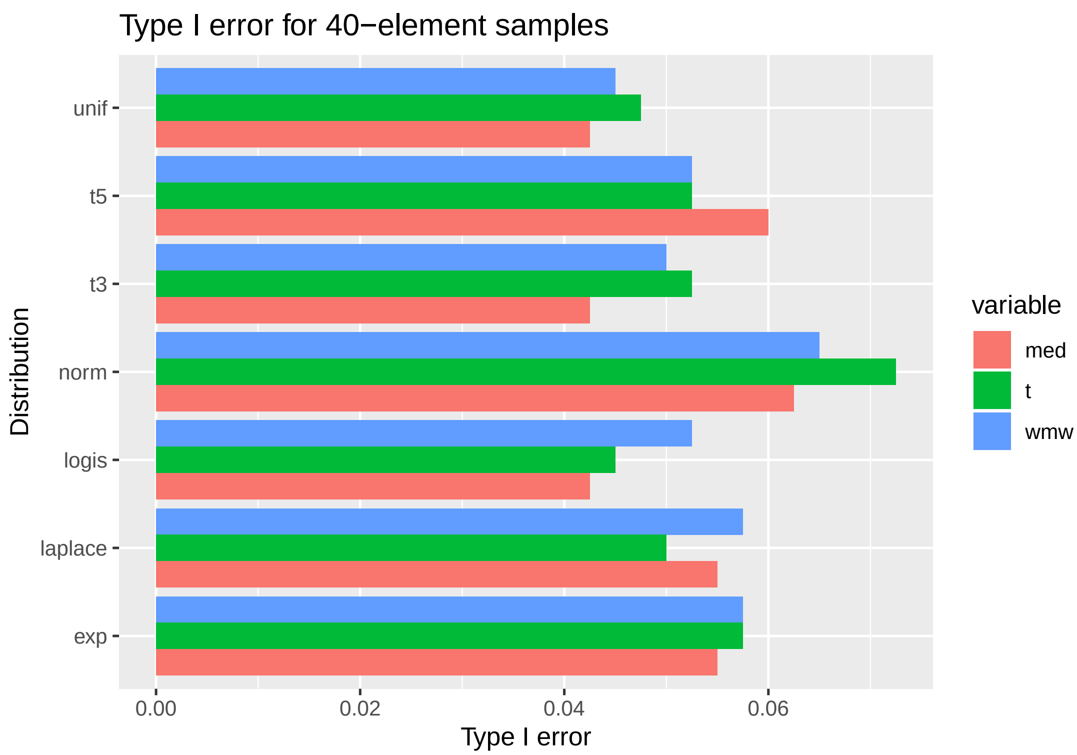


Figure 1: Type I error for 40-element samples

4 Addendum: full code

Additionally to the function `median.test(x,y)`, we use a function `calc.rejection(N, n, delta, p.val = 0.05)` 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, 2*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))
  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
    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:

```
# Simulations
N <- 400    # test simulations
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=''))
  }
}
```

The influence of the number of simulations is evaluated with the following code. This was performed because the calculations took (to our surprise) very long to run. To keep the computational time somewhat manageable, the number of Monte Carlo simulations was reduced. The idea of the following graphs is to estimate difference extra simulations make in stabilizing the result. We can observe that: + The WMW tests seems to deliver a stable result from 300 Monte Carlo simulations on + Both the median test and the T-test seem to be barely stabilized at 400 or 500 simulations.

```
# Simulations
N <- c(20, 50, 100, 150, 200, 300, 400, 500)    # test simulations
error.med <- error.wmw <- error.t <- c()

delta <- 0
n <- 40
i <- 1

for(N_ in N){
  res <- calc.rejection(N_, 'rnorm', list(n), shift = 1/2, p.val = 0.05)
  error.med[i] <- res[1]
  error.wmw[i] <- res[2]
  error.t[i] <- res[3]
  i <- i+1
}
write.csv(error.med, "med.csv")
write.csv(error.wmw, "wmw.csv")
write.csv(error.t, 'Ttest.csv')
```

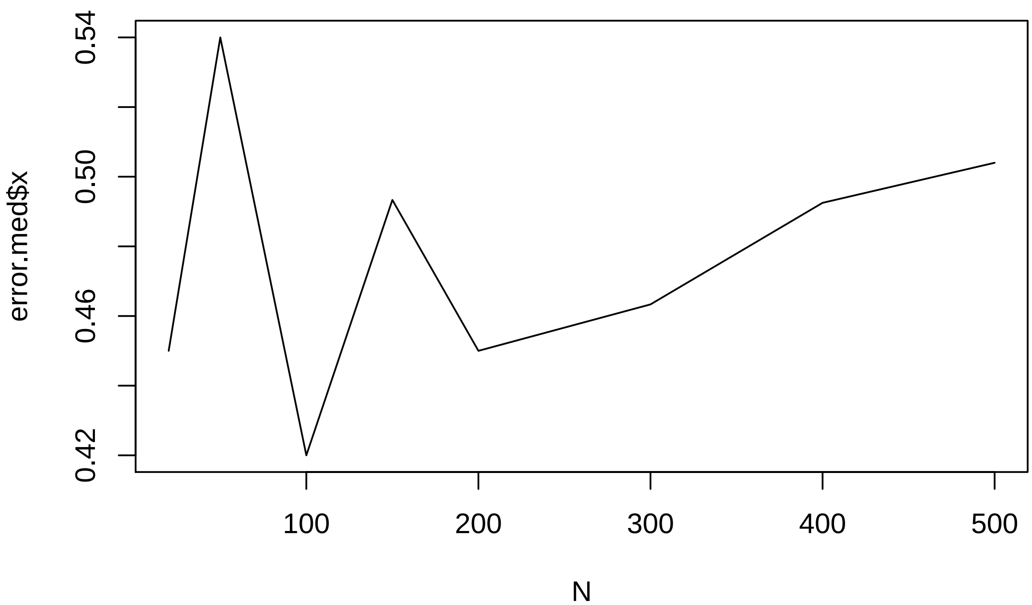


Figure 2: Evolution of power with N, for the median test

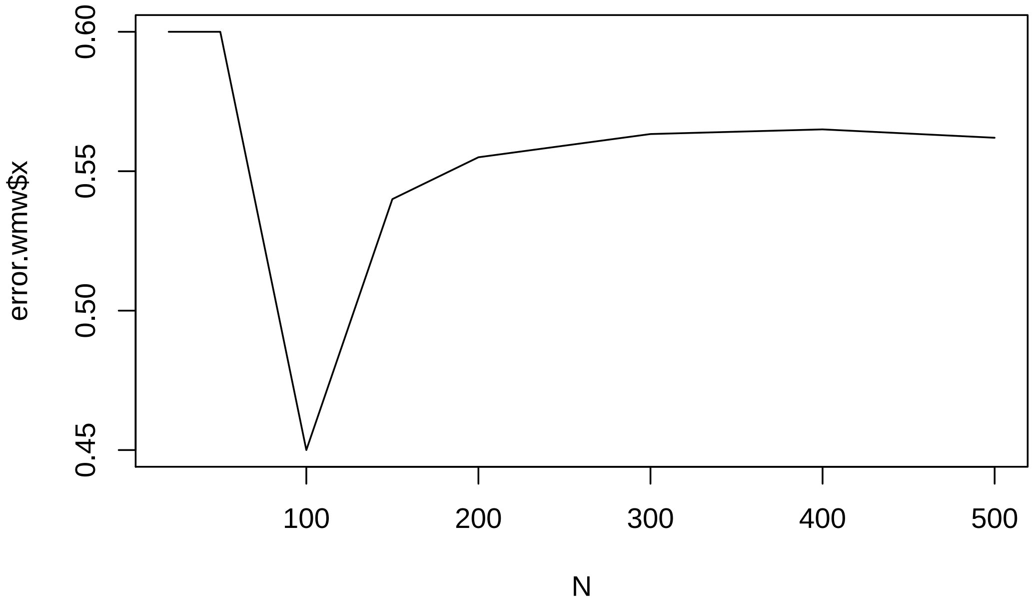


Figure 3: Evolution of power with N, for the WMW test

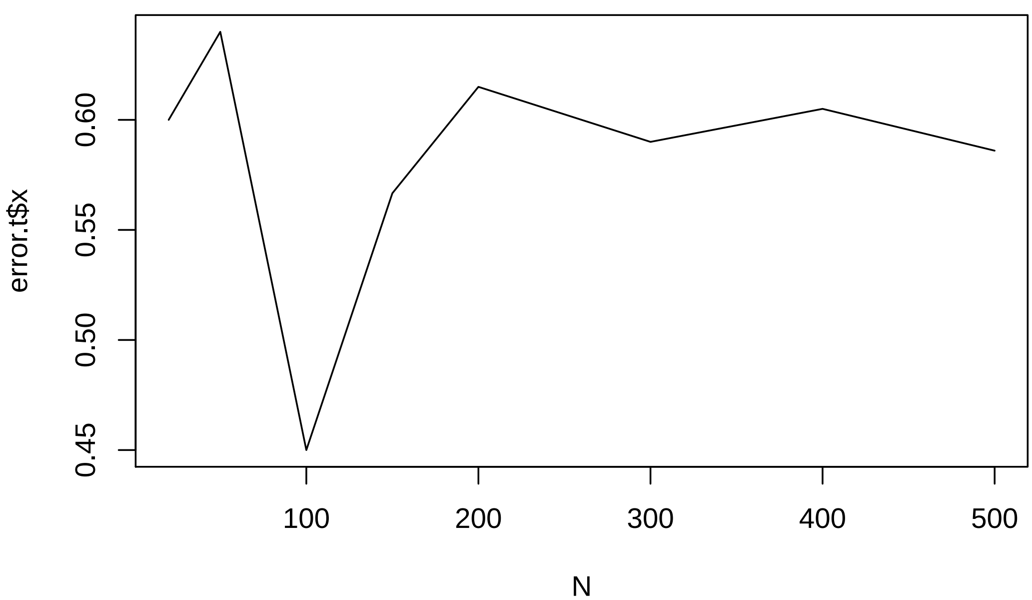


Figure 4: Evolution of power with N, for the T-test