Power analysis of closed testing methods with Simes, Wilcoxon-Mann-Whitney and LMPI T3 as local tests considering Lehmann's alternative of order k

2023-12-01

Libraries and functions

```
library(tidyverse)
library(doSNOW)
library(nout)

gen.data <- function(m,n) {
  Z <- rnorm((m+n))
  return(Z)
}

gen.scores_Lehmann <- function(m, n, n1, k){
  if(n1==0){
    S_Z = gen.data(m,n)
    S_cal = S_Z[1:m]
    S_te = S_Z[(m+1):length(S_Z)]
  }

  if(n1==n){
    augmented.S_Z = gen.data(m,n*k)
    S_cal = augmented.S_Z[1:m]
    augmented.S_te = augmented.S_Z[(m+1):length(augmented.S_Z)]
    S_te = sapply(1:n1, FUN=function(i) max(augmented.S_te[(1+k*(i-1)):(i*k)]))
  }

  if(0<n1&n1<n){
    augmented.S_Z = gen.data(m=m,n=n-n1+n1*k)
    S_cal = augmented.S_Z[1:m]
    augmented.S_te = augmented.S_Z[(m+1):length(augmented.S_Z)]
    inlier.S_te = augmented.S_te[1:(n-n1)]
    outlier.augmented.S_te = augmented.S_te[(n-n1+1):length(augmented.S_te)]
    outlier.S_te = sapply(1:n1, FUN=function(i) max(outlier.augmented.S_te[(1+k*(i-1)):(i*k)]))
    S_te = c(inlier.S_te, outlier.S_te)
  }

  return(list("S_cal" = S_cal,
              "S_te" = S_te,
              "k" = k,
              "n1" = n1))
}
```

```r
compute_lb.d = function(B, m, n, n1, k, alpha){

  foreach(b = 1:B, .combine=cbind) %dopar% {

    scores = gen.scores_Lehmann(m, n, n1, k)
    S_cal = scores$S_cal
    S_te = scores$S_te

    d_WMW = nout::d_MannWhitney(S_Y = S_te, S_X = S_cal, alpha=alpha)
    d_T3 = nout::d_MannWhitneyk3(S_Y = S_te, S_X = S_cal, alpha=alpha)
    d_Sim = nout::d_Simes(S_X = S_cal, S_Y = S_te, alpha = alpha)
    # StoSimes = nout::d_StoreySimes(S_X = S_cal, S_Y = S_te, alpha = alpha)
    # d_StoSimes = StoSimes$d
    # pi.not = StoSimes$pi.not
    d_BH = nout::d_benjhoch(S_X = S_cal, S_Y = S_te, alpha = alpha)
    # d_StoBH = nout::d_StoreyBH(S_X = S_cal, S_Y = S_te, alpha = alpha)

    return(list("m" = m,
                "n" = n,
                "k" = k,
                "n1" = n1,
                "alpha" = alpha,
                # "S_cal" = S_cal,
                # "S_te" = S_te,
                "d_BH" = d_BH,
                #"d_StoBH" = d_StoBH,
                "d_Sim" = d_Sim,
                #"d_StoSimes" = d_StoSimes,
                "d_WMW" = d_WMW,
                "d_T3" = d_T3
                #"pi.not" = pi.not
                ))
  }
}



compact_results = function(res, ks, n1.index, n){

  mean.lb.d_n1_k =  matrix(nrow = length(ks), ncol = 4)

  rnames = vector()
  for(i in 1:length(ks)){
    rnames[i] = paste0("k=", ks[i])
  }
  # cnames.lb.d = c("mean.lb.d_BH", "mean.lb.d_StoBH", "mean.lb.d_Sim",
  #                 "mean.lb.d_StoSim", "mean.lb.d_WMW", "mean.lb.d_T3")
  cnames.lb.d = c("mean.lb.d_BH", "mean.lb.d_Sim",
                  "mean.lb.d_WMW", "mean.lb.d_T3")
  rownames(mean.lb.d_n1_k) = rnames
  colnames(mean.lb.d_n1_k) = cnames.lb.d
```

```
  for(i in 1:length(ks)){
    mean.lb.d_n1_k[i,"mean.lb.d_BH"] = mean(unlist(res[[i]][[n1.index]]["d_BH",]))
    #mean.lb.d_n1_k[i,"mean.lb.d_StoBH"] = mean(unlist(res[[i]][[n1.index]]["d_StoBH",]))
    mean.lb.d_n1_k[i,"mean.lb.d_Sim"] = mean(unlist(res[[i]][[n1.index]]["d_Sim",]))
    #mean.lb.d_n1_k[i,"mean.lb.d_StoSim"] = mean(unlist(res[[i]][[n1.index]]["d_StoSimes",]))
    mean.lb.d_n1_k[i,"mean.lb.d_WMW"] = mean(unlist(res[[i]][[n1.index]]["d_WMW",]))
    mean.lb.d_n1_k[i,"mean.lb.d_T3"] = mean(unlist(res[[i]][[n1.index]]["d_T3",]))
  }


  mean.power_n1_k =  matrix(nrow = length(ks), ncol = 4)

  cnames.power = c("mean.power_BH", "mean.power_Sim",
                   "mean.power_WMW", "mean.power_T3")
  rownames(mean.power_n1_k) = rnames
  colnames(mean.power_n1_k) = cnames.power

  for(i in 1:length(ks)){
    mean.power_n1_k[i,"mean.power_BH"] = mean(unlist(res[[i]][[n1.index]]["d_BH",])>0)
    #mean.power_n1_k[i,"mean.power_StoBH"] = mean(unlist(res[[i]][[n1.index]]["d_StoBH",])>0)
    mean.power_n1_k[i,"mean.power_Sim"] = mean(unlist(res[[i]][[n1.index]]["d_Sim",])>0)
    #mean.power_n1_k[i,"mean.power_StoSim"] = mean(unlist(res[[i]][[n1.index]]["d_StoSimes",])>0)
    mean.power_n1_k[i,"mean.power_WMW"] = mean(unlist(res[[i]][[n1.index]]["d_WMW",])>0)
    mean.power_n1_k[i,"mean.power_T3"] = mean(unlist(res[[i]][[n1.index]]["d_T3",])>0)
  }

  results = list("mean.power_n1_k" = mean.power_n1_k,
                 "mean.lb.d_n1_k" = mean.lb.d_n1_k)

  return(results)
}
```

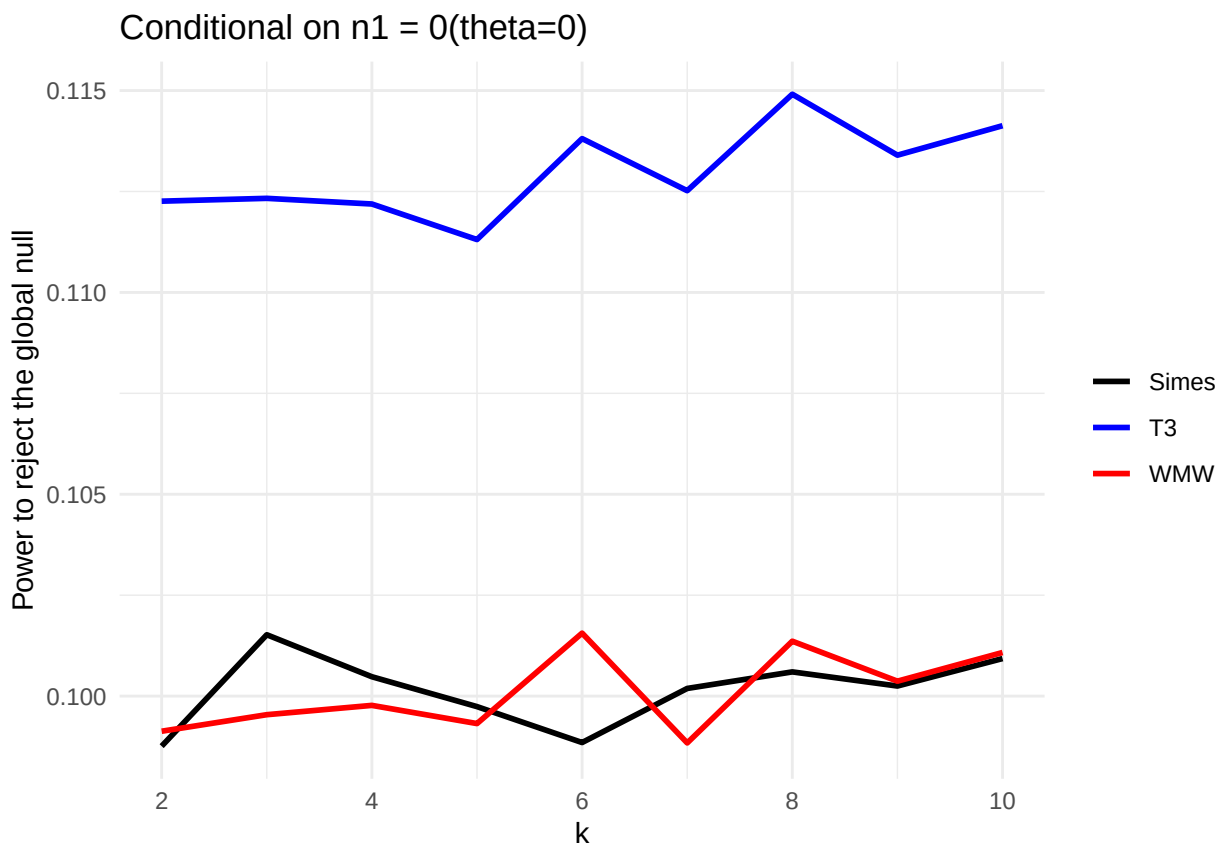
```
set.seed(321)

B = 10^5

m = 3999
n = 400
alpha = n/(m+1)

thetas = c(0, 0.005, 0.01, 0.05, 0.1)
n1s = floor(n*thetas)

# Order of the Lehmann's alternative
ks = 2:10

cluster <- makeCluster(parallel::detectCores()-1)
registerDoSNOW(cluster)
clusterEvalQ(cluster, {list(library(isotree), library(nout))})
```

```
## [[1]]
## [[1]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
```

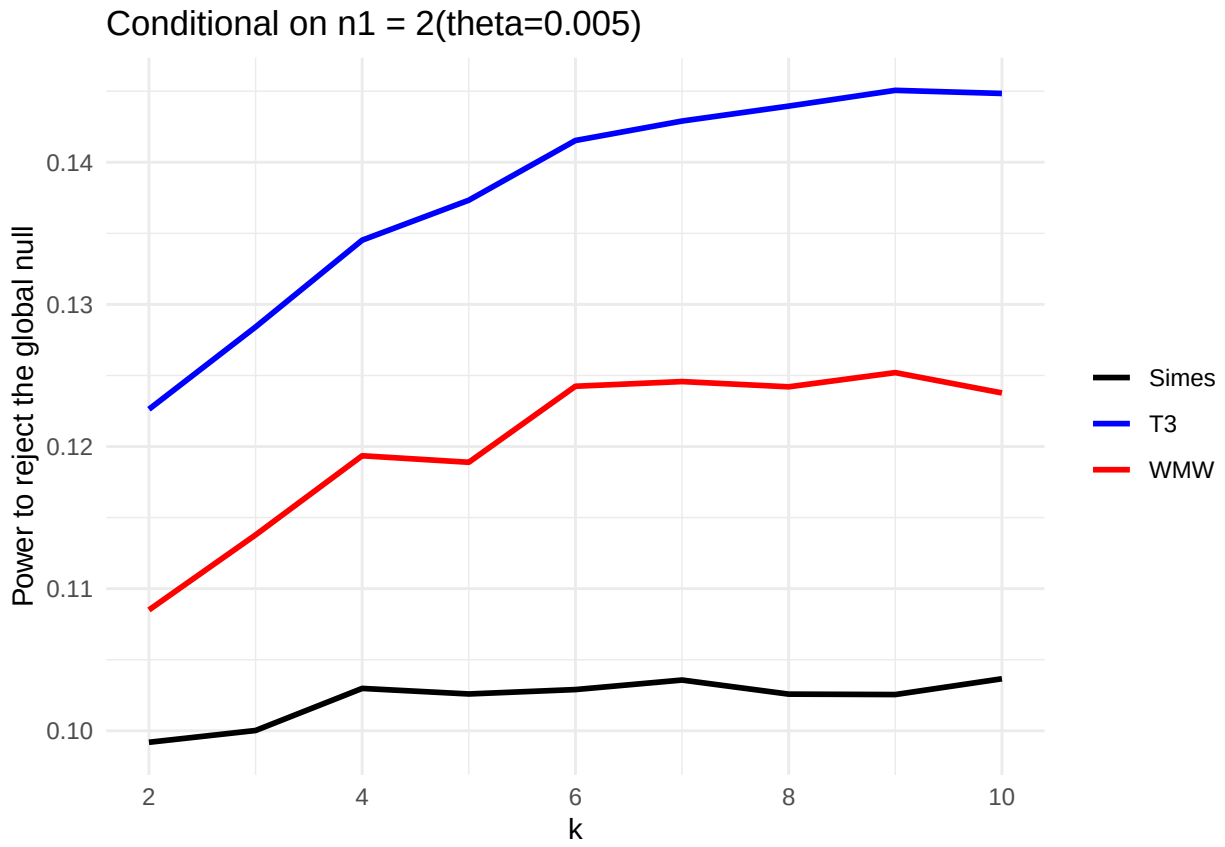
```
##
## [[1]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
##
## [[2]]
## [[2]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[2]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
##
## [[3]]
## [[3]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[3]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
```

```
clusterExport(cluster, list("n", "m", "ks", "n1s", "alpha", "gen.data", "gen.scores_Lehmann"))

res <- lapply(1:length(ks), function(i){
  lapply( 1:length(n1s), function(j) compute_lb.d(B=B, m=m, n=n,
                                                  n1=n1s[j], k=ks[i], alpha=alpha))
  }
)

stopCluster(cluster)

results = lapply(1:length(n1s),
                 function(j) compact_results(res=res, ks=ks, n=n, n1.index=j) )

pp = list()
for(i in 1:length(n1s)){
  pow_BH = results[[i]]$mean.power_n1_k[,"mean.power_BH"]
  pow_Sim = results[[i]]$mean.power_n1_k[,"mean.power_Sim"]
  pow_WMW = results[[i]]$mean.power_n1_k[,"mean.power_WMW"]
  pow_T3 = results[[i]]$mean.power_n1_k[,"mean.power_T3"]

  dfpower <- data.frame(
    x = ks,
    Simes = pow_BH,
    WMW = pow_WMW,
    T3 = pow_T3
  )
  df_long_power <- tidyr::pivot_longer(dfpower, cols = -x, names_to = "group", values_to = "y")

  pp[[i]] = ggplot(df_long_power, aes(x = x, y = y, color = group)) +
```

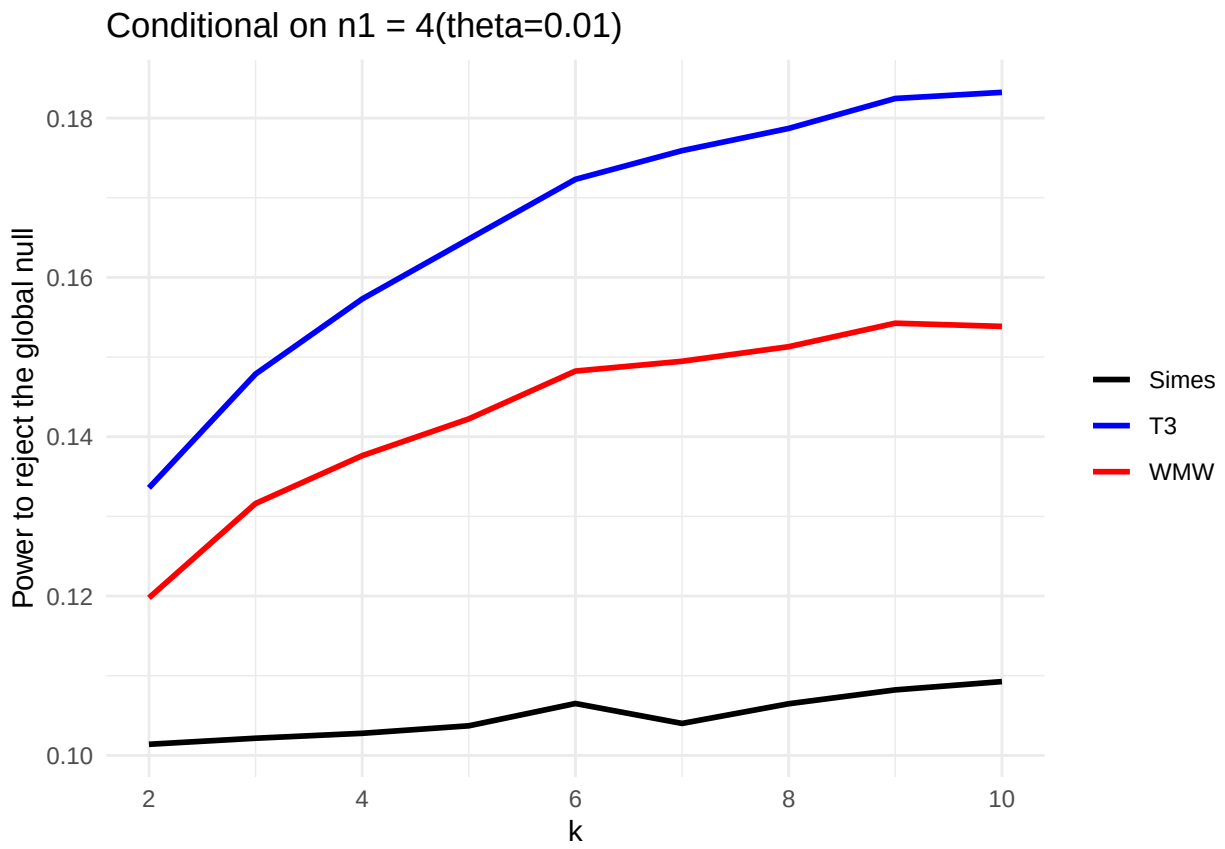
```
                    geom_line(size=1) +
                    scale_color_manual(values = c("black","blue","red")) +
                    ggtitle(paste0("Conditional on n1 = ", n1s[i], "(theta=", thetas[i], ")")) +
                    labs(x = "k", y = "Power to reject the global null") +
                    theme_minimal() +
                    theme(legend.title = element_blank())
  print(pp[[i]])
  print(cbind(pow_BH, pow_WMW, pow_T3))
}
```



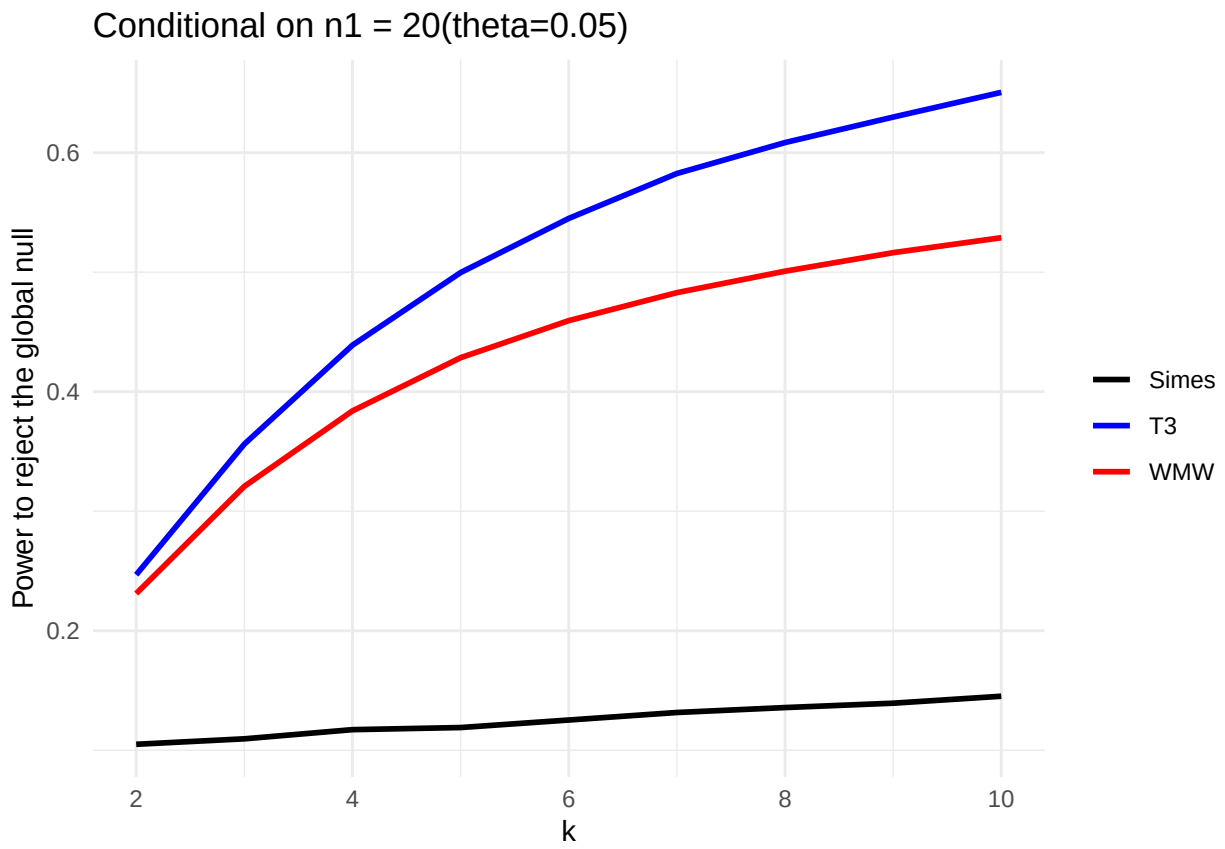

```
##        pow_BH pow_WMW  pow_T3
## k=2  0.09876 0.09913 0.11226
## k=3  0.10152 0.09954 0.11233
## k=4  0.10048 0.09977 0.11219
## k=5  0.09974 0.09932 0.11131
## k=6  0.09885 0.10156 0.11381
## k=7  0.10019 0.09884 0.11252
## k=8  0.10060 0.10136 0.11491
## k=9  0.10025 0.10037 0.11340
## k=10 0.10093 0.10108 0.11413
```



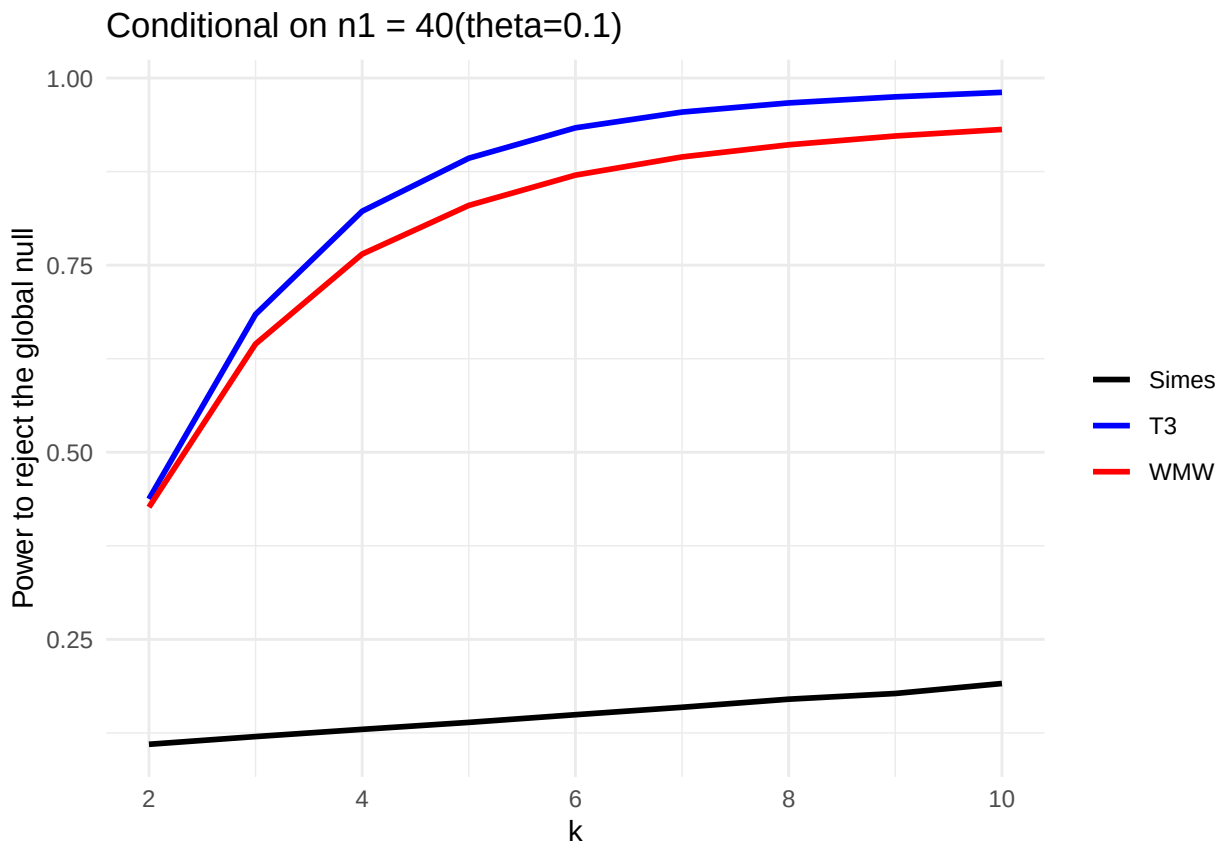

```
##        pow_BH pow_WMW  pow_T3
## k=2  0.09919 0.10850 0.12262
## k=3  0.10002 0.11378 0.12841
## k=4  0.10298 0.11935 0.13452
## k=5  0.10259 0.11889 0.13733
## k=6  0.10290 0.12424 0.14153
## k=7  0.10357 0.12457 0.14290
## k=8  0.10258 0.12420 0.14395
## k=9  0.10255 0.12520 0.14506
## k=10 0.10366 0.12377 0.14484
```




```
##        pow_BH pow_WMW  pow_T3
## k=2  0.10139 0.11977 0.13358
## k=3  0.10215 0.13162 0.14787
## k=4  0.10277 0.13762 0.15729
## k=5  0.10371 0.14224 0.16481
## k=6  0.10651 0.14824 0.17231
## k=7  0.10401 0.14947 0.17591
## k=8  0.10648 0.15130 0.17870
## k=9  0.10822 0.15425 0.18246
## k=10 0.10926 0.15385 0.18323
```




```
##       pow_BH pow_WMW  pow_T3
## k=2  0.10498 0.23110 0.24684
## k=3  0.10957 0.32079 0.35616
## k=4  0.11725 0.38394 0.43893
## k=5  0.11901 0.42848 0.49970
## k=6  0.12529 0.45950 0.54501
## k=7  0.13158 0.48290 0.58257
## k=8  0.13572 0.50084 0.60847
## k=9  0.13937 0.51634 0.62981
## k=10 0.14520 0.52887 0.65050
```




```
##       pow_BH pow_WMW  pow_T3
## k=2  0.10979 0.42659 0.43755
## k=3  0.12010 0.64471 0.68424
## k=4  0.12975 0.76482 0.82206
## k=5  0.13905 0.82984 0.89290
## k=6  0.14931 0.87026 0.93350
## k=7  0.15926 0.89460 0.95456
## k=8  0.17007 0.91081 0.96675
## k=9  0.17769 0.92257 0.97488
## k=10 0.19101 0.93133 0.98092
```