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

2023-11-26

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))
```

```
}



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_T3 = nout::d_MannWhitneyk3(S_Y = S_te, S_X = S_cal, alpha=alpha)
    d_T3

    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 = 6)

  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")
  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 = 6)

  cnames.power = c("mean.power_BH", "mean.power_StoBH", "mean.power_Sim",
                   "mean.power_StoSim", "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 = 10^4

m = 1999
n = 200
alpha = n/(m+1)

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

# Order of the Lehmann's alternative
ks = 2:5

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

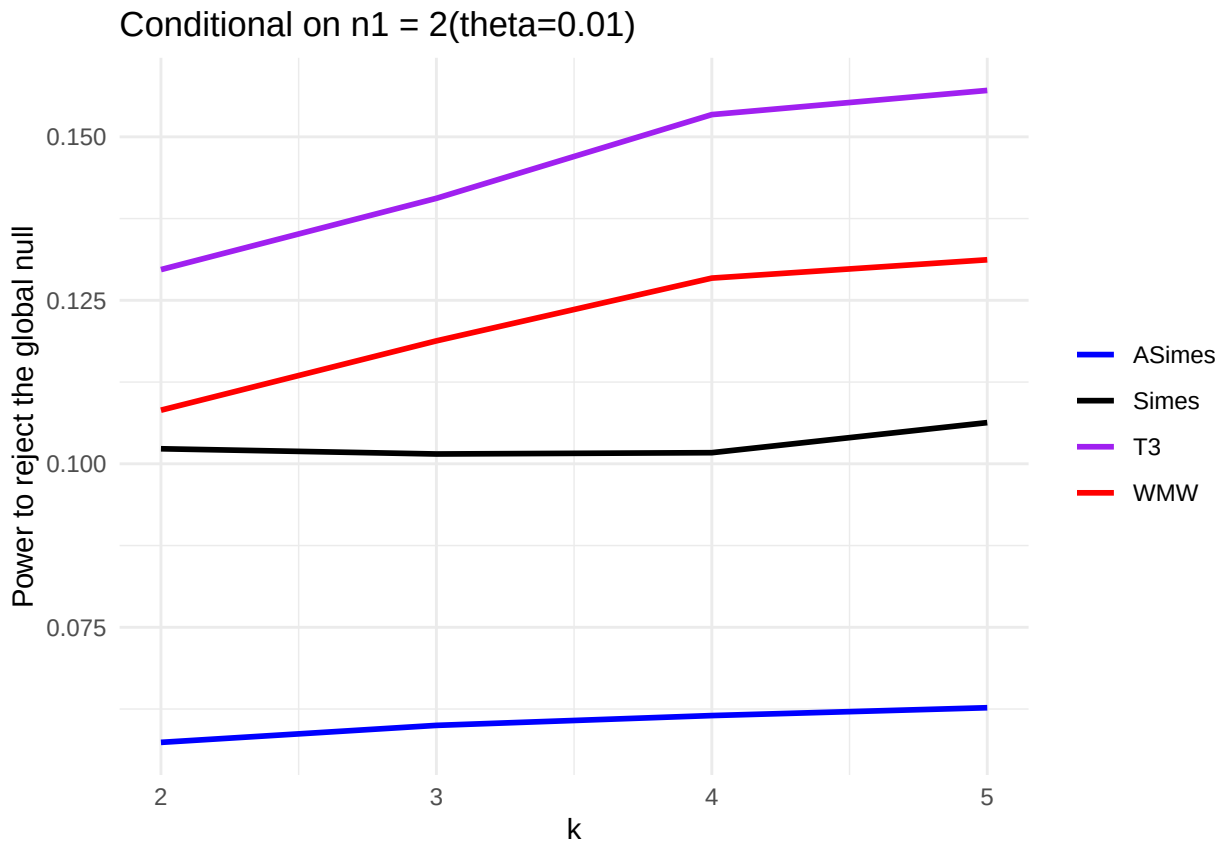
```
## [[1]]
```

```
## [[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"
##
##
## [[4]]
## [[4]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[4]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
##
## [[5]]
## [[5]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[5]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
##
## [[6]]
## [[6]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
```

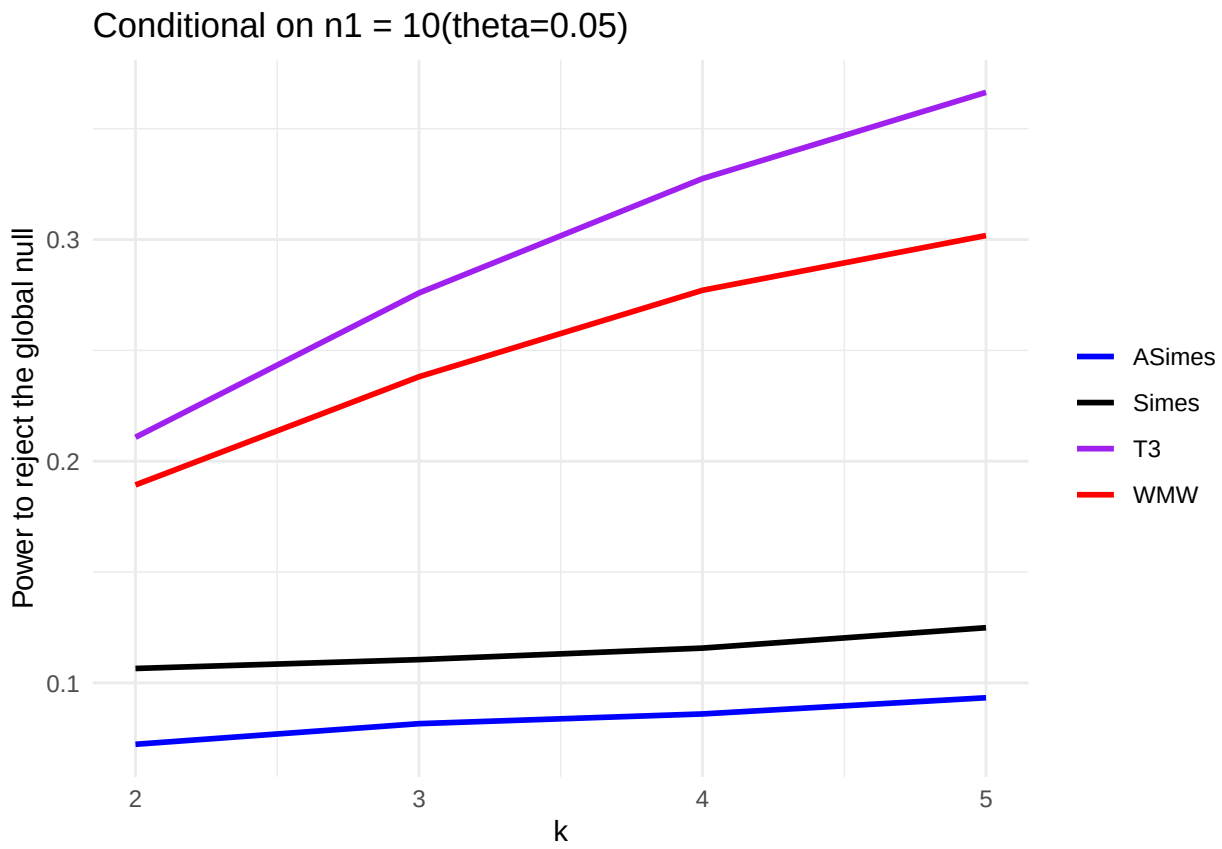
```
## [[6]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
##
## [[7]]
## [[7]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[7]][[2]]
##  [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]]$mean.power_n1_k[,"mean.power_BH"]
  pow_StoBH = results[[i]]$mean.power_n1_k[,"mean.power_StoBH"]
  pow_Sim = results[[i]]$mean.power_n1_k[,"mean.power_Sim"]
  pow_StoSim = results[[i]]$mean.power_n1_k[,"mean.power_StoSim"]
  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,
    ASimes = pow_StoBH,
    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("blue","black","purple","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_StoBH, pow_WMW, pow_T3))
```

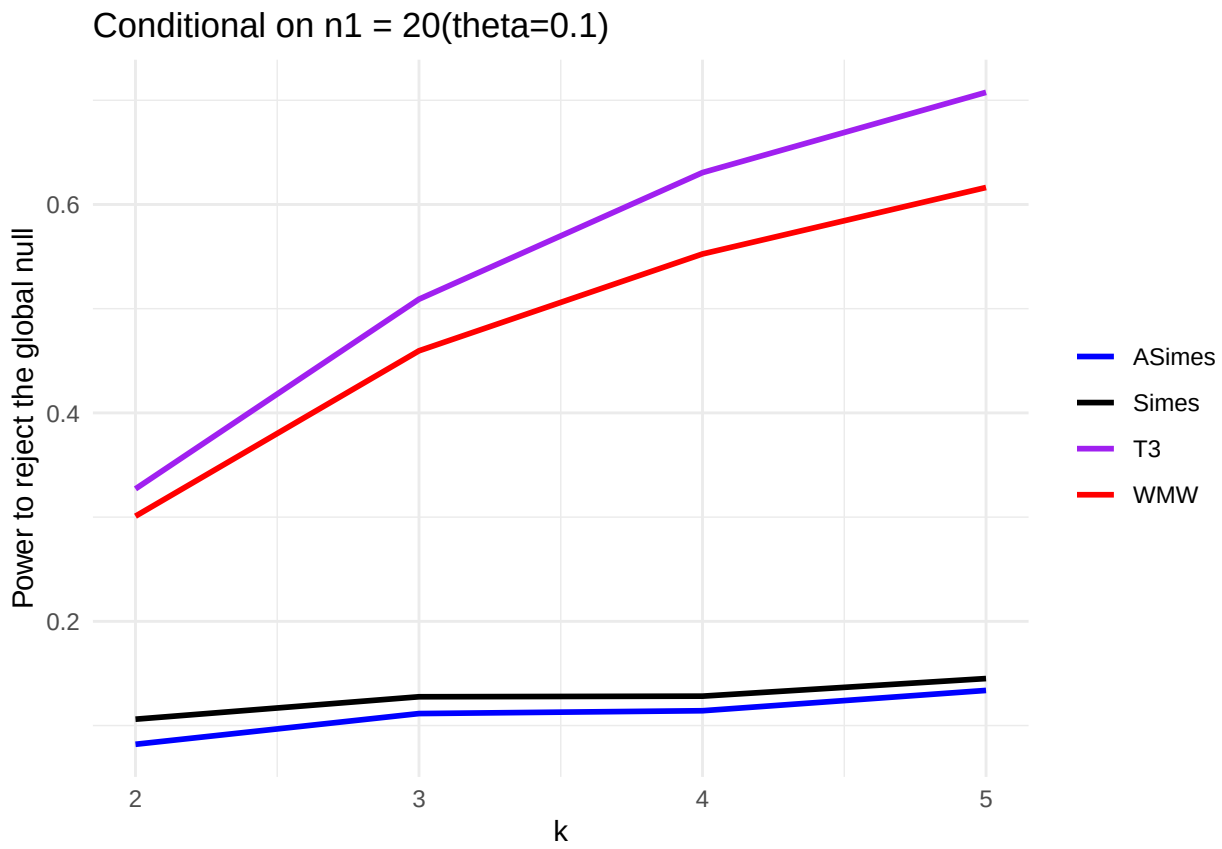
```
}
```




```
##      pow_BH pow_StoBH pow_WMW pow_T3
## k=2 0.1023    0.0574  0.1082 0.1297
## k=3 0.1015    0.0600  0.1188 0.1406
## k=4 0.1017    0.0615  0.1284 0.1534
## k=5 0.1063    0.0627  0.1312 0.1571
```




```
##      pow_BH pow_StoBH pow_WMW pow_T3
## k=2 0.1065    0.0723  0.1893 0.2108
## k=3 0.1105    0.0816  0.2381 0.2759
## k=4 0.1157    0.0860  0.2771 0.3275
## k=5 0.1249    0.0933  0.3018 0.3664
```




```
##      pow_BH pow_StoBH pow_WMW pow_T3
## k=2 0.1061    0.0820  0.3009 0.3271
## k=3 0.1276    0.1115  0.4596 0.5091
## k=4 0.1282    0.1142  0.5525 0.6306
## k=5 0.1451    0.1337  0.6163 0.7076
```