Comparison between different local tests: Simes, Simes with Storey and Wilcoxon-Mann-Whitney using the natural outliers distribution

2023-07-28

The aim is to compare on real datasets the performance of three closed testing procedures, which respectively use Simes local test with and without Storey estimator for the proportion of true null hypotheses and Wilcoxon-Mann-Whitney local test. We will consider outlier population to be the set of observations tagged as "outlier" in the dataset of interest.

R functions and libraries

```
library(nout)
library(R.matlab)
library(isotree)
library(farff)
library(tictoc)
library(tidyverse)
library(doSNOW)
library(ggplot2)
library(hommel)

compact_results = function(res){
  resT=as.data.frame(t(res))

  results = list()
  for(j in 1:length(n1s)){
    lb.d = as.data.frame(
      cbind("d_BH"=unlist(res[[j]][rownames(res[[j]])=="d_BH",]),
            "d_StoBH"=unlist(res[[j]][rownames(res[[j]])=="d_StoBH",]),
            "d_Sim"=unlist(res[[j]][rownames(res[[j]])=="d_Sim",]),
            "d_StoSimes"=unlist(res[[j]][rownames(res[[j]])=="d_StoSimes",]),
            "d_WMW"=unlist(res[[j]][rownames(res[[j]])=="d_WMW",])
            )
    )
    mean.lb.d = apply(lb.d, MARGIN = 2, FUN = mean)

    power.GlobalNull = as.data.frame(lb.d>0)
    mean.powerGlobalNull = apply(power.GlobalNull, MARGIN = 2, FUN = mean)

    n.disc = as.data.frame(
      cbind("n.disc.Simes" = unlist(res[[j]][rownames(res[[j]])=="n.disc.Simes",]),
            "n.disc.Simes2" = unlist(res[[j]][rownames(res[[j]])=="n.disc.Simes2",]),
             "n.disc.StoSimes" = unlist(res[[j]][rownames(res[[j]])=="n.disc.StoSimes",]),
            "n.disc.WMW" = unlist(res[[j]][rownames(res[[j]])=="n.disc.WMW",]),
            "n.disc.WMW.cpp" = unlist(res[[j]][rownames(res[[j]])=="n.disc.WMW.cpp",])
            )
    )
```

```
    mean.n.disc = apply(n.disc, MARGIN = 2, FUN = mean)
    #mean.n.disc_pos = apply(n.disc>0, MARGIN = 2, FUN = mean)

    results[[j]] = list("lb.d" = lb.d,
                        "mean.lb.d" = mean.lb.d,
                        "power.GlobalNull" = power.GlobalNull,
                        "mean.powerGlobalNull" = mean.powerGlobalNull,
                        "n.disc" = n.disc,
                        "mean.n.disc" = mean.n.disc,
                        #"mean.n.disc>0" = mean.n.disc_pos,
                        "pi.not" = res[[j]][rownames(res[[j]])=="pi.not",],
                        "uniques" = res[[j]][rownames(res[[j]])=="uniques",],
                        "n1" = res[[j]][rownames(res[[j]])=="n1",1],
                        "alpha" = res[[j]][rownames(res[[j]])=="alpha",1])
  }
  return(results)
}


TrainingIsoForest = function(l, dataset){

  tr_ind = sample(in_ind, size = l)
  tr = dataset[tr_ind,]
  isofo.model = isotree::isolation.forest(tr, ndim=ncol(dataset), ntrees=10, nthreads=1,
                            scoring_metric = "depth", output_score = TRUE)$model
  in_index2 = setdiff(in_ind, tr_ind)

  return(list("model"=isofo.model, "inlier_remaining" = in_index2))

}



CompareMethodNaturalOutliers = function(B, n1, n, out_ind, inlier_remaining, isofo.model, dataset){

  n0 = n-n1
  foreach(b = 1:B, .combine=cbind) %dopar% {
    if(n1==0){
      N = n0 + m
      in_index3 = sample(inlier_remaining, size = N)
      cal_ind = in_index3[1:m]
      te_ind = in_index3[(m+1):N]
      cal = dataset[cal_ind,]
      te = dataset[te_ind,]
      S_cal = predict.isolation_forest(isofo.model, cal, type = "score")
      S_te = predict.isolation_forest(isofo.model, te, type = "score")

      d_WMW = nout::d_MannWhitney(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)
  uniques = length(unique(c(S_cal, S_te)))
  return(list("d_BH" = d_BH,
              "d_StoBH" = d_StoBH,
              "d_Sim" = d_Sim,
              "n.disc.Simes" = 0,
              "n.disc.Simes2" = 0,
              "d_StoSimes" = d_StoSimes,
              "n.disc.StoSimes" = 0,
              "d_WMW" = d_WMW,
              "n.disc.WMW" = 0,
              "n.disc.WMW.cpp" = 0,
              "uniques" = uniques,
              "n1" = n1,
              "pi.not" = pi.not,
              "alpha" = alpha))
}

else{
  N = n0 + m
  in_index3 = sample(inlier_remaining, size = N)
  cal_ind = in_index3[1:m]
  if(n0!=0)
    tein_ind = in_index3[(m+1):N]
  else
    tein_ind = NULL
  teout_ind = sample(out_ind, size = n1)
  cal = dataset[cal_ind,]
  te = dataset[c(tein_ind, teout_ind),]
  S_cal = predict.isolation_forest(isofo.model, cal, type = "score")
  S_te = predict.isolation_forest(isofo.model, te, type = "score")

  d_WMW = nout::d_MannWhitney(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)
  uniques = length(unique(c(S_cal, S_te)))

  # outlier identification with WMW
  conf.pval = sapply(1:n, function(j) (1+sum(S_cal >= S_te[j]))/(m+1))
  confvalid.pval = conf.pval<alpha
  confvalid.index = which(conf.pval<alpha)

  n.disc.WMW.cpp=0
  n.disc.WMW=0
  if(d_WMW>0 & length(confvalid.index)!=0){
    outlierTF.WMW.cpp = sapply(confvalid.index, function(h)
      nout::dselection_MannWhitney(S_Y = S_te, S_X = S_cal, S = h, alpha=alpha))
    #outlier.identified_MannWhitney = confvalid.index[as.logical(outlierTF)]
    n.disc.WMW.cpp = sum(outlierTF.WMW.cpp)
```

```
        outlierTF.WMW = sapply(confvalid.index, function(h)
          nout::dselection.prova_MannWhitney(S_Y = S_te, S_X = S_cal, S = h, alpha=alpha))
        n.disc.WMW = sum(outlierTF.WMW)
      }

      # outlier identification with Simes
      n.disc.Simes=0
      n.disc.Simes2=0
      if(d_Sim>0 & length(confvalid.index)!=0){
        outlierTF.Simes = sapply(confvalid.index, function(h)
            nout::dselection_Simes(S_Y = S_te, S_X = S_cal, S = h, alpha=alpha))
        #outlier.identified_Simes = confvalid.index[as.logical(outlierTF)]
        n.disc.Simes = sum(outlierTF.Simes)
        p = hommel(conf.pval)
        n.disc.Simes2 = sum(p@adjusted <= alpha)
      }

      # outlier identification with StoreySimes
      n.disc.StoSimes=0
      if(d_StoSimes>0 & length(confvalid.index)!=0){
        outlierTF.StoSim = sapply(confvalid.index, function(h)
          nout::dselection_StoreySimes(S_Y = S_te, S_X = S_cal, S = h, alpha=alpha))
        #outlier.identified_StoSimes = confvalid.index[as.logical(outlierTFStoSim)]
        n.disc.StoSimes = sum(outlierTF.StoSim)
      }

      return(list("d_BH" = d_BH,
                  "d_StoBH" = d_StoBH,
                  "d_Sim" = d_Sim,
                  "n.disc.Simes" = n.disc.Simes,
                  "n.disc.Simes2" = n.disc.Simes2,
                  "d_StoSimes" = d_StoSimes,
                  "n.disc.StoSimes" = n.disc.StoSimes,
                  "d_WMW" = d_WMW,
                  "n.disc.WMW" = n.disc.WMW,
                  "n.disc.WMW.cpp" = n.disc.WMW.cpp,
                  "uniques" = uniques,
                  "n1" = n1,
                  "pi.not" = pi.not,
                  "alpha" = alpha))
    }
  }
}


estimatek = function(B, inlier_remaining, out_ind, isofo.model, dataset){
  ress = foreach(b = 1:B, .combine=c) %dopar% {
   inlier_ind = sample(inlier_remaining, size = 1)
   outlier_ind = sample(out_ind, size = 1)
   inlier = dataset[inlier_ind,]
   outlier = dataset[outlier_ind,]
   S_inlier = predict.isolation_forest(isofo.model, inlier, type = "score")
   S_outlier = predict.isolation_forest(isofo.model, outlier, type = "score")
```

```
   greater.logi = S_inlier<S_outlier

   return(greater.logi)
  }

  greater.prob = mean(ress)
  k=greater.prob/(1-greater.prob)
  return(k)
}
```

In the following we set the calibration set and the test set size, respectively l and m, so that the nominal level α is proportional to $\frac{m}{l+1}$. The train set size is equal to n and the number of iterations is $B = 10^4$.

Digits dataset

The dataset is available at http://odds.cs.stonybrook.edu/pendigits-dataset.

```
set.seed(321)

# Initializing parameters
B = 10^4
m = 199
l = 199
n = 20
alpha = n/(l+1)
n1s = seq(from=0, to=n, by=1)

data = readMat("~/nout/trials/RealData/Datasets/Dataset digits/pendigits.mat")
dataset = cbind(data$X, data$y); colnames(dataset)[ncol(dataset)] = "y"
in_ind = which(dataset[,ncol(dataset)]==0)
out_ind = which(dataset[,ncol(dataset)]==1)

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

```
## [[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"
##
## [[1]][[3]]
##  [1] "hommel"    "nout"      "isotree"   "snow"      "stats"     "graphics"
##  [7] "grDevices" "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"
##
## [[2]][[3]]
##  [1] "hommel"    "nout"      "isotree"   "snow"      "stats"     "graphics"
##  [7] "grDevices" "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"
##
## [[3]][[3]]
##  [1] "hommel"    "nout"      "isotree"   "snow"      "stats"     "graphics"
##  [7] "grDevices" "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"
##
## [[4]][[3]]
##  [1] "hommel"    "nout"      "isotree"   "snow"      "stats"     "graphics"
##  [7] "grDevices" "utils"     "datasets"  "methods"   "base"
```

```
clusterExport(cluster, list("n", "m", "l", "in_ind", "out_ind", "dataset", "alpha"))

tic()
modeltrain = TrainingIsoForest(l=l, dataset=dataset)
kest = estimatek(B=B, inlier_remaining=modeltrain$inlier_remaining,
          out_ind=out_ind, isofo.model=modeltrain$model, dataset=dataset)
res = lapply(1:length(n1s),
             function(j) CompareMethodNaturalOutliers(B=B, n1=n1s[j], n=n,
                                                      dataset=dataset,
                                                      isofo.model=modeltrain$model,
                                                      out_ind=out_ind,
                                                      inlier_remaining=modeltrain$inlier_remaining))
toc()
```

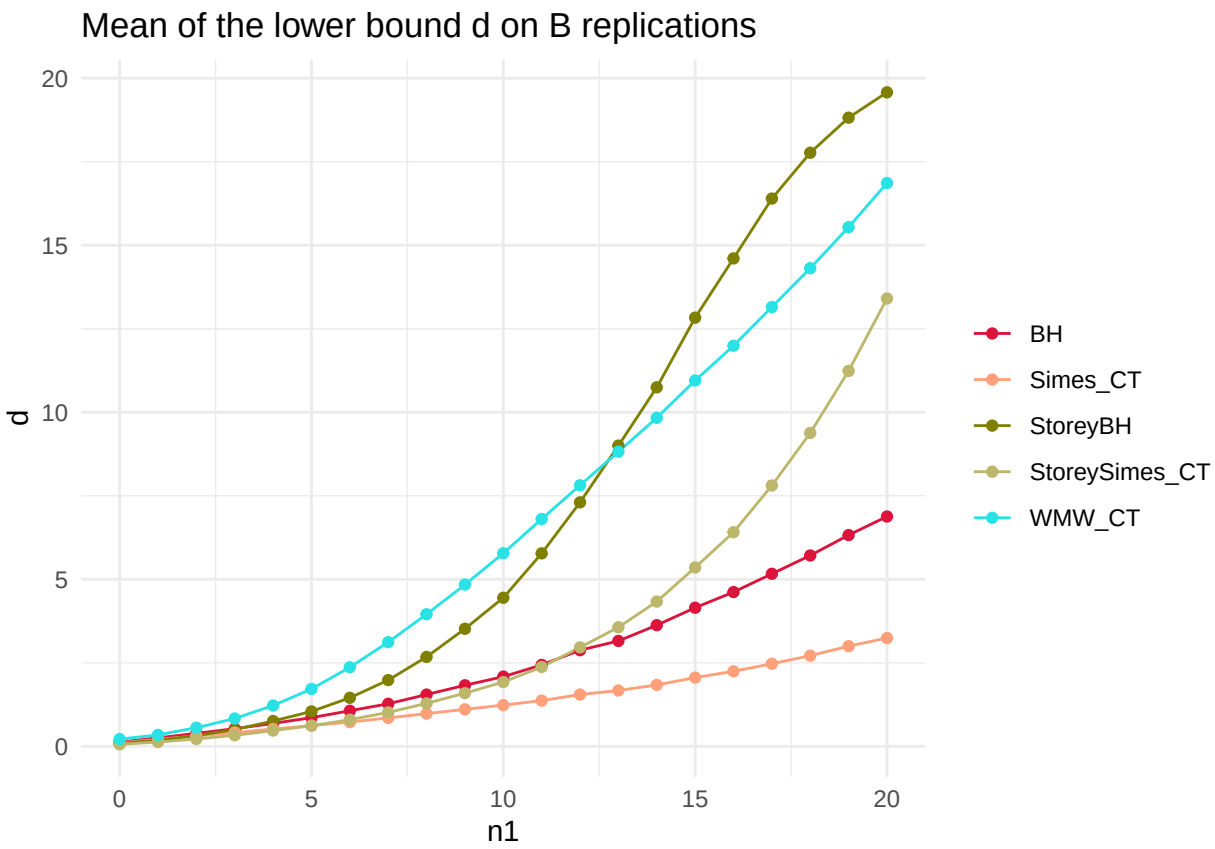
```
## 58444.48 sec elapsed
```

```
stopCluster(cluster)
```

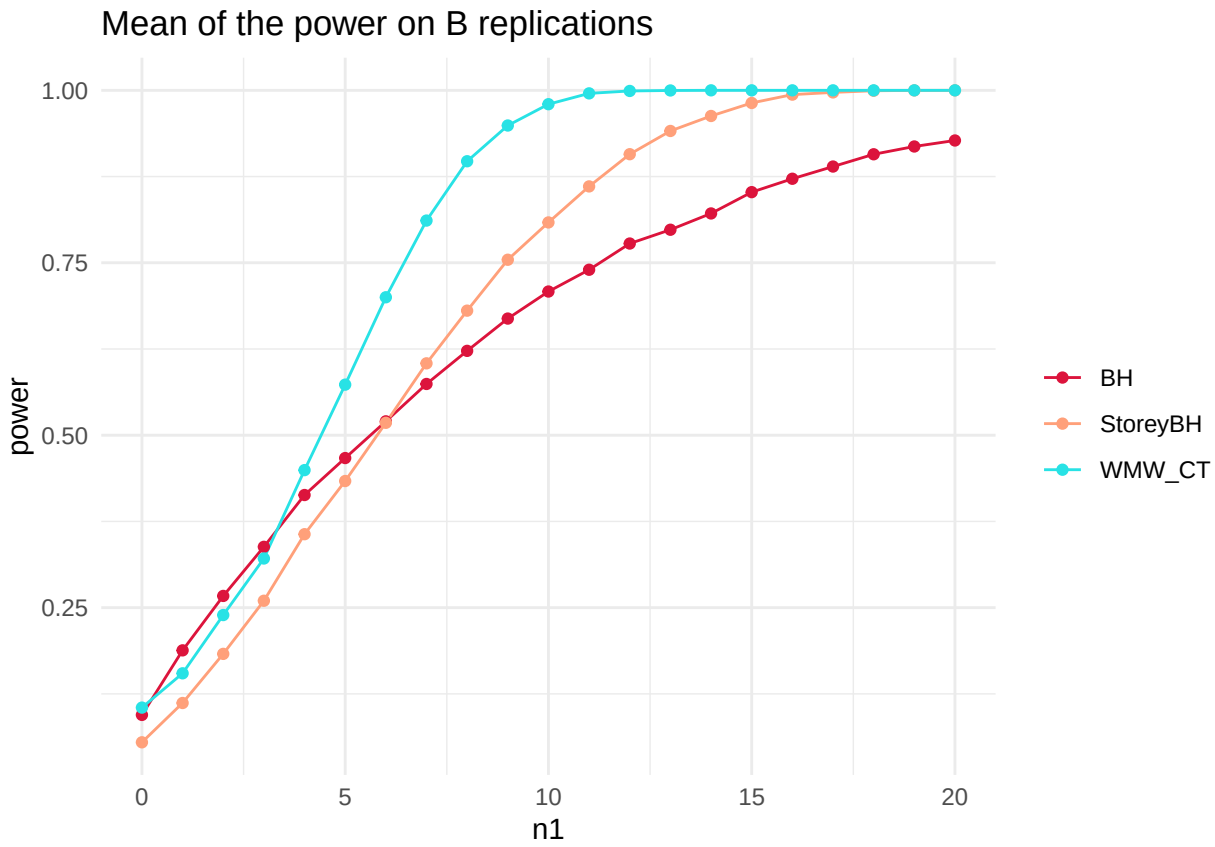
```
kest
```

```
## [1] 8.049774
```

```
results = compact_results(res)

d_BH = vector()
d_StoBH = vector()
d_Sim = vector()
d_StoSimes = vector()
d_WMW = vector()

pow_BH = vector()
pow_StoBH = vector()
pow_Sim = vector()
pow_StoSimes = vector()
pow_WMW = vector()

for(j in 1:length(n1s)){
  d_BH[j] = results[[j]]$mean.lb.d[1]
  d_StoBH[j] = results[[j]]$mean.lb.d[2]
  d_Sim[j] = results[[j]]$mean.lb.d[3]
  d_StoSimes[j] = results[[j]]$mean.lb.d[4]
  d_WMW[j] = results[[j]]$mean.lb.d[5]

  pow_BH[j] = results[[j]]$mean.powerGlobalNull[1]
  pow_StoBH[j] = results[[j]]$mean.powerGlobalNull[2]
  pow_Sim[j] = results[[j]]$mean.powerGlobalNull[3]
  pow_StoSimes[j] = results[[j]]$mean.powerGlobalNull[4]
  pow_WMW[j] = results[[j]]$mean.powerGlobalNull[5]

}

# Plot discoveries
df <- data.frame(
  x = n1s,
  BH = d_BH,
  StoreyBH = d_StoBH,
  Simes_CT = d_Sim,
  StoreySimes_CT = d_StoSimes,
  WMW_CT = d_WMW
)
df_long <- tidyr::pivot_longer(df, cols = -x, names_to = "group", values_to = "y")

ggplot(df_long, aes(x = x, y = y, color = group)) +
  geom_line() +
  geom_point()+
  scale_color_manual(values = c("#DC143C", "#FFA07A", "#808000", "#BDB76B", 5)) +
  labs(x = "n1", y = "d", title = "Mean of the lower bound d on B replications") +
  theme_minimal() +
  theme(legend.title = element_blank())
```




```
# Plot power
dfpower <- data.frame(
  x = n1s,
  BH = pow_BH,
  StoreyBH = pow_StoBH,
  WMW_CT = pow_WMW
)
df_long_power <- tidyr::pivot_longer(dfpower, cols = -x, names_to = "group", values_to = "y")

# Plot the lines with different colors and legends
ggplot(df_long_power, aes(x = x, y = y, color = group)) +
  geom_line() +
  geom_point()+
  scale_color_manual(values = c("#DC143C","#FFA07A",5)) +
  labs(x = "n1", y = "power", title = "Mean of the power on B replications") +
  theme_minimal() +
  theme(legend.title = element_blank())
```




```
n.disc.tablelist = list()
for(i in 1:length(n1s)){
  n.disc.tablelist[[i]] = matrix(ncol = 5, nrow = 2)
  colnames(n.disc.tablelist[[i]]) = c("Simes", "Simes2", "StoSimes", "WMW", "WMW.cpp")
  rownames(n.disc.tablelist[[i]]) = c("mean.n.disc", "mean.d")
  n.disc.tablelist[[i]][1,] = apply(results[[i]][["n.disc"]], MARGIN = 2, FUN = mean)
  n.disc.tablelist[[i]][2,] = results[[i]]$mean.lb.d[c(3,3,4,5,5)]
}

for(i in 1:length(n1s)){
  cat("\n")
  cat(paste("n1=", n1s[i]))
  cat("\n")
  print(n.disc.tablelist[[i]])
}
```

```
##
## n1= 0
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 0.0000 0.0000   0.0000 0.0000  0.0000
## mean.d      0.1025 0.1025   0.0614 0.2171  0.2171
##
## n1= 1
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 0.1925 0.1925   0.1008 0.0000  0.5312
## mean.d      0.2096 0.2096   0.1287 0.3425  0.3425
```

```
##
## n1= 2
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 0.2846 0.2846   0.1751 0.0000  0.8732
## mean.d      0.3103 0.3103   0.2209 0.5584  0.5584
##
## n1= 3
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 0.3774 0.3774   0.2631 0.0000  1.2806
## mean.d      0.4116 0.4116   0.3310 0.8331  0.8331
##
## n1= 4
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 0.4704 0.4704   0.3667 0.0000  1.9028
## mean.d      0.5195 0.5195   0.4721 1.2222  1.2222
##
## n1= 5
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 0.5571 0.5571   0.4773 0.0000  2.5729
## mean.d      0.6211 0.6211   0.6160 1.7174  1.7174
##
## n1= 6
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 0.6395 0.6395   0.5969 0.0000  3.3641
## mean.d      0.7264 0.7264   0.7977 2.3667  2.3667
##
## n1= 7
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 0.7500 0.7500   0.7436 0.0000  4.1499
## mean.d      0.8516 0.8516   1.0167 3.1176  3.1176
##
## n1= 8
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 0.8429 0.8429   0.8890 0.0008  4.9044
## mean.d      0.9808 0.9808   1.2855 3.9548  3.9548
##
## n1= 9
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 0.9376 0.9376   1.0362 0.0002  5.5307
## mean.d      1.1078 1.1078   1.5929 4.8416  4.8416
##
## n1= 10
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 1.0387 1.0387   1.1901 0.0021  6.1189
## mean.d      1.2347 1.2347   1.9239 5.7800  5.7800
##
## n1= 11
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 1.1211 1.1211   1.3482 0.0016  6.6731
## mean.d      1.3691 1.3691   2.3760 6.8059  6.8059
##
## n1= 12
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 1.2378 1.2378   1.5963 0.0090  7.1973
```

```
## mean.d      1.5507 1.5507   2.9622 7.8135  7.8135
##
## n1= 13
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 1.2996 1.2996   1.7791 0.0132  7.6289
## mean.d      1.6719 1.6719   3.5645 8.8210  8.8210
##
## n1= 14
##               Simes Simes2 StoSimes    WMW WMW.cpp
## mean.n.disc 1.3815 1.3815   2.0276 0.0348  8.1466
## mean.d      1.8393 1.8393   4.3337 9.8341  9.8341
##
## n1= 15
##               Simes Simes2 StoSimes     WMW WMW.cpp
## mean.n.disc 1.4852 1.4852   2.4335  0.0802  8.6053
## mean.d      2.0577 2.0577   5.3542 10.9518 10.9518
##
## n1= 16
##               Simes Simes2 StoSimes     WMW WMW.cpp
## mean.n.disc 1.5812 1.5812   2.8627  0.1624  9.0902
## mean.d      2.2464 2.2464   6.4093 11.9925 11.9925
##
## n1= 17
##               Simes Simes2 StoSimes     WMW WMW.cpp
## mean.n.disc 1.6971 1.6971   3.4768  0.3975  9.5935
## mean.d      2.4734 2.4734   7.8078 13.1507 13.1507
##
## n1= 18
##               Simes Simes2 StoSimes     WMW WMW.cpp
## mean.n.disc 1.7922 1.7922   4.3056  0.9642 10.0716
## mean.d      2.7144 2.7144   9.3810 14.3120 14.3120
##
## n1= 19
##               Simes Simes2 StoSimes     WMW WMW.cpp
## mean.n.disc 1.9019 1.9027   5.4591  2.1279 10.5644
## mean.d      2.9991 2.9991  11.2371 15.5413 15.5413
##
## n1= 20
##               Simes Simes2 StoSimes     WMW WMW.cpp
## mean.n.disc 1.9703 1.9703   7.0978  5.1422 11.0693
## mean.d      3.2433 3.2433  13.4056 16.8610 16.8610
```

```
resDigits0.1 = list("raw.res"=res,
                    "k.est" = kest,
                    "compact.results" = results,
                    "n.disc.tablelist" = n.disc.tablelist)
save(resDigits0.1, file="~/nout/trials/RealData/PowerStudy/FinalSimu/Digits/resDigits0.1")
```