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

2023-07-24

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)

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

  results = list()
  for(j in 1:length(n1s)){
    discoveries = 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.discoveries = apply(discoveries, MARGIN = 2, FUN = mean)

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

    out_identification = as.data.frame(
      cbind("out.identif_WMW"=
              unlist(res[[j]][rownames(res[[j]])=="outlier.identified_WMW",]),
            "out.identif_StoSimes"=
              unlist(res[[j]][rownames(res[[j]])=="outlier.identified_StoSimes",]),
            "out.identif_Simes"=
              unlist(res[[j]][rownames(res[[j]])=="outlier.identified_Simes",])
            )
    )
```

```
    mean.out_identification = apply(out_identification, MARGIN = 2, FUN = mean)
    mean.out_identification_pos = apply(out_identification>0, MARGIN = 2, FUN = mean)

    results[[j]] = list("discoveries" = discoveries,
                        "mean.discoveries" = mean.discoveries,
                        "power.GlobalNull" = power.GlobalNull,
                        "mean.powerGlobalNull" = mean.powerGlobalNull,
                        "out_identification" = out_identification,
                        "mean.out_identification" = mean.out_identification,
                        "mean.out_identification>0" = mean.out_identification_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){
      n0 = n
      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,
              "d_StoSimes" = d_StoSimes,
              "d_WMW" = d_WMW,
              "outlier.identified_WMW" = 0,
              "outlier.identified_Simes" = 0,
              "outlier.identified_StoSimes" = 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)

  if(d_WMW>0){
    outlierTF = sapply(confvalid.index, function(h)
        nout::dselection_MannWhitney(S_Y = S_te, S_X = S_cal, S = h, alpha=alpha))
    outlier.identified_WMW = confvalid.index[as.logical(outlierTF)]
  }
  else  outlier.identified_WMW = NULL

  # outlier identification with Simes
```

```
      if(d_Sim>0){
        outlierTF = 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)]
      }
      else  outlier.identified_Simes = NULL

      # outlier identification with StoreySimes
      if(d_StoSimes>0){
        outlierTF = 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(outlierTF)]
      }
      else  outlier.identified_StoSimes = NULL

      return(list("d_BH" = d_BH,
                  "d_StoBH" = d_StoBH,
                  "d_Sim" = d_Sim,
                  "d_StoSimes" = d_StoSimes,
                  "d_WMW" = d_WMW,
                  "outlier.identified_WMW" = length(outlier.identified_WMW),
                  "outlier.identified_Simes" = length(outlier.identified_Simes),
                  "outlier.identified_StoSimes" = length(outlier.identified_StoSimes),
                  "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 $B = 10^4$.

Covertype dataset

The dataset is available at http://odds.cs.stonybrook.edu/forestcovercovertype-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 cover type/cover.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))})
```

```
## [[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"
```

```
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()
```

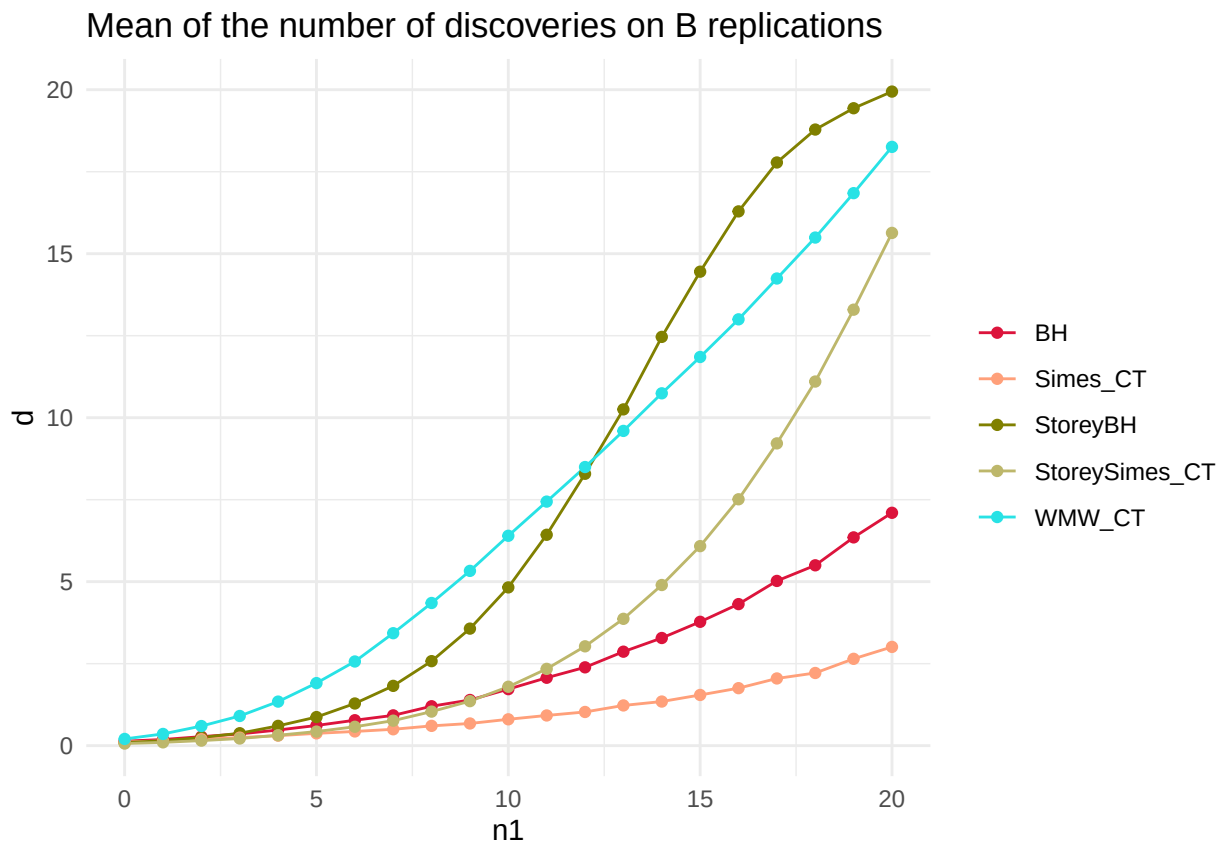
```
## 9382.81 sec elapsed
```

```
stopCluster(cluster)

kest
```

```
## [1] 10.56069
```

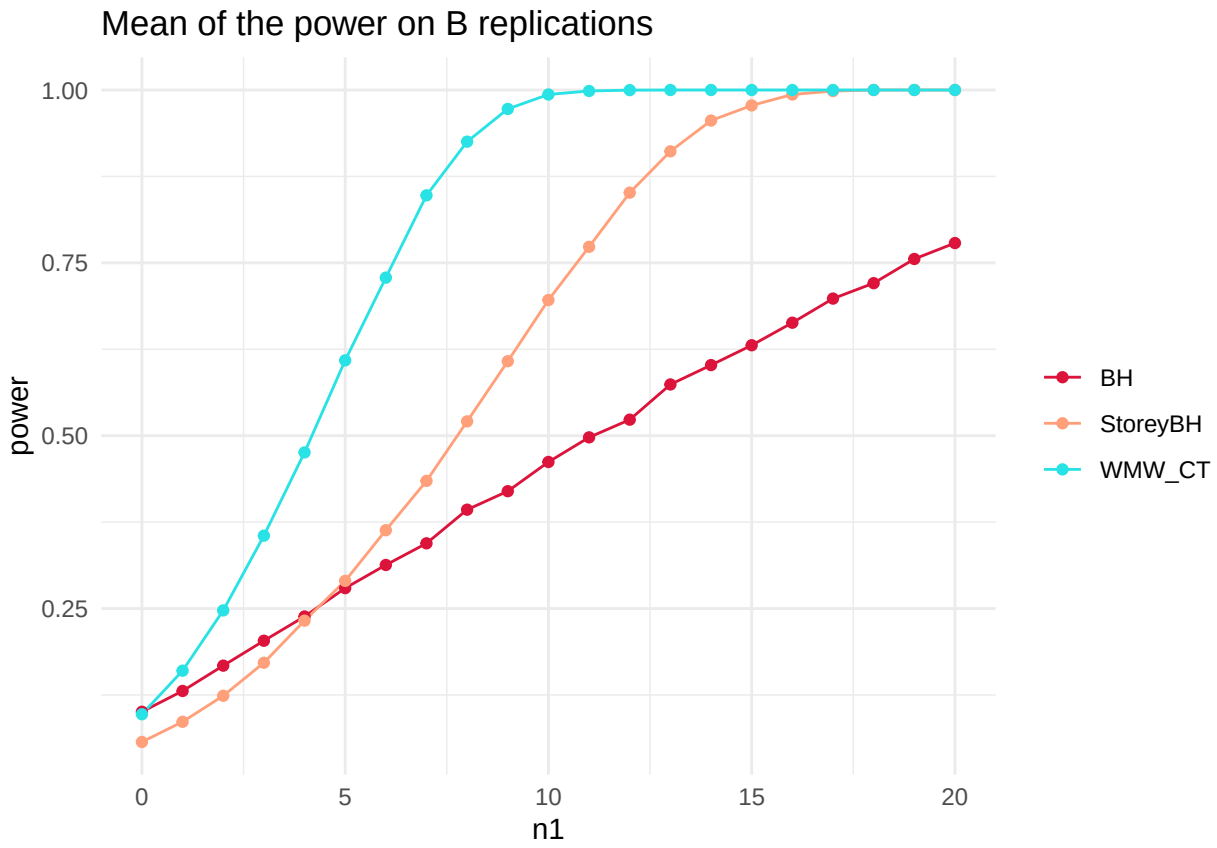
```
results = compact_results(res)

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]]$mean.discoveries[1]
  d_StoBH[j] = results[[j]]$mean.discoveries[2]
  d_Sim[j] = results[[j]]$mean.discoveries[3]
  d_StoSimes[j] = results[[j]]$mean.discoveries[4]
  d_WMW[j] = results[[j]]$mean.discoveries[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 number of discoveries 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())
```




```
outlier.identification = list()
for(i in 1:length(n1s)){
  outlier.identification[[i]] = matrix(nrow = 3, ncol = 4)
  rownames(outlier.identification[[i]]) = c("WMW", "Simes", "StoSimes")
  colnames(outlier.identification[[i]]) = c("mean.out.identif", "%successful.identification",
                                            "mean.d", "mean.d>0(power)")
  outlier.identification[[i]][,1] = apply(
    results[[i]][["out_identification"]], MARGIN = 2, FUN = mean)
  outlier.identification[[i]][,2] = apply(
    results[[i]][["out_identification"]]>0, MARGIN = 2, FUN = mean)
  outlier.identification[[i]][,3] = results[[i]]$mean.discoveries[c(3,4,5)]
  outlier.identification[[i]][,4] = results[[i]]$mean.powerGlobalNull[c(3,4,5)]
}
```

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

```
##
## n1= 0         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                        0                          0 0.1120          0.1007
## Simes                      0                          0 0.0644          0.0570
## StoSimes                   0                          0 0.2049          0.0971
##
## n1= 1         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   0.5700                     0.1583 0.1481          0.1308
## Simes                 0.0000                     0.0000 0.1006          0.0862
## StoSimes              0.1322                     0.1181 0.3574          0.1601
##
## n1= 2         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   0.9686                     0.2455 0.1975          0.1674
## Simes                 0.0000                     0.0000 0.1539          0.1238
## StoSimes              0.1636                     0.1425 0.5976          0.2473
##
## n1= 3         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   1.4933                     0.3542 0.2451          0.2034
## Simes                 0.0000                     0.0000 0.2178          0.1716
## StoSimes              0.1990                     0.1699 0.9051          0.3554
##
## n1= 4         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   2.1643                     0.4740 0.3047          0.2385
## Simes                 0.0000                     0.0000 0.3168          0.2325
## StoSimes              0.2323                     0.1889 1.3431          0.4758
##
## n1= 5         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   2.9990                     0.6082 0.3753          0.2795
## Simes                 0.0000                     0.0000 0.4295          0.2900
## StoSimes              0.2712                     0.2134 1.9079          0.6090
##
## n1= 6         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   3.8962                     0.7282 0.4333          0.3130
## Simes                 0.0000                     0.0000 0.5762          0.3633
## StoSimes              0.2908                     0.2305 2.5663          0.7286
##
## n1= 7         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   4.8678                     0.8470 0.4994          0.3443
## Simes                 0.0000                     0.0000 0.7624          0.4346
## StoSimes              0.3316                     0.2498 3.4292          0.8476
##
## n1= 8         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   5.7626                     0.9252 0.6031          0.3930
## Simes                 0.0000                     0.0000 1.0387          0.5207
## StoSimes              0.3686                     0.2708 4.3501          0.9253
##
## n1= 9         mean.out.identif %successful.identification mean.d mean.d>0(power)
```

```
## WMW               6.4933                      0.9724 0.6764          0.4197
## Simes             0.0000                      0.0000 1.3576          0.6077
## StoSimes          0.3907                      0.2848 5.3292          0.9726
##
## n1= 10         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               7.1530                      0.9933 0.8037          0.4620
## Simes             0.0000                      0.0000 1.7962          0.6962
## StoSimes          0.4409                      0.3054 6.3995          0.9935
##
## n1= 11         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               7.7183                      0.9984 0.9208          0.4976
## Simes             0.0000                      0.0000 2.3411          0.7732
## StoSimes          0.4622                      0.3184 7.4419          0.9986
##
## n1= 12         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               8.2413                      0.9998 1.0274          0.5232
## Simes             0.0000                      0.0000 3.0289          0.8515
## StoSimes          0.4735                      0.3209 8.4958          0.9998
##
## n1= 13         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               8.8007                      1.0000 1.2260          0.5741
## Simes             0.0000                      0.0000 3.8674          0.9113
## StoSimes          0.5427                      0.3562 9.5970          1.0000
##
## n1= 14         mean.out.identif %successful.identification  mean.d mean.d>0(power)
## WMW               9.3419                      1.0000  1.3454          0.6021
## Simes             0.0000                      0.0000  4.8982          0.9558
## StoSimes          0.5628                      0.3609 10.7437          1.0000
##
## n1= 15         mean.out.identif %successful.identification  mean.d mean.d>0(power)
## WMW               9.8650                      1.0000  1.5455          0.6308
## Simes             0.0000                      0.0000  6.0838          0.9777
## StoSimes          0.6001                      0.3729 11.8513          1.0000
##
## n1= 16         mean.out.identif %successful.identification  mean.d mean.d>0(power)
## WMW              10.3830                      0.9999  1.7529          0.6634
## Simes             0.0000                      0.0000  7.5122          0.9935
## StoSimes          0.6275                      0.3802 12.9994          1.0000
##
## n1= 17         mean.out.identif %successful.identification  mean.d mean.d>0(power)
## WMW              10.9930                      1.0000  2.0489          0.6983
## Simes             0.0000                      0.0000  9.2163          0.9985
## StoSimes          0.6891                      0.4036 14.2438          1.0000
##
## n1= 18         mean.out.identif %successful.identification  mean.d mean.d>0(power)
## WMW              11.4485                      1.0000  2.2162          0.7206
## Simes             0.0000                      0.0000 11.1020          1.0000
## StoSimes          0.7101                      0.4125 15.4926          1.0000
##
## n1= 19         mean.out.identif %successful.identification  mean.d mean.d>0(power)
## WMW              12.0652                       1.000  2.6480          0.7556
## Simes             0.0000                       0.000 13.2934          1.0000
## StoSimes          0.8156                       0.439 16.8466          1.0000
##
```

```
## n1= 20         mean.out.identif %successful.identification  mean.d mean.d>0(power)
## WMW               12.6174                     1.0000  3.0110          0.7786
## Simes              0.0000                     0.0000 15.6342          1.0000
## StoSimes           0.9142                     0.4483 18.2577          1.0000
```

```
resCover0.1 = list("raw.res"=res,
                   "k.est" = kest,
                   "compact.results" = results,
                   "outlier.identification" = outlier.identification)
save(resCover0.1, file="~/nout/trials/RealData/PowerStudy/FinalSimu/Cover/resCover0.1")
```