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

2023-08-04

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(readr)
library(foreign)
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 $B = 10^4$.

ALOI dataset

The dataset is available at https://www.dbs.ifi.lmu.de/research/outlier-evaluation/DAMI/literature/ALOI.

```
set.seed(321)

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

dataset = read.arff("~/nout/trials/RealData/Datasets/Dataset ALOI/ALOI_withoutdupl.arff")
out_ind = which(dataset$outlier=="yes")
in_ind = which(dataset$outlier=="no")

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()
```

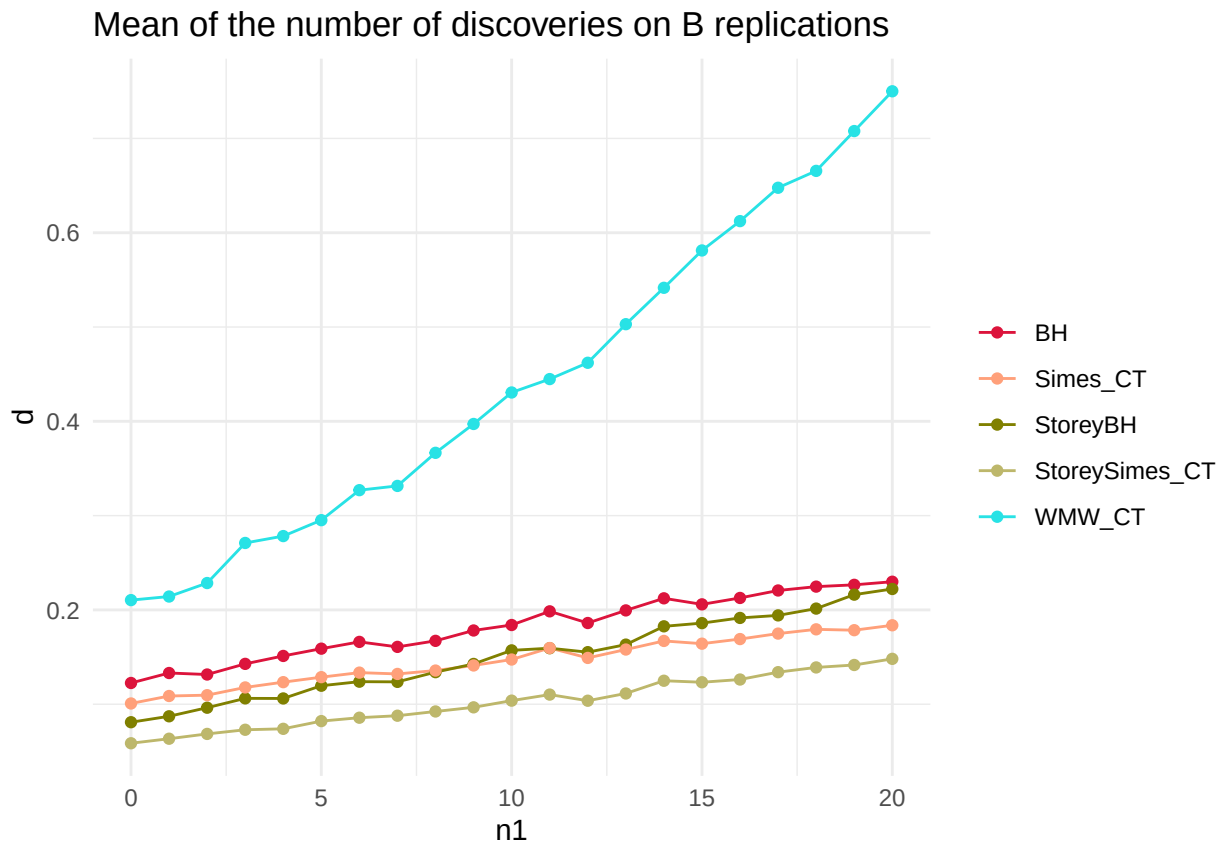
```
## 9387.75 sec elapsed
```

```
stopCluster(cluster)

kest
```

```
## [1] 1.200704
```

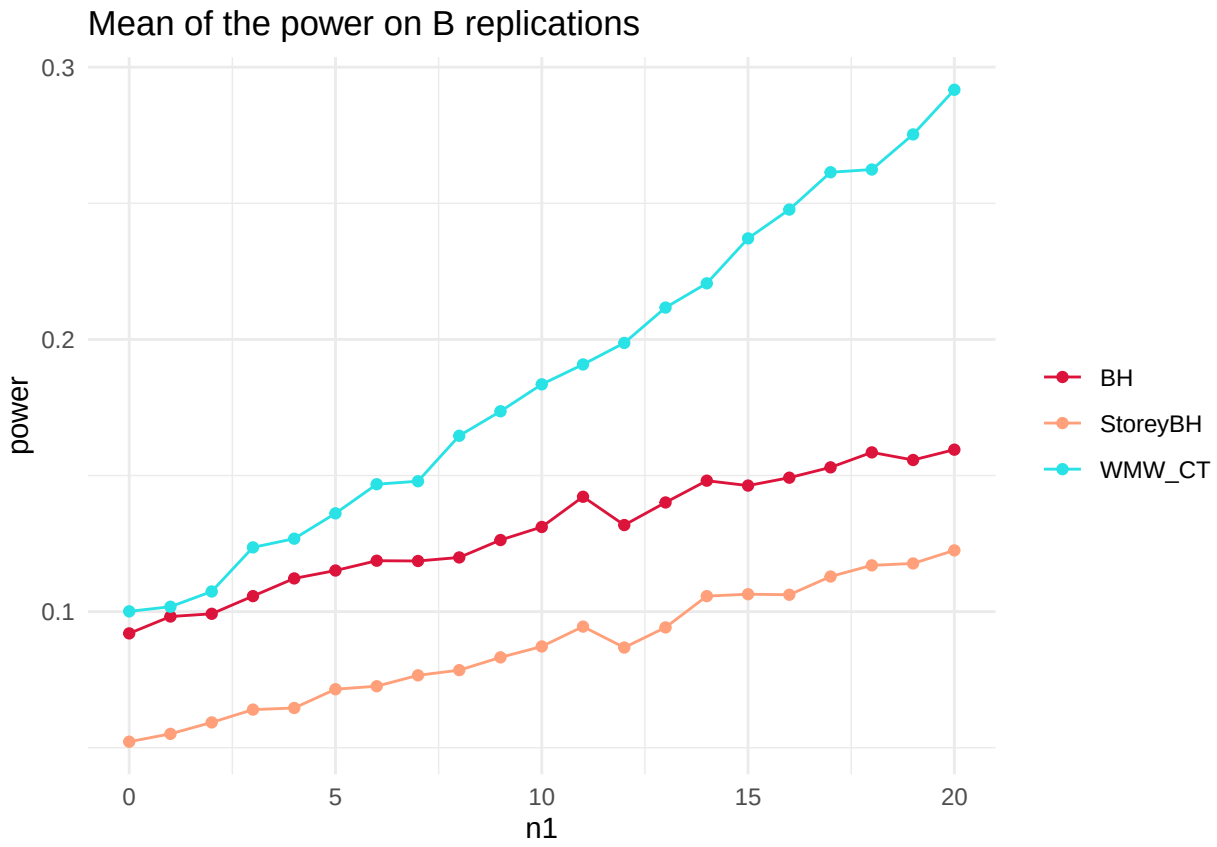
```
results = compact_results(res)

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]

}
```

```r
# 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())
```




```r
# Plot power
dfpower <- data.frame(
  x = n1s,
  BH = pow_BH,
  StoreyBH = pow_StoBH,
  WMW_CT = pow_WMW
)
```

```r
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())
```




```r
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)]
}

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 0.1008          0.0920
## Simes                      0                           0 0.0586          0.0522
## StoSimes                   0                           0 0.2104          0.1001
##
## n1= 1         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   0.3182                      0.0981 0.1087          0.0982
## Simes                 0.0492                      0.0436 0.0634          0.0551
## StoSimes              0.0981                      0.0891 0.2142          0.1018
##
## n1= 2         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   0.3410                      0.1045 0.1096          0.0992
## Simes                 0.0532                      0.0467 0.0685          0.0593
## StoSimes              0.0994                      0.0905 0.2285          0.1074
##
## n1= 3         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   0.3987                      0.1211 0.1178          0.1057
## Simes                 0.0560                      0.0501 0.0729          0.0640
## StoSimes              0.1056                      0.0959 0.2710          0.1236
##
## n1= 4         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   0.4036                      0.1240 0.1234          0.1122
## Simes                 0.0559                      0.0508 0.0739          0.0646
## StoSimes              0.1098                      0.1020 0.2783          0.1268
##
## n1= 5         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   0.4321                      0.1324 0.1287          0.1151
## Simes                 0.0623                      0.0557 0.0821          0.0715
## StoSimes              0.1159                      0.1051 0.2952          0.1361
##
## n1= 6         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   0.4682                      0.1435 0.1335          0.1187
## Simes                 0.0657                      0.0570 0.0857          0.0726
## StoSimes              0.1184                      0.1063 0.3270          0.1468
##
## n1= 7         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   0.4669                      0.1441 0.1321          0.1186
## Simes                 0.0682                      0.0606 0.0878          0.0766
## StoSimes              0.1189                      0.1079 0.3315          0.1479
##
## n1= 8         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   0.5361                      0.1602 0.1357          0.1199
## Simes                 0.0707                      0.0615 0.0923          0.0785
## StoSimes              0.1205                      0.1084 0.3666          0.1646
##
## n1= 9         mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW                   0.5559                      0.1692 0.1411          0.1263
## Simes                 0.0749                      0.0654 0.0967          0.0832
```

```
## StoSimes          0.1253                    0.1124 0.3972          0.1736
##
## n1= 10        mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               0.5908                    0.1792 0.1474          0.1311
## Simes             0.0801                    0.0697 0.1038          0.0872
## StoSimes          0.1328                    0.1196 0.4306          0.1835
##
## n1= 11        mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               0.6194                    0.1861 0.1596          0.1422
## Simes             0.0859                    0.0757 0.1102          0.0945
## StoSimes          0.1435                    0.1295 0.4448          0.1908
##
## n1= 12        mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               0.6463                    0.1947 0.1491          0.1318
## Simes             0.0798                    0.0691 0.1037          0.0868
## StoSimes          0.1328                    0.1182 0.4621          0.1987
##
## n1= 13        mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               0.6912                    0.2081 0.1580          0.1401
## Simes             0.0864                    0.0757 0.1114          0.0942
## StoSimes          0.1423                    0.1279 0.5030          0.2117
##
## n1= 14        mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               0.7266                    0.2154 0.1671          0.1481
## Simes             0.0974                    0.0848 0.1249          0.1057
## StoSimes          0.1484                    0.1325 0.5416          0.2206
##
## n1= 15        mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               0.7760                    0.2317 0.1642          0.1463
## Simes             0.0963                    0.0845 0.1233          0.1064
## StoSimes          0.1487                    0.1333 0.5812          0.2371
##
## n1= 16        mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               0.8071                    0.2422 0.1691          0.1492
## Simes             0.0970                    0.0843 0.1262          0.1062
## StoSimes          0.1514                    0.1343 0.6123          0.2477
##
## n1= 17        mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               0.8627                    0.2551 0.1749          0.1530
## Simes             0.1046                    0.0914 0.1340          0.1129
## StoSimes          0.1559                    0.1387 0.6477          0.2614
##
## n1= 18        mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               0.8567                    0.2558 0.1794          0.1585
## Simes             0.1103                    0.0961 0.1390          0.1170
## StoSimes          0.1601                    0.1434 0.6656          0.2624
##
## n1= 19        mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               0.9200                    0.2707 0.1785          0.1557
## Simes             0.1077                    0.0936 0.1416          0.1177
## StoSimes          0.1594                    0.1408 0.7078          0.2753
##
## n1= 20        mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW               0.9726                    0.2858 0.1837          0.1595
```

```
## Simes              0.1172                    0.1004 0.1481          0.1225
## StoSimes           0.1650                    0.1458 0.7500          0.2917
```

```
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")
```