

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

2023-07-22

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(nls)){
    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 = 1)
    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]
  tein_ind = in_index3[(m+1):N]
  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.probab = mean(ress)
  k=greater.probab/(1-greater.probab)
  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 = 2
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))})

## [[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"
##
##
## [[8]]
## [[8]][[1]]
## [1] "isotree"    "snow"      "stats"      "graphics"   "grDevices" "utils"
## [7] "datasets"   "methods"   "base"
##
## [[8]][[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=1, 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()

```

```
## 9.61 sec elapsed
```

```
stopCluster(cluster)
```

```
kest
```

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

```

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

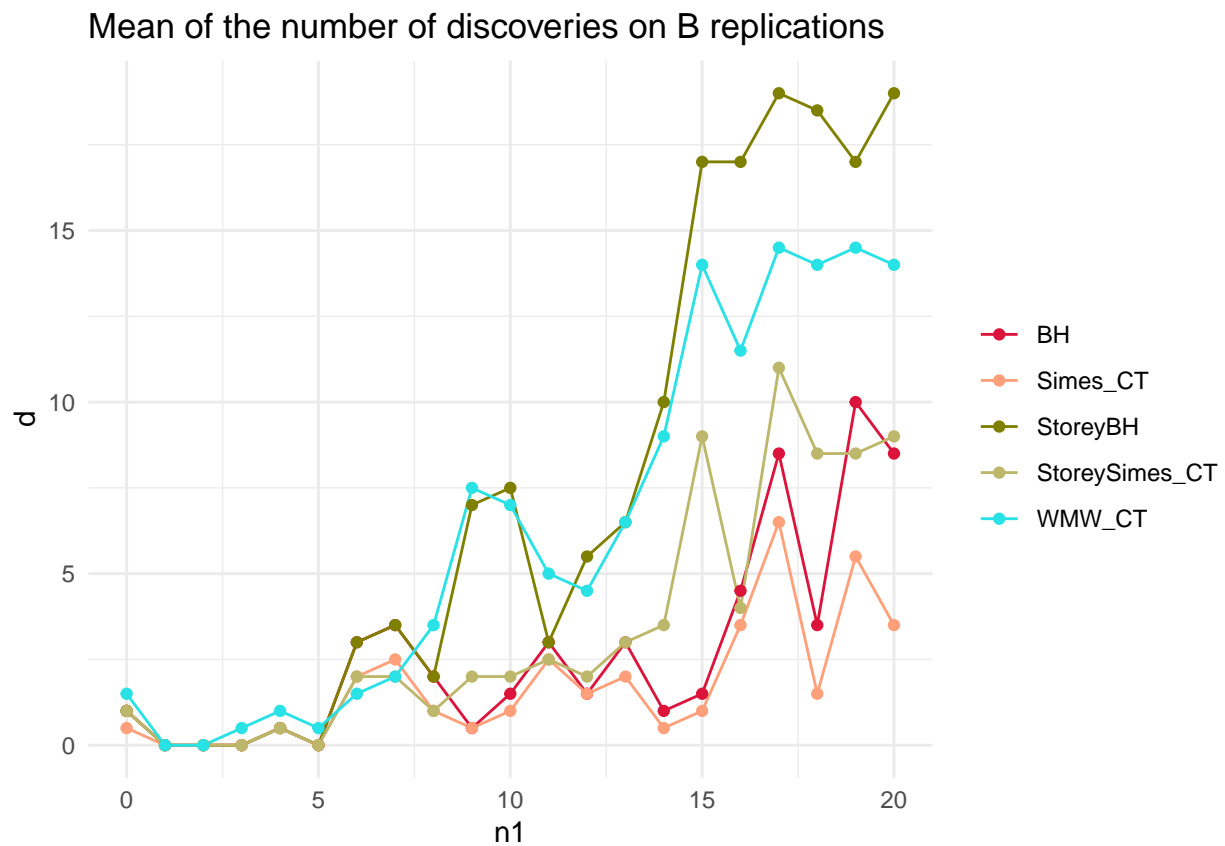
```

```

)
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
)

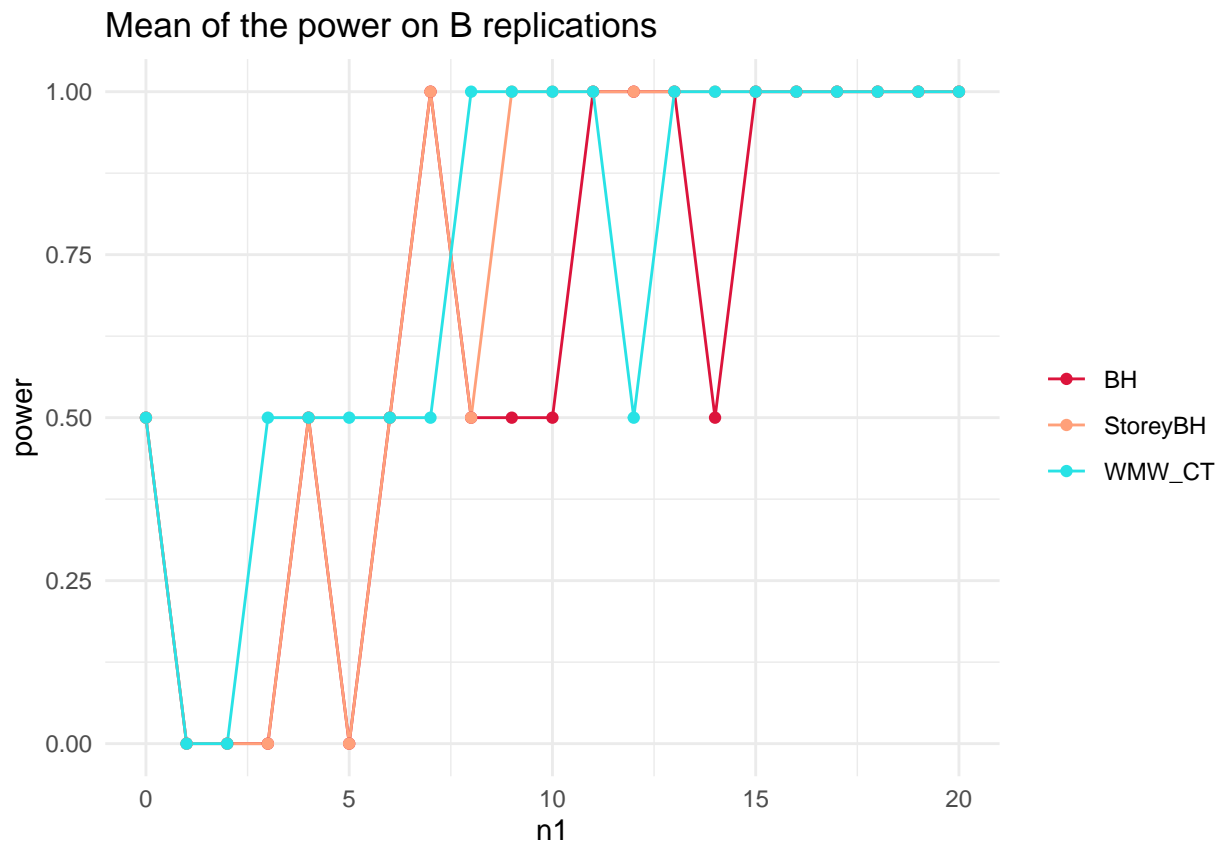
df_long_power <- tidyr::pivot_longer(dfpower, cols = -x, names_to = "group", values_to = "y")

# 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(nls)){
  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(nls)){
  cat("\n")
  cat(paste("n1=", nls[i]))
  print(outlier.identification[[i]])
}
```

```
##
```

```

## n1= 0      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW              0              0      0.5              0.5
## Simes              0              0      1.0              0.5
## StoSimes          0              0      1.5              0.5
##
## n1= 1      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW              0              0      0              0
## Simes              0              0      0              0
## StoSimes          0              0      0              0
##
## n1= 2      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW              0              0      0              0
## Simes              0              0      0              0
## StoSimes          0              0      0              0
##
## n1= 3      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW              1              0.5      0.0              0.0
## Simes              0              0.0      0.0              0.0
## StoSimes          0              0.0      0.5              0.5
##
## n1= 4      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW              2.5              0.5      0.5              0.5
## Simes              0.5              0.5      0.5              0.5
## StoSimes          0.5              0.5      1.0              0.5
##
## n1= 5      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW              1              0.5      0.0              0.0
## Simes              0              0.0      0.0              0.0
## StoSimes          0              0.0      0.5              0.5
##
## n1= 6      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW              2.0              0.5      2.0              0.5
## Simes              0.5              0.5      2.0              0.5
## StoSimes          0.5              0.5      1.5              0.5
##
## n1= 7      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW              2.0              0.5      2.5              1.0
## Simes              0.5              0.5      2.0              1.0
## StoSimes          2.5              1.0      2.0              0.5
##
## n1= 8      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW              6.5              1.0      1.0              0.5
## Simes              1.0              0.5      1.0              0.5
## StoSimes          1.0              0.5      3.5              1.0
##
## n1= 9      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW              7.5              1.0      0.5              0.5
## Simes              0.5              0.5      2.0              1.0
## StoSimes          0.5              0.5      7.5              1.0
##
## n1= 10     mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW              6.5              1.0      1              0.5
## Simes              1.0              0.5      2              1.0
## StoSimes          1.0              0.5      7              1.0

```

```

##
## n1= 11      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW          4.0              1      2.5              1
## Simes        2.5              1      2.5              1
## StoSimes     2.5              1      5.0              1
##
## n1= 12      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW          3.5              0.5    1.5              1.0
## Simes        1.5              1.0    2.0              1.0
## StoSimes     1.5              1.0    4.5              0.5
##
## n1= 13      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW          7.5              1      2.0              1
## Simes        1.5              1      3.0              1
## StoSimes     1.5              1      6.5              1
##
## n1= 14      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW          5.5              1.0    0.5              0.5
## Simes        1.0              1.0    3.5              1.0
## StoSimes     0.5              0.5    9.0              1.0
##
## n1= 15      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW          5.5              1      1              1
## Simes        3.0              1      9              1
## StoSimes     1.0              1      14             1
##
## n1= 16      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW          8.5              1      3.5              1
## Simes        4.0              1      4.0              1
## StoSimes     3.5              1      11.5             1
##
## n1= 17      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW          13              1      6.5              1
## Simes        7              1      11.0             1
## StoSimes     3              1      14.5             1
##
## n1= 18      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW          8.0              1      1.5              1
## Simes        2.0              1      8.5              1
## StoSimes     1.5              1      14.0             1
##
## n1= 19      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW          9.0              1      5.5              1
## Simes        6.5              1      8.5              1
## StoSimes     3.5              1      14.5             1
##
## n1= 20      mean.out.identif %successful.identification mean.d mean.d>0(power)
## WMW          10.5             1      3.5              1
## Simes        3.5              1      9.0              1
## StoSimes     1.5              1      14.0             1

```

```

resDigits0.1 = list("raw.res"=res,
                    "k.est" = kest,
                    "compact.results" = results,

```

```
        "outlier.identification" = outlier.identification)  
save(resDigits0.1, file="~/nout/trials/RealData/PowerStudy/FinalSimu/Digits/resDigits0.1")
```