Comparison between different local tests: Simes, Simes with Storey and Wilcoxon-Mann-Whitney using the Lehmann alternative distribution with k=2

2023-11-22

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 simulate outliers distribution so that it will be to the Lehmann's alternative with $k = 2$. Denoting inliers distribution by F, we are going to simulate the outliers distribution corresponding to F^k with $k = 2$ in order to perform a power analysis and to show that closed testing procedure with Wilcoxon-Mann-Whitney local test is more powerful than closed testing with Simes local test with and without Storey estimator.

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){

  results = list()
  for(j in 1:length(n1s)){
    lb.d = as.data.frame(
      cbind("d_BH"=unlist(res[[j]]["d_BH",]),
            "d_StoBH"=unlist(res[[j]]["d_StoBH",]),
            "d_Sim"=unlist(res[[j]]["d_Sim",]),
            "d_StoSimes"=unlist(res[[j]]["d_StoSimes",]),
            "d_WMW"=unlist(res[[j]]["d_WMW",]),
            "d_T3"=unlist(res[[j]]["d_T3",])
            )
    )
    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)

    results[[j]] = list("lb.d" = lb.d,
                        "mean.lb.d" = lb.d,
                        "power.GlobalNull" = power.GlobalNull,
```

```
                        "mean.powerGlobalNull" = mean.powerGlobalNull,
                        "pi.not" = res[[j]]["pi.not",],
                        "S_cal" = (res[[j]][rownames(res[[j]])=="S_cal",]),
                        "S_te" = (res[[j]][rownames(res[[j]])=="S_te",]),
                        "uniques" = res[[j]]["uniques",],
                        "n1" = res[[j]]["n1",1],
                        "alpha" = 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))

}



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

  n0 = n-n1
  foreach(b = 1:B, .combine=cbind) %dopar% {

    N = n0 + m + k*n1
    in_index3 = sample(inlier_remaining, size = N)
    cal_ind = in_index3[1:m]
    te_ind.augmented = in_index3[(m+1):N]
    cal = dataset[cal_ind,]
    te = dataset[te_ind.augmented,]
    S_cal = predict.isolation_forest(isofo.model, cal, type = "score")
    augmented.S_te = predict.isolation_forest(isofo.model, te, type = "score")

    if(n1==0)
      S_te = augmented.S_te
    if(n1==n)
      S_te = sapply(1:n1, FUN=function(i) max(augmented.S_te[(1+k*(i-1)):(i*k)]))
    if(0<n1&n1<n)
      S_te = c(augmented.S_te[(1+k*n1):(n0+k*n1)],
               sapply(1:n1, FUN=function(i) max(augmented.S_te[(1+k*(i-1)):(i*k)])))

      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)
    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,
                "d_T3" = d_T3,
                "uniques" = uniques,
                "n1" = n1,
                "pi.not" = pi.not,
                "alpha" = alpha))
  }
}
```

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 $B = 10^5$.

Digits dataset

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

```
set.seed(321)

# Initializing parameters
B = 10^4
m = 1999
l = 1999
n = 200
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)
theta = length(out_ind)/nrow(dataset) # proportion of outliers in the entire dataset

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"
##
##
```

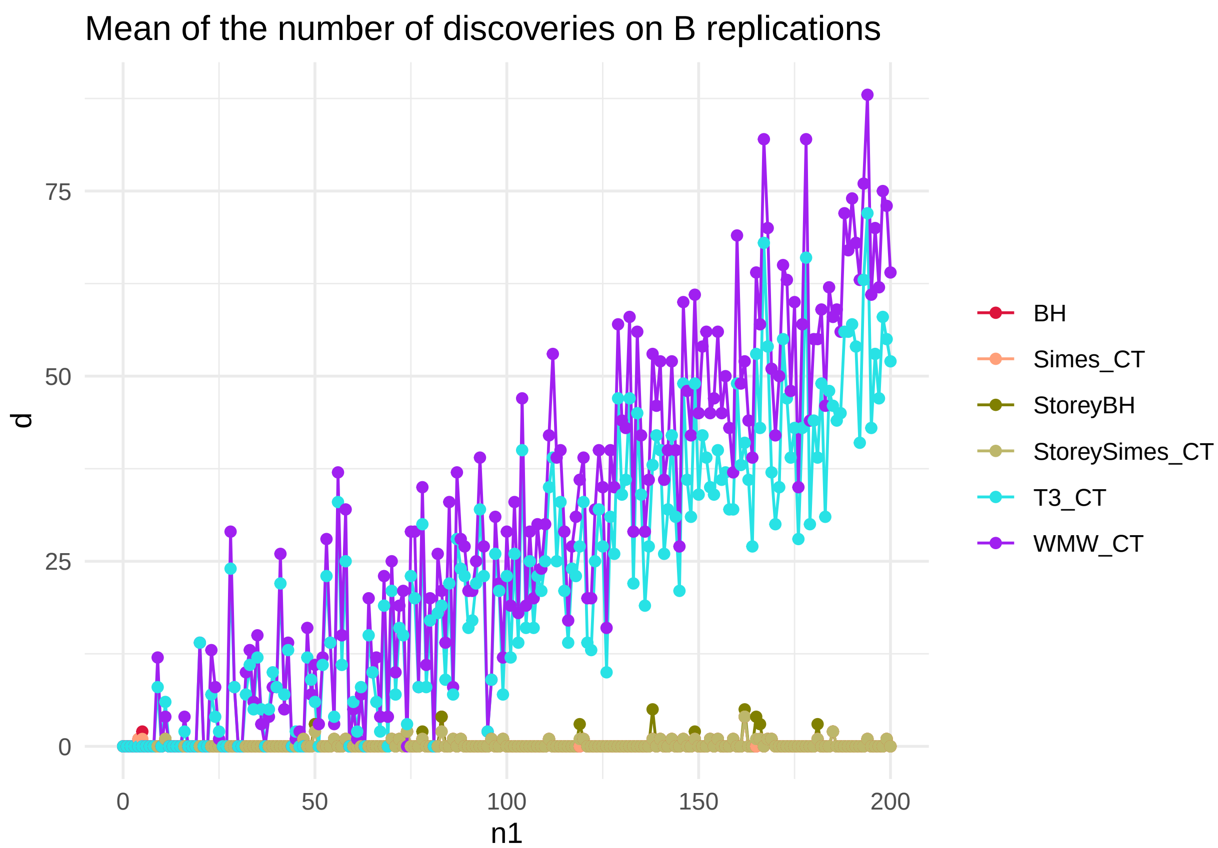
```
## [[2]]
## [[2]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[2]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
##
## [[3]]
## [[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)
res = lapply(1:length(n1s),
             function(j) CompareMethodLehmannOutliers(B=B, k=2, n1=n1s[j], n=n,
                                 dataset=dataset,
                                 isofo.model=modeltrain$model,
                                 out_ind=out_ind,
                                 inlier_remaining=modeltrain$inlier_remaining))
toc()
```

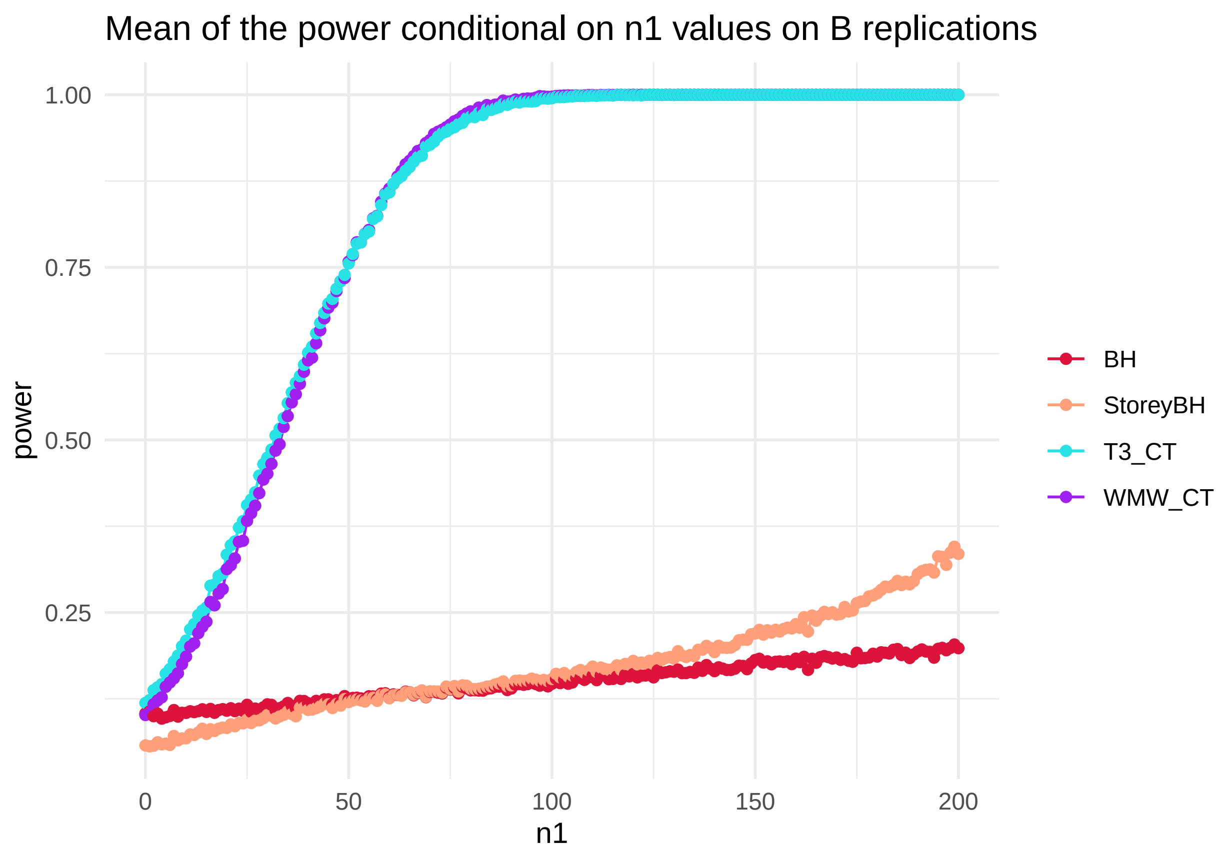
```
## 44529.91 sec elapsed
```

```
stopCluster(cluster)

results = compact_results(res)

d_BH = vector()
d_StoBH = vector()
d_Sim = vector()
d_StoSimes = vector()
d_WMW = vector()
d_T3 = vector()

pow_BH = vector()
```

```
pow_StoBH = vector()
pow_Sim = vector()
pow_StoSimes = vector()
pow_WMW = vector()
pow_T3 = vector()

for(j in 1:length(n1s)){
  d_BH[j] = unlist(results[[j]]$mean.lb.d[1])
  d_StoBH[j] = unlist(results[[j]]$mean.lb.d[2])
  d_Sim[j] = unlist(results[[j]]$mean.lb.d[3])
  d_StoSimes[j] = unlist(results[[j]]$mean.lb.d[4])
  d_WMW[j] = unlist(results[[j]]$mean.lb.d[5])
  d_T3[j] = unlist(results[[j]]$mean.lb.d[6])

  pow_BH[j] = unlist(results[[j]]$mean.powerGlobalNull[1])
  pow_StoBH[j] = unlist(results[[j]]$mean.powerGlobalNull[2])
  pow_Sim[j] = unlist(results[[j]]$mean.powerGlobalNull[3])
  pow_StoSimes[j] = unlist(results[[j]]$mean.powerGlobalNull[4])
  pow_WMW[j] = unlist(results[[j]]$mean.powerGlobalNull[5])
  pow_T3[j] = unlist(results[[j]]$mean.powerGlobalNull[6])

}

# Plot discoveries conditional on n1
df <- data.frame(
  x = n1s,
  BH = d_BH,
  StoreyBH = d_StoBH,
  Simes_CT = d_Sim,
  StoreySimes_CT = d_StoSimes,
  WMW_CT = d_WMW,
  T3_CT = d_T3
)
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, "purple")) +
  labs(x = "n1", y = "d", title = "Mean of the number of discoveries on B replications") +
  theme_minimal() +
  theme(legend.title = element_blank())
```




```
# Plot power conditional on n1
dfpower <- data.frame(
  x = n1s,
  BH = pow_BH,
  StoreyBH = pow_StoBH,
  WMW_CT = pow_WMW,
  T3_CT = pow_T3
)
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, "purple")) +
  labs(x = "n1", y = "power", title = "Mean of the power conditional on n1 values on B replications") +
  theme_minimal() +
  theme(legend.title = element_blank())
```




```r
# Table unconditional power
thetas = seq(from = 0, to = 1, by = 0.02)
probsn1 = sapply(thetas,
                 function(theta) sapply(1:n,
                                        function(k) choose(n,k)*(1-theta)^(n-k)*theta^(k)))
colnames(probsn1) = as.character(thetas)
rownames(probsn1) = as.character(1:n)
unconditional.power = cbind("theta" = thetas,
                            "uncond.pow_BH" = apply(pow_BH[-1]*probsn1, MARGIN = 2, sum),
                            "uncond.pow_StoreyBH" = apply(pow_StoBH[-1]*probsn1, MARGIN = 2, sum),
                            "uncond.pow_WMW" = apply(pow_WMW[-1]*probsn1, MARGIN = 2, sum),
                            "uncond.pow_T3" = apply(pow_T3[-1]*probsn1, MARGIN = 2, sum))
print(unconditional.power)
```

```
##      theta uncond.pow_BH uncond.pow_StoreyBH uncond.pow_WMW uncond.pow_T3
## 0     0.00    0.00000000          0.00000000      0.0000000     0.0000000
## 0.02  0.02    0.09897485          0.05920235      0.1293100     0.1489020
## 0.04  0.04    0.10314376          0.06637986      0.1669812     0.1902505
## 0.06  0.06    0.10604055          0.07324103      0.2091338     0.2338708
## 0.08  0.08    0.10774012          0.07907518      0.2559256     0.2805987
## 0.1   0.10    0.10890045          0.08444683      0.3076863     0.3319604
## 0.12  0.12    0.11014426          0.08982431      0.3641666     0.3875413
## 0.14  0.14    0.11175027          0.09496575      0.4238955     0.4456841
## 0.16  0.16    0.11380908          0.09986376      0.4857104     0.5049600
## 0.18  0.18    0.11627542          0.10472588      0.5482657     0.5641377
## 0.2   0.20    0.11886238          0.10952709      0.6096337     0.6216874
```

```
## 0.22  0.22   0.12127531         0.11404685      0.6680771     0.6760828
## 0.24  0.24   0.12352125         0.11818180      0.7222826     0.7263993
## 0.26  0.26   0.12572424         0.12194590      0.7715358     0.7724155
## 0.28  0.28   0.12789995         0.12538077      0.8157468     0.8140858
## 0.3   0.30   0.12995664         0.12851116      0.8548170     0.8510539
## 0.32  0.32   0.13183043         0.13135807      0.8883320     0.8828733
## 0.34  0.34   0.13358865         0.13400539      0.9160540     0.9094940
## 0.36  0.36   0.13531924         0.13654363      0.9382827     0.9312646
## 0.38  0.38   0.13700358         0.13898435      0.9556676     0.9486972
## 0.4   0.40   0.13863450         0.14137216      0.9689285     0.9623898
## 0.42  0.42   0.14032587         0.14387328      0.9787685     0.9729600
## 0.44  0.44   0.14217600         0.14662242      0.9858711     0.9809461
## 0.46  0.46   0.14415963         0.14961855      0.9908681     0.9868209
## 0.48  0.48   0.14624203         0.15285182      0.9942898     0.9910539
## 0.5   0.50   0.14846012         0.15637350      0.9965497     0.9940725
## 0.52  0.52   0.15079125         0.16015468      0.9979741     0.9961888
## 0.54  0.54   0.15304824         0.16401614      0.9988343     0.9976174
## 0.56  0.56   0.15503921         0.16780594      0.9993448     0.9985379
## 0.58  0.58   0.15679880         0.17156147      0.9996467     0.9991102
## 0.6   0.60   0.15855403         0.17542882      0.9998182     0.9994606
## 0.62  0.62   0.16046595         0.17951298      0.9999065     0.9996782
## 0.64  0.64   0.16248917         0.18385509      0.9999484     0.9998165
## 0.66  0.66   0.16449482         0.18848050      0.9999706     0.9999015
## 0.68  0.68   0.16645733         0.19345828      0.9999851     0.9999480
## 0.7   0.70   0.16852258         0.19898132      0.9999942     0.9999699
## 0.72  0.72   0.17092819         0.20528467      0.9999984     0.9999800
## 0.74  0.74   0.17366778         0.21221494      0.9999997     0.9999858
## 0.76  0.76   0.17622064         0.21908137      1.0000000     0.9999904
## 0.78  0.78   0.17806848         0.22548819      1.0000000     0.9999947
## 0.8   0.80   0.17941782         0.23196475      1.0000000     0.9999979
## 0.82  0.82   0.18083185         0.23912635      1.0000000     0.9999995
## 0.84  0.84   0.18233529         0.24688557      1.0000000     0.9999999
## 0.86  0.86   0.18376514         0.25543821      1.0000000     1.0000000
## 0.88  0.88   0.18578928         0.26582878      1.0000000     1.0000000
## 0.9   0.90   0.18872926         0.27763075      1.0000000     1.0000000
## 0.92  0.92   0.19094951         0.28830716      1.0000000     1.0000000
## 0.94  0.94   0.19145740         0.29794543      1.0000000     1.0000000
## 0.96  0.96   0.19263902         0.31057166      1.0000000     1.0000000
## 0.98  0.98   0.19615321         0.32596841      1.0000000     1.0000000
## 1     1.00   0.19830000         0.33490000      1.0000000     1.0000000
```

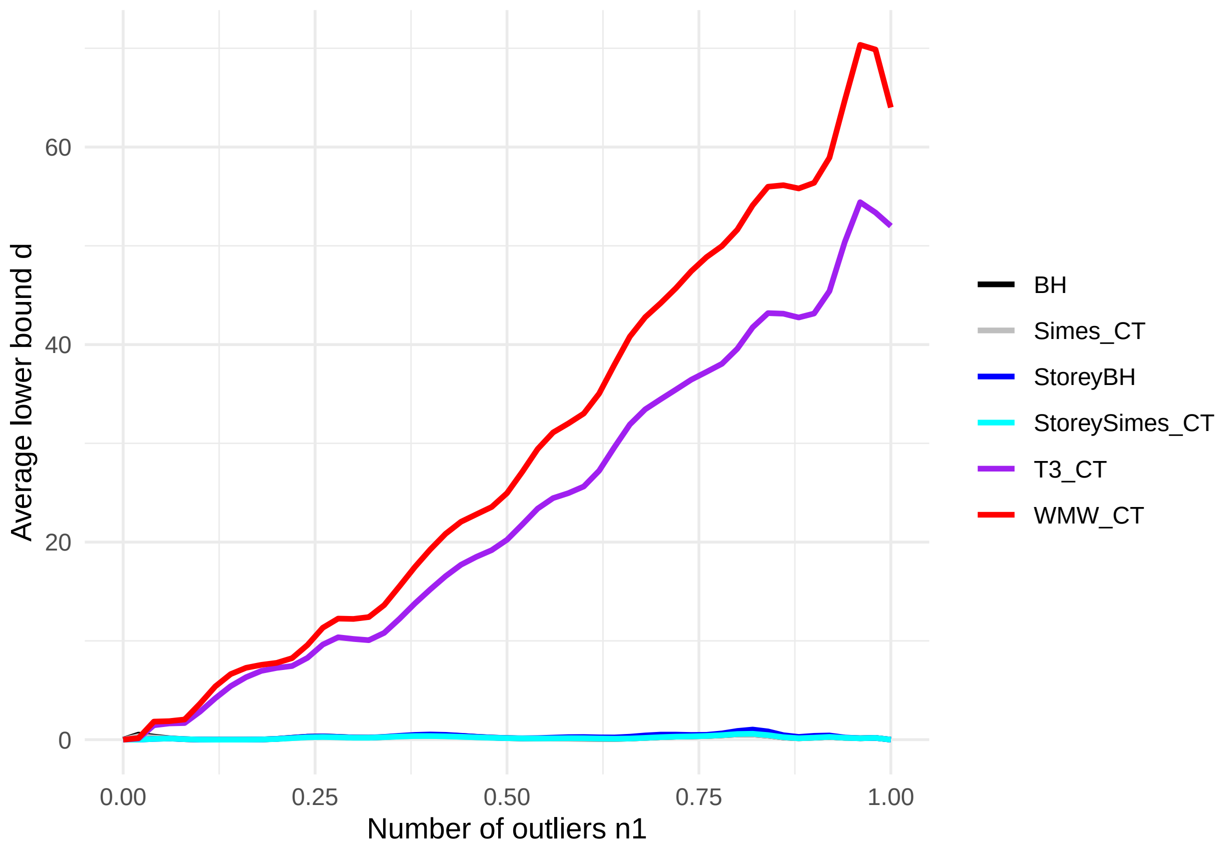
```
resDigits0.1k2_1999 = list("raw.res"=res)
save(resDigits0.1k2_1999, file="~/nout/trials/RealData/PowerStudy/FinalSimu/Digits/Lehmannk2/resDigits0
```

```
# load(file="~/nout/Examples/Digits/Lehmannk3/matrixDigits0.1k2_1999")
# results = compact_results(matrixDigits0.1k2_1999$raw.res)

# Compacting intermediate results in a matrix

d_BH = vector()
d_StoBH = vector()
d_Sim = vector()
d_StoSimes = vector()
d_WMW = vector()
d_T3 = vector()
```

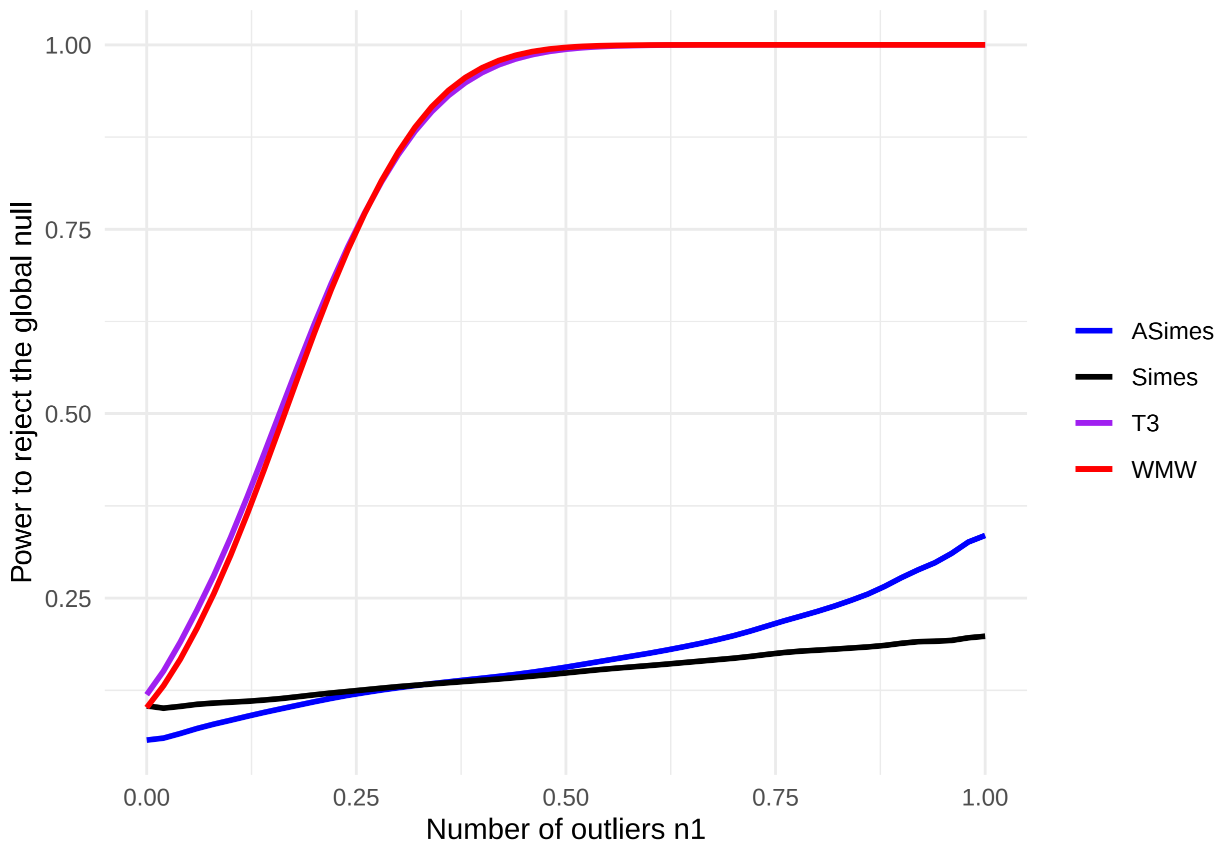
```r
pow.rejGlob_BH = vector()
pow.rejGlob_StoBH = vector()
pow.rejGlob_Sim = vector()
pow.rejGlob_StoSimes = vector()
pow.rejGlob_WMW = vector()
pow.rejGlob_T3 = vector()

for(j in 1:length(n1s)){
  d_BH[j] = unlist(results[[j]]$mean.lb.d[1])
  d_StoBH[j] = unlist(results[[j]]$mean.lb.d[2])
  d_Sim[j] = unlist(results[[j]]$mean.lb.d[3])
  d_StoSimes[j] = unlist(results[[j]]$mean.lb.d[4])
  d_WMW[j] = unlist(results[[j]]$mean.lb.d[5])
  d_T3[j] = unlist(results[[j]]$mean.lb.d[6])

  pow.rejGlob_BH[j] = unlist(results[[j]]$mean.powerGlobalNull[1])
  pow.rejGlob_StoBH[j] = unlist(results[[j]]$mean.powerGlobalNull[2])
  pow.rejGlob_Sim[j] = unlist(results[[j]]$mean.powerGlobalNull[3])
  pow.rejGlob_StoSimes[j] = unlist(results[[j]]$mean.powerGlobalNull[4])
  pow.rejGlob_WMW[j] = unlist(results[[j]]$mean.powerGlobalNull[5])
  pow.rejGlob_T3[j] = unlist(results[[j]]$mean.powerGlobalNull[6])

}

lb.d = matrix(nrow = (n+1), ncol = 6)
rownames(lb.d) = as.character(n1s)
colnames(lb.d) = c("FDR-BH", "FDR-Storey", "CT-Simes",
                   "CT-Storey", "CT-WMW", "CT-T3")
lb.d[,1] = d_BH
lb.d[,2] = d_StoBH
lb.d[,3] = d_Sim
lb.d[,4] = d_StoSimes
lb.d[,5] = d_WMW
lb.d[,6] = d_T3

pow.rejGlob = matrix(nrow = (n+1), ncol = 6)
rownames(pow.rejGlob) = as.character(seq(from=0, to=n, by=1))
colnames(pow.rejGlob) = c("FDR-BH", "FDR-Storey", "CT-Simes",
                          "CT-Storey", "CT-WMW", "CT-T3")
pow.rejGlob[,1] = pow.rejGlob_BH
pow.rejGlob[,2] = pow.rejGlob_StoBH
pow.rejGlob[,3] = pow.rejGlob_Sim
pow.rejGlob[,4] = pow.rejGlob_StoSimes
pow.rejGlob[,5] = pow.rejGlob_WMW
pow.rejGlob[,6] = pow.rejGlob_T3

matrixDigits0.1k2_1999 = list("lb.d.matrix" = lb.d,
                              "pow.rejGlob.matrix" = pow.rejGlob)
save(matrixDigits0.1k2_1999,
     file = paste0("~/nout/Examples/Digits/Lehmannk3","/matrixDigits0.1k2_1999"))
```

```r
#load(file = paste0("~/nout/Examples/Digits/Lehmannk3","/matrixDigits0.1k2_1999"))
```

```r
res = matrixDigits0.1k2_1999

thetas = seq(0,1, length.out=51)

pow_BH = round(sapply(thetas, function(p)
  sum( dbinom(0:n,size=n,prob=p) * res$pow.rejGlob.matrix[,1])),4)
pow_StoBH = round(sapply(thetas, function(p)
  sum( dbinom(0:n,size=n,prob=p) * res$pow.rejGlob.matrix[,2])),4)
pow_Simes = round(sapply(thetas, function(p)
  sum( dbinom(0:n,size=n,prob=p) * res$pow.rejGlob.matrix[,3])),4)
pow_ASimes = round(sapply(thetas, function(p)
  sum( dbinom(0:n,size=n,prob=p) * res$pow.rejGlob.matrix[,4])),4)
pow_WMW = round(sapply(thetas, function(p)
  sum( dbinom(0:n,size=n,prob=p) * res$pow.rejGlob.matrix[,5])),4)
pow_T3 = round(sapply(thetas, function(p)
  sum( dbinom(0:n,size=n,prob=p) * res$pow.rejGlob.matrix[,6])),4)

lb.d.BH = round(sapply(thetas, function(p)
  sum( dbinom(0:n,size=n,prob=p) * res$lb.d.matrix[,1])),4)
lb.d.StoBH = round(sapply(thetas, function(p)
  sum( dbinom(0:n,size=n,prob=p) * res$lb.d.matrix[,2])),4)
lb.d.Simes = round(sapply(thetas, function(p)
  sum( dbinom(0:n,size=n,prob=p) * res$lb.d.matrix[,3])),4)
lb.d.ASimes = round(sapply(thetas, function(p)
  sum( dbinom(0:n,size=n,prob=p) * res$lb.d.matrix[,4])),4)
lb.d.WMW = round(sapply(thetas, function(p)
  sum( dbinom(0:n,size=n,prob=p) * res$lb.d.matrix[,5])),4)
lb.d.T3 = round(sapply(thetas, function(p)
  sum( dbinom(0:n,size=n,prob=p) * res$lb.d.matrix[,6])),4)


# Plot lower bound d
df <- data.frame(
  x = thetas,
  BH = lb.d.BH,
  StoreyBH = lb.d.StoBH,
  Simes_CT = lb.d.Simes,
  StoreySimes_CT = lb.d.ASimes,
  WMW_CT = lb.d.WMW,
  T3_CT = lb.d.T3
)
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(size=1) +
  scale_color_manual(values = c("black","gray","blue", "cyan","purple","red")) +
  labs(x = "Number of outliers n1", y = "Average lower bound d") +
  theme_minimal() +
  theme(legend.title = element_blank())
```




```r
# Plot power
dfpower <- data.frame(
  x = thetas,
  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")

ggplot(df_long_power, aes(x = x, y = y, color = group)) +
  geom_line(size=1) +
  scale_color_manual(values = c("blue","black","purple","red")) +
  labs(x = "Number of outliers n1", y = "Power to reject the global null") +
  theme_minimal() +
  theme(legend.title = element_blank())
```




```
pow_WMW
```

```
##  [1] 0.1015 0.1311 0.1670 0.2091 0.2559 0.3077 0.3642 0.4239 0.4857 0.5483
## [11] 0.6096 0.6681 0.7223 0.7715 0.8157 0.8548 0.8883 0.9161 0.9383 0.9557
## [21] 0.9689 0.9788 0.9859 0.9909 0.9943 0.9965 0.9980 0.9988 0.9993 0.9996
## [31] 0.9998 0.9999 0.9999 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
## [41] 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
## [51] 1.0000
```

```
pow_T3
```

```
##  [1] 0.1190 0.1510 0.1903 0.2339 0.2806 0.3320 0.3875 0.4457 0.5050 0.5641
## [11] 0.6217 0.6761 0.7264 0.7724 0.8141 0.8511 0.8829 0.9095 0.9313 0.9487
## [21] 0.9624 0.9730 0.9809 0.9868 0.9911 0.9941 0.9962 0.9976 0.9985 0.9991
## [31] 0.9995 0.9997 0.9998 0.9999 0.9999 1.0000 1.0000 1.0000 1.0000 1.0000
## [41] 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
## [51] 1.0000
```