Rank tests for outlier detection

Chiara Gaia Magnani, Aldo Solari

2023-09-06

Installing *nout* package from Github

```
# install.packages("devtools")
devtools::install_github("chiaragaiamagnani/nout")
```

Paths

It may be the case that you need to change the following paths in order to run this Rmd file.

```
pathDatasets = file.path("C:", "Users", "chiar", "Documents",
                         "BiomJ - RankTestsForOutlierDetection", "R code", "Datasets",
                         fsep="\\")

pathResults = file.path("C:", "Users", "chiar", "Documents",
                        "BiomJ - RankTestsForOutlierDetection", "R code", fsep="\\")
```

R functions and libraries

```
library(nout)
library(R.matlab)
library(readr)
library(isotree)
library(tictoc)
library(foreign)
library(tidyverse)
library(doSNOW)
library(ggplot2)
library(hommel)
library(mvtnorm)
library(multcomp)

# Natural outlier distribution

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))

}
```

```r
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,
                  "d_StoSimes" = d_StoSimes,
                  "n.disc.StoSimes" = 0,
                  "d_WMW" = d_WMW,
                  "n.disc.WMW" = 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=0
      if(d_WMW>0 & length(confvalid.index)!=0){
        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
      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))
        n.disc.Simes = sum(outlierTF.Simes)
      }

      # 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))
        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,
                  "d_StoSimes" = d_StoSimes,
                  "n.disc.StoSimes" = n.disc.StoSimes,
                  "d_WMW" = d_WMW,
                  "n.disc.WMW" = n.disc.WMW,
                  "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)
}



# Lehmann's alternative

compact_resultsLehmann = 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)

    results[[j]] = list("discoveries" = discoveries,
                        "mean.discoveries" = mean.discoveries,
                        "power.GlobalNull" = power.GlobalNull,
                        "mean.powerGlobalNull" = mean.powerGlobalNull,
                        "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)
}
```

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

Intermediate results for Figure 4,5,6,7 and Table 3

Since training the one-class classification model and using it to make predictions on the calibration and test samples to get the conformity scores is time consuming, intermediate results are loaded from the workspace. The R code in this section produces intermediate results for each dataset which are saved as Rdata files. Then, intermediate results are compacted into matrices.

First we initialize the parameters using the same notation as in the paper.

```
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)
```

Pen-digits

```
data = readMat(paste0(pathDatasets, "\\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))})
clusterExport(cluster, list("n", "m", "l", "in_ind", "out_ind", "dataset", "alpha"))

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))
stopCluster(cluster)

kest

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()

disc_Sim = vector()
disc_StoSimes = vector()
```

```
disc_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]

  disc_Sim[j] = results[[j]]$mean.n.disc[1]
  disc_StoSimes[j] = results[[j]]$mean.n.disc[3]
  disc_WMW[j] = results[[j]]$mean.n.disc[4]
}


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", "StoSimes", "WMW")
  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(3,4,5)]
}

resDigits0.1 = list("raw.res"=res,
                    "k.est" = kest,
                    "compact.results" = results,
                    "n.disc.tablelist" = n.disc.tablelist)
save(resDigits0.1, file=paste0(pathResults,"\\resDigits0.1"))


# Compacting intermediate results in a matrix
n.disc_Sim = vector()
n.disc_StoreySim = vector()
n.disc_WMW = vector()

pow.rejGlob_BH = vector()
pow.rejGlob_StoBH = vector()
pow.rejGlob_Sim = vector()
pow.rejGlob_StoSimes = vector()
pow.rejGlob_WMW = vector()

lb.d_BH = vector()
lb.d_StoBH = vector()
lb.d_Sim = vector()
lb.d_StoSimes = vector()
lb.d_WMW = vector()
```

```
for(j in 1:length(n1s)){
  lb.d_BH[j] = resDigits0.1$compact.results[[j]]$mean.lb.d[1]
  lb.d_StoBH[j] = resDigits0.1$compact.results[[j]]$mean.lb.d[2]
  lb.d_Sim[j] = resDigits0.1$compact.results[[j]]$mean.lb.d[3]
  lb.d_StoSimes[j] = resDigits0.1$compact.results[[j]]$mean.lb.d[4]
  lb.d_WMW[j] = resDigits0.1$compact.results[[j]]$mean.lb.d[5]

  n.disc_Sim[j] = resDigits0.1$compact.results[[j]]$mean.n.disc[1]
  n.disc_StoreySim[j] = resDigits0.1$compact.results[[j]]$mean.n.disc[3]
  n.disc_WMW[j] = resDigits0.1$compact.results[[j]]$mean.n.disc[4]

  pow.rejGlob_BH[j] = resDigits0.1$compact.results[[j]]$mean.powerGlobalNull[1]
  pow.rejGlob_StoBH[j] = resDigits0.1$compact.results[[j]]$mean.powerGlobalNull[2]
  pow.rejGlob_Sim[j] = resDigits0.1$compact.results[[j]]$mean.powerGlobalNull[3]
  pow.rejGlob_StoSimes[j] = resDigits0.1$compact.results[[j]]$mean.powerGlobalNull[4]
  pow.rejGlob_WMW[j] = resDigits0.1$compact.results[[j]]$mean.powerGlobalNull[5]

}


lb.d = matrix(nrow = (n+1), ncol = 5)
rownames(lb.d) = as.character(n1s)
colnames(lb.d) = c("FDR-BH", "FDR-Storey", "CT-Simes", "CT-Storey", "CT-WMW")

lb.d[,1] = lb.d_BH
lb.d[,2] = lb.d_StoBH
lb.d[,3] = lb.d_Sim
lb.d[,4] = lb.d_StoSimes
lb.d[,5] = lb.d_WMW

pow.rejGlob = matrix(nrow = (n+1), ncol = 5)
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")
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

n.disc = matrix(nrow = (n+1), ncol = 3)
rownames(n.disc) = as.character(seq(from=0, to=n, by=1))
colnames(n.disc) = c("CT-Simes", "CT-StoreySimes", "CT-WMW")
n.disc[,1] = n.disc_Sim
n.disc[,2] = n.disc_StoreySim
n.disc[,3] = n.disc_WMW

matrixDigits0.1 = list("lb.d.matrix" = lb.d,
                       "pow.rejGlob.matrix" = pow.rejGlob,
                       "n.disc" = n.disc)
save(matrixDigits0.1, file = paste0(pathResults,"\\matrixDigits0.1"))
```

Credit card

```
dataset = read_csv(paste0(pathDatasets,"\\creditcard.csv"))
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))})
clusterExport(cluster, list("n", "m", "l", "in_ind", "out_ind", "dataset", "alpha"))

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))
stopCluster(cluster)

kest

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()

disc_Sim = vector()
disc_StoSimes = vector()
disc_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]

  disc_Sim[j] = results[[j]]$mean.n.disc[1]
  disc_StoSimes[j] = results[[j]]$mean.n.disc[3]
  disc_WMW[j] = results[[j]]$mean.n.disc[4]
}

n.disc.tablelist = list()
for(i in 1:length(n1s)){
  n.disc.tablelist[[i]] = matrix(ncol = 3, nrow = 2)
  colnames(n.disc.tablelist[[i]]) = c("Simes", "StoSimes", "WMW")
  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(3,4,5)]
}

resCreditCard0.1v2 = list("raw.res"=res,
                          "k.est" = kest,
                          "compact.results" = results,
                          "n.disc.tablelist" = n.disc.tablelist)
save(resCreditCard0.1v2,
     file=paste0(pathResults,"\\resCreditCard0.1v2"))


# Compacting intermediate results in a matrix
n.disc_Sim = vector()
n.disc_StoreySim = vector()
n.disc_WMW = vector()

pow.rejGlob_BH = vector()
pow.rejGlob_StoBH = vector()
pow.rejGlob_Sim = vector()
pow.rejGlob_StoSimes = vector()
pow.rejGlob_WMW = vector()

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

for(j in 1:length(n1s)){
  lb.d_BH[j] = resCreditCard0.1v2$compact.results[[j]]$mean.lb.d[1]
  lb.d_StoBH[j] = resCreditCard0.1v2$compact.results[[j]]$mean.lb.d[2]
  lb.d_Sim[j] = resCreditCard0.1v2$compact.results[[j]]$mean.lb.d[3]
  lb.d_StoSimes[j] = resCreditCard0.1v2$compact.results[[j]]$mean.lb.d[4]
  lb.d_WMW[j] = resCreditCard0.1v2$compact.results[[j]]$mean.lb.d[5]

  n.disc_Sim[j] = resCreditCard0.1v2$compact.results[[j]]$mean.n.disc[1]
  n.disc_StoreySim[j] = resCreditCard0.1v2$compact.results[[j]]$mean.n.disc[3]
  n.disc_WMW[j] = resCreditCard0.1v2$compact.results[[j]]$mean.n.disc[4]

  pow.rejGlob_BH[j] = resCreditCard0.1v2$compact.results[[j]]$mean.powerGlobalNull[1]
```

```
  pow.rejGlob_StoBH[j] = resCreditCard0.1v2$compact.results[[j]]$mean.powerGlobalNull[2]
  pow.rejGlob_Sim[j] = resCreditCard0.1v2$compact.results[[j]]$mean.powerGlobalNull[3]
  pow.rejGlob_StoSimes[j] = resCreditCard0.1v2$compact.results[[j]]$mean.powerGlobalNull[4]
  pow.rejGlob_WMW[j] = resCreditCard0.1v2$compact.results[[j]]$mean.powerGlobalNull[5]

}

lb.d = matrix(nrow = (n+1), ncol = 5)
rownames(lb.d) = as.character(n1s)
colnames(lb.d) = c("FDR-BH", "FDR-Storey", "CT-Simes", "CT-Storey", "CT-WMW")

lb.d[,1] = lb.d_BH
lb.d[,2] = lb.d_StoBH
lb.d[,3] = lb.d_Sim
lb.d[,4] = lb.d_StoSimes
lb.d[,5] = lb.d_WMW

pow.rejGlob = matrix(nrow = (n+1), ncol = 5)
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")
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

n.disc = matrix(nrow = (n+1), ncol = 3)
rownames(n.disc) = as.character(seq(from=0, to=n, by=1))
colnames(n.disc) = c( "CT-Simes", "CT-StoreySimes", "CT-WMW")
n.disc[,1] = n.disc_Sim
n.disc[,2] = n.disc_StoreySim
n.disc[,3] = n.disc_WMW

matrixCredit0.1 = list("lb.d.matrix" = lb.d,
                       "pow.rejGlob.matrix" = pow.rejGlob,
                       "n.disc" = n.disc)
save(matrixCredit0.1, file = paste0(pathResults,"\\matrixCredit0.1"))
```

Covertype

```
data = readMat(paste0(pathDatasets,"\\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), library(hommel))})
clusterExport(cluster, list("n", "m", "l", "in_ind", "out_ind", "dataset", "alpha"))

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))
stopCluster(cluster)

kest

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]

}

resCover0.1 = list("raw.res"=res,
                   "k.est" = kest,
                   "compact.results" = results
                   )
save(resCover0.1, file=paste0(pathResults,"\\resCover0.1"))



# Compacting intermediate results in a matrix
d_BH = vector()
d_StoBH = vector()
d_Sim = vector()
```

```
d_StoSimes = vector()
d_WMW = vector()

pow.rejGlob_BH = vector()
pow.rejGlob_StoBH = vector()
pow.rejGlob_Sim = vector()
pow.rejGlob_StoSimes = vector()
pow.rejGlob_WMW = vector()

for(j in 1:length(n1s)){
  d_BH[j] = resCover0.1$compact.results[[j]]$mean.discoveries[1]
  d_StoBH[j] = resCover0.1$compact.results[[j]]$mean.discoveries[2]
  d_Sim[j] = resCover0.1$compact.results[[j]]$mean.discoveries[3]
  d_StoSimes[j] = resCover0.1$compact.results[[j]]$mean.discoveries[4]
  d_WMW[j] = resCover0.1$compact.results[[j]]$mean.discoveries[5]

  pow.rejGlob_BH[j] = resCover0.1$compact.results[[j]]$mean.powerGlobalNull[1]
  pow.rejGlob_StoBH[j] = resCover0.1$compact.results[[j]]$mean.powerGlobalNull[2]
  pow.rejGlob_Sim[j] = resCover0.1$compact.results[[j]]$mean.powerGlobalNull[3]
  pow.rejGlob_StoSimes[j] = resCover0.1$compact.results[[j]]$mean.powerGlobalNull[4]
  pow.rejGlob_WMW[j] = resCover0.1$compact.results[[j]]$mean.powerGlobalNull[5]
}

lb.d = matrix(nrow = (n+1), ncol = 5)
rownames(lb.d) = as.character(n1s)
colnames(lb.d) = c("FDR-BH", "FDR-Storey", "CT-Simes", "CT-Storey", "CT-WMW")

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

pow.rejGlob = matrix(nrow = (n+1), ncol = 5)
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")
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

matrixCover0.1 = list("lb.d.matrix" = lb.d,
                      "pow.rejGlob.matrix" = pow.rejGlob)
save(matrixCover0.1, file = paste0(pathResults,"\\matrixCover0.1"))
```

ALOI

```
dataset = read.arff(paste0(pathDatasets,"\\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), library(hommel))})
clusterExport(cluster, list("n", "m", "l", "in_ind", "out_ind", "dataset", "alpha"))

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))
stopCluster(cluster)

kest

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()

disc_Sim = vector()
disc_StoSimes = vector()
disc_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]

  disc_Sim[j] = results[[j]]$mean.n.disc[1]
  disc_StoSimes[j] = results[[j]]$mean.n.disc[3]
  disc_WMW[j] = results[[j]]$mean.n.disc[4]
}
```

```
resALOI0.1 = list("raw.res"=res,
                  "k.est" = kest,
                  "compact.results" = results
                  )
save(resALOI0.1, file=paste0(pathResults,"\\resALOI0.1"))
```

Shuttle

```
data = readMat(paste0(pathDatasets,"\\shuttle.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))})
clusterExport(cluster, list("n", "m", "l", "in_ind", "out_ind", "dataset", "alpha"))

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))
stopCluster(cluster)

kest

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()

disc_Sim = vector()
disc_StoSimes = vector()
disc_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]

  disc_Sim[j] = results[[j]]$mean.n.disc[1]
  disc_StoSimes[j] = results[[j]]$mean.n.disc[3]
  disc_WMW[j] = results[[j]]$mean.n.disc[4]
}

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)]
}

resShuttle0.1v2 = list("raw.res"=res,
                       "k.est" = kest,
                       "compact.results" = results,
                       "n.disc.tablelist" = n.disc.tablelist)
save(resShuttle0.1v2,
     file=paste0(pathResults,"\\resShuttle0.1v2"))


# Compacting intermediate results in a matrix
n.disc_Sim = vector()
n.disc_StoreySim = vector()
n.disc_WMW = vector()

pow.rejGlob_BH = vector()
pow.rejGlob_StoBH = vector()
pow.rejGlob_Sim = vector()
pow.rejGlob_StoSimes = vector()
pow.rejGlob_WMW = vector()

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

for(j in 1:length(n1s)){
  lb.d_BH[j] = resShuttle0.1v2$compact.results[[j]]$mean.lb.d[1]
  lb.d_StoBH[j] = resShuttle0.1v2$compact.results[[j]]$mean.lb.d[2]
```

```
  lb.d_Sim[j] = resShuttle0.1v2$compact.results[[j]]$mean.lb.d[3]
  lb.d_StoSimes[j] = resShuttle0.1v2$compact.results[[j]]$mean.lb.d[4]
  lb.d_WMW[j] = resShuttle0.1v2$compact.results[[j]]$mean.lb.d[5]

  n.disc_Sim[j] = resShuttle0.1v2$compact.results[[j]]$mean.n.disc[1]
  n.disc_StoreySim[j] = resShuttle0.1v2$compact.results[[j]]$mean.n.disc[3]
  n.disc_WMW[j] = resShuttle0.1v2$compact.results[[j]]$mean.n.disc[4]

  pow.rejGlob_BH[j] = resShuttle0.1v2$compact.results[[j]]$mean.powerGlobalNull[1]
  pow.rejGlob_StoBH[j] = resShuttle0.1v2$compact.results[[j]]$mean.powerGlobalNull[2]
  pow.rejGlob_Sim[j] = resShuttle0.1v2$compact.results[[j]]$mean.powerGlobalNull[3]
  pow.rejGlob_StoSimes[j] = resShuttle0.1v2$compact.results[[j]]$mean.powerGlobalNull[4]
  pow.rejGlob_WMW[j] = resShuttle0.1v2$compact.results[[j]]$mean.powerGlobalNull[5]
}

lb.d = matrix(nrow = (n+1), ncol = 5)
rownames(lb.d) = as.character(n1s)
colnames(lb.d) = c("FDR-BH", "FDR-Storey", "CT-Simes", "CT-Storey", "CT-WMW")

lb.d[,1] = lb.d_BH
lb.d[,2] = lb.d_StoBH
lb.d[,3] = lb.d_Sim
lb.d[,4] = lb.d_StoSimes
lb.d[,5] = lb.d_WMW

pow.rejGlob = matrix(nrow = (n+1), ncol = 5)
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")
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

n.disc = matrix(nrow = (n+1), ncol = 3)
rownames(n.disc) = as.character(seq(from=0, to=n, by=1))
colnames(n.disc) = c("CT-Simes", "CT-StoreySimes", "CT-WMW")
n.disc[,1] = n.disc_Sim
n.disc[,2] = n.disc_StoreySim
n.disc[,3] = n.disc_WMW

matrixShuttle0.1 = list("lb.d.matrix" = lb.d,
                        "pow.rejGlob.matrix" = pow.rejGlob,
                        "n.disc" = n.disc)
save(matrixShuttle0.1, file = paste0(pathResults,"\\matrixShuttle0.1"))
```

Mammography

```
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)

data = readMat(paste0(pathDatasets,"\\mammography.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))})
clusterExport(cluster, list("n", "m", "l", "in_ind", "out_ind", "dataset", "alpha"))

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))
stopCluster(cluster)

kest

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]

}

resMammo0.1 = list("raw.res"=res,
                   "k.est" = kest,
                   "compact.results" = results)
save(resMammo0.1, file=paste0(pathResults,"\\resMammo0.1"))
```

Figure 2

Load data from Ludbrook and Dudley (1998).

```
# Data from Ludbrook and Dudley (1998)
x = c(5.42, 5.86, 6.16, 6.55, 6.8, 7, 7.11)
y = c(6.51, 7.56, 7.61, 7.84, 11.5)
z = c(x,y)
m = length(x); n = length(y); N = length(z)

set.seed(123)
```

Compute the permutation distribution of the Fisher's, Simes', Wilcoxon-Mann-Whitney and LMPI for $k = 7$ statistics.

```
# How we chose k=7 for the LMPI test statistic
(pr.XlessequalY = mean(replicate(90000, sample(x,1) < sample(y,1))))
```

```
## [1] 0.8853667
```

```
(k = pr.XlessequalY/(1-pr.XlessequalY))
```

```
## [1] 7.723466
```

```
# Fisher's statistic
pstar = sapply(1:n, function(i)
  wilcox.test(x=y[i],y=x,
              exact=T,
              alternative="greater")$p.value
)

# T*_Fisher
fisherstar = -2*sum(log(pstar))
1-pchisq(fisherstar, df=2*n)
```

```
## [1] 0.06255995
```

```
# tilde T*_Fisher
gamma = n/m
(fisherstaradj = (fisherstar + 2*(sqrt(1+gamma)-1)*n )/(sqrt(1+gamma)))
```

```
## [1] 15.78591
```

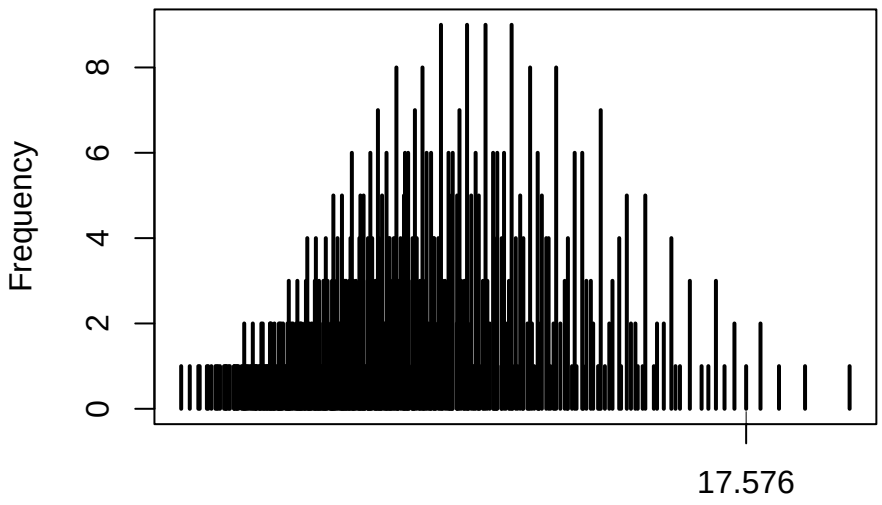
```
1-pchisq(fisherstaradj, df=2*n)
```

```
## [1] 0.1059271
```

```
# Permutation distributions
perms = combn(1:N,m)
B = ncol(perms) # choose(N,m)

fisherstat <- vector()
simesstat <- vector()
WMWstat <- vector()
rank7stat <- vector()
Pmat<-matrix(NA, ncol=5, nrow=B)

for (b in 1:B){
  Zperm = c(z[perms[,b]], z[-perms[,b]])
  R = rank(Zperm) # ranks in the pooled vector

 Pmat[b,] = sapply(1:n, function(i)
   wilcox.test(x=Zperm[m+i],y=Zperm[1:m], # x = test sample, y = control sample
               exact=T,
               alternative="greater")$p.value)

fisherstat[b] = -2*sum(log(Pmat[b,]))

simesstat[b] = min(sort(Pmat[b,])*n/(1:n))

WMWstat[b] = sum(R[(m+1):(m+n)])


rank7stat[b] = sum((R[(m+1):(m+n)])*(R[(m+1):(m+n)] + 1)*
                     (R[(m+1):(m+n)] + 2)*(R[(m+1):(m+n)] + 3)*
                     (R[(m+1):(m+n)] + 4)*(R[(m+1):(m+n)] + 5)#*
#                     (R[(m+1):(m+n)] + 6)
                   )
}

# Fisher's statistic
plot(table(fisherstat),
     xlab = "Fisher's statistic",
     ylab = "Frequency", xaxt="n")
rug(fisherstat[1])
mean( fisherstat >= fisherstat[1] )
```

```
## [1] 0.007575758
```

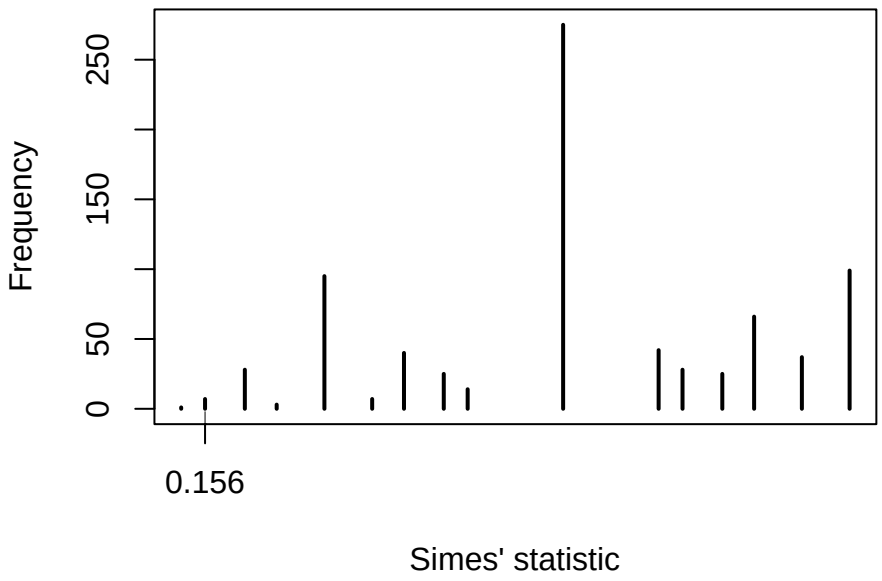
```
axis(side=1, at=fisherstat[1], round(fisherstat[1],3) )
```




```
# Simes' statistic
plot(table(simesstat),
     xlab = "Simes' statistic",
     ylab = "Frequency", xaxt="n")
rug(simesstat[1])
mean( simesstat <= simesstat[1] )
```

```
## [1] 0.01010101
```

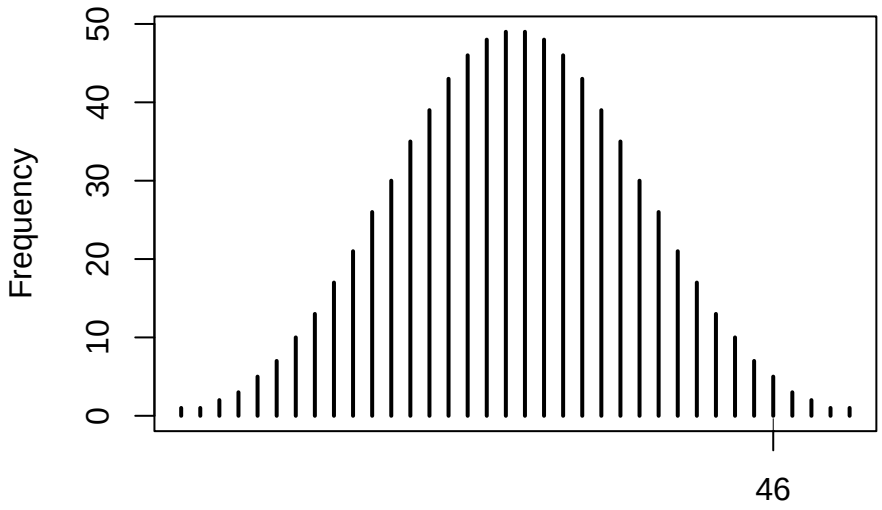
```
axis(side=1, at=simesstat[1], round(simesstat[1],3) )
```




```
# Wilcoxon-Mann-Whitney statistic
plot(table(WMWstat),
     xlab = "Wilcoxon rank sum statistic",
     ylab = "Frequency", xaxt="n")
rug(WMWstat[1])
mean( WMWstat >= WMWstat[1] )
```

```
## [1] 0.01515152
```

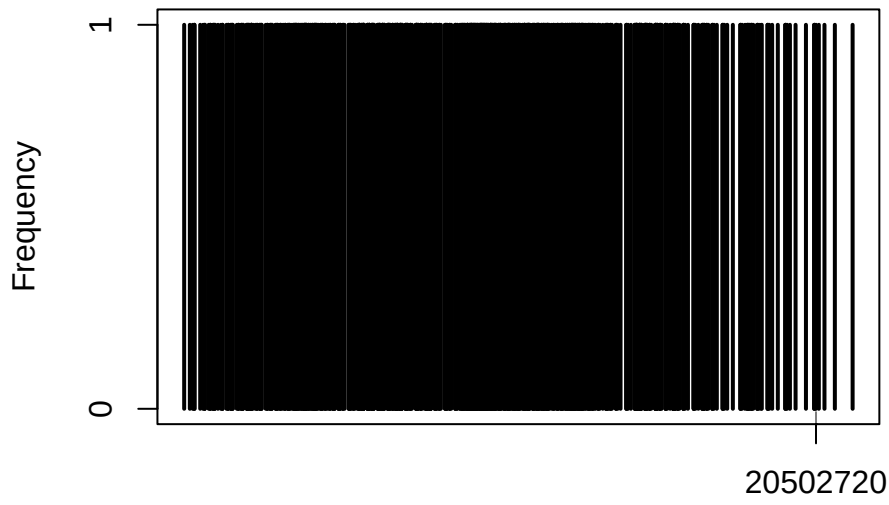
```
axis(side=1, at=WMWstat[1], round(WMWstat[1],3) )
```




```
# LMPI statistic for k=7
plot(table(rank7stat),
     xlab = "LMPI rank statistic for Lehmann's alternative with k=7",
     ylab = "Frequency", xaxt="n",
     yaxt="n")
rug(rank7stat[1])
mean( rank7stat>= rank7stat[1] )
```

```
## [1] 0.006313131
```

```
axis(side=1, at=rank7stat[1], round(rank7stat[1],3) )
axis(2,c(0,1), c(0,1) )
```



LMPI rank statistic for Lehmann's alternative with k=7

Table 2

Dunnett one-sided single step and step-down

```
group = as.factor(c(rep(0,length(x)),1:length(y)))
dat=data.frame(z,group)
fit <- aov(z ~ group, data = dat)

dun_onesided <- glht(fit,
                     linfct = mcp(group = "Dunnett"),
                     alternative = "greater")

# Single step Dunnett test
summary(dun_onesided)
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Dunnett Contrasts
##
##
## Fit: aov(formula = z ~ group, data = dat)
##
## Linear Hypotheses:
##            Estimate Std. Error t value Pr(>t)
## 1 - 0 <= 0  0.09571    0.66972   0.143  0.912
```

```
## 2 - 0 <= 0  1.14571    0.66972   1.711  0.258
## 3 - 0 <= 0  1.19571    0.66972   1.785  0.236
## 4 - 0 <= 0  1.42571    0.66972   2.129  0.153
## 5 - 0 <= 0  5.08571    0.66972   7.594 <0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

```
# Step-down Dunnett test procedure
(sum.free = summary(dun_onesided, test = adjusted(type = "free")))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Dunnett Contrasts
##
##
## Fit: aov(formula = z ~ group, data = dat)
##
## Linear Hypotheses:
##            Estimate Std. Error t value   Pr(>t)
## 1 - 0 <= 0  0.09571    0.66972   0.143 0.445517
## 2 - 0 <= 0  1.14571    0.66972   1.711 0.160832
## 3 - 0 <= 0  1.19571    0.66972   1.785 0.160832
## 4 - 0 <= 0  1.42571    0.66972   2.129 0.128887
## 5 - 0 <= 0  5.08571    0.66972   7.594 0.000834 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- free method)
```

```
# d = |D|
sum(sum.free$test$pvalues<=0.1)
```

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

BH*

```
pstar = sapply(1:n, function(i)
  wilcox.test(x=y[i],y=x,
              exact=T,
              alternative="greater")$p.value
)
# or equivalently
# pstar = sapply(1:n, function(i) (1+sum(x>=y[i]))/(m+1))
p.adjust(pstar,"BH")
```

```
## [1] 0.62500 0.15625 0.15625 0.15625 0.15625
```

```r
# |D|
sum(p.adjust(pstar,"BH")<=0.1)
```

```
## [1] 0
```

```r
# d = |D|
nout::d_benjhoch(S_X = x, S_Y = y, alpha = 0.1)
```

```
## [1] 0
```

BH

```r
# Unadjusted "parametric" p-values from t-test
(sum.none = summary(dun_onesided, test = adjusted(type = "none")))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Dunnett Contrasts
##
##
## Fit: aov(formula = z ~ group, data = dat)
##
## Linear Hypotheses:
##            Estimate Std. Error t value   Pr(>t)
## 1 - 0 <= 0  0.09571    0.66972   0.143 0.445517
## 2 - 0 <= 0  1.14571    0.66972   1.711 0.068990 .
## 3 - 0 <= 0  1.19571    0.66972   1.785 0.062222 .
## 4 - 0 <= 0  1.42571    0.66972   2.129 0.038668 *
## 5 - 0 <= 0  5.08571    0.66972   7.594 0.000136 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- none method)
```

```r
(unadjusted.pvals = round(sum.none$test$pvalues, 3))
```

```
## 1 - 0 2 - 0 3 - 0 4 - 0 5 - 0
## 0.446 0.069 0.062 0.039 0.000
```

```r
(t.pvals = round(p.adjust(unadjusted.pvals,"BH"),3))
```

```
## 1 - 0 2 - 0 3 - 0 4 - 0 5 - 0
## 0.446 0.086 0.086 0.086 0.000
```

```r
# d = |D|
sum(t.pvals<=0.1)
```

```
## [1] 4
```

Simes closed testing

```
pstar = sapply(1:n, function(i)
  wilcox.test(x=y[i],y=x,
              exact=T,
              alternative="greater")$p.value
)
# or equivalently
# pstar = sapply(1:n, function(i) (1+sum(x>=y[i]))/(m+1))

p.adjust(pstar,"hommel")
```

```
## [1] 0.625 0.250 0.250 0.250 0.250
```

```
# or equivalently
# hom <- hommel(pstar, simes = TRUE)
# hom@adjusted

# d
nout::d_Simes(S_X = x, S_Y = y, alpha = 0.1)
```

```
## [1] 0
```

```
# |D|
dselection_Simes(S_X = x, S_Y = y, S = 1, alpha = 0.1)+
dselection_Simes(S_X = x, S_Y = y, S = 2, alpha = 0.1)+
dselection_Simes(S_X = x, S_Y = y, S = 3, alpha = 0.1)+
dselection_Simes(S_X = x, S_Y = y, S = 4, alpha = 0.1)+
dselection_Simes(S_X = x, S_Y = y, S = 5, alpha = 0.1)
```

```
## [1] 0
```

Wilcoxon-Mann-Whitney closed testing

```
(pstar = sapply(1:n, function(i) (1+sum(x>=y[i]))/(m+1)))
```

```
## [1] 0.625 0.125 0.125 0.125 0.125
```

```
# d
nout::d_MannWhitney(S_Y=y, S_X=x, alpha=0.1)
```

```
## [1] 3
```

```
# |D|
dselection.singleton_MannWhitney(S_Y=y, S=1, S_X=x, alpha=0.1)+
dselection.singleton_MannWhitney(S_Y=y, S=2, S_X=x, alpha=0.1)+
dselection.singleton_MannWhitney(S_Y=y, S=3, S_X=x, alpha=0.1)+
dselection.singleton_MannWhitney(S_Y=y, S=4, S_X=x, alpha=0.1)+
dselection.singleton_MannWhitney(S_Y=y, S=5, S_X=x, alpha=0.1)
```

```
## [1] 0
```

Figure 3

Intermediate results for Figure 3

Since training the one-class classification model and using it to make predictions on the calibration and test samples to get the conformity scores is time consuming, intermediate results are loaded from the workspace. The R code in this section produces intermediate results for ***Pen-digits*** dataset and is saved as an Rdata file. Then, intermediate results are compacted into a matrix.

First we initialize the parameters using the same notation as in the paper.

```
set.seed(321)

# Initializing parameters
B = 10^5
m = 199
l = 199
n = 20
alpha = n/(l+1)
n1s = seq(from=0, to=n, by=1)
```

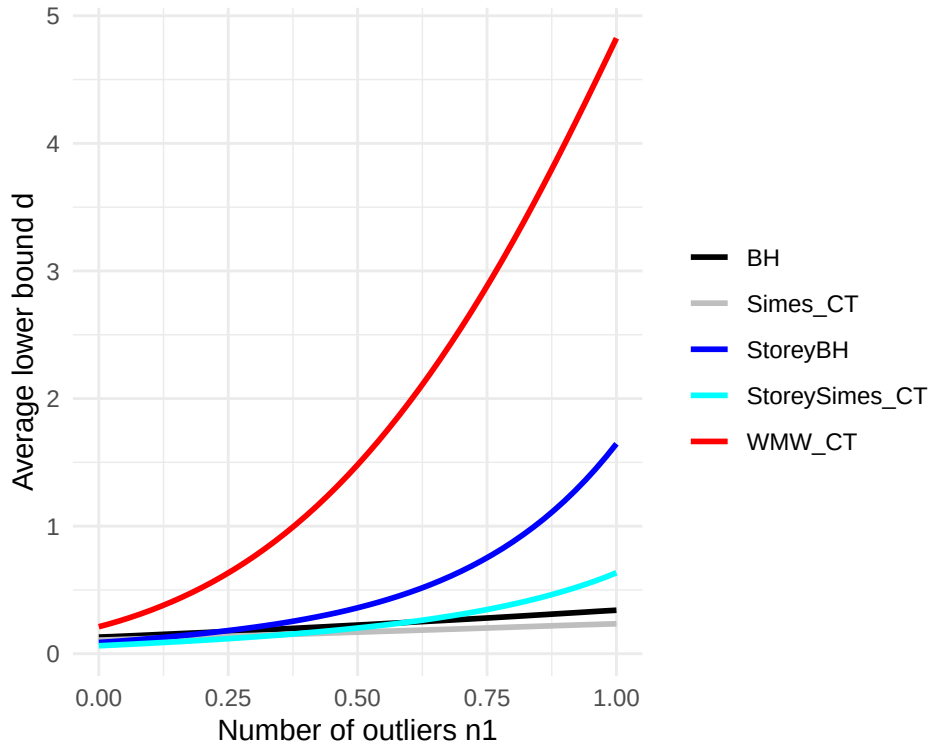
```
data = readMat(paste0(pathDatasets,"\\pendigits.mat"))
dataset = cbind(data$X, data$y); colnames(dataset)[ncol(dataset)] = "y"
in_ind = which(dataset[,ncol(dataset)]==0)
out_ind = which(dataset[,ncol(dataset)]==1)
theta = length(out_ind)/nrow(dataset) # proportion of outliers in the entire dataset

cluster <- makeCluster(parallel::detectCores())
registerDoSNOW(cluster)
clusterEvalQ(cluster, {list(library(isotree), library(nout))})
clusterExport(cluster, list("n", "m", "l", "in_ind", "out_ind", "dataset", "alpha"))

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))
stopCluster(cluster)

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]

}

# 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("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))
print(unconditional.power)

resDigits0.1k2 = list("raw.res"=res,
                      "unconditional.power" = unconditional.power,
                      "compact.results" = results)
save(resDigits0.1k2, file=paste0(pathResults,"\\resDigits0.1k2"))



# Compacting intermediate results in a matrix

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

pow.rejGlob_BH = vector()
pow.rejGlob_StoBH = vector()
pow.rejGlob_Sim = vector()
pow.rejGlob_StoSimes = vector()
pow.rejGlob_WMW = vector()

for(j in 1:length(n1s)){
  d_BH[j] = resDigits0.1k2$compact.results[[j]]$mean.discoveries[1]
  d_StoBH[j] = resDigits0.1k2$compact.results[[j]]$mean.discoveries[2]
  d_Sim[j] = resDigits0.1k2$compact.results[[j]]$mean.discoveries[3]
  d_StoSimes[j] = resDigits0.1k2$compact.results[[j]]$mean.discoveries[4]
```

```
  d_WMW[j] = resDigits0.1k2$compact.results[[j]]$mean.discoveries[5]

  pow.rejGlob_BH[j] = resDigits0.1k2$compact.results[[j]]$mean.powerGlobalNull[1]
  pow.rejGlob_StoBH[j] = resDigits0.1k2$compact.results[[j]]$mean.powerGlobalNull[2]
  pow.rejGlob_Sim[j] = resDigits0.1k2$compact.results[[j]]$mean.powerGlobalNull[3]
  pow.rejGlob_StoSimes[j] = resDigits0.1k2$compact.results[[j]]$mean.powerGlobalNull[4]
  pow.rejGlob_WMW[j] = resDigits0.1k2$compact.results[[j]]$mean.powerGlobalNull[5]

}

lb.d = matrix(nrow = (n+1), ncol = 5)
rownames(lb.d) = as.character(n1s)
colnames(lb.d) = c("FDR-BH", "FDR-Storey", "CT-Simes", "CT-Storey", "CT-WMW")

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

pow.rejGlob = matrix(nrow = (n+1), ncol = 5)
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")
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

matrixDigits0.1k2 = list("lb.d.matrix" = lb.d,
                         "pow.rejGlob.matrix" = pow.rejGlob)
save(matrixDigits0.1k2, file = paste0(pathResults,"\\matrixDigits0.1k2"))
```

```
load(paste0(pathResults,"\\matrixDigits0.1k2"))
res = matrixDigits0.1k2

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)

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)
```

```r
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)


# 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
)
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", "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
)
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","red")) +
  labs(x = "Number of outliers n1", y = "Power to reject the global null") +
  theme_minimal() +
  theme(legend.title = element_blank())
```

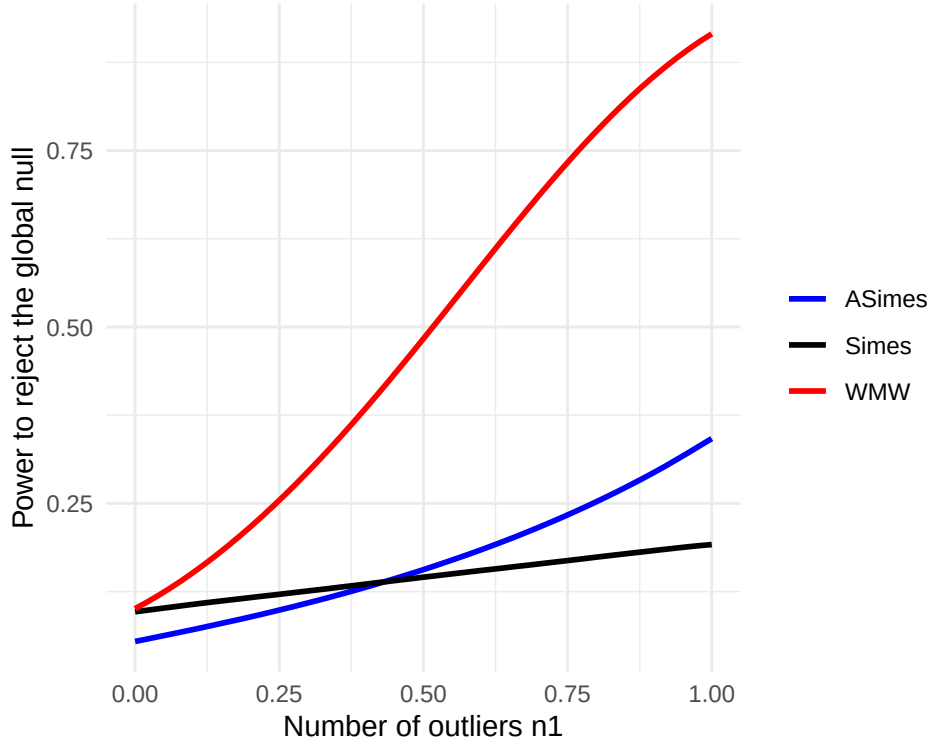
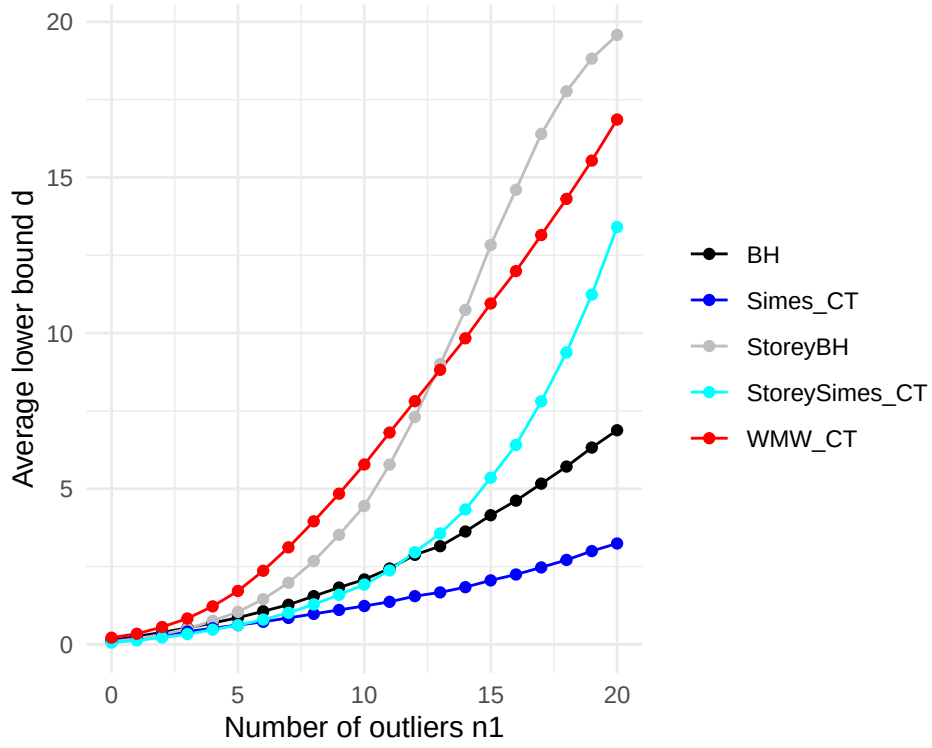


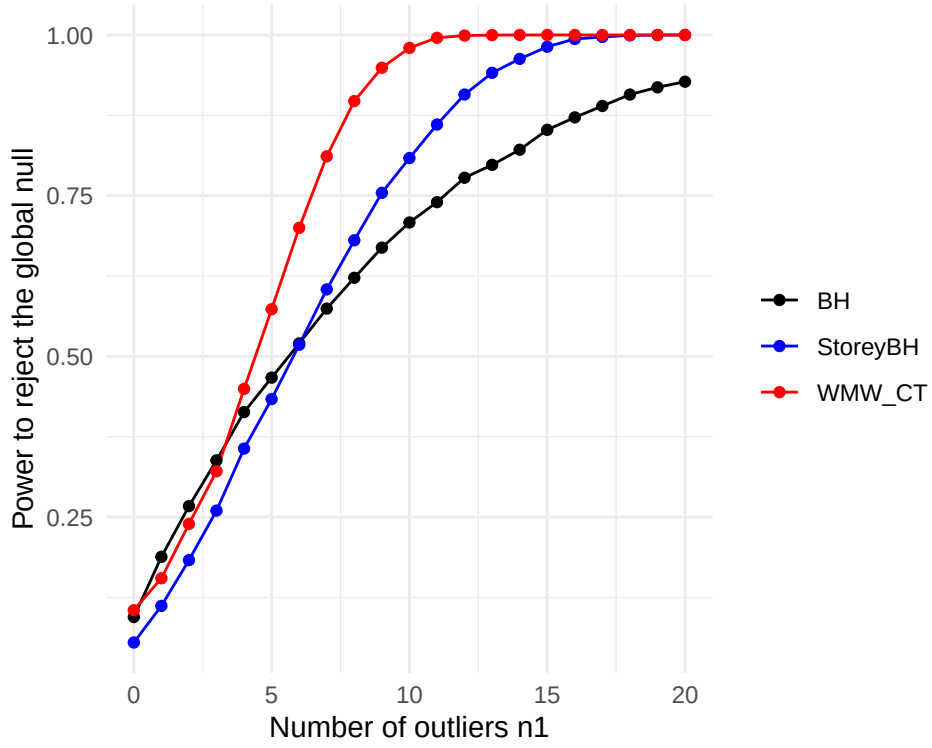
Figure 4

```r
# Pen-digits

load(paste0(pathResults,"\\resDigits0.1"))
results = resDigits0.1$compact.results
n1s = seq(from = 0, to = n, by=1)

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()

disc_Sim = vector()
disc_StoSimes = vector()
disc_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]

  disc_Sim[j] = results[[j]]$mean.n.disc[1]
  disc_StoSimes[j] = results[[j]]$mean.n.disc[3]
  disc_WMW[j] = results[[j]]$mean.n.disc[4]
}

# Plot lower bound d
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("black", "blue", "gray", "cyan", "red")) +
  labs(x = "Number of outliers n1", y = "Average lower bound d") +
  theme_minimal() +
  theme(legend.title = element_blank())
```

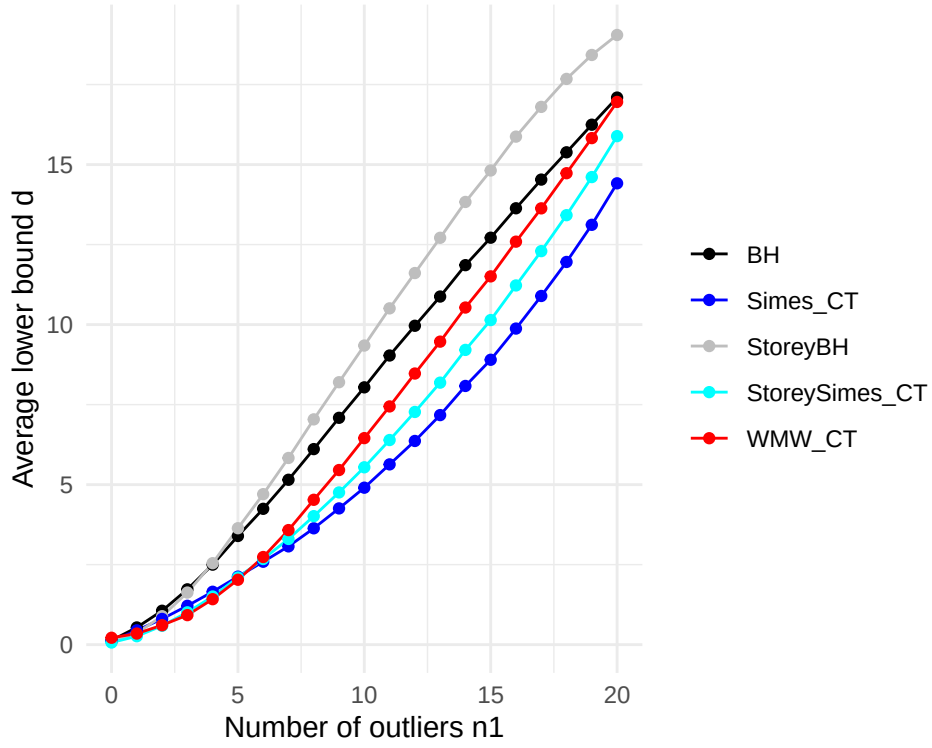



```
# Plot power
dfpower <- data.frame(
  x = n1s,
  BH = pow_BH,
  StoreyBH = pow_StoBH,
  WMW_CT = pow_WMW
)
df_long_power <- tidyr::pivot_longer(dfpower, cols = -x, names_to = "group", values_to = "y")

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

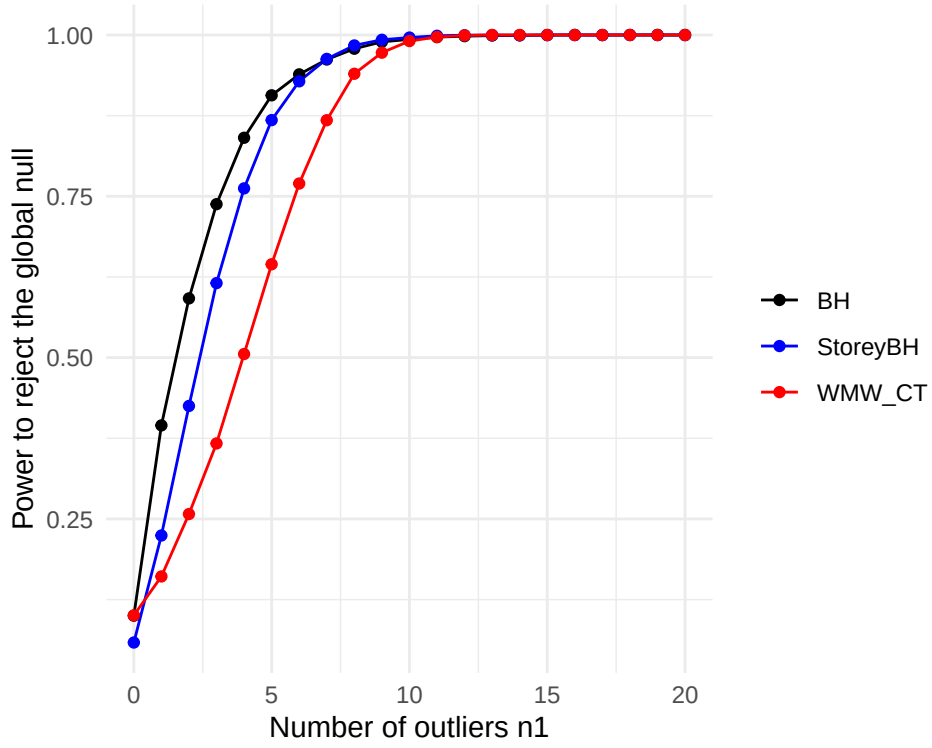



```
# Credit Card

load(paste0(pathResults,"\\resCreditCard0.1v2"))
results = resCreditCard0.1v2$compact.results
n1s = seq(from = 0, to = n, by=1)

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()

disc_Sim = vector()
disc_StoSimes = vector()
disc_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]

  disc_Sim[j] = results[[j]]$mean.n.disc[1]
  disc_StoSimes[j] = results[[j]]$mean.n.disc[3]
  disc_WMW[j] = results[[j]]$mean.n.disc[4]
}

# Plot lower bound d
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("black", "blue", "gray", "cyan", "red")) +
  labs(x = "Number of outliers n1", y = "Average lower bound d") +
  theme_minimal() +
  theme(legend.title = element_blank())
```

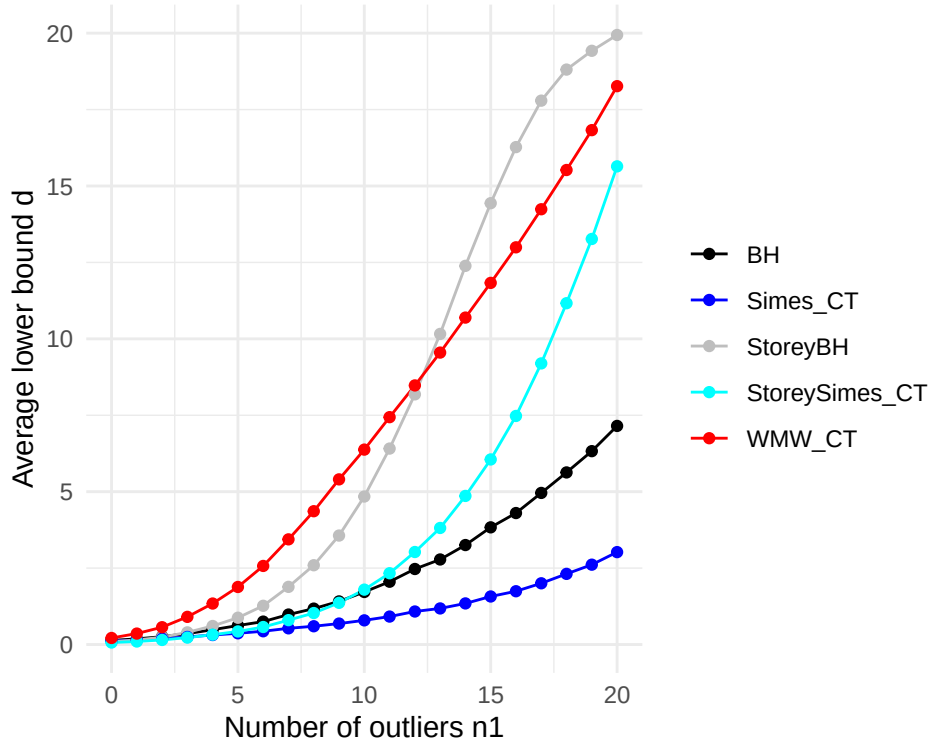



```
# Plot power
dfpower <- data.frame(
  x = n1s,
  BH = pow_BH,
  StoreyBH = pow_StoBH,
  WMW_CT = pow_WMW
)
df_long_power <- tidyr::pivot_longer(dfpower, cols = -x, names_to = "group", values_to = "y")

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




```
# Cover

load(paste0(pathResults,"\\resCover0.1"))
results = resCover0.1$compact.results
n1s = seq(from = 0, to = n, by=1)

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()

disc_Sim = vector()
disc_StoSimes = vector()
disc_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]

  disc_Sim[j] = results[[j]]$mean.n.disc[1]
  disc_StoSimes[j] = results[[j]]$mean.n.disc[3]
  disc_WMW[j] = results[[j]]$mean.n.disc[4]
}

# Plot lower bound d
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("black", "blue", "gray", "cyan", "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 = 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")

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

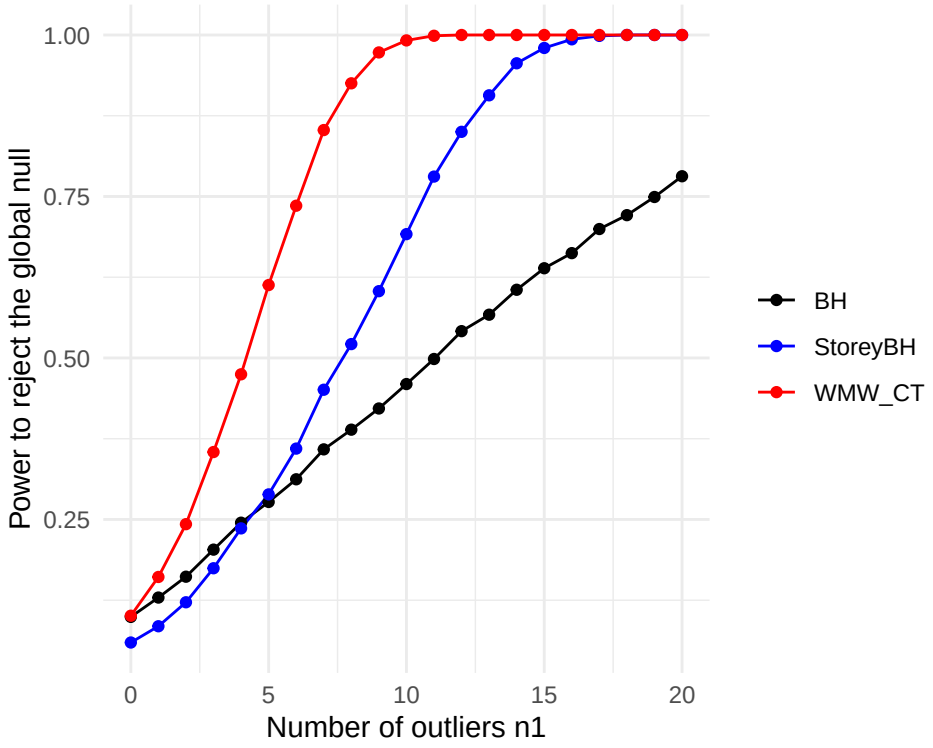
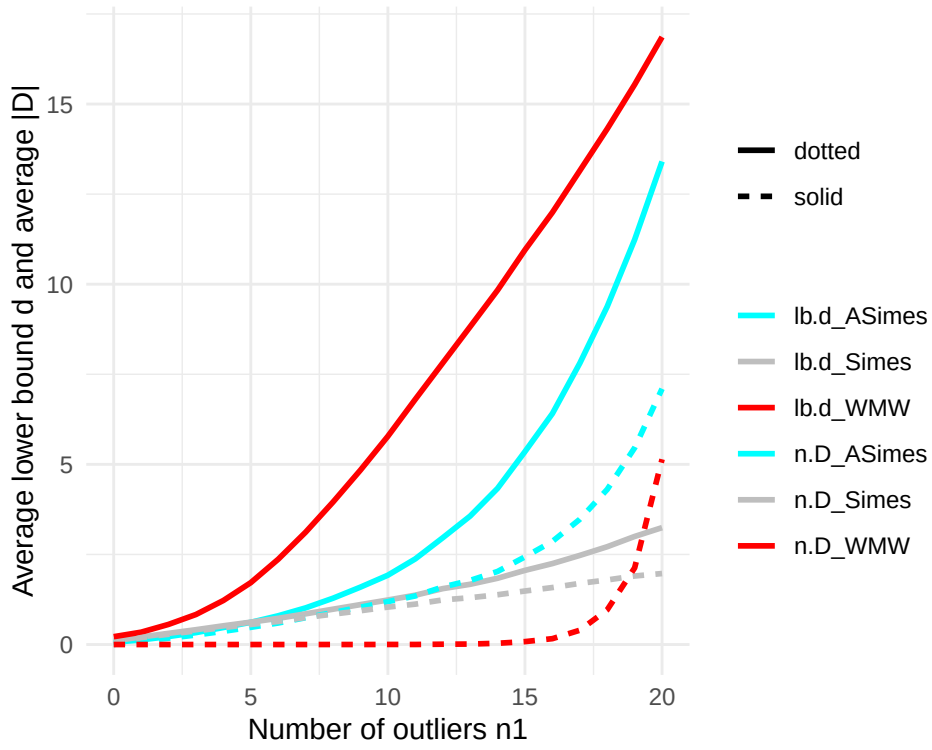


Figure 5

```
# Pen-digits
load(paste0(pathResults,"\\matrixDigits0.1"))

df <- data.frame(
  x = n1s,
  lb.d_Simes = matrixDigits0.1$lb.d.matrix[,3],
  lb.d_ASimes = matrixDigits0.1$lb.d.matrix[,4],
  lb.d_WMW = matrixDigits0.1$lb.d.matrix[,5],
  n.D_Simes = matrixDigits0.1$n.disc[,1],
  n.D_ASimes = matrixDigits0.1$n.disc[,2],
  n.D_WMW = matrixDigits0.1$n.disc[,3]
)
df_long <- tidyr::pivot_longer(df, cols = -x, names_to = "group", values_to = "y")

lineType <- function(group){
  if(startsWith(group, "n.D")) return("solid")
  else return("dotted")
}

ggplot(df_long, aes(x = x, y = y, color = group, linetype = sapply(group, lineType))) +
  geom_line(size=1) +
  scale_color_manual(values = c("cyan","gray", "red","cyan", "gray", "red")) +
  labs(x = "Number of outliers n1", y = "Average lower bound d and average |D|") +
  theme_minimal() +
  theme(legend.title = element_blank())
```




```
# Credit Card

load(paste0(pathResults,"\\matrixCredit0.1"))

df <- data.frame(
  x = n1s,
  lb.d_Simes = matrixCredit0.1$lb.d.matrix[,3],
  lb.d_ASimes = matrixCredit0.1$lb.d.matrix[,4],
  lb.d_WMW = matrixCredit0.1$lb.d.matrix[,5],
  n.D_Simes = matrixCredit0.1$n.disc[,1],
  n.D_ASimes = matrixCredit0.1$n.disc[,2],
  n.D_WMW = matrixCredit0.1$n.disc[,3]
)
df_long <- tidyr::pivot_longer(df, cols = -x, names_to = "group", values_to = "y")

lineType <- function(group){
  if(startsWith(group, "n.D")) return("solid")
  else return("dotted")
}

ggplot(df_long, aes(x = x, y = y, color = group, linetype = sapply(group, lineType))) +
  geom_line(size=1) +
  scale_color_manual(values = c("cyan","gray", "red","cyan", "gray", "red")) +
  labs(x = "Number of outliers n1", y = "Average lower bound d and average |D|") +
  theme_minimal() +
  theme(legend.title = element_blank())
```

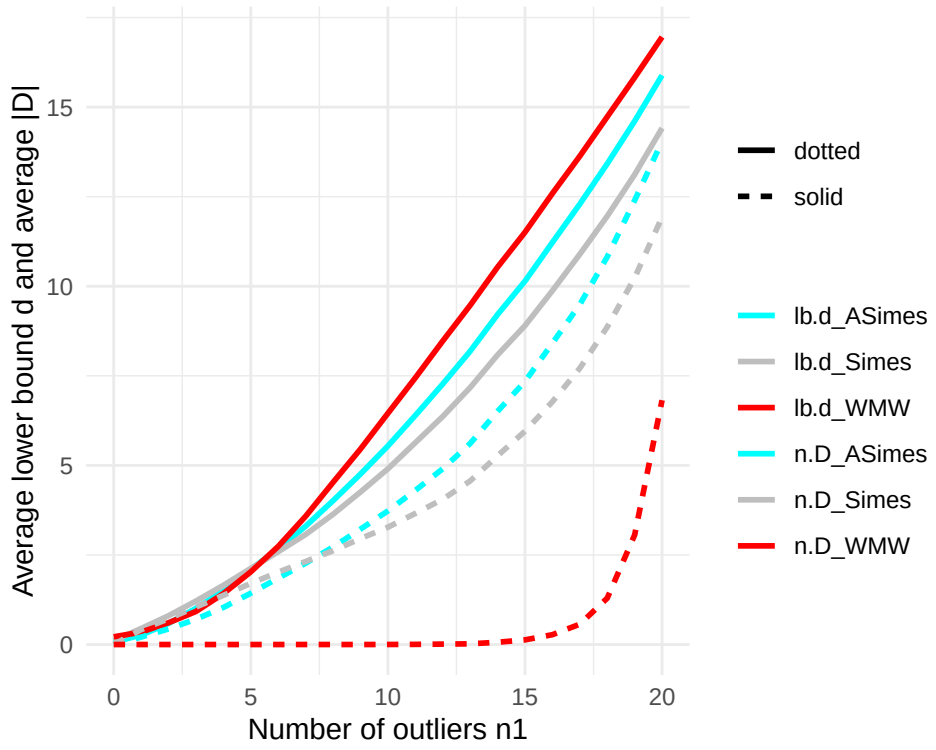


Table 3

Pen-digits

```
data = readMat(paste0(pathDatasets,"\\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)
load(paste0(pathResults,"\\matrixDigits0.1"))
load(paste0(pathResults,"\\resDigits0.1"))

hat.k = resDigits0.1$k.est

theta = length(out_ind)/nrow(dataset)

probsn1 = sapply(1:n, function(k) (choose(n,k)*(1-theta)^(n-k)*theta^k)/(1-(1-theta)^n))

power.n1pos = cbind(
  "power.n1pos_BH" =
    round(sum(matrixDigits0.1$pow.rejGlob.matrix[-1,"FDR-BH"]*probsn1),4),
  "power.n1pos_StoreyBH" =
    round(sum(matrixDigits0.1$pow.rejGlob.matrix[-1,"FDR-Storey"]*probsn1),4),
  "power.n1pos_WMW" =
    round(sum(matrixDigits0.1$pow.rejGlob.matrix[-1,"CT-WMW"]*probsn1),4)
  )

d.n1pos = cbind(
```

```
  "d.n1pos_BH" =
    round(sum(matrixDigits0.1$lb.d.matrix[-1,"FDR-BH"]*probsn1),4),
  "d.n1pos_StoreyBH" =
    round(sum(matrixDigits0.1$lb.d.matrix[-1,"FDR-Storey"]*probsn1),4),
  "d.n1pos_Simes" =
    round(sum(matrixDigits0.1$lb.d.matrix[-1,"CT-Simes"]*probsn1),4),
  "d.n1pos_CTStorey" =
    round(sum(matrixDigits0.1$lb.d.matrix[-1,"CT-Storey"]*probsn1),4),
  "d.n1pos_WMW" =
    round(sum(matrixDigits0.1$lb.d.matrix[-1,"CT-WMW"]*probsn1),4)
  )

theta
```

```
## [1] 0.02270742
```

```
hat.k
```

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

```
power.n1pos
```

```
##      power.n1pos_BH power.n1pos_StoreyBH power.n1pos_WMW
## [1,]         0.2063               0.1286          0.1745
```

```
d.n1pos
```

```
##      d.n1pos_BH d.n1pos_StoreyBH d.n1pos_Simes d.n1pos_CTStorey d.n1pos_WMW
## [1,]     0.2858            0.217        0.2331           0.1508      0.3949
```

Credit Card

```
dataset = read_csv(paste0(pathDatasets,"\\creditcard.csv"))
in_ind = which(dataset[,ncol(dataset)]==0)
out_ind = which(dataset[,ncol(dataset)]==1)
load(paste0(pathResults,"\\matrixCredit0.1"))
load(paste0(pathResults,"\\resCreditCard0.1v2"))

hat.k = resCreditCard0.1v2$k.est

theta = length(out_ind)/nrow(dataset)

probsn1 = sapply(1:n, function(k) (choose(n,k)*(1-theta)^(n-k)*theta^k)/(1-(1-theta)^n))

power.n1pos = cbind(
  "power.n1pos_BH" =
    round(sum(matrixCredit0.1$pow.rejGlob.matrix[-1,"FDR-BH"]*probsn1),4),
  "power.n1pos_StoreyBH" =
    round(sum(matrixCredit0.1$pow.rejGlob.matrix[-1,"FDR-Storey"]*probsn1),4),
```

```
  "power.n1pos_WMW" =
    round(sum(matrixCredit0.1$pow.rejGlob.matrix[-1,"CT-WMW"]*probsn1),4)
  )

d.n1pos = cbind(
  "d.n1pos_BH" =
    round(sum(matrixCredit0.1$lb.d.matrix[-1,"FDR-BH"]*probsn1),4),
  "d.n1pos_StoreyBH" =
    round(sum(matrixCredit0.1$lb.d.matrix[-1,"FDR-Storey"]*probsn1),4),
  "d.n1pos_Simes" =
    round(sum(matrixCredit0.1$lb.d.matrix[-1,"CT-Simes"]*probsn1),4),
  "d.n1pos_CTStorey" =
    round(sum(matrixCredit0.1$lb.d.matrix[-1,"CT-Storey"]*probsn1),4),
  "d.n1pos_WMW" =
    round(sum(matrixCredit0.1$lb.d.matrix[-1,"CT-WMW"]*probsn1),4)
  )

theta
```

```
## [1] 0.001727486
```

```
hat.k
```

```
## [1] 14.03759
```

```
power.n1pos
```

```
##      power.n1pos_BH power.n1pos_StoreyBH power.n1pos_WMW
## [1,]         0.3982               0.2277          0.1625
```

```
d.n1pos
```

```
##      d.n1pos_BH d.n1pos_StoreyBH d.n1pos_Simes d.n1pos_CTStorey d.n1pos_WMW
## [1,]     0.5462           0.3711        0.4522           0.2687      0.3513
```

Covertype

```
data = readMat(paste0(pathDatasets,"\\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)
load(paste0(pathResults,"\\matrixCover0.1"))
load(paste0(pathResults,"\\resCover0.1"))

hat.k = resCover0.1$k.est

theta = length(out_ind)/nrow(dataset)

probsn1 = sapply(1:n, function(k) (choose(n,k)*(1-theta)^(n-k)*theta^k)/(1-(1-theta)^n))
```

```
power.n1pos = cbind(
  "power.n1pos_BH" =
    round(sum(matrixCover0.1$pow.rejGlob.matrix[-1,"FDR-BH"]*probsn1),4),
  "power.n1pos_StoreyBH" =
    round(sum(matrixCover0.1$pow.rejGlob.matrix[-1,"FDR-Storey"]*probsn1),4),
  "power.n1pos_WMW" =
    round(sum(matrixCover0.1$pow.rejGlob.matrix[-1,"CT-WMW"]*probsn1),4)
  )

d.n1pos = cbind(
  "d.n1pos_BH" =
    round(sum(matrixCover0.1$lb.d.matrix[-1,"FDR-BH"]*probsn1),4),
  "d.n1pos_StoreyBH" =
    round(sum(matrixCover0.1$lb.d.matrix[-1,"FDR-Storey"]*probsn1),4),
  "d.n1pos_Simes" =
    round(sum(matrixCover0.1$lb.d.matrix[-1,"CT-Simes"]*probsn1),4),
  "d.n1pos_CTStorey" =
    round(sum(matrixCover0.1$lb.d.matrix[-1,"CT-Storey"]*probsn1),4),
  "d.n1pos_WMW" =
    round(sum(matrixCover0.1$lb.d.matrix[-1,"CT-WMW"]*probsn1),4)
  )

theta
```

```
## [1] 0.009603283
```

```
hat.k
```

```
## [1] 10.89061
```

```
power.n1pos
```

```
##      power.n1pos_BH power.n1pos_StoreyBH power.n1pos_WMW
## [1,]         0.1342               0.0898          0.1684
```

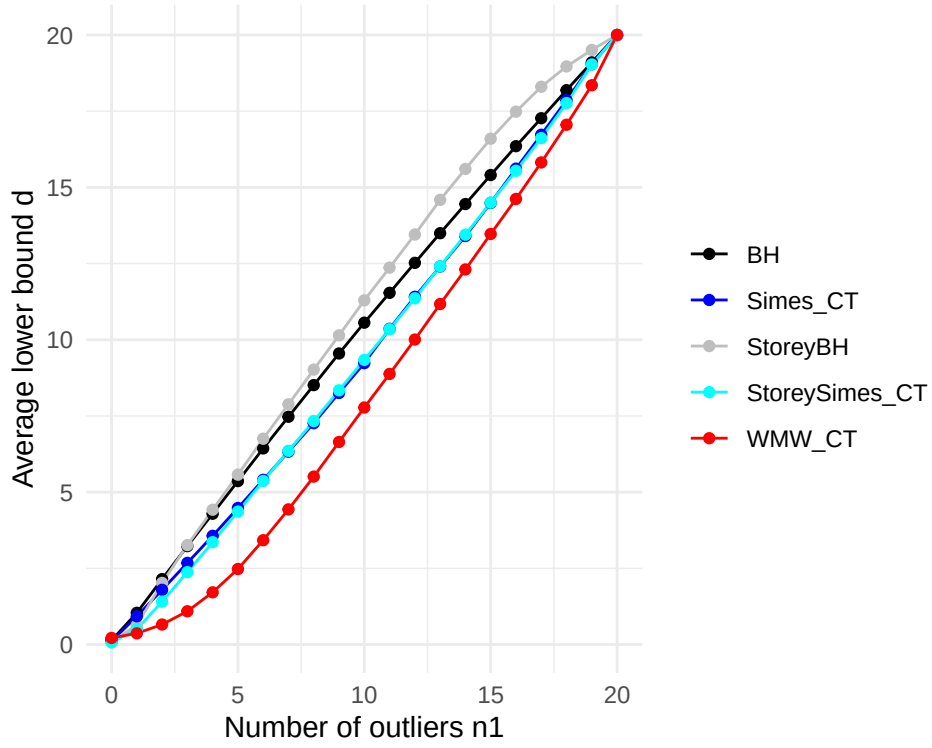
```
d.n1pos
```

```
##      d.n1pos_BH d.n1pos_StoreyBH d.n1pos_Simes d.n1pos_CTStorey d.n1pos_WMW
## [1,]     0.1948           0.1586        0.1528           0.1057      0.3804
```

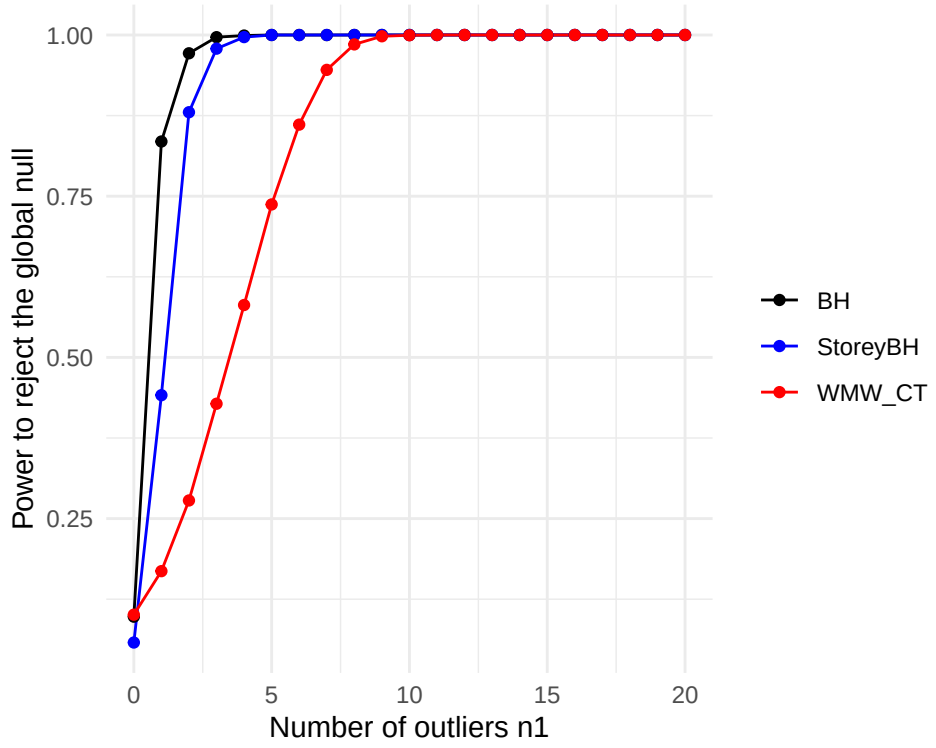
Figure 6

```
# Shuttle

load(paste0(pathResults,"\\resShuttle0.1v2"))
results = resShuttle0.1v2$compact.results
n1s = seq(from = 0, to = n, by=1)

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()

disc_Sim = vector()
disc_StoSimes = vector()
disc_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]

  disc_Sim[j] = results[[j]]$mean.n.disc[1]
  disc_StoSimes[j] = results[[j]]$mean.n.disc[3]
  disc_WMW[j] = results[[j]]$mean.n.disc[4]
}

# Plot lower bound d
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("black", "blue", "gray", "cyan", "red")) +
  labs(x = "Number of outliers n1", y = "Average lower bound d") +
  theme_minimal() +
  theme(legend.title = element_blank())
```

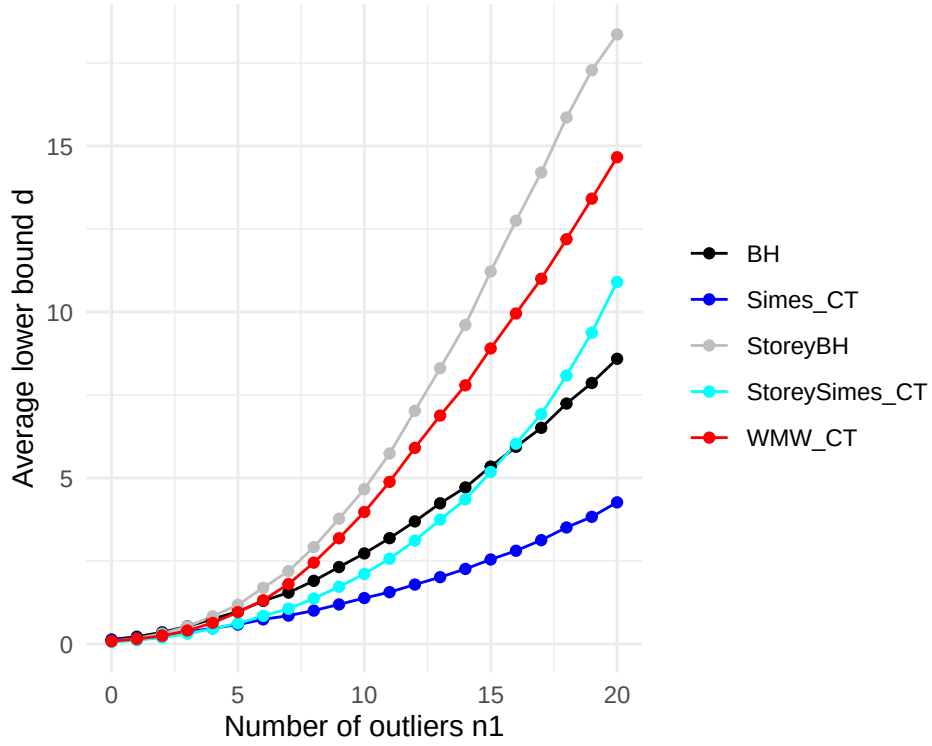



```r
# Plot power
dfpower <- data.frame(
  x = n1s,
  BH = pow_BH,
  StoreyBH = pow_StoBH,
  WMW_CT = pow_WMW
)
df_long_power <- tidyr::pivot_longer(dfpower, cols = -x, names_to = "group", values_to = "y")

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



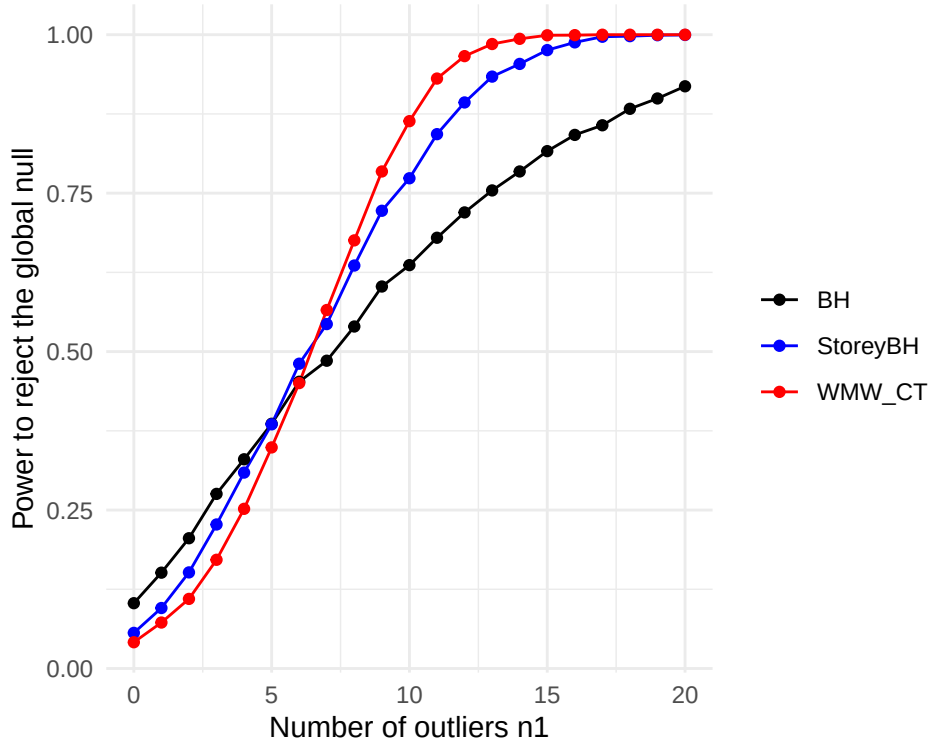

```
# Mammography

load(paste0(pathResults,"\\resMammo0.1"))
results = resMammo0.1$compact.results
n1s = seq(from = 0, to = n, by=1)

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 lower bound d
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("black", "blue", "gray", "cyan", "red")) +
  labs(x = "Number of outliers n1", y = "Average lower bound d") +
  theme_minimal() +
  theme(legend.title = element_blank())
```



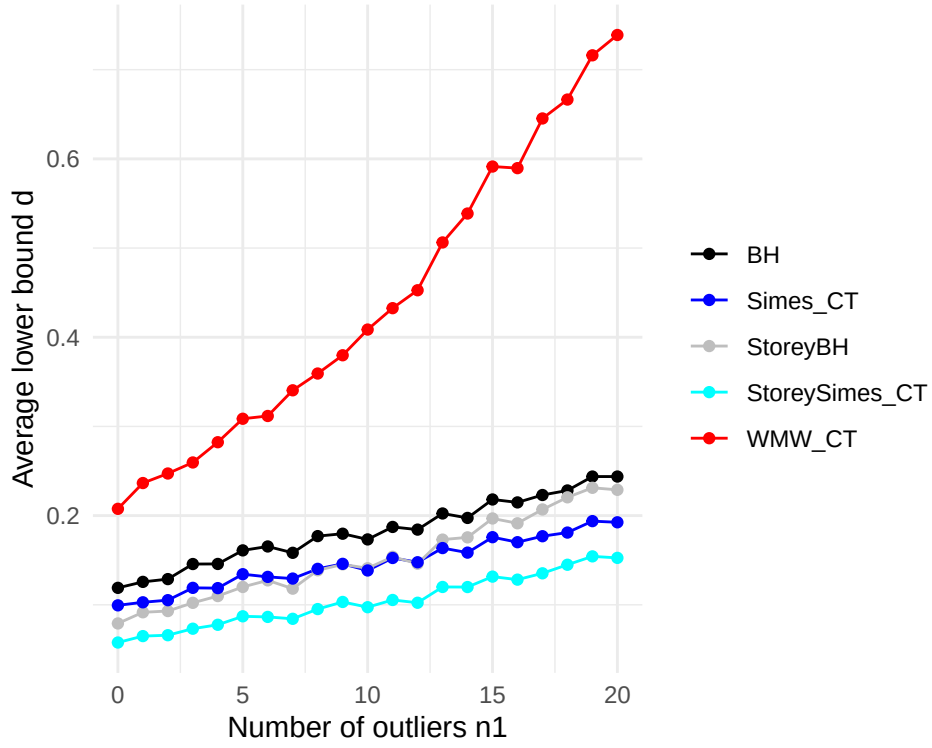

```
# Plot power
dfpower <- data.frame(
  x = n1s,
  BH = pow_BH,
  StoreyBH = pow_StoBH,
  WMW_CT = pow_WMW
```

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

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




```
# ALOI

load(paste0(pathResults,"\\resALOI0.1"))
results = resALOI0.1$compact.results
n1s = seq(from = 0, to = n, by=1)

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()
```

```r
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 lower bound d
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("black", "blue", "gray", "cyan", "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 = 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")

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

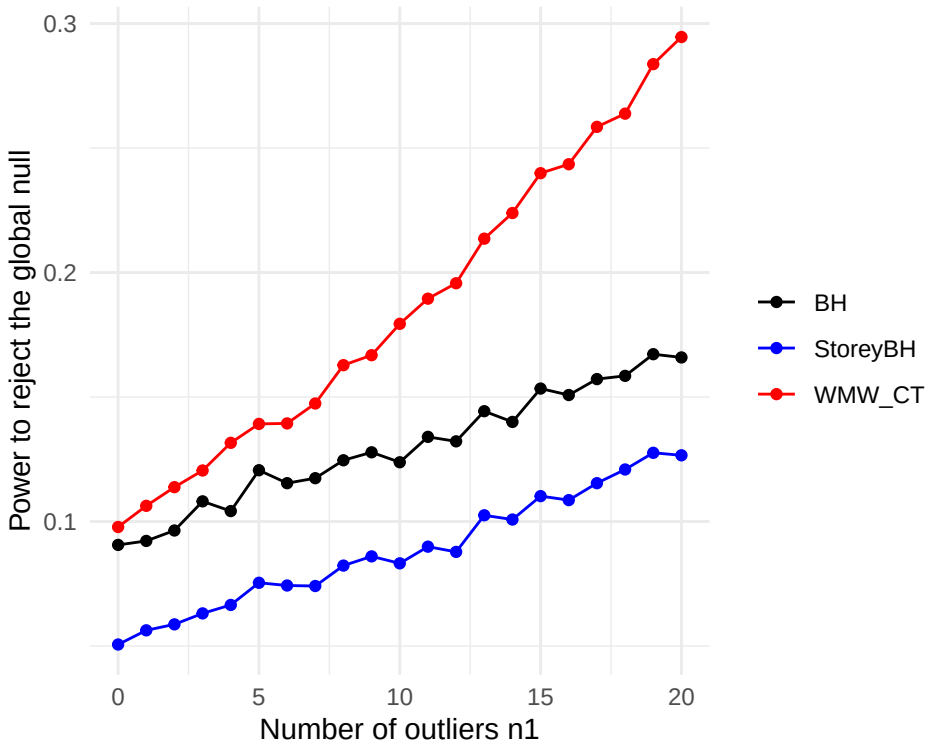
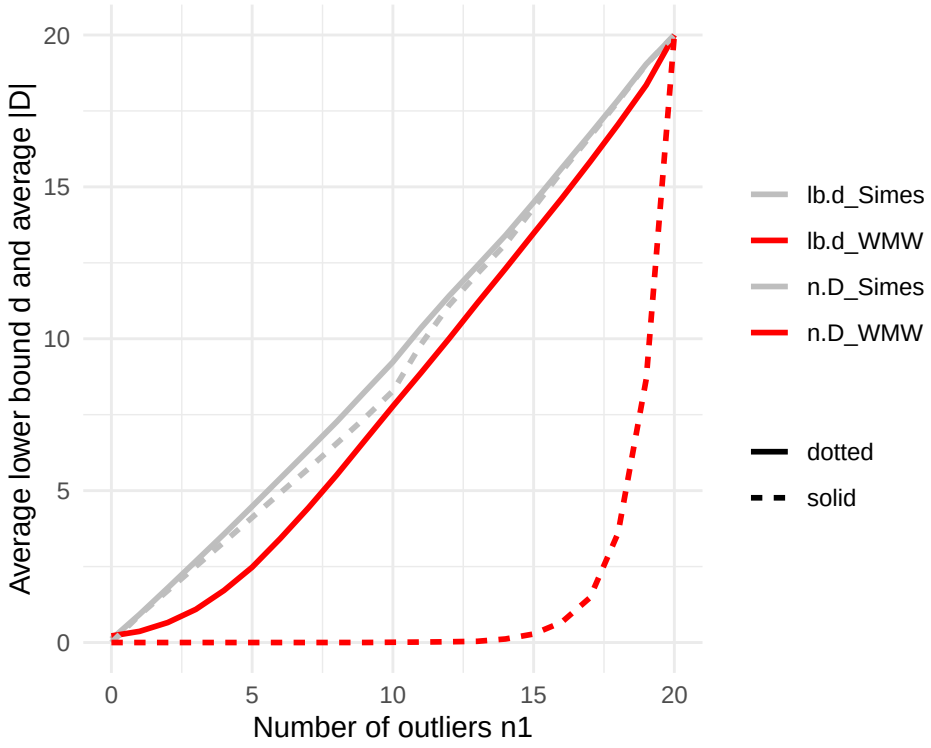


Figure 7

```r
# Shuttle

load(paste0(pathResults,"\\matrixShuttle0.1"))

df <- data.frame(
  x = n1s,
  lb.d_Simes = matrixShuttle0.1$lb.d.matrix[,3],
  lb.d_WMW = matrixShuttle0.1$lb.d.matrix[,5],
  n.D_Simes = matrixShuttle0.1$n.disc[,1],
  n.D_WMW = matrixShuttle0.1$n.disc[,3]
)
df_long <- tidyr::pivot_longer(df, cols = -x, names_to = "group", values_to = "y")

lineType <- function(group){
  if(startsWith(group, "n.D")) return("solid")
  else return("dotted")
}

ggplot(df_long, aes(x = x, y = y, color = group, linetype = sapply(group, lineType))) +
  geom_line(size=1) +
  scale_color_manual(values = c("gray", "red", "gray", "red")) +
  labs(x = "Number of outliers n1", y = "Average lower bound d and average |D|") +
  theme_minimal() +
  theme(legend.title = element_blank())
```


Average lower bound d and average |D|
20
15
10
5
0
0
5
10
15
20
Number of outliers n1
lb.d_Simes
lb.d_WMW
n.D_Simes
n.D_WMW
dotted
solid