Power study with Lehmann's alternative with $k = 3$

Chiara Gaia Magnani

2023-11-07

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 $k = 3$. Denoting inliers distribution by F, we are going to simulate the outliers distribution corresponding to F^k with $k = 3$ in order to perform a power analysis and to show that closed testing procedure with LMPI test statistic T_3 as local test is more powerful than closed testing with Simes local test with and without Storey estimator and than closed testing with Wilcoxon-Mann-Whitney local test.

Paths

```
# 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="\\")

pathDatasets = file.path("G:", "Il mio Drive", "PHD", "Progetto di ricerca",
                         "RankTestsForOutlierDetection",
                         "BiomJ - RankTestsForOutlierDetection", "R code", "Datasets",
                         fsep="\\")

pathResults = file.path("G:", "Il mio Drive", "PHD", "Progetto di ricerca",
                        "RankTestsForOutlierDetection",
                        "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)
```

```r
# Lehmann's outlier distribution for k=3

compact_resultsk3 = 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",]),
            "d_T3"=unlist(res[[j]][rownames(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)

    # 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",]),
    #         "n.disc.T3" = unlist(res[[j]][rownames(res[[j]])=="n.disc.T3",])
    #         )
    # )
    # 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",],
                        "S_cal" = (res[[j]][rownames(res[[j]])=="S_cal",]),
                        "S_te" = (res[[j]][rownames(res[[j]])=="S_te",]),
                        "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))

}




CompareMethodLehmannOutliersk3 = 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,
                  "S_cal" = S_cal,
                  "S_te" = S_te,
```

```
                    "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
l = 199
m = 199
n = 20
alpha = n/(m+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) CompareMethodLehmannOutliersk3(B=B, k=3, n1=n1s[j], n=n,
                                                        dataset=dataset,
                                                        isofo.model=modeltrain$model,
                                                        out_ind=out_ind,
                                                        inlier_remaining=modeltrain$inlier_remaining))
stopCluster(cluster)

results = compact_resultsk3(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] = 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]]$mean.lb.d[5]
  d_T3[j] = results[[j]]$mean.lb.d[6]

  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]]$mean.powerGlobalNull[5]
  pow_T3[j] = results[[j]]$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, "blue")) +
  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
```

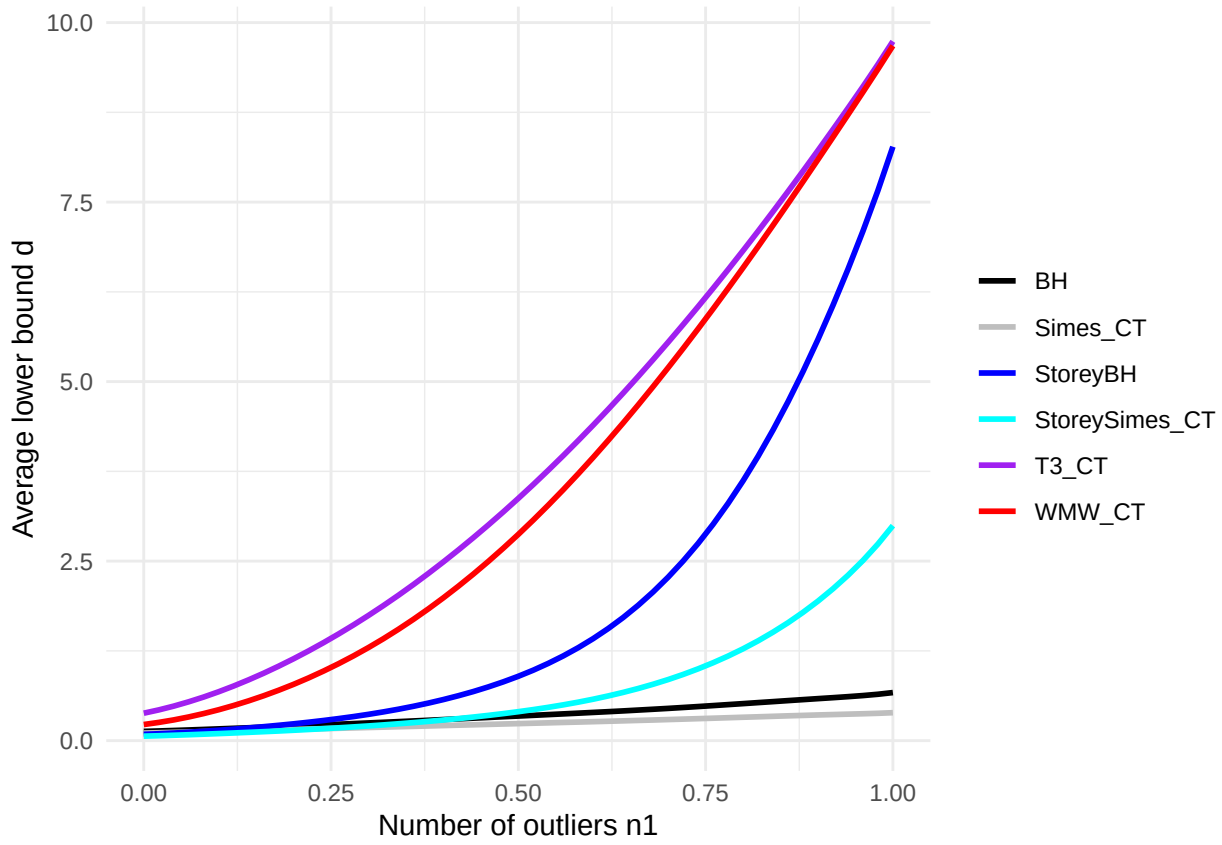
```
ggplot(df_long_power, aes(x = x, y = y, color = group)) +
  geom_line() +
  geom_point()+
  scale_color_manual(values = c("#DC143C","#FFA07A",5, "blue")) +
  labs(x = "n1", y = "power", title = "Mean of the power conditional on n1 values on B replications") +
  theme_minimal() +
  theme(legend.title = element_blank())


# 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),
                            "uncond.pow_T3" = apply(pow_T3[-1]*probsn1, MARGIN = 2, sum))
print(unconditional.power)

resDigits0.1k3 = list("raw.res"=res,
                      "unconditional.power" = unconditional.power,
                      "compact.results" = results)
save(resDigits0.1k3, file="~/nout/Examples/Digits/Lehmannk3/resDigits0.1k3")
```

```
load(file="~/nout/Examples/Digits/Lehmannk3/resDigits0.1k3")

# Compacting intermediate results in a matrix

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

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] = resDigits0.1k3$compact.results[[j]]$mean.lb.d[1]
  d_StoBH[j] = resDigits0.1k3$compact.results[[j]]$mean.lb.d[2]
  d_Sim[j] = resDigits0.1k3$compact.results[[j]]$mean.lb.d[3]
  d_StoSimes[j] = resDigits0.1k3$compact.results[[j]]$mean.lb.d[4]
  d_WMW[j] = resDigits0.1k3$compact.results[[j]]$mean.lb.d[5]
  d_T3[j] = resDigits0.1k3$compact.results[[j]]$mean.lb.d[6]
```

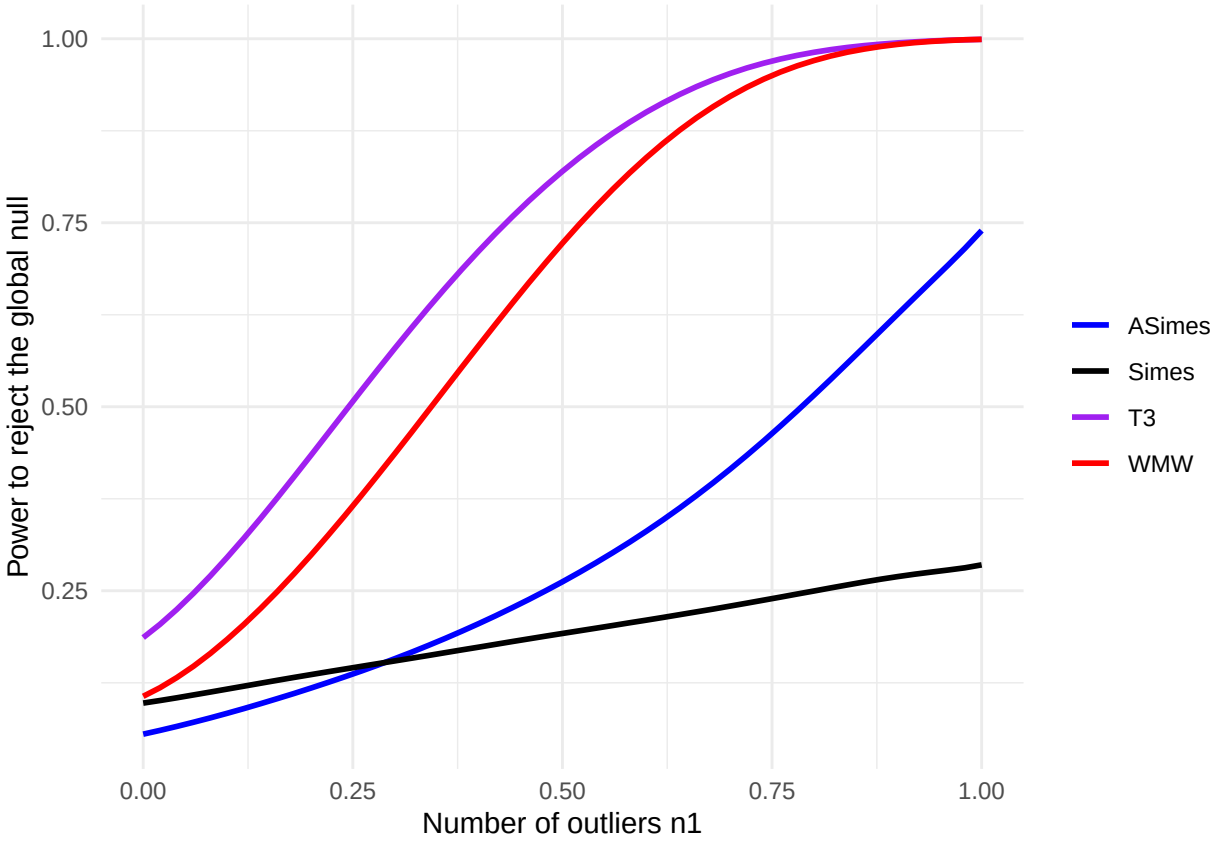
```
  pow.rejGlob_BH[j] = resDigits0.1k3$compact.results[[j]]$mean.powerGlobalNull[1]
  pow.rejGlob_StoBH[j] = resDigits0.1k3$compact.results[[j]]$mean.powerGlobalNull[2]
  pow.rejGlob_Sim[j] = resDigits0.1k3$compact.results[[j]]$mean.powerGlobalNull[3]
  pow.rejGlob_StoSimes[j] = resDigits0.1k3$compact.results[[j]]$mean.powerGlobalNull[4]
  pow.rejGlob_WMW[j] = resDigits0.1k3$compact.results[[j]]$mean.powerGlobalNull[5]
  pow.rejGlob_T3[j] = resDigits0.1k3$compact.results[[j]]$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

matrixDigits0.1k3 = list("lb.d.matrix" = lb.d,
                         "pow.rejGlob.matrix" = pow.rejGlob)
save(matrixDigits0.1k3,
     file = paste0("~/nout/Examples/Digits/Lehmannk3","/matrixDigits0.1k3"))
```

```
load(file = paste0("~/nout/Examples/Digits/Lehmannk3","/matrixDigits0.1k3"))

res = matrixDigits0.1k3

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())
```




```
# 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())
```


1.00
0.75
0.50
0.25
Power to reject the global null
0.00
0.25
0.50
0.75
1.00
Number of outliers n1
ASimes
Simes
T3
WMW