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

2023-07-27

The aim is to compare on real datasets the performance of three closed testing procedures, which respectively use Simes local test with and without Storey estimator for the proportion of true null hypotheses and Wilcoxon-Mann-Whitney local test. We will consider outlier population to be the set of observations tagged as "outlier" in the dataset of interest.

R functions and libraries

```
library(nout)
library(R.matlab)
library(isotree)
library(farff)
library(tictoc)
library(tidyverse)
library(doSNOW)
library(ggplot2)
library(hommel)

compact_results = function(res){
  resT=as.data.frame(t(res))

  results = list()
  for(j in 1:length(n1s)){
    discoveries = as.data.frame(
      cbind("d_BH"=unlist(res[[j]][rownames(res[[j]])=="d_BH",]),
            "d_StoBH"=unlist(res[[j]][rownames(res[[j]])=="d_StoBH",]),
            "d_Sim"=unlist(res[[j]][rownames(res[[j]])=="d_Sim",]),
            "d_StoSimes"=unlist(res[[j]][rownames(res[[j]])=="d_StoSimes",]),
            "d_WMW"=unlist(res[[j]][rownames(res[[j]])=="d_WMW",])
            )
    )
    mean.discoveries = apply(discoveries, MARGIN = 2, FUN = mean)

    power.GlobalNull = as.data.frame(discoveries>0)
    mean.powerGlobalNull = apply(power.GlobalNull, MARGIN = 2, FUN = mean)

    out_identification = as.data.frame(
      cbind("out.identif_WMW"=
              unlist(res[[j]][rownames(res[[j]])=="outlier.identified_WMW",]),
            "out.identif_StoSimes"=
              unlist(res[[j]][rownames(res[[j]])=="outlier.identified_StoSimes",]),
            "out.identif_Simes"=
              unlist(res[[j]][rownames(res[[j]])=="outlier.identified_Simes",])
            #"l1"=unlist(res[[j]][rownames(res[[j]])=="l1",]),
```

```
            #"l2"=unlist(res[[j]][rownames(res[[j]])=="l2",])
            )
    )
    mean.out_identification = apply(out_identification, MARGIN = 2, FUN = mean)
    mean.out_identification_pos = apply(out_identification>0, MARGIN = 2, FUN = mean)

    mean.l1 = mean(unlist(res[[j]][rownames(res[[j]])=="l1",]))
    mean.l2 = mean(unlist(res[[j]][rownames(res[[j]])=="l2",]))


    results[[j]] = list("discoveries" = discoveries,
                        "mean.discoveries" = mean.discoveries,
                        "power.GlobalNull" = power.GlobalNull,
                        "mean.powerGlobalNull" = mean.powerGlobalNull,
                        "out_identification" = out_identification,
                        "mean.out_identification" = mean.out_identification,
                        "mean.out_identification>0" = mean.out_identification_pos,
                        "mean.l1" = mean.l1,
                        "mean.l2" = mean.l2,
                        "pi.not" = res[[j]][rownames(res[[j]])=="pi.not",],
                        "uniques" = res[[j]][rownames(res[[j]])=="uniques",],
                        "n1" = res[[j]][rownames(res[[j]])=="n1",1],
                        "alpha" = res[[j]][rownames(res[[j]])=="alpha",1])
  }
  return(results)
}


TrainingIsoForest = function(l, dataset){

  tr_ind = sample(in_ind, size = l)
  tr = dataset[tr_ind,]
  isofo.model = isotree::isolation.forest(tr, ndim=ncol(dataset), ntrees=10, nthreads=1,
                            scoring_metric = "depth", output_score = TRUE)$model
  in_index2 = setdiff(in_ind, tr_ind)

  return(list("model"=isofo.model, "inlier_remaining" = in_index2))

}



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

  n0 = n-n1
  foreach(b = 1:B, .combine=cbind) %dopar% {
    if(n1==0){
      N = n0 + m
      in_index3 = sample(inlier_remaining, size = N)
      cal_ind = in_index3[1:m]
      te_ind = in_index3[(m+1):N]
      cal = dataset[cal_ind,]
      te = dataset[te_ind,]
```

```
  S_cal = predict.isolation_forest(isofo.model, cal, type = "score")
  S_te = predict.isolation_forest(isofo.model, te, type = "score")

  d_WMW = nout::d_MannWhitney(S_Y = S_te, S_X = S_cal, alpha=alpha)
  d_Sim = nout::d_Simes(S_X = S_cal, S_Y = S_te, alpha = alpha)
  StoSimes = nout::d_StoreySimes(S_X = S_cal, S_Y = S_te, alpha = alpha)
  d_StoSimes = StoSimes$d
  pi.not = StoSimes$pi.not
  d_BH = nout::d_benjhoch(S_X = S_cal, S_Y = S_te, alpha = alpha)
  d_StoBH = nout::d_StoreyBH(S_X = S_cal, S_Y = S_te, alpha = alpha)
  uniques = length(unique(c(S_cal, S_te)))
  return(list("d_BH" = d_BH,
              "d_StoBH" = d_StoBH,
              "d_Sim" = d_Sim,
              "d_StoSimes" = d_StoSimes,
              "d_WMW" = d_WMW,
              "outlier.identified_WMW" = 0,
              "outlier.identified_Simes" = 0,
              "outlier.identified_StoSimes" = 0,
              "l1" = 0,
              "l2" = 0,
              "uniques" = uniques,
              "n1" = n1,
              "pi.not" = pi.not,
              "alpha" = alpha))
}

else{
  N = n0 + m
  in_index3 = sample(inlier_remaining, size = N)
  cal_ind = in_index3[1:m]
  if(n0!=0)
    tein_ind = in_index3[(m+1):N]
  else
    tein_ind = NULL
  teout_ind = sample(out_ind, size = n1)
  cal = dataset[cal_ind,]
  te = dataset[c(tein_ind, teout_ind),]
  S_cal = predict.isolation_forest(isofo.model, cal, type = "score")
  S_te = predict.isolation_forest(isofo.model, te, type = "score")

  d_WMW = nout::d_MannWhitney(S_Y = S_te, S_X = S_cal, alpha=alpha)
  d_Sim = nout::d_Simes(S_X = S_cal, S_Y = S_te, alpha = alpha)
  StoSimes = nout::d_StoreySimes(S_X = S_cal, S_Y = S_te, alpha = alpha)
  d_StoSimes = StoSimes$d
  pi.not = StoSimes$pi.not
  d_BH = nout::d_benjhoch(S_X = S_cal, S_Y = S_te, alpha = alpha)
  d_StoBH = nout::d_StoreyBH(S_X = S_cal, S_Y = S_te, alpha = alpha)
  uniques = length(unique(c(S_cal, S_te)))

  # outlier identification with WMW
  conf.pval = sapply(1:n, function(j) (1+sum(S_cal >= S_te[j]))/(m+1))
  confvalid.pval = conf.pval<alpha
```

```
      confvalid.index = which(conf.pval<alpha)

      if(d_WMW>0){
        outlierTF = sapply(confvalid.index, function(h)
            nout::dselection_MannWhitney(S_Y = S_te, S_X = S_cal, S = h, alpha=alpha))
        outlier.identified_WMW = confvalid.index[as.logical(outlierTF)]
      }
      else  outlier.identified_WMW = NULL

      # outlier identification with Simes
      l1=0;l2=0
      if(d_Sim>0){
        outlierTF = sapply(confvalid.index, function(h)
            nout::dselection_Simes(S_Y = S_te, S_X = S_cal, S = h, alpha=alpha))
        outlier.identified_Simes = confvalid.index[as.logical(outlierTF)]
        # Second method
        p = hommel(conf.pval)
        # number of discoveries <= d
        l1 = sum(p@adjusted <= alpha)
        # equivalent to
        l2 = sum(sapply(1:length(conf.pval), function(i)
          discoveries(p, ix=i, alpha=alpha)==1))
      }
      else  outlier.identified_Simes = NULL

      # outlier identification with StoreySimes
      if(d_StoSimes>0){
        outlierTF = sapply(confvalid.index, function(h)
            nout::dselection_StoreySimes(S_Y = S_te, S_X = S_cal, S = h, alpha=alpha))
        outlier.identified_StoSimes = confvalid.index[as.logical(outlierTF)]
      }
      else  outlier.identified_StoSimes = NULL

      return(list("d_BH" = d_BH,
                  "d_StoBH" = d_StoBH,
                  "d_Sim" = d_Sim,
                  "d_StoSimes" = d_StoSimes,
                  "d_WMW" = d_WMW,
                  "outlier.identified_WMW" = length(outlier.identified_WMW),
                  "outlier.identified_Simes" = length(outlier.identified_Simes),
                  "outlier.identified_StoSimes" = length(outlier.identified_StoSimes),
                  "l1" =  l1,
                  "l2" = l2,
                  "uniques" = uniques,
                  "n1" = n1,
                  "pi.not" = pi.not,
                  "alpha" = alpha))
    }
  }
}


estimatek = function(B, inlier_remaining, out_ind, isofo.model, dataset){
```

```
  ress = foreach(b = 1:B, .combine=c) %dopar% {
   inlier_ind = sample(inlier_remaining, size = 1)
   outlier_ind = sample(out_ind, size = 1)
   inlier = dataset[inlier_ind,]
   outlier = dataset[outlier_ind,]
   S_inlier = predict.isolation_forest(isofo.model, inlier, type = "score")
   S_outlier = predict.isolation_forest(isofo.model, outlier, type = "score")

   greater.logi = S_inlier<S_outlier

   return(greater.logi)
  }

  greater.prob = mean(ress)
  k=greater.prob/(1-greater.prob)
  return(k)
}
```

In the following we set the calibration set and the test set size, respectively l and m, so that the nominal level α is proportional to $\frac{m}{l+1}$. The train set size is equal to n and the number of iterations is $B = 10^4$.

Digits dataset

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

```
set.seed(321)

# Initializing parameters
B = 1000
m = 199
l = 199
n = 20
alpha = n/(l+1)
n1s = seq(from=0, to=n, by=1)

data = readMat("~/nout/trials/RealData/Datasets/Dataset digits/pendigits.mat")
dataset = cbind(data$X, data$y); colnames(dataset)[ncol(dataset)] = "y"
in_ind = which(dataset[,ncol(dataset)]==0)
out_ind = which(dataset[,ncol(dataset)]==1)

cluster <- makeCluster(parallel::detectCores())
registerDoSNOW(cluster)
clusterEvalQ(cluster, {list(library(isotree), library(nout), library(hommel))})
```

```
## [[1]]
## [[1]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[1]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
## [[1]][[3]]
##  [1] "hommel"    "nout"      "isotree"   "snow"      "stats"     "graphics"
```

```
##  [7] "grDevices" "utils"     "datasets"  "methods"   "base"
##
##
## [[2]]
## [[2]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[2]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
## [[2]][[3]]
##  [1] "hommel"    "nout"      "isotree"   "snow"      "stats"     "graphics"
##  [7] "grDevices" "utils"     "datasets"  "methods"   "base"
##
##
## [[3]]
## [[3]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[3]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
## [[3]][[3]]
##  [1] "hommel"    "nout"      "isotree"   "snow"      "stats"     "graphics"
##  [7] "grDevices" "utils"     "datasets"  "methods"   "base"
##
##
## [[4]]
## [[4]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[4]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
## [[4]][[3]]
##  [1] "hommel"    "nout"      "isotree"   "snow"      "stats"     "graphics"
##  [7] "grDevices" "utils"     "datasets"  "methods"   "base"
```

```
clusterExport(cluster, list("n", "m", "l", "in_ind", "out_ind", "dataset", "alpha"))

tic()
modeltrain = TrainingIsoForest(l=l, dataset=dataset)
# kest = estimatek(B=B, inlier_remaining=modeltrain$inlier_remaining,
#          out_ind=out_ind, isofo.model=modeltrain$model, dataset=dataset)
res = lapply(1:length(n1s),
             function(j) CompareMethodNaturalOutliers(B=B, n1=n1s[j], n=n, dataset=dataset,
                                 isofo.model=modeltrain$model,
                                 out_ind=out_ind,
```

```
                                inlier_remaining=modeltrain$inlier_remaining))
toc()
```

```
## 950.05 sec elapsed
```

```
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]
#
# }
#
# # Plot discoveries
# df <- data.frame(
#   x = n1s,
#   BH = d_BH,
#   StoreyBH = d_StoBH,
#   Simes_CT = d_Sim,
#   StoreySimes_CT = d_StoSimes,
#   WMW_CT = d_WMW
# )
# df_long <- tidyr::pivot_longer(df, cols = -x, names_to = "group", values_to = "y")
#
# ggplot(df_long, aes(x = x, y = y, color = group)) +
#   geom_line() +
#   geom_point()+
#   scale_color_manual(values = c("#DC143C", "#FFA07A", "#808000", "#BDB76B", 5)) +
#   labs(x = "n1", y = "d", title = "Mean of the number of discoveries on B replications") +
```

```
#   theme_minimal() +
#   theme(legend.title = element_blank())
#
#
# # Plot power
# dfpower <- data.frame(
#   x = n1s,
#   BH = pow_BH,
#   StoreyBH = pow_StoBH,
#   WMW_CT = pow_WMW
# )
# df_long_power <- tidyr::pivot_longer(dfpower, cols = -x, names_to = "group", values_to = "y")
#
# # Plot the lines with different colors and legends
# ggplot(df_long_power, aes(x = x, y = y, color = group)) +
#   geom_line() +
#   geom_point()+
#   scale_color_manual(values = c("#DC143C","#FFA07A",5)) +
#   labs(x = "n1", y = "power", title = "Mean of the power on B replications") +
#   theme_minimal() +
#   theme(legend.title = element_blank())

outlier.identification = list()
for(i in 1:length(n1s)){
  outlier.identification[[i]] = matrix(nrow = 3, ncol = 4)
  rownames(outlier.identification[[i]]) = c("WMW", "StoSimes","Simes")
  colnames(outlier.identification[[i]]) = c("mean.out.ident", "%successful.ident",
                                    "mean.d", "mean.d>0(power)")
   outlier.identification[[i]][,1] = apply(
    results[[i]][["out_identification"]], MARGIN = 2, FUN = mean)
  outlier.identification[[i]][,2] = apply(
    results[[i]][["out_identification"]]>0, MARGIN = 2, FUN = mean)
  outlier.identification[[i]][,3] = results[[i]]$mean.discoveries[c(3,4,5)]
  outlier.identification[[i]][,4] = results[[i]]$mean.powerGlobalNull[c(3,4,5)]

}

for(i in 1:length(n1s)){
  cat("\n")
  cat(paste("n1=", n1s[i]))
  print(outlier.identification[[i]])
  print(paste("l1.mean = ", results[[i]]$mean.l1))
  print(paste("l2.mean = ", results[[i]]$mean.l2))
}
```

```
##
## n1= 0         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW                       0                 0  0.121           0.105
## StoSimes                  0                 0  0.069           0.055
## Simes                     0                 0  0.182           0.087
## [1] "l1.mean =  0"
## [1] "l2.mean =  0"
##
## n1= 1         mean.out.ident %successful.ident mean.d mean.d>0(power)
```

```
## WMW               0.528             0.152  0.204           0.181
## StoSimes          0.000             0.000  0.129           0.106
## Simes             0.184             0.167  0.357           0.153
## [1] "l1.mean =  0.184"
## [1] "l2.mean =  0.184"
##
## n1= 2         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               0.931             0.235  0.277           0.247
## StoSimes          0.000             0.000  0.205           0.175
## Simes             0.260             0.235  0.583           0.235
## [1] "l1.mean =  0.26"
## [1] "l2.mean =  0.26"
##
## n1= 3         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               1.430             0.349  0.397           0.329
## StoSimes          0.000             0.000  0.328           0.255
## Simes             0.368             0.306  0.906           0.352
## [1] "l1.mean =  0.368"
## [1] "l2.mean =  0.368"
##
## n1= 4         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               2.001             0.448  0.538           0.419
## StoSimes          0.000             0.000  0.490           0.363
## Simes             0.484             0.383  1.163           0.448
## [1] "l1.mean =  0.484"
## [1] "l2.mean =  0.484"
##
## n1= 5         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               2.773             0.592  0.647           0.473
## StoSimes          0.000             0.000  0.646           0.448
## Simes             0.570             0.432  1.793           0.593
## [1] "l1.mean =  0.57"
## [1] "l2.mean =  0.57"
##
## n1= 6         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               3.628             0.704  0.740           0.511
## StoSimes          0.000             0.000  0.828           0.523
## Simes             0.640             0.462  2.446           0.704
## [1] "l1.mean =  0.64"
## [1] "l2.mean =  0.64"
##
## n1= 7         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               4.415             0.817  0.869           0.583
## StoSimes          0.000             0.000  1.016           0.620
## Simes             0.781             0.548  3.100           0.817
## [1] "l1.mean =  0.781"
## [1] "l2.mean =  0.781"
##
## n1= 8         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               5.297             0.909  0.986           0.627
## StoSimes          0.000             0.000  1.298           0.686
## Simes             0.861             0.581  3.980           0.909
## [1] "l1.mean =  0.861"
## [1] "l2.mean =  0.861"
```

```
##
## n1= 9         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               5.908             0.959  1.135           0.673
## StoSimes          0.000             0.000  1.581           0.756
## Simes             0.963             0.624  4.830           0.959
## [1] "l1.mean =  0.963"
## [1] "l2.mean =  0.963"
##
## n1= 10         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               6.523             0.982  1.218           0.708
## StoSimes          0.000             0.000  1.930           0.799
## Simes             1.010             0.654  5.747           0.982
## [1] "l1.mean =  1.01"
## [1] "l2.mean =  1.01"
##
## n1= 11         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               7.109             0.997  1.329           0.734
## StoSimes          0.000             0.000  2.310           0.847
## Simes             1.091             0.668  6.739           0.997
## [1] "l1.mean =  1.091"
## [1] "l2.mean =  1.091"
##
## n1= 12         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               7.463             0.998  1.568           0.789
## StoSimes          0.000             0.000  2.919           0.920
## Simes             1.298             0.739  7.778           0.998
## [1] "l1.mean =  1.298"
## [1] "l2.mean =  1.298"
##
## n1= 13         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               8.018              1.00  1.698           0.801
## StoSimes          0.000              0.00  3.509           0.943
## Simes             1.319              0.73  8.717           1.000
## [1] "l1.mean =  1.319"
## [1] "l2.mean =  1.319"
##
## n1= 14         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               8.576             1.000  1.900           0.854
## StoSimes          0.000             0.000  4.434           0.970
## Simes             1.455             0.775  9.870           1.000
## [1] "l1.mean =  1.455"
## [1] "l2.mean =  1.455"
##
## n1= 15         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               9.016             1.000  2.029           0.863
## StoSimes          0.000             0.000  5.240           0.981
## Simes             1.466             0.796 10.814           1.000
## [1] "l1.mean =  1.466"
## [1] "l2.mean =  1.466"
##
## n1= 16         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW               9.417             1.000  2.179           0.877
## StoSimes          0.000             0.000  6.383           0.993
## Simes             1.548             0.805 11.940           1.000
```

```
## [1] "l1.mean =  1.548"
## [1] "l2.mean =  1.548"
##
## n1= 17         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW              10.054             1.000  2.498           0.891
## StoSimes          0.000             0.000  7.831           0.997
## Simes             1.730             0.821 13.110           1.000
## [1] "l1.mean =  1.73"
## [1] "l2.mean =  1.73"
##
## n1= 18         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW              10.468             1.000  2.701           0.893
## StoSimes          0.000             0.000  9.395           0.999
## Simes             1.794             0.841 14.317           1.000
## [1] "l1.mean =  1.794"
## [1] "l2.mean =  1.794"
##
## n1= 19         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW              11.023             1.000  3.061           0.924
## StoSimes          0.000             0.000 11.176           0.999
## Simes             1.940             0.854 15.523           1.000
## [1] "l1.mean =  1.94"
## [1] "l2.mean =  1.94"
##
## n1= 20         mean.out.ident %successful.ident mean.d mean.d>0(power)
## WMW              11.306             1.000  3.188           0.928
## StoSimes          0.000             0.000 13.318           1.000
## Simes             1.977             0.862 16.840           1.000
## [1] "l1.mean =  1.977"
## [1] "l2.mean =  1.977"
```

```
CheckDiscovierieSimesDigits0.1 = list("raw.res"=res,
                    #"k.est" = kest,
                    "compact.results" = results,
                    "outlier.identification" = outlier.identification)
save(CheckDiscovierieSimesDigits0.1, file="~/nout/trials/RealData/PowerStudy/FinalSimu/Digits/CheckDisc
```