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

2023-08-09

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(mlbench)
library(tictoc)
library(tidyverse)
library(doSNOW)
library(ggplot2)
library(hommel)

compact_results = function(res){

  lb.d = as.data.frame(
    cbind("d_Sim"=unlist(res["d_Sim"]),
          "d_StoSimes"=unlist(res["d_StoSimes"]),
          "d_WMW"=unlist(res["d_WMW"]))
    )
  rownames(lb.d) = ""
  power.GlobalNull = as.data.frame(t(as.numeric(lb.d>0)))
  colnames(power.GlobalNull) = c("d_Sim","d_StoSimes", "d_WMW")

  n.disc = as.data.frame(
      cbind("n.disc.Simes" = unlist(res["d_Sim"]),
            "n.disc.StoSimes" = unlist(res["d_StoSimes"]),
            "n.disc.WMW" = unlist(res["d_WMW"]),
            "n.disc.WMW.cpp" = unlist(res["d_WMW.cpp"]))
  )
  rownames(n.disc) = ""

  results = list("lb.d" = lb.d,
                 "power.GlobalNull" = power.GlobalNull,
                 "n.disc" = n.disc,
                 "uniques" = unlist(res["uniques"]),
                 "n1" = unlist(res["n1"]),
```

```r
                    "alpha" = unlist(res["alpha"]))
  return(results)
}


TrainingIsoForest.S = function(l, dataset, in_ind){

  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))

}



CompareMethod.S = function( m, n, S, inlier_remaining, isofo.model, dataset, alpha){
  n1 = length(S)
  n0 = n-n1
  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(S)
  cal = dataset[cal_ind,]
  te = dataset[c(teout_ind, tein_ind),]
  S_cal = predict.isolation_forest(isofo.model, cal, type = "score")
  S_te = predict.isolation_forest(isofo.model, te, type = "score")

  d_WMW = nout::dselection_MannWhitney(S_Y = S_te, S_X = S_cal, S = 1:length(S), alpha=alpha)
  d_Sim = nout::dselection_Simes(S_X = S_cal, S_Y = S_te, S = 1:length(S),  alpha = alpha)
  d_StoSimes = nout::dselection_StoreySimes(S_X = S_cal, S_Y = S_te,
                                            S = 1:length(S), 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.cpp=0
  n.disc.WMW=0
  if(d_WMW>0 & length(confvalid.index)!=0){
    outlierTF.WMW.cpp = sapply(confvalid.index, function(h)
      nout::dselection_MannWhitney(S_Y = S_te, S_X = S_cal, S = h, alpha=alpha))
    #outlier.identified_MannWhitney = confvalid.index[as.logical(outlierTF)]
  n.disc.WMW.cpp = sum(outlierTF.WMW.cpp)
  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))
  }

  # 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))
      #outlier.identified_StoSimes = confvalid.index[as.logical(outlierTFStoSim)]
    n.disc.StoSimes = sum(outlierTF.StoSim)
  }

  return(list("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,
              "n.disc.WMW.cpp" = n.disc.WMW.cpp,
              "uniques" = uniques,
              "n1" = n1,
              "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 $B = 10^4$.

Statlog (Shuttle) dataset in library mlbench

The dataset is available in the R library ***mlbench***.

```
# Load the data
data("Shuttle")
levels(Shuttle$Class) = c("1", "2", "3", "4", "5", "6", "7")
table(Shuttle$Class)
```

```
##
##     1     2     3     4     5     6     7
## 45586    50   171  8903  3267    10    13
```

```
# Delete class 4
fours = which(Shuttle[,10]== "4")
Shuttle2 = Shuttle[-fours,]

# Different classes of outliers
```

```r
out.classes = c("2","3","5","6","7")

out2.ind = which(Shuttle2$Class == levels(Shuttle2[,10])[2])
out3.ind = which(Shuttle2$Class == levels(Shuttle2[,10])[3])
out5.ind = which(Shuttle2$Class == levels(Shuttle2[,10])[5])
out6.ind = which(Shuttle2$Class == levels(Shuttle2[,10])[6])
out7.ind = which(Shuttle2$Class == levels(Shuttle2[,10])[7])

inliers.ind = which(Shuttle2[,10] == "1")
outliers.ind = setdiff(1:nrow(Shuttle2), inliers.ind)
#length(outliers.ind)==length(out2.ind)+length(out3.ind)+length(out5.ind)+length(out6.ind)+length(out7.

# Creating Outlier column
# =1 if observation i is outlier
# =0 if observations i is inlier
outlier = rep(0, times = nrow(Shuttle2))
outlier[outliers.ind] = 1
#sum(outliers)

Shuttle2 = cbind(Shuttle2, "Outlier" = outlier)



set.seed(321)

# Initializing parameters
# l = 19999
# m = 19999
# n = 4000
l = 3999
m = 3999
n = 400
alpha = n/(m+1)

S2 = out2.ind
S3 = out3.ind
S5 = out5.ind
S6 = out6.ind
S7 = out7.ind

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"
```

```r
clusterExport(cluster, list( "l", "Shuttle2", "inliers.ind"))

iso.fo = TrainingIsoForest.S(l=l, in_ind = inliers.ind, dataset=Shuttle2)

resS2 = CompareMethod.S(n=n, m=m, S=S2, alpha = alpha,
                      dataset=Shuttle2,
                      isofo.model=iso.fo$model,
                      inlier_remaining=iso.fo$inlier_remaining)
```

```
resS3 = CompareMethod.S(n=n, m=m, S=S3, alpha = alpha,
                      dataset=Shuttle2,
                      isofo.model=iso.fo$model,
                      inlier_remaining=iso.fo$inlier_remaining)


resS6 = CompareMethod.S(n=n, m=m, S=S6, alpha = alpha,
                      dataset=Shuttle2,
                      isofo.model=iso.fo$model,
                      inlier_remaining=iso.fo$inlier_remaining)


resS7 = CompareMethod.S(n=n, m=m, S=S7, alpha = alpha,
                      dataset=Shuttle2,
                      isofo.model=iso.fo$model,
                      inlier_remaining=iso.fo$inlier_remaining)



stopCluster(cluster)

resultsS2 = compact_results(resS2)
print(resultsS2$n1)
```

```
## n1
## 50
```

```
print(resultsS2$lb.d)
```

```
##  d_Sim d_StoSimes d_WMW
##     26         27     0
```

```
print(resultsS2$power.GlobalNull)
```

```
##   d_Sim d_StoSimes d_WMW
## 1     1          1     0
```

```
print(resultsS2$n.disc)
```

```
##  n.disc.Simes n.disc.StoSimes n.disc.WMW
##            26              27          0
```

```
resultsS3 = compact_results(resS3)
print(resultsS3$n1)
```

```
##  n1
## 171
```

```
print(resultsS3$lb.d)
```

```
##  d_Sim d_StoSimes d_WMW
##    160        160   105
```

```
print(resultsS3$power.GlobalNull)
```

```
##   d_Sim d_StoSimes d_WMW
## 1     1          1     1
```

```
print(resultsS3$n.disc)
```

```
##  n.disc.Simes n.disc.StoSimes n.disc.WMW
##           160             160        105
```

```
resultsS6 = compact_results(resS6)
print(resultsS6$n1)
```

```
## n1
## 10
```

```
print(resultsS6$lb.d)
```

```
##  d_Sim d_StoSimes d_WMW
##      7          7     4
```

```
print(resultsS6$power.GlobalNull)
```

```
##   d_Sim d_StoSimes d_WMW
## 1     1          1     1
```

```
print(resultsS6$n.disc)
```

```
##  n.disc.Simes n.disc.StoSimes n.disc.WMW
##             7               7          4
```

```
resultsS7 = compact_results(resS7)
print(resultsS7$n1)
```

```
## n1
## 13
```

```
print(resultsS7$lb.d)
```

```
##  d_Sim d_StoSimes d_WMW
##     12         11     0
```

```
print(resultsS7$power.GlobalNull)
```

```
##   d_Sim d_StoSimes d_WMW
## 1     1          1     0
```

```
print(resultsS7$n.disc)
```

```
##  n.disc.Simes n.disc.StoSimes n.disc.WMW
##            12              11          0
```

```
resShuttle.S = list("resultsS2"=resultsS2,
                    "resultsS3"=resultsS3,
                    "resultsS6"=resultsS6,
                    "resultsS7"=resultsS7)
save(resShuttle.S,
     file="~/nout/trials/RealData/PowerStudy/FinalSimu/Shuttle/resShuttle.S")
```