Real Data Power Analysis

Significance level 0.2

2023-05-16

```
library(nout)
library(R.matlab)
library(isotree)
library(tictoc)
library(doSNOW)


simParallel = function(B, dataset, in_index, out_index, n_te, n_outliers, n_cal,
                       iso.ntree, iso.samplesize, alpha, lambda = 0.5, seed=321){

  set.seed(seed)

  tr_ind =  sample(in_index, size = l)
  in_ind2 = setdiff(in_index, tr_ind)
  tr = dataset[tr_ind,]
  n_cpus = parallel::detectCores()
  iso.fo = isotree::isolation.forest(tr, ndim=ncol(dataset), ntrees=iso.ntree, sample_size = iso.samples
                                     nthreads=n_cpus,
                                     scoring_metric = "depth", output_score = TRUE)
  isofo.model=iso.fo$model

  mycrit = nout::critWMW(m=n,n=m,alpha=alpha)

  # Create the cluster with ncpus-2 cores:
  cluster <- makeCluster(n_cpus-1, type = "SOCK")
  registerDoSNOW(cluster)
  # export dependencies in cluster
  clusterEvalQ(cluster, {c(library(nout), library(hommel), library(isotree), library(dplyr))})
  clusterExport(cluster, list("single_sim", "isofo.model", "alpha", "in_index", "out_index",
                              "n_te", "n_outliers", "n_cal", "mycrit", "lambda"),
                envir=environment())

  res = snow::parSapply(cl = cluster,
                        X = 1:B,
                        FUN = single_sim,
                        dataset=dataset, model=isofo.model, in_index=in_index,
                        out_index=out_index, n_outliers=n_outliers, n_te=n_te,
                        n_cal=n_cal, crit=mycrit, alpha=alpha, lambda = lambda)

  # Stop cluster on master
  stopCluster(cluster)
```

```
  return(res)
}



single_sim = function(b, dataset, model, in_index, out_index, n_cal, n_te,
                      n_outliers, crit, alpha, lambda = 0.5){
  if(n_outliers==0){

    N=n_te+n_cal
    in_index3 = sample(in_index, size = N)
    cal_ind = in_index3[1:n_cal]
    te_ind = in_index3[(n_cal+1):N]
    cal = dataset[cal_ind,]
    te = dataset[te_ind,]

  }
  else{

    n_inliers = n-n_outliers
    N=n_inliers+n_cal
    in_index3 = sample(in_index, size = N)
    cal_ind = in_index3[1:n_cal]
    tein_ind = in_index3[(n_cal+1):N]
    teout_ind = sample(out_index, size = n_outliers)
    cal = dataset[cal_ind,]
    te = dataset[c(tein_ind, teout_ind),]

  }

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

  d_WMW = nout::d_mannwhitney(S_Y=S_te, S_X=S_cal, crit = crit)
  d_Simes = 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, lambda=lambda)
  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, lambda=lambda)
  uniques = length(unique(c(S_cal, S_te)))

  return(as.data.frame(cbind( "d_BH"=d_BH,
                              "d_StoBH"=d_StoBH,
                              "d_Simes"=d_Simes,
                              "d_StoSimes"=d_StoSimes,
                              "d_WMW"=d_WMW,
                              "uniques"=uniques,
                              "n_outliers"=n_outliers,
                              "pi.not"=pi.not,
                              "alpha"=alpha)))
}
```

```
compact_results = function(res){
  resT=as.data.frame(t(res))
  colnames(resT) = c("d_BH", "d_StoBH", "d_Simes", "d_StoSimes", "d_WMW", "uniques", "n_outliers", "pi.n

  discoveries = as.data.frame(cbind("d_BH"=unlist(resT$d_BH),
                                    "d_StoBH"=unlist(resT$d_StoBH),
                                    "d_Simes"=unlist(resT$d_Simes),
                                    "d_StoSimes"=unlist(resT$d_StoSimes),
                                    "d_WMW"=unlist(resT$d_WMW)))
  colnames(discoveries) = c("BH", "BHSto", "CTSim", "CTSimSto", "CTWMW")
  mean.discoveries = apply(discoveries, MARGIN = 2, FUN = mean)

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

  return(list("discoveries" = discoveries,
              "mean.discoveries" = mean.discoveries,
              "power.GlobalNull" = power.GlobalNull,
              "mean.powerGlobalNull" = mean.powerGlobalNull,
              "pi.not" = unlist(resT$pi.not),
              "uniques"=unlist(resT$uniques),
              "n1"=unlist(resT$n_outliers),
              "alpha"=unlist(resT$alpha)))
}
```

The aim is to compare on Shuttle 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 fix the train set on which we train the isolation forest algorithm and we generate $B = 10^4$ calibration and test sets. For each $b = 1, \dots, B$ we compute the number of discoveries obtained by Benjamini-Hochberg procedure with and without Storey's estimator for the proportion of true null hypotheses, by closed testing using Simes local test with and without Storey's estimator and by closed testing using Wilcoxon-Mann-Whitney local test.

Shuttle (Statlog) dataset

Shuttle dataset (available at http://odds.cs.stonybrook.edu/shuttle-dataset) consists of 49097 observations, among which $n_{inliers} = 45586$ items are inliers and the remaining $n_{outliers} = 3511$ are outliers. We will denote by l, m, n respectively the train set, the calibration set and the test set size. And reproducing the same setting as in [1], we have that $m + l = n_{inliers}/2$, $m = min\{2000, l/2\}$ and $n = min\{2000, l/3\}$. Moreover, in order to have exact control of type I errors at the significance level $\alpha = 0.2$. we require $\alpha = n/(m+1)$. In the case of Shuttle dataset we obtain $l = 12794, \quad m = 9999, \quad n = 2000$.

Load the data and set the parameters as described above.

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

# Initializing parameters
B=10^4
l = 12794
```

```
m = 9999
n = 2000
myalpha = n/(m+1)
```

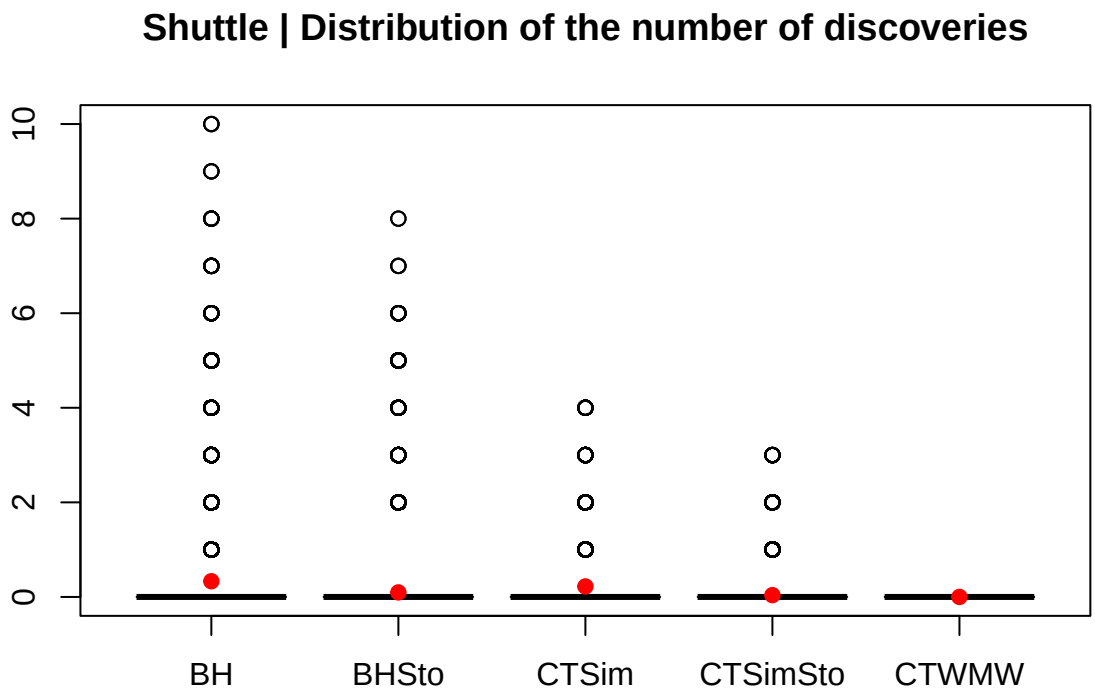
All inliers

We now set the proportion of inliers equal to 1, so that the number of outliers $n_1 = 0$.

```
n1=0
re = simParallel(B=B, dataset=dataset, in_index=in_ind, out_index=out_ind, iso.ntree=3, iso.samplesize=
                 n_te=n, n_outliers=n1, n_cal=m, alpha=myalpha, lambda = 0.5, seed=321)

results = compact_results(re)

boxplot(results$discoveries, main="Shuttle | Distribution of the number of discoveries")
points(x=1:5, y=results$mean.discoveries, pch=19, col="red")
```




```
results$mean.discoveries
```

```
##       BH    BHSto    CTSim CTSimSto    CTWMW
##   0.3322   0.0959   0.2222   0.0379   0.0000
```

```
results$mean.powerGlobalNull
```

```
##       BH    BHSto    CTSim CTSimSto    CTWMW
##   0.1839   0.0310   0.1839   0.0310   0.0000
```

```
resShuttle0 = results
save(resShuttle0,
     file="~/nout/trials/RealData/PowerStudy/New!/alpha0.2/ShuttleOnly0.2/resShuttle0")
```

10% outliers

We now set the proportion of inliers equal to 0.9. Referring to Digits dataset we have that the number of inliers is $n_0 = 1800$ and the number of outliers is $n_1 = 200$.

```
n1=round(0.1*n)

set.seed(321)

tr_ind =  sample(in_ind, size = 1)
in_ind2 = setdiff(in_ind, tr_ind)
tr = dataset[tr_ind,]
n_cpus = parallel::detectCores()
iso.fo = isotree::isolation.forest(tr, ndim=ncol(dataset), ntrees=150, sample_size = 256,
                                   nthreads=n_cpus, scoring_metric = "depth",
                                   output_score = TRUE)
isofo.model=iso.fo$model

mycrit = nout::critWMW(m=n,n=m,alpha=myalpha)

d_WMW = vector()
d_Sim = vector()
d_StoSimes = vector()
pi.not = vector()
d_BH = vector()
d_StoBH = vector()
uniques = vector()

for(b in 1:B){

  n0 = n-n1
  N=n0+m
  in_index3 = sample(in_ind, size = N)
  cal_ind = in_index3[1:m]
  tein_ind = in_index3[(m+1):N]
  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[b] = nout::d_mannwhitney(S_Y=S_te, S_X=S_cal, crit = mycrit)
  d_Sim[b] = nout::d_Simes(S_X=S_cal, S_Y=S_te, alpha=myalpha)
  StoSimes = nout::d_StoreySimes(S_X=S_cal, S_Y=S_te, alpha=myalpha)
  d_StoSimes[b] = StoSimes$d
  pi.not[b] = StoSimes$pi.not
  d_BH[b] = nout::d_benjhoch(S_X=S_cal, S_Y=S_te, alpha=myalpha)
  d_StoBH[b] = nout::d_StoreyBH(S_X=S_cal, S_Y=S_te, alpha=myalpha)
```

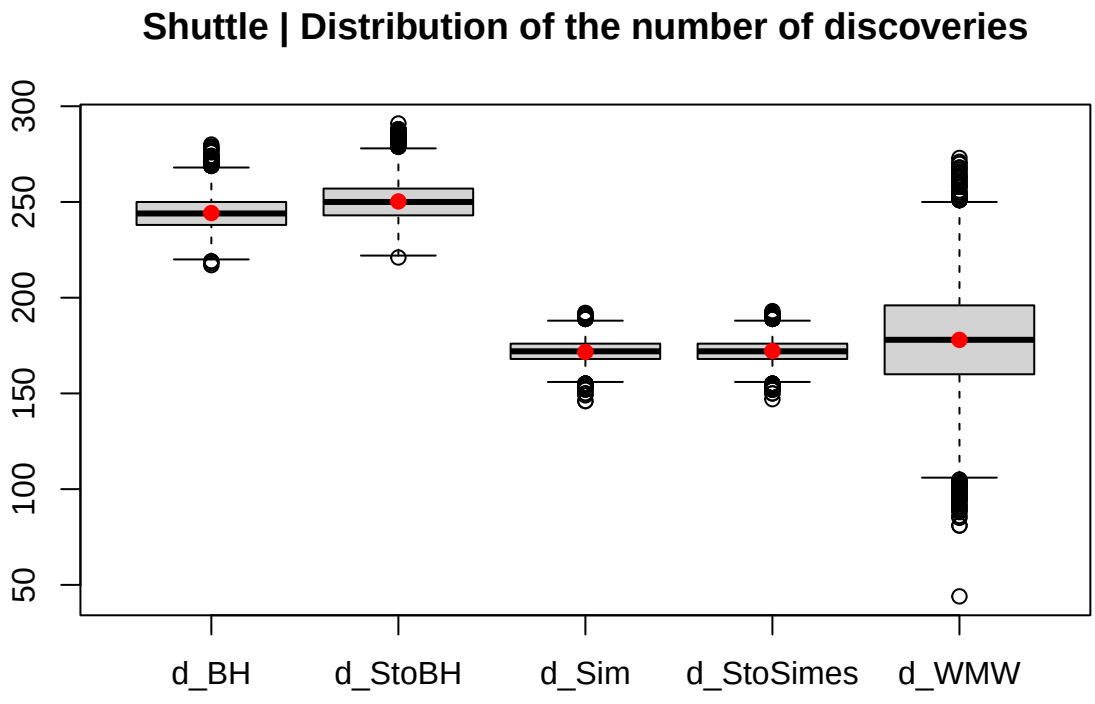
```
  uniques[b] = length(unique(c(S_cal, S_te)))
}

  res=as.data.frame(cbind( "d_BH"=d_BH,
                           "d_StoBH"=d_StoBH,
                           "d_Sim"=d_Sim,
                           "d_StoSimes"=d_StoSimes,
                           "d_WMW"=d_WMW,
                           "uniques"=uniques,
                           "n_outliers"=n1,
                           "pi.not"=pi.not,
                           "alpha"=myalpha))

discoveries = res[,1:5]
mean.discoveries = apply(discoveries, MARGIN=2, FUN=mean)

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

boxplot(discoveries, main="Shuttle | Distribution of the number of discoveries")
points(x=1:5, y=mean.discoveries, pch=19, col="red")
```




```
mean.discoveries
```

```
##       d_BH    d_StoBH      d_Sim d_StoSimes      d_WMW
##   244.2083   250.2927   171.6461   172.1562   177.9860
```

```
mean.powerGlobalNull
```

```
##       d_BH    d_StoBH      d_Sim d_StoSimes      d_WMW
##          1          1          1          1          1
```

```
results = list("discoveries"=discoveries,
               "mean.discoveries"=mean.discoveries,
               "powerGlobalNull"=powerGlobalNull,
               "mean.powerGlobalNull"=mean.powerGlobalNull,
               "alpha"=myalpha,
               "n1"=n1,
               "uniques" = res$uniques,
               "pi.not" = res$pi.not)

resShuttle10 = results
save(resShuttle10,
     file="~/nout/trials/RealData/PowerStudy/New!/alpha0.2/ShuttleOnly0.2/resShuttle10")
```