Comparison between different local tests: Simes, Simes with Storey, Wilcoxon-Mann-Whitney and Wilcoxon-Mann-Whitney of order k=3

2023-07-20

The aim is to compare on real datasets the performance of four closed testing procedures, which respectively use Simes local test with and without Storey estimator for the proportion of true null hypotheses, Wilcoxon-Mann-Whitney local test and the test statistic corresponding to $k = 3$ derived in Theorem 1 of *Testing for outliers with ranks*. Denoting inliers distribution by F, we are going to simulate an outliers distribution corresponding to F^k with $k = 3$ in order to show that closed testing procedure with the local test based on the test statistic corresponding to $k = 3$ 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.

R functions and libraries

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

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",]),
                                      "d_WMWk3"=unlist(res[[j]][rownames(res[[j]])=="d_WMWk3",])))
    mean.discoveries = apply(discoveries, MARGIN = 2, FUN = mean)

    power.GlobalNull = as.data.frame(discoveries>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)
}


foreachfz = function(B, n1){

  k = 3
  n0 = n-n1

  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)

  if(n1!=0){
    foreach(b = 1:B, .combine=cbind) %dopar% {
      N = n0 + m
      in_index3 = sample(in_index2, size = N)

      cal_ind = in_index3[1:m]
      cal = dataset[cal_ind,]

      tein_ind = in_index3[(m + 1):N]
      teoutaug_ind = sample(out_ind, size = (k*n1))
      teout_ind = vector()
      for(j in 1:n1){
        inds = teout_ind[(j+(j-1)*(k-1)):(j+j*(k-1))]
        ind = max(inds)
        teout_ind[j] = ind
      }
      te = dataset[c(tein_ind, teout_ind),]

      S_cal = isotree::predict.isolation_forest(isofo.model, cal, type = "score")
      S_te = isotree::predict.isolation_forest(isofo.model, te, type = "score")
      d_WMW = nout::d_MannWhitney(S_Y = S_te, S_X = S_cal, alpha=myalpha)
      d_WMWk3 = nout::d_MannWhitneyk3(S_Y = S_te, S_X = S_cal, alpha=myalpha)
      d_Sim = 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 = StoSimes$d
      pi.not = StoSimes$pi.not
      d_BH = nout::d_benjhoch(S_X = S_cal, S_Y = S_te, alpha = myalpha)
      d_StoBH = nout::d_StoreyBH(S_X = S_cal, S_Y = S_te, alpha = myalpha)
      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_WMWk3" = d_WMWk3,
                  "uniques" = uniques,
                  "n1" = n1,
```

```
                  "pi.not" = pi.not,
                  "alpha" = myalpha))
    }
  }

  else{
    foreach(b = 1:B, .combine=cbind) %dopar% {
      n0 = n
      N = n0 + m
      in_index3 = sample(in_index2, 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 = isotree::predict.isolation_forest(isofo.model, cal, type = "score")
      S_te = isotree::predict.isolation_forest(isofo.model, te, type = "score")
      d_WMW = nout::d_MannWhitney(S_Y = S_te, S_X = S_cal, alpha=myalpha)
      d_WMWk3 = nout::d_MannWhitneyk3(S_Y = S_te, S_X = S_cal, alpha=myalpha)
      d_Sim = 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 = StoSimes$d
      pi.not = StoSimes$pi.not
      d_BH = nout::d_benjhoch(S_X = S_cal, S_Y = S_te, alpha = myalpha)
      d_StoBH = nout::d_StoreyBH(S_X = S_cal, S_Y = S_te, alpha = myalpha)
      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_WMWk3" = d_WMWk3,
                  "uniques" = uniques,
                  "n1" = n1,
                  "pi.not" = pi.not,
                  "alpha" = myalpha))
    }
  }
}
```

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 $B = 10^5$.

Covertype dataset

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

```
set.seed(321)

# Initializing parameters
B = 10^4
m = 199
l = 199
n = 20
myalpha = n/(l+1)
```

```
n1s = seq(from=0, to=n, by=1)

data = readMat("~/nout/trials/RealData/Datasets/Dataset cover type/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))})
```

```
## [[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"
##
##
## [[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"
##
##
## [[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"
##
##
## [[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"
```

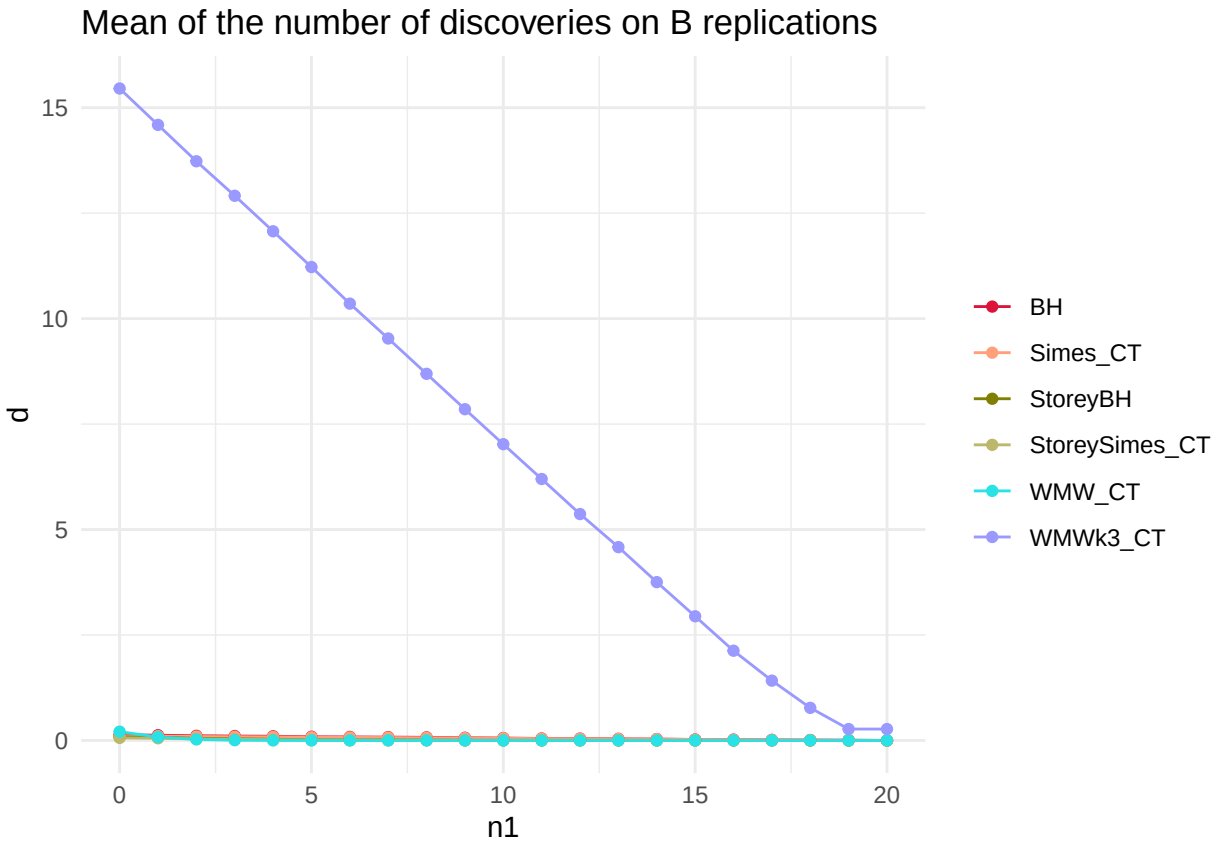
```
clusterExport(cluster, list("n", "m", "l", "in_ind", "out_ind", "dataset", "myalpha"))

tic()
res = lapply(1:length(n1s), function(j) foreachfz(B=B, n1=n1s[j]))
```

```
toc()
```

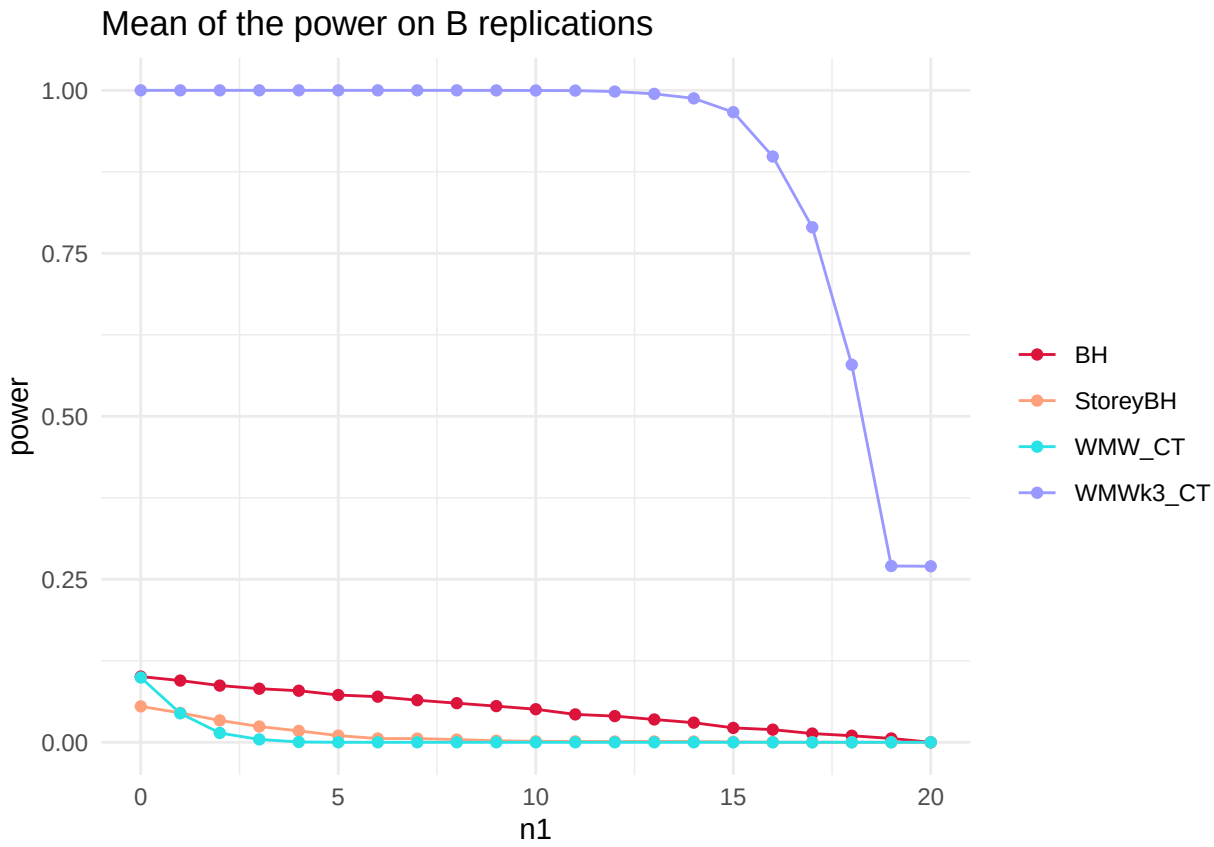
```
## 8888.08 sec elapsed
```

```
stopCluster(cluster)

results = compact_results(res)

d_BH = vector()
d_StoBH = vector()
d_Sim = vector()
d_StoSimes = vector()
d_WMW = vector()
d_WMWk3 = vector()

pow_BH = vector()
pow_StoBH = vector()
pow_Sim = vector()
pow_StoSimes = vector()
pow_WMW = vector()
pow_WMWk3 = 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]]$mean.discoveries[5]
  d_WMWk3[j] = results[[j]]$mean.discoveries[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]]$mean.powerGlobalNull[5]
  pow_WMWk3[j] = results[[j]]$mean.powerGlobalNull[6]

}

# 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,
  WMWk3_CT = d_WMWk3
)
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, "#9999FF")) +
  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,
  WMWk3_CT = pow_WMWk3
)
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, "#9999FF")) +
  labs(x = "n1", y = "power", title = "Mean of the power on B replications") +
  theme_minimal() +
  theme(legend.title = element_blank())
```




```
resCovertypek3Par = res
save(resCovertypek3Par, file="~/nout/trials/RealData/PowerStudy/New!/alpha0.2/CoverOnly0.2/resCovertypek
```