Comparison between different local tests: Simes, Simes with Storey and Wilcoxon-Mann-Whitney using the Lehmann alternative distribution with k=3

2023-08-03

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 $k = 2$. Denoting inliers distribution by F, we are going to simulate the outliers distribution corresponding to F^k with $k = 2$ in order to perform a power analysis and to show that closed testing procedure with Wilcoxon-Mann-Whitney local test is more powerful than closed testing with Simes local test with and without Storey estimator.

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)){
    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_WMWk3"=unlist(res[[j]][rownames(res[[j]])=="d_WMWk3",])
            )
    )
    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)

    results[[j]] = list("lb.d" = lb.d,
```

```r
                          "mean.lb.d" = mean.lb.d,
                          "power.GlobalNull" = power.GlobalNull,
                          "mean.powerGlobalNull" = mean.powerGlobalNull,
                          "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_WMWk3 = 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)
```

```r
      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_WMWk3" = d_WMWk3,
                  "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.

```r
set.seed(321)

# Initializing parameters
B = 10^4
m = 19
l = 19
n = 4
alpha = n/(m+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))})
```

```
## [[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"
##
##
## [[5]]
## [[5]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[5]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
##
## [[6]]
## [[6]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[6]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
```

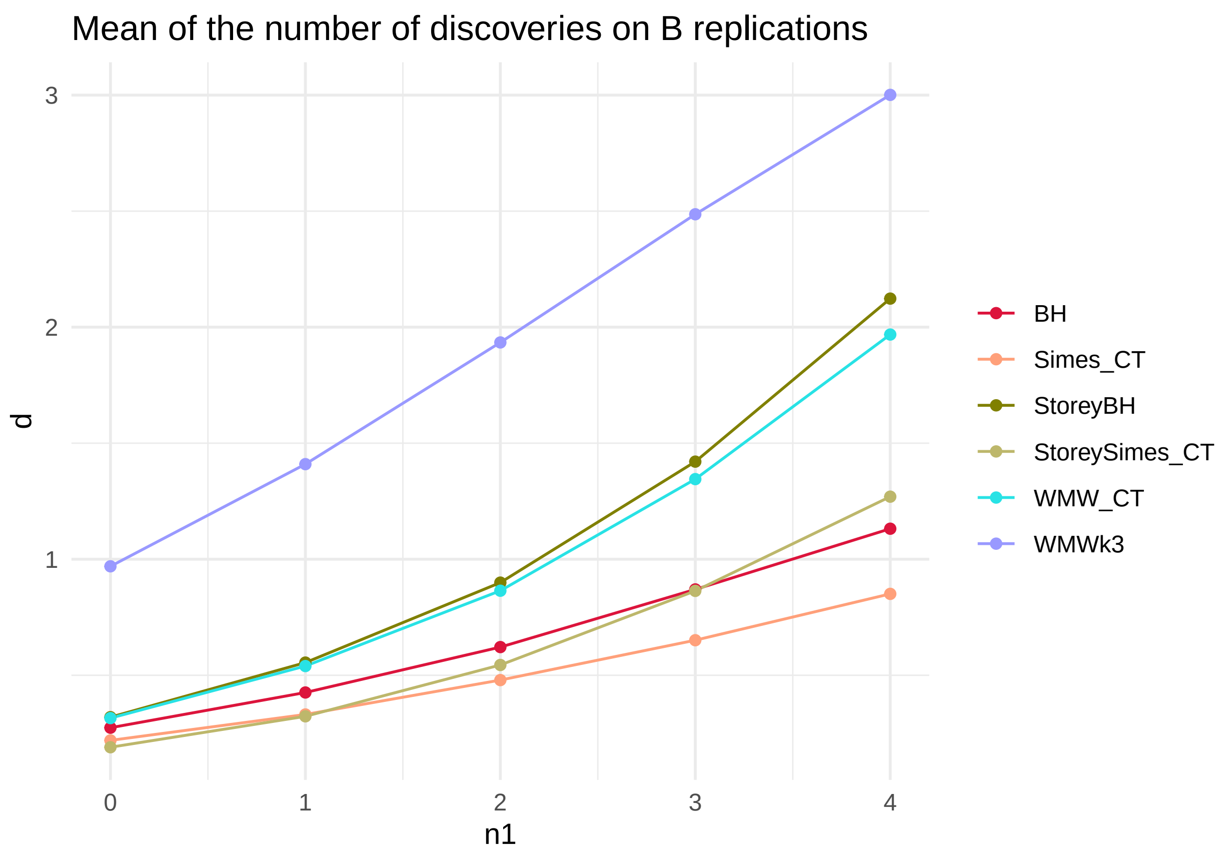
```
##
##
## [[7]]
## [[7]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[7]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
##
## [[8]]
## [[8]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[8]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
```

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

tic()
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))
toc()
```

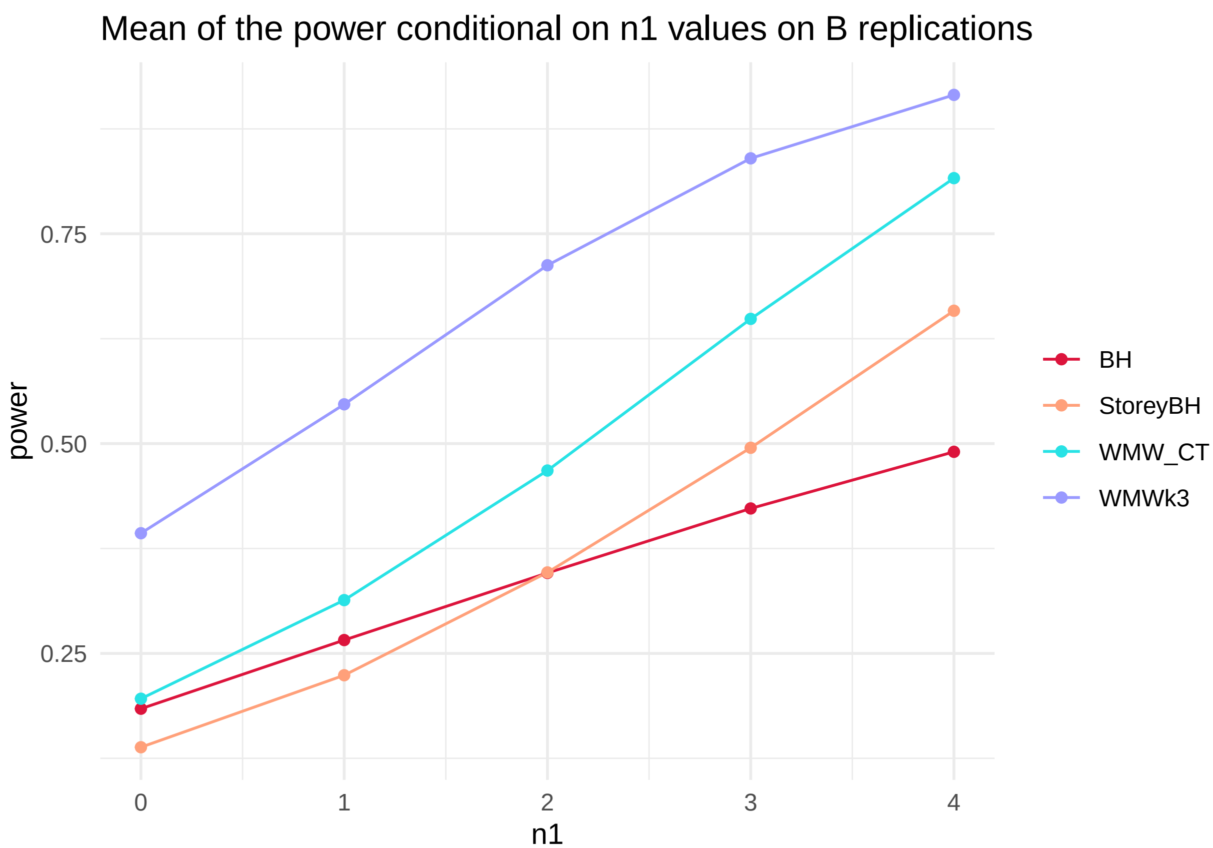
```
## 70.84 sec elapsed
```

```r
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()
```

```r
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]]$mean.lb.d[5]
  d_WMWk3[j] = results[[j]]$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]]$mean.powerGlobalNull[5]
  pow_WMWk3[j] = results[[j]]$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,
  WMWk3 = 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())
```




```r
# Plot power conditional on n1
dfpower <- data.frame(
  x = n1s,
  BH = pow_BH,
  StoreyBH = pow_StoBH,
  WMW_CT = pow_WMW,
  WMWk3 = 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 conditional on n1 values on B replications") +
  theme_minimal() +
  theme(legend.title = element_blank())
```




```r
# 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_WMWk3" = apply(pow_WMWk3[-1]*probsn1, MARGIN = 2, sum))

print(unconditional.power)
```

```
##      uncond.pow_BH uncond.pow_StoreyBH uncond.pow_WMW uncond.pow_WMWk3
## 0       0.00000000          0.00000000     0.00000000       0.00000000
## 0.02    0.02083166          0.01768046     0.02470406       0.04284027
## 0.04    0.04080565          0.03489788     0.04867953       0.08391628
## 0.06    0.05995597          0.05167596     0.07196071       0.12329180
## 0.08    0.07831588          0.06803784     0.09458106       0.16102922
## 0.1     0.09591792          0.08400613     0.11657313       0.19718957
## 0.12    0.11279391          0.09960292     0.13796865       0.23183251
## 0.14    0.12897495          0.11484976     0.15879847       0.26501634
## 0.16    0.14449138          0.12976766     0.17909257       0.29679797
## 0.18    0.15937287          0.14437712     0.19888009       0.32723297
## 0.2     0.17364832          0.15869808     0.21818928       0.35637552
```

```
## 0.22    0.18734592          0.17274997     0.23704754       0.38427844
## 0.24    0.20049315          0.18655167     0.25548142       0.41099319
## 0.26    0.21311673          0.20012153     0.27351658       0.43656985
## 0.28    0.22524269          0.21347739     0.29117783       0.46105712
## 0.3     0.23689632          0.22663653     0.30848913       0.48450237
## 0.32    0.24810218          0.23961571     0.32547356       0.50695157
## 0.34    0.25888412          0.25243114     0.34215334       0.52844932
## 0.36    0.26926525          0.26509853     0.35854983       0.54903888
## 0.38    0.27926796          0.27763304     0.37468353       0.56876211
## 0.4     0.28891392          0.29004928     0.39057408       0.58765952
## 0.42    0.29822407          0.30236136     0.40624024       0.60577025
## 0.44    0.30721862          0.31458283     0.42169992       0.62313207
## 0.46    0.31591707          0.32672673     0.43697017       0.63978139
## 0.48    0.32433818          0.33880555     0.45206718       0.65575322
## 0.5     0.33250000          0.35083125     0.46700625       0.67108125
## 0.52    0.34041984          0.36281527     0.48180185       0.68579776
## 0.54    0.34811428          0.37476850     0.49646758       0.69993369
## 0.56    0.35559920          0.38670131     0.51101615       0.71351859
## 0.58    0.36288975          0.39862354     0.52545945       0.72658066
## 0.6     0.37000032          0.41054448     0.53980848       0.73914672
## 0.62    0.37694462          0.42247290     0.55407338       0.75124223
## 0.64    0.38373561          0.43441704     0.56826343       0.76289127
## 0.66    0.39038554          0.44638460     0.58238704       0.77411657
## 0.68    0.39690591          0.45838275     0.59645178       0.78493948
## 0.7     0.40330752          0.47041813     0.61046433       0.79537997
## 0.72    0.40960044          0.48249684     0.62443052       0.80545667
## 0.74    0.41579400          0.49462446     0.63835532       0.81518682
## 0.76    0.42189683          0.50680602     0.65224283       0.82458630
## 0.78    0.42791682          0.51904604     0.66609629       0.83366962
## 0.8     0.43386112          0.53134848     0.67991808       0.84244992
## 0.82    0.43973619          0.54371679     0.69370971       0.85093898
## 0.84    0.44554773          0.55615388     0.70747183       0.85914720
## 0.86    0.45130074          0.56866213     0.72120424       0.86708362
## 0.88    0.45699949          0.58124337     0.73490585       0.87475591
## 0.9     0.46264752          0.59389893     0.74857473       0.88217037
## 0.92    0.46824764          0.60662958     0.76220808       0.88933193
## 0.94    0.47380194          0.61943556     0.77580224       0.89624417
## 0.96    0.47931180          0.63231659     0.78935269       0.90290926
## 0.98    0.48477784          0.64527186     0.80285402       0.90932805
## 1       0.49020000          0.65830000     0.81630000       0.91550000
```

```r
resDigits0.1k3 = list("raw.res"=res,
                    "unconditional.power" = unconditional.power,
                    "compact.results" = results #,
                    #"outlier.identification" = outlier.identification
                    )
save(resDigits0.1k3, file="~/nout/trials/RealData/PowerStudy/FinalSimu/Digits/Lehmannk3/resDigits0.1k3")
```