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

2023-08-08

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

  results = list()
  for(j in 1:length(n1s)){
    lb.d = as.data.frame(
      cbind("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.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)

    n.disc = as.data.frame(
      cbind("n.disc.Simes" = unlist(res[[j]][rownames(res[[j]])=="n.disc.Simes",]),
             "n.disc.StoSimes" = unlist(res[[j]][rownames(res[[j]])=="n.disc.StoSimes",]),
            "n.disc.WMW" = unlist(res[[j]][rownames(res[[j]])=="n.disc.WMW",]),
            "n.disc.WMW.cpp" = unlist(res[[j]][rownames(res[[j]])=="n.disc.WMW.cpp",])
            )
    )
    mean.n.disc = apply(n.disc, MARGIN = 2, FUN = mean)
```

```
    #mean.n.disc_pos = apply(n.disc>0, MARGIN = 2, FUN = mean)

    results[[j]] = list("lb.d" = lb.d,
                        "mean.lb.d" = mean.lb.d,
                        "power.GlobalNull" = power.GlobalNull,
                        "mean.powerGlobalNull" = mean.powerGlobalNull,
                        "n.disc" = n.disc,
                        "mean.n.disc" = mean.n.disc,
                        #"mean.n.disc>0" = mean.n.disc_pos,
                        "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.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(B, m, n, n1, S, inlier_remaining, isofo.model, dataset, alpha){

  n0 = n-n1
  foreach(b = 1:B, .combine=cbind) %dopar% {
      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, size = n1)
      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:n1, alpha=alpha)
      d_Sim = nout::dselection_Simes(S_X = S_cal, S_Y = S_te, S = 1:n1,  alpha = alpha)
      d_StoSimes = nout::dselection_StoreySimes(S_X = S_cal, S_Y = S_te,
                                                S = 1:n1, 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
    # n.disc.Simes2=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.identified_Simes = confvalid.index[as.logical(outlierTF)]
      # n.disc.Simes = sum(outlierTF.Simes)
      # p = hommel(conf.pval)
      # n.disc.Simes2 = sum(p@adjusted <= 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,
                # "n.disc.Simes2" = n.disc.Simes2,
                "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))
  }
}
```

```
estimatek.S = function(B, inlier_remaining, S, isofo.model, dataset){
  ress = foreach(b = 1:B, .combine=c) %dopar% {
   inlier_ind = sample(inlier_remaining, size = 1)
   outlier_ind = sample(S, 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 $B = 10^4$.

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
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
```

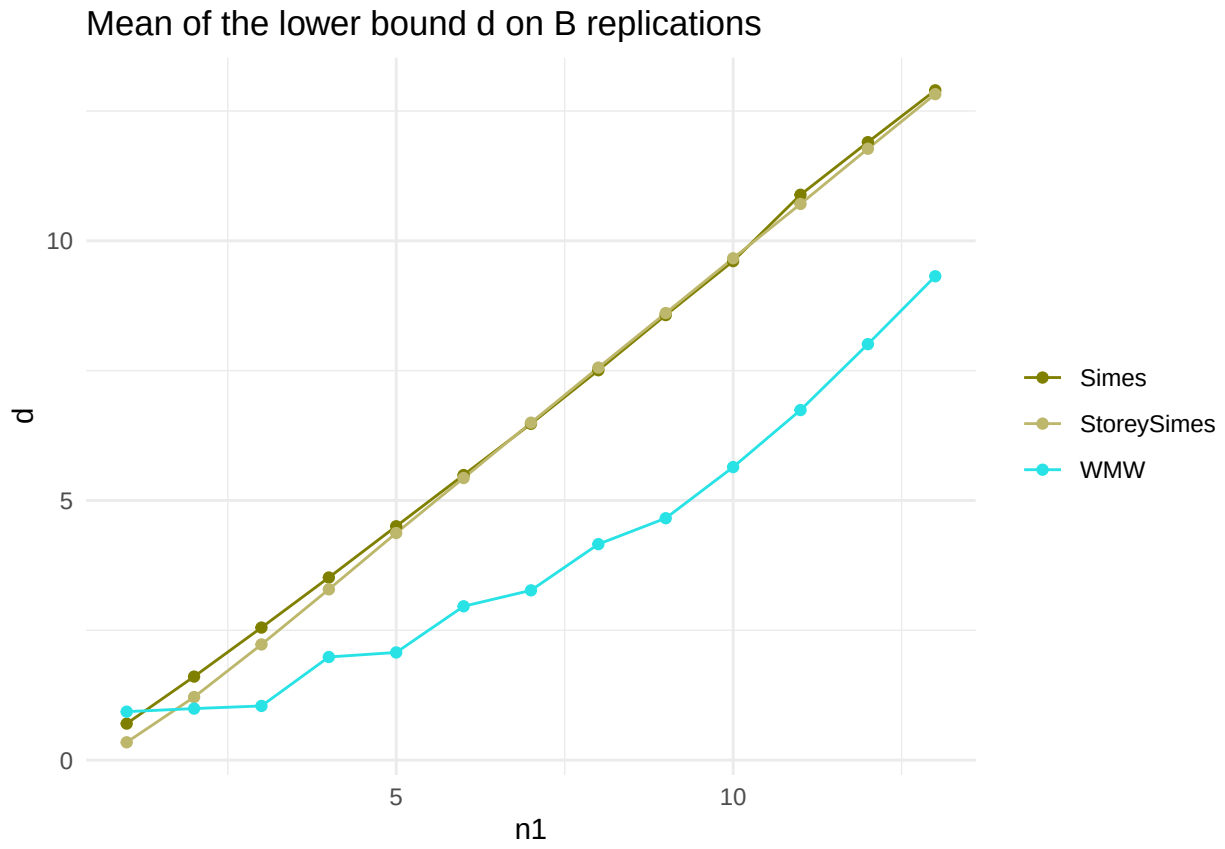
```r
outlier = rep(0, times = nrow(Shuttle2))
outlier[outliers.ind] = 1
#sum(outliers)

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



set.seed(321)

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

S = out7.ind
s = length(S)
n1s = seq(from = 1, to = s, by = 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"
##
##
```

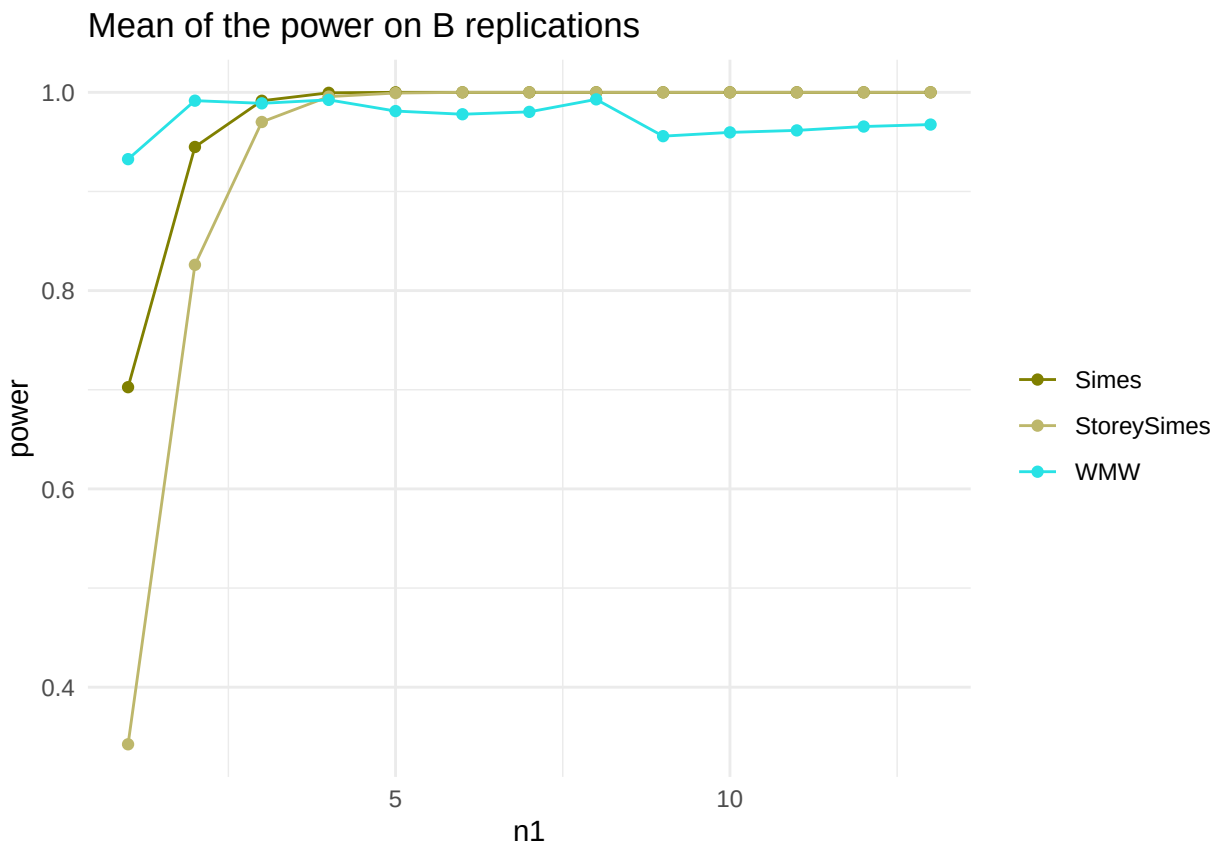
```
## [[3]]
## [[3]][[1]]
## [1] "isotree"   "snow"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[3]][[2]]
##  [1] "nout"      "isotree"   "snow"      "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
## [[3]][[3]]
##  [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( "l", "Shuttle2", "inliers.ind"))
iso.fo = TrainingIsoForest.S(l=l, in_ind = inliers.ind, dataset=Shuttle2)

# kest = estimatek.S(B=B,
#                    inlier_remaining=iso.fo$inlier_remaining,
#                    S=S,
#                    isofo.model=iso.fo$model,
#                    dataset = Shuttle2)

res = lapply(1:length(n1s),
             function(j) CompareMethod.S(B=B, n1=n1s[j], n=n, m=m, S=S, alpha = alpha,
                                         dataset=Shuttle2,
                                         isofo.model=iso.fo$model,
                                         inlier_remaining=iso.fo$inlier_remaining)
             )

stopCluster(cluster)

# kest

results = compact_results(res)

d_Sim = vector()
d_StoSimes = vector()
d_WMW = vector()

pow_Sim = vector()
```

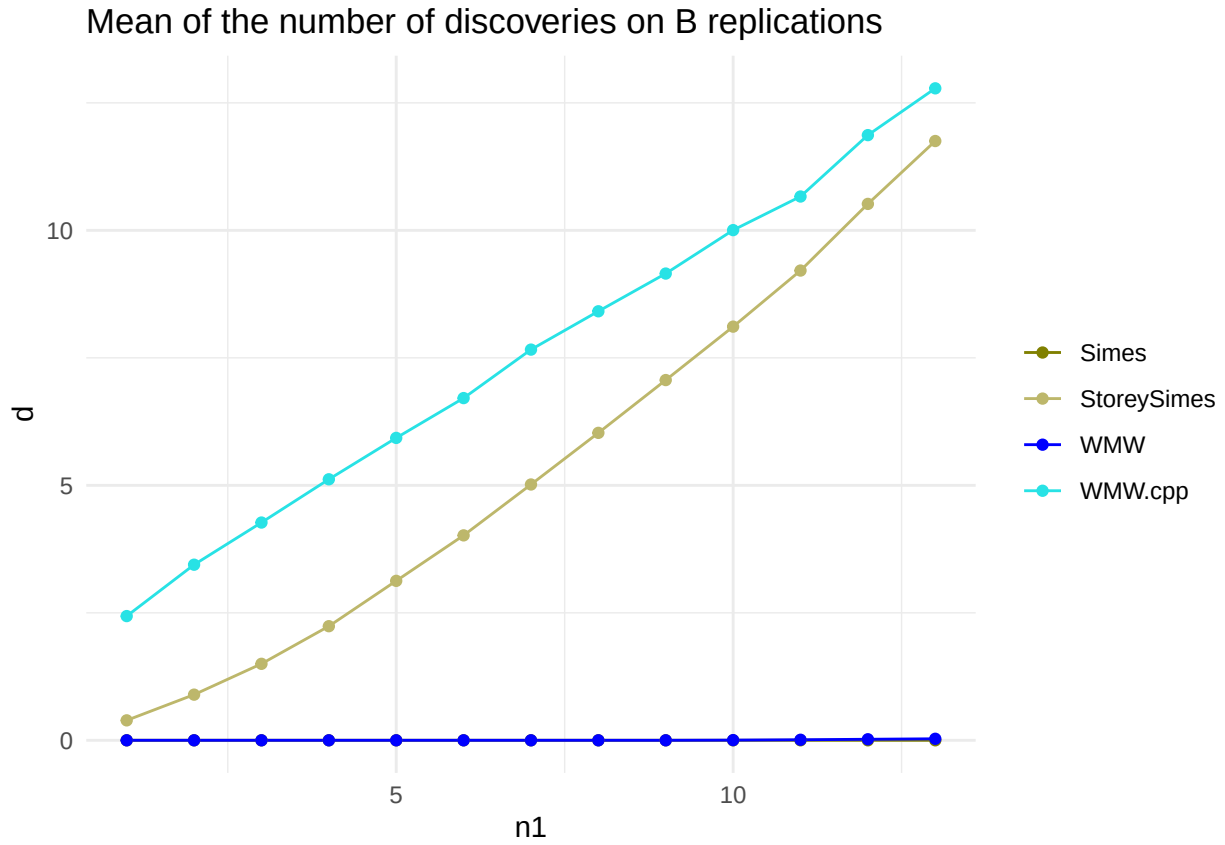
```
pow_StoSimes = vector()
pow_WMW = vector()

disc_Sim = vector()
disc_StoSimes = vector()
disc_WMW = vector()
disc_WMW.cpp = vector()

for(j in 1:length(n1s)){
  d_Sim[j] = results[[j]]$mean.lb.d[1]
  d_StoSimes[j] = results[[j]]$mean.lb.d[2]
  d_WMW[j] = results[[j]]$mean.lb.d[3]

  pow_Sim[j] = results[[j]]$mean.powerGlobalNull[1]
  pow_StoSimes[j] = results[[j]]$mean.powerGlobalNull[2]
  pow_WMW[j] = results[[j]]$mean.powerGlobalNull[3]

  disc_Sim[j] = results[[j]]$mean.n.disc[1]
  disc_StoSimes[j] = results[[j]]$mean.n.disc[2]
  disc_WMW[j] = results[[j]]$mean.n.disc[3]
  disc_WMW.cpp[j] = results[[j]]$mean.n.disc[4]
}

# Plot lower bound d
df <- data.frame(
  x = n1s,
  Simes = d_Sim,
  StoreySimes = d_StoSimes,
  WMW = 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( "#808000", "#BDB76B", 5)) +
  labs(x = "n1", y = "d", title = "Mean of the lower bound d on B replications") +
  theme_minimal() +
  theme(legend.title = element_blank())
```




```
# Plot power
dfpower <- data.frame(
  x = n1s,
  Simes = pow_Sim,
  StoreySimes = pow_StoSimes,
  WMW = pow_WMW
)
df_long_power <- tidyr::pivot_longer(dfpower, cols = -x, names_to = "group", values_to = "y")

ggplot(df_long_power, aes(x = x, y = y, color = group)) +
  geom_line() +
  geom_point()+
  scale_color_manual(values = c("#808000", "#BDB76B",5)) +
  labs(x = "n1", y = "power", title = "Mean of the power on B replications") +
  theme_minimal() +
  theme(legend.title = element_blank())
```




```
# Plot discoveries
df <- data.frame(
  x = n1s,
  Simes = disc_Sim,
  StoreySimes = disc_StoSimes,
  WMW.cpp = disc_WMW.cpp,
  WMW = disc_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("#808000", "#BDB76B", "blue", 5)) +
  labs(x = "n1", y = "d", title = "Mean of the number of discoveries on B replications") +
  theme_minimal() +
  theme(legend.title = element_blank())
```




```
n.disc.tablelist = list()
for(i in 1:length(n1s)){
  n.disc.tablelist[[i]] = matrix(ncol = 4, nrow = 2)
  colnames(n.disc.tablelist[[i]]) = c("Simes", "StoSimes", "WMW", "WMW.cpp")
  rownames(n.disc.tablelist[[i]]) = c("mean.n.disc", "mean.d")
  n.disc.tablelist[[i]][1,] = apply(results[[i]][["n.disc"]], MARGIN = 2, FUN = mean)
  n.disc.tablelist[[i]][2,] = results[[i]]$mean.lb.d[c(1,2,3,3)]
}

for(i in 1:length(n1s)){
  cat("\n")
  cat(paste("n1=", n1s[i]))
  cat("\n")
  print(n.disc.tablelist[[i]])
}
```

```
##
## n1= 1
##              Simes StoSimes    WMW WMW.cpp
## mean.n.disc 0.0000   0.3913 0.0000  2.4356
## mean.d      0.7026   0.3424 0.9326  0.9326
##
## n1= 2
##              Simes StoSimes    WMW WMW.cpp
## mean.n.disc 0.0000   0.8953 0.0000  3.4435
## mean.d      1.6079   1.2137 0.9916  0.9916
```

```
##
## n1= 3
##               Simes StoSimes    WMW WMW.cpp
## mean.n.disc 0.0000   1.4998 0.0000  4.2703
## mean.d      2.5519   2.2264 1.0425  1.0425
##
## n1= 4
##             Simes StoSimes    WMW WMW.cpp
## mean.n.disc 0.000   2.2375 0.0000  5.1182
## mean.d      3.516   3.2881 1.9854  1.9854
##
## n1= 5
##               Simes StoSimes    WMW WMW.cpp
## mean.n.disc 0.0000   3.1266 0.0000  5.9297
## mean.d      4.5034   4.3730 2.0728  2.0728
##
## n1= 6
##               Simes StoSimes   WMW WMW.cpp
## mean.n.disc 0.0000   4.0184 0.000  6.7106
## mean.d      5.4895   5.4317 2.962  2.9620
##
## n1= 7
##               Simes StoSimes    WMW WMW.cpp
## mean.n.disc 0.0000   5.0161 0.0000  7.6614
## mean.d      6.4763   6.4981 3.2689  3.2689
##
## n1= 8
##               Simes StoSimes    WMW WMW.cpp
## mean.n.disc 0.0000   6.0286 0.0000  8.4128
## mean.d      7.5078   7.5590 4.1586  4.1586
##
## n1= 9
##               Simes StoSimes    WMW WMW.cpp
## mean.n.disc 0.0000   7.0639 0.0000  9.1527
## mean.d      8.5702   8.6089 4.6579  4.6579
##
## n1= 10
##               Simes StoSimes    WMW WMW.cpp
## mean.n.disc 0.0000   8.1113 0.0036 10.0044
## mean.d      9.6097   9.6640 5.6426  5.6426
##
## n1= 11
##                Simes StoSimes    WMW WMW.cpp
## mean.n.disc  0.0000   9.2117 0.0107 10.6635
## mean.d      10.8875  10.7107 6.7409  6.7409
##
## n1= 12
##             Simes StoSimes    WMW WMW.cpp
## mean.n.disc   0.0  10.5185 0.0203 11.8673
## mean.d       11.9  11.7735 8.0102  8.0102
##
## n1= 13
##                Simes StoSimes    WMW WMW.cpp
## mean.n.disc  0.0000  11.7513 0.0297 12.7847
```

```
## mean.d      12.8982  12.8248 9.3178  9.3178
```

```
resShuttle.S7 = list("raw.res"=res,
                     #"k.est" = kest,
                     "compact.results" = results,
                     "n.disc.tablelist" = n.disc.tablelist)
save(resShuttle.S7,
     file="~/nout/trials/RealData/PowerStudy/FinalSimu/Shuttle/resShuttle.S7")
```