Comparison between different local tests: Simes, Simes with Storey and Wilcoxon-Mann-Whitney

20-04-2023

The aim is to compare 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 consider a null distribution F from which inliers come, while outliers come from the alternative distribution F^k with $k > 0$. If $k \in \mathbb{N}_{>0}$ we know that F^k is the distribution of the random variable defined as the maximum of k observations drawn from F. So, we consider a sample drawn from the mixture distribution

$$G = (1 - \theta)F + \theta F^k$$

where $\theta \in [0, 1]$ is the proportion of outliers.

Since we deal with conformal p-values, we are interested not really in the sample distribution, but rather in the scores distribution. In our simulation study we draw n observations for the train set, l for the calibration set and mk for the test set. All observations are drawn from a d-multivariate standard normal distribution with $d = 3$ and using the algorithm of ***isolation forest*** trained on the training samples we compute the scores for the calibration and test samples. Outlier observations are for simplicity the first m_1 observations of the test set and we consider the first m_1 blocks of k observations in the test set. For each i with $i = 1, \ldots, m_1$, the score related to observation i will be the maximum of the scores of the i-th block. For the last m_0 blocks, which correspond to inliers, the scores are randomly sampled from the k scores of the i-th block with $i = m_1 + 1, \ldots, m$.

```
library(Cairo)
library(nout)

# scores_from_mixture = function(k, raw_scores, m1){
#
#   if(theta>1 || theta<0){
#     stop("Error: argument theta should in [0,1] interval")
#   }
#
#   ll = length(raw_scores)
#   if(ll<k){
#     stop("Error: length of raw_scores is smaller than k.")
#   }
#
#   quotient = ll%/%k # is m
#   remainder = ll%%k
#
#   if(remainder != 0){
#     cat("Warning: length of raw_scores is not a multiple of k. Last ",
#         remainder, "elements of raw_scores will not be used.")
#   }
#
#   usable.raw_scores = raw_scores[1:(ll-remainder)]
```

```
#
#   #m1 = ifelse((theta*m)%%1!=0, round(theta*m), theta*m)
#
#   scores = rep(0, times = quotient)
#   outlier = rep(0, times = quotient)
#
#   if(m1==0){
#     for(i in 0:(m-1)){
#       scores[i+1] = sample(usable.raw_scores[(i*k+1):(i*k+k)], size=1)
#     }
#   }
#
#   if(m1==m){
#     for(i in 0:(m-1)){
#       scores[i+1] = max(usable.raw_scores[(i*k+1):(i*k+k)])
#       outlier[i+1]=T
#     }
#   }
#
#   if(0<m1 & m1<m){
#     for(i in 0:(m1-1)){
#       scores[i+1] = max(usable.raw_scores[(i*k+1):(i*k+k)])
#       outlier[i+1]=T
#     }
#
#     for(i in m1:(m-1)){
#       scores[i+1] = sample(usable.raw_scores[(i*k+1):(i*k+k)], size=1)
#     }
#   }
#
#   return(list("scores"=scores, "outlier"=outlier))
# }



scores_from_mixture = function(k, raw_scores, m, m1){

  if(m1==0)  scores = raw_scores

  if(m1==m){
    scores = sapply(0:(m1-1), function(i) max(raw_scores[(i*k+1):(i*k+k)]))
  }

  if(0<m1 & m1<m){
    scores.out = sapply(0:(m1-1), function(i) max(raw_scores[(i*k+1):(i*k+k)]))
    scores.in = raw_scores[(k*m1+1):length(raw_scores)]
    scores = c(scores.out, scores.in)
  }

  return("scores"=scores)
}
```

```r
simuLMPI = function(B=10^4, n, l, m, d = 3, k = 2, theta, alpha = m/(l+1)){

  train = mvtnorm::rmvnorm(n=n, mean=rep(0,d))
  iso.fo = isotree::isolation.forest(train, ndim=d, ntrees=10, nthreads=1,
                                     scoring_metric = "depth", output_score = TRUE)

  crit=critWMW(m=m, n=n, alpha=alpha)
  m1 = round(theta*m)
  m0 = m-m1

  d_WMW = rep(0,B)
  d_Simes = rep(0,B)
  d_StoSimes = rep(0,B)
  d_BH = rep(0,B)
  d_StoBH = rep(0,B)

  for(b in 1:B){
    cal = mvtnorm::rmvnorm(n=l, mean=rep(0,d))
    te = mvtnorm::rmvnorm(n=k*m1+m0, mean=rep(0,d))

    S_cal = isotree::predict.isolation_forest(iso.fo$model, cal, type = "score")
    rawS_te = isotree::predict.isolation_forest(iso.fo$model, te, type = "score")
    S_te = scores_from_mixture(k=k, raw_scores=rawS_te, m=m, m1=m1)

    d_WMW[b] = d_mannwhitney(S_X=S_cal, S_Y=S_te, crit=crit)
    d_Simes[b] = d_Simes(S_X=S_cal, S_Y=S_te, alpha=alpha)
    d_StoSimes[b] = d_StoreySimes(S_X=S_cal, S_Y=S_te, alpha=alpha)
    d_BH[b] = d_benjhoch(S_X=S_cal, S_Y=S_te, alpha=alpha)
    d_StoBH[b] = d_StoreyBH(S_X=S_cal, S_Y=S_te, alpha=alpha)
  }

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

  powerGlobalNull = as.data.frame(cbind("d_BH"=d_BH>0, "d_StoBH"=d_StoBH>0, "d_Simes"=d_Simes>0,
                               "d_StoSimes"=d_StoSimes>0, "d_WMW"=d_WMW>0))
  colnames(powerGlobalNull) = c("BH", "BHSto", "CTSim", "CTSimSto", "CTWMW")
  mean.powerGlobalNull = apply(powerGlobalNull, MARGIN = 2, FUN = mean)

  return(list("discoveries"=discov, "mean.discoveries" = mean.discov,
              "powerGlobalNull"=powerGlobalNull, "mean.powerGlobalNull"=mean.powerGlobalNull,
              "theta"=theta, "alpha"=alpha))
}
```

K=2

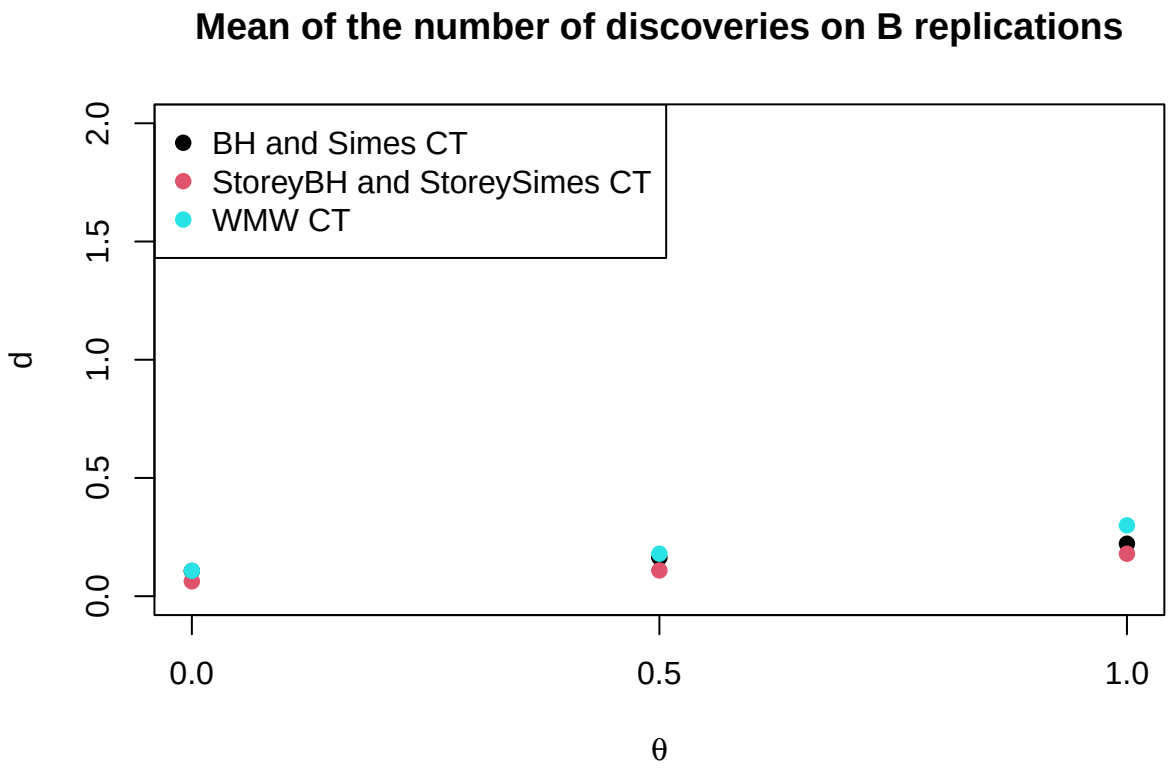
```
set.seed(321)

# Initializing parameters
B=10^5
n = 19
l = 19
m = 2
d = 3
k = 2
alpha = m/(l+1)
m1s = seq(from=0, to=m, by=1)
thetas = m1s/m

# Results
res = lapply(thetas, function(theta) simuLMPI(B=B, n=n, l=l, m=m, d = d,
                                              k = k, theta, alpha = m/(l+1)))

# Storing results
store_res = list("mean.discov" = matrix(nrow=length(thetas), ncol = 5),
                 "mean.powerGlobalNull" = matrix(nrow=length(thetas), ncol = 5))
row.names = rep(NA, times=length(thetas))
for(i in 1:length(thetas)){
  row.names[i] = paste("theta =",thetas[i])
}
rownames(store_res$mean.discov) = row.names
colnames(store_res$mean.discov) = c("BH", "StoBH", "Simes", "StoSimes", "WMW")
rownames(store_res$mean.powerGlobalNull) = row.names
colnames(store_res$mean.powerGlobalNull) = c("BH", "StoBH", "Simes", "StoSimes", "WMW")


for(i in 1:length(res)){
  store_res$mean.discov[i,] = res[[i]]$mean.discov
  store_res$mean.powerGlobalNull[i,] = res[[i]]$mean.powerGlobalNull
}

store_res$mean.discov
```

```
##                   BH   StoBH   Simes StoSimes     WMW
## theta = 0   0.10650 0.06274 0.10650  0.05328 0.10728
## theta = 0.5 0.16335 0.10872 0.16335  0.09221 0.17962
## theta = 1   0.22188 0.17951 0.22188  0.14926 0.29946
```

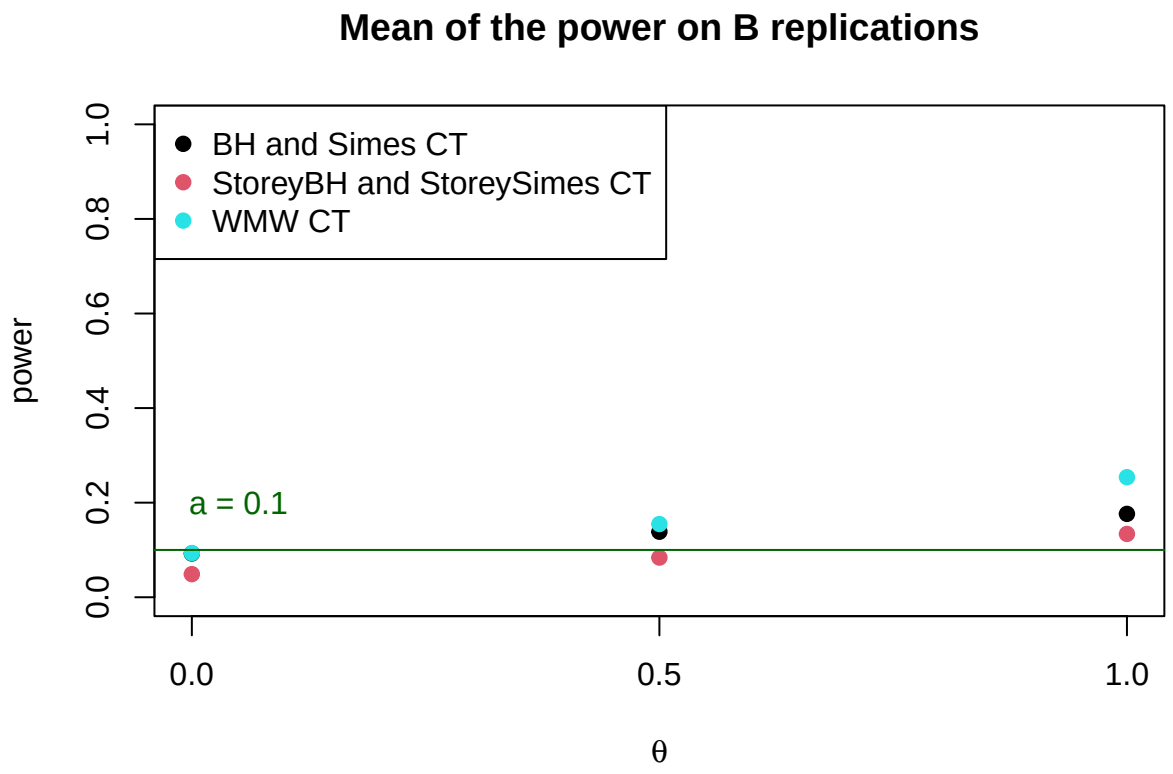
```
store_res$mean.powerGlobalNull
```

```
##                   BH   StoBH   Simes StoSimes     WMW
## theta = 0   0.09258 0.04882 0.09258  0.04882 0.09336
## theta = 0.5 0.13853 0.08390 0.13853  0.08390 0.15480
## theta = 1   0.17627 0.13390 0.17627  0.13390 0.25385
```

```
plot(x = thetas, y = store_res$mean.discov[,1], col = 1, ylab = "d",
     xlab = expression(theta), ylim=c(0,m), pch=19, xaxt = "n",
     main = "Mean of the number of discoveries on B replications")
points(x = thetas, y = store_res$mean.discov[,2], col = 2, pch=19)
points(x = thetas, y = store_res$mean.discov[,5], col = 5, pch=19)
axis(side = 1, at = thetas)
legend("topleft", pch = 19, col = c(1,2,5),
       legend =c("BH and Simes CT", "StoreyBH and StoreySimes CT", "WMW CT"))
```




```
plot(x = thetas, y = store_res$mean.powerGlobalNull[,1], col = 1, ylab = "power",
     xlab = expression(theta), ylim=c(0,1), pch = 19, xaxt = "n",
     main = "Mean of the power on B replications")
points(x = thetas, y = store_res$mean.powerGlobalNull[,2], col = 2, pch=19)
points(x = thetas, y = store_res$mean.powerGlobalNull[,5], col = 5, pch=19)
abline(h=alpha, col = "darkgreen")
text(x=0.05, y=alpha+0.1, labels = paste("\u03B1","=", alpha), col = "darkgreen")
axis(side = 1, at = thetas)
legend("topleft", pch = 19, col = c(1,2,5),
       legend = c("BH and Simes CT", "StoreyBH and StoreySimes CT", "WMW CT"))
```



K=5

```
set.seed(321)

# Initializing parameters
B=10^5
n = 19
l = 19
m = 2
d = 3
k = 5
alpha = m/(n+1)
m1s = seq(from=0, to=m, by=1)
thetas = m1s/m

# Results
res = lapply(thetas, function(theta) simuLMPI(B=B, n=n, l=l, m=m, d = d,
                                              k = k, theta, alpha = m/(l+1)))

# Storing results
store_res = list("mean.discov" = matrix(nrow=length(thetas), ncol = 5),
                 "mean.powerGlobalNull" = matrix(nrow=length(thetas), ncol = 5))
row.names = rep(NA, times=length(thetas))
for(i in 1:length(thetas)){
```

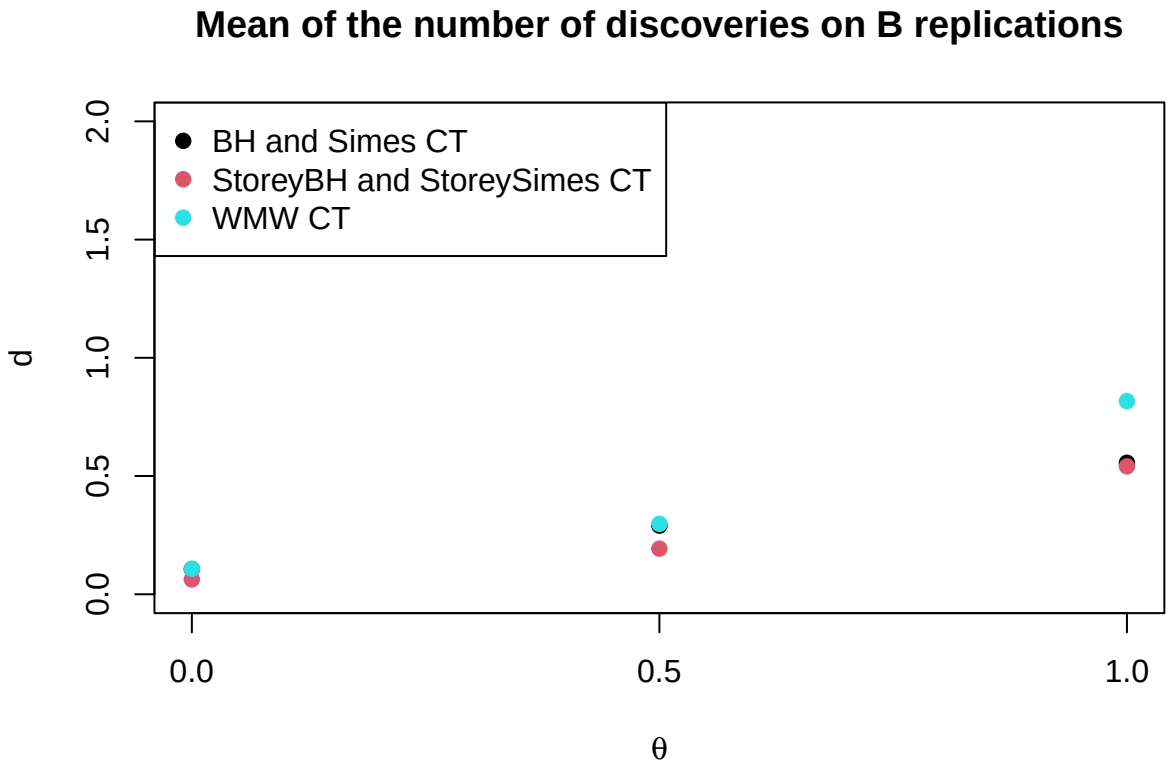
```
  row.names[i] = paste("theta =",thetas[i])
}
rownames(store_res$mean.discov) = row.names
colnames(store_res$mean.discov) = c("BH", "StoBH", "Simes", "StoSimes", "WMW")
rownames(store_res$mean.powerGlobalNull) = row.names
colnames(store_res$mean.powerGlobalNull) = c("BH", "StoBH", "Simes", "StoSimes", "WMW")


for(i in 1:length(res)){
  store_res$mean.discov[i,] = res[[i]]$mean.discov
  store_res$mean.powerGlobalNull[i,] = res[[i]]$mean.powerGlobalNull
}

store_res$mean.discov
```

```
##                  BH   StoBH   Simes StoSimes     WMW
## theta = 0   0.10650 0.06274 0.10650  0.05328 0.10728
## theta = 0.5 0.29008 0.19287 0.29008  0.16195 0.29749
## theta = 1   0.55619 0.54073 0.55619  0.43328 0.81660
```

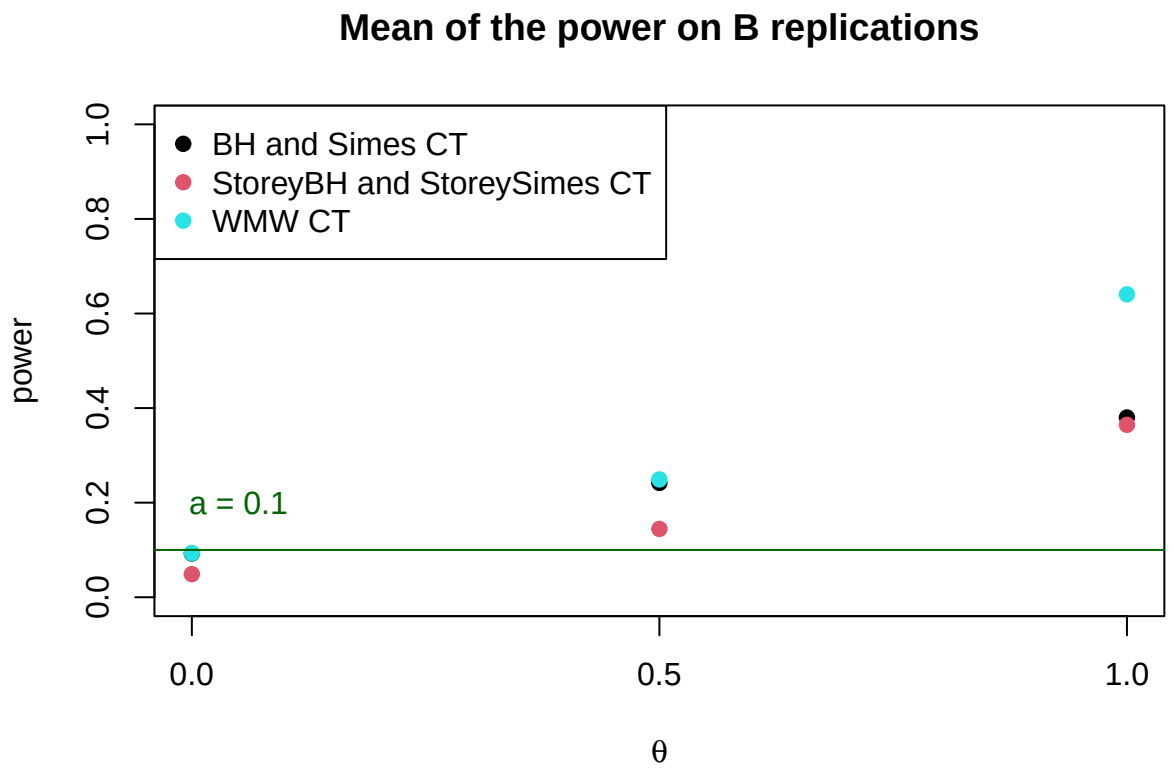
```
store_res$mean.powerGlobalNull
```

```
##                  BH   StoBH   Simes StoSimes     WMW
## theta = 0   0.09258 0.04882 0.09258  0.04882 0.09336
## theta = 0.5 0.24168 0.14447 0.24168  0.14447 0.24909
## theta = 1   0.38013 0.36467 0.38013  0.36467 0.64054
```

```
plot(x = thetas, y = store_res$mean.discov[,1], col = 1, ylab = "d",
     xlab = expression(theta), ylim=c(0,m), pch=19, xaxt = "n",
     main = "Mean of the number of discoveries on B replications")
points(x = thetas, y = store_res$mean.discov[,2], col = 2, pch=19)
points(x = thetas, y = store_res$mean.discov[,5], col = 5, pch=19)
axis(side = 1, at = thetas)
legend("topleft", pch = 19, col = c(1,2,5),
       legend =c("BH and Simes CT", "StoreyBH and StoreySimes CT", "WMW CT"))
```




```
plot(x = thetas, y = store_res$mean.powerGlobalNull[,1], col = 1, ylab = "power",
     xlab = expression(theta), ylim=c(0,1), pch = 19, xaxt = "n",
     main = "Mean of the power on B replications")
points(x = thetas, y = store_res$mean.powerGlobalNull[,2], col = 2, pch=19)
points(x = thetas, y = store_res$mean.powerGlobalNull[,5], col = 5, pch=19)
abline(h=alpha, col = "darkgreen")
text(x=0.05, y=alpha+0.1, labels = paste("\u03B1","=", alpha), col = "darkgreen")
axis(side = 1, at = thetas)
legend("topleft", pch = 19, col = c(1,2,5),
       legend = c("BH and Simes CT", "StoreyBH and StoreySimes CT", "WMW CT"))
```



K=5

```
set.seed(321)

# Initializing parameters
B=10^5
n = 19
l = 19
m = 2
d = 3
k = 5
alpha = m/(l+1)
m1s = seq(from=0, to=m, by=1)
thetas = m1s/m
# Results
res = lapply(thetas, function(theta) simuLMPI(B=B, n=n, l=l, m=m, d = d,
                                              k = k, theta, alpha = m/(l+1)))

# Storing results
store_res = list("mean.discov" = matrix(nrow=length(thetas), ncol = 5),
                 "mean.powerGlobalNull" = matrix(nrow=length(thetas), ncol = 5))
row.names = rep(NA, times=length(thetas))
for(i in 1:length(thetas)){
  row.names[i] = paste("theta =",thetas[i])
```

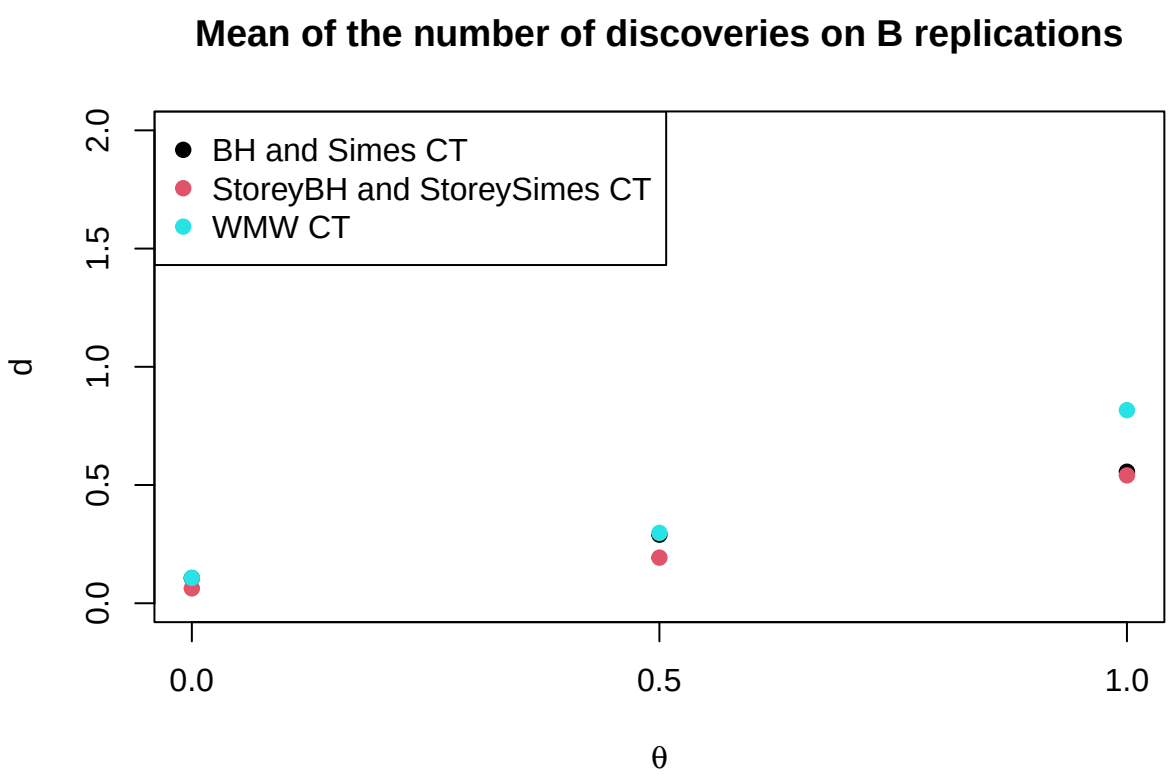
```
}
rownames(store_res$mean.discov) = row.names
colnames(store_res$mean.discov) = c("BH", "StoBH", "Simes", "StoSimes", "WMW")
rownames(store_res$mean.powerGlobalNull) = row.names
colnames(store_res$mean.powerGlobalNull) = c("BH", "StoBH", "Simes", "StoSimes", "WMW")


for(i in 1:length(res)){
  store_res$mean.discov[i,] = res[[i]]$mean.discov
  store_res$mean.powerGlobalNull[i,] = res[[i]]$mean.powerGlobalNull
}

store_res$mean.discov
```

```
##                  BH   StoBH   Simes StoSimes     WMW
## theta = 0   0.10650 0.06274 0.10650  0.05328 0.10728
## theta = 0.5 0.29008 0.19287 0.29008  0.16195 0.29749
## theta = 1   0.55619 0.54073 0.55619  0.43328 0.81660
```

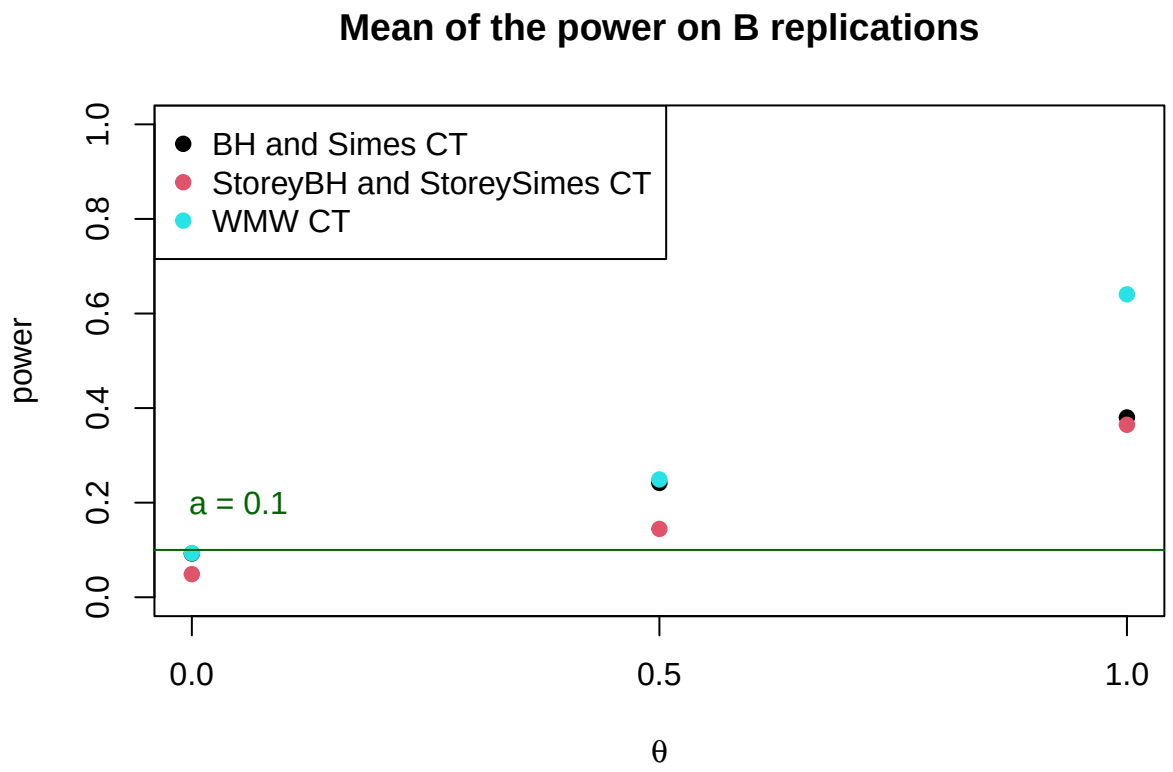
```
store_res$mean.powerGlobalNull
```

```
##                  BH   StoBH   Simes StoSimes     WMW
## theta = 0   0.09258 0.04882 0.09258  0.04882 0.09336
## theta = 0.5 0.24168 0.14447 0.24168  0.14447 0.24909
## theta = 1   0.38013 0.36467 0.38013  0.36467 0.64054
```

```
plot(x = thetas, y = store_res$mean.discov[,1], col = 1, ylab = "d",
     xlab = expression(theta), ylim=c(0,m), pch=19, xaxt = "n",
     main = "Mean of the number of discoveries on B replications")
points(x = thetas, y = store_res$mean.discov[,2], col = 2, pch=19)
points(x = thetas, y = store_res$mean.discov[,5], col = 5, pch=19)
axis(side = 1, at = thetas)
legend("topleft", pch = 19, col = c(1,2,5),
       legend =c("BH and Simes CT", "StoreyBH and StoreySimes CT", "WMW CT"))
```




```
plot(x = thetas, y = store_res$mean.powerGlobalNull[,1], col = 1, ylab = "power",
     xlab = expression(theta), ylim=c(0,1), pch = 19, xaxt = "n",
     main = "Mean of the power on B replications")
points(x = thetas, y = store_res$mean.powerGlobalNull[,2], col = 2, pch=19)
points(x = thetas, y = store_res$mean.powerGlobalNull[,5], col = 5, pch=19)
abline(h=alpha, col = "darkgreen")
text(x=0.05, y=alpha+0.1, labels = paste("\u03B1","=", alpha), col = "darkgreen")
axis(side = 1, at = thetas)
legend("topleft", pch = 19, col = c(1,2,5),
       legend = c("BH and Simes CT", "StoreyBH and StoreySimes CT", "WMW CT"))
```


Mean of the power on B replications
BH and Simes CT
StoreyBH and StoreySimes CT
WMW CT
a = 0.1
power
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
θ