Digits dataset

02-05-2023

The aim is to compare on Digits dataset (available at http://odds.cs.stonybrook.edu/pendigits-dataset) 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.

Digits dataset consists of 6870 observations, among which $n_{inliers} = 6714$ items are inliers and the remaining $n_{outliers} = 156$ are outliers. We will denote by n, l, m respectively the train set, the calibration set and the test set size. And reproducing the same setting as in [1], we have that $n + l = n_{inliers}/2$, $l = min\{2000, n/2\}$ and $m = min\{2000, n/3\}$. In the case of Digits dataset we obtain

$$n + l = 6714/2, \quad l = n/3, \quad m = n/3$$

from which $n = 2517.75, \quad l = 839.25, \quad m = 839.25$. Arbitrarily, we choose to set

$$n = 2518, \quad l = 839, \quad m = 839.$$

The nominal level α is set equal to 0.2, as in [1].

R functions and libraries

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


sim_realdata = function(B, dataset, m1, m, n, l, in_index,
                        out_index=NULL, alpha=m/(l+1), lambda = 0.5){

  m0=m-m1
  if(m1!=0 & is.null(out_index)){
    stop("Error: arg out_index must be initialized.")
  }

  # if(m!=(m1+m0)){
  #   stop("Error: equation m=m1+m0 must be verified.")
  # }

  if(m1!=0){
    tr_ind = sample(in_index, size = n)
    tr = dataset[tr_ind,]
    iso.fo = isolation.forest(tr, ndim=ncol(dataset), ntrees=10, nthreads=1,
                              scoring_metric = "depth", output_score = TRUE)
    in_index2 = setdiff(in_index, tr_ind)

    crit=critWMW(m=m, n=l, alpha=alpha)
```

```
  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_ind = sample(in_index2, size = l)
    in_index3 = setdiff(in_index2, cal_ind)
    tein_ind = sample(in_index3, size = m0)
    teout_ind = sample(out_index, size = m1)

    cal = dataset[cal_ind,]
    te = dataset[c(tein_ind, teout_ind),]

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

    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)
  }
}

else{
  tr_ind = sample(in_index, size = n)
  tr = dataset[tr_ind,]
  iso.fo = isolation.forest(tr, ndim=ncol(dataset), ntrees=10, nthreads=1,
                            scoring_metric = "depth", output_score = TRUE)
  in_index2 = setdiff(in_index, tr_ind)

  crit=critWMW(m=m, n=l, alpha=alpha)

  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_ind = sample(in_index2, size = l)
    in_index3 = setdiff(in_index2, cal_ind)
    te_ind = sample(in_index3, size = m0)

    cal = dataset[cal_ind,]
    te = dataset[te_ind,]

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

    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,
              "m1"=m1, "alpha"=alpha))
}




sim_realdataTFO = function(B, dataset, m1, m, n, l, in_index,
                           out_index=NULL, alpha=m/(l+1), lambda = 0.5){

  m0=m-m1
  if(m1!=0 & is.null(out_index)){
    stop("Error: arg out_index must be initialized.")
  }

  # if(m!=(m1+m0)){
  #   stop("Error: equation m=m1+m0 must be verified.")
  # }

  if(m1!=0){
    tr_ind = sample(in_index, size = n)
    tr = dataset[tr_ind,]
    iso.fo = isolation.forest(tr, ndim=ncol(dataset), ntrees=10, nthreads=1,
                              scoring_metric = "depth", output_score = TRUE)
    in_index2 = setdiff(in_index, tr_ind)
    cal_ind = sample(in_index2, size = l)
    cal = dataset[cal_ind,]
    S_cal = predict.isolation_forest(iso.fo$model, cal, type = "score")

    in_index3 = setdiff(in_index2, cal_ind)

    crit=critWMW(m=m, n=l, alpha=alpha)
```

```
  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){
    tein_ind = sample(in_index3, size = m0)
    teout_ind = sample(out_index, size = m1)
    te = dataset[c(tein_ind, teout_ind),]
    S_te = predict.isolation_forest(iso.fo$model, te, type = "score")

    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)
  }
}

else{
  tr_ind = sample(in_index, size = n)
  tr = dataset[tr_ind,]
  iso.fo = isolation.forest(tr, ndim=ncol(dataset), ntrees=10, nthreads=1,
                            scoring_metric = "depth", output_score = TRUE)
  in_index2 = setdiff(in_index, tr_ind)
  cal_ind = sample(in_index2, size = l)
  cal = dataset[cal_ind,]
  S_cal = predict.isolation_forest(iso.fo$model, cal, type = "score")
  in_index3 = setdiff(in_index2, cal_ind)


  crit=critWMW(m=m, n=l, alpha=alpha)

  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){
    te_ind = sample(in_index3, size = m0)
    te = dataset[te_ind,]
    S_te = predict.isolation_forest(iso.fo$model, te, type = "score")

    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,
              "m1"=m1, "alpha"=alpha))
}
```

Load the data and set the parameters ad described above.

```
set.seed(321)

# Initializing parameters
n = 2518
l = 839
m = 839
alpha = 0.2

data = readMat("G:\\Il mio Drive\\PHD\\Progetto di ricerca\\Conformal Inference Project\\Simulazioni\\7
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)
```

Fixed train set, random calibration and test sets

We fixed the train set on which we train the isolation forest algorithm and we generate $B = 100$ 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 estimator and by closed testing using Wilcoxon-Mann-Whitney local test.

All inliers

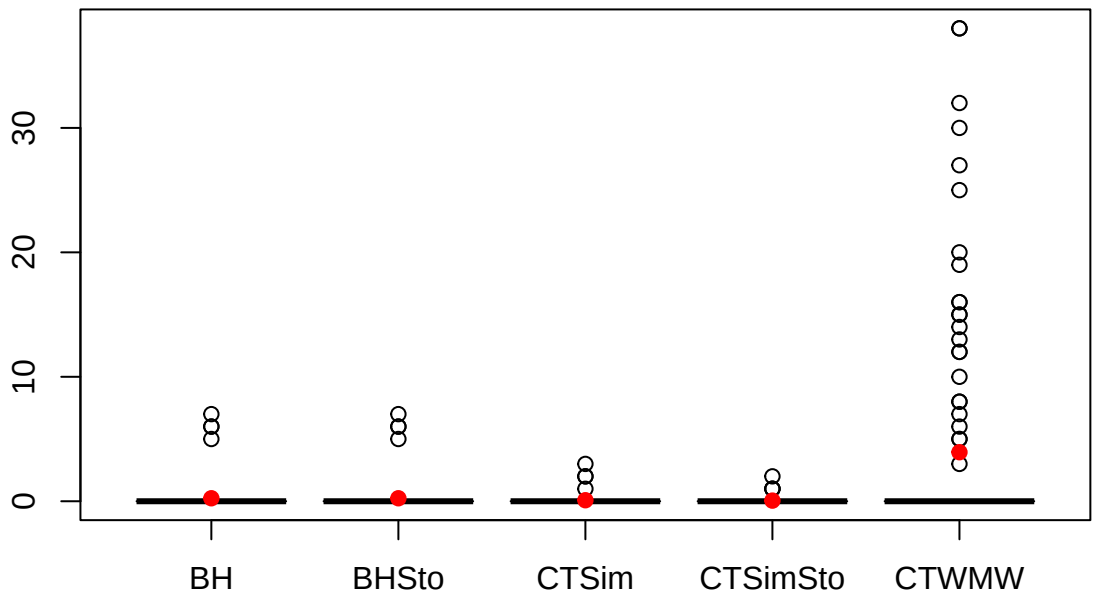
We now set the proportion of inliers equal to 1, so that the number of outliers $m_1 = 0$.

```
B=100
m1=0

tic()
res = sim_realdata(B=B, in_index=in_ind, out_index=out_ind, dataset=dataset,
                   alpha=alpha,l=l, n=n, m=m, m1=m1)
toc()
```

```
## 9.48 sec elapsed
```

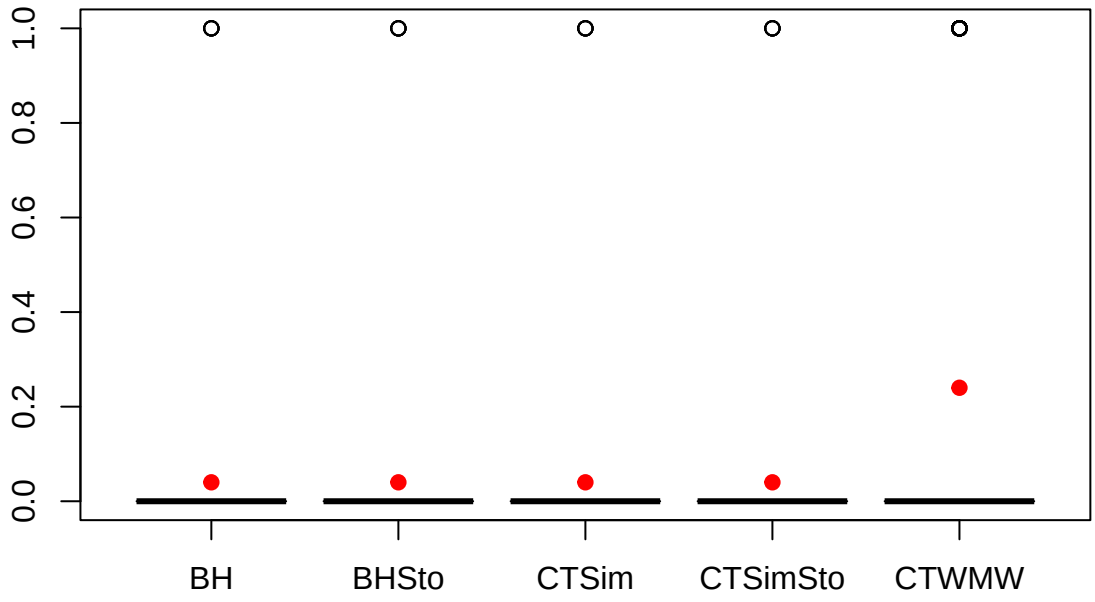
```
boxplot(res$discoveries, main="Distribution of the number of discoveries")
points(x=1:5, y=res$mean.discoveries, pch=19, col="red")
```




```
res$mean.discoveries
```

```
##       BH    BHSto    CTSim CTSimSto    CTWMW
##     0.24     0.24     0.08     0.05     3.94
```

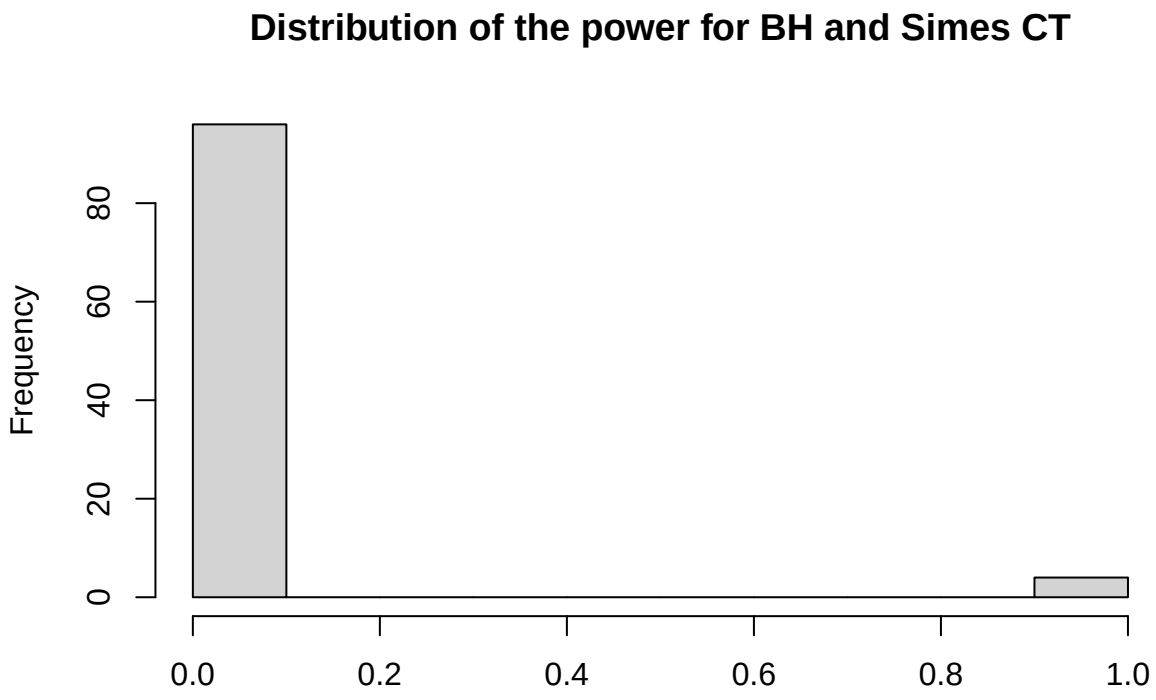
```
boxplot(res$powerGlobalNull, main="Distribution of the power")
points(x=1:5, y=res$mean.powerGlobalNull, pch=19, col="red")
```



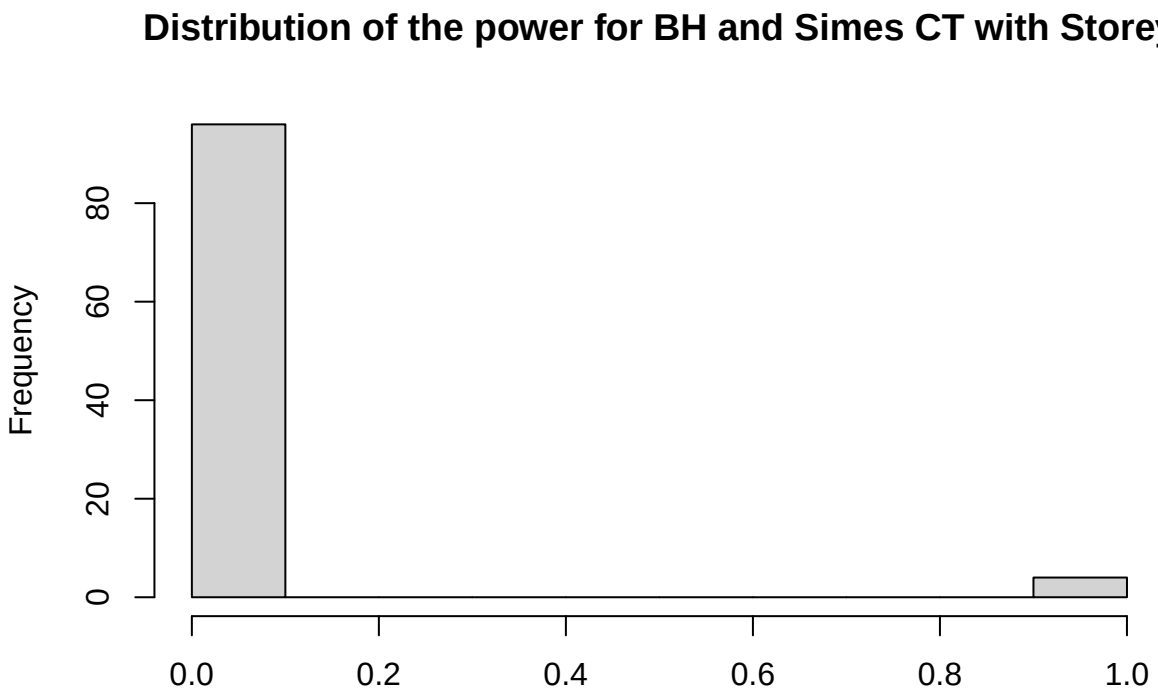

```
res$mean.powerGlobalNull
```

```
##       BH    BHSto    CTSim CTSimSto    CTWMW
##     0.04     0.04     0.04     0.04     0.24
```

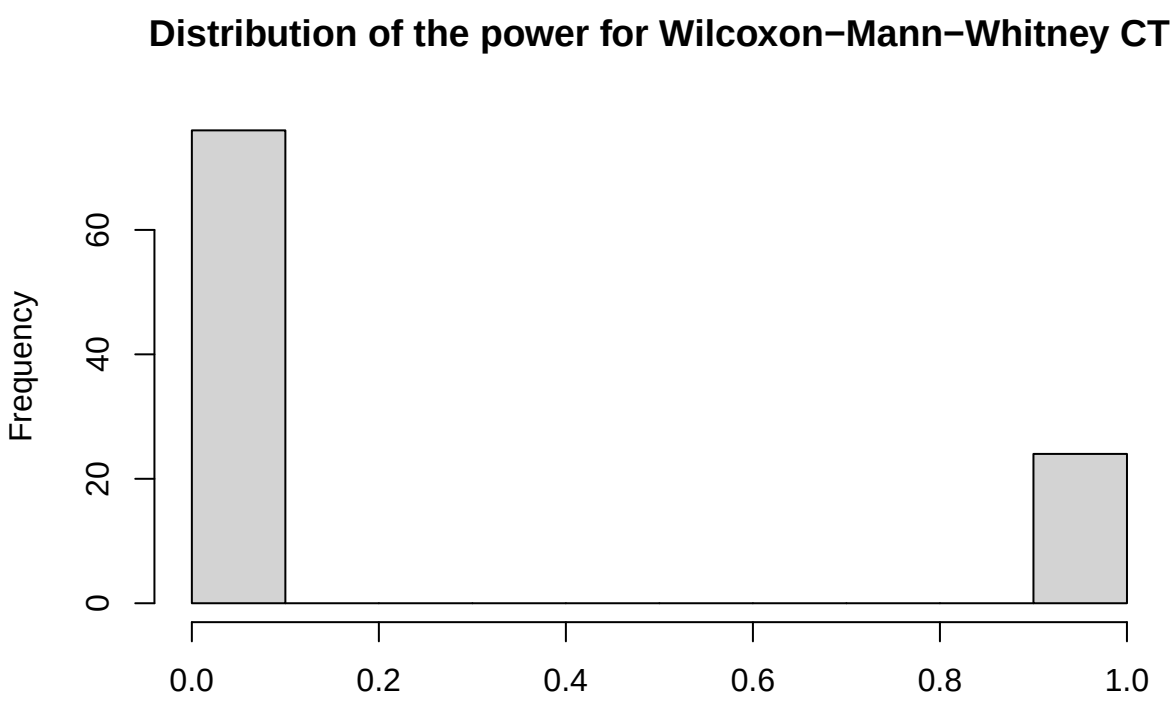
```
hist(as.integer(res$powerGlobalNull[[1]]), xlab="",
     main="Distribution of the power for BH and Simes CT")
```




```
hist(as.integer(res$powerGlobalNull[[2]]), xlab="",
     main="Distribution of the power for BH and Simes CT with Storey")
```




```
hist(as.integer(res$powerGlobalNull[[5]]), xlab="",
     main="Distribution of the power for Wilcoxon-Mann-Whitney CT")
```




```
# resDigits_v2 = res
# save(resDigits_v2,
#      file="C:/Users/c.magnani9/Documents/nout/trials/RealData/PowerStudy/resDigits_v2")
```

10% outliers

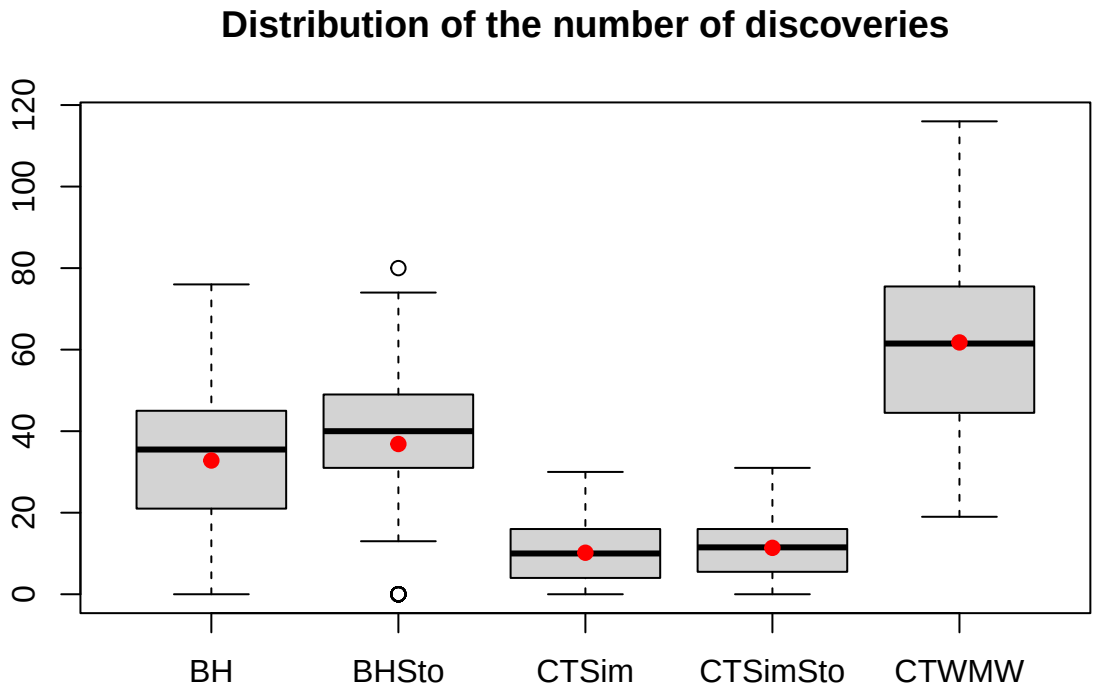
We now set the proportion of inliers equal to 0.9. Referring to Digits dataset we have that the number of inliers is $m_0 = 755.1$ and the number of outliers is $m_1 = 83.9$. Arbitrarily, we choose to set $m_0 = 755$ and $m_1 = 84$.

```
B=100
m1=84

tic()
res = sim_realdata(B=B, in_index=in_ind, out_index=out_ind, dataset=dataset,
                   alpha=alpha,l=l, n=n, m=m, m1=m1)
toc()
```

```
## 9.96 sec elapsed
```

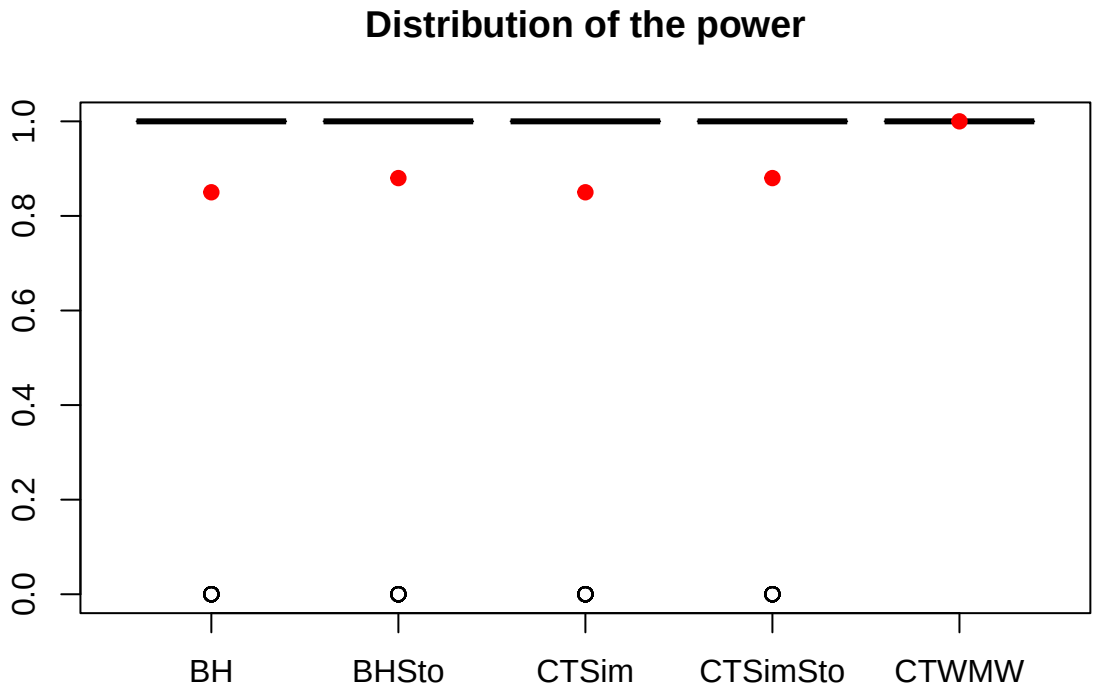
```
boxplot(res$discoveries, main="Distribution of the number of discoveries")
points(x=1:5, y=res$mean.discoveries, pch=19, col="red")
```




```
res$mean.discoveries
```

```
##       BH    BHSto    CTSim CTSimSto    CTWMW
##    32.80    36.85    10.18    11.38    61.77
```

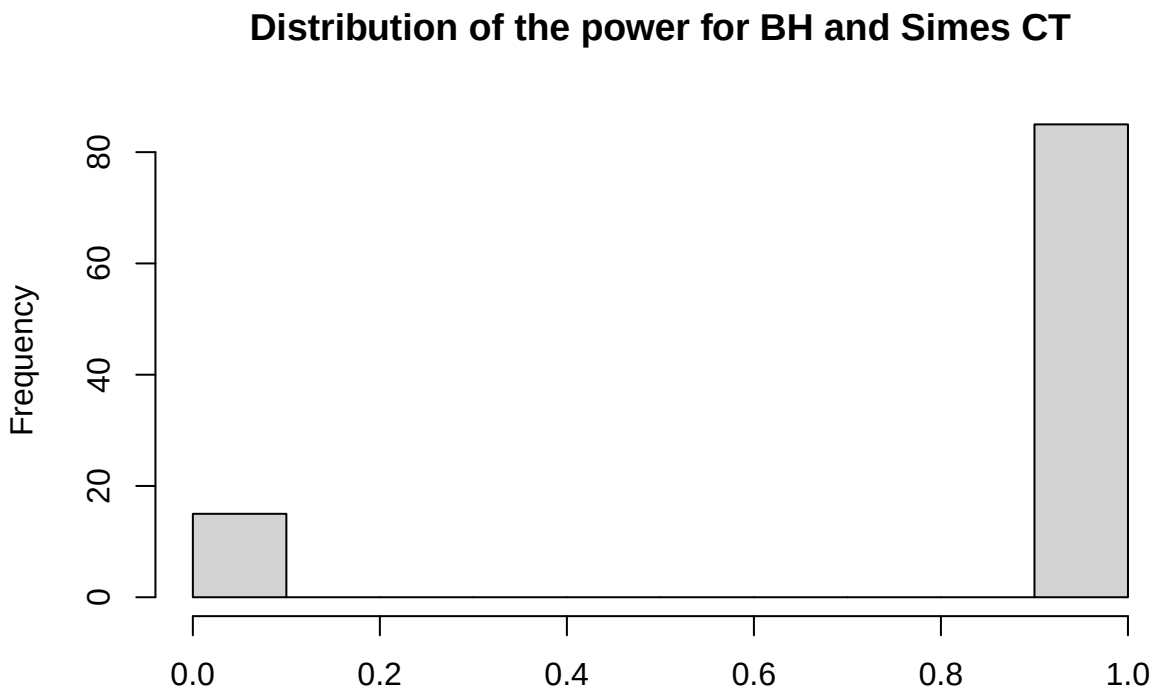
```
boxplot(res$powerGlobalNull, main="Distribution of the power")
points(x=1:5, y=res$mean.powerGlobalNull, pch=19, col="red")
```



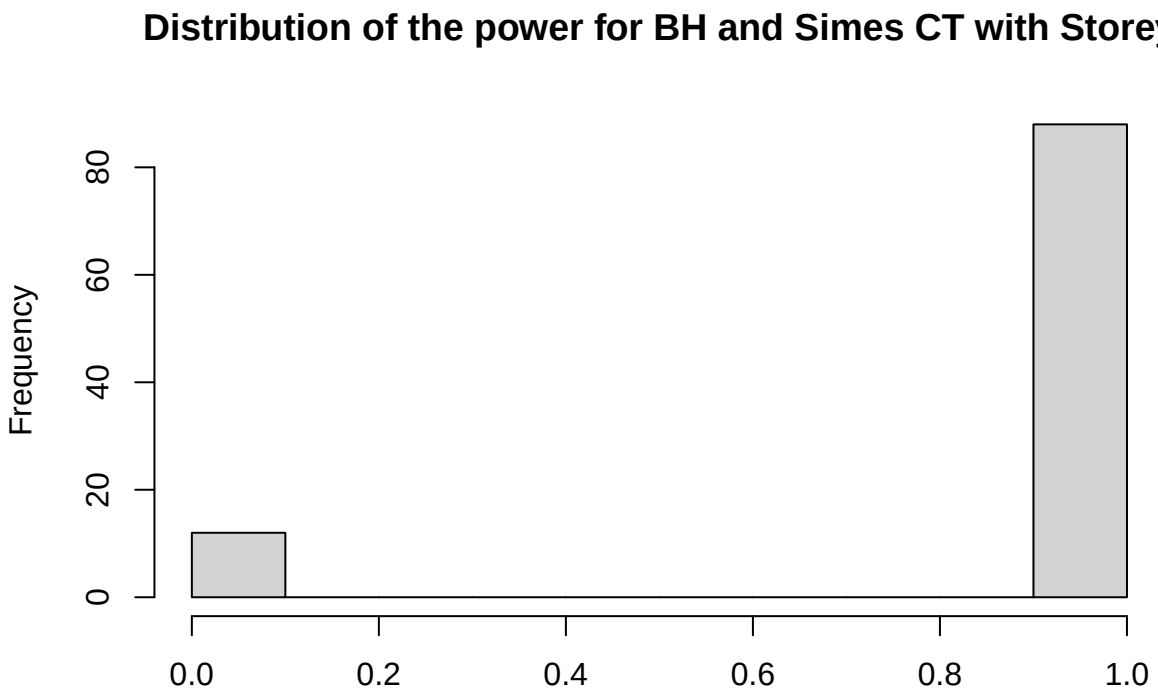

```
res$mean.powerGlobalNull
```

```
##       BH    BHSto    CTSim CTSimSto    CTWMW
##     0.85     0.88     0.85     0.88     1.00
```

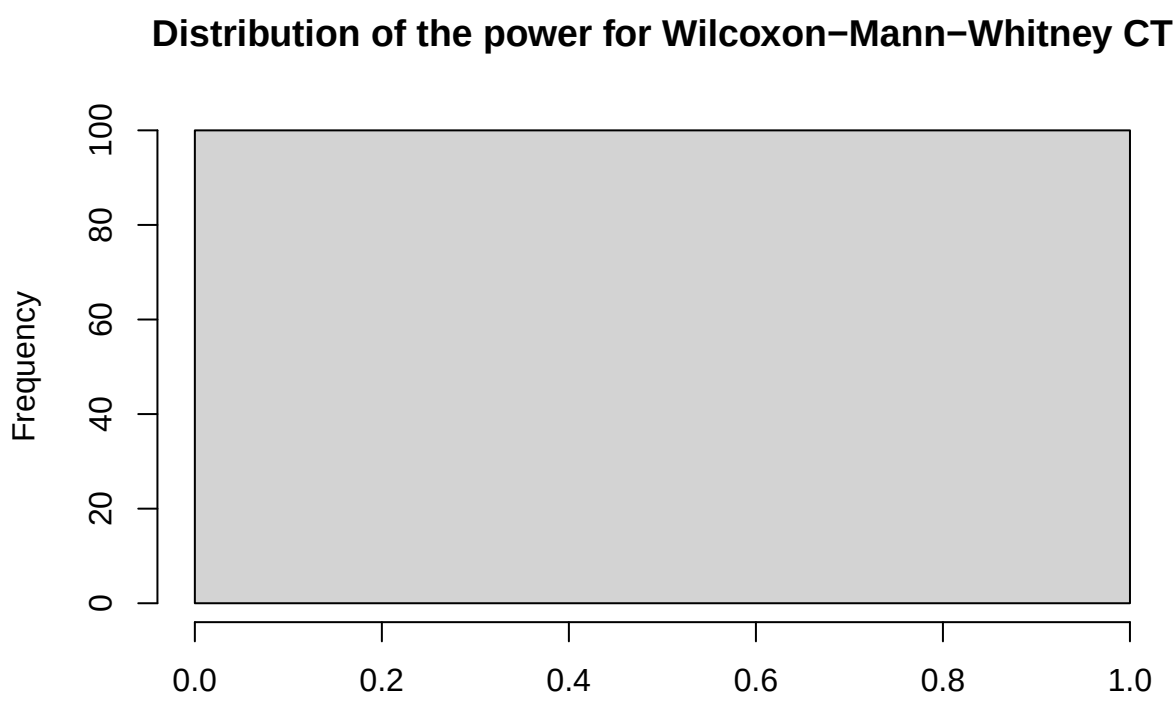
```
hist(as.integer(res$powerGlobalNull[[1]]), xlab="",
     main="Distribution of the power for BH and Simes CT")
```




```
hist(as.integer(res$powerGlobalNull[[2]]), xlab="",
     main="Distribution of the power for BH and Simes CT with Storey")
```




```
hist(as.integer(res$powerGlobalNull[[5]]), xlab="",
     main="Distribution of the power for Wilcoxon-Mann-Whitney CT")
```




```
# resDigits_v2 = res
# save(resDigits_v2,
#      file="C:/Users/c.magnani9/Documents/nout/trials/RealData/PowerStudy/resDigits_v2")
```

Fixed train and calibration sets, random test set

In [1] the train set and the calibration set are seen as fixed and only the test set is random. For a fixed pair of train and calibration set, $B = 100$ test sets are generated. 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 estimator and by closed testing using Wilcoxon-Mann-Whitney local test.

All inliers

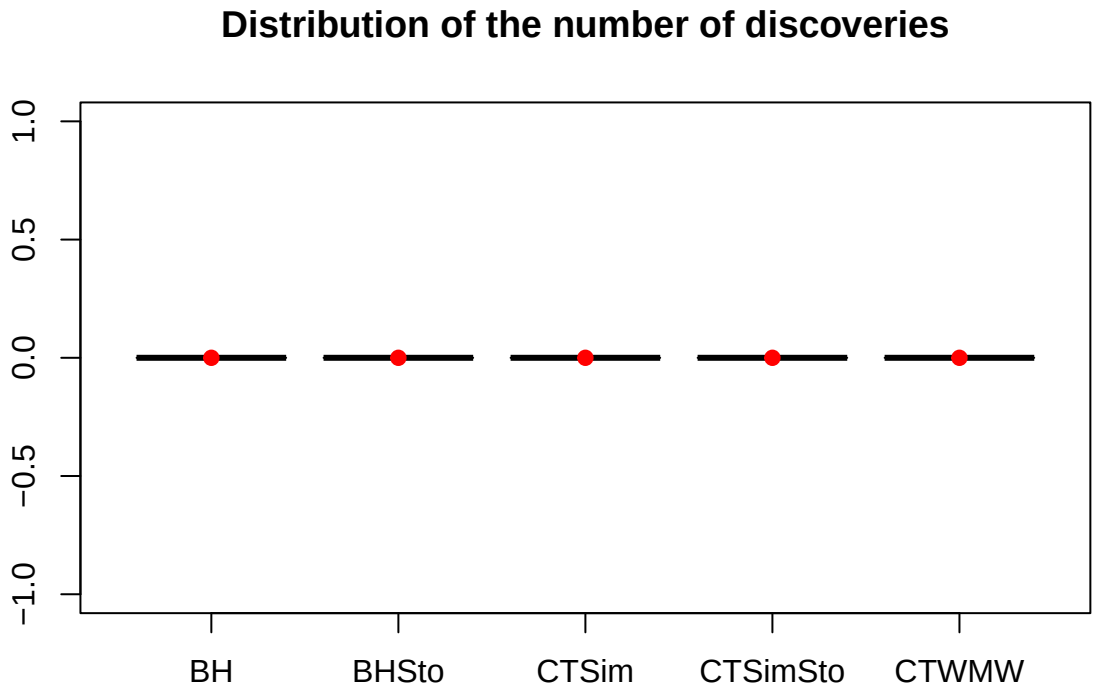
We now set the proportion of inliers equal to 1, so that the number of outliers $m_1 = 0$.

```
B=100
m1=0

tic()
res = sim_realdataTFO(B=B, in_index=in_ind, out_index=out_ind, dataset=dataset,
                      alpha=alpha,l=l, n=n, m=m, m1=m1)
toc()
```

```
## 9.78 sec elapsed
```

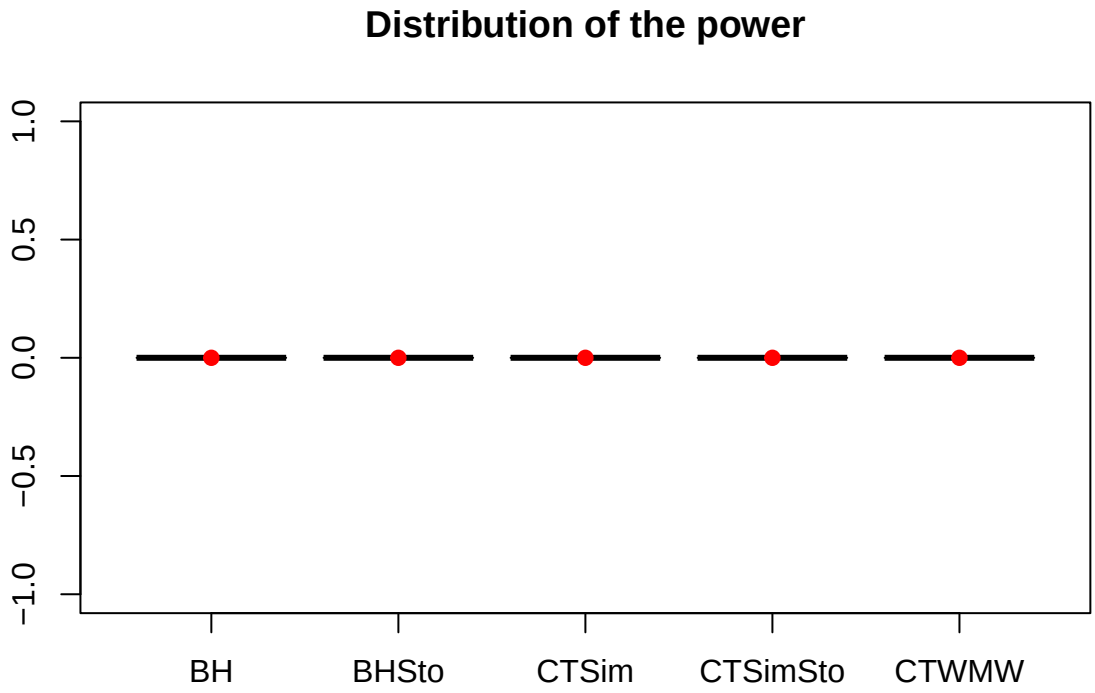
```
boxplot(res$discoveries, main="Distribution of the number of discoveries")
points(x=1:5, y=res$mean.discoveries, pch=19, col="red")
```




```
res$mean.discoveries
```

```
##       BH    BHSto    CTSim CTSimSto    CTWMW
##        0        0        0        0        0
```

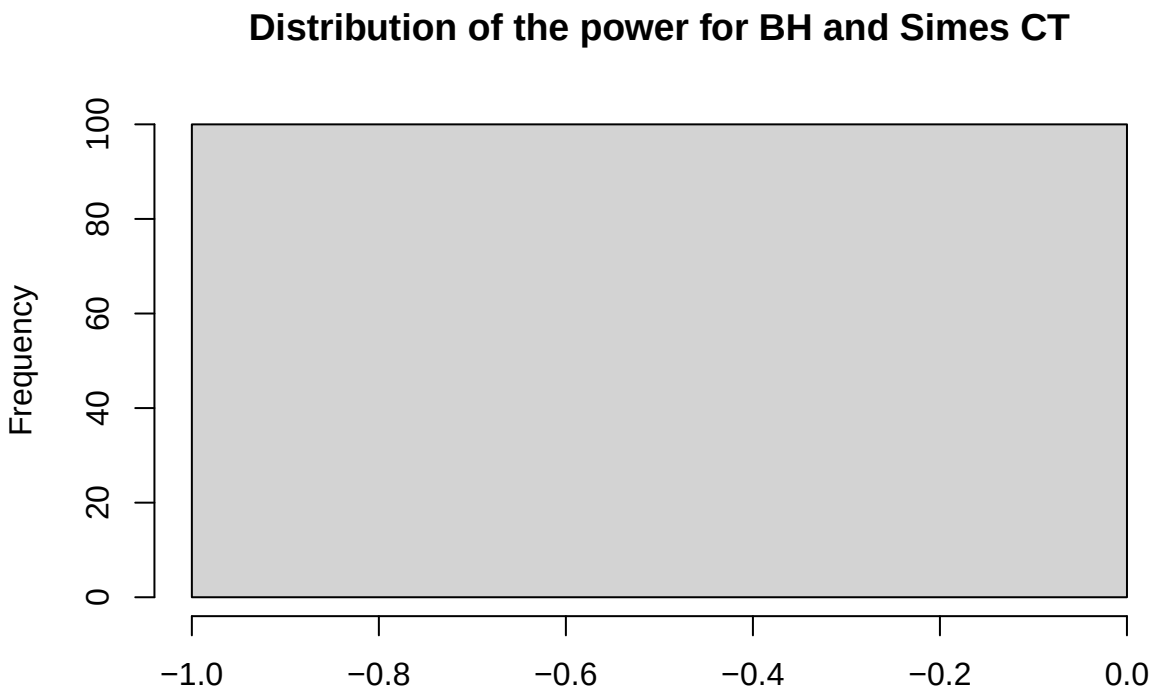
```
boxplot(res$powerGlobalNull, main="Distribution of the power")
points(x=1:5, y=res$mean.powerGlobalNull, pch=19, col="red")
```



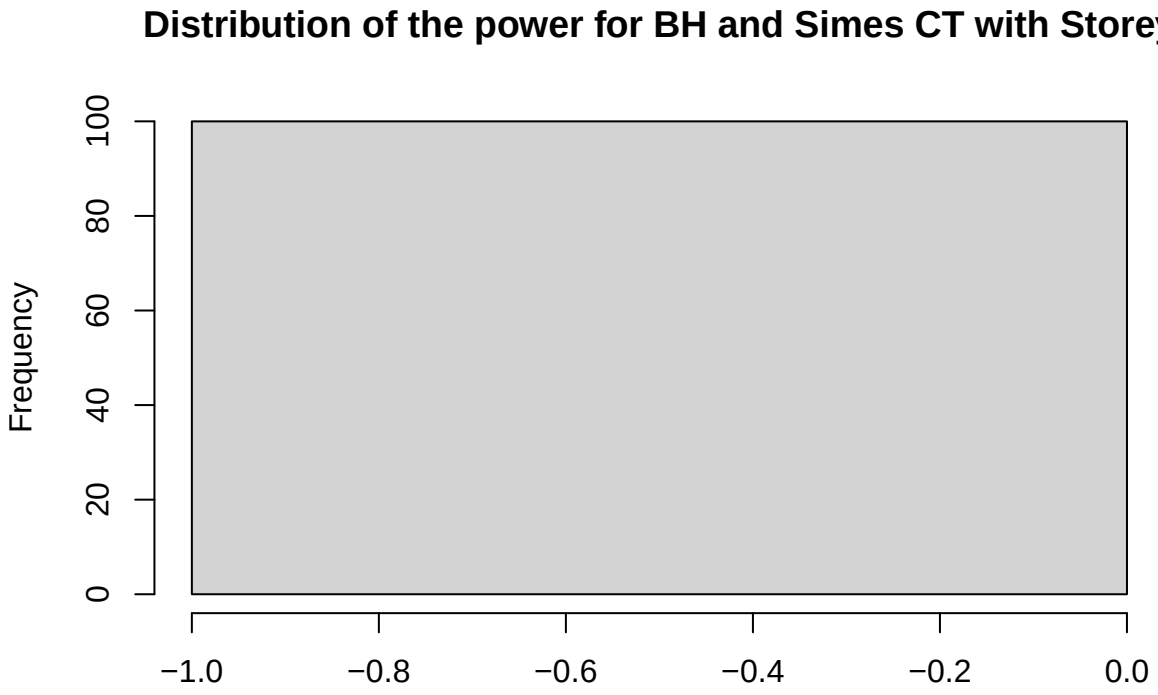

```
res$mean.powerGlobalNull
```

```
##       BH    BHSto    CTSim CTSimSto    CTWMW
##        0        0        0        0        0
```

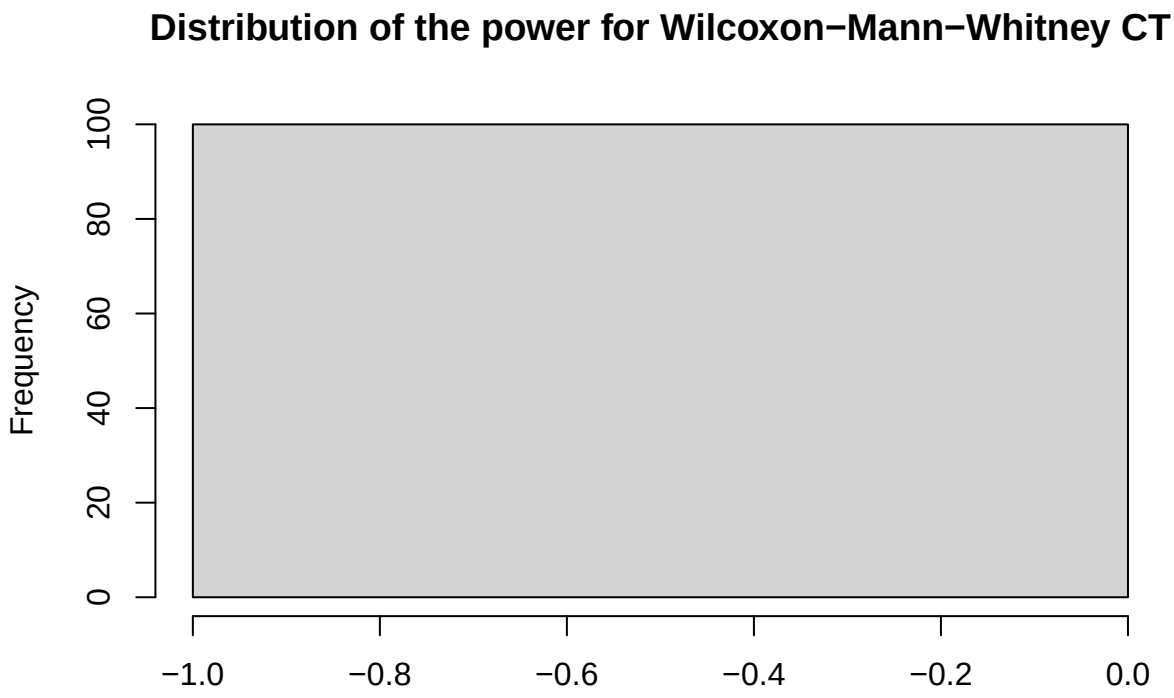
```
hist(as.integer(res$powerGlobalNull[[1]]), xlab="",
     main="Distribution of the power for BH and Simes CT")
```




```
hist(as.integer(res$powerGlobalNull[[2]]), xlab="",
     main="Distribution of the power for BH and Simes CT with Storey")
```




```
hist(as.integer(res$powerGlobalNull[[5]]), xlab="",
     main="Distribution of the power for Wilcoxon-Mann-Whitney CT")
```




```
# resDigits_v2 = res
# save(resDigits_v2,
#      file="C:/Users/c.magnani9/Documents/nout/trials/RealData/PowerStudy/resDigits_v2")
```

10% outliers

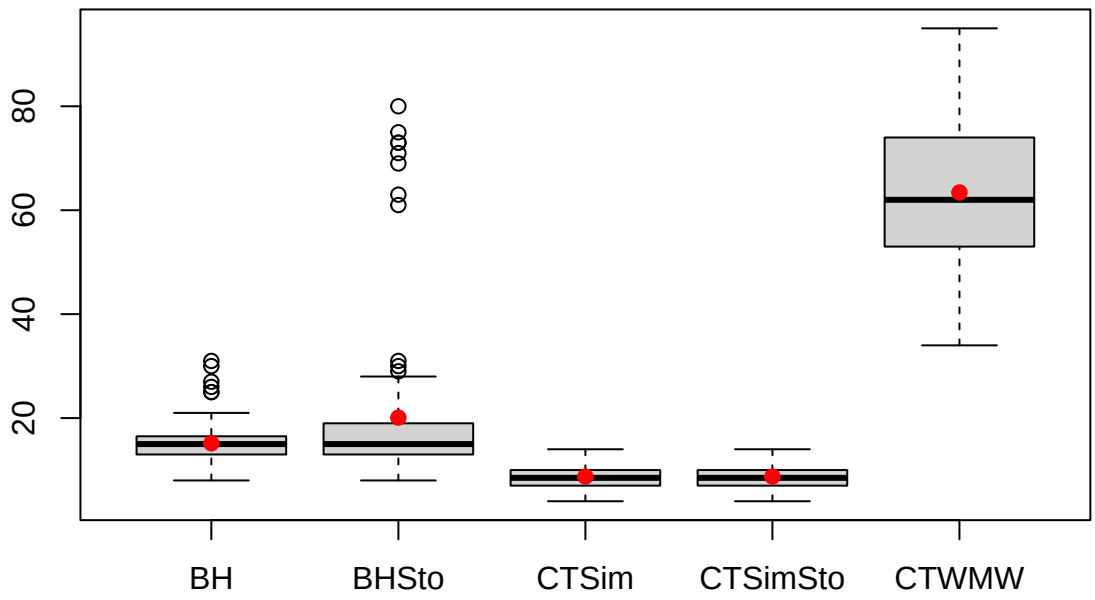
We now set the proportion of inliers equal to 0.9. Referring to Digits dataset we have that the number of inliers is $m_0 = 755.1$ and the number of outliers is $m_1 = 83.9$. Arbitrarily, we choose to set $m_0 = 755$ and $m_1 = 83$.

```
B=100
m1=84

tic()
res = sim_realdataTFO(B=B, in_index=in_ind, out_index=out_ind, dataset=dataset,
                      alpha=alpha,l=l, n=n, m=m, m1=m1)
toc()
```

```
## 9.7 sec elapsed
```

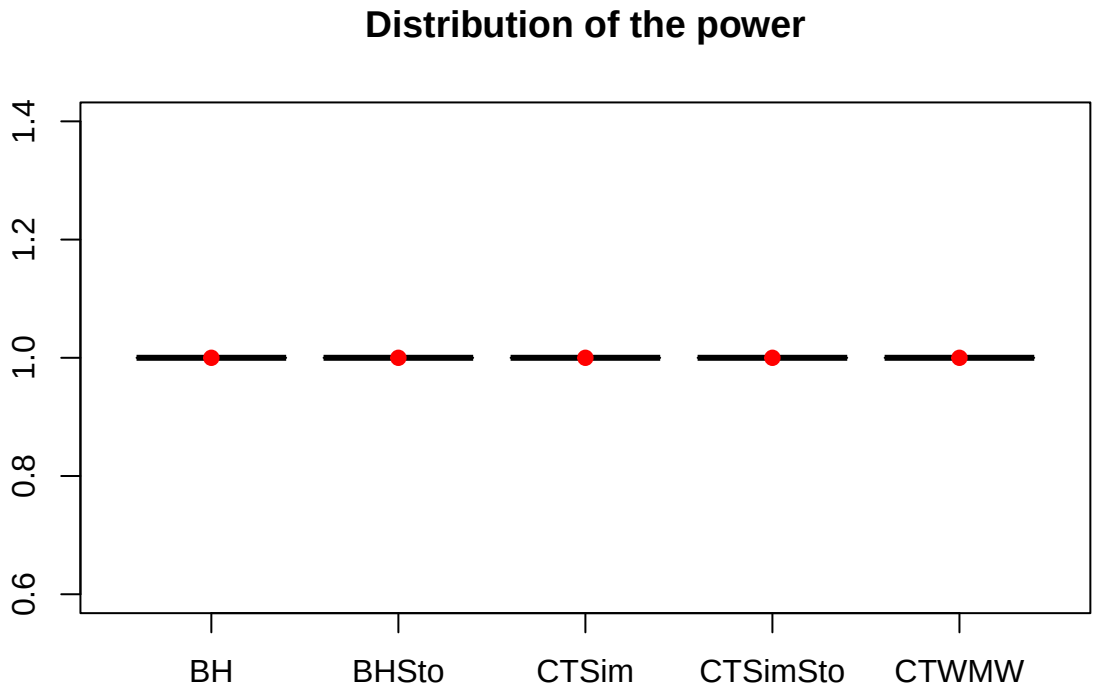
```
boxplot(res$discoveries, main="Distribution of the number of discoveries")
points(x=1:5, y=res$mean.discoveries, pch=19, col="red")
```




```
res$mean.discoveries
```

```
##       BH    BHSto    CTSim CTSimSto    CTWMW
##    15.15    20.09     8.74     8.74    63.42
```

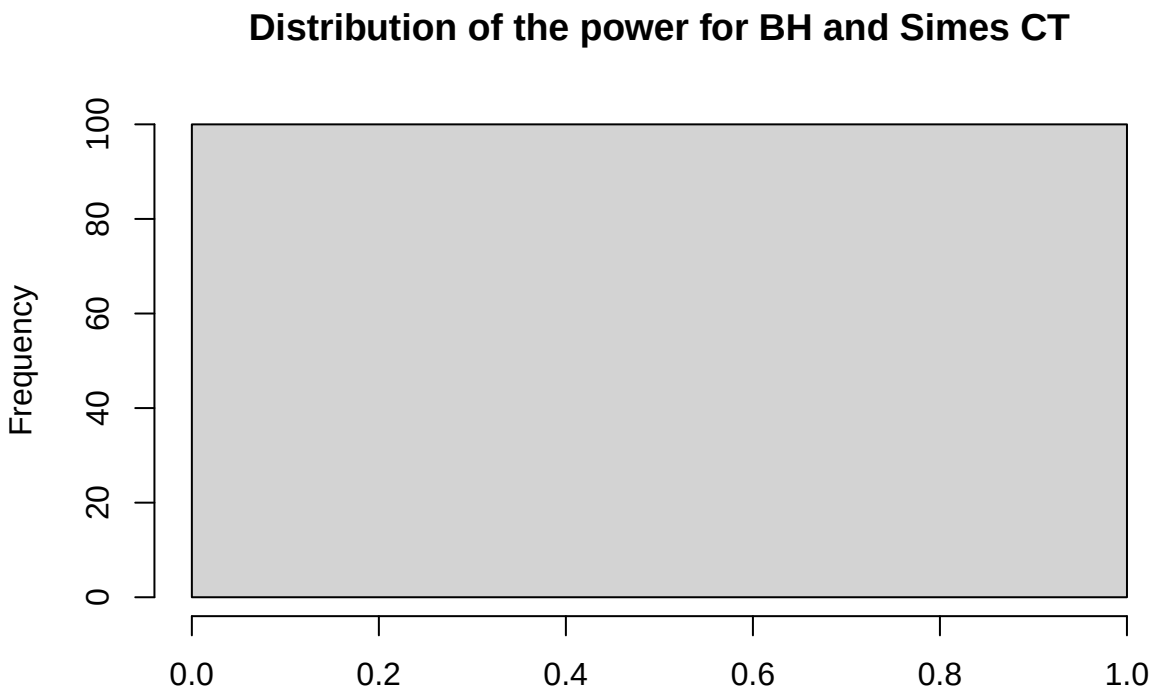
```
boxplot(res$powerGlobalNull, main="Distribution of the power")
points(x=1:5, y=res$mean.powerGlobalNull, pch=19, col="red")
```



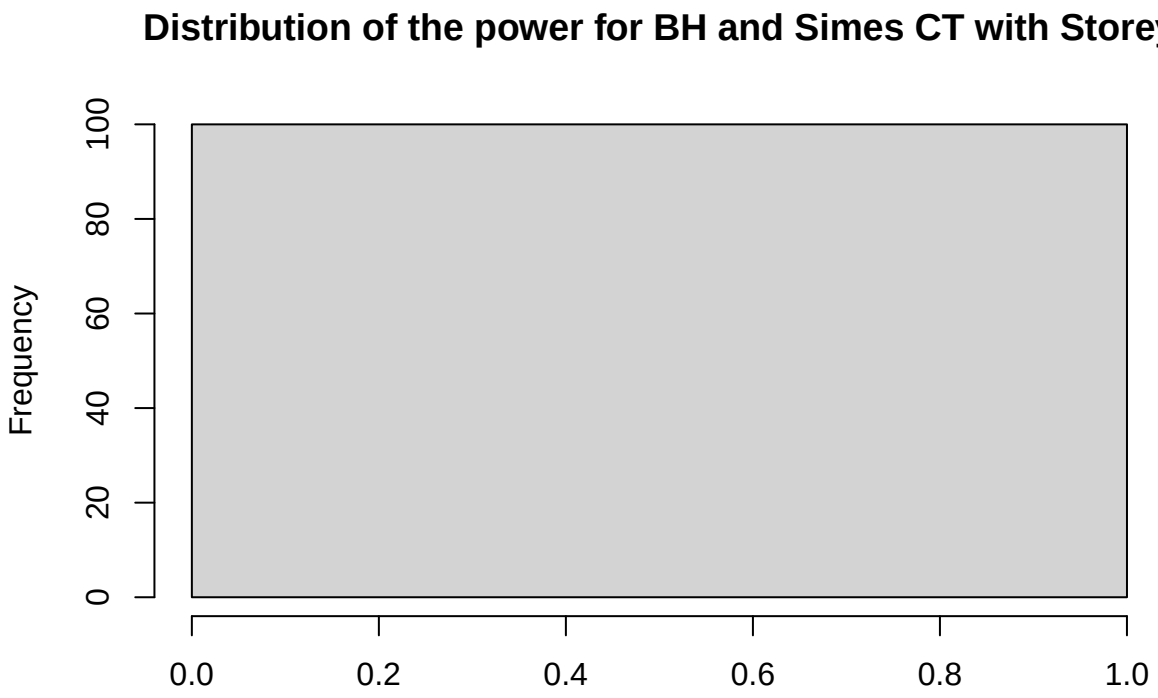

```
res$mean.powerGlobalNull
```

```
##       BH    BHSto    CTSim CTSimSto    CTWMW
##        1        1        1        1        1
```

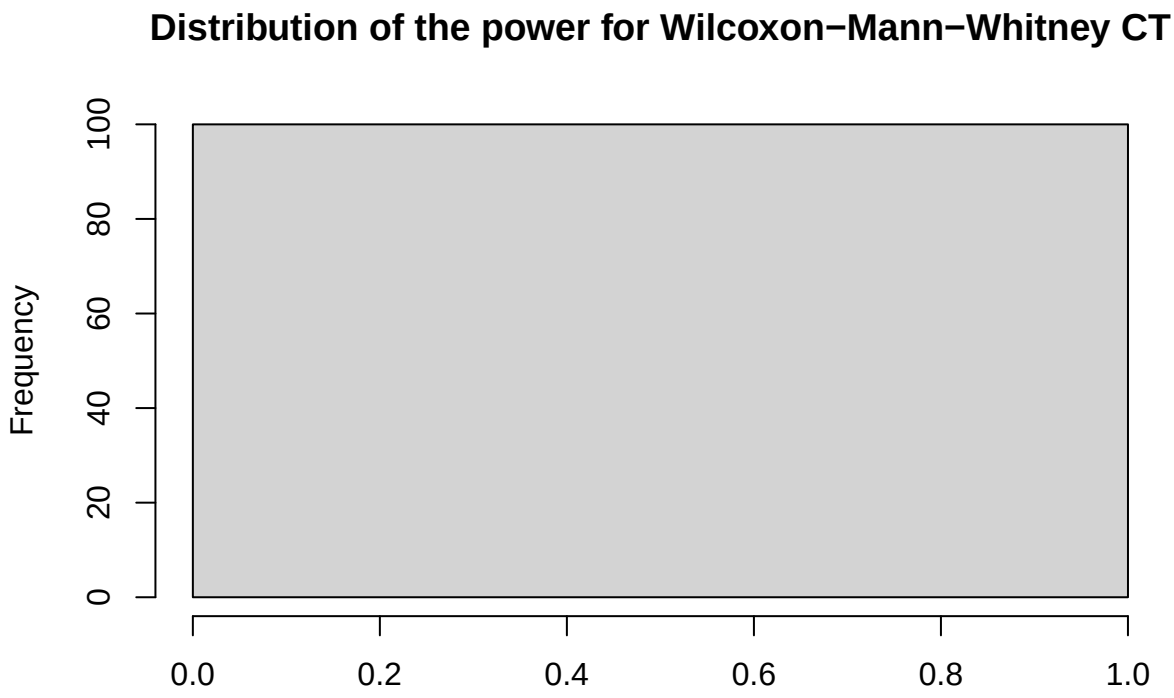
```
hist(as.integer(res$powerGlobalNull[[1]]), xlab="",
     main="Distribution of the power for BH and Simes CT")
```




```
hist(as.integer(res$powerGlobalNull[[2]]), xlab="",
     main="Distribution of the power for BH and Simes CT with Storey")
```




```
hist(as.integer(res$powerGlobalNull[[5]]), xlab="",
     main="Distribution of the power for Wilcoxon-Mann-Whitney CT")
```



References

[1] Bates, S., E. Candes, L. Lei, Y. Romano, and M. Sesia (2023). Testing for outliers with conformal p-values. *Annals of Statistics*, **{51}, 149–178.**