TP3 JF

Kevin McKenna - Ismaïl Ramdé - El Hadrami N'Doye - Elysé Barbaza

20, décembre, 2021

Chagement des données

```
read.table('lipid.csv',h=T,sep=';')->lipid
head(lipid)
```

```
##          col1       col2        col3        col4        col5          col6
## 1 -0.25012480 -0.7738188 -0.47680458 -1.32557060  0.15549541 -0.8909305447
## 2  0.03254362  0.1926563  0.35949791  0.17652456  0.78034146 -0.0146125876
## 3 -0.20603798 -0.6377574 -0.62027132 -0.82695574  1.34802426 -0.3619676622
## 4 -0.22336113 -0.1748458 -0.21520779  0.01909601  0.40227352 -0.0008898606
## 5 -0.85371419 -0.2098944 -0.08148255 -0.14715049 -0.48506322 -0.0148280214
## 6 -0.33748933  0.0204029 -0.33826514 -0.22753874 -0.03369828  0.0847015388
##          col7         col8        col9       col10       col11      col12
## 1 -0.19979326 -0.932716283  0.24266650 -0.39111905  0.75481274 -0.7415818
## 2  0.94087721  0.025915752  0.65528586  0.66312812  1.17664868 -0.1834614
## 3 -0.05109474 -0.020367424 -0.06613075 -0.40918893 -0.37604230  0.2370193
## 4  0.20816291 -0.111880610  0.39673780  0.21400419 -0.03529551 -0.4618848
## 5 -0.28993867 -0.073991195  0.18497677  0.28694626 -0.19648949 -0.3252170
## 6 -0.42257774 -0.002771535 -0.16329070  0.09904386 -0.27648856 -0.4853094
##         col13        col14      col15       col16
## 1  0.35090905  0.033290764 0.31837445 -0.34336780
## 2  1.43683869  1.277460568 1.31763184  0.59198376
## 3 -0.48121397  0.956261124 0.53258575  0.61286698
## 4 -0.25833689  0.202384411 0.09730717  0.09494422
## 5  0.08885928 -0.121147379 0.40310518 -0.22228748
## 6  0.11133227  0.002444174 0.30087432 -0.43316745
```

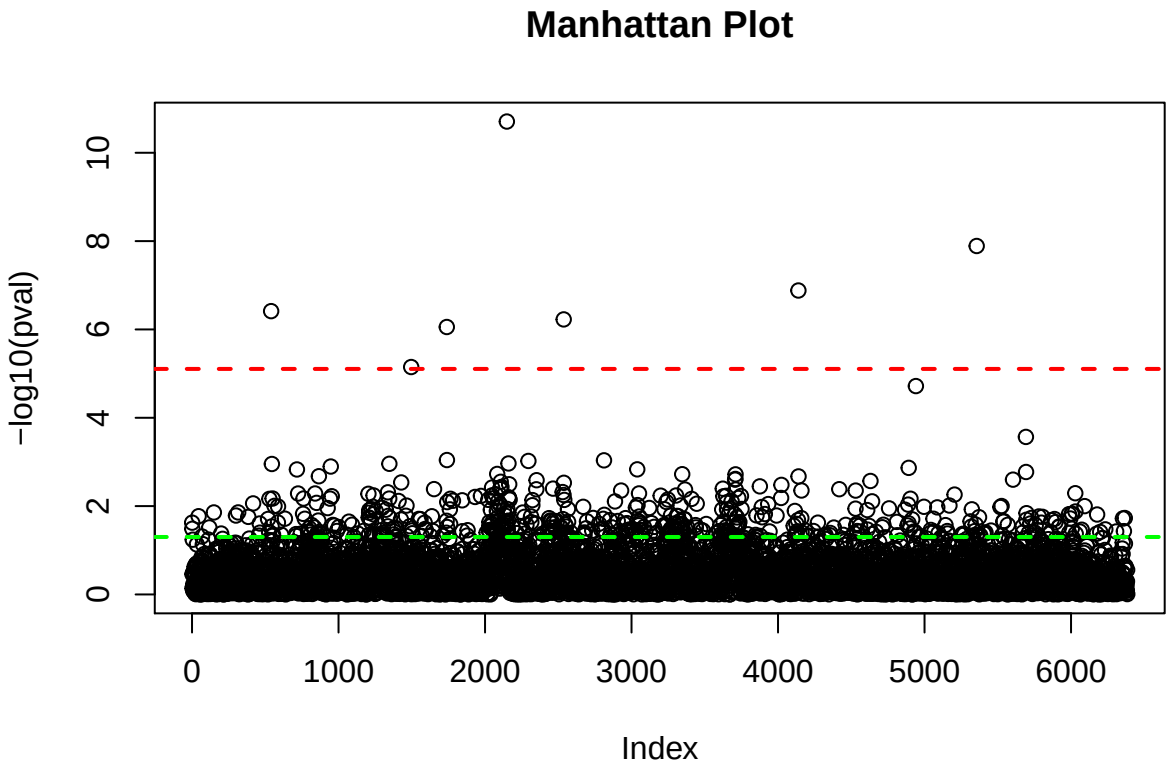
```
alpha = 0.05
```

Calcul des p-valeurs

```
pval = sapply(1:nrow(lipid), function(i) t.test(lipid[i,1:8],lipid[i,9:16],
                                                alternative = 'two.sided')$p.value)
m = length(pval)
pval.ord = sort(pval)
threshBH = alpha*(1:length(pval))/length(pval)
thresh.BY=alpha*(1:m)/(m*sum(1/1:m))
mat = rbind(pval.ord, threshBH)
```

Représentation graphique des p-valeurs

```
plot(-log10(pval))
title("Manhattan Plot")
abline(h = -log10(0.05), col = 'green', lty = 2, lwd=2)
abline(h = -log10(0.05/6384),col = 'red', lty = 2, lwd=2)
```



Exercice 1

1. Implementation des procédures BH et BY qui contrôlent le FDR

La fonction de BH

```
BH <- function(pval,alpha,m = length(pval)){
  pval.ord = sort(pval,ind=T)
  threshBH = alpha*(1:length(pval))/m
  reject = c()
  for (i in 1:m){
    reject[i] = ifelse(threshBH[i]>=pval.ord$x[i],1,0)
  }
  for(i in m:1){
    if(reject[i] ==1){
      return(pval.ord$ix[1:i])
    }
  }
}

TestBH = BH(pval,0.05)
TestBH;length(TestBH)
```

```
## [1] 2149 5356 4139  540 2537 1739 1496 4941
```

```
## [1] 8
```

Le nombre de gène rejeté est de 8, et les gènes concernés sont 2149, 5356, 4139, 540, 2537, 1739, 1496, 4941

La fonction BY

```
BY <- function(pval,alpha, m = length(pval)){
  pval.ord = sort(pval,ind=T)
  threshBY = alpha*(1:length(pval))/(m*sum(1/(1:m)))
  reject = c()
  for (i in 1:m){
    reject[i] = ifelse(threshBY[i]<=pval.ord$x[i],0,1)
  }
  for(i in m:1){
    if(reject[i] ==1){
      return(pval.ord$ix[1:i])
    }
  }
}

TestBY=BY(pval,0.05)
TestBY;length(TestBY)
```

```
## [1] 2149 5356 4139  540 2537 1739
```

```
## [1] 6
```

Le nombre de gène rejeté est de 6, et les gènes concernés sont 2149, 5356, 4139, 540, 2537, 1739

2. Comparaison du résultat à ceux obtenus à l'aide de la fonction p.ajust (BH et BY)

p.adjust avec method = 'BH'

```
BHadjusted = sort(p.adjust(pval, method = 'BH'), ind =T)
BHadjusted$ix[BHadjusted$x<=0.05]
```

```
## [1] 2149 5356 4139  540 2537 1739 1496 4941
```

p.adjust avec method = 'BY'

```
BYadjusted = sort(p.adjust(pval, method = 'BY'), ind =T)
BYadjusted$ix[BYadjusted$x<=0.05]
```

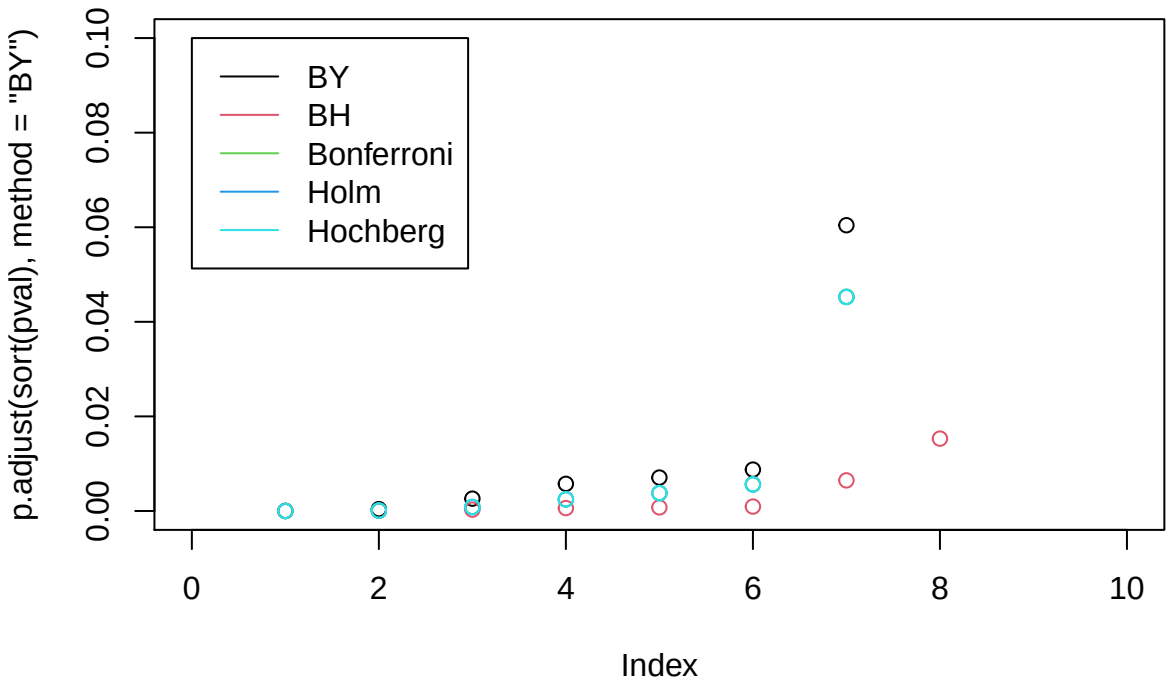
```
## [1] 2149 5356 4139  540 2537 1739
```

Dans les deux cas, les gènes obtenus avec la méthode p.adjust sont les mêmes que ceux obtenus en utilisant les fonctions ci-dessus.

3. Représentation des p-valeurs ajustées pour BH, BY Bonferroni, Holm et Hochberg

```
plot(p.adjust(sort(pval), method = 'BY'),xlim = c(0,10),ylim = c(0,0.1))
points(p.adjust(sort(pval), method = 'BH'),ylim = c(0,0.1),col = 2)
points(p.adjust(sort(pval), method = 'bonferroni'),xlim = c(0,10),ylim = c(0,0.1),col = 3)
points(p.adjust(sort(pval), method = 'holm'),xlim = c(0,10),ylim = c(0,0.1),col = 4)
```

```
points(p.adjust(sort(pval), method = 'hochberg'),xlim = c(0,10),ylim = c(0,0.1),col = 5)
legend(0,.1,legend = c('BY','BH','Bonferroni','Holm','Hochberg'), col = 1:5, lty = 1)
```



Ici on voit les points des différentes méthodes. On aperçoit 3 couleurs, mais en fait il y en a toutes les 5, elles sont juste superposées. En ajoutant du bruit avec la fonction jitter on obtient le même résultat. On voit que le nombre de p-valeurs rejetés est similaire, et que les valeurs sont aussi très similaires.

4. Comparaison des gènes sélectionnées par les deux procédures

```
BYcheck <- sort(p.adjust(pval, method = 'BY'),ind = T)
BHcheck<- sort(p.adjust(pval, method = 'BH'),ind = T)
bonferroni_check<- sort(p.adjust(pval, method = 'bonferroni'),ind = T)
holm_check<- sort(p.adjust(pval, method = 'holm'),ind = T)
hochberg_check<- sort(p.adjust(pval, method = 'hochberg'),ind = T)
BYcheck$ix[BYcheck$x<0.05]
```

```
## [1] 2149 5356 4139  540 2537 1739
```

```
BHcheck$ix[BHcheck$x<0.05]
```

```
## [1] 2149 5356 4139  540 2537 1739 1496 4941
```

```
bonferroni_check$ix[bonferroni_check$x<0.05]
```

```
## [1] 2149 5356 4139  540 2537 1739 1496
```

```
holm_check$ix[holm_check$x<0.05]
```

```
## [1] 2149 5356 4139  540 2537 1739 1496
```

```
hochberg_check$ix[hochberg_check$x<0.05]
```

```
## [1] 2149 5356 4139  540 2537 1739 1496
```

On n'observe pas une grande différence entre les gènes sélectionnés. Ils sont quasiment les mêmes sauf que le BY prend une valeur de moins que les 3 méthodes de FWER et le BH prend une valeur de plus.

Question 5

```
BH(pval, alpha, m = 5198)
```

```
## [1] 2149 5356 4139  540 2537 1739 1496 4941
```

```
BY(pval,alpha,m =5198)
```

```
## [1] 2149 5356 4139  540 2537 1739 1496
```

Quand on change le m a m0_chap de 5198, le BH ne change pas le nombre de gènes qu'il prend, mais le BY prend une de plus, par rapport au cas avec m0 inconnu.

Exercice 2

Question 1

```
sim.pval = function(rho = 0, m = 10, mu = rep(0,m)){
  pval = c()
  X = c()
  W <- rnorm(1)
  Xi = mu + sqrt(rho)*W+sqrt(1-rho)*rnorm(m)
  pval = 1-pnorm(Xi)
  pval
}
```

Question 2

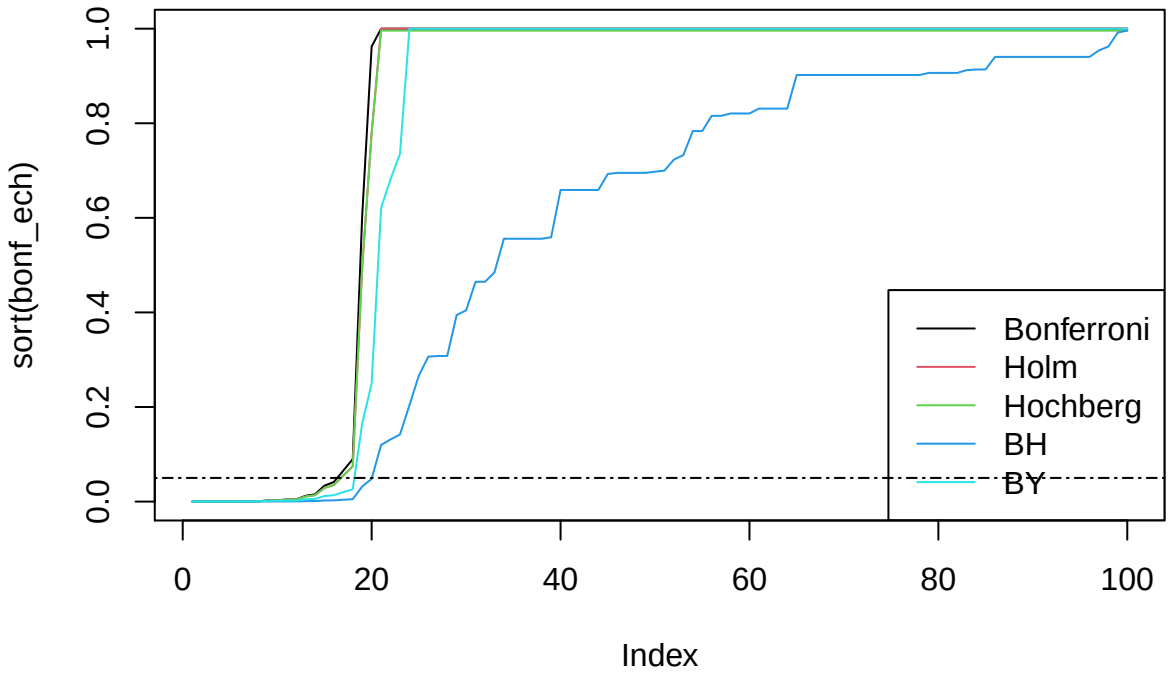
```
m = 100
rho = 0
mu = c(rep(0,m*.80),rep(4,m*.20))
ech <- sim.pval(rho,m,mu)
ech.ord <- sort(ech,ind = TRUE)
bonf_ech <- p.adjust(ech,method='bonferroni')
holm_ech <- p.adjust(ech,method='holm')
hoch_ech <- p.adjust(ech,method='hochberg')
bh_ech <- p.adjust(ech,method='BH')
by_ech <- p.adjust(ech,method='BY')
Rs <- t(c('Bonf' = sum(bonf_ech<0.05), 'Holm' = sum(holm_ech<0.05),
          'Hoch' = sum(hoch_ech<0.05),'BH' = sum(bh_ech<0.05), 'BY' = sum(by_ech<0.05)))
Vs <- t(c('Bonf' = sum(bonf_ech[1:80]<0.05), 'Holm' = sum(holm_ech[1:80]<0.05),
          'Hoch' = sum(hoch_ech[1:80]<0.05),'BH' = sum(bh_ech[1:80]<0.05),
          'BY' = sum(by_ech[1:80]<0.05)))

FDP = Vs/Rs
FDP
```

```
##      Bonf Holm Hoch        BH         BY
## [1,]    0    0    0 0.1363636 0.05555556
```

On remarque que BH et BY ont tous les deux un meilleur FDP que les trois méthodes de FWER.

Question 3

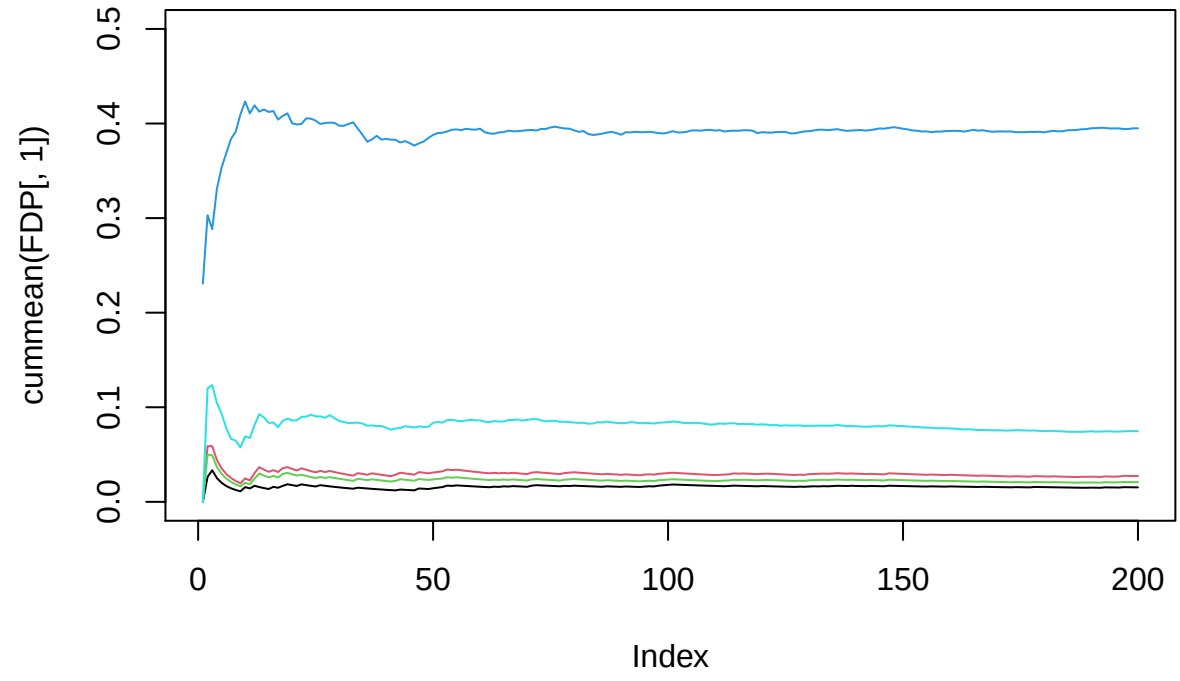
```
m = 100
rho = 0
mu = c(rep(0,80),rep(4,20))
B = 200
FDP = c()

for (i in 1:B){
  ech <- sim.pval(rho,m,mu)
  V <- sum(ech[1:80]<0.05)
  bonf_ech <- p.adjust(ech,method='bonferroni')
  holm_ech <- p.adjust(ech,method='holm')
  hoch_ech <- p.adjust(ech,method='hochberg')
  bh_ech <- p.adjust(ech,method='BH')
  by_ech <- p.adjust(ech,method='BY')
  Rs <- t(c('Bonf' = sum(bonf_ech<0.5), 'Holm' = sum(holm_ech<0.05),
            'Hoch' = sum(hoch_ech<0.5),'BH' = sum(bh_ech<0.5), 'BY' = sum(by_ech<0.5)))
  Vs <- t(c('Bonf' = sum(bonf_ech[1:80]<0.5), 'Holm' = sum(holm_ech[1:80]<0.5),
            'Hoch' = sum(hoch_ech[1:80]<0.5),'BH' = sum(bh_ech[1:80]<0.5),
            'BY' = sum(by_ech[1:80]<0.5)))

  FDP = rbind(FDP,Vs/Rs)
}

plot(sort(bonf_ech), type = 'l', col = 1)
lines(sort(holm_ech), col = 2)
lines(sort(hoch_ech), col = 3)
lines(sort(bh_ech), col = 4)
lines(sort(by_ech), col = 5)
abline(h = 0.05, lty = 6)
legend('bottomright', legend = c('Bonferroni','Holm','Hochberg','BH','BY'), col = 1:5,
       lty = 1)
```




```
apply(FDP,2,mean)
```

```
##       Bonf       Holm       Hoch         BH         BY
## 0.01529751 0.02740106 0.02101124 0.39495184 0.07465832
```

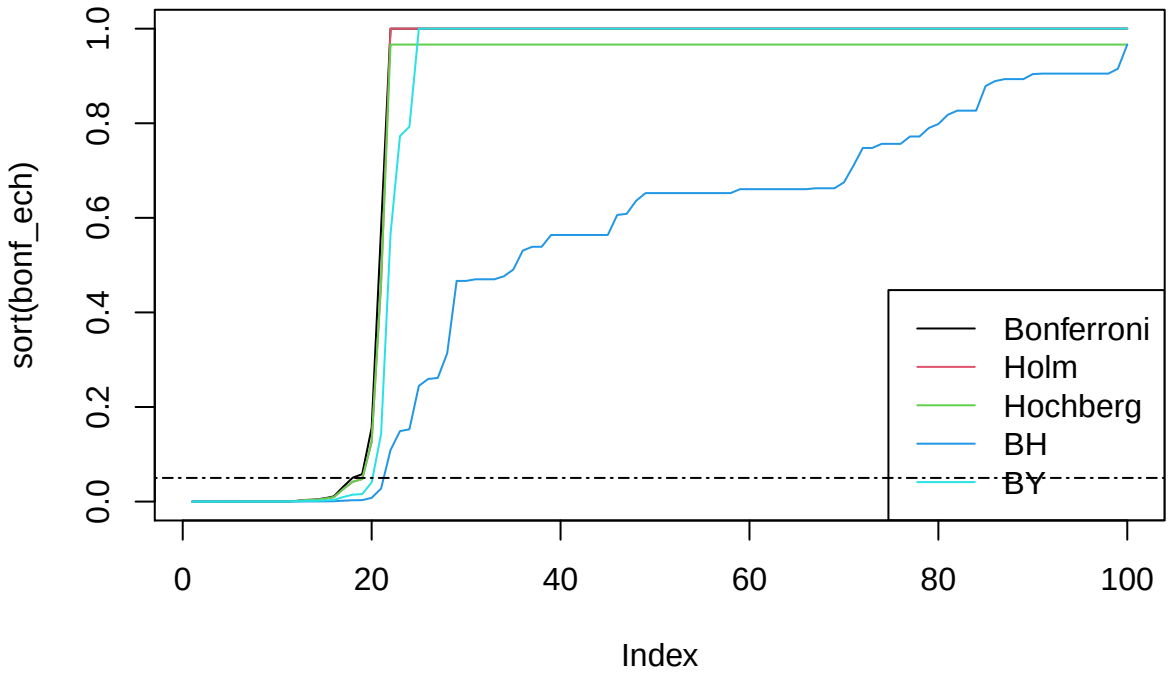
```
plot(cummean(FDP[,1]), type = 'l', ylim = c(0,0.5))
lines(cummean(FDP[,2]),col = 2)
lines(cummean(FDP[,3]),col = 3)
lines(cummean(FDP[,4]),col = 4)
lines(cummean(FDP[,5]),col = 5)
```



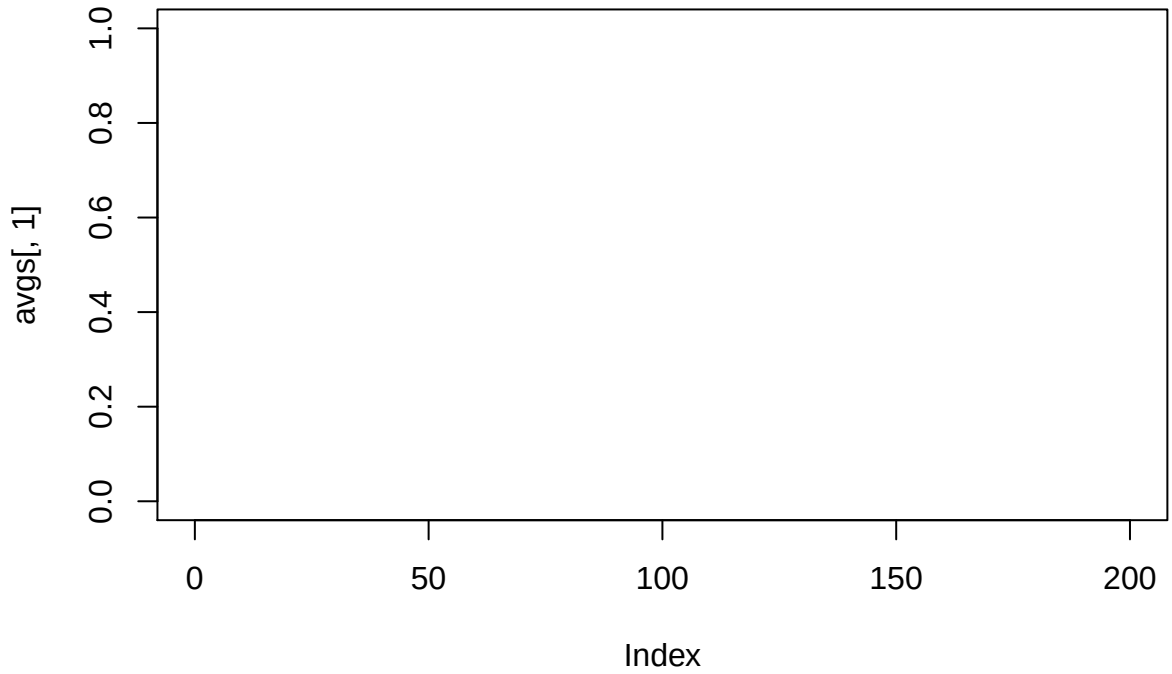
```r
# plot(density(FDP[,1]), type = 'l')
# lines(density(FDP[,2]),col = 2)
# lines(density(FDP[,3]),col = 3)
# plot(density(FDP[,4]),col = 4)
# lines(density(FDP[,5]),col = 5)
# legend('topleft', legend = c('Bonferroni','Holm','Hochberg','BH','BY'), col = 1:5, lty = 1)
```

```r
m = 100
rho = 0
mu = c(rep(0,80),rep(4,20))
B = 200
ms = seq(200,4000,by = 200)
FDP = c()
avgs = c()
# for(j in 1:20){
#   m = ms[j]
  for (i in 1:B){
    ech <- sim.pval(rho,m,mu)
    bonf_ech <- p.adjust(ech,method='bonferroni')
    holm_ech <- p.adjust(ech,method='holm')
    hoch_ech <- p.adjust(ech,method='hochberg')
    bh_ech <- p.adjust(ech,method='BH')
    by_ech <- p.adjust(ech,method='BY')
    Rs <- t(c('Bonf' = sum(bonf_ech<0.05), 'Holm' = sum(holm_ech<0.05),
              'Hoch' = sum(hoch_ech<0.05),'BH' = sum(bh_ech<0.05), 'BY' = sum(by_ech<0.05)))
    Vs <- t(c('Bonf' = sum(bonf_ech[1:(length(ech)*0.8)]<0.05),
              'Holm' = sum(holm_ech[1:(length(ech)*0.8)]<0.05),
              'Hoch' = sum(hoch_ech[1:(length(ech)*0.8)]<0.05),
              'BH' = sum(bh_ech[1:(length(ech)*0.8)]<0.05),
              'BY' = sum(by_ech[1:(length(ech)*0.8)]<0.05)))
    Ss <- t(c('Bonf' = sum(bonf_ech[(length(ech)*0.8):length(ech)]<0.05),
              'Holm' = sum(holm_ech[(length(ech)*0.8):length(ech)]<0.05),
              'Hoch' = sum(hoch_ech[(length(ech)*0.8):length(ech)]<0.05),
              'BH' = sum(bh_ech[(length(ech)*0.8):length(ech)]<0.05),
              'BY' = sum(by_ech[(length(ech)*0.8):length(ech)]<0.05)))
    Rs <- Vs+Ss
    FDP = rbind(FDP,Vs[1:5]/Rs[1:5])
}
#   avgs = rbind(avgs,apply(FDP,2,mean))
# }
plot(sort(bonf_ech), type = 'l', col = 1)
lines(sort(holm_ech), col = 2)
lines(sort(hoch_ech), col = 3)
lines(sort(bh_ech), col = 4)
lines(sort(by_ech), col = 5)
abline(h = 0.05, lty = 6)
legend('bottomright', legend = c('Bonferroni','Holm','Hochberg','BH','BY'), col = 1:5, lty = 1)
```

Ici je pense que c'est la premiere courbe est correct mais j'ai pas confiance au v/r donc la deuxieme graphe est incorrect. Mais c'est bizarre car si tu augmente le m a 1000, il semble plutot correct...donc j'en sais rien. ??




```
plot(avgs[,1], type = 'l', xlim = c(0,200), ylim = c(0,1))
lines(avgs[,2])
lines(avgs[,3])
lines(avgs[,4])
lines(avgs[,5])
```

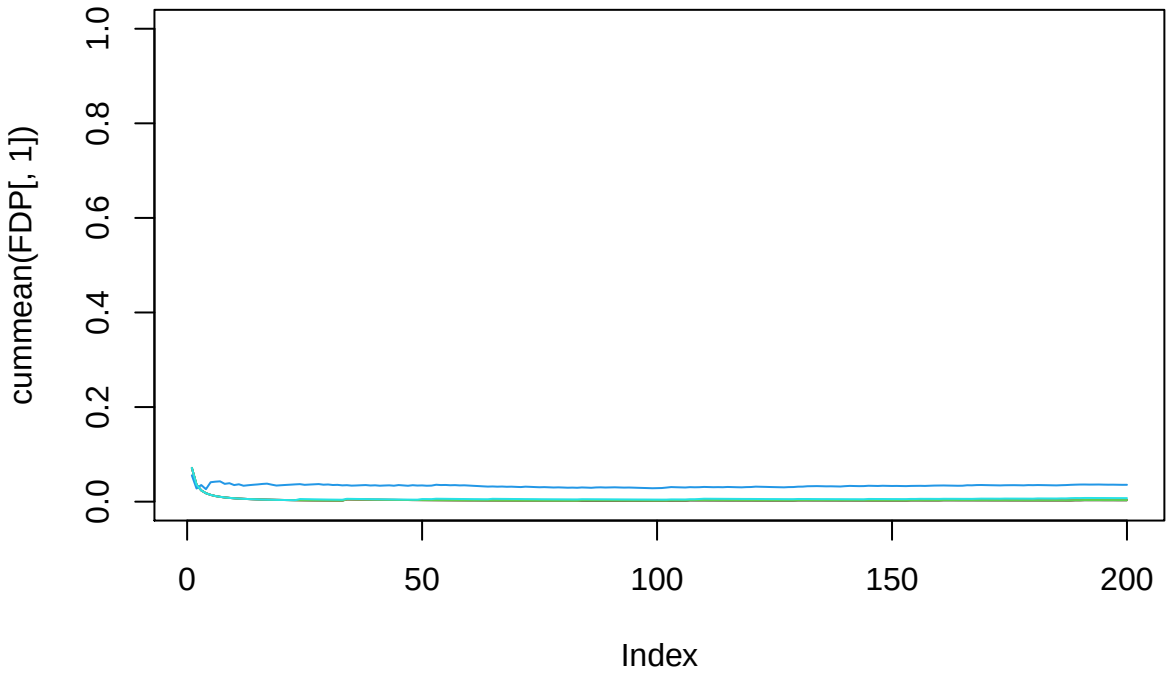



```
apply(FDP,2,mean)
```

```
## [1] 0.002964158 0.003190653 0.003190653 0.035615072 0.007432894
```

```
plot(cummean(FDP[,1]), type = 'l',ylim = c(0,1))
lines(cummean(FDP[,2]),col = 2)
```

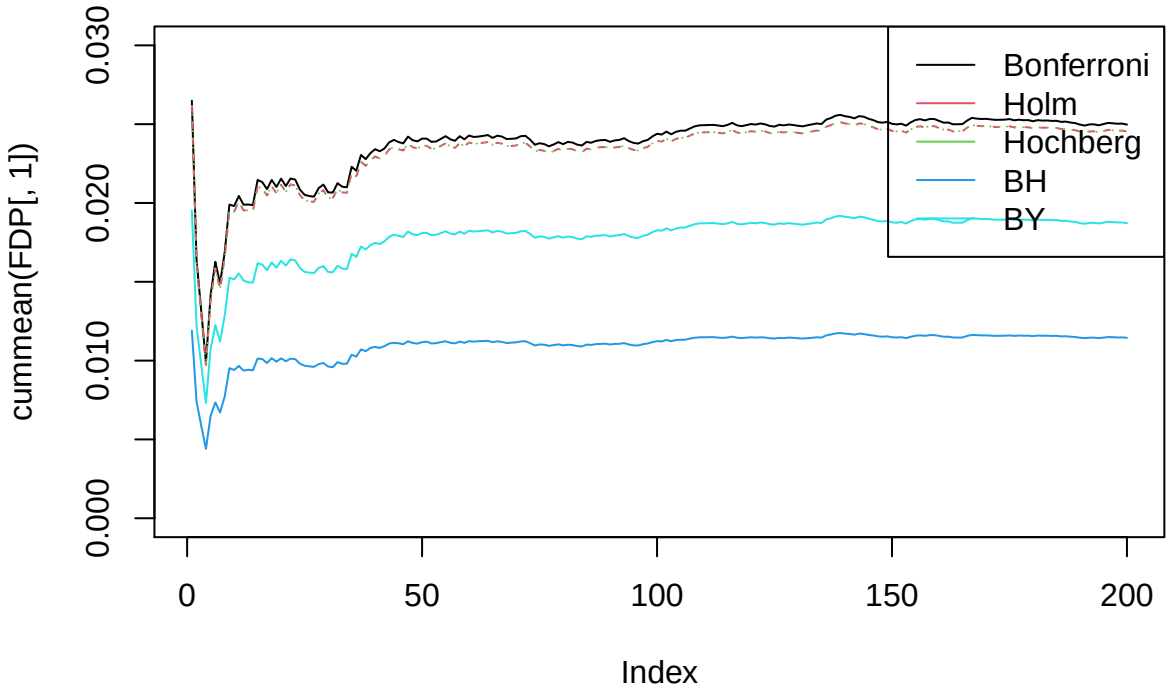
```
lines(cummean(FDP[,3]),col = 3)
lines(cummean(FDP[,4]),col = 4)
lines(cummean(FDP[,5]),col = 5)
```




```
m = 1000
rho = 0
mu = c(rep(0,80),rep(4,20))
B = 200
FDP = c()
for (i in 1:B){
  ech <- sim.pval(rho,m,mu)
  V <- sum(ech[1:80]<0.05)
  bonf_ech <- p.adjust(ech,method='bonferroni')
  holm_ech <- p.adjust(ech,method='holm')
  hoch_ech <- p.adjust(ech,method='hochberg')
  bh_ech <- p.adjust(ech,method='BH')
  by_ech <- p.adjust(ech,method='BY')
  Rs <- c('Bonf' = sum(bonf_ech<0.5), 'Holm' = sum(holm_ech<0.5),'Hoch' = sum(hoch_ech<0.5),
          'BH' = sum(bh_ech<0.5), 'BY' = sum(by_ech<0.5))
  FDP = rbind(FDP,V/Rs)
}
apply(FDP,2,mean)
```

```
##       Bonf       Holm       Hoch         BH         BY
## 0.02497455 0.02453005 0.02453005 0.01144548 0.01872011
```

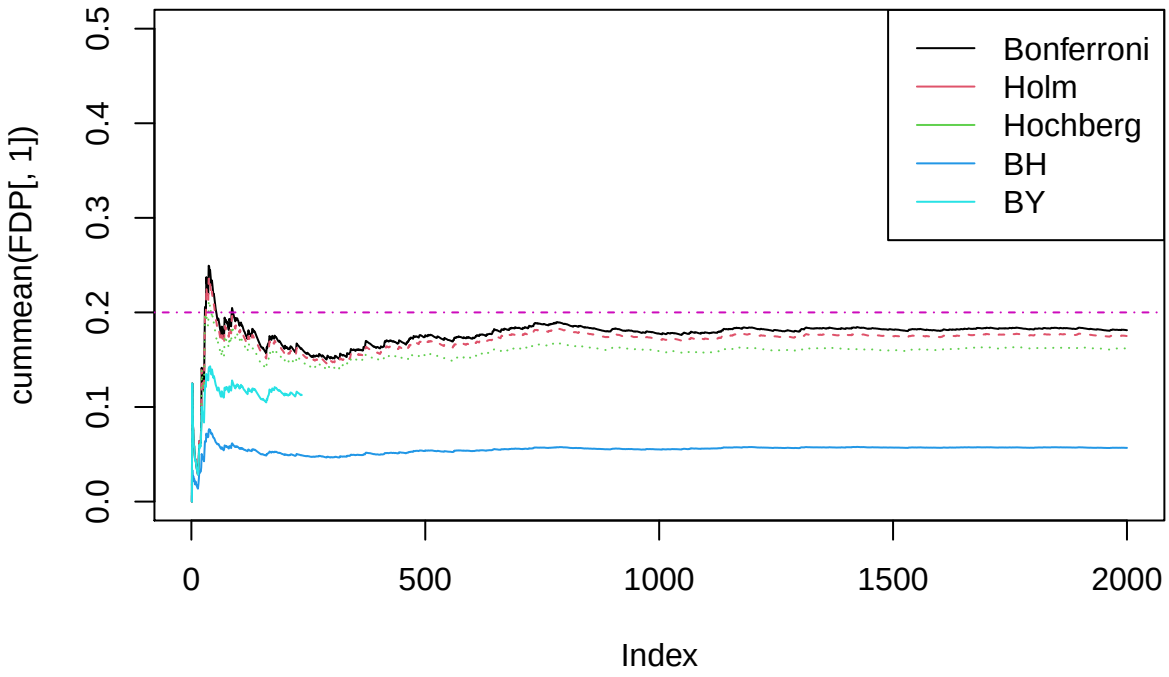
```
plot(cummean(FDP[,1]), type = 'l',ylim = c(0,0.03))
lines(cummean(FDP[,2]),col = 2, lty = 2)
lines(cummean(FDP[,3]),col = 3, lty = 3)
lines(cummean(FDP[,4]),col = 4)
lines(cummean(FDP[,5]),col = 5)
legend('topright', legend = c('Bonferroni','Holm','Hochberg','BH','BY'), col = 1:5, lty = 1)
```




```
m = 100
rho = 0.5
mu = c(rep(0,80),rep(4,20))
B = 2000
FDP = c()
for (i in 1:B){
  ech <- sim.pval(rho,m,mu)
  V <- sum(ech[1:80]<0.05)
  bonf_ech <- p.adjust(ech,method='bonferroni')
  holm_ech <- p.adjust(ech,method='holm')
  hoch_ech <- p.adjust(ech,method='hochberg')
  bh_ech <- p.adjust(ech,method='BH')
  by_ech <- p.adjust(ech,method='BY')
  Rs <- c('Bonf' = sum(bonf_ech<0.5), 'Holm' = sum(holm_ech<0.5),'Hoch' = sum(hoch_ech<0.5),
          'BH' = sum(bh_ech<0.5), 'BY' = sum(by_ech<0.5))
  FDP = rbind(FDP,V/Rs)
}
apply(FDP,2,mean)
```

```
##       Bonf       Holm       Hoch         BH         BY
## 0.18120844 0.17502109 0.16164184 0.05678065        NaN
```

```
plot(cummean(FDP[,1]), type = 'l',ylim = c(0,0.5))
lines(cummean(FDP[,2]),col = 2, lty = 2)
lines(cummean(FDP[,3]),col = 3, lty = 3)
lines(cummean(FDP[,4]),col = 4)
lines(cummean(FDP[,5]),col = 5)
abline(h = 0.2, col = 6, lty = 4)
legend('topright', legend = c('Bonferroni','Holm','Hochberg','BH','BY'), col = 1:5, lty = 1)
```



Exercice 3

Question 1

L'analogue à la procédure de Bonferroni

```
pval_k = pval[pval<0.05]
k = length(pval_k)
test_bonferroni = function(alpha,m,pval) {
  pval_k_ord = sort(pval,index.return=TRUE)$x
  pval_k_ord_ind = sort(pval,index.return = TRUE)$ix
  dec_bonf = c()
  mat = cbind(pval_k_ord, threshold = (k*alpha) / m)
  rownames(mat) = paste('test', 1:m)
  mat = as.data.frame(mat)
  for (i in 1:m) {
    if (mat[i, 1] <= mat[i, 2]) {
      dec_bonf = c(dec_bonf, "On rejette H0i")
    }
    else{
      dec_bonf = c(dec_bonf, "on accepte H0i")
    }
  }
  mat$dec_bonf = dec_bonf
  gene_bonf = lipid[pval_k_ord_ind[which(mat$dec_bonf == "On rejette H0i")], ]
  pval_bonf = pval_k_ord[which(mat$dec_bonf == "On rejette H0i")]
  return(list(gene=gene_bonf,pval_bonferroni=pval_bonf))
}
test_b = test_bonferroni(alpha = 0.05,m=length(pval),pval)
test_b = sort(test_b$pval_bonferroni,ind=T)
test_b$x
```

```
##  [1] 1.963690e-11 1.290756e-08 1.313776e-07 3.850496e-07 5.930427e-07
```

```
##  [6] 8.804668e-07 7.094646e-06 1.916922e-05 2.715334e-04 9.059274e-04
## [11] 9.204069e-04 9.495250e-04 1.082301e-03 1.102650e-03 1.112820e-03
## [16] 1.269842e-03 1.361324e-03 1.472066e-03 1.480917e-03 1.689717e-03
## [21] 1.869637e-03 1.896057e-03 1.912469e-03 2.107444e-03 2.126525e-03
## [26] 2.426704e-03 2.516830e-03 2.517709e-03 2.591257e-03 2.698279e-03
## [31] 2.814787e-03 2.914371e-03 2.967985e-03 3.201096e-03 3.310262e-03
## [36] 3.575886e-03 3.672145e-03 3.801682e-03 3.861792e-03 4.013826e-03
## [41] 4.032269e-03 4.126417e-03 4.143487e-03 4.159100e-03 4.164903e-03
```

```
# nombre de gènes
length(test_b$ix)
```

```
## [1] 45
```

L'analogue à la procédure de Holm

```
pval_k = pval[pval < 0.05]
k = length(pval_k)
test_holm = function(alpha, m, pval) {
  pval_k_ord = sort(pval, index.return = TRUE)$x
  pval_k_ord_ind = sort(pval, index.return = TRUE)$ix
  dec_holm = c()
  for (i in 1:m) {
    if (i <= k && pval_k_ord[i] <= (k * alpha) / m) {
      dec_holm = c(dec_holm, "On rejette HOi")
    }
    if(i > k && pval_k_ord[i] <= (k * alpha) / (m + k - 1)) {
      dec_holm = c(dec_holm, "on rejette HOi")
    }
    else{
      dec_holm = c(dec_holm, "on acceptte HOi")
    }
  }
  gene_holm = lipid[pval_k_ord_ind[which(dec_holm == "On rejette HOi")], ]

  pval_holm = pval_k_ord[which(dec_holm == "On rejette HOi")]
  return(list(gene=gene_holm, pval_holm=pval_holm))
}

test_h = test_holm(alpha = 0.05, m=length(pval),pval)
test_h = sort(test_h$pval_holm,ind=T)
test_h$x
```

```
##  [1] 1.963690e-11 1.313776e-07 5.930427e-07 7.094646e-06 2.715334e-04
##  [6] 9.204069e-04 1.082301e-03 1.112820e-03 1.361324e-03 1.480917e-03
## [11] 1.869637e-03 1.912469e-03 2.126525e-03 2.516830e-03 2.591257e-03
## [16] 2.814787e-03 2.967985e-03 3.310262e-03 3.672145e-03 3.861792e-03
## [21] 4.032269e-03 4.143487e-03 4.164903e-03 4.221510e-03 4.447258e-03
## [26] 4.811139e-03 5.110894e-03 5.176187e-03 5.266416e-03 5.485376e-03
## [31] 5.734505e-03 5.755500e-03 5.861953e-03 5.911608e-03 6.012857e-03
## [36] 6.346014e-03 6.517752e-03 6.550360e-03 6.732304e-03 6.760086e-03
## [41] 6.809363e-03 6.862782e-03 7.090270e-03 7.208987e-03 7.313242e-03
```

```
# nombre de gènes
length(test_h$ix)
```

```
## [1] 45
```

Ces deux méthodes sélectionnent chacune 45 gènes, elles contrôlent bien le K-FWER.

Représentation des p-valeurs des différentes procédures

```
pval_B = test_bonferroni(alpha = 0.05,m=length(pval),pval)$pval_bonferroni
pval_H = test_holm(alpha = 0.05, m = length(pval), pval)$pval_holm

plot(pval_B, main = 'P-valeurs', pch=20, xlim = c(0,50), ylim = c(0,.01))
points(pval_H, col = 'red')
legend("bottomright", legend=c("Bonferroni", "Holm"),
       col=c("black", "red"), lty = 15,
       title="Méthodes")
```