Exercise3

Group C

6/1/2022

Loading Data

We load dati3 which contains three variables, dati which contains raw counts, normali which contains normalized raw counts and normalilog which contains normalized data in log scale

```
load('dati3')
```

EdgeR Analysis

We select the first 19 columns, the ones relative to the two groups we're studying

```
dati<-dati[,1:19]
```

We now create the design matrix we will use to fit the model

```
#assigning factors to positions corresponding to subjects
group1<-grep("Group1",colnames(dati))
group2<-grep("Group2",colnames(dati))
subjects<-c()
subjects[group1]<-1
subjects[group2]<-2
Group<-factor(subjects)

#create the design matrix
design <- model.matrix(~0+Group)
rownames(design)<-colnames(dati)
```

Now that we have the design matrix, we make the necessary steps to prepare the data in order to calculate the pvalues for the DE analysis.

```
y<-DGEList(dati)

#filtering low expression genes
keep<-filterByExpr(y,design)
y<-y[keep, ,keep.lib.sizes=FALSE]
y<-calcNormFactors(y)

#estimating dispersion
```

```
y<-estimateDisp(y,design)

#fitting the model
#fit<-glmQLFit(y,design)
fit<-glmFit(y,design)
contrasts <- makeContrasts(DE = Group1-Group2,levels=design)
RES<-glmLRT(fit,contrast=contrasts[,"DE"])
#RES<-glmQLFTest(fit,contrast=contrasts[,"DE"])
```

We can now extract the pvalues we obtained and visualize the first ten

```
#extracting p-values
pvalues<-RES$table$PValue
names(pvalues)<-rownames(RES$table)
head(pvalues, 10)
```

```
##         RFC2        HSPA6         PAX8       GUCA1A         UBA7         THRA
## 7.143792e-01 8.337671e-01 3.478801e-01 6.614234e-01 8.555494e-02 1.482313e-01
##       PTPN21         CCL5       CYP2E1        EPHB3
## 5.946946e-01 2.290693e-05 9.371521e-01 9.146354e-01
```

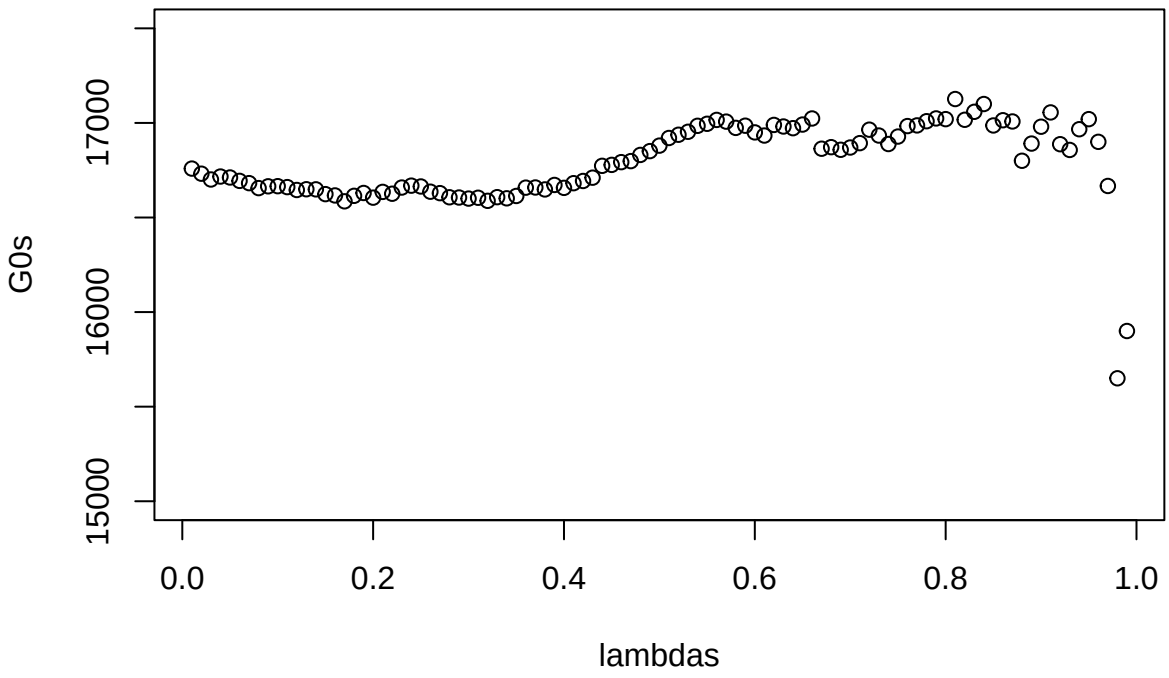
We now make an estimate of TN, FP, TP and FN assuming G0 is equal to the total number of genes

```
totgenes<-length(pvalues)
G0<-totgenes
selected<-length(pvalues[pvalues<=0.05])
expFP<-min(G0*0.05, selected)
expTP<-max(0,selected-expFP)
expTN<-min(G0*(1-0.05),totgenes-selected)
expFN<-totgenes-selected-expTN
valori<-c(expTN,expFP,expTP,expFN)
names(valori)<-c('E[TN]','E[FP]','E[TP]','E[FN]')
print(valori)
```

```
##    E[TN]    E[FP]    E[TP]    E[FN]
## 15876.00   851.15   295.85     0.00
```

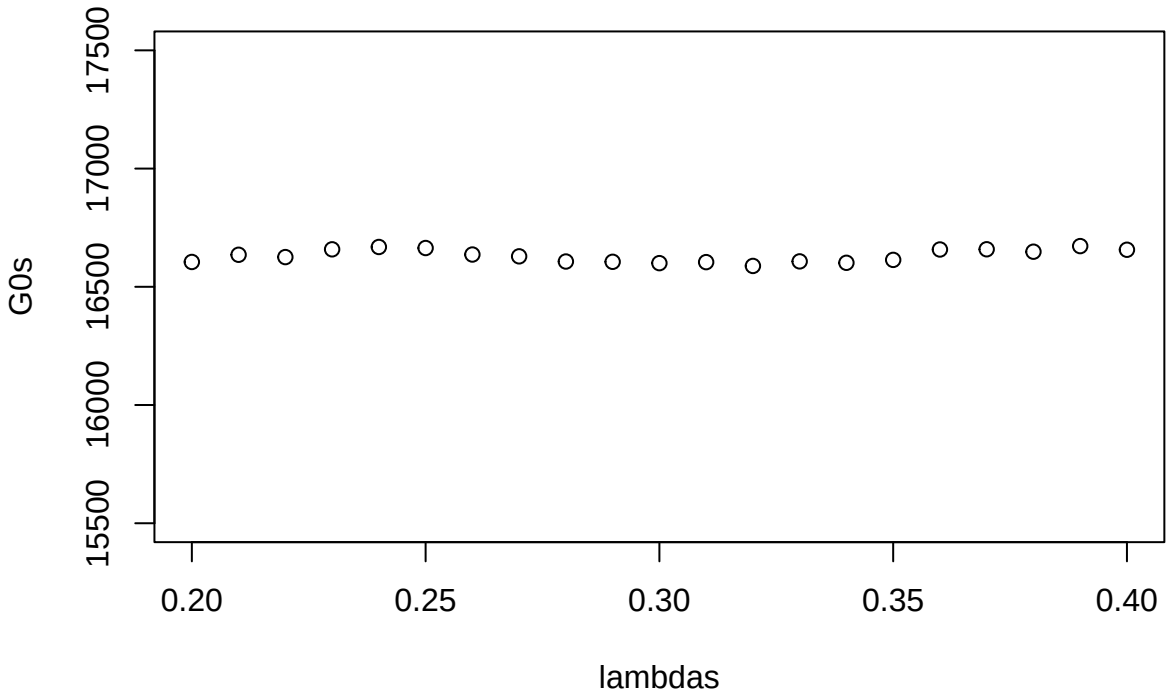
We estimate G0 using the method we've seen in class

```
totgenes<-length(pvalues)
#setting the lambda grid
lambdas<-seq(0.01,0.99,by=0.01)
G0s<-c()
for (i in lambdas){
  selected<-length(pvalues[pvalues<=i])
  predict<-(totgenes-selected)/(1-i)
  G0s<-append(G0s,predict)
}
plot(lambdas,G0s,ylim=c(15000,17500))
```



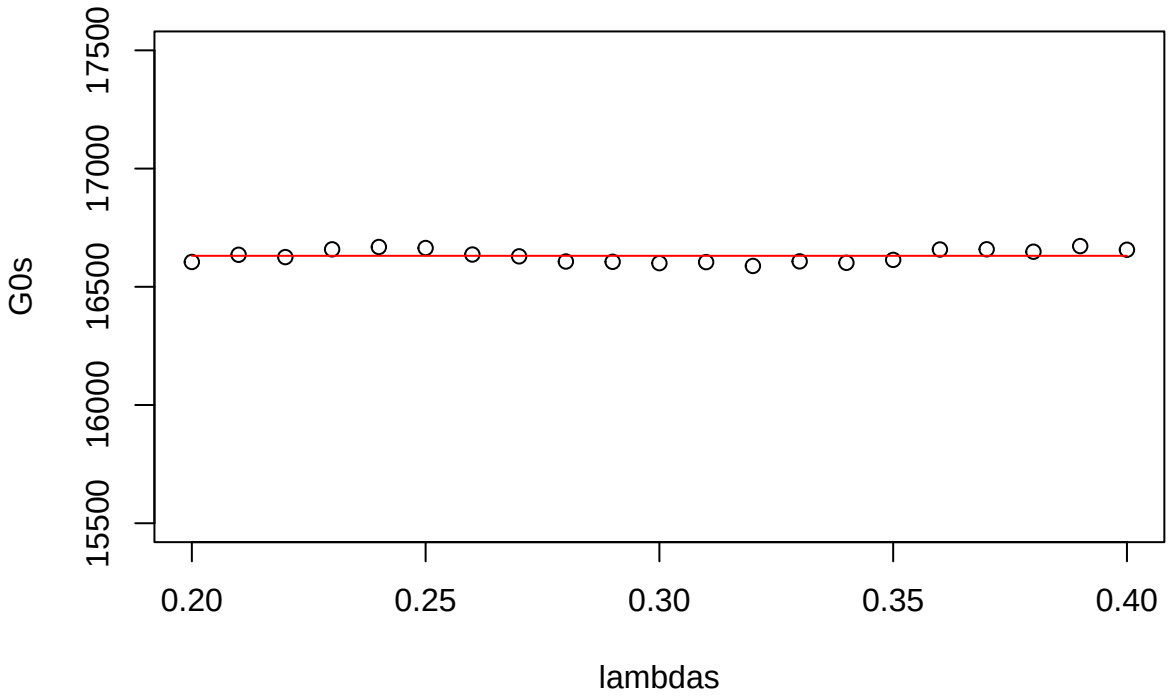
We focus on the area on which the estimate of G0 gets more or less stable between 0.2 and 0.4

```
portion<-G0s[match(0.2,lambdas):match(0.4,lambdas)]
lambdasportion<-lambdas[match(0.2,lambdas):match(0.4,lambdas)]
plot(lambdasportion,portion,ylim=c(15500,17500),xlab='lambdas',ylab='G0s')
```



And we find the value of the estimate that minimizes the RSS of this portion of graph

```
stimeG0<-seq(round(min(portion)),round(max(portion,by=10)))
somme<-c()
#find the value of G0 that minimizes average quadratic error
for (stima in stimeG0){
  somma<-0
  for (i in portion){
    somma<-somma+(stima-i)^2
  }
  somma<-somma/length(portion)
  somme<-append(somme,somma)
}
indice<-which(somme==min(somme))
valore<-stimeG0[indice]
plot(lambdasportion,portion,ylim=c(15500,17500),xlab='lambdas',ylab='G0s')
linea<-rep(valore,length(lambdasportion))
lines(lambdasportion,linea,col='red')
```



Let's see how the estimates of TN, FP, TP and FN change with the new G0

```
totgenes<-length(pvalues)
stima<-valore
G0<-stima
selected<-length(pvalues[pvalues<=0.05])
expFP<-min(G0*0.05, selected)
expTP<-max(0,selected-expFP)
expTN<-min(G0*(1-0.05),totgenes-selected)
expFN<-totgenes-selected-expTN
valori<-c(expTN,expFP,expTP,expFN)
names(valori)<-c('E[TN]','E[FP]','E[TP]','E[FN]')
print(valori)
```

```
##    E[TN]    E[FP]    E[TP]    E[FN]
## 15799.45   831.55   315.45    76.55
```

Now that we have our estimate of G0, we find the confidence level alpha that gives a FDR=0.05. Firstly we create the grid of the alphas we will use to find our confidence level

```
ordinati<-sort(pvalues)
ordinati<-unique(ordinati)
distanza<-1
#find minimum distance
  for ( i in 1:(length(ordinati)-1)){
  if (ordinati[i+1]-ordinati[i]<distanza){
```

```
    distanza<-ordinati[i+1]-ordinati[i]
  }
}
#set epsilon as half the minimum distance
epsilon<-distanza/2
#create grid of pvalues
alfatest<-c()
alfatest<-ordinati+epsilon
```

Then we look for the alpha for which the distance between FDR and 0.05 is minimum

```
#find the alfa which gives the closest FDR to the one requested
FDRlist<-c()
for (alfa in alfatest){
  selected<-length(pvalues[pvalues<=alfa])
  expFP<-stima*alfa#round?
  FDRlist<-append(FDRlist,expFP/selected)
}
diff<-FDRlist-0.05
indice<-which(abs(diff)==min(abs(diff)))
selectedalfa<-alfatest[indice]
coppia<-c(FDRlist[indice],selectedalfa)
names(coppia)<-c('FDR','alpha')
print(coppia)
```

```
##          FDR        alpha
## 0.0497862255 0.0003352809
```

Now we select genes with pvalues<=alpha

```
selezionati<-pvalues[pvalues<=selectedalfa]
selezionati<-names(selezionati)
print(selezionati)
```

```
##   [1] "CCL5"     "NEXN"     "STON2"    "MAB21L3"  "GPNMB"    "SDC3"
##   [7] "SCML4"    "ENTPD4"   "LILRA5"   "SLA2"     "FAM111B"  "WHAMMP2"
##  [13] "A2M"      "PIK3R6"   "ATXN1"    "DAB2"     "NRP1"     "SIGLEC16"
##  [19] "SMPDL3A"  "FCER1A"   "ITGAX"    "A2M-AS1"  "ITPRIPL2" "SLC1A3"
##  [25] "SPARC"    "SCD"      "CST3"     "TGFBI"    "RNASE1"   "PSRC1"
##  [31] "AXL"      "LTBP1"    "CSF1R"    "DAPK1"    "F13A1"    "TBC1D4"
##  [37] "RTN1"     "HMOX1"    "CXCL12"   "PRKAR2B"  "GUCY1B3"  "PHACTR2"
##  [43] "MRC1"     "ACP5"     "TGFBR3"   "DSC2"     "VSIG4"    "FOLR2"
##  [49] "PDGFA"    "GNLY"     "MMRN1"    "CD86"     "GFRA2"    "ABLIM3"
##  [55] "ADAM17"   "CPM"      "ADORA3"   "PLA2G7"   "EGF"      "PF4"
##  [61] "CD36"     "ITGA2B"   "SIGLEC6"  "CD5L"     "TFEC"     "RAB27B"
##  [67] "CD226"    "GP1BA"    "P2RY1"    "PROS1"    "PF4V1"    "NR4A3"
##  [73] "GRAP2"    "TUBB1"    "RGL1"     "CCL18"    "PRKCQ"    "PMP22"
##  [79] "TLR5"     "GZMH"     "MTUS1"    "SDC2"     "FAM171A1" "SORT1"
##  [85] "FXYD6"    "C1QA"     "MAFB"     "SDPR"     "LYVE1"    "SLAMF8"
##  [91] "TMEM40"   "C1orf54"  "SIGLEC1"  "MS4A6A"   "GAL3ST4"  "CLIP4"
##  [97] "MYCT1"    "CLEC1B"   "SH3BP4"   "CYP2S1"   "ACRBP"    "RGS18"
## [103] "FGFBP2"   "P2RY12"   "SH3BGRL2" "ESAM"     "ELOVL7"   "SMIM10"
## [109] "CMTM5"    "TDRD6"    "C3orf35"  "UTS2D"
```

We now see the final estimates of TN, FP, TP and FN

```
totgenes<-length(pvalues)
G0<-stima
selected<-length(pvalues[pvalues<=selectedalfa])
expFP<-min(G0*selectedalfa, selected)
expTP<-max(0,selected-expFP)
expTN<-min(G0*(1-selectedalfa),totgenes-selected)
expFN<-totgenes-selected-expTN
valori<-c(expTN,expFP,expTP,expFN)
names(valori)<-c('E[TN]','E[FP]','E[TP]','E[FN]')
print(valori)
```

```
##        E[TN]        E[FP]        E[TP]        E[FN]
## 16625.423943     5.576057   106.423943   285.576057
```

T-test analysis

Using the normalized data, we apply the t-test and extract the pvalues

```
group1 <- normali[,15:19]
group2 <- normali[,1:14]
alpha <- 0.05
n_rows <- length(normali[,1])
t_test_result <- c()
for (i in (1:n_rows)){
  t_test_result[i] <- t.test(group1[i,], group2[i,], alternative = "two.sided",
                             0, conf.level = alpha)$p.value
}
names(t_test_result)<-rownames(normali)
```

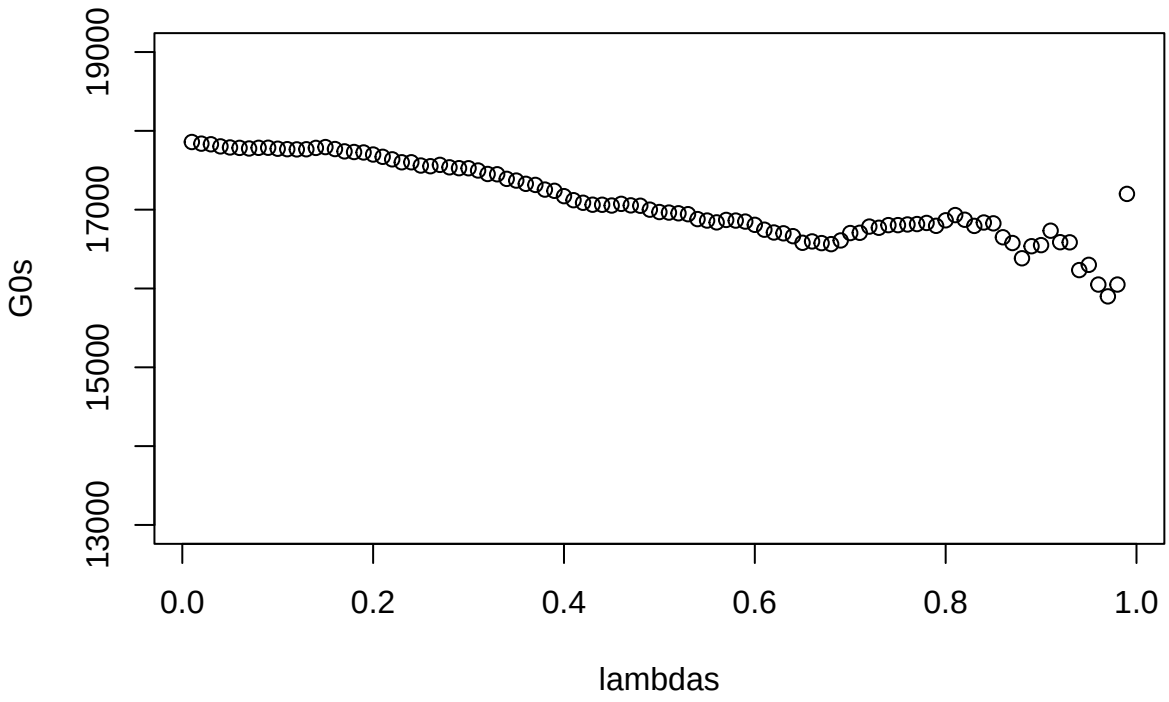
We now make an estimate of TN, FP, TP and FN assuming G0 is equal to the total number of genes

```
totgenes<-length(t_test_result)
G0<-totgenes
selected<-length(t_test_result[t_test_result<=0.05])
expFP<-min(G0*0.05, selected)
expTP<-max(0,selected-expFP)
expTN<-min(G0*(1-0.05),totgenes-selected)
expFN<-totgenes-selected-expTN
valori<-c(expTN,expFP,expTP,expFN)
names(valori)<-c('E[TN]','E[FP]','E[TP]','E[FN]')
print(valori)
```

```
##   E[TN]   E[FP]   E[TP]   E[FN]
## 16900.0   896.7   137.3     0.0
```

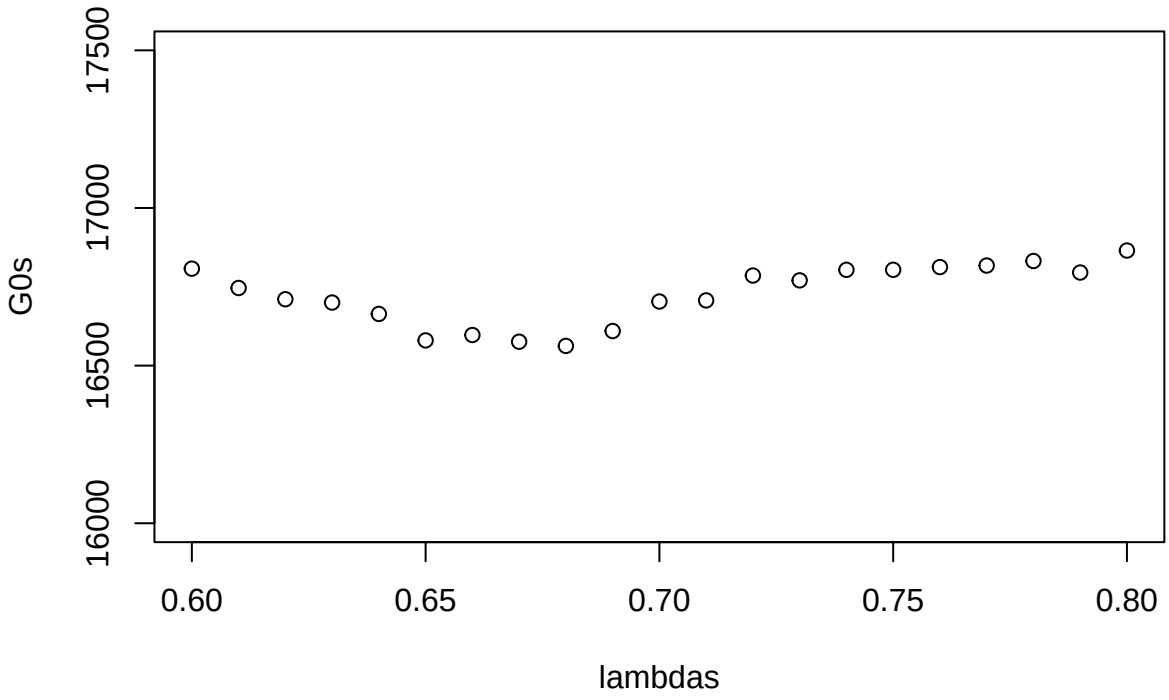
We estimate G0 using the method we've seen in class

```
totgenes<-length(t_test_result)
#setting the lambda grid
lambdas<-seq(0.01,0.99,by=0.01)
G0s<-c()
for (i in lambdas){
  selected<-length(t_test_result[t_test_result<=i])
  predict<-(totgenes-selected)/(1-i)
  G0s<-append(G0s,predict)
}
plot(lambdas,G0s,ylim=c(13000,19000))
```



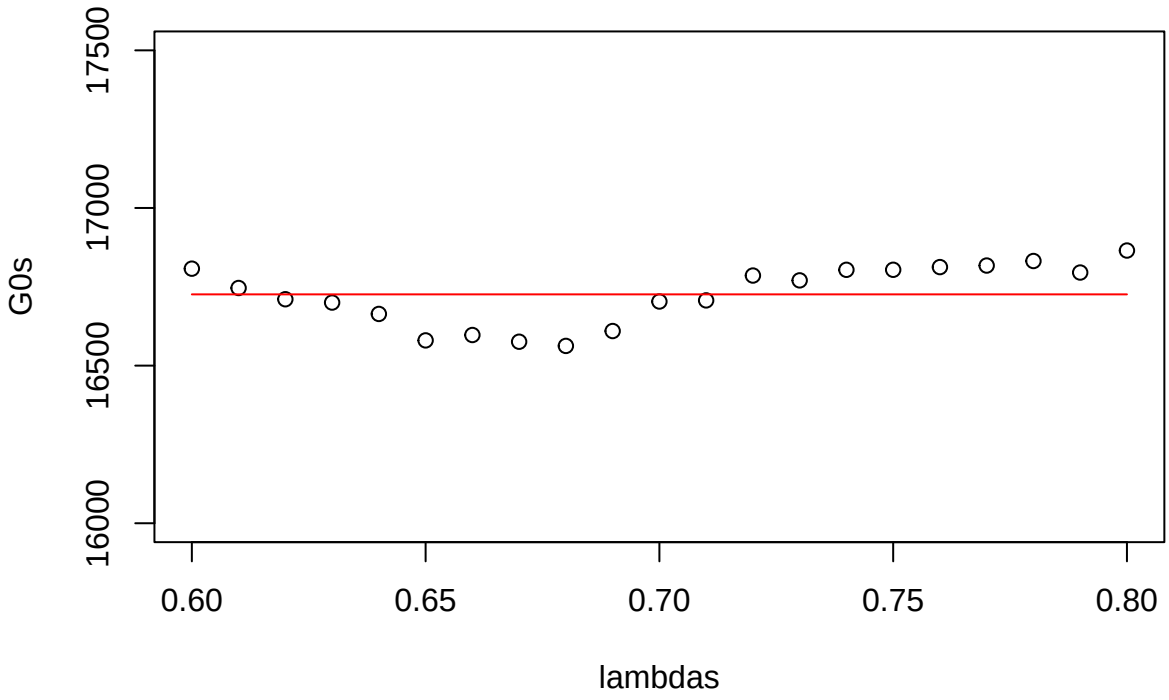
We focus on the area on which the estimate of G0 gets more or stable between 0.6 and 0.8

```
portion<-G0s[match(0.6,lambdas):match(0.8,lambdas)]
lambdasportion<-lambdas[match(0.6,lambdas):match(0.8,lambdas)]
plot(lambdasportion,portion,ylim=c(16000,17500),xlab='lambdas',ylab='G0s')
```



And we find the value of the estimate that minimizes the RSS of this portion of graph

```
stimeG0<-seq(round(min(portion)),round(max(portion,by=10)))
somme<-c()
#find the value of G0 that minimizes average quadratic error
for (stima in stimeG0){
  somma<-0
  for (i in portion){
    somma<-somma+(stima-i)^2
  }
  somma<-somma/length(portion)
  somme<-append(somme,somma)
}
indice<-which(somme==min(somme))
valore<-stimeG0[indice]
plot(lambdasportion,portion,ylim=c(16000,17500),xlab='lambdas',ylab='G0s')
linea<-rep(valore,length(lambdasportion))
lines(lambdasportion,linea,col='red')
```



Let's see how the estimates of TN, FP, TP and FN change with the new G0

```
totgenes<-length(t_test_result)
stima<-valore
selected<-length(t_test_result[t_test_result<=0.05])
expFP<-min(stima*0.05, selected)
expTP<-max(0,selected-expFP)
expTN<-min(stima*(1-0.05),totgenes-selected)
expFN<-totgenes-selected-expTN
valori<-c(expTN,expFP,expTP,expFN)
names(valori)<-c('E[TN]','E[FP]','E[TP]','E[FN]')
print(valori)
```

```
##   E[TN]   E[FP]   E[TP]   E[FN]
## 15889.7   836.3   197.7  1010.3
```

Now that we have our estimate of G0, we find the confidence level alpha that gives a FDR=0.05. Firstly we create the grid of the alphas we will use to find our confidence level

```
ordinati<-sort(t_test_result)
ordinati<-unique(ordinati)
distanza<-1
#find minimum distance
  for ( i in 1:(length(ordinati)-1)){
  if (ordinati[i+1]-ordinati[i]<distanza){
    distanza<-ordinati[i+1]-ordinati[i]
```

```
  }
}
#set epsilon as half the minimum distance
epsilon<-distanza/2
#create grid of pvalues
alfatest<-c()
alfatest<-ordinati+epsilon
```

Then we look for the alpha for which the distance between FDR and 0.05 is minimum

```
#find the alfa which gives the closest FDR to the one requested
FDRlist<-c()
for (alfa in alfatest){
  selected<-length(t_test_result[t_test_result<=alfa])
  expFP<-stima*alfa#round?
  FDRlist<-append(FDRlist,expFP/selected)
}
diff<-FDRlist-0.05
indice<-which(abs(diff)==min(abs(diff)))
selectedalfa<-alfatest[indice]
coppia<-c(FDRlist[indice],selectedalfa)
names(coppia)<-c('FDR','alpha')
print(coppia)
```

```
##         FDR       alpha
## 8.85103e-03 5.29178e-07
```

Since we do not obtain a good FDR, we can try to print the first six sorted values of the absolut values of the difference between the obtained FDR and 0.05

```
print(head(sort(abs(diff))))
```

```
## [1] 0.04114897 0.06974425 0.22560144 0.26782035 0.29430469 0.50568662
```

We can see that already the third closest FDR is 0.2256 away from 0.05

Now we select genes with pvalues<=alpha

```
selezionati<-t_test_result[t_test_result<=selectedalfa]
selezionati<-names(selezionati)
print(selezionati)
```

```
## [1] "PRICKLE2"
```

These bad results may be due to the gaussian assumption at the base of the t-test

Wilcoxon test analysis

Using the normalized data, we apply the Wilcoxon test and extract the pvalues

```
group1 <- normali[, c(15:19)]
group2 <- normali[, c(1:14)]
alpha <- 0.05
n_rows <- length(normali[,1])
wilcoxon_test_result <- c()
for (i in (1:n_rows)){
  wilcoxon_test_result[i] <- wilcox.test(as.numeric(group1[i,]), as.numeric(group2[i,]),
                                         alternative = "two.sided", mu = 0, paired = FALSE)$p.value

}
names(wilcoxon_test_result)<-rownames(data)
```

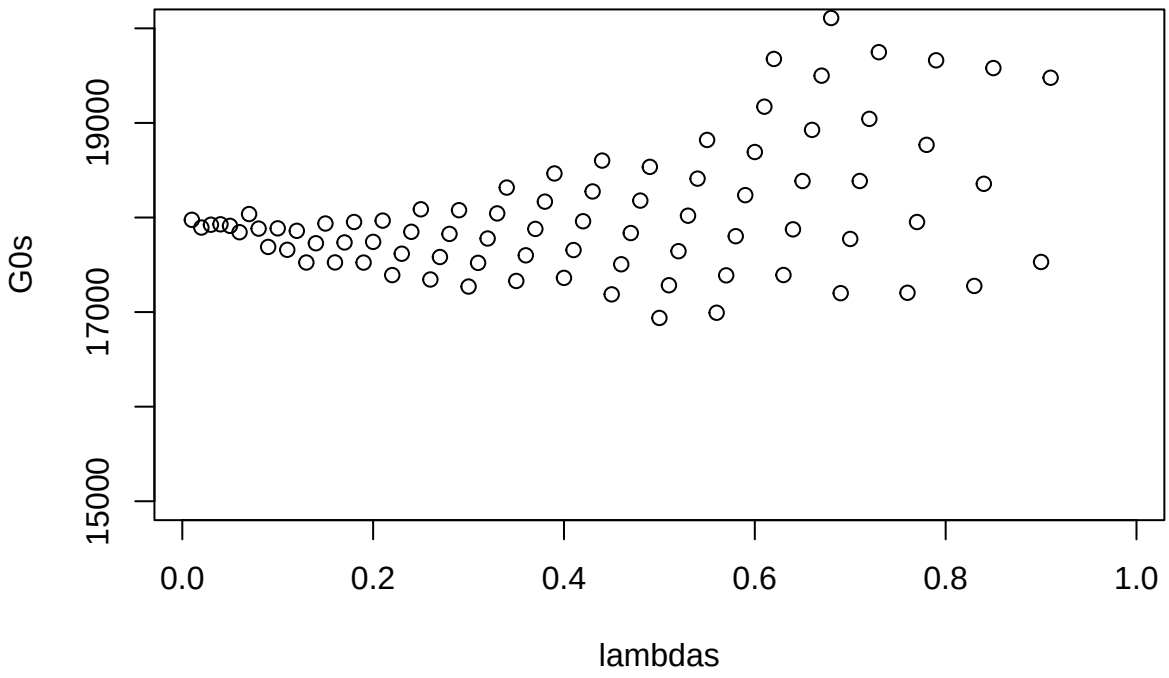
We now make an estimate of TN, FP, TP and FN assuming G0 is equal to the total number of genes

```
pvalues<-wilcoxon_test_result
totgenes<-length(pvalues)
G0<-totgenes
selected<-length(pvalues[pvalues<=0.05])
expFP<-min(G0*0.05, selected)
expTP<-max(0,selected-expFP)
expTN<-min(G0*(1-0.05),totgenes-selected)
expFN<-totgenes-selected-expTN
valori<-c(expTN,expFP,expTP,expFN)
names(valori)<-c('E[TN]','E[FP]','E[TP]','E[FN]')
print(valori)
```

```
##   E[TN]   E[FP]   E[TP]   E[FN]
## 17016.0   896.7    21.3     0.0
```

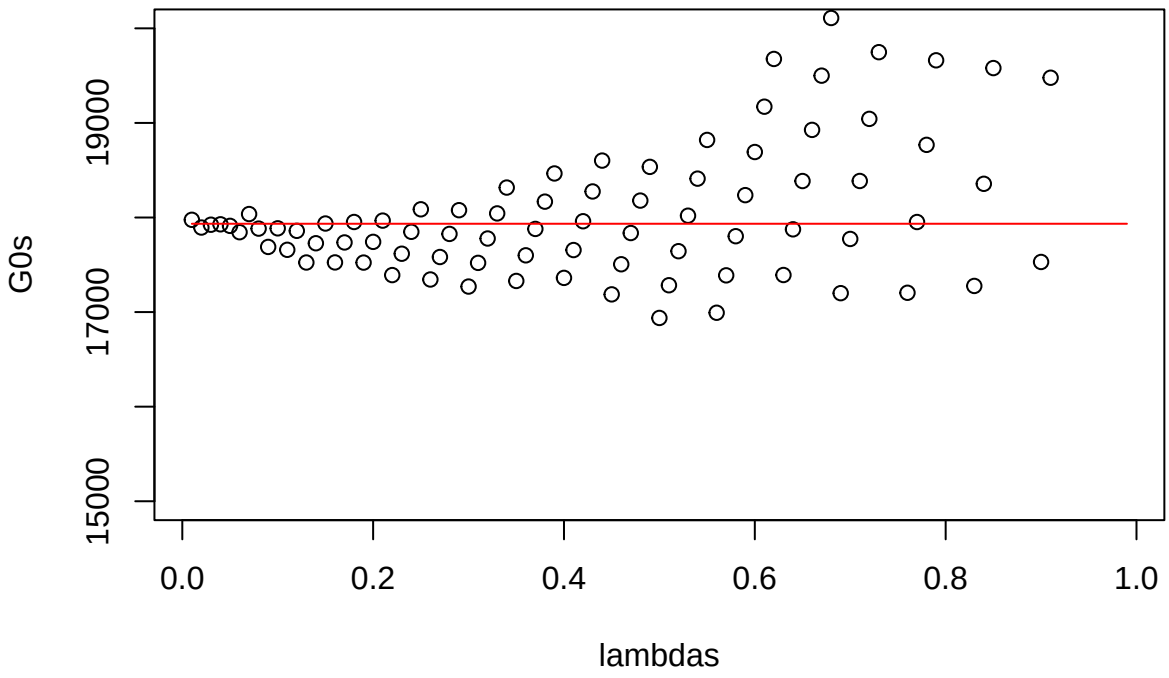
We estimate G0 using the method we've seen in class

```
totgenes<-length(pvalues)
#setting the lambda grid
lambdas<-seq(0.01,0.99,by=0.01)
G0s<-c()
for (i in lambdas){
  selected<-length(pvalues[pvalues<=i])
  predict<-(totgenes-selected)/(1-i)
  G0s<-append(G0s,predict)
}
plot(lambdas,G0s,ylim=c(15000,20000))
```



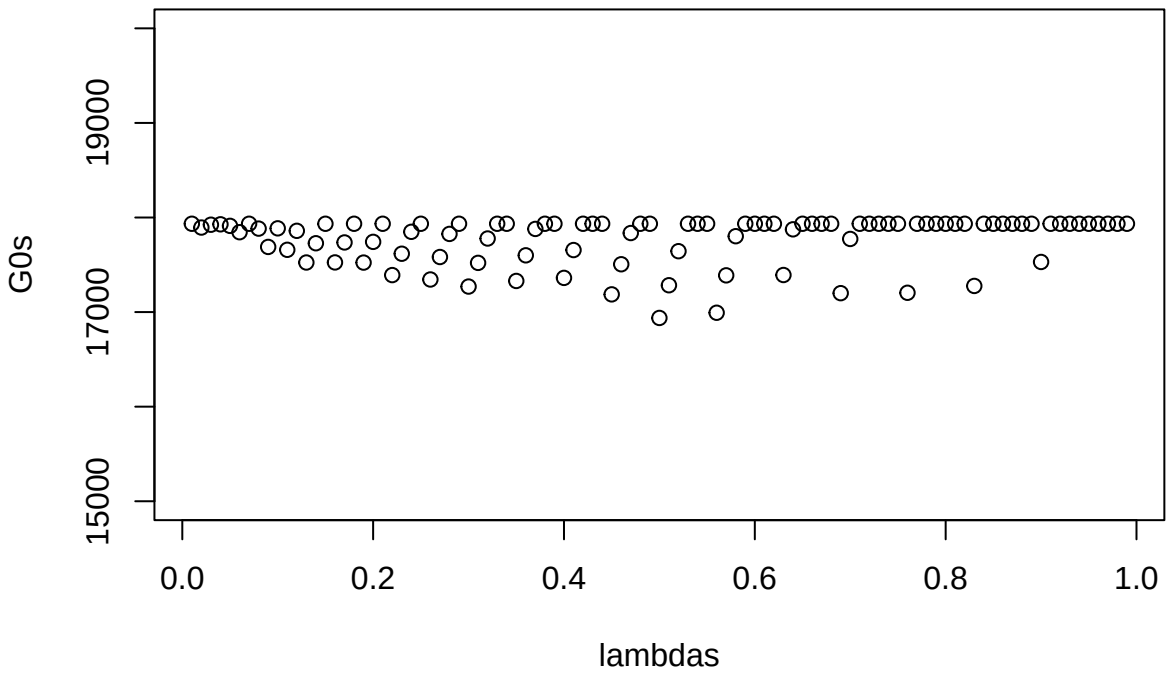
If we add to the plot the line corresponding to the total number of genes, we can see that a good number of the estimates of G0 are greater than this value.

```
plot(lambdas,G0s,ylim=c(15000,20000))
linea<-rep(totgenes,length(lambdas))
lines(lambdas,linea,col='red')
```



We can fix think about fixing as maximum value for the estimate the total number of genes, obtaining the following graph

```
totgenes<-length(pvalues)
#setting the lambda grid
lambdas<-seq(0.01,0.99,by=0.01)
G0s<-c()
for (i in lambdas){
  selected<-length(pvalues[pvalues<=i])
  predict<-min((totgenes-selected)/(1-i),totgenes)
  G0s<-append(G0s,predict)
}
plot(lambdas,G0s,ylim=c(15000,20000))
```



But even applying this correction, it's almost useless to try to estimate G0 using the method seen in class. These bad results may be due to the fact that the wilcoxon test considers only the order of values. In addition to this, the fact that we have a little number of samples(only 5 for group 1) makes it hard for this type of test to obtain consistent results.