Exercise on PWMs

Introduction to R for Bioinformatics HY-390.51

Mourouzidou Eleni

24/03/2023

RETRIEVE DATA

After downloading from Ensembl a file containing all the gene IDs of homo sapiens (at the 1st column), we used *sample* function to pick 1000 gene IDs out of a total number of 69299 and saved it at a new table file named "rand1000IDs.tsv".
This file was used as a query to Ensembl in order to download data containing the 500 nt long upstream regions for each of these 1000 genes. These data were downloaded and saved to a new file called "1000randgenes.txt".
Our two other datasets called "glycolysis.txt" and "gluconeogenesis.txt", similarly contain the sequences of 500 nt long upstream regions of each gene of glycolysis and gluconeogenesis pathway respectively.

Using *grep* function gene IDs for each dataset were isolated and then with *intersect* function we found the common gene IDs between the two pathways of interest.

Question 1:
Do these two lists of gene names have more common genes than expected by chance?
Between the two pathways there are 14 common gene IDs.
Using the hypergeometric distribution (*phyper)* we find out whether this number of common genes is greater than the expected number of common genes that would occur by chance.
The p-value that comes as a result is `3.313567e-34` or `0.00000000000000000000000000000000003313567,` which is very small and enough to assume that is unlikely to find that many common genes between two random datasets of the same size

```
#read the downloaded file containing the IDs of all human genes
geneIDs <- read.table("allgeneids.txt", sep = "\t")
IDs <- unique(geneIDs$V1)
```

```r
#pick 1000 random genes and save it to a file to get the entries from
Ensembl
random_gene_ids <- unique(sample(IDs, size = 1000, replace = FALSE))

write.table(random_gene_ids, "rand1000IDs.tsv", row.names = FALSE,
col.names = FALSE, quote = FALSE)


glycolysis <- "glycolysis.txt"
gluconeogenesis <- "gluconeogeneis.txt"
rand1000genes <- "1000randgenes.txt"

#readLines for every file-sample
glyc_lines <- readLines(glycolysis)
glucon_lines <- readLines(gluconeogenesis)
all_lines <- readLines(rand1000genes)

# find the gene IDs for each pathway
glycolysisIDs <- grep("^>", glyc_lines, value = TRUE)
gluconeogenesisIDs <- grep("^>", glucon_lines, value = TRUE)

commonGenes <- intersect(gluconeogenesisIDs, glycolysisIDs)

ngly <- length(glycolysisIDs)
nglu <- length(gluconeogenesisIDs)
sample_size <- ngly + nglu

#calculate the p-value to check the significance of common occurancies
common_prob <- phyper(length(commonGenes)-1, nglu, length(IDs)-nglu,
ngly, lower.tail = FALSE)

format(common_prob, scientific = FALSE)
```

Using the getData function we created a list of all the gene IDs and the upstream sequence for each one.

```r
getData = function(filepath) {
  con = file(filepath, "r")
  data = list()
  while ( TRUE ) {
    line = readLines(con, n = 1)
    if ( length(line) == 0 ) {
      break
    }
    if(length(grep(">", line)) > 0){
      name = gsub("^>(\\w+)", replacement="\\1", x=line)
      data[[name]] = ""
    }
    else{
      data[[name]] = paste(data[[name]], line, sep="")
    }
  }
  close(con)
  return(data)
}

glycolysis_data <- getData(glycolysis)
gluconeogenesis_data <- getData(gluconeogenesis)
background <- getData(rand1000genes)
```

Using the processSequence function we count the number of occurrences of all the possible 8 nt long substrings that are present on every gene of each dataset. Then, using the getProb function we calculate the p-value, using the hypergeometric distribution of the overrepresentation of the substrings in the foreground datasets (glycolysis and gluconeogenesis) compared to the background dataset of the 1000 random genes.

Question 2:
How many substrings with p-value < 0.001 there are in the glycolysis genes and how many in the gluconeogenesis genes?

As a result, 487 substrings of the glycolysis pathway and 175 substrings of the gluconeogenesis pathway can be characterized as overrepresented based on the p-value < 0.001

Are there common substrings that are overrepresented in the two pathways? i.e. how many substrings with p-value < 0.001 are common in both pathways?

There are 28 overrepresented substrings with p-value < 0.001 that are common in both pathways.

```
processSequences = function(seqList, len=5){
  x = sapply(seqList, function(i){
    v = strsplit(i,"")[[1]]
    sapply(1:(length(v)-len+1), function(j){paste(v[j:(j+len-1)],
collapse="")})
```

```
  })
  table(x)
}

motifLength=8

fgGluconeogenesisDataset <- processSequences(gluconeogenesis_data,
motifLength)
fgGlycolysisDataset <- processSequences(glycolysis_data, motifLength)
bgDataset <- processSequences(background, motifLength)

fgGlycolCounts <- sum(fgGlycolysisDataset)
fgGluconCounts <- sum(fgGluconeogenesisDataset)
bgCounts <- sum(bgDataset)

getProb = function(foreground, background){
  probs = vector("numeric", length=length(foreground))
  sumforground = sum(foreground)
  sumbackground = sum(background)
  for(i in 1:length(foreground)){
    bcounts = 0
    if( names(foreground)[i] %in% names(background)){
      bcounts = background[[names(foreground)[i]]]
      prob = phyper(q=foreground[i]-1,  m = bcounts, n = sumbackground -
bcounts, k = sumforground, lower.tail = FALSE)
      probs[i] = prob
    }else{
      bcounts = 1
      prob = phyper(q=foreground[i]-1,  m = bcounts, n = sumbackground -
bcounts, k = sumforground, lower.tail = FALSE)
      probs[i] = prob
    }
  }
  names(probs) = names(foreground)
  return(sort(probs, decreasing=FALSE))
}

hyperGluconprobs = getProb(fgGluconeogenesisDataset, bgDataset)
# count the number of 8-mer substrings of gluconeogenesis pathway with
p-value<0.001
overGluc <- names(hyperGluconprobs)[which(hyperGluconprobs<0.001)]
length(overGluc)
#res = 175
hyperGlycolprobs = getProb(fgGlycolysisDataset, bgDataset)
# count the number of 8-mer substrings of glycolysis pathway with
p-value<0.001
```

```
overGly <- names(hyperGlycolprobs)[which(hyperGlycolprobs<0.001)]
length(overGly)
#res = 487

#find the common overrepresenting substring in both pathways
intersect(overGluc, overGly)
```

Question 3:
Build the PWM for each pathway for the substring with the smallest p-value.
The substring with the smallest p-value for gluconeogenesis pathway is AATCTCGC and for glycolysis pathway AAAGACGC.

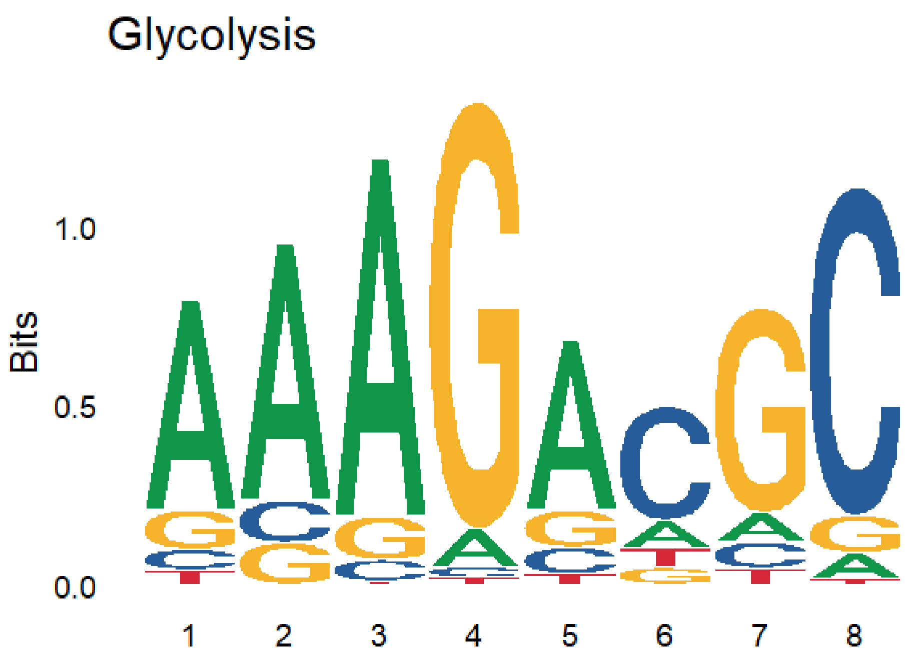
```
# get the 8mer with the smallest p-value for each pathway
mostfreqGlu <- names(hyperGluconprobs[1])
#                  _____AATCTCGC____

mostfreqGly <- names(hyperGlycolprobs[1])
#                  _____AAAGACGC____
```

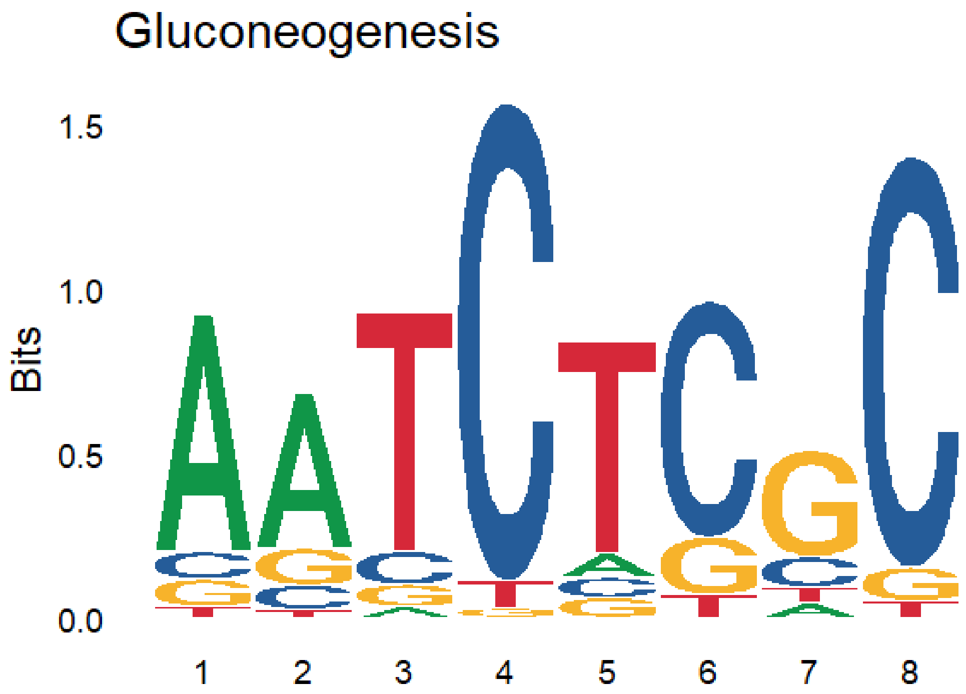
```
# find sequences similar to the substring with the smallest p-value for each pathway
getAllInstances = function(candidate, foreground, threshold){
  allnames = names(foreground)
  motifstrings = c()
  for(i in 1:length(allnames)){
    if( stringdist(candidate, allnames[i], method = "hamming") < threshold){
      motifstrings = c(motifstrings, rep(allnames[i], foreground[i]))
    }
  }
  return(motifstrings)
}

stringGlycolMotifs = getAllInstances(mostfreqGly, fgGlycolysisDataset,
3)
stringGluconMotifs = getAllInstances(mostfreqGlu,
fgGluconeogenesisDataset, 3)




getACGT = function(dataset, alphabet=c("A", "C", "G", "T")){
  counts = vector("numeric", length=length(alphabet))
  names(counts) = alphabet
  for(i in 1:length(names(dataset))){
    nam = strsplit(names(dataset)[i], "")[[1]]
    ntall = rep(nam, each=dataset[i])
    ntcounts = table(factor(ntall, levels=alphabet))
    counts[names(ntcounts)] = counts[names(ntcounts)] + ntcounts
  }
  counts/sum(counts)
}

basefreqs = getACGT(bgDataset)
```

Then, using the getPWM function we build the position weight matrix for the substrings with the smallest p-value for each pathway and visualize the results with ggseqlogo.

```
getPWM = function(stringMotifs, length=6, alphabet =c("A", "C", "G",
"T"),  freqs = rep(0.25, 4)){
  pfm = matrix(0, nrow=4, ncol=length)
  row.names(pfm) = alphabet
  for(i in 1:length(stringMotifs)){
    v = unlist(strsplit(stringMotifs[i], "")) ## or
strsplit(stringMotifs[i], "")[[1]]
    for(j in 1:length(v)){
      pfm[v[j], j] = pfm[v[j], j] + 1
    }
  }
  ppm = pfm/colSums(pfm)
  pwm = pwm = log2((ppm+1e-4)/freqs)
  return(list(pwm=pwm, ppm=ppm))
}

psGlucon <- getPWM(stringMotifs = stringGluconMotifs,
```

```r
length=motifLength, freqs = basefreqs)
psGlycol <- getPWM(stringMotifs = stringGlycolMotifs,
length=motifLength, freqs = basefreqs)

ggseqlogo(psGlucon$ppm) + ggtitle("Gluconeogenesis")
ggseqlogo(psGlycol$ppm) + ggtitle("Glycolysis")
```


Glycolysis
Bits
1.0
0.5
0.0
1
2
3
4
5
6
7
8


Gluconeogenesis
Bits
1.5
1.0
0.5
0.0
1
2
3
4
5
6
7
8

```r
getScoreSimple  = function(vstring, pwm){
  score = 0
  v = strsplit(vstring, "")[[1]]
  scores = vector("numeric", length=length(v)-ncol(pwm)+1)
  for(i in 1:(length(v)-ncol(pwm)+1)){
    score = 0
    for(j in 1:ncol(pwm)){
      letter = v[i+j-1]
      score = score + pwm[letter, j]
    }
    scores[i] = score
  }
  return(scores)
}

#Gluconeogenesis
resGluc = t(sapply(gluconeogenesis_data, getScoreSimple, psGlucon$pwm))
matrix(as.numeric(resGluc > 3), nrow=nrow(resGluc))


#glycolysis
resGly = t(sapply(glycolysis_data, getScoreSimple, psGlycol$pwm))

#   find the maximum score for each sequence in gluconeogenesis and
glycolysis
maxScoresGluc <- apply(resGluc, 1, max)
maxScoresGly <- apply(resGly, 1, max)

# glycolysis matrix to scan gluconeogenesis
glyScanGlu = t(sapply(gluconeogenesis_data, getScoreSimple,
psGlycol$pwm))
maxGluScannedByGly <- apply(glyScanGlu, 1, max)

# gluconeogenesis matrix to scan glycolysis
gluScanGly = t(sapply(glycolysis_data, getScoreSimple, psGlucon$pwm))
maxGlyScannedByGlu <- apply(gluScanGly, 1, max)



logicalresGluc = matrix(as.numeric(resGluc > -4), nrow=nrow(resGluc))
heatmap.2(logicalresGluc, dendrogram='none', Rowv=F, Colv=F, trace =
'none', main = "Gluconeogenesis")
```

```r
logicalresGly = matrix(as.numeric(resGly > -4), nrow=nrow(resGly))
logicalresGlu = matrix(as.numeric(resGluc > -4), nrow=nrow(resGluc))

heatmap.2(logicalresGly, dendrogram='none', Rowv=F, Colv=F, trace =
'none', main = "Glycolysis")
heatmap.2(logicalresGlu, dendrogram='none', Rowv=F, Colv=F, trace =
'none', main = "Gluconeogenesis")
```

Here are the 67 maximum scores for glycolysis pathway after scanning the glycolysis dataset (the maximum score for each sequence), and the 35 maximum scores for each sequence of gluconeogenesis pathway respectively:

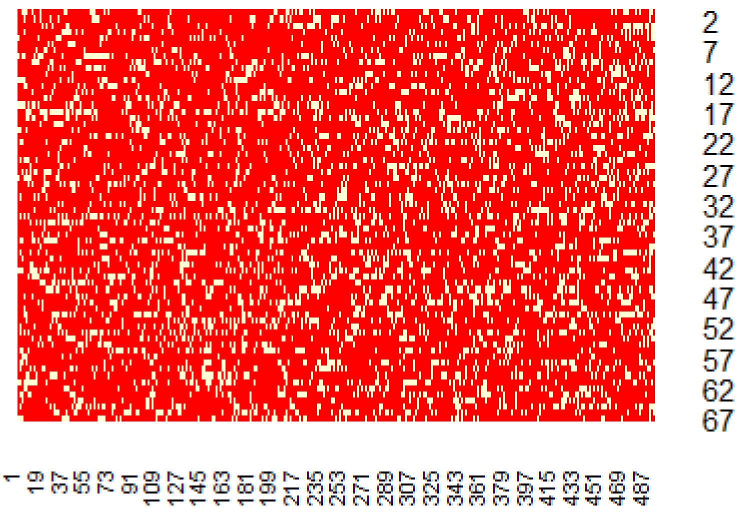
```
          Glycolysis

ENSG00000124789 7.8159702313515
ENSG00000196313 6.15436256886049
ENSG00000170950 7.85835575369791
ENSG00000165059 3.9731897716705
ENSG00000164708 7.55065090792738
ENSG00000102900 6.87902971011562
ENSG00000155561 3.93931619497958
ENSG00000111640 7.11071337574655
ENSG00000113552 7.8159702313515
ENSG00000171314 9.89821214242804
ENSG00000138750 7.69264371615224
ENSG00000158571 12.7386996916188
ENSG00000120253 8.0869659359312
ENSG00000157020 8.2753180491825
ENSG00000125450 9.76540767962366
ENSG00000058804 7.61559017345463
ENSG00000141959 8.2753180491825
ENSG00000104695 6.83624582773489
ENSG00000136243 7.8159702313515
ENSG00000109107 8.12054443279028
ENSG00000106633 5.12998794194299
ENSG00000213024 7.68316576854713
```

```
ENSG00000160883 5.78113814203749
ENSG00000172331 8.54063737260663
ENSG00000105220 5.25331445714225
ENSG00000075188 7.86206117976794
ENSG00000113575 7.60875410305284
ENSG00000108515 7.44509888428335
ENSG00000093000 10.3911384571181
ENSG00000153201 6.63242963462173
ENSG00000069248 9.97417616163352
ENSG00000067057 5.27530343231847
ENSG00000112640 8.04295760663548
ENSG00000072062 7.61559017345463
ENSG00000074800 6.23775038794978
ENSG00000149925 5.32244240594595
ENSG00000272391 6.15436256886049
ENSG00000132182 7.00673711690617
ENSG00000156515 5.01473756017542
ENSG00000142875 8.30889654604158
ENSG00000123836 9.47291718381246
ENSG00000111674 12.7386996916188
ENSG00000114268 5.30202603718739
ENSG00000152556 10.1446674508048
ENSG00000105568 7.8159702313515
ENSG00000047410 7.79710621630414
ENSG00000159399 5.1231518715412
ENSG00000030066 7.417846445123
ENSG00000163002 8.99485011805123
ENSG00000165434 7.69264371615224
ENSG00000101146 8.54063737260663
ENSG00000108559 7.2458083498786
ENSG00000288516 7.61559017345463
ENSG00000137713 9.76540767962366
ENSG00000113569 6.24458645835156
ENSG00000111581 8.09380200633299
ENSG00000136872 10.1587780145711
ENSG00000126883 8.06242553972826
ENSG00000095319 7.61559017345463
ENSG00000159322 7.85835575369791
ENSG00000170525 5.74624077183233
ENSG00000102144 9.77488562722877
ENSG00000163281 9.47291718381246
ENSG00000094914 8.30889654604158
ENSG00000184207 7.44509888428335
ENSG00000105679 4.591453761762
ENSG00000111669 7.02084768067246
```

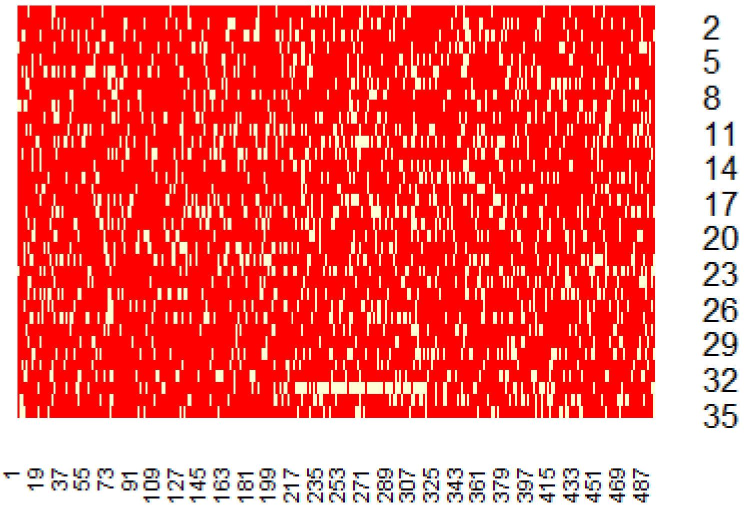
```
        Gluconeogenesis

ENSG00000170950 11.0259180531638
ENSG00000130957 9.18535117216109
ENSG00000164708 8.20719668855226
ENSG00000281500 9.65828472452806
ENSG00000285241 9.22596577424311
ENSG00000149925 8.98405781789421
ENSG00000125166 9.90373227149585
ENSG00000108515 9.64291940537734
ENSG00000160190 6.19911498496291
ENSG00000111640 6.5253583706422
ENSG00000074800 6.91246105467485
ENSG00000100075 7.6570390913405
ENSG00000165140 5.62142904164855
ENSG00000100889 9.22596577424311
ENSG00000102144 7.58248155035609
ENSG00000278373 8.15766290016383
ENSG00000014641 10.4482321098494
ENSG00000183048 4.63702437246208
ENSG00000105220 8.14383823490934
ENSG00000111669 8.40637187457985
ENSG00000173599 8.21423424512065
ENSG00000120053 7.58248155035609
ENSG00000108528 13.0271636863514
ENSG00000141349 6.63827000564964
ENSG00000109107 7.1954651594298
ENSG00000111674 5.00354997385417
ENSG00000146701 10.4482321098494
ENSG00000152254 8.15766290016383
ENSG00000136872 7.06398782887542
ENSG00000115840 7.09204846345271
ENSG00000131482 8.10377999257517
ENSG00000124253 9.22596577424311
ENSG00000171314 10.9853034510818
ENSG00000004864 13.0271636863514
ENSG00000105679 5.38635715570144
```

Scanning each dataset by the PWM of the other pathway gave different maximum scores for each sequence

Glycolysis



Gluconeogenesis


2
5
8
11
14
17
20
23
26
29
32
35
1 19 37 55 73 91 109 127 145 163 181 199 217 235 253 271 289 307 325 343 361 379 397 415 433 451 469 487