BioINF HW3

Ryan Gallagher

2023-10-09

### 1) Download R packages ‘SpikeIn’ and ‘hgu95acdf’ from bioconductor website. We are going to use this dataset to practice normalization methods.

#=== How to Install those packages ======  
#if (!require("BiocManager", quietly = TRUE))  
# install.packages("BiocManager")  
#  
#BiocManager::install("SpikeIn")  
  
#if (!require("BiocManager", quietly = TRUE))  
# install.packages("BiocManager")  
  
#BiocManager::install("hgu95acdf")  
#  
#========================================  
  
library(SpikeIn)  
library(hgu95acdf)

#### a) Use to load in R objects and run to extract a gene expression intensity matrix from perfect match probes. How many probesets and how many samples are in the matrix?

dat = data(SpikeIn95)  
matrix = pm(SpikeIn95)  
  
print(paste('There are', nrow(matrix), 'probesets in this matrix'))

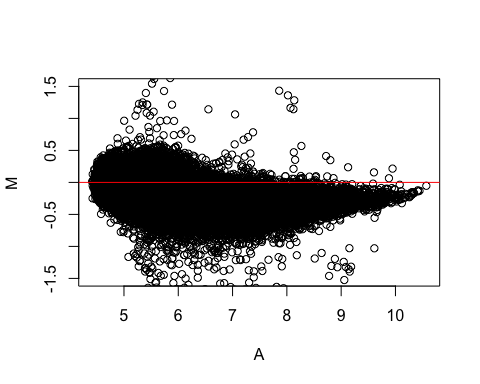
## [1] "There are 201807 probesets in this matrix"

print(paste('There are', ncol(matrix), 'samples in this matrix'))

## [1] "There are 59 samples in this matrix"

#### b) Draw a MA plot for any two samples. What do you observe?

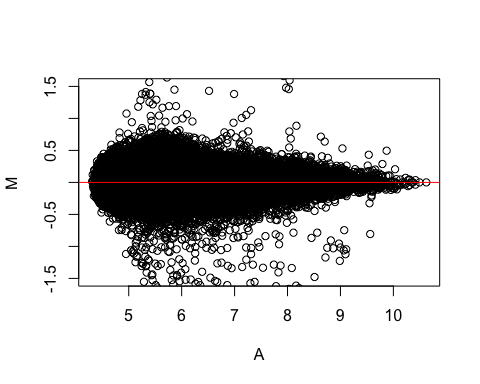
#Picking samples 25 and 29  
  
M = log(matrix[,25])-log(matrix[,29])  
A = (log(matrix[,25]) + log(matrix[,29]))/2  
  
plot(A,M, ylim=c(-1.5,1.5))  
abline(h=0, col='red')



This data is clearly not normal, as the data drifts in the negative M direction for larger A.

#### c) Write your own R code to implement quantile normalization. Apply to this dataset and show again MA plot to the samples that you choose for (b). What do you observe?

# We can normalize the data by using the log with base 2  
  
#Picking samples 25 and 29  
  
set.seed(1)  
qc = as.data.frame(matrix)  
  
quant.norm = function(df) {  
 df\_r = apply(df, 2,rank,ties.method='min')  
 df\_s = data.frame(apply(df,2,sort))  
 df\_mean = apply(df\_s,1,mean)  
   
 ind = function(m\_inx, m\_mean){  
 return(m\_mean[m\_inx])  
 }  
   
 df\_f = apply(df\_r, 2, ind, m\_mean=df\_mean)  
 rownames(df\_f) = rownames(df)  
 return(df\_f)  
}  
   
df\_norm = quant.norm(qc)  
  
M = log(df\_norm[,25])-log(df\_norm[,29])  
A = (log(df\_norm[,25]) + log(df\_norm[,29]))/2  
  
plot(A,M, ylim=c(-1.5,1.5))  
abline(h=0, col='red')



The data has been normalized around M=0.

### 2) Download R package ‘impute’. Use data(khanmiss) to access the R object khanmiss. This dataset includes random missing data values. We are going to use it to try imputation method. To save time running this dataset, subset the first 100 genes and save it as a new dataset.

#### a) How many missing values in the dataset? How many genes have missing values?

#if (!require("BiocManager", quietly = TRUE))  
# install.packages("BiocManager")  
#BiocManager::install("impute")  
  
library(impute)  
  
data(khanmiss)  
babykhan = head(khanmiss, 100)  
  
print(paste("There are", sum(is.na(babykhan)), "missing values in this dataset"))

## [1] "There are 45 missing values in this dataset"

print(paste("The below matrix shows the amount of missing values by gene:"))

## [1] "The below matrix shows the amount of missing values by gene:"

rowSums(is.na(babykhan))

## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5   
## 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40   
## 0 0 0 0 0 0 0 0 11 0 0 0 0 0 0 0 0 0 0 0   
## 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60   
## 6 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80   
## 5 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0   
## 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100   
## 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

#### b) Write your own code to define K-nearest neighbors for genes that you found in (a). Use the neighbors you defined to do imputation. Try different selection of K and summarize what you observe.

minikhan = babykhan[-1,-c(1,2)]  
 #Compute Euclidean distance  
 euc\_distance = function(a,b) {  
 return(sqrt(sum((a-b)^2, na.rm=TRUE)))  
 }  
   
 #Find neighbors  
 find\_neighbors = function(data, target\_row, k) {  
 distances = apply(data, 1, function(row) euc\_distance(target\_row, row))  
 neighbors = order(distances)[2:(k+1)]  
 return(neighbors)  
 }  
   
my\_knn = function(data, k) {  
 #K needs to be odd to avoid ties  
 k = ifelse(k %% 2 == 0, k+1, k)  
   
 for (i in 1:nrow(data)) {  
 missing\_cols = which(is.na(data[i, ]))   
 if (length(missing\_cols) > 0) {  
 neighbors = find\_neighbors(data, data[i, ], k)  
 for (col in missing\_cols) {  
 data[i, col] = mean(data[neighbors, col], na.rm=TRUE)  
 }  
 }  
 }  
   
 return(data)  
}  
  
imputed = my\_knn(minikhan, 5)  
  
print(paste("There are", sum(is.na(imputed)), "missing values in this dataset"))

## [1] "There are 43 missing values in this dataset"

print(paste("The below matrix shows the amount of missing values by gene:"))

## [1] "The below matrix shows the amount of missing values by gene:"

rowSums(is.na(imputed))

## 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0   
## 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41   
## 0 0 0 0 0 0 0 11 0 0 0 0 0 0 0 0 0 0 0 6   
## 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61   
## 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5   
## 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81   
## 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100   
## 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

I toubleshooted this for a while, and just can’t figure out what’s wrong with the above function. The dataframe comes out the same as it went in.

#### c) Use function impute.knn and compare with results using your own code. Are they the same?

#Rid of first two columns:  
minikhan = babykhan[-1,-c(1,2)]  
imputed = impute.knn(as.matrix(minikhan), k = 3)$data  
  
print(paste("There are", sum(is.na(imputed)), "missing values in this dataset"))

## [1] "There are 0 missing values in this dataset"

print(paste("The below matrix shows the amount of missing values by gene:"))

## [1] "The below matrix shows the amount of missing values by gene:"

rowSums(is.na(imputed))

## 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   
## 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100   
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

It works!