Project 3

Alejandro Roca

28 November, 2018

NMF Analysis

In this project we try to reproduce the results shown by Alexandrov et al. (2013). This is a NMF analysis of mutation count data in which they found four signatures, number we will follow for our analysis. Using the same data and check if we find these same four signatures.

Loading the Data

Starting by loading the data from `BRCA21.RData` we get a V matrix that we transpose to match the one seen in the slides from class. The transposed V matrix contains mutation counts of 96 types in 21 patients (21x96), as we can see with the following code:

```
load("BRCA21.RData")
V <- t(V)
dim(V) # [1] Number of rows   Number of columns
```

```
[1] 21 96
```

```
head(V, 2)[,1:8] # Show first 2 rows and 8 columns
```

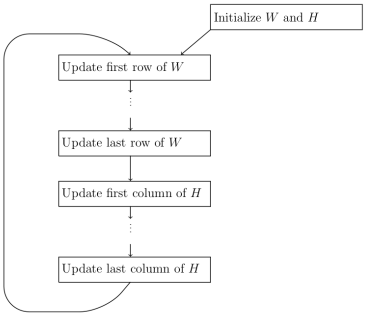
```
        A[C>A]A A[C>A]C A[C>A]G A[C>A]T C[C>A]A C[C>A]C C[C>A]G C[C>A]T
PD4194a      24      19       2      10      10       5       3       7
PD3851a      29      30       7      18      21      19       5      23
```

1. Applying the alternating NLS algorithm

We are facing a High-dimensional optimization problem and as mentioned before, we will use NMF to solve it. Being:

- M = Number of patients = 21
- N = Number of mutation types = 96
- K = Number of signatures = 4

The idea is to find a matrix $W_{(M\times K)}$ and a matrix $H_{(K\times N)}$ that can approximate V by $V \approx W \times H$ and can be used to study the $K = 4$ signatures. The algorithm is summarized in the next figure from the slides:



Let $\text{RSS}^{(t)} = \| V - W^{(t)}H^{(t)} \|$.
Stop when $\triangle\text{RSS}^{(t)} = \text{RSS}^{(t-1)} - \text{RSS}^{(t)}$ is small.

Update row m of W

For each row v in V:

- Calculate $A = 2HH'$
- Calculate $b = -2Hv$
- Minimize $f(w) = \frac{1}{2}w'Aw + b'w$

Update column n of H

For each column v in V:

- Calculate $A = 2W'W$
- Calculate $b = -2W'v$
- Minimize $f(h) = \frac{1}{2}h'Ah + b'h$

Solution to the NLS problem

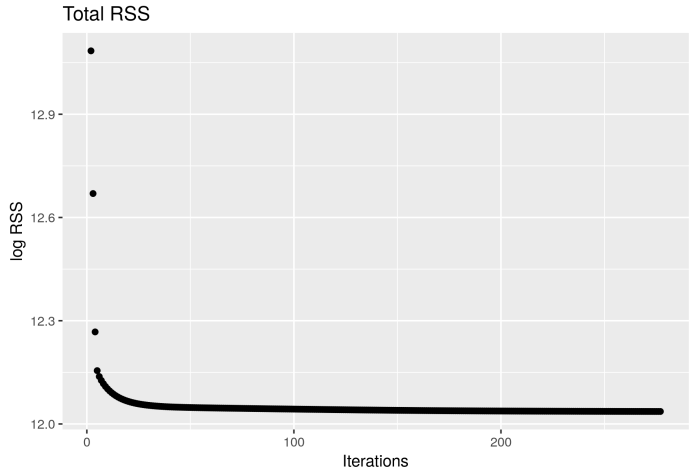
For each minimization we use the MM-algorithm which will approximate the correct values for W and H so that $||V - WH||^2$ is minimized.

Algorithm parameters

We set the maximum number of iterations on the whole process to 300 and the stopping criteria to $\Delta log(RSS^{(t)}) = log(RSS^{(t-1)}) - log(RSS^{(t)}) < s$, being $s = 10^{-4}$. The MM-algorithm performed for each row/column has a maximum number of iterations of 10000 and a tolerance of 10^{-15}

2. Studying the convergence of the algorithm

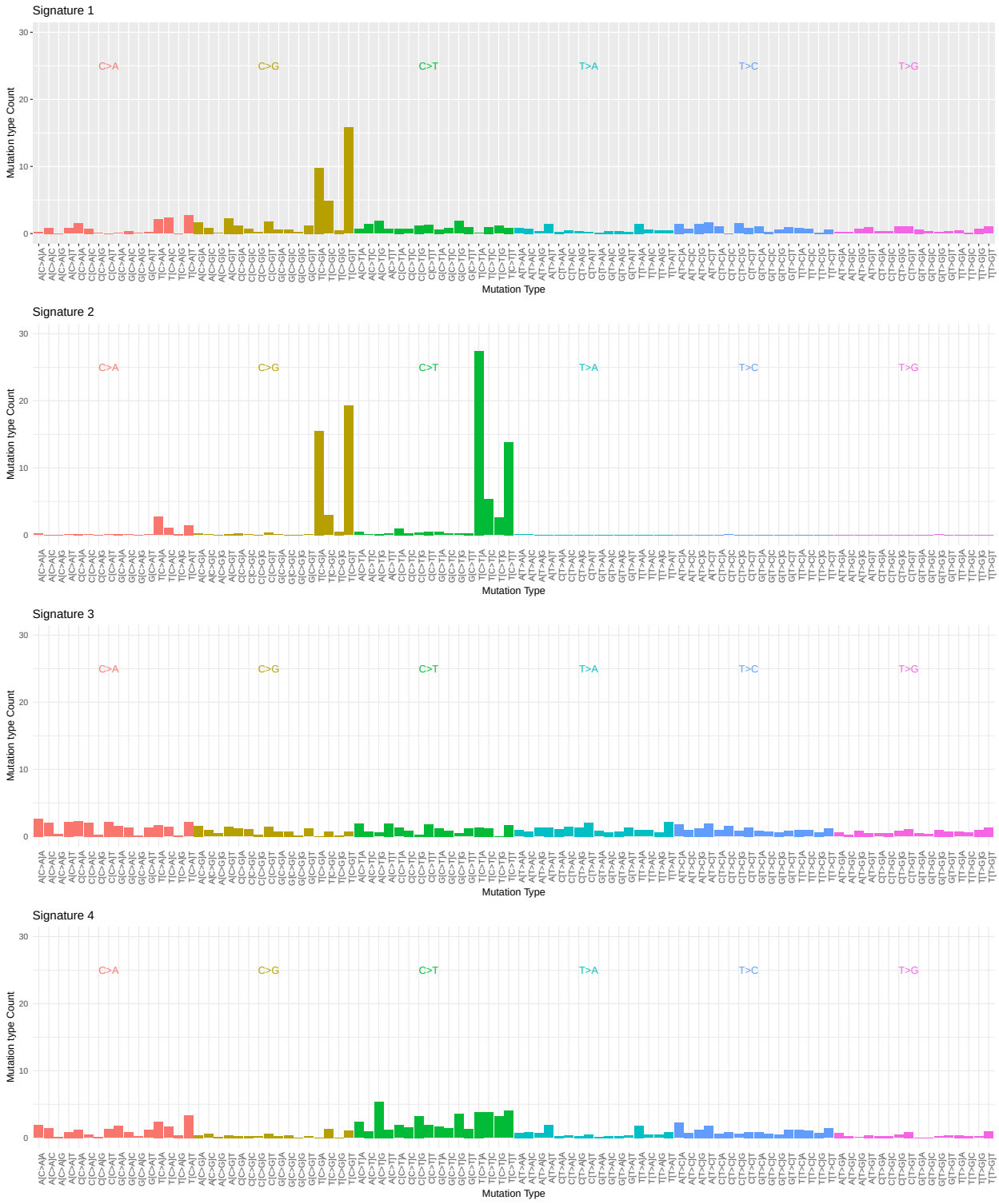
During the algorithm run, we calculate the $log(RSS^{(t)})$ and we can see the evolution in the following plot:



We can observe an asymptotic behaviour of the $log(RSS)$ through the different iterations. Moreover, by the number of RSS calculated (277) we can see that the algorithm converged, as the stopping criteria was met at this point.

3. Comparing the four signatures to Alexandrov et al. (2013)?

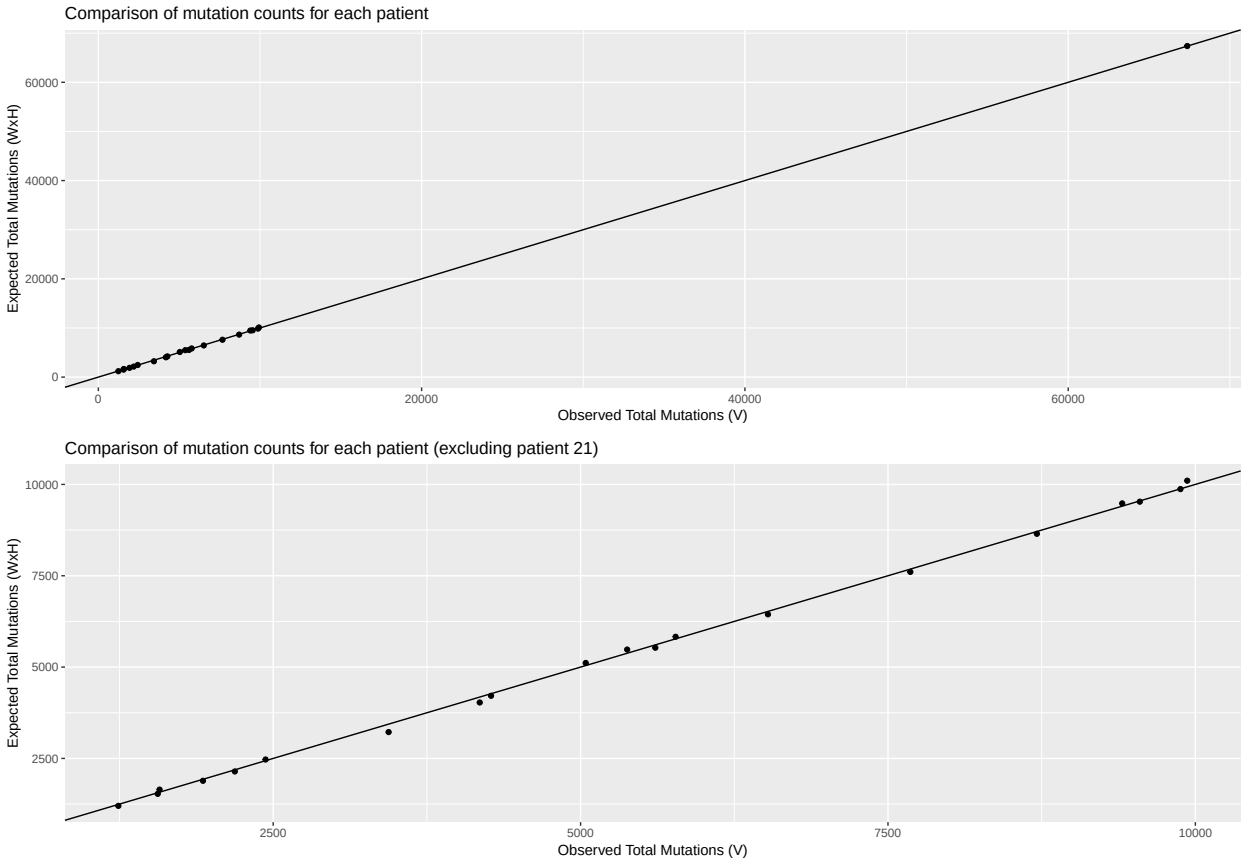
The signatures that I obtained can be visualized here:



Comparing these to the signatures shown in Figure 4A of Alexandrov et al. (2013), we see the same 4 signatures. Note: Signature 1 here corresponds to Signature 4 in Alexandrov et al. (2013) and viceversa.

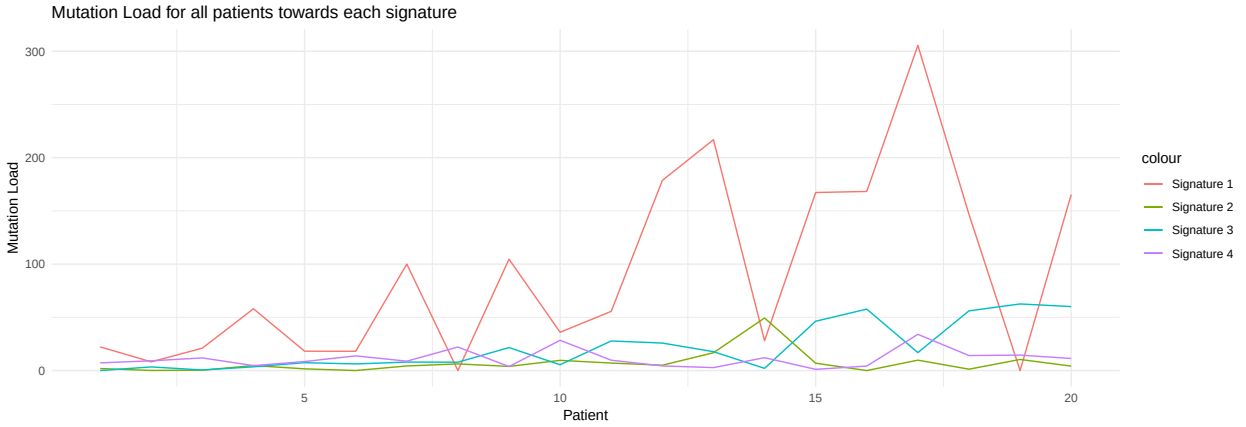
4. Total expected vs observed mutation count

The total expected ($\sum W \times H$) and the observed ($\sum V$) mutation count for each patient should be close to each other as can be seen in these plots



5. Studying mutation loading of each signature for each patient

Plotting the loadings for each signature on each patient, we can see that Signature 1 has clearly higher loading for most of the patients. As well, if we don't include patient 21 (with a very high load on signature 2), we get the following plot in which we can observe Signature 3 being higher than the other two in the majority of the patients.



Appendix

This section includes the R code used to complete the assignment.

Alternating NLS algorithm

```
n_patients <- nrow(V)
n_mutations <- ncol(V)
nIter <- 300    # maximum number of iterations
n_signatures <- 4
s <- 1e-5      # threshold used for convergence (stopping when lower)

# Initialize W and H
W <- matrix(runif(n_patients*n_signatures, 0, 1), nrow = n_patients, ncol = n_signatures)
H <- matrix(runif(n_signatures*n_mutations, 0, 1), nrow = n_signatures, ncol = n_mutations)

# dim(W) 21 x 4
# dim(H) 4 x 96

# Blank vectors where we will store RSS
W.RSS <- c(); H.RSS <- c(); MM.RSS <- c()

# Update W and H
for (iter in 1:nIter) {

    # Update rows in W; every patient
    for (i in 1:n_patients) {  # nrow(W)
        v <- V[i, ]     # we use rows of V

        # For A and b, opposite dimensions as in column update in H
        A = 2 * H %*% t(H)
        b = as.vector(-2 * H %*% v)

        MMres <- MajorizeMinimizeNQP(A = A, b = b, init=W[i,], maxIter = 10000, tol = 1e-16)

        # Replace old by updated row
        W[i, ] <- MMres$x
    }

    # Calculate RSS for each update in W
    W.RSS[iter] <- sum((V - W %*% H) ^ 2)

    # Update columns in H; every mutation
    for (i in 1:n_mutations) {      # ncol(H)
        v <- V[ ,i]     # we use columns of V

        A = 2 * t(W) %*% W
        b = as.vector(-2 * t(W) %*% v)

        MMres <- MajorizeMinimizeNQP(A = A, b = b, init=H[,i], maxIter = 10000, tol = 1e-16)

        # Replace old by updated column
```

```
        H[ ,i] <- MMres$x
    }

    # Calculate RSS for each update in H
    H.RSS[iter] <- sum((V - W %*% H) ^ 2)

    # Calculate RSS for each iteration of the algorithm
    MM.RSS[iter] <- sum((V - W %*% H) ^ 2)

    # Keep track of the convergence condition evolution
    if (iter %% 10 == 0) {cat("Iteration ", iter, ", log RSS decrease: ",
                              log(MM.RSS[iter - 1]) - log(MM.RSS[iter]), "\n",
                sep = '')}

    # Convergence conditioned on log(RSS) increase
    if ( (iter > 1) && ((log(MM.RSS[iter - 1]) - log(MM.RSS[iter])) < s) ) {break}

}
```

MajorizeMinimizeNQP function

```
MajorizeMinimizeNQP <- function(A,b,init,maxIter=1000,tol=0.001){
  ## Objective function
  f = function(x) as.numeric(1/2*crossprod(x, A)%*%x+crossprod(b, x))
  ## Initial values of xList, objFctList and time
  K = ncol(A)
  if (missing(init)) {
    x = matrix(runif(K), nrow=K, byrow=T)
  } else {
    x = init
  }
  xList = x
  objfctList = c(f(x))
  time = c(0)
  start = Sys.time()
  ##-----------------------------------------
  ## Iterative procedure for solving the NQP
  ##-----------------------------------------
  ## Minimization
  for (i in 1:maxIter){
    x = -b*x/(A %*% x)
    objfct = f(x)
    if (abs(objfct - objfctList[i]) < tol) break
    xList = cbind(xList, x)
    objfctList = c(objfctList, objfct)
    time = c(time, Sys.time() - start)
  }
  return (list(x=x, objfct=objfct, xList = xList, objfctList = objfctList, time=time))
}
```

Ploting the different signatures

```
# Function used to assign the right color to the text
# inside the plot (so it matches the colors on the bars)
gg_color_hue <- function(n) {
  hues = seq(15, 375, length = n + 1)
  hcl(h = hues, l = 65, c = 100)[1:n]
}

H_plot <- data.frame(matrix(0, nrow = ncol(H), ncol = nrow(H)+2))
colnames(H_plot) <- c("Signature1", "Signature2", "Signature3", "Signature4",
                      "MutType", "MutGroup")
for (i in 1:nrow(H)) {     # Transform to percentages
    H_plot[,i] <- H[i,]/sum(H[i,]) * 100
}

# Associate mutation with color by group
colors <- c(rep("C>A", 16), rep("C>G", 16), rep("C>T", 16),
            rep("T>A", 16), rep("T>C", 16), rep("T>G", 16))
H_plot[,6] <- colors

# Order by mutation type
H_plot[,5] <- reorder(factor(colnames(V)), seq(1, 96))
# H_plot <- H_plot %>% arrange(MutType)

labels <- c("C>A", "C>G", "C>T", "T>A", "T>C", "T>G")
p1 <- ggplot(H_plot) +
    geom_col(mapping = aes (x = MutType, y = Signature1, fill = MutGroup)) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    ylim(c(0, 30)) +
    xlab("Mutation Type") + ylab("Mutation type Count") +
    ggtitle("Signature 1") +
    annotate("text", x = seq(from = 8, to = 88, by = 16), y = 25, label = labels,
             colour = gg_color_hue(6)) +
    guides(fill=FALSE)

p2 <- ggplot(H_plot) +
    geom_col(mapping = aes (x = MutType, y = Signature2, fill = MutGroup)) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    ylim(c(0, 30)) +
    xlab("Mutation Type") + ylab("Mutation type Count") +
    ggtitle("Signature 2") +
    annotate("text", x = seq(from = 8, to = 88, by = 16), y = 25, label = labels,
             colour = gg_color_hue(6)) +
    guides(fill=FALSE)

p3 <- ggplot(H_plot) +
    geom_col(mapping = aes (x = MutType, y = Signature3, fill = MutGroup)) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    ylim(c(0, 30)) +
    xlab("Mutation Type") + ylab("Mutation type Count") +
```

```
    ggtitle("Signature 3") +
    annotate("text", x = seq(from = 8, to = 88, by = 16), y = 25, label = labels,
             colour = gg_color_hue(6)) +
    guides(fill=FALSE)

p4 <- ggplot(H_plot) +
    geom_col(mapping = aes (x = MutType, y = Signature4, fill = MutGroup)) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    ylim(c(0, 30)) +
    xlab("Mutation Type") + ylab("Mutation type Count") +
    ggtitle("Signature 4") +
    annotate("text", x = seq(from = 8, to = 88, by = 16), y = 25, label = labels,
             colour = gg_color_hue(6)) +
    guides(fill=FALSE)

p1
p2
p3
p4
```