Cancer methylome analysis. Protocol group 2.

Krank K., Kuchina M., Mendoza-Alcala A., Spatuzzi M.

Introduction

DNA methylation is a well studied epigenetic mechanism which influences many processes in a cell including cell proliferation and differentiation but also development of common diseases such as cancer. Studies show that methylomes of cancer cells have a different methylation pattern, especially in promoter regions, which indicates that gene expression is inhibited (Baylin 2005). Out team worked with the DNA methylation data collected from patients with acute myeloid leukemia (AML) and healthy individuals. We chose to work with promoters not only because of reasons named above but also because it would provide us with more reliable information while interpreting the results. The main goal of the project was to identify differentially methylated regions (DMRs) - DNA sequences with different methylation between the two groups. This is the matter of scientific and clinical interest because reliable DMRs might help to discern diseased and healthy patients. In the following protocol we provide a comprehensive workflow of our project.

Data preprocessing

The data-set we had been working with contained a lot of irrelevant information. It was also partly incomplete - with many not available or missing values. The coverage depth variated from unreliably low to suspiciously high (possibly resulted by PCR duplicates). A transfromation of heteroscedastic beta-values to homoscedastic M-values would be benefitial for direct applying of most statistical tests (Du et al. 2010). Therefore, the data must be cleaned and normalized to make further analysis possible. The goal is to find the balance between proper data cleaning and loosing too much of reliable information. We decided that 15% of lost information would be acceptable.

Data clean-up

Goals:

- remove regions with low or too high coverage
- remove regions with high amount of missing values
- replace `NA`s with the mean value of the cohort

Work process:

1. Data importation and visualization

Data-set has been imported and separated into specific subgroups. We defined `promoters_only` variable and split it into `beta` and `coverage` datasets for further operations.

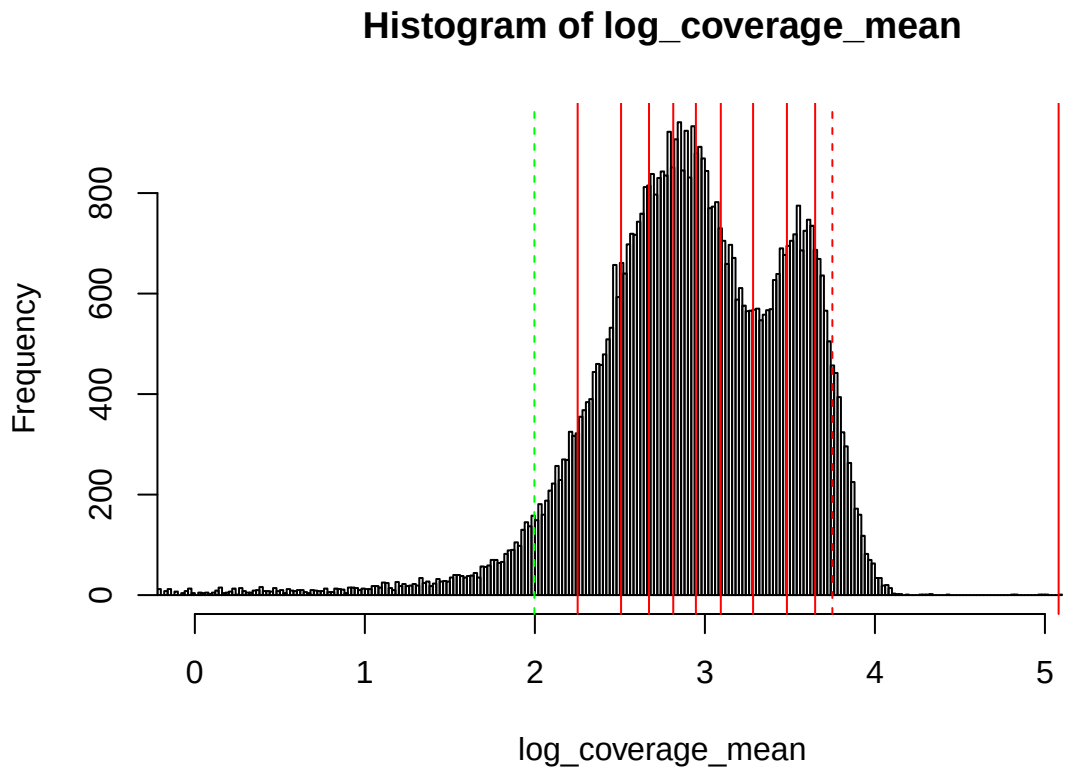
```
data <- readRDS(file = "AML_gran_list.RDS")
annotation <- read.csv(file="sample_annotation.csv")
promoters <- data$promoters;
promoters_only <- promoters[,-c(1:10)]
beta <- promoters_only[,c(1:18)]
coverage<- promoters_only[,c(19:36)]
PatientInfo <- annotation[,c(3,4,11,29)]
```

We shortened the names of the patient-info providers so it would be more convenient to work with them. We also changed the sample names to simplify them.

```
PatientInfo[,4] <- as.character(PatientInfo[,4])
PatientInfo[which(PatientInfo[,4] ==
                    "Prof: Lucia Altucci  (SECONDA UNIVERSITA' di NAPOLI- IT)"),4]  <-
  "Universita di Napoli"
PatientInfo[which(PatientInfo[,4] ==
                    "Prof.dr. E. Vellenga, University Medical Centre Groningen - Department of Hematolo
  "University Groningen"
PatientInfo[,4] <- as.factor(PatientInfo[,4])
rownames(PatientInfo) <- c(
  paste("AML", 1:9, sep=""),
  paste("CON", 1:9, sep="")
)
```

The general distribution of the coverage was visualized in the following plot:

```
par(mfrow=c(1,1))
coverage_tot_mean <- rowMeans(coverage)
log_coverage_mean <- log10(coverage_tot_mean)
hist(log_coverage_mean, breaks = 250, xlim = c(0,5.5))
abline(v=quantile(log_coverage_mean, probs = c(seq(0.1, 1.0, by= 0.1))), col="red")
abline(v=quantile(log_coverage_mean, probs =0.05), col="green", lty=2)
abline(v=quantile(log_coverage_mean, probs =0.95), col="red", lty=2)
```




```
quantile(coverage_tot_mean, probs=c(0.05, 0.95))
```

```
##         5%        95%
##   99.33333 5620.00000
```

The green line identifies the 5% quantile while the red one is the 95% quantile. We can see that the distribution of our data is bimodal and does not follow the normal Gaussian distribution. We supposed that it might be a sign of bias-presence as a result of batch effects which will be discussed later.

2. Removing sex chromosomes.

Several studies show that there are sex differences in methylation status. For example the second X-chromosome in females and the heterochromatic region of Y-chromosome are strongly methylated which does not correlate with any disease but is a normal occurence (Bernardino et al. 2000). Therefore, correction for sex is necessary and X- and Y-chromosomes must be removed to avoid gender bias.

```
cover_nox <- coverage[-which(promoters$Chromosome == "chrX"), ]
cover_noxy <- cover_nox[-which(promoters$Chromosome == "chrY"), ]
rm(coverage, cover_nox)
beta_nox <- beta[-which(promoters$Chromosome == "chrX"), ]
beta_noxy <- beta_nox[-which(promoters$Chromosome == "chrY"), ]
rm(beta, beta_nox)
print( (nrow(beta_noxy)*100) / nrow(promoters) )
```

```
## [1] 95.93893
```

As shown above, the X- and Y-chromosomes represented about 4% of the information.

3. Coverage thresholds setting.

3.1 The lower threshold

Having the sex chromosomes removed we took the recommended value for the minimum coverage of 30x to set our lower threshold. (https://www.encodeproject.org/wgbs/)

```
lower_threshold <- 30
cover_low_trimmed <- as.matrix(cover_noxy)
cover_low_trimmed_list <- which(cover_low_trimmed < lower_threshold)
beta_low_trimmed <- as.matrix(beta_noxy)
beta_low_trimmed[cover_low_trimmed_list] <- NA
beta_low_NA_rm <- beta_low_trimmed[-which(rowSums(is.na(beta_low_trimmed)) > 2), ]
which(rowSums(is.na(beta_low_NA_rm)) > 2) ## RESULT = 0
```

```
## named integer(0)
```

```
print( (nrow(beta_low_NA_rm)*100) / nrow(beta_noxy))
```

```
## [1] 94.87028
```

```
print( (nrow(beta_low_NA_rm)*100) / nrow(promoters_only))
```

```
## [1] 91.01752
```

3.2 The upper threshold.

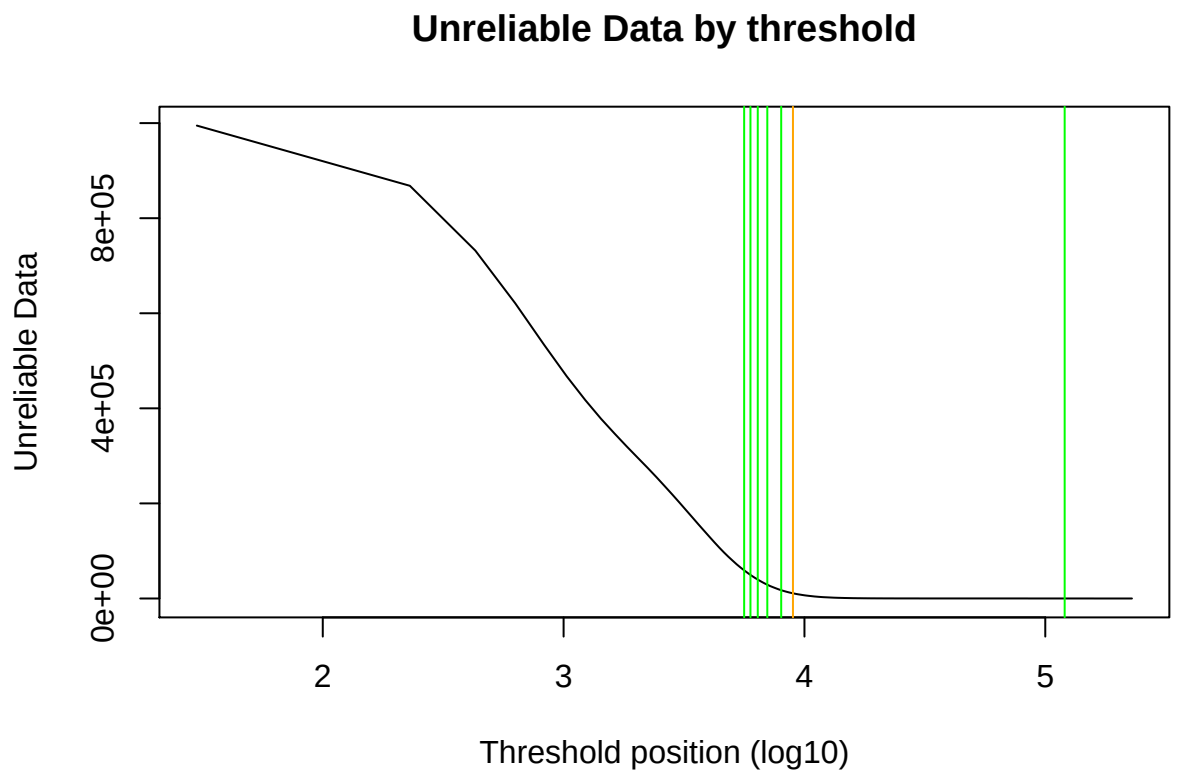
First we decided to create a plot which would show the amount of values we would lose depending on the threshold settings. We used the `unlist` function in order to freely inspect the whole dataframe. We inspected the results for three quantiles: 75%, 95% and 100%.

```
coverage_histo <- unlist(cover_noxy)
log_cover_histo <- unlist(log10(cover_noxy))
quantile(coverage_histo, probs=c(.75, .95, 1))
```

```
##     75%     95%    100%
##    2396    5888 229153
```

We supposed that the upper threshold should have been located between 95% and 100% quantiles. Vectors `NAs_up` and `thresholds_up` were created. The following loop goes through the data-set in 200-steps and looks for the values which coverage is higher than the threshold.

```
NAs_up <- c()
thresholds_up <- c()
for(i in seq(30, max(coverage_histo),200)){
  threshold <- i
  thresholds_up <- append(thresholds_up, threshold)
  Unreliable <- coverage_histo[which(coverage_histo > threshold)]
  N_Unreliable <- length(Unreliable)
  NAs_up <- append(NAs_up, N_Unreliable)
}
par(mfrow=c(1,1))
plot(x=log10(thresholds_up),
     y=NAs_up, type= "l",
     main = "Unreliable Data by threshold",
     xlab= "Threshold position (log10)", ylab = "Unreliable Data")
abline(v=quantile(log_coverage_mean, probs = c(seq(0.95, 1.0, by= 0.01))), col="green")
abline(v=quantile(log_coverage_mean, probs = 0.995), col="orange")
abline(v=quantile(coverage_tot_mean, probs = c(seq(0, 1.0, by= 0.1))), col="green")
```




```
quantile(coverage_histo, probs=.95)
```

```
##  95%
## 5888
```

The plot shows how much information would be transformed to `NA`s depending on the position of the threshold.

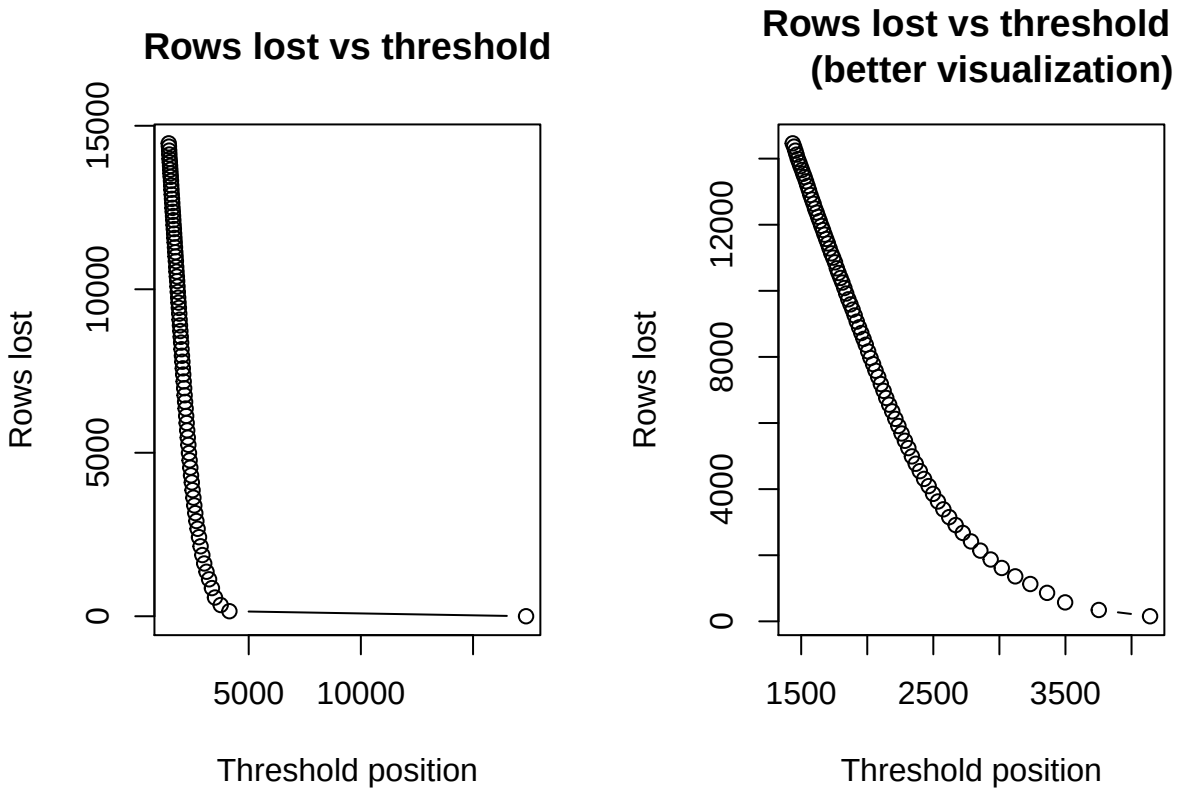
On this basis we decided that 95% quantile with the value 5888 could be our upper threshold. Soon enough we realized that it would be a wrong decision because we did not consider the fact that even the genes without NAs have to be removed if the same genes in the other cohort contain unreliable information. With every row we remove we also lose information in form of reliable values but at the same time values that have too low or loo high coverage are unreliable and therefore don't count as lost information.

The following code demonstrates another approach to establish the upper threshold. We created the empty vectors `rows_lost` and a zero vector `rows_lost_tot` containing a zero to make further addition operation (accumulation of values) possible. `AML_position_above_limit <- apply(df[,c(1:9)], 1, function(x){ length(which(x > threshold)) })` would give us the amount of promoters from AML cohort that would have coverage above the tested threshold. `CON_position_above_limit <- apply(df[,c(10:18)], 1, function(x){ length(which(x > threshold)) })` executes the same command for the control group. All values above the tested threshold should be turned into NAs. To make the dataframe smaller (and the analysis faster) the loop creates a copy of the original dataframe and eliminates the rows that fulfill the criterium. The vector `rows_lost` helps to keep track on the amount of removed rows.

```
df <- as.matrix(cover_noxy)
rows_lost_tot <- c(0)
for(threshold in quantile(df, probs=seq(1, 0.84, by=-0.002))){
  rows_lost <- c()

  AML_position_above_limit <- apply(df[,c(1:9)], 1, function(x){
    length(which(x > threshold))  })
  CON_position_above_limit <- apply(df[,c(10:18)], 1, function(x){
    length(which(x > threshold))  })

  rows_lost <- which(AML_position_above_limit > 2 | CON_position_above_limit > 2)
  rows_lost_tot <- append( rows_lost_tot,
                           ( rows_lost_tot[length(rows_lost_tot)] + length(rows_lost) )
                           )

  if(length(rows_lost) > 0){
    df <- df[-rows_lost,]
  }
}
rows_lost_tot <- rows_lost_tot[-1]
thresholds <- quantile(df, probs=seq(1, 0.84, by=-0.002))
```

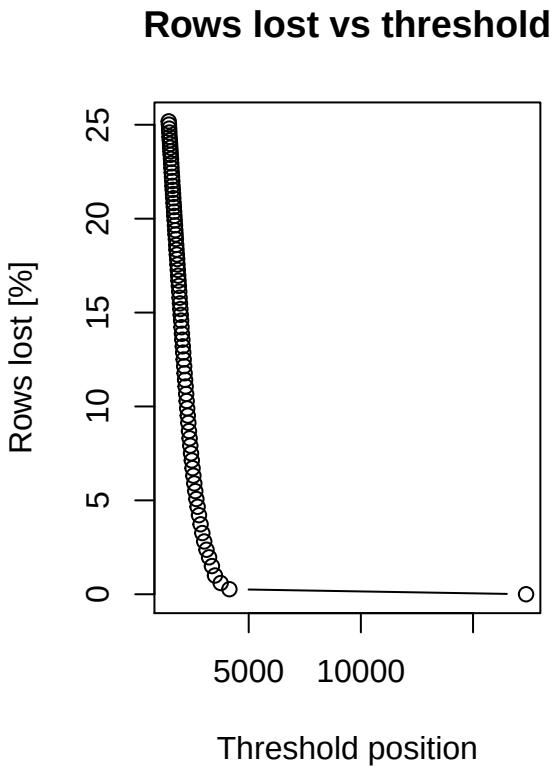
First we plotted the lost rows depending on the threshold. Each row with more than two NAs has been removed.

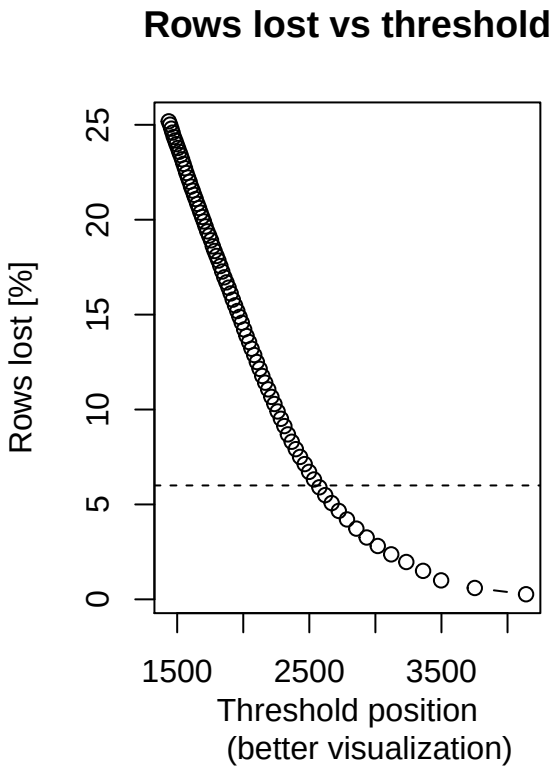
```
par(mfrow=c(1,2))
plot(x=thresholds, y=rows_lost_tot, type="b",
     main="Rows lost vs threshold",
     xlab="Threshold position", ylab="Rows lost")
plot(x=thresholds[2:length(thresholds)],
    y=rows_lost_tot[2:length(rows_lost_tot)],
     type="b",
     main="Rows lost vs threshold
    (better visualization)",
    xlab="Threshold position", ylab="Rows lost")
```



Then we plotted the percentage of information we would lose. The plots on the left provide a better visualization by hiding the last value.

```
rows_lost_per <- (rows_lost_tot*100) / nrow(cover_noxy)
par(mfrow=c(1,2))
plot(x=thresholds, y=rows_lost_per, type="b",
     main="Rows lost vs threshold",
     xlab="Threshold position", ylab="Rows lost [%]")
plot(x=thresholds[2:length(thresholds)],
     y=rows_lost_per[2:length(rows_lost_tot)], type="b",
     main="Rows lost vs threshold",
     xlab="Threshold position
     (better visualization)", ylab="Rows lost [%]")
abline(h=6, lty=2)
```





As already mentioned, we decided not to lose more than 15% of information. Up until the removal of coverage values below 30, we've lost 10% of the information. Accordingly, the upper threshold should be chosen so that no more than 15% of the information (rows) are lost.

3.3 Applying coverage thresholds to the data-set

After establishing the upper and lower threshold positions we applied them to our data-set. All of the beta-values with coverage higher or lower than the thresholds were converted into `NA`.

```
lower_threshold <- 30
upper_threshold <- quantile(coverage_histo, probs=0.995)
beta_trimmed <- as.matrix(beta_noxy)
cover_trimmed <- as.matrix(cover_noxy)
future_NAs <- which(cover_trimmed < lower_threshold | cover_trimmed > upper_threshold)
beta_trimmed[future_NAs] <- NA
```

Genes with more than two `NA`s were first removed from both cancer and healthy cohort. We tolerated this amount of missing values because this way the data-set would still be statistically significant (Dong and Peng 2013).

```
beta_AML <- beta_trimmed[,c(1:9)]
beta_con <- beta_trimmed[,c(10:18)]
NAs_AML <- rowSums(is.na(beta_AML))
NAs_con <- rowSums(is.na(beta_con))
AML <- cbind(beta_AML, NAs_AML)
con <- cbind(beta_con, NAs_con)
beta_qc <- cbind(AML, con)
beta_partly_cleaned <- beta_qc[-which(beta_qc[, 10] > 2 | beta_qc[, 20] > 2), ]
```

And finally the NAs were replaced with the mean value of the cohort (healthy or diseased) so it wouldn't affect further statistical tests.

```
for (i in 1:nrow(beta_partly_cleaned)) {
  n <- beta_partly_cleaned[i,10]

  if (n==1 | n==2) {
    for (j in 1:9){
      if (is.na(beta_partly_cleaned[i,j])){
        beta_partly_cleaned[i,j] <- mean(beta_partly_cleaned[i,1:9], na.rm = T)
      }
    }
  }

  n <- beta_partly_cleaned[i,20]

  if (n==1 | n==2) {
    for (j in 11:19){
      if (is.na(beta_partly_cleaned[i,j])){
        beta_partly_cleaned[i,j] <- mean(beta_partly_cleaned[i,11:19], na.rm = T)
      }
    }
  }
}
beta_both_clean <- beta_partly_cleaned[,-c(10,20)]
```

Data normalization

Goal:

- transfer beta-values to more statistically valid M-values

Work process:

The M-value is calculated as the log2 ratio of the intensities of methylated probe versus unmethylated probe so here we normalized our data by transforming the beta-values according to the following equation:

$$M = log_2(\frac{\beta}{1-\beta})$$

```
for (j in 1:18) {
  for (i in 1:nrow(beta_both_clean)) {
    beta_both_clean[[i,j]] <- log2(beta_both_clean[i,j] / (1-beta_both_clean[i,j]) )

  }
}
promoters_normalized <- beta_both_clean
rm(beta_both_clean)
#Export table
write.table(promoters_normalized, file="promoters_normalized.csv",sep=",",row.names=T)
promoters_normalized <- read.csv(file="promoters_normalized.csv")
```

Statistical analysis

Dimensionality reduction and PCA

High dimensional data provides many problems by analysing: from high demands on computing resources to impracticability of understanding the underlying structure and problems with visualization. There are many ways of dimensional reduction such as principal component analysis (PCA), heatmaps, t-SNE plots, multidimensional scaling etc. We focused on the PCA, one of the main methods to reduce data dimensionality while retaining most of the variaton in the data set. It is based on a process of identifying directions (or principal components) along which the variation in the data is maximal. (Ringnér 2008). By using a few components instead of thousands of variables we can plot samples and determine whether they can be grouped. We applied PCA to check for batch effects and to look for any differences in methylation between cancer (AML) and control (con) group. To use the `prcomp` function in R we must first turn our data into a matrix.

```
promoters_matrix <- as.matrix(promoters_normalized)
class(promoters_matrix)
```

```
## [1] "matrix"
```

During the data normalization beta values equal 0 or 1 have been turned into `-Inf` and `Inf`. The PCA can not work with such M values so we assigned extremeley low (for `-Inf`) and high (for `Inf`) M values so we could run PCA without any problems.

```
promoters_matrix_noInf <- promoters_matrix
promoters_matrix_noInf[which(promoters_matrix_noInf == Inf)] <- log2(0.999999/(1-0.999999))
promoters_matrix_noInf[which(promoters_matrix_noInf == -Inf)] <- -log2(0.999999/(1-0.999999))
```

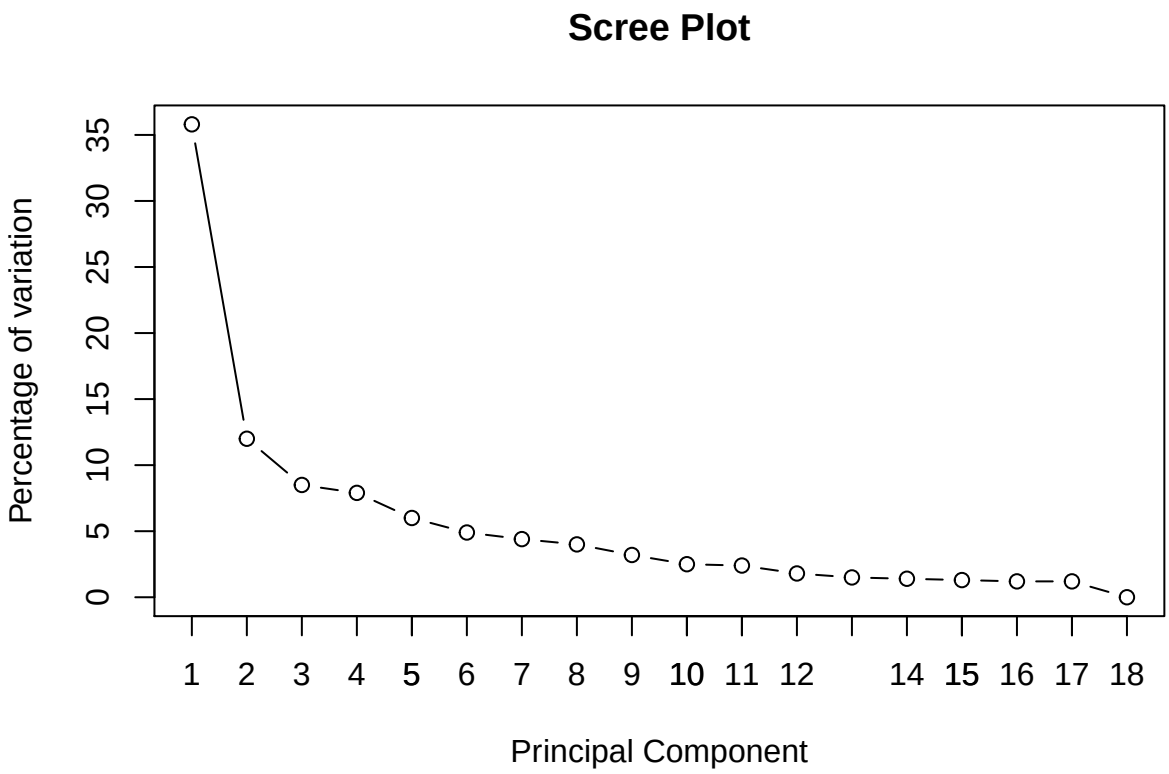
The column names were turned into intuitive names and the PCA was run.

```
colnames(promoters_matrix_noInf) <- c(
  paste("AML", 1:9, sep=""),
  paste("CON", 1:9, sep=""))

promoters_pca <- prcomp(t(promoters_matrix_noInf), scale = TRUE)
```

The object `promoters_pca` contains three types of information: x, sdev and rotation. Here, `x` contains the principal components, `sdev` contains the information about how much standard deviation each component accounts for and `rotation` shows how much influence (loading scores) each gene has on each PC. Positive influences push points on a plot towards positive values and negative influences push the data points toward negative values.

Quadratic value of `sdev` can be used to calculate how much variance each component accounts for. For a better comparison of the PCs we transformed that value into percents.

```
pca.var <- promoters_pca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
plot(pca.var.per,
     x = c(1 :length(pca.var.per)),
     type = "b", main = "Scree Plot",
     xlab= "Principal Component", ylab= "Percentage of variation" )
axis(at=c(1:18), side=1, tick=T)
axis(side=1, at=c(11,15,17))
```



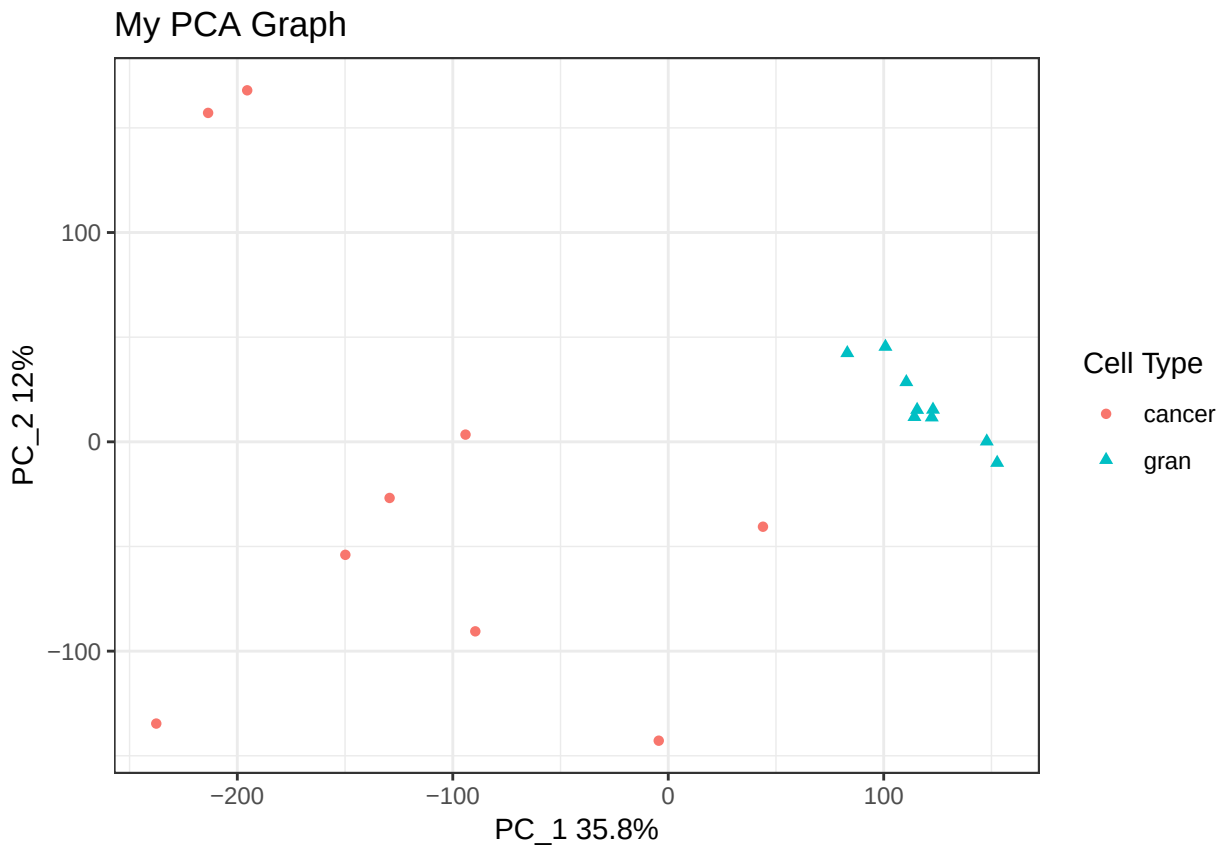
The first three PCs explain most of the variation within the data. To plot the samples along the PCs we programmed several functions. We used ggplots which let us easily combine or add different types of visualization components (or layers) together.

```
library(ggplot2)
ggplot.pca.cellTypeGroup <- function(i, j, npar=TRUE,print=TRUE){

  pca.data <- data.frame(Sample=rownames(promoters_pca$x), X=promoters_pca$x[,i],
                         Y=promoters_pca$x[,j], PatientInfo$cellTypeGroup )

  ggplot(data=pca.data, aes(x=X, y=Y, label="", group = PatientInfo.cellTypeGroup)) +
    geom_text() +
    xlab(paste("PC_", i," ", pca.var.per[i], "%", sep ="")) +
    ylab(paste("PC_", j," ", pca.var.per[j], "%", sep ="")) +
    theme_bw() +
    ggtitle("My PCA Graph") +
    geom_point(aes(shape=PatientInfo.cellTypeGroup, color=PatientInfo.cellTypeGroup)) +
    labs(color='Cell Type') +
    labs(shape='Cell Type')
}
ggplot.pca.BIOMATERIAL_PROVIDER <- function(i, j, npar=TRUE,print=TRUE){

  pca.data <- data.frame(Sample=rownames(promoters_pca$x), X=promoters_pca$x[,i],
                         Y=promoters_pca$x[,j], PatientInfo$BIOMATERIAL_PROVIDER )

  ggplot(data=pca.data, aes(x=X, y=Y, label="", group = PatientInfo.BIOMATERIAL_PROVIDER)) +
    geom_text() +
```

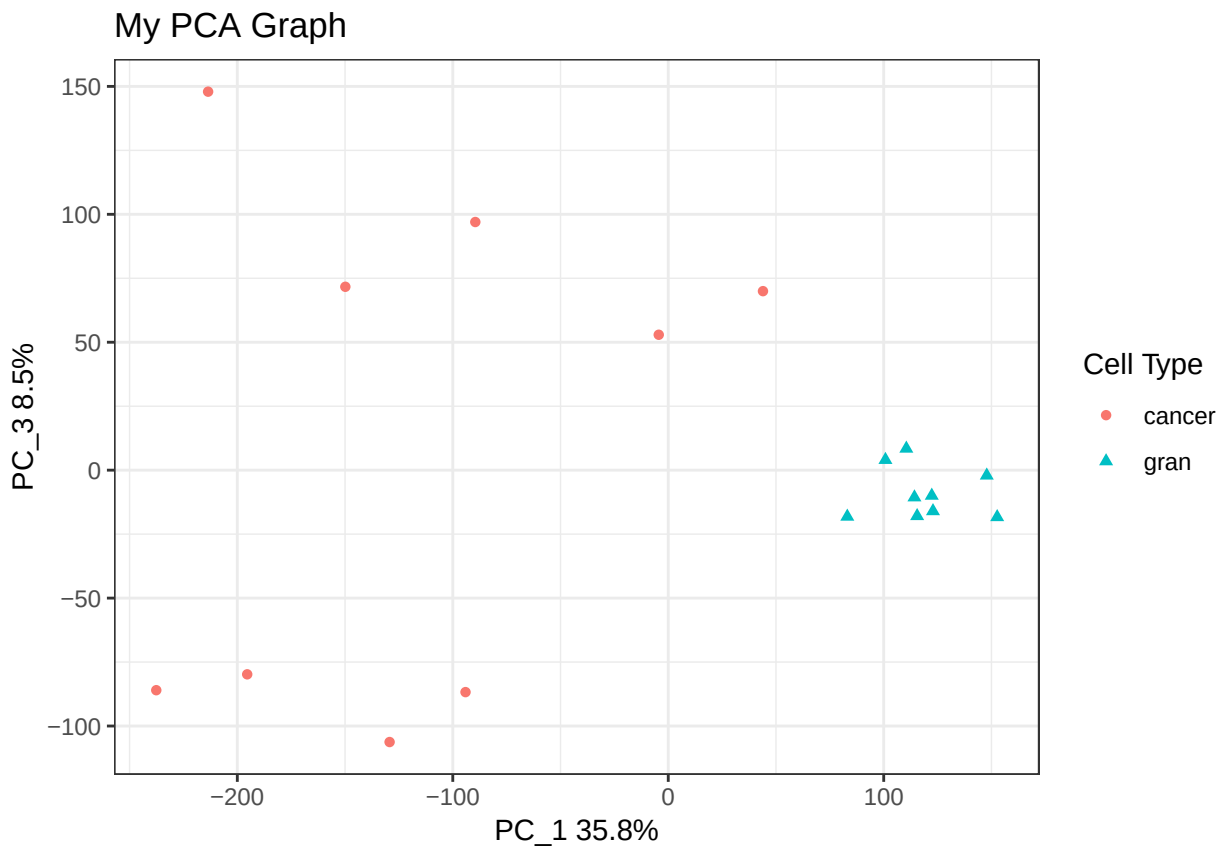
```
    xlab(paste("PC_", i," ", pca.var.per[i], "%", sep =""))  +
    ylab(paste("PC_", j," ", pca.var.per[j], "%", sep =""))  +
    theme_bw()  +
    ggtitle("My PCA Graph") +
    geom_point(aes(shape=PatientInfo.BIOMATERIAL_PROVIDER,
                   color=PatientInfo.BIOMATERIAL_PROVIDER)) +
    labs(color='Biomat. Provider') +
    labs(shape='Biomat. Provider')

}
```

At first we tried to visualize the cellType as a biological reason for differences in methylation.

```
ggplot.pca.cellTypeGroup(1,2)
```

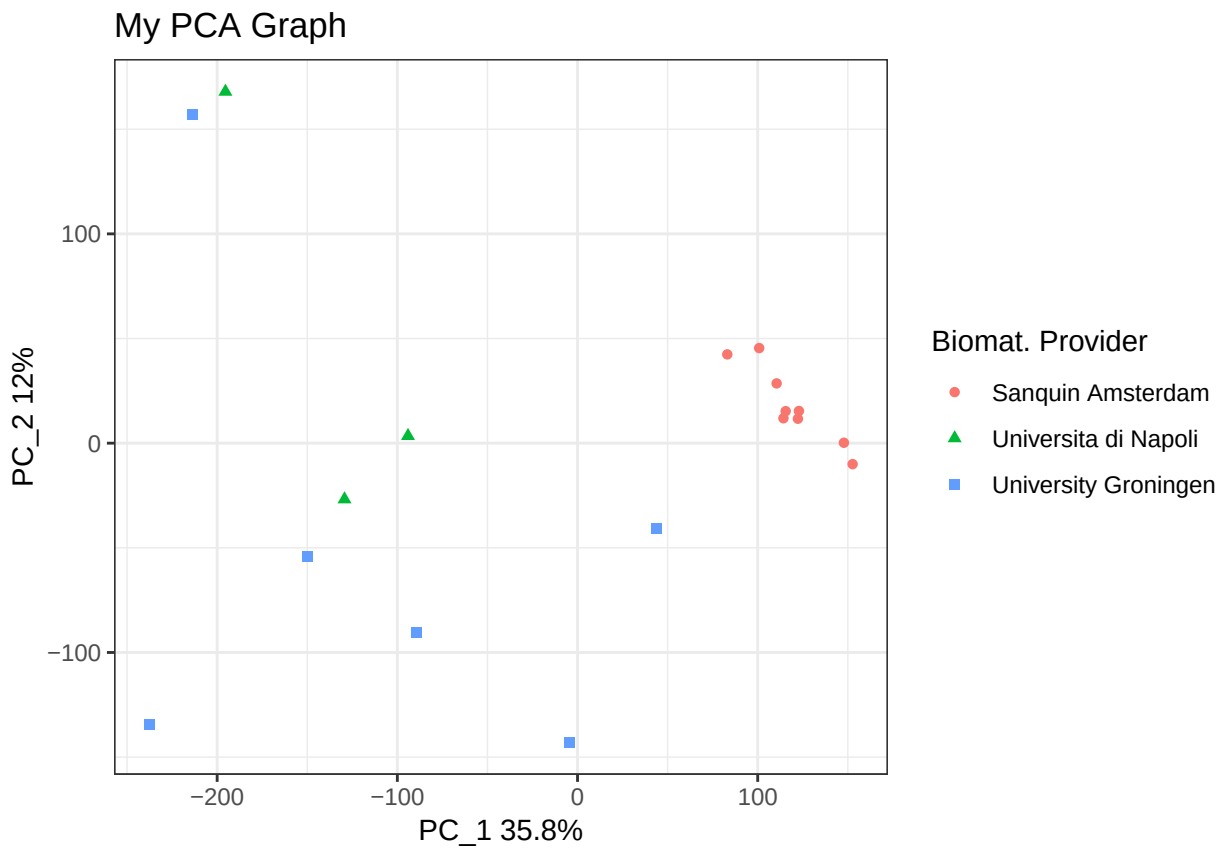



```
ggplot.pca.cellTypeGroup(1,3)
```

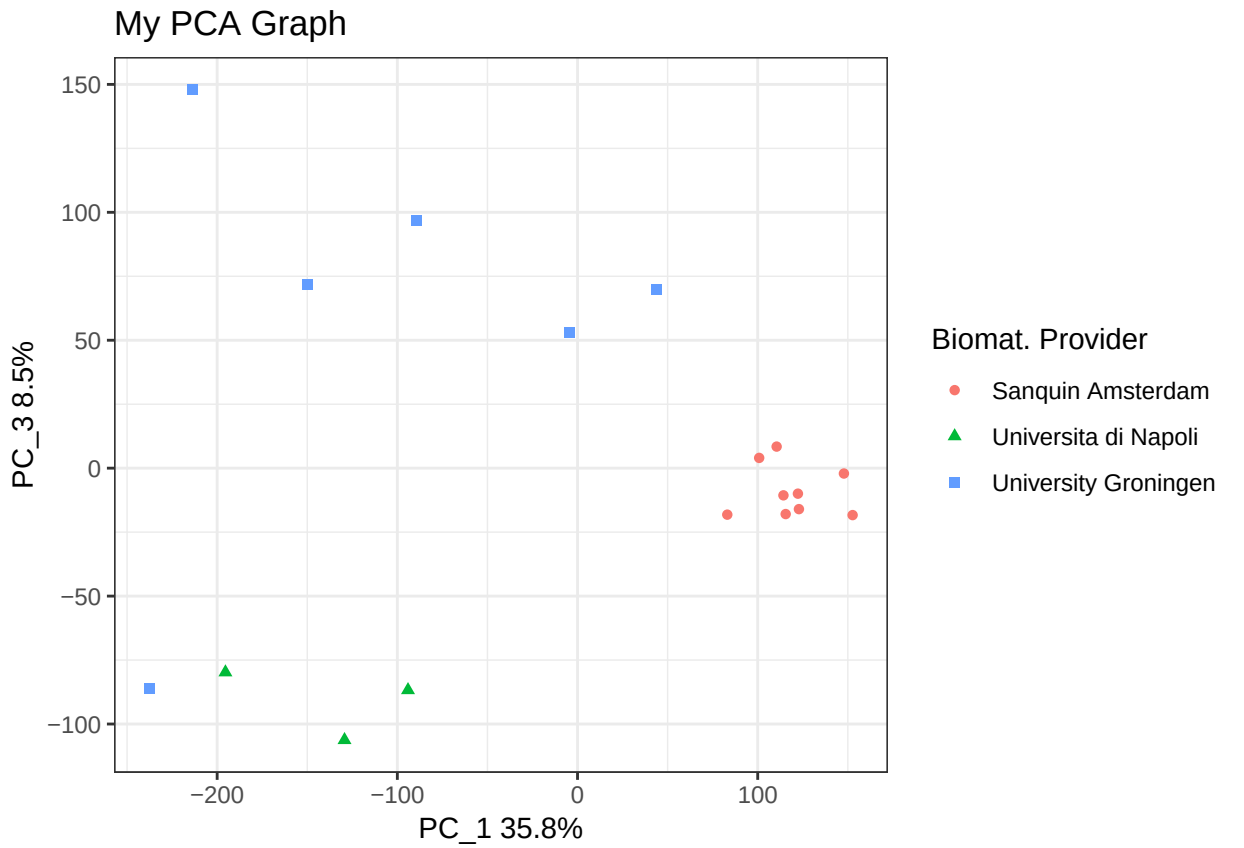


Other possible reasons for variation in methylation data are technical reasons also known as batch effects. One of the batch effects we tried to visualize was the Biomaterial Provider.

```
ggplot.pca.BIOMATERIAL_PROVIDER(1,2)
```




```
ggplot.pca.BIOMATERIAL_PROVIDER(1,3)
```



PC3 strongly separates AML patients at two extremes while still preserving the control group. The two groups are characterised by two different biomaterial providers, except for one patient. We may concern that the biomaterial provider influenced the result of the sequencing. These conclusions are more qualitative than quantitative. Plotting every possible source of variation takes too much time and does not give a lot of usefull information. We used statistical test to get more reliable and quantifiable results.

Kruskal-Wallis test

This test helps to determine whether the samples originate from the same distribution or not. It is a non-parametric method and the Kruskal Wallis test does not assume a normal distribution of the residuals. Assuming that all groups are identically shaped and have a scaled distribution, the null hypothesis is that the medians of all groups are equal, and the alternative hypothesis is that at least one population median of one group is different from the population median of at least one other group. We created a dataframe with possible sources of variation and the PCs to inspect them closely.

```
pca_promoters <- promoters_pca$x
sample_annotation <- read.csv(file="sample_annotation.csv")
PatientInfo_kruskal <- sample_annotation[,c(3,28,29,33)]
kruskal_df <-cbind(PatientInfo_kruskal,pca_promoters)
colnames(kruskal_df) <- c(1:22)
```

We applied the Kruskal-Wallis test on PC 1 to 5 (columns 5 to 10 in the dataframe) to inspect if there is any bias provided by different cell type (biological bias, not batch), disease (expactable), donor-ID and biomaterial provider (columns 1 to 4 respectively). For that a matrix of p-values was created with PCs as columns and sources of variation as rows.

```
heatmap_kruskal <- matrix(nrow=4,ncol=5)
```

```
for (j in 1:4){
  for (i in 5:9) {
    heatmap_kruskal[j,i-4] <- kruskal.test(kruskal_df[,i] ~ kruskal_df[,j],
                                           data = kruskal_df)$p.value

  }
}
```

Wilcoxon rank-sum test

Wilcoxon rank-sum test is similar to previously used Kruskal-Wallis test. It is also nonparametric and doesn't require normal distribution of the data. The null hypothesis states that it is equally likely that a randomly selected value from one sample will be less than or greater than a randomly selected value from a second sample. Wilcoxon rank-sum test can not be used to compare samples from more than two groups. Therefore, we applied Wilcoxon rank-sum test to investigate the influence of gender on patients' methylome profile.

```
gender <- sample_annotation[,36]
wilcoxon_df <- cbind(gender,pca_promoters)
colnames(wilcoxon_df) <- c(1:19)
```

P values were put into a matrix.

```
heatmap_wilcoxon <- matrix(nrow=1,ncol=5)
for (i in 2:6) {
  heatmap_wilcoxon[1,i-1] <- wilcox.test(wilcoxon_df[,i] ~ wilcoxon_df[,1],
                                         data = wilcoxon_df)$p.value

}
```

Permutation test

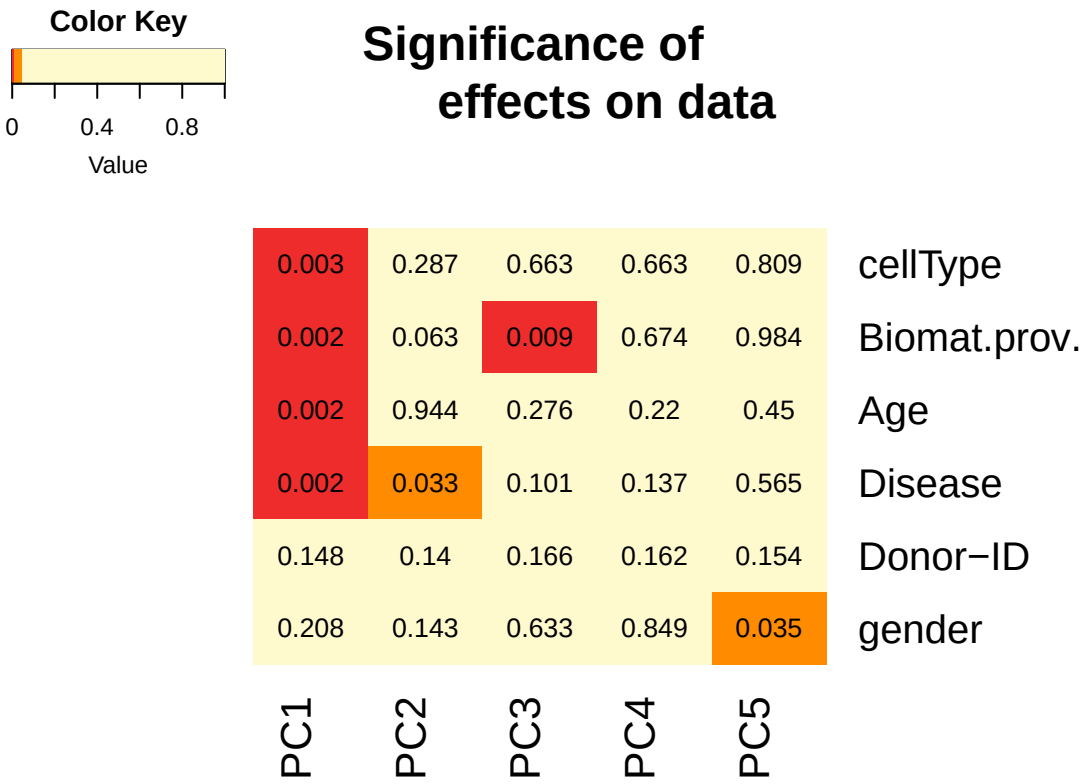
Permutation test (or a re-randomization test) is a type of statistical significance test in which the distribution of the test statistic under the null hypothesis is obtained by calculating all possible values of the test statistic under rearrangements of the labels on the observed data points. First we calculate Pearsons r correlation coefficient and its significance is tested by parametric t-test.

```
age <- sample_annotation[,c(34)]
age_separated <- strcapture("(.*)-(.*)", as.character(age), data.frame(type_1 = "", type_2 = ""))
age <- age_separated[,1]
correlation_df <- cbind(age, pca_promoters)
heatmap_correlation <- matrix(nrow=1,ncol=5)
cor.perm <- function (x,y, nperm = 499)
{
  r.obs <- cor (x = x, y = y)
  P.par <- cor.test (x = x, y = y)$p.value
  r.per <- sapply (1:nperm, FUN = function (i) cor (x = x, y = sample (y)))
  r.per <- c(r.per, r.obs)
  P.per <- sum (abs (r.per) >= abs (r.obs))/(nperm + 1)
  return (P.per = P.per)
}
for (i in 2:6){
  heatmap_correlation[1, i-1] <- cor.perm(x=correlation_df[,1], y=correlation_df[,i])


}
```

Heatmap

In order to visualize the significance of the effects on our PCs, a heatmap with the p values of the different tests was plotted. Red cells have a p value below 0.01, orange means $0.01 < p < 0.05$, while white means above 0.5 and therefore not significant.

```
heatmap <- rbind(heatmap_kruskal,heatmap_wilcoxon,heatmap_correlation)
colnames(heatmap) <- c("PC1","PC2","PC3","PC4","PC5")
rownames(heatmap) <- c("cellType","Disease","Biomat.prov.",
                       "Donor-ID","gender","Age")
library(gplots)
my_palette <- colorRampPalette(c("firebrick2","darkorange","lemonchiffon"))(n = 3)
col_breaks <- c(0,0.01,0.05,1)
par(mar=c(1,1,1,1))
heatmap.2(heatmap,
          main = "Significance of
          effects on data",
          density.info="none",
          trace="none",
          lwid= c(2,5),
          margins =c(6,10),
          col=my_palette,
          breaks=col_breaks,
          dendrogram = "none" ,
          Colv = "NA",
          cellnote= round(heatmap, digits=3),
          notecol="black"
)
```

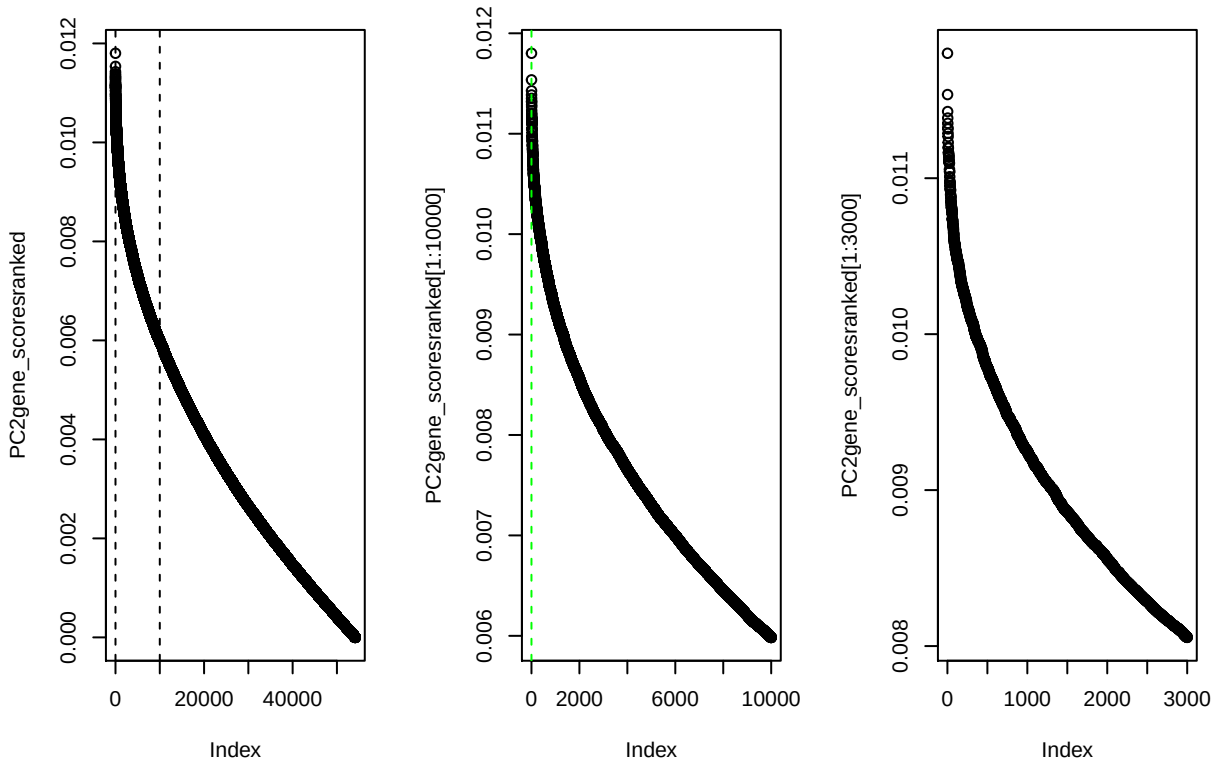


The results of this heatmap show us that PC1 is heavily influenced by effects that we could consider batch (such as Biomaterial Provider). Therefore, we rely on PC2 for our analysis from now on.

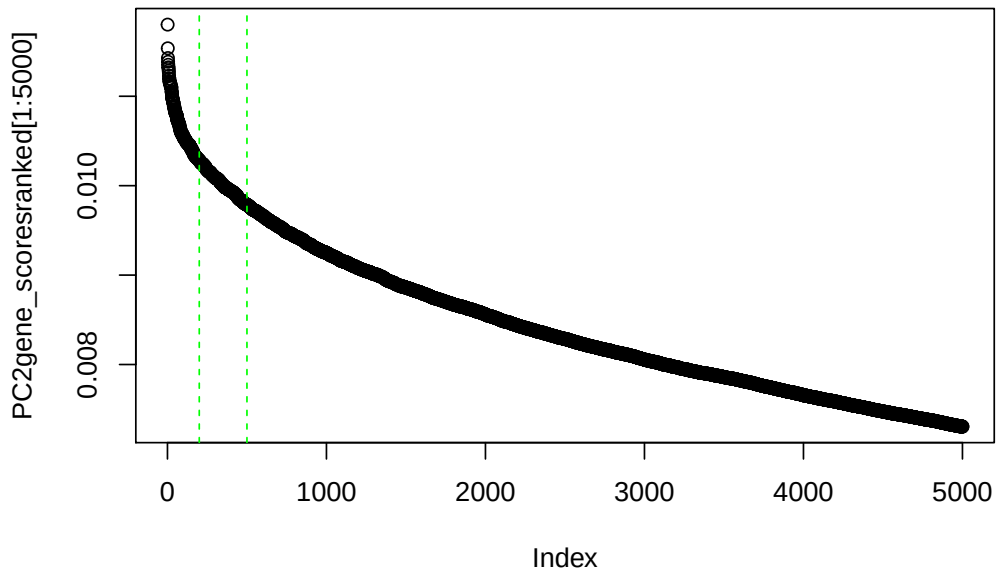
Feature Selection

We selected the most relevant promoters in our analysis by looking at the loading scores of the promoters for PC2. We ranked their absolute values (negative contributions are also relevant) in decreasing order and tried to find an elbow sopt in the plot.

```
loading_scores <- promoters_pca$rotation
gene_scores <- abs(loading_scores)
PC2gene_scoresranked <- sort(gene_scores[,2], decreasing = TRUE)
par(mfrow=c(1,3)) #Overview of PC2
plot(PC2gene_scoresranked)
abline(v=c(0,10000), lty=2)
plot(PC2gene_scoresranked[1:10000])
abline(v=c(0), col="green", lty=2)
plot(PC2gene_scoresranked[1:3000])
```




```
plot(PC2gene_scoresranked[1:5000])
abline(v=c(200, 500), col="green", lty=2)
```



We performed k clustering to check if the amount of promoters selected suffices to separate AML and healthy patients.

```
GOI_200 <- PC2gene_scoresranked[1:200]
GOI_200_names <- names(GOI_200)
GOI_200_cluster <- promoters_matrix_noInf[GOI_200_names,]
GOI_200_k <- kmeans(t(GOI_200_cluster), centers=2)
print(GOI_200_k$cluster)
```

```
## AML1 AML2 AML3 AML4 AML5 AML6 AML7 AML8 AML9 CON1 CON2 CON3 CON4 CON5 CON6
##    2    1    1    2    2    2    1    1    1    2    2    2    2    2    2
## CON7 CON8 CON9
##    2    2    2
```

```
GOI_500 <- PC2gene_scoresranked[1:500]
GOI_500_names <- names(GOI_500)
GOI_500_cluster <- promoters_matrix_noInf[GOI_500_names,]
GOI_500_k <- kmeans(t(GOI_500_cluster), centers=2)
print(GOI_500_k$cluster)
```

```
## AML1 AML2 AML3 AML4 AML5 AML6 AML7 AML8 AML9 CON1 CON2 CON3 CON4 CON5 CON6
##    2    1    1    2    2    2    1    1    1    2    2    2    2    2    2
## CON7 CON8 CON9
##    2    2    2
```

Neither of the two "elbow spots" manages to split the patients correctly. We first check if all the promoters can separate the patients correctly.

```
GOI_Everything <- kmeans(t(promoters_matrix_noInf), centers=2)
print(GOI_Everything$cluster)
```

```
## AML1 AML2 AML3 AML4 AML5 AML6 AML7 AML8 AML9 CON1 CON2 CON3 CON4 CON5 CON6
##    1    1    2    1    1    1    1    1    1    2    2    2    2    2    2
## CON7 CON8 CON9
##    2    2    2
```

```
GOI_22750 <- PC2gene_scoresranked[1:22750]
GOI_22750_names <- names(GOI_22750)
GOI_22750_cluster <-promoters_matrix_noInf[GOI_22750_names,]
set.seed(999)
GOI_22750_k <- kmeans(t(GOI_22750_cluster), centers=2)
print(GOI_22750_k$cluster)
```

```
## AML1 AML2 AML3 AML4 AML5 AML6 AML7 AML8 AML9 CON1 CON2 CON3 CON4 CON5 CON6
##    1    1    2    1    1    1    1    1    1    2    2    2    2    2    2
## CON7 CON8 CON9
##    2    2    2
```

From the clustering we can see that patient AML 3 is an outsider, as it doesn't cluster correctly despite using all the information available. For that reason, it was removed from the analysis

```
GOI_Feature_Selection <- GOI_22750_cluster[,-3]
set.seed(999)
GOI_Feature_Selection_k <- kmeans(t(GOI_Feature_Selection), centers=2)
print(GOI_Feature_Selection_k$cluster)
```

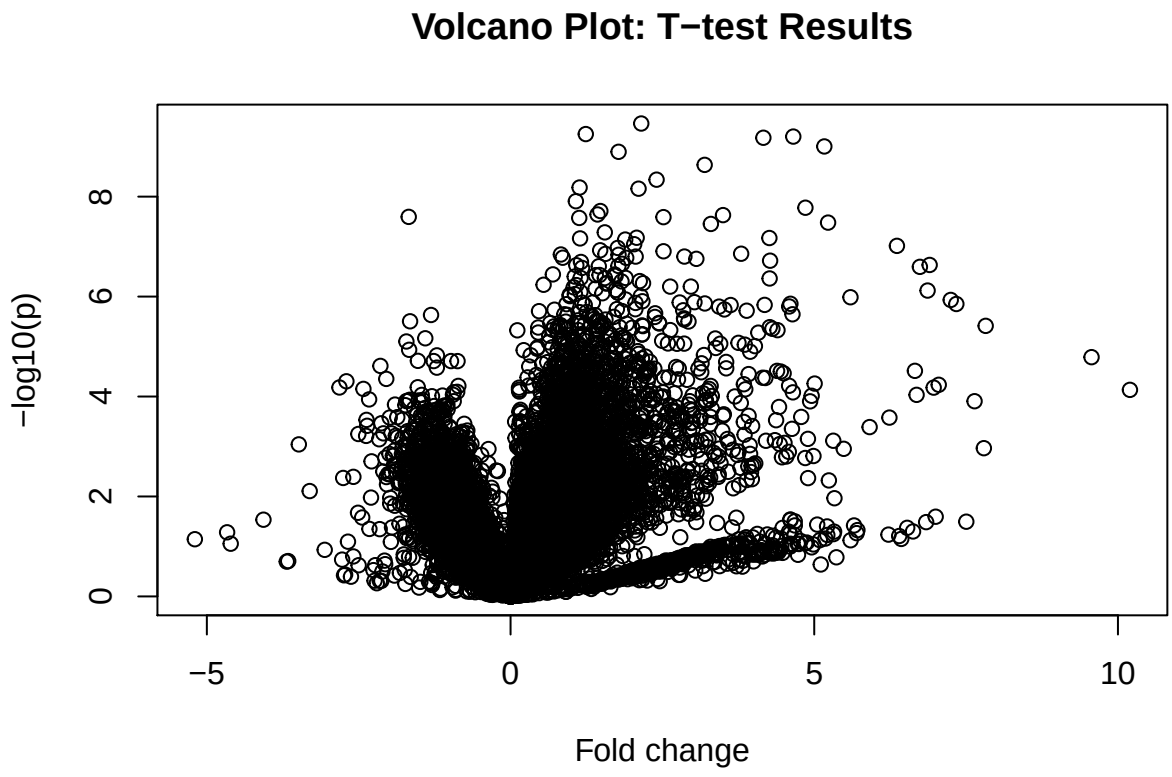
```
## AML1 AML2 AML4 AML5 AML6 AML7 AML8 AML9 CON1 CON2 CON3 CON4 CON5 CON6 CON7
##    1    1    1    1    1    1    1    1    2    2    2    2    2    2    2
## CON8 CON9
```

```
##     2     2
```

DMR Detection

We used a Students' (two-sided, unpaired) t-test to check for differentially methylated regions.

```
Ttest_results <- data.frame(Stat = rep(0,nrow(GOI_Feature_Selection)),
                            pval = rep(0,nrow(GOI_Feature_Selection)),
                            mean_difference = rep(0,nrow(GOI_Feature_Selection))
                            )
GOI_Feature_Selection <- as.matrix(GOI_Feature_Selection)
for (i in 1:nrow(GOI_Feature_Selection)) {

  x = GOI_Feature_Selection[i,1:8]
  y = GOI_Feature_Selection[i,9:17]
  tt <- t.test(x,y)
  Ttest_results[i,1] <- tt$statistic
  Ttest_results[i,2] <- tt$p.value
  Ttest_results[i,3] <- mean(x) - mean(y)

}
rownames(Ttest_results) <- rownames(GOI_Feature_Selection)
plot(Ttest_results$mean_difference, -log10(Ttest_results$pval),
     main = "Volcano Plot: T-test Results",
     xlab="Fold change",
     ylab="-log10(p)")
```



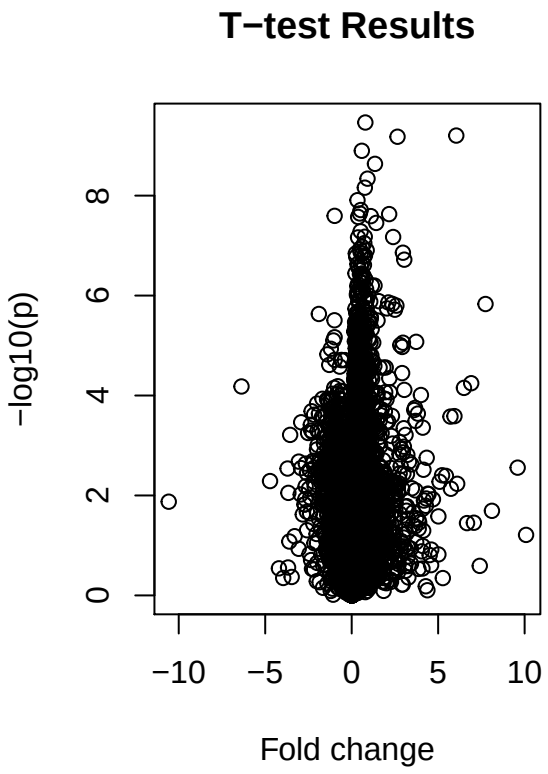
We put a threshold to the amount of difference in methylation that we will consider. We decided to keep promoters which had a mean difference between cohorts of at least one.

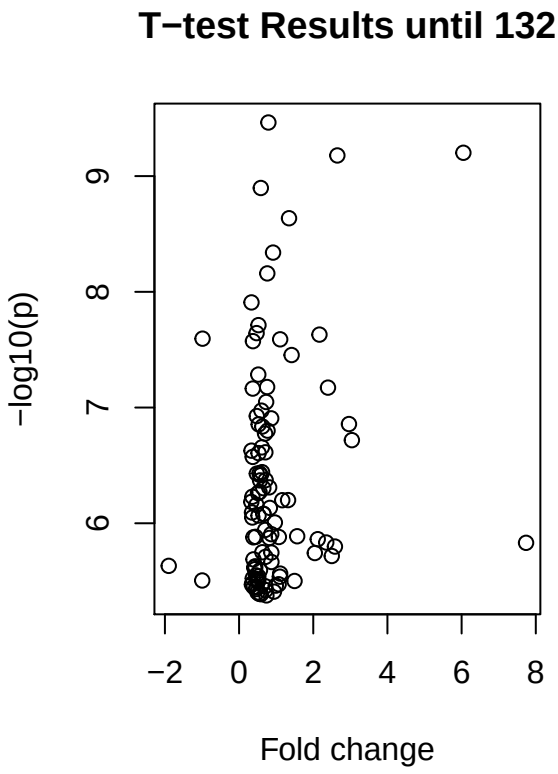
```
Ttest_results_rankedMean <- Ttest_results[order(abs(Ttest_results$mean_difference)),]
Ttest_results_trimmedMean <-
  Ttest_results_rankedMean[which(Ttest_results_rankedMean$mean_difference >= 1 |
                                   Ttest_results_rankedMean$mean_difference <= -1 ),]
Ttest_results_rankedPValue <- Ttest_results_trimmedMean[order(Ttest_results_trimmedMean$pval),]
```

We performed a multiple comparison correction to the results of the t-tests. We're going to use the Holm-Šidák correction which is more powerful than the Holm-Bonferroni correction and the Bonferroni correction.

```
Fail_reject <- 0
Index <- 1
alpha <- 0.05
m <- length(t(Ttest_results_rankedPValue))

while (Fail_reject == 0) {

  alpha2 = 1-((1-alpha)^(1/(m-(Index-1))))

  if(Ttest_results_rankedPValue[Index,2] > alpha2 ){

    Fail_reject <- Index

  }else{

   Index = Index+1
  }
}



par(mfrow=c(1,2))
#Results before correction


Unreliable_GOI_AML <- cbind(GOI_Feature_Selection[,1:8], rowMeans(GOI_Feature_Selection[,1:8]))
Unreliable_GOI_CON <- cbind(GOI_Feature_Selection[,9:17], rowMeans(GOI_Feature_Selection[,9:17]))

plot(x = log2(Unreliable_GOI_AML[,9])
        -log2(Unreliable_GOI_CON[,10]),
     y =-log10(Ttest_results$pval),
     main = "T-test Results",
     xlab="Fold change",
     ylab="-log10(p)")

#Results after correction

ReliableGOI <- GOI_Feature_Selection[rownames(Ttest_results_rankedPValue[1:132,]),]
ReliableGOI_AML <- cbind(ReliableGOI[,1:8], rowMeans(ReliableGOI[,1:8]))
ReliableGOI_CON <- cbind(ReliableGOI[,9:17], rowMeans(ReliableGOI[,9:17]))

plot(x = log2(ReliableGOI_AML[,9]) -
         log2(ReliableGOI_CON[,10]),
```

```
    y = -log10(Ttest_results_rankedPValue[1:132,2]),
    main = paste("T-test Results until", Fail_reject-1),
    xlab="Fold change",
    ylab="-log10(p)")
```





At 133 the p value start become unreliable, so we keep a list with the first 132 ones.

```
Reliable_Ttest_results_p <- Ttest_results_rankedPValue[1:132,]
Reliable_Ttest_results_fold <- Reliable_Ttest_results_p[order(Reliable_Ttest_results_p$mean_difference,
                                                              decreasing = T),]
```

To make the inspection easier, we created tables with the information of the 132 promoters with realiable p values.

```
names <- rownames(Reliable_Ttest_results_p)
promotersQC <- promoters[names, ]
Selected_promoters_p <- promotersQC[rownames(Reliable_Ttest_results_p[1:20,]),1:10]
Selected_promoters_fold <- promotersQC[rownames(Reliable_Ttest_results_fold[1:20,]),1:10]
#Export the tables
write.csv(Selected_promoters_p, file="Selected_promoters_p.csv", row.names = T)
write.csv(Selected_promoters_fold, file="Selected_promoters_fold.csv", sep=",", row.names = T)
```

```
## Warning in write.csv(Selected_promoters_fold, file =
## "Selected_promoters_fold.csv", : attempt to set 'sep' ignored
```

Interpretation

Overview of the top 20 genes

After performing our analysis, we obtained 2 lists of 20 elements each, which gave us the most differentially methylated regions in the genome of AML patients compared to control patients.

Table 1: The top 20 genes based on significance (p) and magnitude of difference (fold)

p	fold
ENSG00000244425	ENSG00000213244
ENSG00000233893	ENSG00000277775
ENSG00000166676	ENSG00000275713
ENSG00000227196	ENSG00000273983
ENSG00000187553	ENSG00000277075
ENSG00000233469	ENSG00000278588
ENSG00000253921	ENSG00000218690
ENSG00000248339	ENSG00000124788
ENSG00000214188	ENSG00000226425
ENSG00000230848	ENSG00000187553
ENSG00000276489	ENSG00000267260
ENSG00000267260	ENSG00000166676
ENSG00000252639	ENSG00000215841
ENSG00000240449	ENSG00000159374
ENSG00000211909	ENSG00000170128
ENSG00000259082	ENSG00000185156
ENSG00000279624	ENSG00000186047
ENSG00000271754	ENSG00000128573
ENSG00000226425	ENSG00000229659
ENSG00000224183	ENSG00000197992

Promoters to inspect

We chose following promoters because of their relevance in previous studies of AML. Further explanaition is given below.

Based on significance	Based on difference
RN7SL268P	HIST2H3DP1
EZR-AS1	HIST1H variants
TVP23A	CYP26C1
IGHD4-23	DLEU7
CYP26C1	MFSD6L

Significance

RN7SL268P

The GeneCards Summary for RN7SL268P Gene states:

"RN7SL268P (RNA, 7SL, Cytoplasmic 268, Pseudogene) is a pseudogene, and is affiliated with the SRP RNA class.""

This gene has been linked to Bladder Urothelial Carcinoma, where together with other genes located in the

same chromosome, it shows an abnormal high number of copy numbers. (Center 2016).

EZR-AS1

The EZR-AS1 gene is an antisense lncRNA which has been found to have a very strong link with esophageal squamous cell carcinoma (ESCC). It promotes the invasiveness of the tumor by enhancing the transcription of the EZR gene, which participates in cell migration. (X.-D. Zhang et al. 2017).

TVP23A

TVP23A is a **T**rans-Golgi network **V**esicle **P**rotein, which is thought to be involved in the delivery secretory and membrane proteins to the endosome (Atlas 1999). According to The Human Protein Atlas, a low expression of this gene is related to slight increase in the survival rate of glioma patients; whereas a high expression is related to a worse prognosis in melanoma.
Although, the evidence level for this gene is at a transcript level, which reduces the significance of these results. [//]: # (Comment: I'm not super sure about the sentence above) #### IGHD4-23 IGHD4 is one gene casette part of the immunoglobulin genes. It participates in VDJ (V - variable, D - diversity and J - joining) rearrangements that are responsible for B/T-cell receptor specificity and therefore cell clonality. These rearrangements are used currently as biomarkers for blood cancer cells, as they arise from one cell and are therefore clonal. #### CYP26C1 As a memeber of the Cytochrome P450 family, this protein is involved in the methabolism of drugs and synthesis of lipids. The promoter of this protein was also found to be highly methylated in AML cells (Cecotka and Polanska 2018). CYP26C1 is involved in the metabolism of retinoic acid, and it is more potent than CYP26A1 and CYP26B1 in the clearance of all-trans retinoic acid and its derived metabolites (Zhong et al. 2018). Retinoic acid plays an important role in the differentiation of myeloid cells, which is why it is used as a treatment for Acute Promyelocytic Leukemia (M. Huang et al. 1988). The higher methylation rate of the promoter of CYP26C1 might seem counter intuitive, seeing that a downregulation of this protein would mean a higher concentration of retinoic acid and therefore a differentiation of cancer cells to their corresponding myeloid stages. Yet the methylation could have an activating effect: the promoter's affinity to a Transcription Factor could be increased, thereby enhancing the expression of this gene, which would in turn lower the concentration of retinoic acid and prevent the cancerous cells from differentiating. A comparison with gene expression data could shed more light into this topic. ## Fold change The 7 most differentially methylated promoters belong to genes coding for histones #### HIST2H3DP1 & HIST1H variants HIST1H variants: HIST1H3F HIST1H2BH HIST1H3G HIST1H2AE HIST1H2BI The methylation of histones has a well-known importance in the context of epigenetics, yet it is normally referred to **histone modifications**, such as the methylation of the Lysine residue in the 4th position of the protein H3 (H3K4me). The role of DNA methylation in the loci that code for those genes is not as clear.

Histone synthesis, a vital process of the S phase, is downregulated as a result of DNA damage (Su et al. 2004). This downregulation is mediated by the inhibition of cyclin E-Cdk2 which is resposible for activating NPAT, which in turn promotes the transcription of histone genes. AML cells, like other cancer cells, accumulate DNA damage over time as a result in mutations of genes related to DNA repair (Esposito and So 2014). Perhaps, a long exposure to DNA damage could end up in the methylation of histone genes as a measure of permanent silencing, instead of the "transient" silencing caused by the inhibition of cyclin E-Cdk2.

CYP26C1

The role of this protein has already been discussed in the previous section. Its presence in both top 20 rankings could imply a special relevance of the role of this gene in the development of AML.

DLEU7

This protein (Leukemia-Associated Protein 7, also known as Deleted In Lymphocytic Leukemia 7), owns its name precisely to its relationship with leukemia. The deletion of the 4th band in the 1st region of the large arm of chromose 13 (or deletion of 13q14 for short) is present in about 50% of CLL patients. DLEU7 was characterized as the gene found in that region possibly related with tumor development (Hammarsund

et al. 2004). Consequently, the fact that it is highly methylated (and therefore, silenced) in cancer cells is consistent with the link between its deletion and tumor development.

MFSD6L

MFSD6L is a paralog of MFSD6, which stands for "Major Facilitator Superfamily Domain-Containing Protein 6", or also "Macrophage MHC Receptor 2". This gene has been found to be a biomarker for the distinction between monoclonal and dual AML cell populations (Nepstad et al. 2018). It has also been found to be upregulated in Leukemic Stem Cells (defined as CD34+CD38-cells from AML patients), compared to CD34+CD38+ cells (Pabst et al. 2016).

Comparison with literature

1: Saied et al (2012), "Genome Wide Analysis of Acute Myeloid Leukemia Reveal Leukemia Specific Methylome and Subtype Specific Hypomethylation of Repeats"

By accessing the suplementary information of the above mentioned paper, we could compare the results of our analysis to the work of other researchers working in the same topic. The most useful element of comparison could be the ENSEMBL identifier, which was used as row names for our entire dataset.

```
library(readxl) #package to import excel tables to R
DMR_Saied <- read_excel("~/R/Literature/DMRs_in_literature(Saied_et_al).xls",
                        sheet = 1) #Sheet 1 for promoters
DMR_Saied <- as.data.frame( DMR_Saied[,c(1:4)] )
```

```
DMR_lit_names <- DMR_Saied[,"ID"]
DMR_lit_symbol <- which(data$promoters$symbol %in% DMR_lit_names) #which are found?
DMR_lit_ensembl <- rownames( promoters[DMR_lit_symbol,] ) #Ensebl identifier
print(length(DMR_lit_ensembl))
```

```
## [1] 99
```

Now that we have a vector of ENSEMBL names (e.g "ENSG00000012345"), we will try to match it to our dataset. First we check how many of those 99 promoters can be found in the dataset we obtained after Quality Control:

```
promoters_normalized <- read.csv(file="promoters_normalized.csv")
promoters_normalized_in_Saied <- promoters_normalized[DMR_lit_ensembl,]
print(nrow(promoters_normalized_in_Saied))
```

```
## [1] 99
```

Then we check for the dataset after Feature Selection:

```
GOI_Feature_Selection_in_Saied <-
  GOI_Feature_Selection[which(rownames(GOI_Feature_Selection) %in% DMR_lit_ensembl),]
print(nrow(GOI_Feature_Selection_in_Saied))
```

```
## [1] 14
```

And finally in the final top 20 by significance and by fold change:

```
top20_p <- read.csv(file="Selected_promoters_p.csv")
top20_fold <- read.csv(file="Selected_promoters_fold.csv")
top20_fold_names <- sapply(top20_fold[, "X"], as.character)
top20_p_names <- sapply(top20_p[, "X"], as.character)
print( which(top20_fold_names %in% DMR_lit_ensembl) )
```

```
## integer(0)
```

```r
print( which(top20_p_names %in% DMR_lit_ensembl) )
```

```
## integer(0)
```

The promoters that were found to be differentially methylated by this paper were kept for the feature selection but failed to be found among the 20 most significantly DM promoters or the 20 with the biggest methylation difference.

2: Agnieszka Cecotka and Joanna Polanska (2018), "Region-Specific Methylation Profiling in Acute Myeloid Leukemia"

We proceed with a similar approach as with the previous source: Access the list of top methylated promoters via the Supplementary Materials and compare it to the different steps of our analysis.

In this case, the list of promoters was divided into the degree of methylation, which allows another, more quantitative level of comparison. After importing the dataset, we could see that there were 5 levels of methylation: down, up, at least medium, at least high and extremely up; all referring to the status of methylation in AML cells compared to healty Hematopoetic Stem Cells.

While the comparison might be limited by the different cell types used for the analysis, it still serves as a reference to take into account.

```r
library(readxl) #package to import excel tables to R
DMR_Cecotka <- read_excel("~/R/Literature/Cecotka_Region_specific_meth_AML_SM2.xlsx",
                          sheet = 2) #Sheet 2 for Transcription Starting Sites (= promoters)
TSS_down <- DMR_Cecotka[2:(which(is.na(DMR_Cecotka[,1]))[1]-1),1]
TSS_up <- DMR_Cecotka[2:nrow(DMR_Cecotka),4]
TSS_at_least_med <- DMR_Cecotka[2:(which(is.na(DMR_Cecotka[,7]))[1]-1),7]
TSS_at_least_high <- DMR_Cecotka[2:(which(is.na(DMR_Cecotka[,10]))[1]-1),10]
TSS_extreme <- DMR_Cecotka[2:(which(is.na(DMR_Cecotka[,13]))[1]-1),13]
#We need to turn the names into character vectors in order to compare them
TSS_down <- sapply(TSS_down, as.character)
TSS_up <- sapply(TSS_up, as.character)
TSS_at_least_med <- sapply(TSS_at_least_med, as.character)
TSS_at_least_high <- sapply(TSS_at_least_high, as.character)
TSS_extreme <- sapply(TSS_extreme, as.character)
```

We compared the names found in "down", "up" and "extreme up" to the promoters in our dataset.In order to make the comparison more quantitative, we also calculated the mean methylation difference between cancer and control patients for each promoter. In principle, the differences should be "down" < "up" < "extreme".

- Down

```r
DMR_down_symbol <- which(data$promoters$symbol %in% TSS_down)
DMR_down_ensembl <- rownames( promoters[DMR_down_symbol,] )
GOI_Feature_Selection_in_Cecotka_down <-
  GOI_Feature_Selection[which(rownames(GOI_Feature_Selection) %in% DMR_down_ensembl),]
GOI_Feature_Selection_in_Cecotka_down_diff <- apply(GOI_Feature_Selection_in_Cecotka_down,
                                                    1, function(x){
                                                   mean(x[1:8]) - mean(x[9:17])
                                                   })
```

- Up

```r
DMR_up_symbol <- which(data$promoters$symbol %in% TSS_up)
DMR_up_ensembl <- rownames( promoters[DMR_up_symbol,] )
GOI_Feature_Selection_in_Cecotka_up <-
  GOI_Feature_Selection[which(rownames(GOI_Feature_Selection) %in% DMR_up_ensembl),]
```

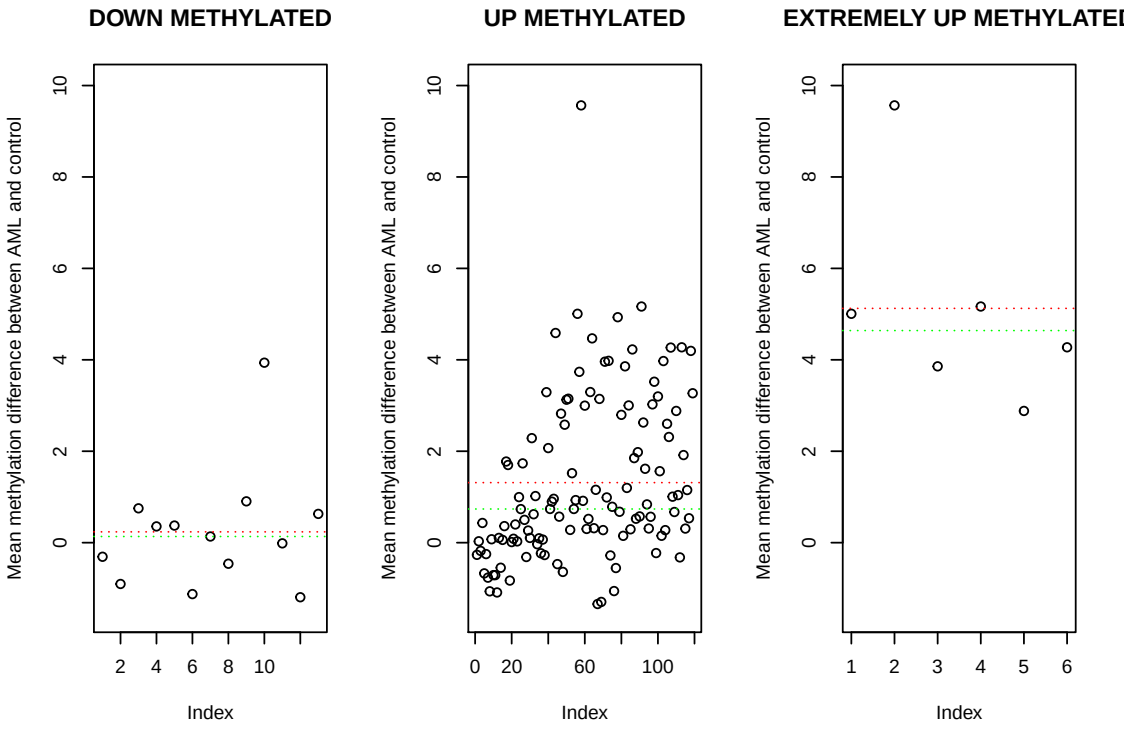
```
GOI_Feature_Selection_in_Cecotka_up_diff <- apply(GOI_Feature_Selection_in_Cecotka_up,
                                                 1, function(x){
                                                  mean(x[1:8]) - mean(x[9:17])
                                                 })
```

- Extreme up

```
DMR_extreme_symbol <- which(data$promoters$symbol %in% TSS_extreme)
DMR_extreme_ensembl <- rownames( promoters[DMR_extreme_symbol,] )
GOI_Feature_Selection_in_Cecotka_extreme <-
  GOI_Feature_Selection[which(rownames(GOI_Feature_Selection) %in% DMR_extreme_ensembl),]

GOI_Feature_Selection_in_Cecotka_extreme_diff <- apply(GOI_Feature_Selection_in_Cecotka_extreme,
                                                 1, function(x){
                                                  mean(x[1:8]) - mean(x[9:17])
                                                 })
```

All differences plotted together (green lines: median ; red lines: mean).

```
par(mfrow=c(1,3))
plot(GOI_Feature_Selection_in_Cecotka_down_diff,
     ylim=c(-1.5, 10),
     main="DOWN METHYLATED",
     ylab="Mean methylation difference between AML and control")
abline( h=mean(GOI_Feature_Selection_in_Cecotka_down_diff, na.rm=T), lty=3,col="red")
abline( h=median(GOI_Feature_Selection_in_Cecotka_down_diff, na.rm=T), lty=3,col="green")
plot(GOI_Feature_Selection_in_Cecotka_up_diff,
     ylim=c(-1.5, 10),
     main="UP METHYLATED",
     ylab="Mean methylation difference between AML and control")
abline( h=mean(GOI_Feature_Selection_in_Cecotka_up_diff, na.rm=T), lty=3,col="red")
abline( h=median(GOI_Feature_Selection_in_Cecotka_up_diff, na.rm=T), lty=3,col="green")
plot(GOI_Feature_Selection_in_Cecotka_extreme_diff,
     ylim=c(-1.5, 10),
     main="EXTREMELY UP METHYLATED",
     ylab="Mean methylation difference between AML and control")
abline( h=mean(GOI_Feature_Selection_in_Cecotka_extreme_diff, na.rm=T), lty=3,col="red")
abline( h=median(GOI_Feature_Selection_in_Cecotka_extreme_diff, na.rm=T), lty=3,col="green")
```



Next, we sought to check how many of those promoters had been selected in our analysis.

```
top20_p <- read.csv(file="Selected_promoters_p.csv")
top20_fold <- read.csv(file="Selected_promoters_fold.csv")
top20_fold_names <- sapply(top20_fold[, "X"], as.character)
top20_p_names <- sapply(top20_p[, "X"], as.character)
#DOWN
which(top20_fold_names %in% DMR_down_ensembl)
```

```
## integer(0)
```

```
which(top20_p_names %in% DMR_down_ensembl)
```

```
## integer(0)
```

```
#UP
which(top20_fold_names %in% DMR_up_ensembl)
```

```
## [1] 10 16 17
```

```
print(top20_fold[which(top20_fold$X %in% DMR_up_ensembl),"symbol"])
```

```
## [1] CYP26C1 MFSD6L  DLEU7
## 16 Levels: ATXN1 CLEC9A CYP26C1 DLEU7 FOXP2 GPR25 ... TVP23A
```

```
which(top20_p_names %in% DMR_up_ensembl)
```

```
## [1] 5
```

```
print(top20_p[which(top20_p$X %in% DMR_up_ensembl),"symbol"])
```

```
## [1] CYP26C1
## 11 Levels: CYP26C1 EZR-AS1 IGHD4-23 IGHD5-24 MIR6829 ... TVP23A
```

```
#EXTREME UP
which(top20_fold_names %in% DMR_extreme_ensembl)
```

```
## [1] 10 16
```

```
print(top20_fold[which(top20_fold$X %in% DMR_extreme_ensembl),"symbol"])
```

```
## [1] CYP26C1 MFSD6L
## 16 Levels: ATXN1 CLEC9A CYP26C1 DLEU7 FOXP2 GPR25 ... TVP23A
```

```
which(top20_p_names %in% DMR_extreme_ensembl)
```

```
## [1] 5
```

```
print(top20_p[which(top20_p$X %in% DMR_extreme_ensembl),"symbol"])
```

```
## [1] CYP26C1
## 11 Levels: CYP26C1 EZR-AS1 IGHD4-23 IGHD5-24 MIR6829 ... TVP23A
```

The genes that match the analysis are the upmethylated promoters for the genes 'CYP26C1', 'MFSD6L' and 'DLEU7'. The discussion of their role in AML can be seen under 'Fold change'.

Logistic regression

We tried to build a model to determine the probability of a patient having cancer through the methylation data of a subset of the genes that we have selected to be relevant. The methylation value of these genes should not correlate with each other.

The parameters are chosen by the principle of maximum likelihood and usage of minimum amount of variables possible instead of least sum of squares. The principle of maximum likelihood states that the best logistic curve maximizes the product of all the probabilities of observing the given data.

First we built a model using all the patients, to see if the subset of genes that we chose would suffice to tell the AML and the control patients apart.

```
Health_Vector <- as.factor(sample_annotation$cellTypeGroup[-3])
Mvalues <- as.data.frame(t(GOI_Feature_Selection)[,1:10000])
Log_Regr_Data <- cbind(Mvalues,Health_Vector )
model <- glm(formula = Health_Vector~.,family = binomial(link = "logit"), data = Log_Regr_Data)
predict <- predict(model, newdata = Log_Regr_Data, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
model_1_summary <- summary(model)
print(model_1_summary$coefficients)
```

```
##                      Estimate Std. Error       z value  Pr(>|z|)
## (Intercept)     -2390.5977229 44182787.2 -5.410699e-05 0.9999568
## ENSG00000268390   -46.6733911  2639862.4 -1.768024e-05 0.9999859
## ENSG00000224506    -5.5534273   917517.7 -6.052665e-06 0.9999952
## ENSG00000200997   -43.0362678   800922.3 -5.373339e-05 0.9999571
## ENSG00000106609  -522.3073169  9095146.0 -5.742704e-05 0.9999542
## ENSG00000201351    -7.0959475  1278199.2 -5.551519e-06 0.9999956
## ENSG00000260426   107.4116060  1279197.8  8.396794e-05 0.9999330
## ENSG00000231777    62.1999444  1954360.1  3.182625e-05 0.9999746
## ENSG00000221036   142.1363583  2663924.7  5.335600e-05 0.9999574
## ENSG00000250041   -72.7067877  2343745.2 -3.102163e-05 0.9999752
```

```
## ENSG00000234768    22.6855501  1559393.1  1.454768e-05 0.9999884
## ENSG00000277912    -0.7655278   777973.3 -9.840026e-07 0.9999992
## ENSG00000279993   -43.1955224  1565256.6 -2.759645e-05 0.9999780
## ENSG00000234942     1.3761686   359560.9  3.827359e-06 0.9999969
## ENSG00000274312   -54.0843078   807627.0 -6.696694e-05 0.9999466
## ENSG00000199580   -10.8158463  1470579.8 -7.354818e-06 0.9999941
## ENSG00000224291    73.6427803  1000543.5  7.360278e-05 0.9999413
```

```
print(model_1_summary$aic)
```

```
## [1] 34
```

```
print(predict)
```

```
##         AML1         AML2         AML4         AML5         AML6
## 7.884924e-12 7.884924e-12 7.884924e-12 7.884924e-12 7.884924e-12
##         AML7         AML8         AML9         CON1         CON2
## 7.884924e-12 7.884924e-12 7.884924e-12 1.000000e+00 1.000000e+00
##         CON3         CON4         CON5         CON6         CON7
## 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
##         CON8         CON9
## 1.000000e+00 1.000000e+00
```

Then we did Cross validation. For that, this time we only picked a subset of patients to train the model and we applied the model to predict the diagnosis of the rest of the patients. Usually cross validation is more significant but in our case we do not have enough samples to built a significant model.

```
set.seed(13)
control = which(Log_Regr_Data$Health_Vector == "gran")
control_train = sample(control,floor(0.85*length(control)))
cancer = which(Log_Regr_Data$Health_Vector == "cancer")
cancer_train = sample(cancer,floor(0.85*length(cancer)))
train = c(control_train, cancer_train)
train_set <- Log_Regr_Data[train,]
test_set <- Log_Regr_Data[-train,]
model_2 <- glm(formula = Health_Vector~.,family = binomial(link = "logit"), data = train_set)
predict_2 <- predict(model_2, newdata =test_set , type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
model_2_summary <- summary(model_2)
print(model_2_summary$coefficients)
```

```
##                    Estimate Std. Error       z value  Pr(>|z|)
## (Intercept)     -1215.38344 49755268.3 -2.442723e-05 0.9999805
## ENSG00000268390   -34.83901  2340749.9 -1.488370e-05 0.9999881
## ENSG00000224506    41.61058  1068392.8  3.894690e-05 0.9999689
## ENSG00000200997   -58.17049   601178.5 -9.676077e-05 0.9999228
## ENSG00000106609  -256.07190  9783716.0 -2.617328e-05 0.9999791
## ENSG00000201351    85.83162   412100.1  2.082786e-04 0.9998338
## ENSG00000260426    57.27328   626255.0  9.145360e-05 0.9999270
## ENSG00000231777   -21.26334   919967.7 -2.311314e-05 0.9999816
## ENSG00000221036    54.01851  2102075.6  2.569770e-05 0.9999795
## ENSG00000250041   -16.94575  1779887.9 -9.520685e-06 0.9999924
## ENSG00000234768   -78.70695   464788.2 -1.693394e-04 0.9998649
## ENSG00000277912   -49.03404   756906.5 -6.478216e-05 0.9999483
```

```
## ENSG00000279993    56.82974  1375245.5  4.132334e-05 0.9999670
```

```
print(model_2_summary$aic)
```

```
## [1] 26
```

```
print(predict_2)
```

```
##          AML2          AML7          CON4          CON9
## 2.220446e-16 2.220446e-16 1.000000e+00 1.000000e+00
```

The AML2 and AML7 patients are clearly distinguished from the CON4 and CON9 controls; the model is able to predict accurately with only 85% of the patient to train with. Lower percentages produced less reliable results. The printed promoters are the ones that had a non-NA coefficient in model, which means that they are all independent from each other. All other promoters had NA as coefficient and are correlated to the printed ones and therefore are excluded from the model. Nevertheless, the p values of the coefficients are all higher than 0.999, which means that the reliability of the model is still scarce. This is probably due to the small sample size, as we only rely on the data of less than 20 patients to build it up.

References

Atlas, The Human Protein. 1999. "TVP23A." https://www.proteinatlas.org/ENSG00000166676-TVP23A/pathology.

Baylin, Stephen B. 2005. "DNA Methylation and Gene Silencing in Cancer." *Nature Reviews Clinical Oncology* 2 (S1). Nature Publishing Group: S4.

Bernardino, Jacqueline, Martine Lombard, Alain Niveleau, and Bernard Dutrillaux. 2000. "Common Methylation Characteristics of Sex Chromosomes in Somatic and Germ Cells from Mouse, Lemur and Human." *Chromosome Research* 8 (6). Springer: 513–25.

Cecotka, Agnieszka, and Joanna Polanska. 2018. "Region-Specific Methylation Profiling in Acute Myeloid Leukemia." *Interdisciplinary Sciences: Computational Life Sciences* 10 (1): 33–42. doi:10.1007/s12539-018-0285-4.

Center, Broad Insitute TGCA Genome Data Analysis. 2016. "LowPass Copy Number Analysis (Gistic2)." Broad Institute of MIT; Harvard.

Dong, Yiran, and Chao-Ying Joanne Peng. 2013. "Principled Missing Data Methods for Researchers." *SpringerPlus* 2 (1). Springer: 222.

Du, Pan, Xiao Zhang, Chiang-Ching Huang, Nadereh Jafari, Warren A Kibbe, Lifang Hou, and Simon M Lin. 2010. "Comparison of Beta-Value and M-Value Methods for Quantifying Methylation Levels by Microarray Analysis." *BMC Bioinformatics* 11 (1). BioMed Central: 587.

Esposito, Maria Teresa, and Chi Wai Eric So. 2014. "DNA Damage Accumulation and Repair Defects in Acute Myeloid Leukemia: Implications for Pathogenesis, Disease Progression, and Chemotherapy Resistance." *Chromosoma* 123 (6). Springer: 545–61.

Hammarsund, Marianne, Martin M Corcoran, William Wilson, Chaoyong Zhu, Stefan Einhorn, Olle Sangfelt, and Dan Grandér. 2004. "Characterization of a Novel B-Cll Candidate Gene–DLEU7–located in the 13q14 Tumor Suppressor Locus." *FEBS Letters* 556 (1-3). Wiley Online Library: 75–80.

Huang, ME, Yu-chen Ye, SR Chen, Jin-Ren Chai, Jia-Xiang Lu, L Zhoa, LJ Gu, and Zhen-Yi Wang. 1988. "Use of All-Trans Retinoic Acid in the Treatment of Acute Promyelocytic Leukemia." *Blood* 72 (2). Am Soc Hematology: 567–72.

Nepstad, Ina, Kimberley Joanne Hatfield, Tor Henrik Anderson Tvedt, Håkon Reikvam, and Øystein Bruserud. 2018. "Clonal Heterogeneity Reflected by Pi3k-Akt-Mtor Signaling in Human Acute Myeloid Leukemia Cells

and Its Association with Adverse Prognosis." *Cancers* 10 (9). Multidisciplinary Digital Publishing Institute: 332.

Pabst, Caroline, Anne Bergeron, Vincent-Philippe Lavallée, Jonathan Yeh, Patrick Gendron, Gudmundur L Norddahl, Jana Krosl, et al. 2016. "GPR56 Identifies Primary Human Acute Myeloid Leukemia Cells with High Repopulating Potential in Vivo." *Blood* 127 (16). Am Soc Hematology: 2018–27.

Ringnér, Markus. 2008. "What Is Principal Component Analysis?" *Nature Biotechnology* 26 (3). Nature Publishing Group: 303.

Su, Chuan, Guang Gao, Sandra Schneider, Christopher Helt, Carsten Weiss, Michael A O'Reilly, Dirk Bohmann, and Jiyong Zhao. 2004. "DNA Damage Induces Downregulation of Histone Gene Expression Through the G1 Checkpoint Pathway." *The EMBO Journal* 23 (5). John Wiley & Sons, Ltd: 1133–43.

Zhang, Xiao-Dan, Guo-Wei Huang, Ying-Hua Xie, Jian-Zhong He, Jin-Cheng Guo, Xiu-E Xu, Lian-Di Liao, et al. 2017. "The Interaction of lncRNA Ezr-As1 with Smyd3 Maintains Overexpression of Ezr in Escc Cells." *Nucleic Acids Research* 46 (4). Oxford University Press: 1793–1809.

Zhong, Guo, David Ortiz, Alex Zelter, Abhinav Nath, and Nina Isoherranen. 2018. "CYP26C1 Is a Hydroxylase of Multiple Active Retinoids and Interacts with Cellular Retinoic Acid Binding Proteins." *Molecular Pharmacology* 93 (5). ASPET: 489–503.