Analysis of methylation levels in acute lymphatic leukemia versus B-cell lines

by Wiebke Albers, David Steeb, Nina Winkler, Lilian Dorner

Introduction

DNA methylation

One of the cells regulatory mechanisms for switching genes on and off is methylation of cytosines in promotor or coding regions. The enzymes responsible for this methylation only recognize cytosines which are followed by a guanosine nucleotide in the strand. This conformation is called a CpG (cytosine phosphate guanosine). Upon methylation of the CpGs in promotor regions the transcription machinery is blocked. It can therefore not bind to its activating site which leads to downregulation or silencing of the downstream gene or genes. Methylation within a gene can be interpreted similarly. If an abundant amount of methylation is present within a gene sequence the RNA polymerase struggles to transcribe the DNA and is more likely to stop along the gene. This also leads to a downregulation of the said gene.

These CpGs are normally methylated. The exception to this are CpG-rich promoters called CpG islands which are rarely methylated. These CpG islands are often found as promoters of housekeeping genes which rarely ever show any methylation. In cancer however an epigenetic change in these CpG islands for example can lead to a major driving force for cancerous development. Hypermethylation of tumoursupressorgenes or hypomethylation of oncogenes inevitably favour cancer growth and development.

Tumour type

The type of cancer focussed here is acute lymphoblastic leukemia (ALL) arising from the B-cell line, which is the most common form with 75% of ALL cases (Terwillinger and Abdul-Hay, 2017). It is a very aggressive form of leukemia. Approximately 60% of leukemia-diseases in children under 20 years are due to ALL making it the predominant form of leukemia (Pui *et al.*, 2011). Combining this knowledge with methylation status this project is comparing genome-wide methylation data of diseased and healthy patients. The samples were either taken from the bone marrow (healthy and diseased) or the tonsils (healthy) of 0-18 year old patients.

Biological question - goal

The identification of recurrent genetic changes can help for refined individual disease prognosis and management. In this project the focus lies on the methylation status of promotors in ALL related genes. Based on a publication of Roman-Gomez the methylation status of the 15 mentioned genes is validated and further search for promoters and affiliated genes relevant for ALL is conducted. The data opens the possibility for new potential biomarkers for ALL risk prognosis and therapy.

Outline

First the data is cleaned by removing any non significant data. Additionally any NA is either removd or replaced by a set value. As a next step a PCA is performed in order to reduce the amount of data to compute without losing too much information. In the end a much lower amount of relevant promoters is filtered and proceeds to the next step: testing for differentially methylated regions (DMRs). In this step the diseased group and the healthy group are tested for differential methylation levels via t-tests. Next the promoters are sorted by significance (p-value). The most significant promoters were analysed in order to understand their function and use for the cancer cell. Finally a prediction model is created via logistical regression and tested for its accuracy.

Step1 - Quality control

Before starting any form of analysis the dataset first has to undergo different stages of data cleaning and quality control. In order to do that any non significant or ambiguous rows need to be removed and all NAs are either replaced or deleted. As a last step the Beta-values are converted into M-values for a statistically more accurate representation of the methylation levels.

Trimming the dataset and creation of matrices for later use

The columns “Strand”, “symbol”, “GC”, “C” und “G” are not relevant for this analysis and are therefore removed.

```
ALLBcell <- readRDS("ALL-Bcell_list.RDS.gz")
ALLpromotor <- ALLBcell[["promoters"]]
ALLpromotor <- ALLpromotor[,c(-4,-5,-6,-8,-9,-10)]
```

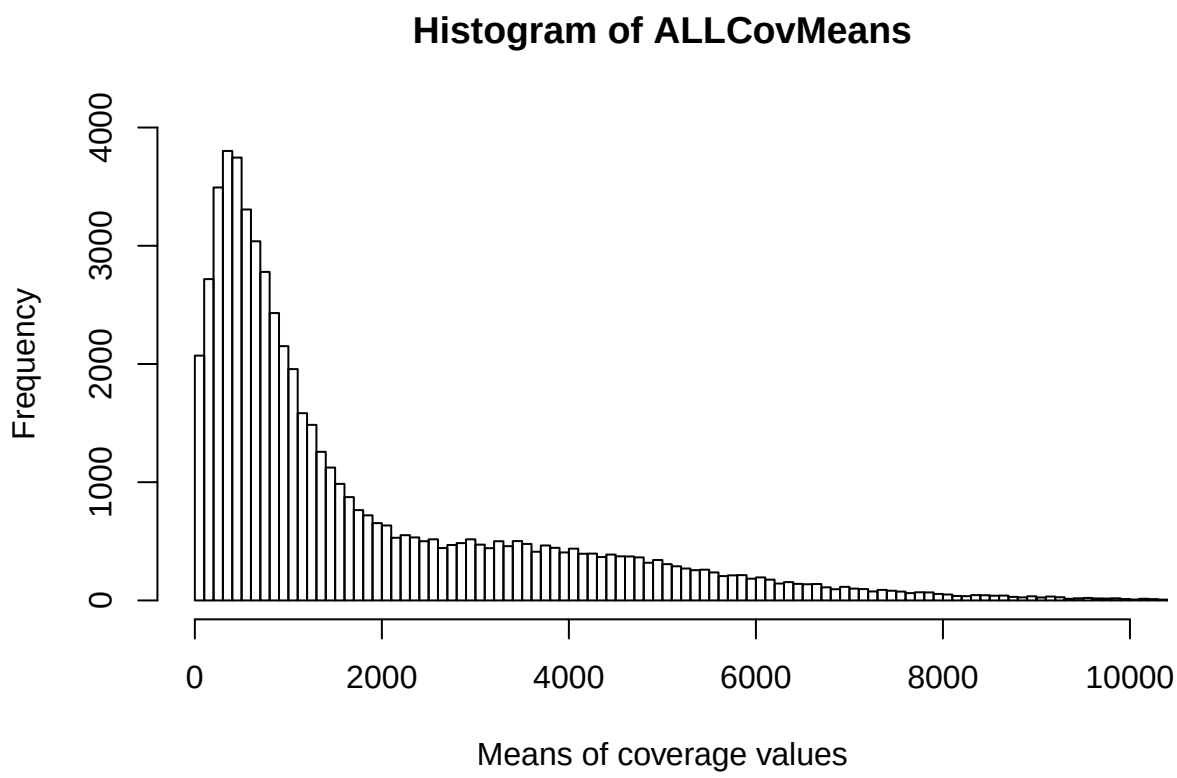
The coverage columns were isolated in a seperate dataframe which was called ALLpromotorCov. Next the means of the coverage of the promoters were calculated and stored in ALLCovMeans. Finally these values were logarithmically converted to obtain a simpler plot.

```
ALLpromotorCov <- ALLpromotor[,c(15:24)]
ALLCovMeans <- rowMeans(ALLpromotorCov)
ALLCovMeansLog <- log(ALLCovMeans)
```

Plotting the histograms for means

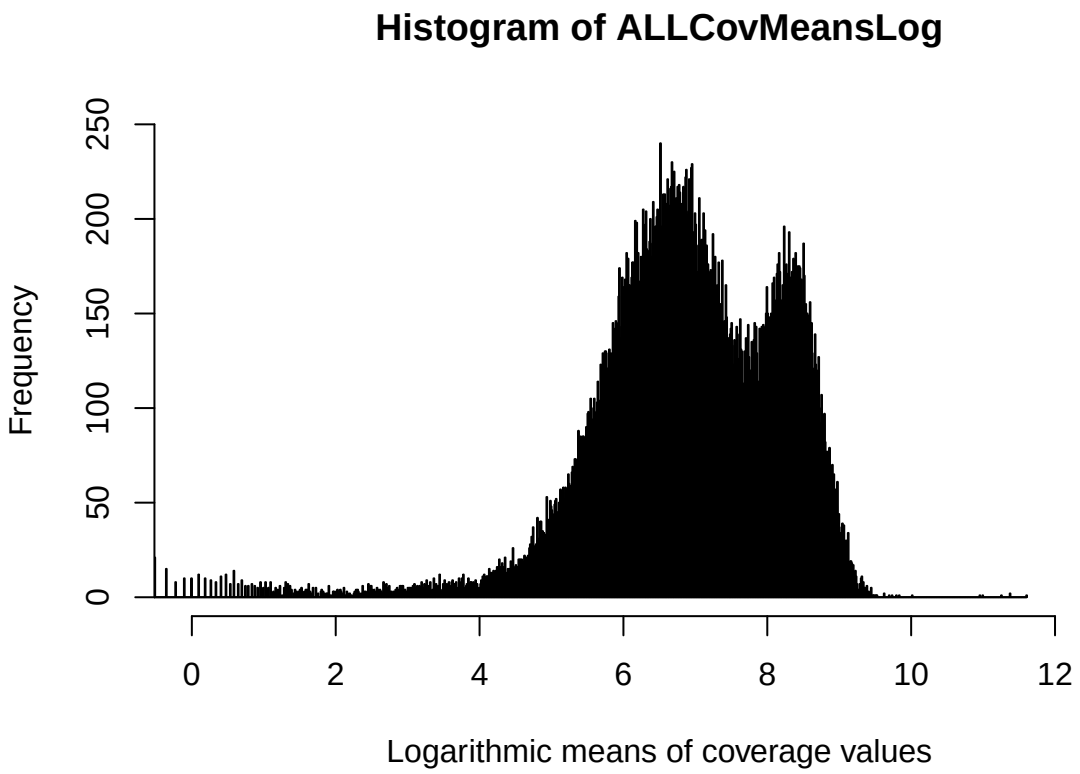
Plotting the means of the coverage values against their frequency.

```
hist(ALLCovMeans, xlab = "Means of coverage values", ylab = , xlim = c(0,10000),
ylim = c(0,4000), breaks = 1000, lwd = 1)
```



Plotting the logarithmic means of the coverage values against their frequency.

```
hist(ALLCovMeansLog, xlab = "Logarithmic means of coverage values", ylab = ,
xlim = c(0,13), ylim = c(0,250), breaks = 1000, lwd = 1)
```



Finding a suited threshold

The coverage equates to the average of the reads per base on any given gene region. During sequencing each base is determined by the majority of its reads. If there are very few (e.g. <25) reads for a single base or region it is prone to error due to statistical insecurity. The lower the amount of reads the higher are the chances that a few "wrong" reads alter the result of the read out base. If there are very many (e.g. >10000) reads per base the result could be altered due to an amplification error during the PCR. This error would then be amplified as well if it appears early during the amplification process. Therefore it is important to find a suited threshold for the coverages in order to assure the significance of the Beta-values which are directly depending on the sequencing accuracy. Upon literature reseach 14 promoters for the genes (CDH1, CDKN2A, CDKN2B, CDKN1C, KLK10, DKK3, CDH13, PYCARD, DAPK1, PARKIN, PTEN, p73, APAF1, LATS1) are defined as "relevant" for our tumour type (Roman-Gomez *et al.*, 2004). These are from now on referred to as selected genes/promoters. In order to find an upper threshold several factors were taken into account. First the amount of the selected promoters which were removed for different thresholds was checked. Even with a threshold of 99% of the coverage 3 promoters for genes (p73, APAF1, LATS1) always only had NA for the coverage values (any value outside of thresholds = NA). Next the percentage of the remaining regions which were removed was calculated, while still keeping as many selected promoters additionally reducing the coverage as much as needed to avoid PCR mediated errors. Finally a lower threshold of 1.5 % (28) and an upper threshold of 98.5 % (7753) of the coverage values were selected since only 10 % of the data and 2 additional selected promoter regions (p57, PTEN) were rejected. It formed the best compromise between the aforementioned factors. Additionally an extra variable ALLpromotorCov1 was defined which was subsequnetly used to compute the amount of data lost during the next steps of data cleaning.

```
quantile(ALLCovMeans, probs = c(0.05, 0.95),na.rm = TRUE)
```

```
##       5%      95%
##  138.900 5957.335
```

```
quantile(ALLCovMeans, probs = c(0.015, 0.985),na.rm = TRUE)
```

```
##      1.5%     98.5%
##   27.7165 7753.4350
```

```
ALLpromotorCov1 <- ALLpromotorCov
```

Next any values of the coverage above and below the thresholds were replaced with NA.

```
ALLpromotorCov1[ALLpromotorCov > 7753] <- NA
ALLpromotorCov1[ALLpromotorCov1 < 28] <- NA
```

Isolation of selected promoters before and after threshold cleanup

The presence of the selected promoters was checked and prepared for later calculation for the rejected amount of coverage values.

```
ALLCovGen <- ALLpromotorCov[c("ENSG00000039068","ENSG00000078900","ENSG00000147889",
"ENSG00000147883","ENSG00000129757","ENSG00000129451","ENSG00000050165","ENSG00000140945",
"ENSG00000103490","ENSG00000120868","ENSG00000196730","ENSG00000185345","ENSG00000131023",
"ENSG00000171862"),]
ALLCov1Gen <- ALLpromotorCov1[c("ENSG00000039068","ENSG00000078900","ENSG00000147889",
"ENSG00000147883","ENSG00000129757","ENSG00000129451","ENSG00000050165","ENSG00000140945",
"ENSG00000103490","ENSG00000120868","ENSG00000196730","ENSG00000185345","ENSG00000131023",
"ENSG00000171862"),]
```

Checking for percentage of rejected coverage values

Finally upon each tested threshold the amout of rejected values was calculated by subtracting the percentage of non NA values after threshold cleanup from 1. This calculation was always done for both selected and total promoter regions.

```
sum(!is.na(ALLpromotorCov))
```

```
## [1] 598120
```

```
rejectTotal <- (1-(sum(!is.na(ALLpromotorCov1))/sum(!is.na(ALLpromotorCov))))
reject <- (1-(sum(!is.na(ALLCov1Gen))/sum(!is.na(ALLCovGen))))
rejectTotal
```

```
## [1] 0.0630325
```

```
reject
```

```
## [1] 0.1857143
```

Pairing coverage and Beta-values

First a matrix containing only Beta-values is created under the name of ALLpromotorBeta.

```
ALLpromotorBeta <- ALLpromotor[,c(5:14)]
```

Since a NA in the coverage means there is no data backing up the Beta-value the latter is also converted to NA. This is done in order to prevent incorporation of not significant Beta-values into downstream analysis.

```
for(i in 1:nrow(ALLpromotorCov1)){
     for(j in 1:ncol(ALLpromotorCov1)){
         if(is.na(ALLpromotorCov1[i,j])){
             ALLpromotorBeta[i,j] <- NA
         }
```

```
    }
}
```

Elimination of all regions 4 or 5 out of 5 NA in Beta-value in both patient groups

Any gene region with 4 or more NA out of 5 for any patient group in the Beta-value coloumns is removed due to its low amount of information it can provide for testing of differentially methylated regions. Therefore if there is only one patient or none with a Beta-value for a specific gene region the latter is removed since the input data is not significant enough to be used in the downstream analysis.

```
for (i in 1:nrow(ALLpromotorBeta)) {
  for(j in  1:ncol(ALLpromotorBeta)){
    if(sum(is.na(ALLpromotorBeta[i,c(1:5)]))>=4){
      ALLpromotorBeta <- ALLpromotorBeta[-i,]
    }
  }
}
```

After execution of this command around 7.9 % of the promoter regions were removed due to 4 or 5 NA in healthy patients Beta-values.

```
for (i in 1:nrow(ALLpromotorBeta)) {
  for (j in 1:ncol(ALLpromotorBeta)) {
    if(sum(is.na(ALLpromotorBeta[i,c(6:10)]))>=4){
      ALLpromotorBeta <- ALLpromotorBeta[-i,]
    }
  }
}
```

After elimination through diseased patients another 1 % of promoter regions were removed. In total 8.9% (5225) of the promoter regions were removed.

Replacing residual NAs by the mean of the patient group

Any remaining NAs are now replaced by the patient group's mean Beta-value in order to completely remove NAs in the Beta-values from our dataset. The mean was selected instead of a random number to ensure that the result of the t-test in downstream analysis is not altered which uses the mean and standard deviation. Therefore the mean stays the same. The standard deviation does change but it seemed like a better option compared to a random number or complete deletion of said promoter.

```
for (i in 1:nrow(ALLpromotorBeta)) {       #replacement for healthy group
  for (j in 1:5) {
    if(is.na(ALLpromotorBeta[i,j])){
      ALLpromotorBeta[i,j] <- rowMeans(ALLpromotorBeta[i,c(1:5)], na.rm = TRUE)
    }
  }
}
```

```
for (i in 1:nrow(ALLpromotorBeta)) {       #replacement for diseased group
  for (j in 6:10) {
    if(is.na(ALLpromotorBeta[i,j])){
      ALLpromotorBeta[i,j] <- rowMeans(ALLpromotorBeta[i,c(6:10)], na.rm = TRUE)
    }
  }
}
```

The previous 4 code chunks were executed with a repeat loop until the number of NA in the dataset was equal to 0.

```
repeat{
  chunk1 #removal of gene regions with >=4 NA (healthy)

  chunk2 #removal of gene regions with >=4 NA (diseased)

  chunk3 #replacement of NA by mean of healthy patients

  chunk4 #replacement of NA by mean of diseased patients

  sum(is.na(ALLpromotorBeta))
  if(sum(is.na(ALLpromotorBeta)) == 0){
    break
  }
}
```

Preparation for normalization

The values 0 and 1 were replaced by 0.00001 and 0.99999 respectively in order to prevent mathematical errors during normalization. If this step is not done the resulting M-values would equate to positive and negative infinity.

```
ALLpromotorBeta1 <- ALLpromotorBeta
for (i in 1:nrow(ALLpromotorBeta)) {
  for (j in 1:ncol(ALLpromotorBeta)) {
    if((ALLpromotorBeta[i,j])==0){
      ALLpromotorBeta[i,j] <- 0.00001
    }
    if((ALLpromotorBeta[i,j])==1){
      ALLpromotorBeta[i,j] <- 0.99999
    }
  }
}
```

Normalization

In the last step of quality control the Beta-values were converted to M-values. M-values provide a much better performance for detection rate and true positive rate at highly methylated and highly umethylated CpG sites. The Beta-values have a higher statistical spread (heteroscedasticity) outside the middle methylation range. So the M-values provide a more accurate estimation of the methylation status (Pan Du *et al.*, 2010). However the M-values have no direct biological interpretation, whereas the Beta-values relate roughly to the methylation percentage. For this reason we also looked at the Beta-values in later on analysis.

```
ALLMvalue <- ALLpromotorBeta
for (i in 1:nrow(ALLMvalue)) {
  for (j in 1:ncol(ALLMvalue)) {
    ALLMvalue[i,j] <- log2(ALLMvalue[i,j]/(1-ALLMvalue[i,j]))
     }
}
```

Step2 - Principal component analysis

Before testing single promoters for differential methylation the amount of data needs to be reduced through principal component analysis (PCA). The dataset is therefore represented through a number of principal components (PC) with the first PC equating to the highest variance of the data. Each PC is influenced by any promoter in the dataset. If the associated value is high the promoter is important for the principal component. If the value is low this particular promoter is not very important for the PC. Upon plotting the variance of each PC/promoter the relevant PCs/promoters can be filtered by cutting off at a noticeable

elbow in the plot (as seen later). An elbow in the plot represents a cutoff point after which the following PCs/promoters do not add much information to the analysis. Therefore any following components/promoters can be removed without losing significant amounts of information about the data. Before selecting a principal component each PC has to be checked for batch effects. Batch effects are a form of technical error which creates correlations through e.g. submission date. The samples collected within a batch would then show correlation between each other without having a biological background. Therefore the PCs are checked for the absence of these effects. Next the previously selected PCs are plotted against each other in order to check for the best PC to pick for downstream analysis. Usually this is the first principal component (PC1). As a last step the variance of each promoter is sorted and plotted for the selected PC. Here an elbow is visible again which marks the significant promoters on its left side (high variance) and the not significant ones to its right (low variance). Finally the significant genes are isolated and proceed to the next major step.

Performing PCA and selection of relevant PCs

First the actual PCA is executed. The transposed matrix is used in order to obtain PCs which are linked to the patients instead of the promoters themselves.

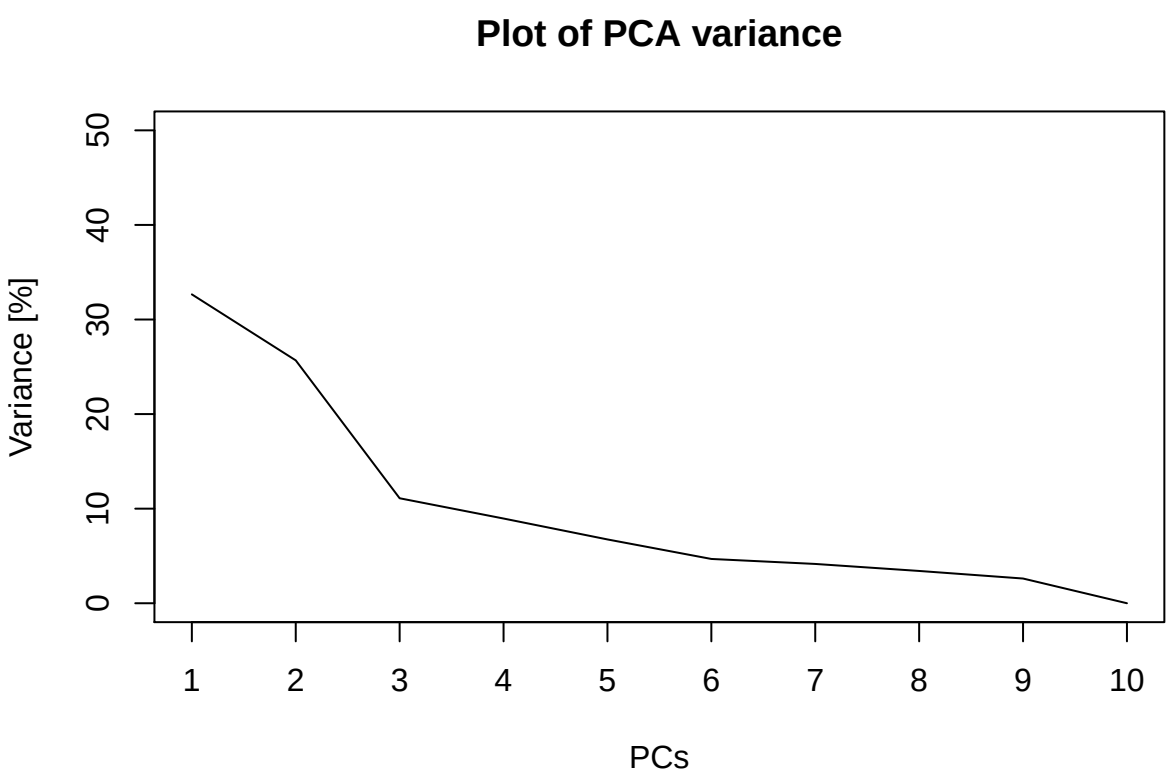
```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
pca <- prcomp(t(ALLMvalue))
```

Next the variance of the PCs is plotted to determine the elbow and therefore the relevant PCs.

```
pVar <- (pca$sdev)^2
pVarRel <- pVar
pVarRel <- (pVar/sum(pVarRel))*100
plot(pVarRel, type = "l",main = "Plot of PCA variance", xlab = "PCs", ylab = "Variance [%]",
ylim = c(0,50), xaxt = "n")
axis(side = 1, at = 1:10)
```



In this case the cutoff was at PC3 so the first three principal components were kept for the following steps.

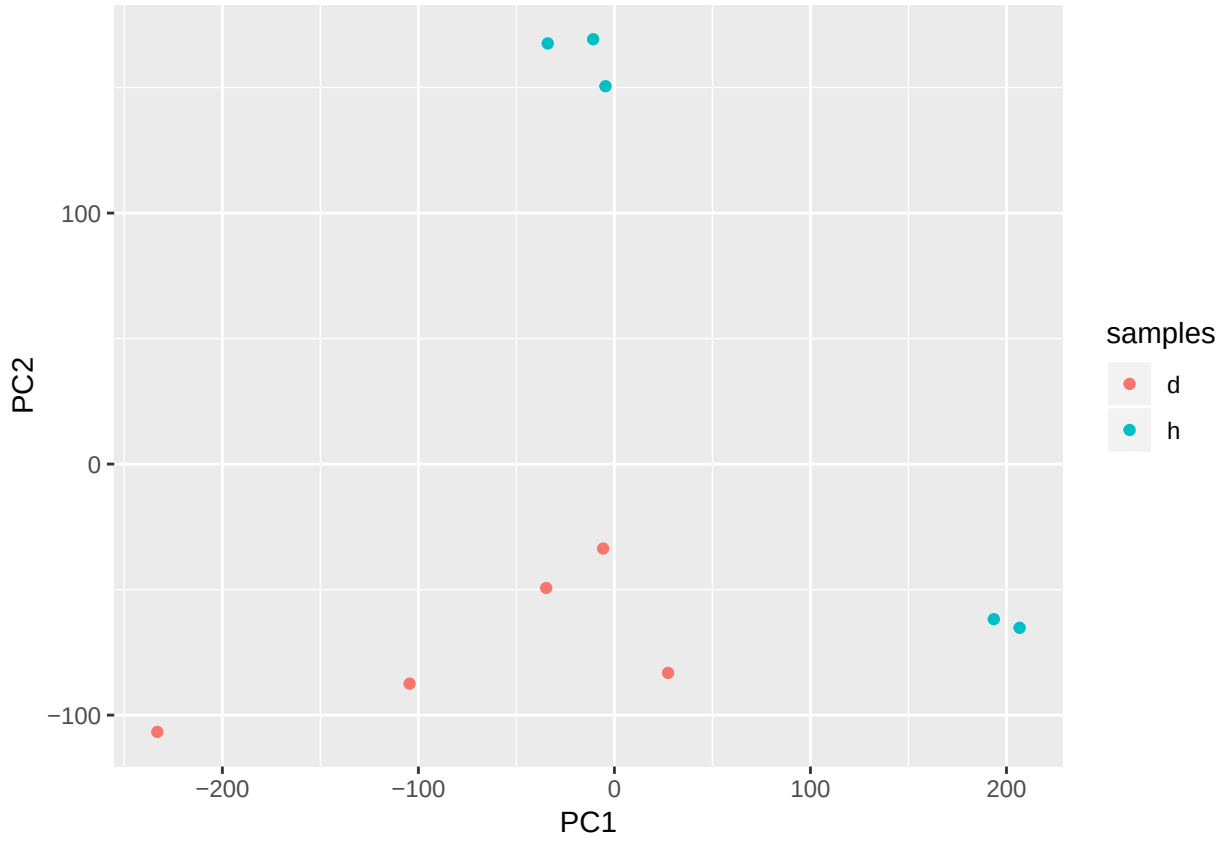
Visual testing for batch effects

The principal components need to be checked for batch effects in order to ensure the significance of the differential methylation. A batch effect would create correlations due to some samples being collected at once. If there is a correlation based on e.g. the sample provider this would result in a technical error since this correlation is not of biological significance. PC1-3 were tested for batch effects visually through plotting as well as statisticaly through several tests. The PCs did not show any significant batcheffect except for the tissue type since the control group contained tonsil and bone marrow samples. For this reason most analyses (e.g. t-test) were performed without the tonsil samples. Here extra columns were added to enable the plot to show the different possible batch effects at once.

```
samples <- c(rep("h",5),rep("d",5))
tissue <- c(rep("tonsil",3),rep("bone marrow",7))
provider <- c(rep("J.I.Martin-Subero",3),rep("J.Wiemels",2),rep("A.Bergmann",5))
date <- c("Aug2013","Mar2015","Aug2015","Aug2013","Mar2015","May2016","May2016",
rep("Jul2016",3))
```

Next a dataframe with the first two PCs is created in order to plot them against each other preventing possible computational errors due to the values not being numeric. The plot additionally shows the healthy (h) and the diseased (d) patients.

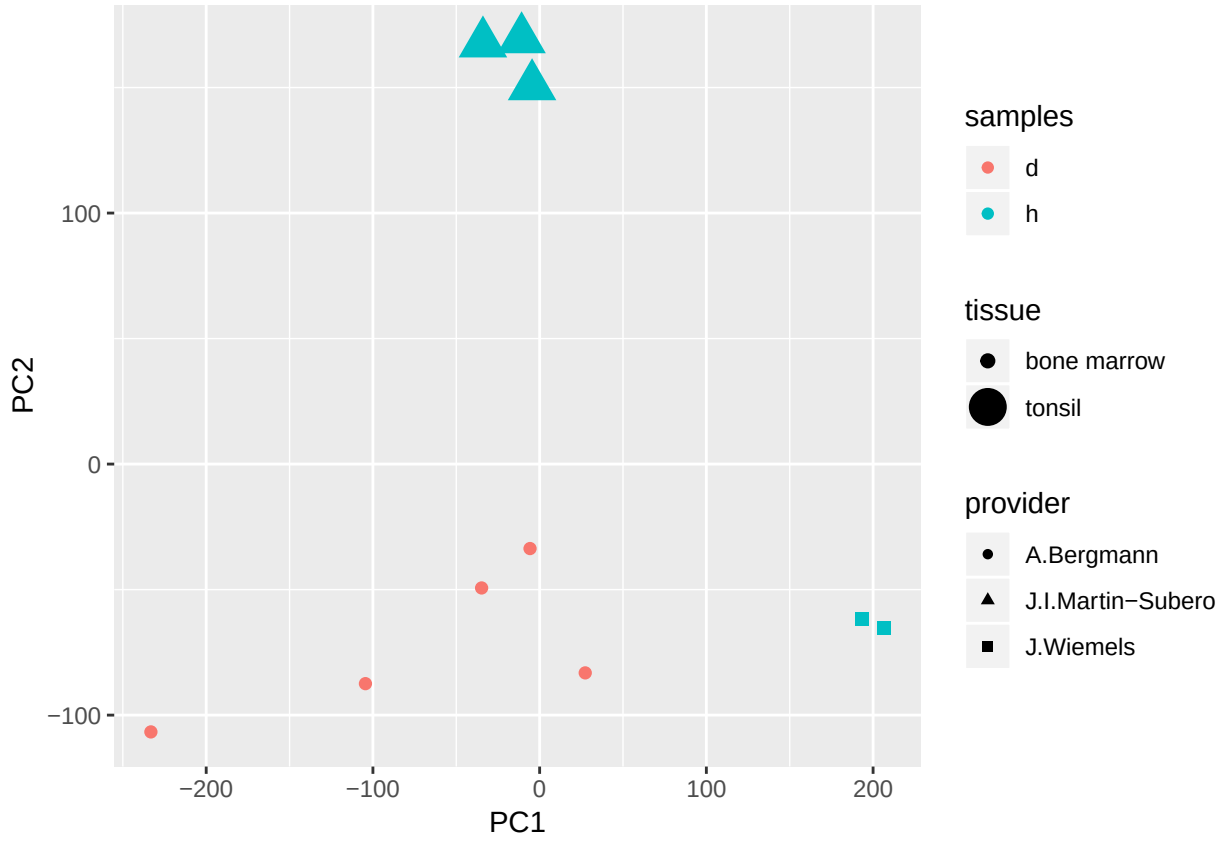
```
PC1 <- as.numeric(pca$x[,1])
PC2 <- as.numeric(pca$x[,2])
pcax12 <- cbind(PC1,PC2)
pcax12 <- data.frame(pcax12)
ggplot(pcax12, aes(x = PC1, y = PC2, colour = samples)) + geom_point()
```



Finally the plots showing if there are batch effects or not are created.

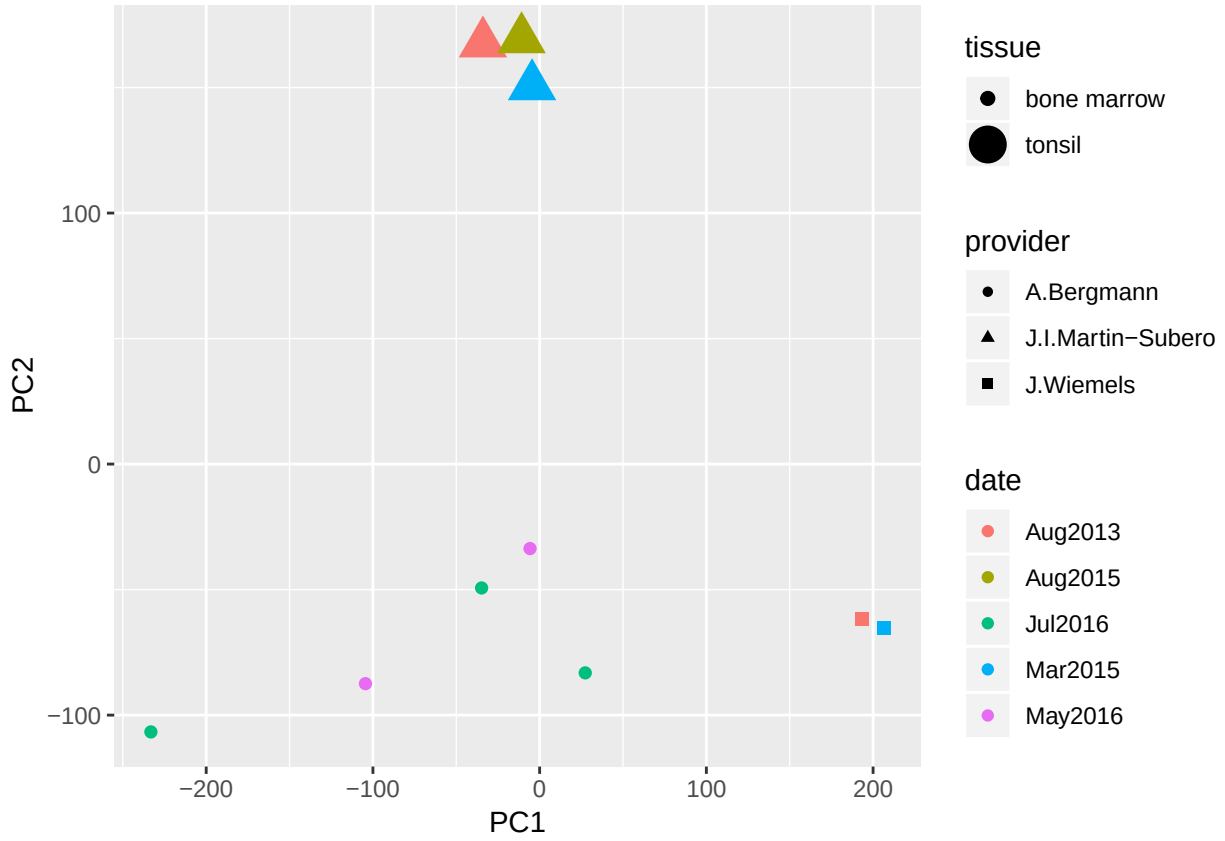
```
batcheffect1 <- ggplot(pcax12, aes(x = PC1, y = PC2, colour = samples, shape = provider,
size = tissue))
batcheffect1 + geom_point()
```

```
## Warning: Using size for a discrete variable is not advised.
```




```
batcheffect2 <- ggplot(pcax12, aes(x = PC1, y = PC2, colour = date, shape = provider,
size = tissue))
batcheffect2 + geom_point()
```

```
## Warning: Using size for a discrete variable is not advised.
```



Statistical testing for batch effects

In order to ascertain the absence of batch effects we used two different statistical tests: the Wilcoxon test and the Kruskal-Wallis test. Every single selected PC was reviewed for the respective batch effects. Using the Wilcoxon test the tissue type and disease status were investigated for batch effects.

```
sample_annotation <- read.csv("sample_annotation.csv")
  wilcoxon_p1 <- wilcox.test(pca$x[,1] ~ sample_annotation$TISSUE_TYPE)
  wilcoxon_p2 <- wilcox.test(pca$x[,2] ~ sample_annotation$TISSUE_TYPE)
  wilcoxon_p3 <- wilcox.test(pca$x[,3] ~ sample_annotation$TISSUE_TYPE)
  wilcoxon_p4 <- wilcox.test(pca$x[,1] ~ sample_annotation$DISEASE)
  wilcoxon_p5 <- wilcox.test(pca$x[,2] ~ sample_annotation$DISEASE)
  wilcoxon_p6 <- wilcox.test(pca$x[,3] ~ sample_annotation$DISEASE)
```

On the other hand the biomaterial provider and first submission date were analysed with the Kruskal-Wallis test. Additionally both investigated columns were converted to factors in order to achieve finite values preventing computational errors.

```
sample_annotation$BIOMATERIAL_PROVIDER <- as.factor(sample_annotation$BIOMATERIAL_PROVIDER)
  kruskal_p1 <- kruskal.test(pca$x[,1] ~ sample_annotation$BIOMATERIAL_PROVIDER)
  kruskal_p2 <- kruskal.test(pca$x[,2] ~ sample_annotation$BIOMATERIAL_PROVIDER)
  kruskal_p3 <- kruskal.test(pca$x[,3] ~ sample_annotation$BIOMATERIAL_PROVIDER)
sample_annotation$FIRST_SUBMISSION_DATE <- as.factor(sample_annotation$FIRST_SUBMISSION_DATE)
  kruskal_p4 <- kruskal.test(pca$x[,1] ~ sample_annotation$FIRST_SUBMISSION_DATE)
  kruskal_p5 <- kruskal.test(pca$x[,2] ~ sample_annotation$FIRST_SUBMISSION_DATE)
  kruskal_p6 <- kruskal.test(pca$x[,3] ~ sample_annotation$FIRST_SUBMISSION_DATE)
```

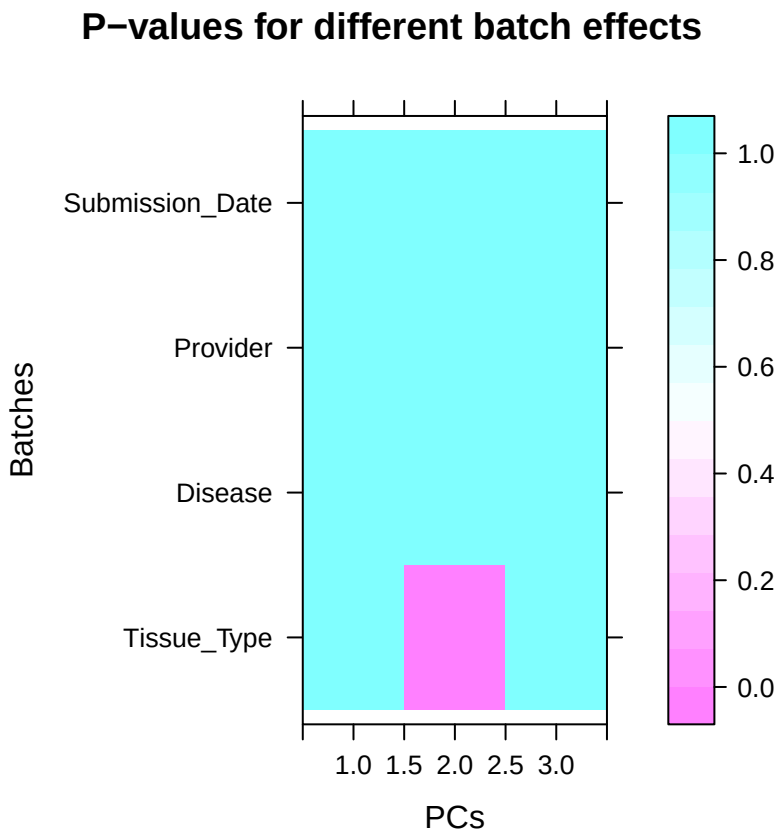
Visualisation of batch effect results

To visualize the results of the batch effect analysis a levelplot containing the p_values for each batch effect and PC is created. Any p-value above 0.05 defines the batch effect as not significant. If a PC therefore does not have any p-value below 0.05 it is free of batch effects.

```
Tissue_Type <- c(wilcoxon_p1$p.value,wilcoxon_p2$p.value,wilcoxon_p3$p.value)
Disease <- c(wilcoxon_p4$p.value,wilcoxon_p5$p.value,wilcoxon_p6$p.value)
Provider <- c(kruskal_p1$p.value,kruskal_p2$p.value,kruskal_p3$p.value)
Submission_Date <- c(kruskal_p4$p.value,kruskal_p5$p.value,kruskal_p6$p.value)
P_values <- rbind(Tissue_Type, Disease, Provider, Submission_Date)
library(lattice)
```

```
## Warning: package 'lattice' was built under R version 3.5.3
```

```
P_values_batch <- P_values
for (i in 1:nrow(P_values)) {
    for (j in 1:ncol(P_values)) {
        if(P_values[i,j] > 0.05){
            P_values_batch[i,j] <- 1
        }
        else{
            P_values_batch[i,j] <- 0
        }
    }
}
levelplot(t(P_values_batch), xlab = "PCs", ylab = "Batches",
main = "P-values for different batch effects")
```



When looking at the levelplot there is no p-value which is below 0.05 except for the disease row for PC1.

This proves that the PC1 does show a significant difference between healthy and diseased patients which is exactly as expected from a first principal component.

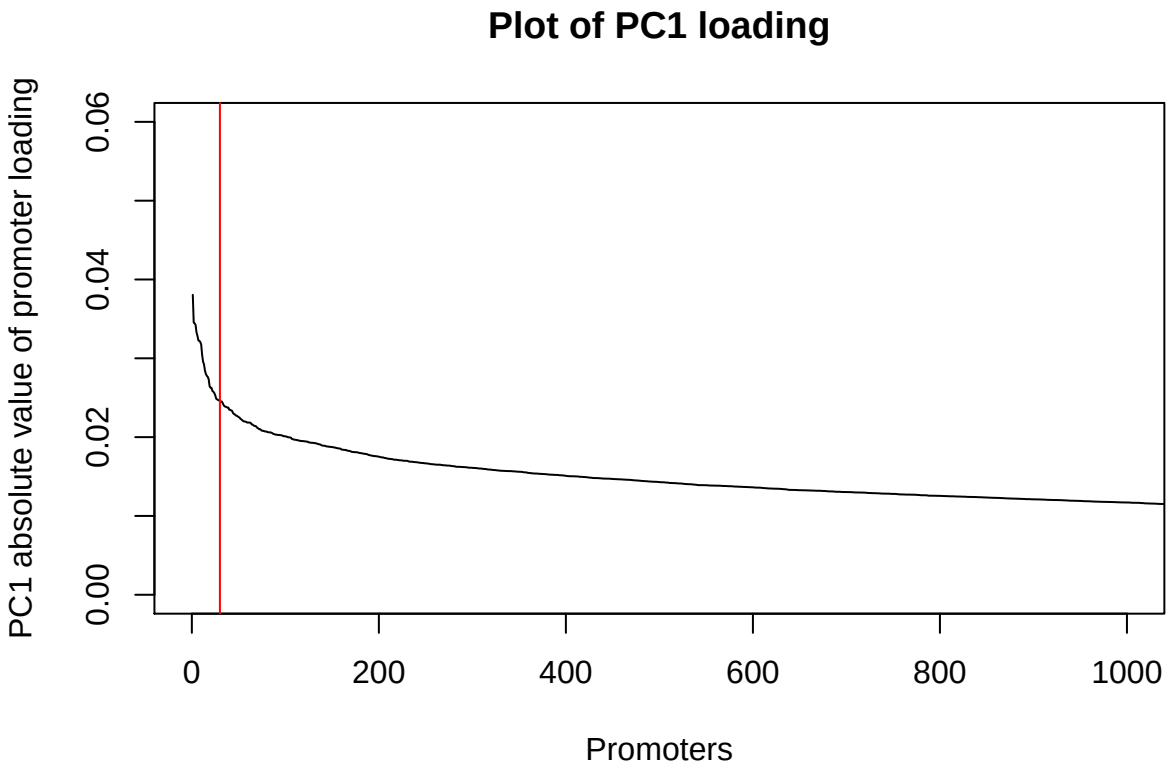
Selection of segregating PC

Next, a PC needs to be chosen for later analysis which segregates the two patient groups. Using the information of the last levelplot the first principal component is selected for further analysis.

Selection of relevant promoters

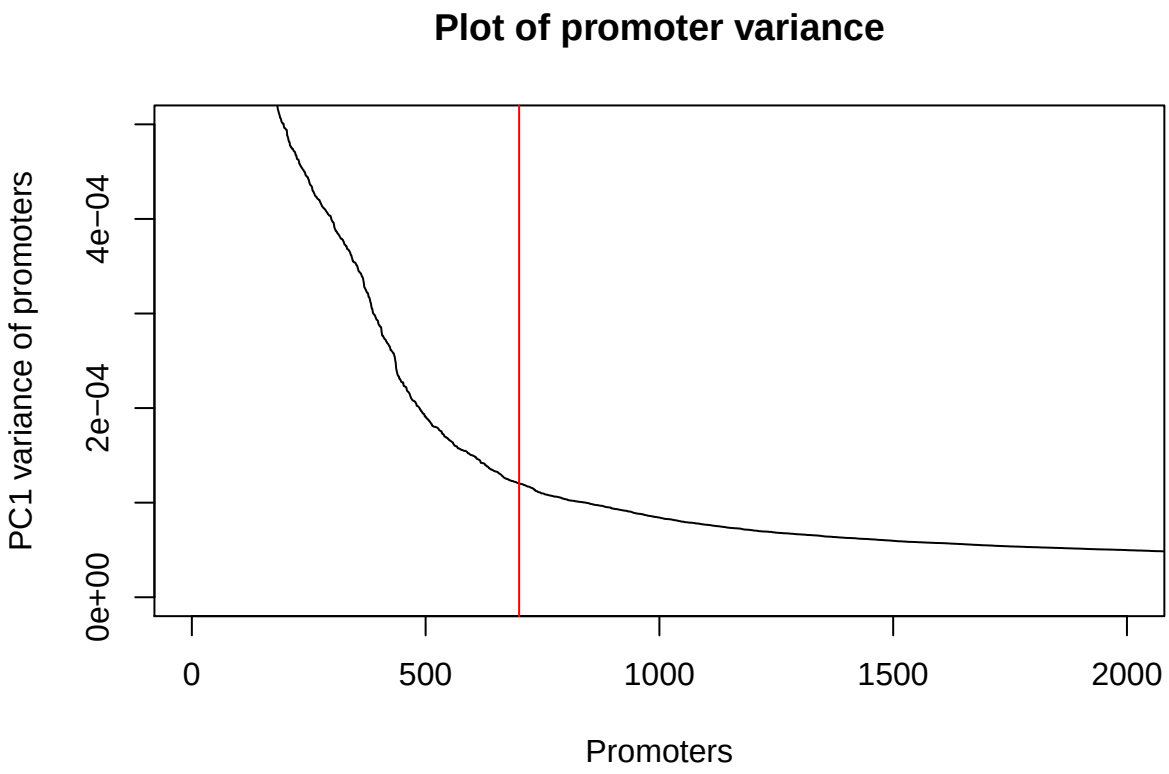
Next the most impotant promoters for that PC needed to be selected. Therefore the loading and variance of the promoters related to the first principal component were plotted. As a first test the absolute values of the loadings of the promoters were plotted and a cutoff was again chosen at the spot where an elbow appeared.

```
pcaRotAbs <- abs(pca$rotation[,1])
pcaRotAbssort <- sort(pcaRotAbs, decreasing = TRUE)
plot(pcaRotAbssort, type = "l", main = "Plot of PC1 loading", xlab = "Promoters",
ylab = "PC1 absolute value of promoter loading", ylim = c(0,0.06), xlim = c(0,1000))
abline(v=30, col = "red")
```



However the elbow was located at approximately 50 promoters which is a very low amount of data to use. Therefore the variance of each promoter was used in a second try since this value is also related to the differentially methylated regions. Next the values ware again sorted in a decreasing order and plotted.

```
pcaRotVar <- apply(pca$rotation, 1, var)
pcaRotVarsort <- sort(pcaRotVar, decreasing = TRUE)
plot(pcaRotVarsort, type = "l", main = "Plot of promoter variance", xlab = "Promoters",
ylab = "PC1 variance of promoters", ylim = c(0,0.0005), xlim = c(0,2000))
abline(v=700, col = "red")
```



The elbow for the variance was located at 700 promoters which represented a much wider base to support downstream analysis. Therefore these 700 relevant promoters were chosen and filtered. First the variances were ordered as to obtain the position of the relevant promoters. Then the promoters above the threshold of 700 were removed and the M-values were extracted to match the remaining 700 promoters.

```
pcaRotVar <- order(pcaRotVar, decreasing = TRUE)
pcaRotVar <- pcaRotVar[-c(701:length(pca$rotation))]
ALLMvalueRemain <- ALLMvalue[pcaRotVar,]
```

In order to incorporate the selected genes from the previous literature research the affiliated promoters were added to the 700 filtered promoters. This lead to a dataframe containing a total of 709 promoters which were the selected as well as the filtered promoters ready for further analysis.

```
ALLMGen <- ALLMvalue[c("ENSG00000039068","ENSG00000147889","ENSG00000147883",
"ENSG00000129451","ENSG00000050165","ENSG00000140945","ENSG00000103490","ENSG00000196730",
"ENSG00000185345"),]
ALLMvalueRemain <- rbind(ALLMvalueRemain,ALLMGen)
```

K-Means clustering

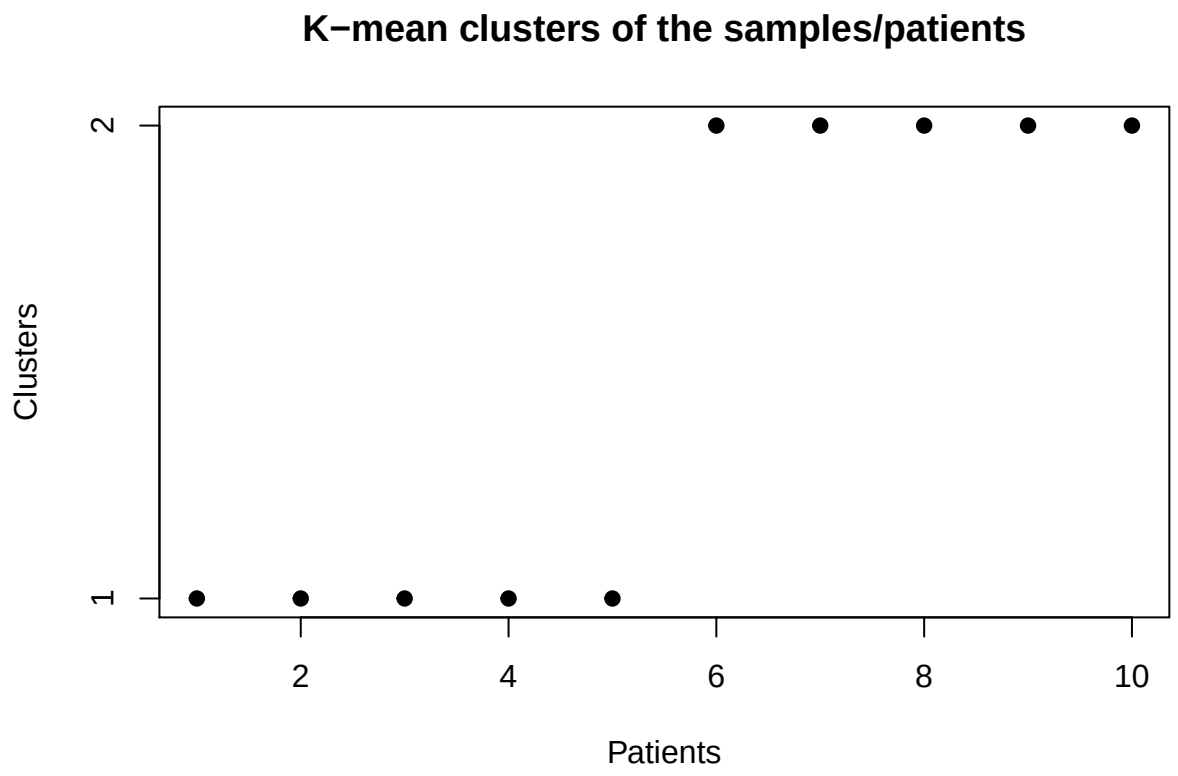
As an additional step k-means clustering was performed to ensure the segregation of the patients into their respective groups. The seperation was analysed through a silhouette plot.

```
M_km <- kmeans(t(x = ALLMvalueRemain), centers = 2, nstart = 10)
M_km$cluster
```

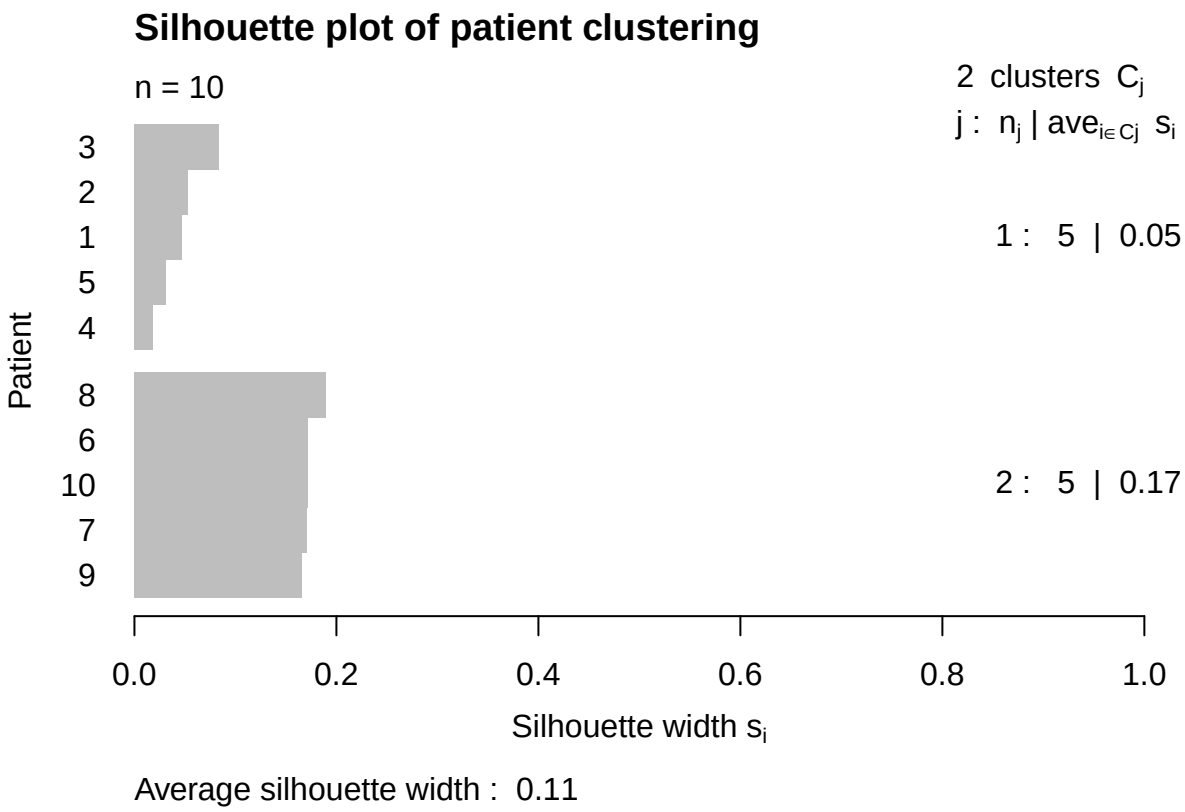
```
##         Bcell_gc_TO_G201.bed    Bcell_gc_TO_GC_T14_10.bed
##                            1                            1
##    Bcell_gc_TO_GC_T14_11.bed        Bcell_pre_BM_G200.bed
```

```
##                              1                              1
## Bcell_pre_BM_PreB2C.V152.bed    Bcell_pre_BM_S0179DA1.bed
##                              1                              2
##    Bcell_pre_BM_S017E3A1.bed    Bcell_pre_BM_S01GRFA1.bed
##                              2                              2
##    Bcell_pre_BM_S01GSDA1.bed    Bcell_pre_BM_S01GTBA1.bed
##                              2                              2
```

```
plot(M_km$cluster, type = "p", xlab = "Patients", ylab = "Clusters", yaxt = "n", pch = 19,
main = "K-mean clusters of the samples/patients")
axis(side = 2, at = 1:2)
```




```
D <- dist(t(ALLMvalueRemain))
library(cluster)
silh <- silhouette(M_km$cluster,D)
plot(silh, ylab = "Patient", main = "Silhouette plot of patient clustering")
```



The silhouette plot and k-means clustering data showed a good seperation of the patients into their respective health status. Now the data is ready to be analysed for differentially methylated regions via t-test.

Step3 - Testing for differentially methylated regions (DMR)

In order to find the promotors which are significantly differentially methylated the 709 filtered promoters are checked via t-test. This is the best method to find recurrent differences since it uses the mean of each patient group and therefore a differential methylation has to occur in most patients to be significant. If there is a DMR in just one patient it will not be recognized as recurrent and will therefore not be significant since the other patients of the group equalize its differentially methylated status through the mean.

Preparing for the t-test First the samples originating from the tonsil of the control group patients were removed from the analysis since none grouped with the samples derived from the bone marrow. Additionally all samples for the diseased group originated from the bone marrow which would lead to false results when comparing to a different tissue type. Then the M-values were splitted into their respective health groups healthy and diseased to allow for the t-test.

```
Mvalues_noTO <- ALLMvalueRemain[,c(4:10)]
Mvalues_healthy <- Mvalues_noTO[,c(1,2)]
Mvalues_disease <- Mvalues_noTO[,c(3:7)]
```

Next a vector was created which subsequently contained the p-values of the t-test.

```
t_test <- c(1:709)
```

Performing the t-test and adjusting the p-values

Then the actual t-test was performed and the resulting p-values were stored in the vector t_test for later use.

```
for (i in 1:length(t_test)) {
    t_test[i] <- t.test(Mvalues_healthy[i,],Mvalues_disease[i,])$p.value
}
```

In order to account for the statistical problem of multiple comparisons the p-values were adjusted using the Holm-Bonferroni method. The corrected values were then ordered and sorted in an increasing manner.

```
p_correction <- p.adjust(t_test, method = "holm", n = length(t_test))
p_correction_order <- order(p_correction)
p_correction_sort <- sort(p_correction)
```

Extraction and selection of significant promoters

Next the 20 most significant promoters were selected for in depth analysis e.g. its function and therefore the advantage for the cell. Additionally the Beta- and M-values of these selected promoters were extracted for later plotting.

```
t_test20 <- p_correction_order[c(1:20)]
ALLMvalueRemain20 <- ALLMvalueRemain[t_test20,]
ALLBetaRemain20 <- ALLpromotorBeta[t_test20,]
```

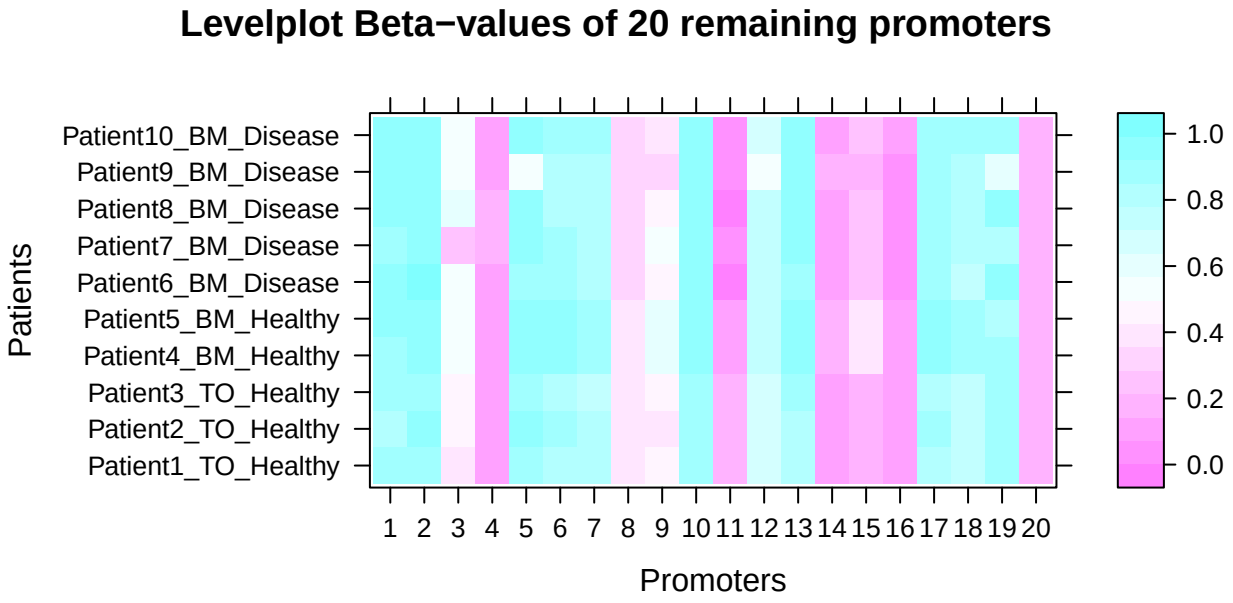
For the following logistical regression the significant promoters ($\alpha <= 0.05$) were calculated. Again a dataframe was created containing the corresponding Beta- and M-values.

```
threshold <- sum(p_correction_sort <= 0.05)
t_test_threshold <- p_correction_order[c(1:threshold)]
ALLMvalueRemain_threshold <- ALLMvalueRemain[t_test_threshold,]
ALLBetaRemain_threshold <- ALLpromotorBeta[t_test_threshold,]
```

Plotting Beta- and M-values for the selected (top 20) promoters

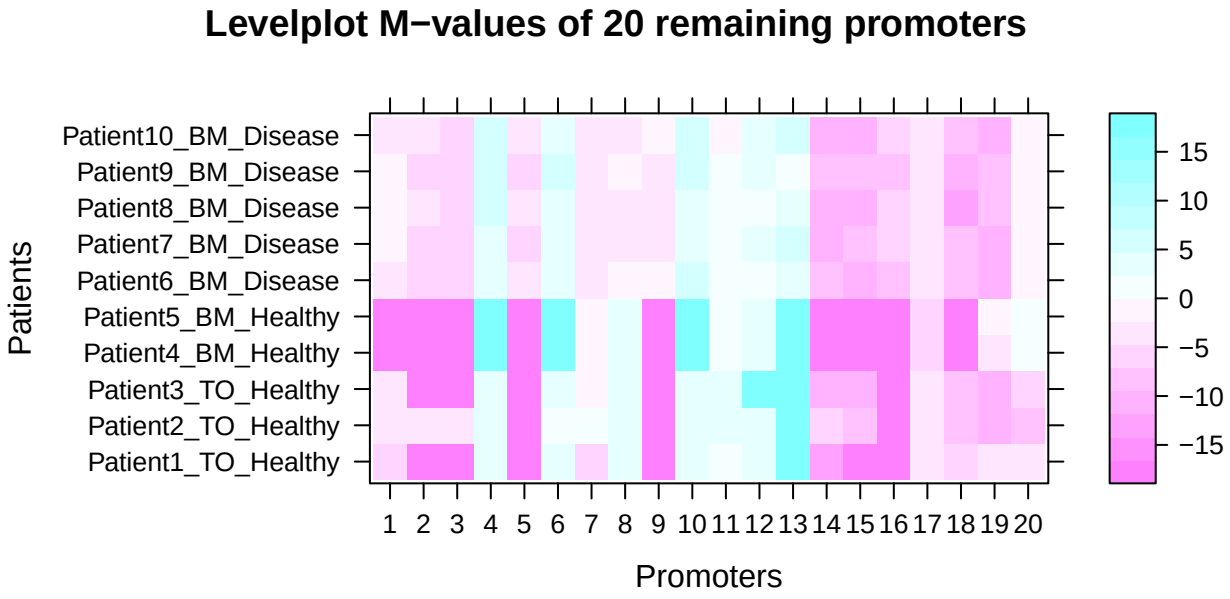
Next a levelplot was created to visualize the Beta-values of the 20 most significant promoters.

```
library(lattice)
ALLBetaRemain20_plot <- as.matrix(ALLBetaRemain20)
rownames(ALLBetaRemain20_plot) <- c(1:nrow(ALLBetaRemain20))
colnames(ALLBetaRemain20_plot) <-c("Patient1_TO_Healthy","Patient2_TO_Healthy",
"Patient3_TO_Healthy","Patient4_BM_Healthy","Patient5_BM_Healthy","Patient6_BM_Disease",
"Patient7_BM_Disease","Patient8_BM_Disease","Patient9_BM_Disease","Patient10_BM_Disease")
levelplot(ALLBetaRemain20_plot, xlab = "Promoters", ylab = "Patients",
main = "Levelplot Beta-values of 20 remaining promoters")
```



The same was done for the M-values. Both matrices were prepared beforehand to ensure no computational errors during plotting.

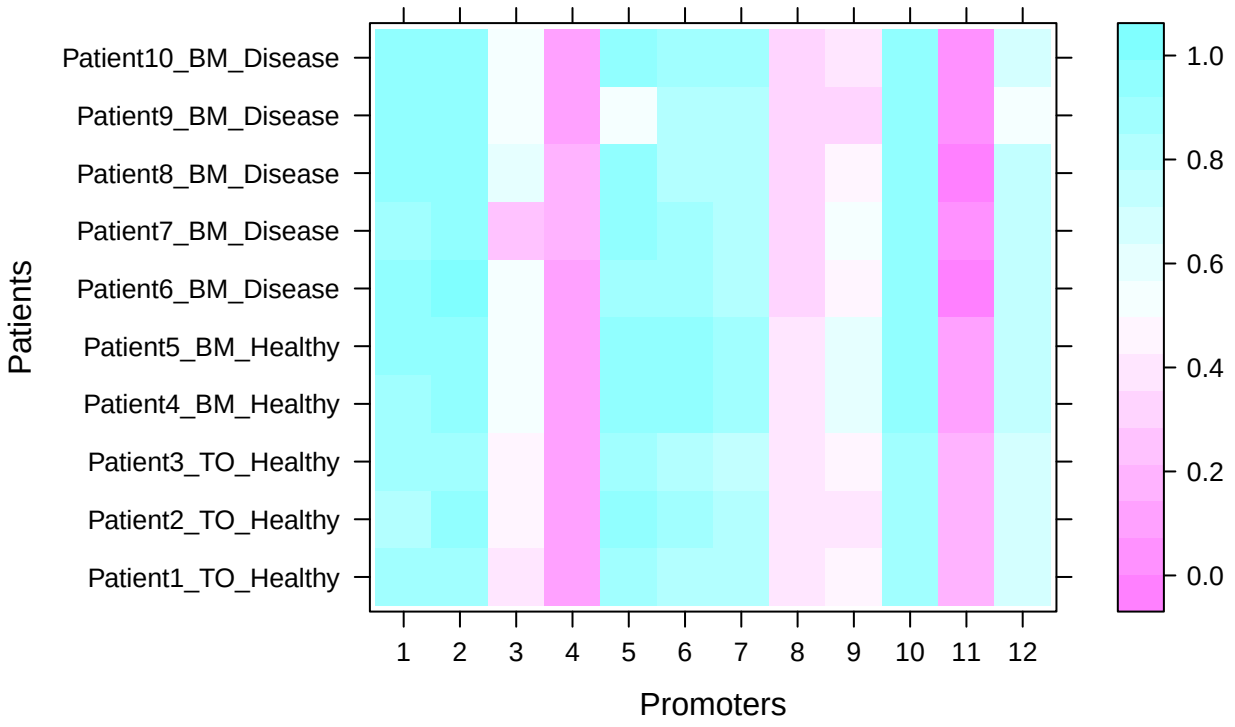
```
for (i in 1:20) {
    for(j in 1:10){
        ALLMvalueRemain20[i,j] <- as.numeric(ALLMvalueRemain20[i,j])
    }
}
ALLMvalueRemain20_plot <- as.matrix(ALLMvalueRemain20)
rownames(ALLMvalueRemain20_plot) <- c(1:nrow(ALLMvalueRemain20))
colnames(ALLMvalueRemain20_plot) <- c("Patient1_TO_Healthy","Patient2_TO_Healthy",
"Patient3_TO_Healthy","Patient4_BM_Healthy","Patient5_BM_Healthy","Patient6_BM_Disease",
"Patient7_BM_Disease","Patient8_BM_Disease","Patient9_BM_Disease","Patient10_BM_Disease")
levelplot(ALLMvalueRemain20_plot, xlab = "Promoters", ylab = "Patients",
main = "Levelplot M-values of 20 remaining promoters")
```



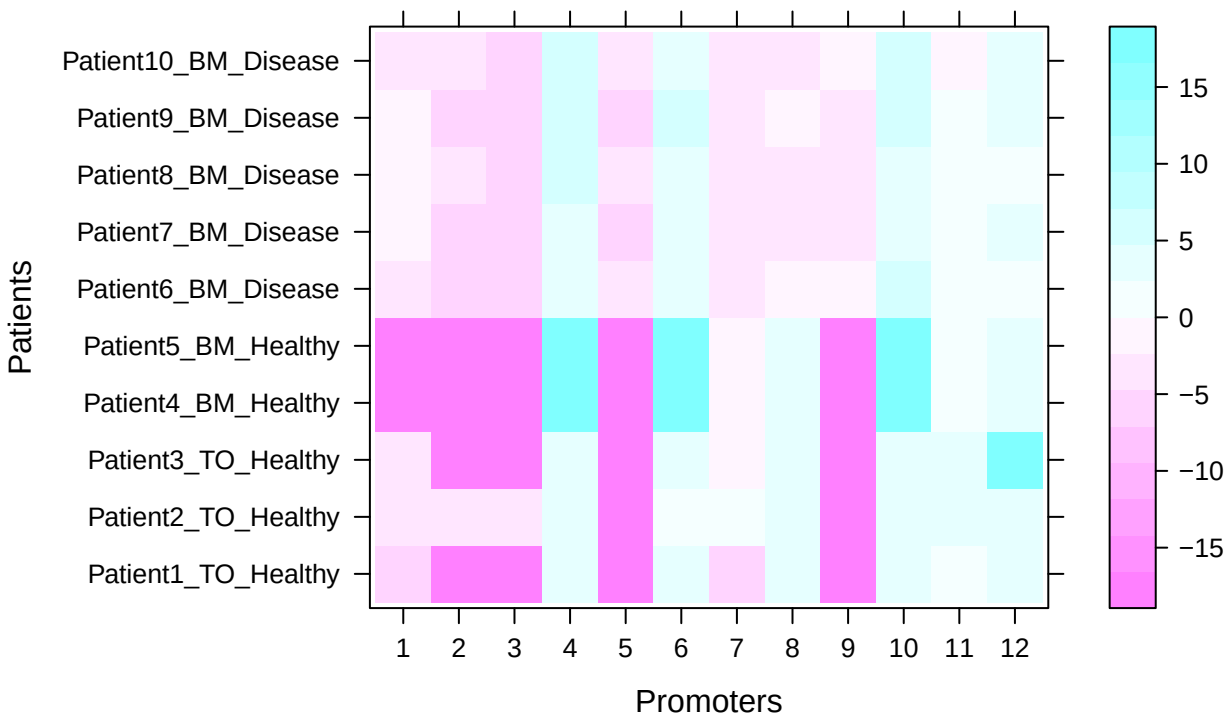
Plotting Beta- and M-values for the significant ($\alpha <= 0.05$) promoters

The same steps as above were repeated for the significant promoters ($\alpha <= 0.05$).

```
ALLBetaRemain_threshold_plot <- as.matrix(ALLBetaRemain_threshold)
rownames(ALLBetaRemain_threshold_plot) <- c(1:nrow(ALLBetaRemain_threshold))
colnames(ALLBetaRemain_threshold_plot) <- c("Patient1_TO_Healthy","Patient2_TO_Healthy",
"Patient3_TO_Healthy","Patient4_BM_Healthy","Patient5_BM_Healthy","Patient6_BM_Disease",
"Patient7_BM_Disease","Patient8_BM_Disease","Patient9_BM_Disease","Patient10_BM_Disease")
levelplot(ALLBetaRemain_threshold_plot, xlab = "Promoters", ylab = "Patients",
main = "Levelplot Beta-values of remaining promoters")
```



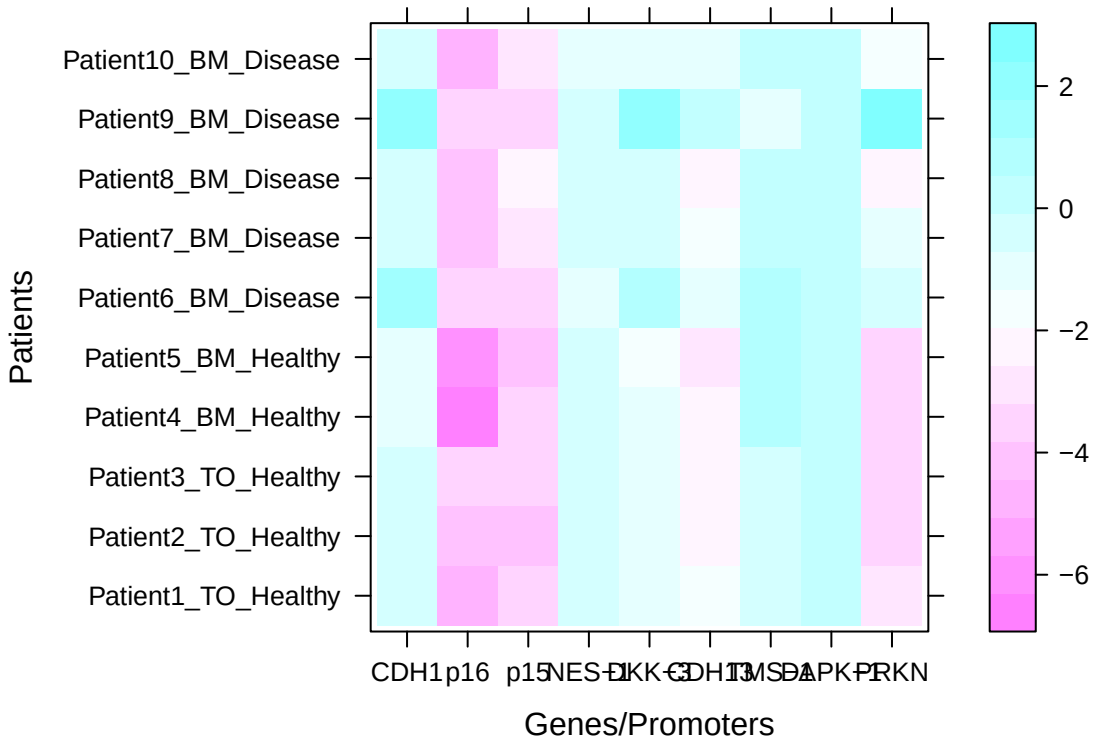

```
ALLMvalueRemain_threshold_plot <- as.matrix(ALLMvalueRemain_threshold)
rownames(ALLMvalueRemain_threshold_plot) <- c(1:nrow(ALLMvalueRemain_threshold))
colnames(ALLMvalueRemain_threshold_plot) <- c("Patient1_TO_Healthy","Patient2_TO_Healthy",
"Patient3_TO_Healthy","Patient4_BM_Healthy","Patient5_BM_Healthy","Patient6_BM_Disease",
"Patient7_BM_Disease","Patient8_BM_Disease","Patient9_BM_Disease","Patient10_BM_Disease")
levelplot(ALLMvalueRemain_threshold_plot, xlab = "Promoters", ylab = "Patients",
main = "Levelplot M-values of remaining promoters")
```



Plotting M-values of selected genes/promoters

Lastly a levelplot depicting the M-values of the selected genes/promoters from the literature research was created in order to analyse them separately.

```
ALLMGen_plot <- ALLMGen
rownames(ALLMGen_plot) <- c("CDH1","p16","p15","NES-1","DKK-3","CDH13","TMS-1","DAPK-1","PRKN")
colnames(ALLMGen_plot) <- c("Patient1_TO_Healthy","Patient2_TO_Healthy",
"Patient3_TO_Healthy","Patient4_BM_Healthy","Patient5_BM_Healthy","Patient6_BM_Disease",
"Patient7_BM_Disease","Patient8_BM_Disease","Patient9_BM_Disease","Patient10_BM_Disease")
levelplot(as.matrix(ALLMGen_plot), xlab = "Genes/Promoters", ylab = "Patients",
main = "Levelplot M-values of selected genes/promoters")
```



Step4 - Logistical regression

As a final step a model which could predict the cancer status of a patient using logistical regression was created. Logistical regression was chosen over linear regression since the latter is much less accurate when it comes to binary events such as methylation. Therefore a logistically regressed model would achieve better results predicting a patients disease status using methylation data. Using the previously defined 12 significant genes a dataframe was created which also defined the health status of the patients data used to create the model. Diseased patients were defined as "1" and healthy patients as "0". Therefore upon prediction a value higher than 0.5 would indicate a cancerous state versus a healthy state if the predicted value is lower than 0.5. First the dataframe was prepared and the additional column with the health status was added.

```
Lg_Mvalues <- data.frame(t(ALLMvalueRemain_threshold))
Tumor <- factor(c(rep("0",5),rep("1",5)))
Lg_Mvalues <- cbind(Lg_Mvalues,Tumor)
```

Next a matrix was created in which the predicted values were stored. Then each of the significant genes was used for logistical regression and prediction of health status. Once all prediction values were stored in the matrix the first column used to create the matrix was removed.

```
LGpred <- as.matrix(c(1:10))
for (i in 1:nrow(ALLMvalueRemain_threshold)) {
     glm99 <- glm(Tumor ~ Lg_Mvalues[,i], family = "binomial", data = Lg_Mvalues)
     pred <- as.matrix(predict(glm99, type = "response"))
     LGpred <- cbind(LGpred,pred)
}
```

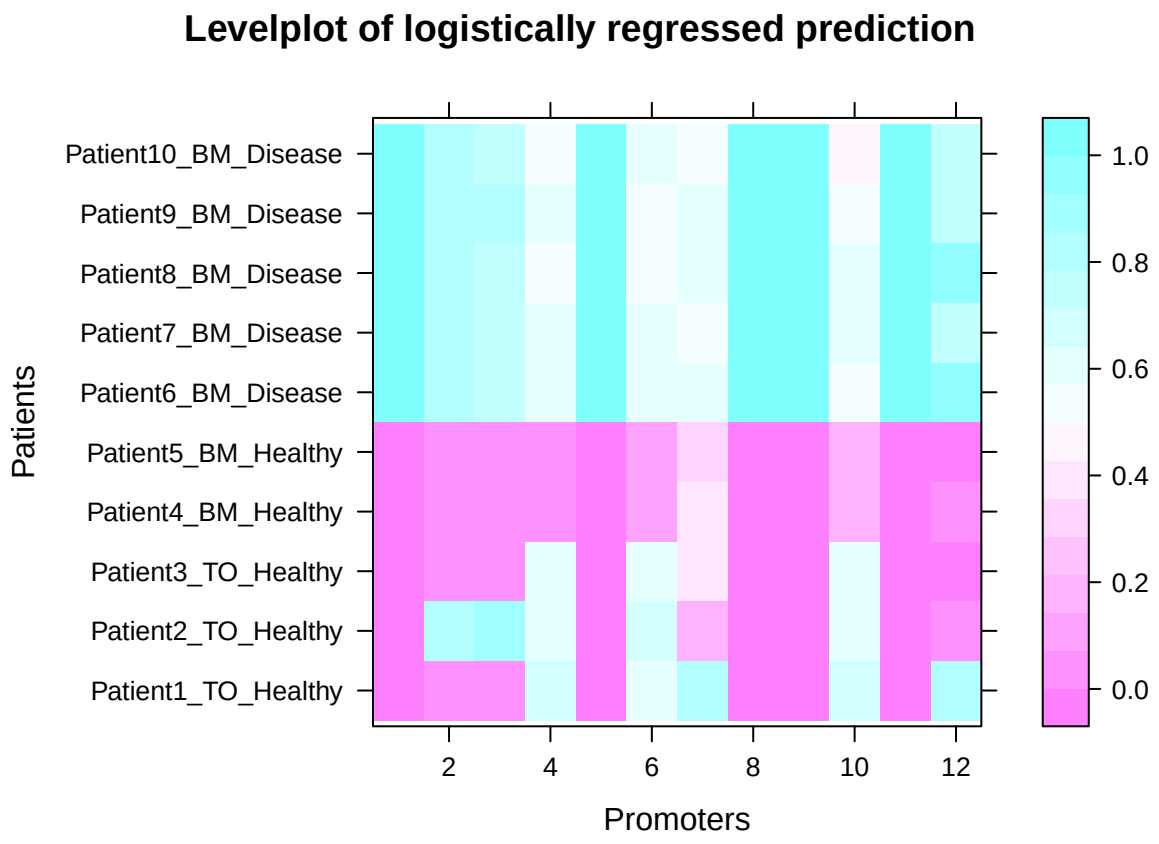
```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
LGpred <- LGpred[,-1]
```

Finally the predicted values were plotted via a levelplot to check for the accuracy of the model.

```
rownames(LGpred) <- c("Patient1_TO_Healthy","Patient2_TO_Healthy","Patient3_TO_Healthy",
"Patient4_BM_Healthy","Patient5_BM_Healthy","Patient6_BM_Disease","Patient7_BM_Disease",
"Patient8_BM_Disease","Patient9_BM_Disease","Patient10_BM_Disease")
levelplot(t(LGpred), xlab = "Promoters",ylab = "Patients",
main = "Levelplot of logistically regressed prediction")
```



Unfortunately the model was not very accurate in predicting the health status of the patients.

Results?

wip?

Conclusion

wip