R Notebook for the analysis of Illumina Infinium DNA methylation arrays dataset

Alessandro Caula

Master's degree in Bioinformatics, University of Bologna (Italy)

Step 1: Import raw data with minfi and create the RGset object in which store the RGChannlSet object

Before even starting is important to remove all the objects already present in the workspace and set the working directory in which save and load the data.

```
rm(list=ls())
setwd("/media/alessandro/DATA/User/BIOINFORMATICA.BOLOGNA/DRD/Project")
```

Then suppress and therefore ignore all the simple diagnostic message from minfi package that we have previously installled from Bioconductor and that we will use in our analysis. Minfi is a flexible tool used for the analysis and visualizing Illumina's methylation array data.

```
suppressMessages(library(minfi))
```

Import the Sample sheet of the experiment with the *read.metharray.sheet()* function and load the *csv* file.

```
baseDir = ("Input_data")
targets = read.metharray.sheet(baseDir)
```

```
## [1] "Input_data/Samplesheet_report_2020.csv"
```

```
Samplesheet_report_2020=read.csv("Input_data/Samplesheet_report_2020.csv",header=T,stringsAsFactors=T)
head(Samplesheet_report_2020)
```

	Sample_Name	Group	Age	Slide	Array	Basename
	<int>	<fctr>	<int>	<dbl>	<fctr>	<fctr>
1	1020	DS	29	5775278051	R01C01	5775278051_R01C01
2	1036	DS	34	5775278051	R04C02	5775278051_R04C02
3	3038	WT	46	5775278078	R02C01	5775278078_R02C01
4	3042	WT	32	5775278078	R05C01	5775278078_R05C01
5	3052	WT	31	5775278078	R05C02	5775278078_R05C02
6	1016	DS	43	5930514034	R01C02	5930514034_R01C02

6 rows

Now I will save the raw experimental data relative to the sample sheet into the RGset object using the minfi *read.methyarray.exp()* function.

```
RGset=read.metharray.exp(targets=targets)
save(RGset,file = "RGset.RData")
dim(RGset)
```

```
## [1] 622399      8
```

Step2: Create the dataframes Red and Green to store the red and green fluorescences.

I will create both the dataframes using the *getRed()* and *getGreen()* minfi functions that will extract the data of the respective color channels from the RGset object.

```
Red = data.frame(getRed(RGset))
Green = data.frame(getGreen(RGset))
```

Step3: Fill the table with the red and green fluorescence for the address: 6176046. Check in the manifest if the address correspond to a type I or II probe.

Checking the intensities of the red channel for address 6176046.

```
Red[rownames(Red)=="61760464",]
```

```
##          X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01
## 61760464               7789              10381               3146
##          X5775278078_R05C01 X5775278078_R05C02 X5930514034_R01C02
## 61760464               9422               7752               4596
##          X5930514035_R04C02 X5930514035_R06C02
## 61760464              11469               9939
```

```
Green[rownames(Green)=="61760464",]
```

```
##          X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01
## 61760464               6270               7860               7273
##          X5775278078_R05C01 X5775278078_R05C02 X5930514034_R01C02
## 61760464               5913               5817               6298
##          X5930514035_R04C02 X5930514035_R06C02
## 61760464               5687               6963
```

Now I will load the Illumina450Manifest_clean which contains all the information on the probes and check the probe Type that is found in the 7th column, for our address (61760464).

```
load("Illumina450Manifest_clean.RData")
Illumina450Manifest_clean[Illumina450Manifest_clean$AddressA_ID=="61760464",7]
```

```
## [1] II
## Levels: I II
```

This addrees belongs a probe of type II, therefore no color needs to be speciefied for this probe. Now that I have retrieved these information I can fill the following table:

Sample	Red fluorescence	Green fluorescence	Type	Color
X5775278051_R01C01	7789	6270	II	/
X5775278051_R04C02	10381	7860	II	/
X5775278078_R02C01	3146	7273	II	/
X5775278078_R05C01	9422	5913	II	/
X5775278078_R05C02	7752	5817	II	/
X5930514034_R01C02	4596	6298	II	/
X5930514035_R04C02	11469	5687	II	/
X5930514035_R06C02	9939	6963	II	/

Step4: Create the object MSet.raw

I want to extract the Methylated and Unmethylated signals. In order to do this I will use the *preprocessRaw()* function on the RGset object and create the MSet.raw. This object will contain beta and M values for the dataset.

```
MSet.raw=preprocessRaw(RGset)
dim(MSet.raw)
```

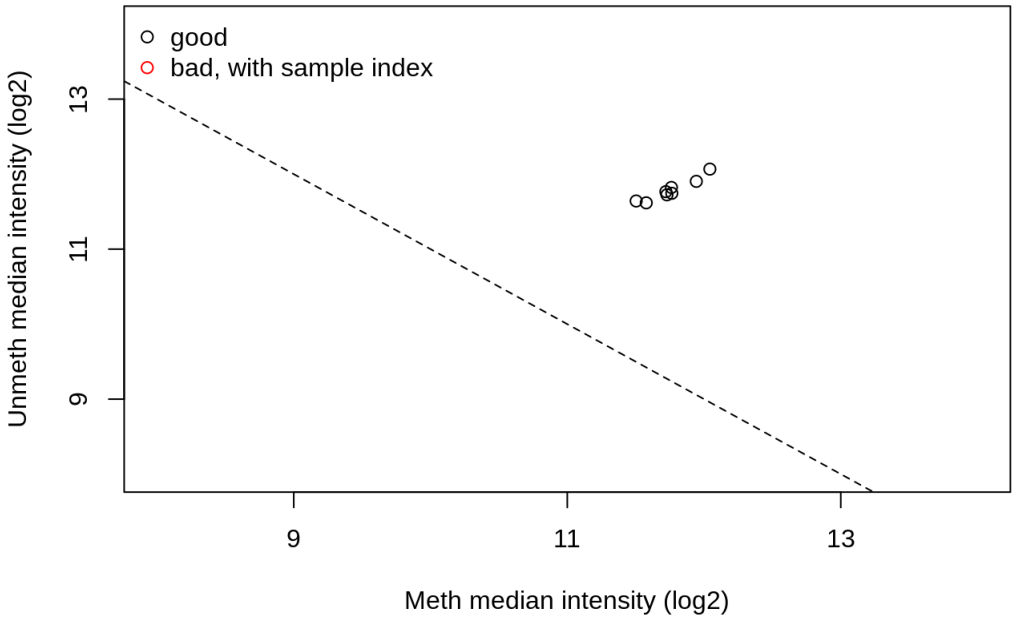
```
## [1] 485512      8
```

Step5: Perform the following quality checks and provide a brief comment to each step:

QCplot

I will use the function *getQC()* that get the Methylated and Unmethylatde intensities for each sample from the MSet.raw object and then plot them with the *plotQC()* function.

```
qc=getQC(MSet.raw)
plotQC(qc)
```

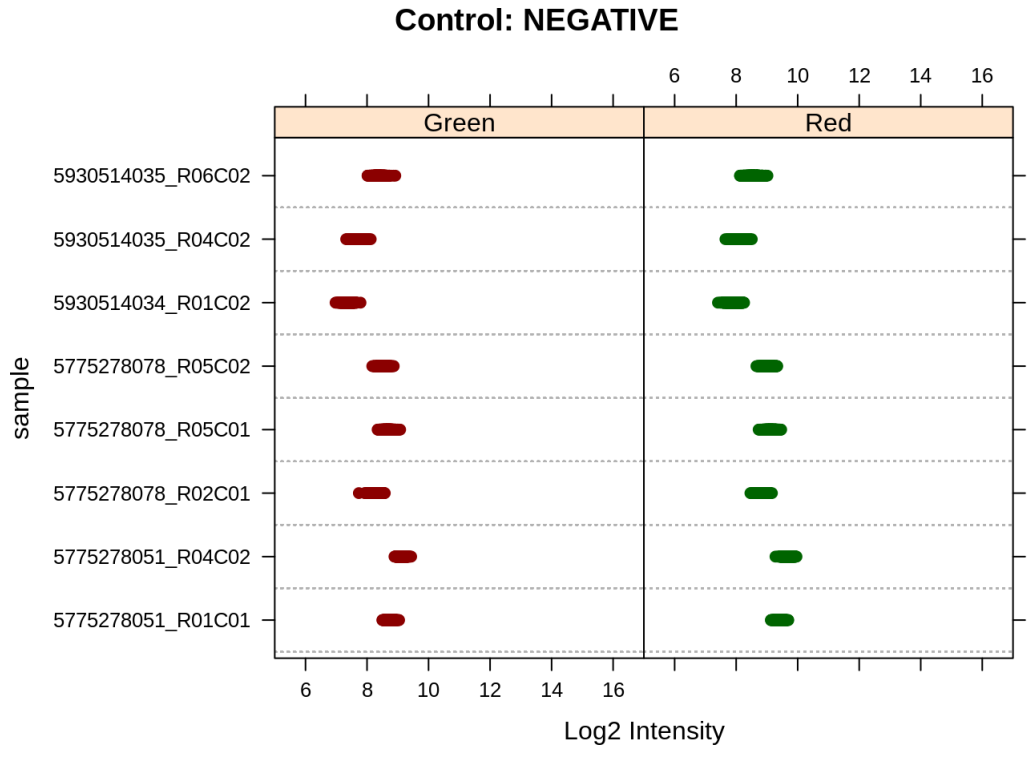


All the eight samples have an high median methylated and unmethylated signals, displaying in this way the good quality of all of them.

Check the insensities of negative control using minfi

Taking advantage of the *controlStripPlot()* function it's possible to plot and evaluate the intensity values of the negative control probes in our dataset.

```
controlStripPlot(RGset, controls="NEGATIVE")
```



N.B. During the lessons we noticed that the colors of the dots in the plot are inverted (red dots should be in green and vice-versa)

Anyway we can observe that the negative controls for both red and green fluorescences are fine since they are below 1000 (*log2(1000)=10*).

Calculate detection pValues; for each sample, how many probes have a detection p-value higher than the threshold of 0.05.

Using the *detectionP()* function all the failed positions defined as both the methylated and unmethylated channels are identified, reporting the background signals levels.

The detection of the p-value is computed from the background model and returned for every genomic positions in every sample.

```
detp=detectionP(RGset)
```

Now we can see from the *detp* object how many probes have a detection p-value higher than the 0.05 threshold.

```
failed=detp>0.05
summary(failed)
```

```
##  5775278051_R01C01 5775278051_R04C02 5775278078_R02C01 5775278078_R05C01
##  Mode :logical     Mode :logical     Mode :logical     Mode :logical
##  FALSE:485265      FALSE:485302      FALSE:485248      FALSE:485099
##  TRUE :247         TRUE :210         TRUE :264         TRUE :413
##  5775278078_R05C02 5930514034_R01C02 5930514035_R04C02 5930514035_R06C02
##  Mode :logical     Mode :logical     Mode :logical     Mode :logical
##  FALSE:485127      FALSE:485421      FALSE:485466      FALSE:485397
##  TRUE :385         TRUE :91          TRUE :46          TRUE :115
```

From the summary visualization of the results, it's possible to see the number of TRUE (their p-value is greater that 0.05, failing the statistics) and FALSE (their p-value is less than the threshold, they don't fail the statistics). The table below summarizes the number of failed positions according to each sample.

Sample	Failed positions
5775278051_R01C01	247
5775278051_R04C02	210
5775278078_R02C01	264
5775278078_R05C01	413
5775278078_R05C02	385
5930514034_R01C02	91
5930514035_R04C02	46
5930514035_R06C02	115

Step6 Calculate raw beta and Mvalues and plot the densities of mean methylation values, dividing the samples in DS and WT

I will first extract the beta values from the *MSet.raw* using the *getBeta()* function.

```
beta=getBeta(MSet.raw)
summary(beta)
```

```
##  5775278051_R01C01 5775278051_R04C02 5775278078_R02C01 5775278078_R05C01
##  Min.   :0.01745   Min.   :0.01828   Min.   :0.01128   Min.   :0.01133
##  1st Qu.:0.09198   1st Qu.:0.09763   1st Qu.:0.08523   1st Qu.:0.09360
##  Median :0.60089   Median :0.60543   Median :0.60102   Median :0.60714
##  Mean   :0.47988   Mean   :0.48371   Mean   :0.48459   Mean   :0.49043
##  3rd Qu.:0.79643   3rd Qu.:0.80112   3rd Qu.:0.81373   3rd Qu.:0.81985
##  Max.   :1.00000   Max.   :0.98415   Max.   :1.00000   Max.   :0.98884
##  NA's   :1         NA's   :2         NA's   :3         NA's   :1
##  5775278078_R05C02 5930514034_R01C02 5930514035_R04C02 5930514035_R06C02
##  Min.   :0.01178   Min.   :0.00000   Min.   :0.00000   Min.   :0.008389
##  1st Qu.:0.09452   1st Qu.:0.06721   1st Qu.:0.07456   1st Qu.:0.080286
##  Median :0.60643   Median :0.58693   Median :0.61593   Median :0.616594
##  Mean   :0.49042   Mean   :0.47988   Mean   :0.49289   Mean   :0.494334
##  3rd Qu.:0.81816   3rd Qu.:0.82893   3rd Qu.:0.84495   3rd Qu.:0.843440
##  Max.   :0.98877   Max.   :1.00000   Max.   :1.00000   Max.   :1.000000
##  NA's   :1         NA's   :10        NA's   :7         NA's   :4
```

Then I extract the M values using the *getM()* function.

```
M=getM(MSet.raw)
summary(M)
```

```
##  5775278051_R01C01 5775278051_R04C02 5775278078_R02C01 5775278078_R05C01
##  Min.   :-5.8153   Min.   :-5.7467   Min.   :-6.4535   Min.   :-6.4468
##  1st Qu.:-3.3034   1st Qu.:-3.2084   1st Qu.:-3.4241   1st Qu.:-3.2756
##  Median : 0.5903   Median : 0.6177   Median : 0.5911   Median : 0.6280
##  Mean   :    Inf   Mean   :-0.3158   Mean   :    Inf   Mean   :-0.2778
##  3rd Qu.: 1.9680   3rd Qu.: 2.0101   3rd Qu.: 2.1271   3rd Qu.: 2.1861
##  Max.   :    Inf   Max.   : 5.9560   Max.   :    Inf   Max.   : 6.4698
##  NA's   :1         NA's   :2         NA's   :3         NA's   :1
##  5775278078_R05C02 5930514034_R01C02 5930514035_R04C02 5930514035_R06C02
##  Min.   :-6.3903   Min.   :   -Inf   Min.   :   -Inf   Min.   :-6.8851
##  1st Qu.:-3.2600   1st Qu.:-3.7947   1st Qu.:-3.6337   1st Qu.:-3.5180
##  Median : 0.6237   Median : 0.5068   Median : 0.6814   Median : 0.6854
##  Mean   :-0.2818   Mean   :    NaN   Mean   :    NaN   Mean   :    Inf
##  3rd Qu.: 2.1697   3rd Qu.: 2.2767   3rd Qu.: 2.4462   3rd Qu.: 2.4296
##  Max.   : 6.4600   Max.   :    Inf   Max.   :    Inf   Max.   :    Inf
##  NA's   :1         NA's   :10        NA's   :7         NA's   :4
```

For the next step using the previously imported *Sample_sheet*, I will generate the DS and WT vectors containing the array names according to their group of samples.

```
DS=Samplesheet_report_2020[Samplesheet_report_2020$Group=="DS",]$Basename
WT=Samplesheet_report_2020[Samplesheet_report_2020$Group=="WT",]$Basename
```

Using the just gained vectors I extract the beta values according to the DS and WT groups:

```
DSbeta=beta[,colnames(beta) %in% DS]
WTbeta=beta[,colnames(beta) %in% WT]
```

And repeat the same procedure for the M values:

```
DSM=M[,colnames(M) %in% DS]
WTM=M[,colnames(M) %in% WT]
```

Now that the two subset are divided, I can plot the density distribution of mean beta and M values for each CpG Island.

First I need to calculate the means of beta and M values for each probe in the two sample groups (DS and WT), using the *mean()* function and *apply()* it to each row of the subset matrices, discarding NA values with the parameter **na.rm=T**.

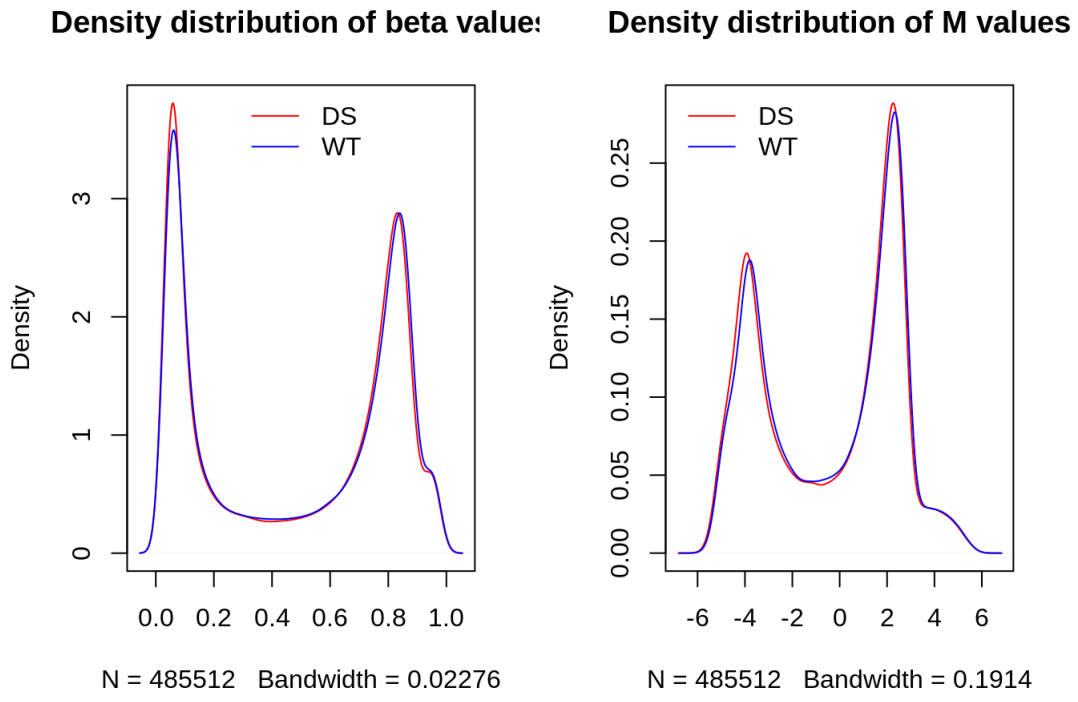
```
mean_of_DSbeta=apply(DSbeta,1,mean,na.rm=T)
mean_of_WTbeta=apply(WTbeta,1,mean,na.rm=T)
mean_of_DSM=apply(DSM,1,mean,na.rm=T)
mean_of_WTM=apply(WTM,1,mean,na.rm=T)
```

I can now calculate the density distribution using the *density()* function and then plot them.

```
d_mean_of_DSbeta=density(mean_of_DSbeta)
d_mean_of_WTbeta=density(mean_of_WTbeta)
d_mean_of_DSM=density(mean_of_DSM)
d_mean_of_WTM=density(mean_of_WTM)
```

Finally plot the density distributions of beta and M values for the DS and WT groups.

```
{
par(mfrow=c(1,2))
plot(d_mean_of_DSbeta,main="Density distribution of beta values",col="red")
lines(d_mean_of_WTbeta,col="blue")
legend("top",legend=c("DS","WT"),col=c("red","blue"),lty=1, bty="n")
plot(d_mean_of_DSM,main="Density distribution of M values",col="red")
lines(d_mean_of_WTM,col="blue")
legend("topleft", legend=c("DS","WT"), col=c("red","blue"), lty=1,bty="n")
}
```



From both the plots it's easily seen how the beta and M values for the beta and M values tend to have a very similar density distribution, even if there are some very small and negligible misalignments between the two curves.

Step7: Normalize the data using the *preprocessSWAN* function and compare raw data and normalized data. Produce a plot with 6 panels in which, for both raw and normalized data, you show the density plots of beta mean values according to the chemistry of the probes, the density plot of beta standard deviation values according to the chemistry of the probes and the boxplot of beta values.

I have to subset the dataframe of beta value according to their array chemistry, Type I and Type II. To do this I will first identify in the Illumina450Manifest_clean the Type I and Type II probes and generate the two new matrices one for the Type I probes and one other for Type II probes.

```
dfI=Illumina450Manifest_clean[Illumina450Manifest_clean$Infinium_Design_Type=="I",]
dfI=droplevels(dfI)
dfII=Illumina450Manifest_clean[Illumina450Manifest_clean$Infinium_Design_Type=="II",]
dfII=droplevels(dfII)
```

Now I can subset the beta matrix gained in the previous step, according to the probes name present in the just created dfI (TypeI probes) and dfII (TypeII probes) matrices. The names of the probes in the matix of the beta values are in the rowname.

```
beta_I=beta[rownames(beta) %in% dfI$IlmnID,]
dim(beta_I)
```

```
## [1] 135476      8
```

```
beta_II=beta[rownames(beta) %in% dfII$IlmnID,]
dim(beta_II)
```

```
## [1] 350036      8
```

I then proceed to calculate the mean of beta values accross the 8 samples in both these two subsets.

```
mean_of_betaI=apply(beta_I,1,mean)
mean_of_betaII=apply(beta_II,1,mean)
```

And then I calculate the density distribution of the vectors of mean values.

```
d_mean_of_betaI=density(mean_of_betaI,na.rm=T)
d_mean_of_betaII=density(mean_of_betaII,na.rm=T)
```

Using the *sd()* function I can compute the standard deviations of the subsets and then calculate the density distribution of the standard deviations.

```
sd_beta_I=apply(beta_I,1,sd,na.rm=T)
sd_beta_II=apply(beta_II,1,sd,na.rm=T)

d_sd_betaI=density(sd_beta_I)
d_sd_betaII=density(sd_beta_II)
```

Next step will be to normalize all the raw density from the RGset object using the *preprocessSWAN()* function that is a whithin-array normalisation method for the Illumina Infinium HumanMethylation450 platform.

```
preproSWAN=preprocessSWAN(RGset)
```

I can proceed now to extract the beta values using the *getBeta()* fuction, separate the Type I and Type II and calculate the mean and standard deviation density distributions in the same way as I did for the raw data.

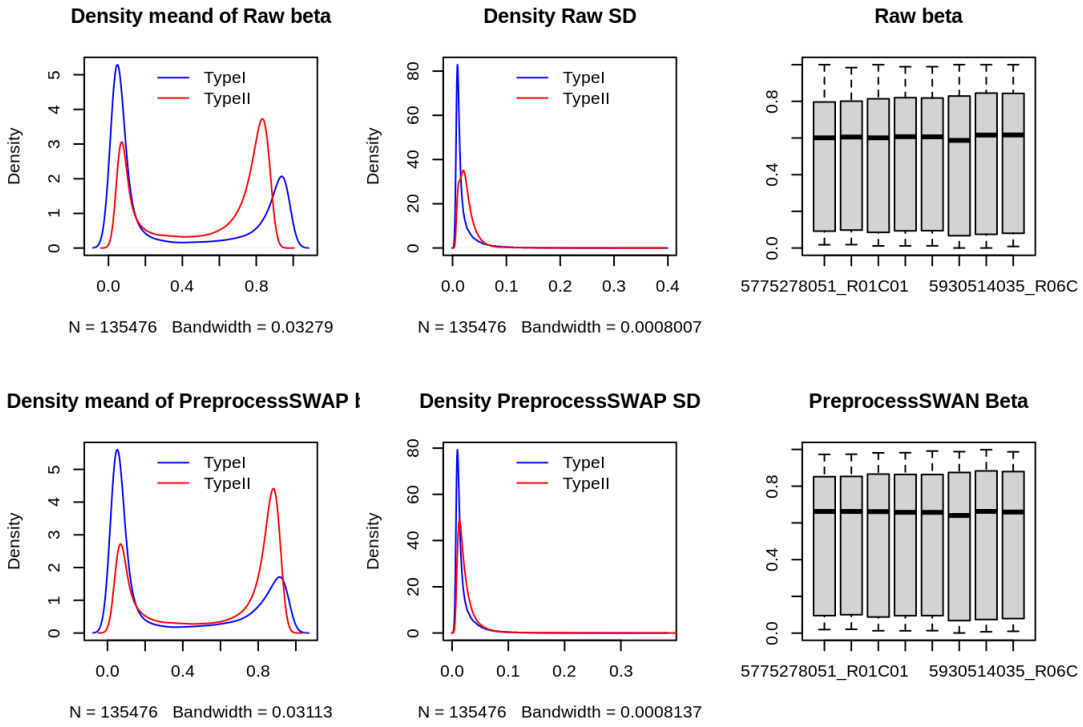
```
beta_preproSWAN=getBeta(preproSWAN)
beta_preproSWAN_I=beta_preproSWAN[rownames(beta_preproSWAN) %in% dfI$IlmnID,]
dim(beta_preproSWAN_I)
```

```
## [1] 135476      8
```

```
beta_preproSWAN_II=beta_preproSWAN[rownames(beta_preproSWAN) %in% dfII$IlmnID,]

mean_beta_preproSWAN_I=apply(beta_preproSWAN_I,1,mean)
mean_beta_preproSWAN_II=apply(beta_preproSWAN_II,1,mean)

d_mean_beta_preproSWAN_I=density(mean_beta_preproSWAN_I,na.rm=T)
d_mean_beta_preproSWAN_II=density(mean_beta_preproSWAN_II,na.rm=T)

sd_beta_preproSWAN_I=apply(beta_preproSWAN_I,1,sd)
sd_beta_preproSWAN_II=apply(beta_preproSWAN_II,1,sd)

d_sd_beta_preproSWAN_I=density(sd_beta_preproSWAN_I,na.rm=T)
d_sd_beta_preproSWAN_II=density(sd_beta_preproSWAN_II,na.rm=T)
```

Eventually I can plot the density distributions of beta means and beta standard deviation values according to the chemistry of the probes, for the raw and normalized data. I also produce a boxplot of the beta values of the raw and normalised data.

```
{
  par(mfrow=c(2,3))
plot(d_mean_of_betaI,col="blue",main="Density meand of Raw beta")
lines(d_mean_of_betaII,col="red")
legend("top", legend=c("TypeI","TypeII"), col=c("blue","red"), lty=1, bty="n")
plot(d_sd_betaI,col="blue",main="Density Raw SD")
lines(d_sd_betaII,col="red")
legend("top", legend=c("TypeI","TypeII"), col=c("blue","red"), lty=1, bty="n")
boxplot(beta, main="Raw beta")
plot(d_mean_beta_preproSWAN_I,col="blue", main="Density meand of PreprocessSWAP beta")
lines(d_mean_beta_preproSWAN_II,col="red")
legend("top", legend=c("TypeI","TypeII"), col=c("blue","red"), lty=1, bty="n")
plot(d_sd_beta_preproSWAN_I,col="blue", main="Density PreprocessSWAP SD")
lines(d_sd_beta_preproSWAN_II,col="red")
legend("top", legend=c("TypeI","TypeII"), col=c("blue","red"), lty=1, bty="n")
boxplot(beta_preproSWAN, main="PreprocessSWAN Beta")
}
```



From these plots it's possible to see that the distributions of the mean beta values for both the Type I and II probes it's not so different between the data before and after the normalization procedure, if not for a slight shift toward the center of the right peak of the blue distribution (Type I probes). The red peak of the standard deviation distribution of the normalized data is higher in respect to the raw standard deviation densities as expected, this is beacuse of the less accuracy of Infinium II probes. Moreover, from the boxplots of the normalized beta values we can see how their distribution is almost the same accross the eight samples, while is more variable for raw beta values.

Step8: Perform a PCA on the beta matrix generated in Step 7.

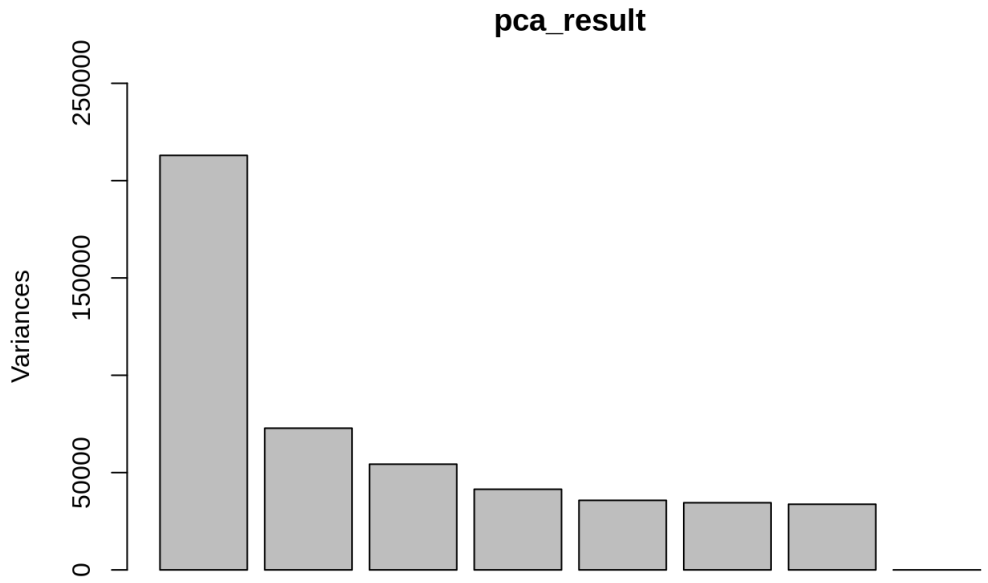
Principal Component Analysis (PCA) is a dimensionality-reduction method that is used to emphasize variation and bring out strong patterns in a datase and it is also used for the detection of outliers. To calculate the PCA I will use the *prcomp()* function to the transpose matrix, obtained with the *t()* function, of the beta preprocessSWAN values.

```
pca_result=prcomp(t(beta_preproSWAN),scale=T)
summary(pca_result)
```

```
## Importance of components:
##                             PC1      PC2      PC3       PC4       PC5       PC6
## Standard deviation     461.4632 269.8043 233.0785 203.53430 189.08073 185.77635
## Proportion of Variance   0.4386   0.1499   0.1119   0.08532   0.07364   0.07109
## Cumulative Proportion    0.4386   0.5885   0.7004   0.78576   0.85939   0.93048
##                             PC7       PC8
## Standard deviation     183.72029 4.028e-12
## Proportion of Variance   0.06952 0.000e+00
## Cumulative Proportion    1.00000 1.000e+00
```

We can see how thee first Principal Component (PC1) cover almost the 44% of variability, with a Cumulative Proportion of 59% with the PC2.

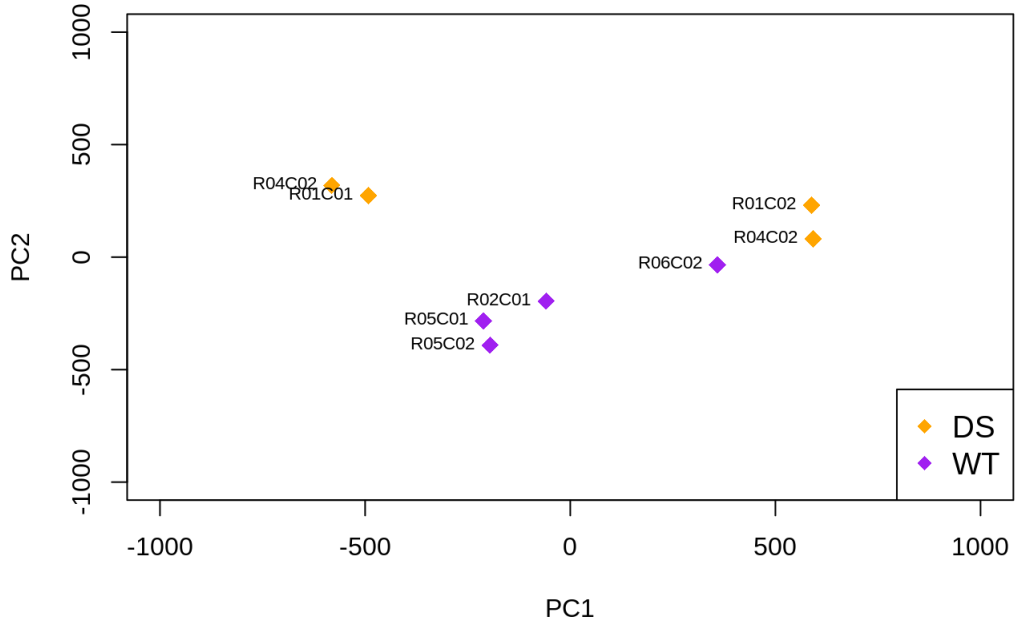
```
plot(pca_result,ylim=c(0,250000))
```



From the plot we can understand how the first 7 PC cover all the variability in the dataset.

I can now plot the first two components and check if the samples (DS and WT groups) cluster according to some criterias.

```
{
palette(c("orange","purple"))
plot(pca_result$x[,1],pca_result$x[,2],cex=1.5,pch=18,col=Samplesheet_report_2020$Group,xlab="PC1",ylab="PC
2",xlim=c(-1000,1000),ylim=c(-1000,1000))
legend("bottomright",legend=levels(Samplesheet_report_2020$Group),col=c(1:nlevels(Samplesheet_report_2020$Gr
oup)),pch=18,cex=1.2)
text(pca_result$x[,1],pca_result$x[,2],labels=targets$Array,pos=2,cex=0.7)
}
```



First of all I can see that there are no outliers, and that the WT Groups (violet) cluster quite well in respect to both the PC1 and also PC2, even if the samples *R06C02* is more toward the DS group (orange). I can also see that the samples corresponding to the *R01C02* and *R04C02* slides of the array have an higer PC1 in respect to the *R04C02* and *R01C01* samples of the same group (DS).

Step9: Using the matrix of normalized beta values generated in the step 7, identify differentially methylated probes between group DS and group WT using the *Mann-Whitney test*.

The aim of this test is to identify the CpG probes that are differentially methyated between samples belonging to DS and WT groups. In order to do this I will the Mann-Whitney U test that is a non-parametric and unpaired test. This test can be used to investigate whether two independent samples were selected from populations having the same distribution. The null hypothesis of this test is that the distributions of both the populations (DS and WT) are equal.

I will use the *apply()* function to extract the p-values from each row of the dataframe, and create an *ad-hoc* function since I want to apply the the Mann-Whitney U test, which is implemented in the *wilcoxon.test()* function, to each probe.

```
Mannwhytney_function=function(x) {
  MW=wilcox.test(x~Samplesheet_report_2020$Group)
  return(MW$p.value)
}
```

I apply it to the normalized beta values from step 7 using DS and WT as sample groups.

```
p_values_MW=apply(beta_preproSWAN,1,Mannwhytney_function)
summary(p_values_MW)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02857 0.34286 0.68571 0.62624 0.88571 1.00000
```

```
length(p_values_MW)
```

```
## [1] 485512
```

I can then create a data frame with all the beta values and the p_values for each probe and sort them from the smallest to the largest p-value.

```
final_MW_test=data.frame(beta_preproSWAN,p_values_MW)
final_MW_test=final_MW_test[order(final_MW_test$p_values_MW),]
summary(final_MW_test)
```

```
##  X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01 X5775278078_R05C01
##  Min.   :0.01940    Min.   :0.02077    Min.   :0.01292    Min.   :0.01282
##  1st Qu.:0.09539    1st Qu.:0.10042    1st Qu.:0.08790    1st Qu.:0.09471
##  Median :0.66217    Median :0.66219    Median :0.66150    Median :0.65757
##  Mean   :0.51016    Mean   :0.51191    Mean   :0.51357    Mean   :0.51449
##  3rd Qu.:0.85154    3rd Qu.:0.85286    3rd Qu.:0.86564    3rd Qu.:0.86382
##  Max.   :0.97269    Max.   :0.97358    Max.   :0.98120    Max.   :0.98219
##  X5775278078_R05C02 X5930514034_R01C02 X5930514035_R04C02 X5930514035_R06C02
##  Min.   :0.01365    Min.   :0.000733   Min.   :0.007434   Min.   :0.01000
##  1st Qu.:0.09543    1st Qu.:0.068359   1st Qu.:0.074042   1st Qu.:0.07929
##  Median :0.65729    Median :0.640344   Median :0.662723   Median :0.65922
##  Mean   :0.51504    Mean   :0.504964   Mean   :0.513997   Mean   :0.51400
##  3rd Qu.:0.86352    3rd Qu.:0.874588   3rd Qu.:0.882959   3rd Qu.:0.87920
##  Max.   :0.99106    Max.   :0.987677   Max.   :0.998505   Max.   :0.98675
##   p_values_MW
##  Min.   :0.02857
##  1st Qu.:0.34286
##  Median :0.68571
##  Mean   :0.62624
##  3rd Qu.:0.88571
##  Max.   :1.00000
```

Step10: Apply multiple test correction and set a significant threshold of 0.05. How many probes do you identify as differentially methylated considering nominal pValues? How many after Bonferroni correction? How many fater BH correction?

To perform the multiple test correction I will use the *p.adjust()* function. Given a set of p-values, the adjustment methods include both the Bonferroni correction in which the p-values are multiplied by the number of comparison, and the False Discovery Rate (BH method), that calculates the expected proportion of false discoveries among the rejected hypothesis. I will apply these methods to the sorted raw p-values stored in the data frame from step 9.

First of all I create a vector storing the p-value from the matrix, that are in the 9th column of the *final_MW_test* object.

```
raw_pvalues=final_MW_test[,9]
summary(raw_pvalues)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02857 0.34286 0.68571 0.62624 0.88571 1.00000
```

I can now test how many probes are differentially methylated by checking how many of them have a nominal p-value equal minor than 0.05.

```
diff_Meth_MW_rawpvalues_0.05=final_MW_test[final_MW_test$p_values_MW <= 0.05,]
dim(diff_Meth_MW_rawpvalues_0.05)
```

```
## [1] 22349     9
```

22441 probes are differentially methylated according to the Mann-Whitney test.

Now I can apply the Benjamini-Hochberg correction:

```
corrected_pValues_BH=p.adjust(raw_pvalues,"BH")
```

And then the Bonferroni correction:

```
corrected_pValues_Bonf=p.adjust(raw_pvalues,"bonferroni")
```

I can also add the new corrected p-values to the *final_MW_test* data frame (they are ordered from the smallest to the largest raw p-value).

```
final_pvalues_corrected_MW=data.frame(final_MW_test,corrected_pValues_BH,corrected_pValues_Bonf)
head(final_pvalues_corrected_MW, n=3)
```

```
##            X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01
## cg03695421          0.6725680          0.6271003          0.6191596
## cg00685229          0.4308417          0.4161036          0.5001259
## cg01370179          0.1679076          0.2025010          0.1514863
##            X5775278078_R05C01 X5775278078_R05C02 X5930514034_R01C02
## cg03695421          0.5845355          0.5951115          0.6820704
## cg00685229          0.4606298          0.4738984          0.4394252
## cg01370179          0.1429529          0.1189646          0.1694240
##            X5930514035_R04C02 X5930514035_R06C02 p_values_MW
## cg03695421          0.6984661          0.5011191  0.02857143
## cg00685229          0.3906338          0.4710031  0.02857143
## cg01370179          0.1532696          0.1434973  0.02857143
##            corrected_pValues_BH corrected_pValues_Bonf
## cg03695421            0.6206887                       1
## cg00685229            0.6206887                       1
## cg01370179            0.6206887                       1
```

Now I check how many probes are differentially methylated after BH and Bonferroni correction.

```
dim(final_pvalues_corrected_MW[final_pvalues_corrected_MW$corrected_pValues_BH <= 0.05,])
```

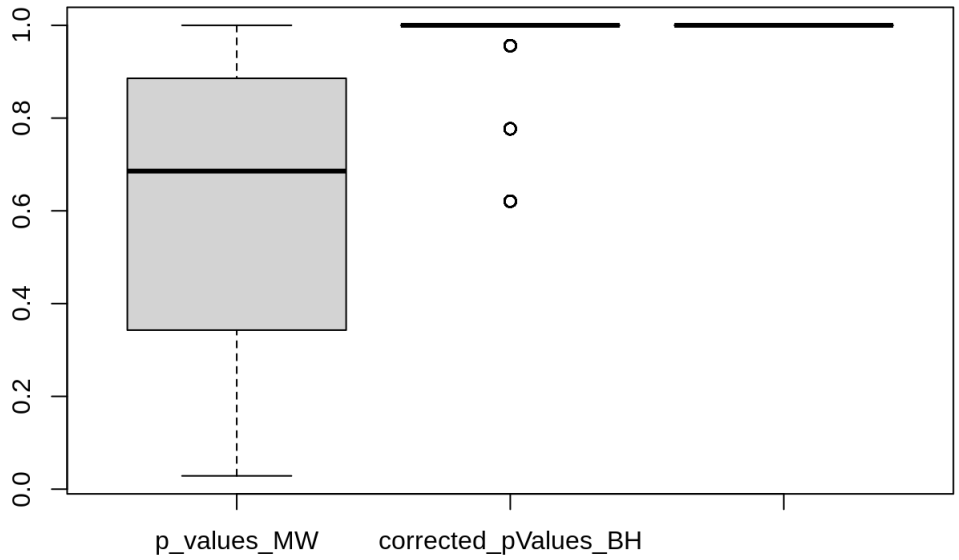
```
## [1]  0 11
```

```
dim(final_pvalues_corrected_MW[final_pvalues_corrected_MW$corrected_pValues_Bonf <= 0.05,])
```

```
## [1]  0 11
```

No significant probes are detected with a p-value minor than 0.05 after the multiple test corrections methods that I've applied. And I can also create a boxplot of the raw and corrected p-values to look at their distribution.

```
boxplot(final_pvalues_corrected_MW[,9:11])
```



Step11: Produce a heatmap of the top 100 differentially methylated probes.

A heatmap is a data visualization thechnique that shows magnitude of a phenomenon as color in two dimensions. The variation in color may be due i.e. for the instensity or the level of differetnially expressde genes. Heatmap are always coupled with hierarchical clusters, givin obvious visual cues about how the phenomenon is clustered or varies over space.

Firstly extract the beta values for the top 100 most significant CpG probes, and convert them in a matrix that will be our input for the *heatmap()* function.

```
input_heatmap=as.matrix(final_pvalues_corrected_MW[1:100,1:8])
summary(input_heatmap)
```

```
##  X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01 X5775278078_R05C01
##  Min.   :0.07208    Min.   :0.07048    Min.   :0.05704    Min.   :0.06483
##  1st Qu.:0.23883    1st Qu.:0.24301    1st Qu.:0.28479    1st Qu.:0.28104
##  Median :0.54022    Median :0.48604    Median :0.49496    Median :0.50117
##  Mean   :0.51010    Mean   :0.50154    Mean   :0.51624    Mean   :0.51019
##  3rd Qu.:0.73893    3rd Qu.:0.76631    3rd Qu.:0.75990    3rd Qu.:0.75608
##  Max.   :0.92956    Max.   :0.93379    Max.   :0.95259    Max.   :0.95110
##  X5775278078_R05C02 X5930514034_R01C02 X5930514035_R04C02 X5930514035_R06C02
##  Min.   :0.0579     Min.   :0.04031    Min.   :0.06735    Min.   :0.05275
##  1st Qu.:0.2725     1st Qu.:0.21786    1st Qu.:0.22370    1st Qu.:0.25858
##  Median :0.5100     Median :0.49228    Median :0.53991    Median :0.47783
##  Mean   :0.5192     Mean   :0.49675    Mean   :0.51099    Mean   :0.50383
##  3rd Qu.:0.7774     3rd Qu.:0.75855    3rd Qu.:0.75846    3rd Qu.:0.73958
##  Max.   :0.9532     Max.   :0.94694    Max.   :0.94598    Max.   :0.95198
```

I will now create the heatmap using a colorbar for the DS and WT group (blue and orange respectively), and then use the default method for the distances that is *euclidean* and *complete* for the linkage method.

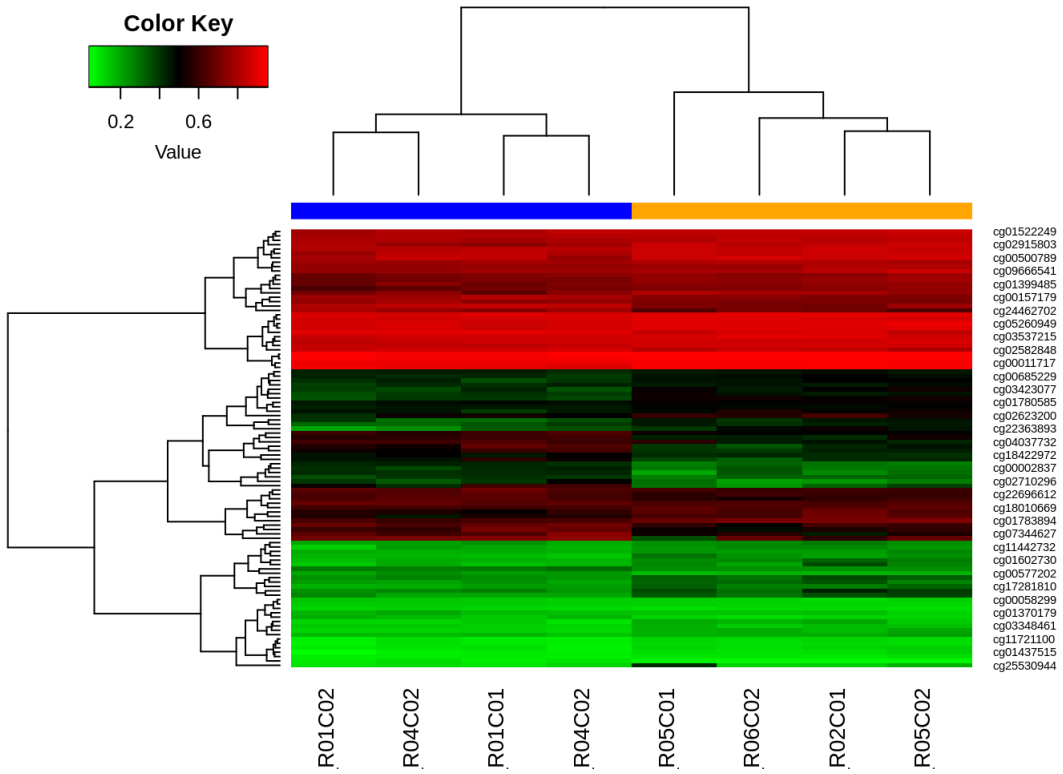
```
Samplesheet_report_2020$Group
```

```
## [1] DS DS WT WT WT DS DS WT
## Levels: DS WT
```

```
colorbar=c("blue","blue","orange","orange","orange","blue","blue","orange")
col2=colorRampPalette(c("green","black","red"))(100)
```

The *heatmap.2()* function that I will use to generate the plots belongs to the *gplots* package that I will procede to install and use.

```
install.packages("gplots")
library(gplots)
heatmap.2(input_heatmap,col = col2,Rowv=T,Colv=T,dendrogram="both",key=T,ColSideColors=colorbar,density.info
="none",trace="none",scale="none",symm=F)
```

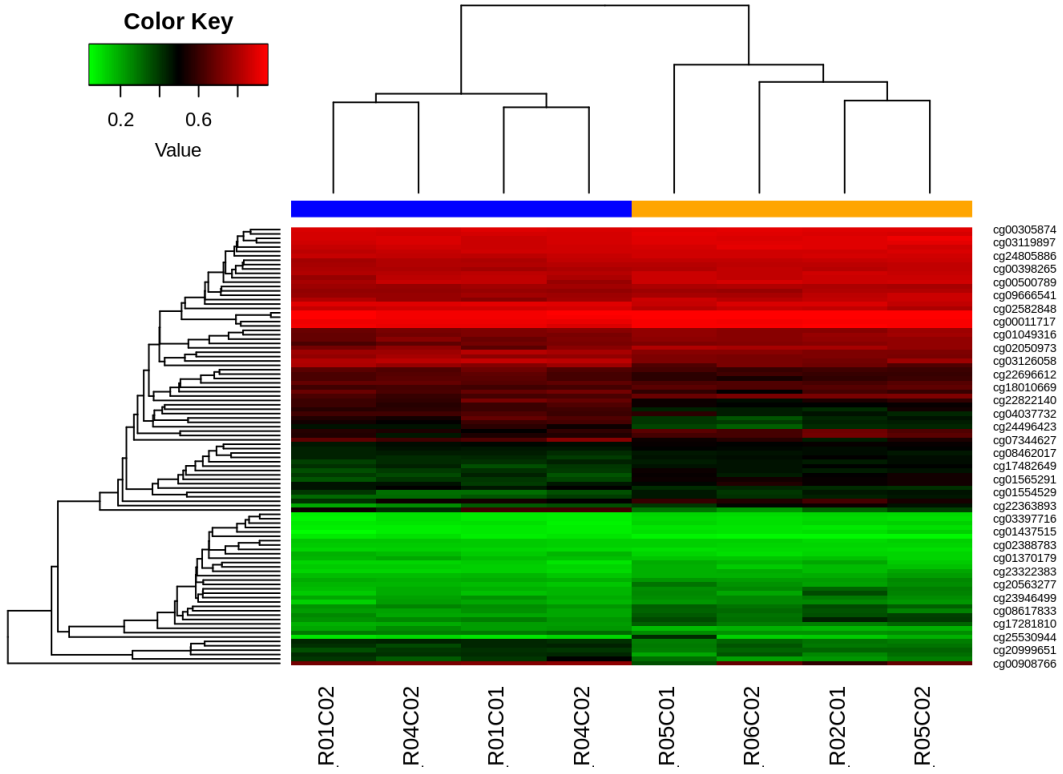


From the graph we can see that the first 100 probes of the two sample groups (DS and WT) are well clustered according to their methylation status.

Now I can generate other heatmaps changing the linkage method in single and average by specify it in the *hclust* parameter.

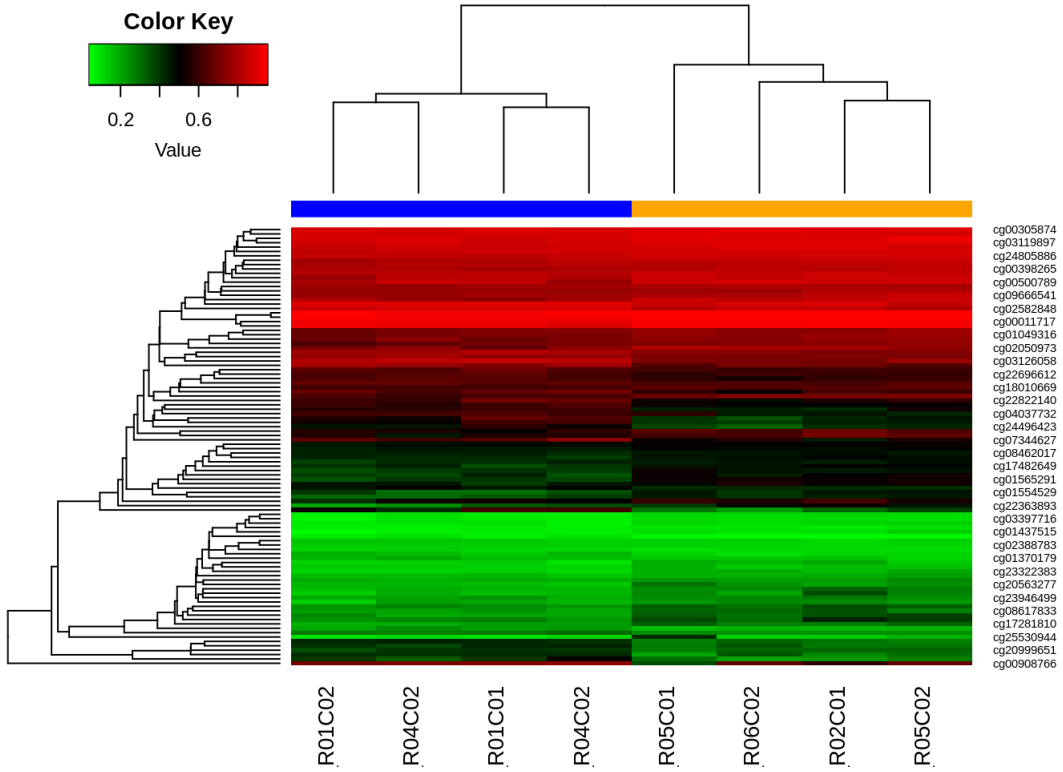
Single linkage

```
heatmap.2(input_heatmap,col=col2,Rowv = T, Colv = T, hclust=function(x) hclust(x,method="single"), dendrogra
m = "both", key = T, ColSideColors = colorbar, density.info = "none", trace = "none", scale = "none",symm =
F)
```



Average linkage

```
heatmap.2(input_heatmap,col=col2,Rowv = T, Colv = T, hclust=function(x) hclust(x,method="single"), dendrogram = "both", key = T, ColSideColors = colorbar, density.info = "none", trace = "none", scale = "none",symm = F)
```



From these three different heatmaps we can notice that the linkage methods used don't influence the clusterization procedure of the DS and WT samples, that are well distinguished in all the tree heatmaps. As well as the disitintion of the probes that are more or less methylated in one group or the other is well preserved in all the three graph.

Step12: Produce a volcano plot and a Manhattan plot of the results of differential methylation analysis.

Volcano plot

A volcano plot is a type of scatterplot that shows statistical significance *-log10(p-value)* versus the magnitude of change *log2(fold-change)*. It enables quick visual identification of genes with large folds changes that are also statistically significant and may be therefore more biologically significant genes.

First of all I have to calculate the fold change as the difference between the average beta values of DS samples and the average beta values of WT samples. I create two matrices containing the beta values of groups DS and WT samples extrapolating the beta values from the *final_pvalues_corrected_MW* data frame.

```
beta_ordered=final_pvalues_corrected_MW[,1:8]
beta_DS=beta_ordered[,Samplesheet_report_2020$Group=="DS"]
beta_WT=beta_ordered[,Samplesheet_report_2020$Group=="WT"]
```

I will then calculate the mean across each probe within these two sustets of beta values.

```
mean_beta_DS=apply(beta_DS,1,mean)
mean_beta_WT=apply(beta_WT,1,mean)
```

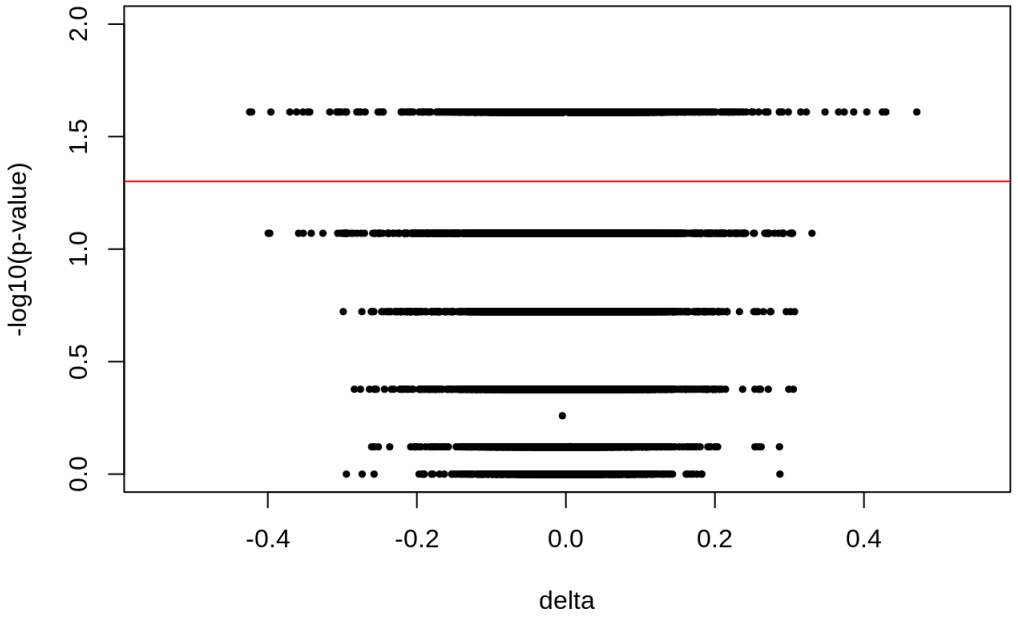
And then calculate the difference between the average beta values of the DS and WT sample probes.

```
delta=mean_beta_WT - mean_beta_DS
head(delta)
```

```
##  cg03695421  cg00685229  cg01370179  cg01522249  cg02639793  cg04037732
## -0.09506978  0.05716321 -0.03405027  0.03511907  0.12499336 -0.13901253
```

I can create a data frame containing in one column the delta values and in the other the *-log10* of the p-values, and then plot these values, respectively on the x and y axis using the *plot()* function. I will also use the *abline()* function to add the threshold (0.05) for the p-value significance.

```
{
toVolPlot=data.frame(delta, -log(final_pvalues_corrected_MW$p_values_MW))
plot(toVolPlot[,1],toVolPlot[,2], ylim=c(0,2.0), pch=19, cex=0.5,xlab="delta",ylab="-log10(p-value)")
abline(a=-log10(0.05), b=0, col="red")
}
```



The resulting shape of the Volcano plot is influenced by the non-parametric test (*Mann-whitney*) that we have used for the detection of the p-values. We can also see those probes, just above the 1.5, that exceed the significant threshold of 0.05.

Manhattan plot

The Manhattan plot is a type of scatter plot usually used to display data with large number of data points. This plot is commonly used in genome-wide association studies. On the x-axis are plotted the genomic coordinae and on the y-axis the *-log10(p-value)*.

First of all I want to annotate the *final_pvalues_corrected_MW* dataframe adding genome information for each CpG probe using the *Illumina450Manifest_clean* object. In order to do this I will use the *merge()* function that performs the merging of a column that is common to both the data frames. Since I want to merge on the basis of the CpG probes, I have to generate a new dataframe in which the CpG probe IDs are stored in the colname and not in the rownames as they currently are in the *final_pvalues_corrected_MW* dataframe.

```
final_pvalues_corrected_MW_CpG=data.frame(rownames(final_pvalues_corrected_MW),final_pvalues_corrected_MW)
colnames(final_pvalues_corrected_MW_CpG)[1]="IlmnID"
head(final_pvalues_corrected_MW_CpG,n=3)
```

```
##               IlmnID X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01
## cg03695421 cg03695421          0.6725680          0.6271003          0.6191596
## cg00685229 cg00685229          0.4308417          0.4161036          0.5001259
## cg01370179 cg01370179          0.1679076          0.2025010          0.1514863
##            X5775278078_R05C01 X5775278078_R05C02 X5930514034_R01C02
## cg03695421          0.5845355          0.5951115          0.6820704
## cg00685229          0.4606298          0.4738984          0.4394252
## cg01370179          0.1429529          0.1189646          0.1694240
##            X5930514035_R04C02 X5930514035_R06C02 p_values_MW
## cg03695421          0.6984661          0.5011191  0.02857143
## cg00685229          0.3906338          0.4710031  0.02857143
## cg01370179          0.1532696          0.1434973  0.02857143
##            corrected_pValues_BH corrected_pValues_Bonf
## cg03695421            0.6206887                       1
## cg00685229            0.6206887                       1
## cg01370179            0.6206887                       1
```

I can now merge the two data frames on the basis of the CpG probe ID.

```
final_pvalues_corrected_MW_annotated=merge(final_pvalues_corrected_MW_CpG,Illumina450Manifest_clean,by="Ilmn
ID")
dim(final_pvalues_corrected_MW_annotated)
```

```
## [1] 485512     44
```

Next step I will create the input for the Manhattan plot.

```
input_manhattan=data.frame(final_pvalues_corrected_MW_annotated$CHR, final_pvalues_corrected_MW_annotated$MA
PINFO, final_pvalues_corrected_MW_annotated$p_values_MW)

levels(input_manhattan$final_pvalues_corrected_MW_annotated.CHR)
```

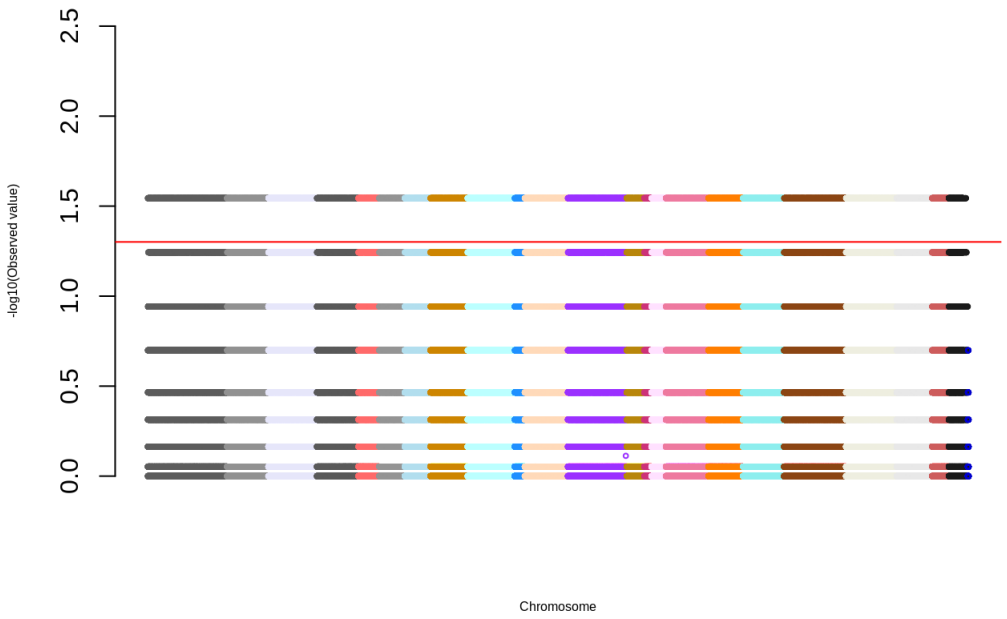
```
##  [1] "1"  "10" "11" "12" "13" "14" "15" "16" "17" "18" "19" "2"  "20" "21" "22"
## [16] "3"  "4"  "5"  "6"  "7"  "8"  "9"  "X"  "Y"
```

```
palette=rainbow(24)
library(gap)
```

```
## gap version 1.2.2
```

The Manhattan plot is created using the *mhtplot()* function of the *gap* package.

```
{
library(gap)
mhtplot(input_manhattan,cotrol=mht.control(colors=palette),ylim=c(0,2.5))
axis(2,cex=0.5)
abline(a=-log10(0.05),b=0,col="red")
}
```



```
## Plotting points  1 - 46857
## Plotting points  46858 - 71245
## Plotting points  71246 - 100039
## Plotting points  100040 - 124578
## Plotting points  124579 - 136863
## Plotting points  136864 - 151941
## Plotting points  151942 - 167200
## Plotting points  167201 - 189169
## Plotting points  189170 - 217048
## Plotting points  217049 - 222970
## Plotting points  222971 - 248491
## Plotting points  248492 - 283301
## Plotting points  283302 - 293680
## Plotting points  293681 - 297923
## Plotting points  297924 - 306475
## Plotting points  306476 - 331634
## Plotting points  331635 - 352098
## Plotting points  352099 - 376425
## Plotting points  376426 - 413036
## Plotting points  413037 - 443053
## Plotting points  443054 - 464003
## Plotting points  464004 - 473864
## Plotting points  473865 - 485096
## Plotting points  485097 - 485512
```

We can see in the Manhattan plot that there are no significant differences across chromosomes. Once again this is probably due to the use of the non-parametric test (Mann-Whitney) that is not able to detect differentially methylated probes.

Optional: As DS is caused by the trisomy of chromosome 21, try to plot the density of the methylation values f the probes mapping on chromosome 21. Do you see a very clear difference between the samples? How many differentially metylated probes do you find in chromosome 21?

I first extact only the CpG probes corresponding to the Chromosome 21 from the *final_pvalues_corrected_MW_annotated* data frame and extrapolate only those line for which I'm interested in (the beta values of the probes and the nominal and corrected p-values) that are found from column 1 to column 12.

```
Chromosome21=final_pvalues_corrected_MW_annotated[final_pvalues_corrected_MW_annotated$CHR=="21",]
Chromosome_21=data.frame(Chromosome21[,1:12])
dim(Chromosome_21)
```

```
## [1] 4243   12
```

I will then create two vectors with the array names corresponding to the DS and WT groups.

```
DS_1=c("X5775278051_R01C01","X5775278051_R04C02","X5930514034_R01C02","X5930514035_R04C02")
WT_1=c("X5775278078_R02C01","X5775278078_R05C01","X5775278078_R05C02","X5930514035_R06C02")
```

I can now divide the data frame according to the DS and WT groups.

```
Chromosome_21_DS=Chromosome_21[,colnames(Chromosome_21) %in% DS_1]
Chromosome_21_WT=Chromosome_21[,colnames(Chromosome_21) %in% WT_1]
```

Now I calculate the mean of the beta values for the probes in the two sample groups.

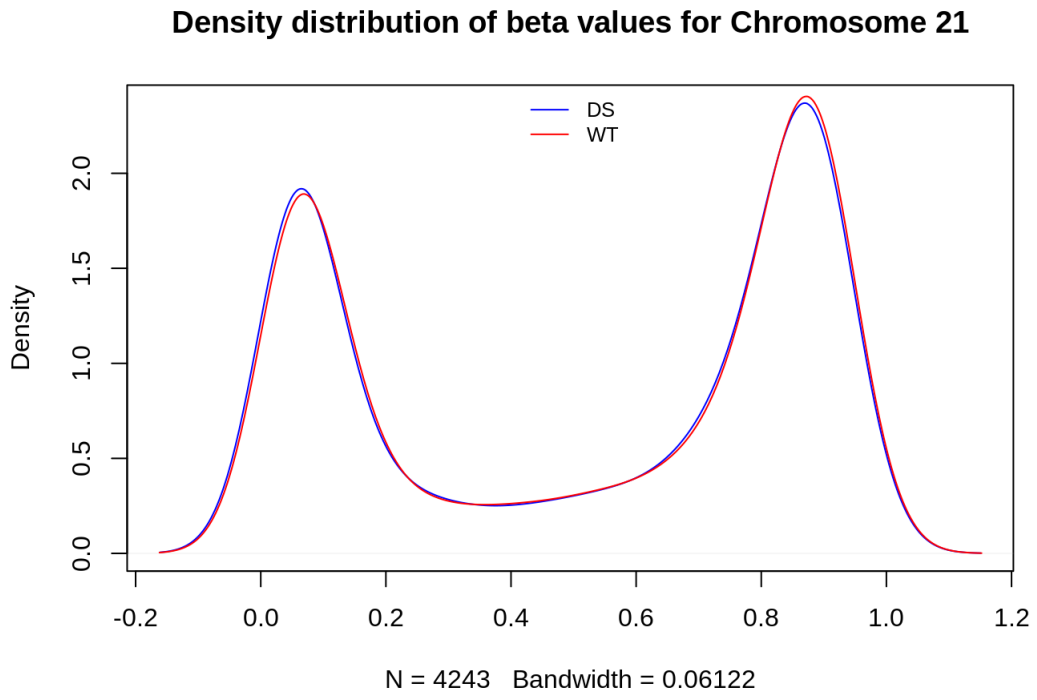
```
mean_Chromosome_21_DS=apply(Chromosome_21_DS,1,mean)
mean_Chromosome_21_WT=apply(Chromosome_21_WT,1,mean)
```

And then the density distributions of beta values.

```
d_mean_Chromosome_21_DS=density(mean_Chromosome_21_DS,na.rm=T)
d_mean_Chromosome_21_WT=density(mean_Chromosome_21_WT,na.rm=T)
```

I can next plot the two density distributions of beta values.

```
{
plot(d_mean_Chromosome_21_DS,col="blue", main="Density distribution of beta values for Chromosome 21")
lines(d_mean_Chromosome_21_WT,col="red")
legend("top", legend=c("DS","WT"), col=c("blue","red"), lty=1, bty="n",cex=0.8)
}
```



Both the distributions for the DS and WT groups are very similar.

I check how many differentually methylated probes I can find in the Chromosome 21 between the 2 groups by looking if their p-value is minor or equal to the threshold of 0.05.

```
diff_methylated_CHR21=Chromosome_21[Chromosome_21$p_values_MW <= 0.05,]
dim(diff_methylated_CHR21)
```

```
## [1] 302  12
```

From the dimension of the *diff_methylated_CHR21* we can see the probes belonging to the chromosome 21 that have a p-value minor or equal to 0.05 and we can therefore say that these probes are differentially metylated between the DS and WT groups.