DRD project

Lui Maria

March 14, 2020

1 Steps of the analysis pipeline

1. Load raw data with minfi and create an object called RGset storing the RGChannelSet object .

```
getwd()
# Load idat files and the experiment samplesheet using minfi
library(minfi)
vignette("minfi")
# Set the directory in which the raw data are stored
baseDir <- ("/home/maria/Desktop/DRD2/Input_data")
targets <- read.metharray.sheet(baseDir)
targets

# Create an object of class RGChannelSet
RGset <- read.metharray.exp(targets = targets)
#These classes represents raw (unprocessed) data from a two
    color micro array; specifically an Illumina methylation array

# Now I change the directory in which I want to save the new
    files
setwd("/home/maria/Desktop/DRD2/newfiles")
save(RGset,file="RGset.RData")

RGset
?RGChannelSet
```

2. Create the dataframes Red and Green to store the red and green fluorescence respectively .

```
Red <- data.frame(getRed(RGset))
#dim(Red)
Green <- data.frame(getGreen(RGset))
#dim(Green)
```

SAMPLE	RED	GREEN	TYPE
X5775278008_R01C01	3445	858	II
X5775278008_R02C01	5408	1223	II
X5775278008_R04C01	863	1204	II
X5775278008_R05C01	1053	1044	II
X5775278035_R01C01	2114	505	II
X5775278035_R02C01	468	579	II
X5775278035_R04C01	6395	1234	II
X5775278035_R05C01	1292	1120	II
X9344737127_R01C02	330	354	II
X9344737127_R03C02	282	493	II
X9344737127_R06C01	330	468	II
X9344737127_R02C02	379	421	II
X9376538140_R02C01	399	390	II
X9376538140_R05C01	333	577	II
X9376538140_R01C01	345	406	II
X9376538140_R04C01	3212	550	II

3. Fill the following table: what are the Red and Green fluorescence for the 41620492 address? Optional: check from the manifest file if the address corresponds to a Type I or a Type II probe and, in case of Type I probe, report its color.

```
#head(Red)
red_address <- Red[rownames(Red)=="41620492",]
red_address
#head(Green)
green_address <- Green[rownames(Green)=="41620492",]
green_address

load('/home/maria/Desktop/DRD2/Illumina450Manifest.RData')
type<-as.character(Illumina450Manifest[Illumina450Manifest$
    AddressA_ID=="41620492",]$Infinium_Design_Type)
type
allcolors <- as.data.frame(cbind(t(red_address),t(green_address)
    ))
allcolors
```

4. Create the object MSet.raw .

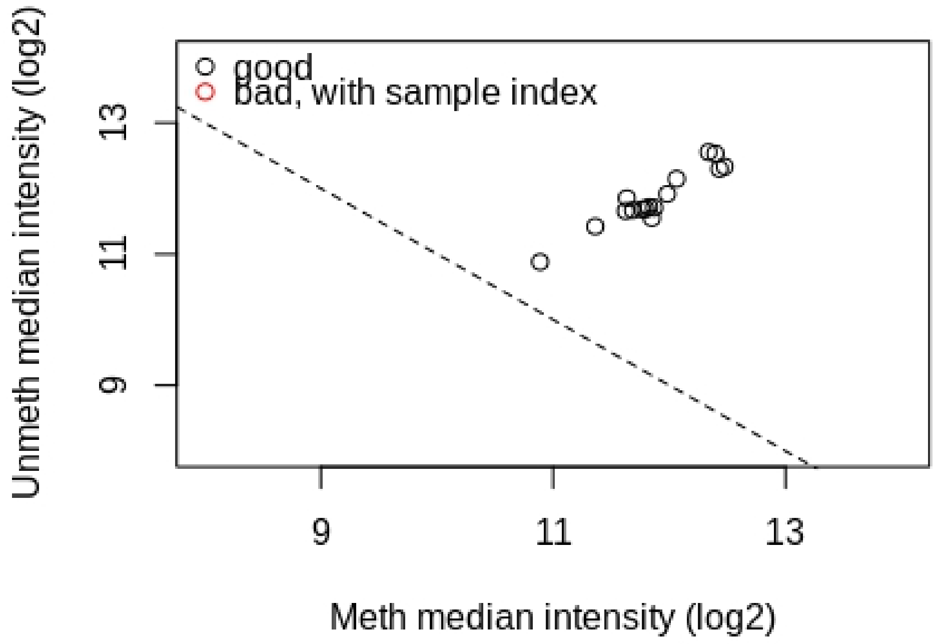
```r
###################################
### Extract methylated and unmethylated signals
###################################

# Accessor functions
#getMeth()
#getUnmeth()
MSet.raw <- preprocessRaw(RGset)
MSet.raw
save(MSet.raw,file="MSet_raw.RData")
Meth <- getMeth(MSet.raw)
```

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

   - QCplot:

   ```r
   qc <- getQC(MSet.raw)
   plotQC(qc)
   ```

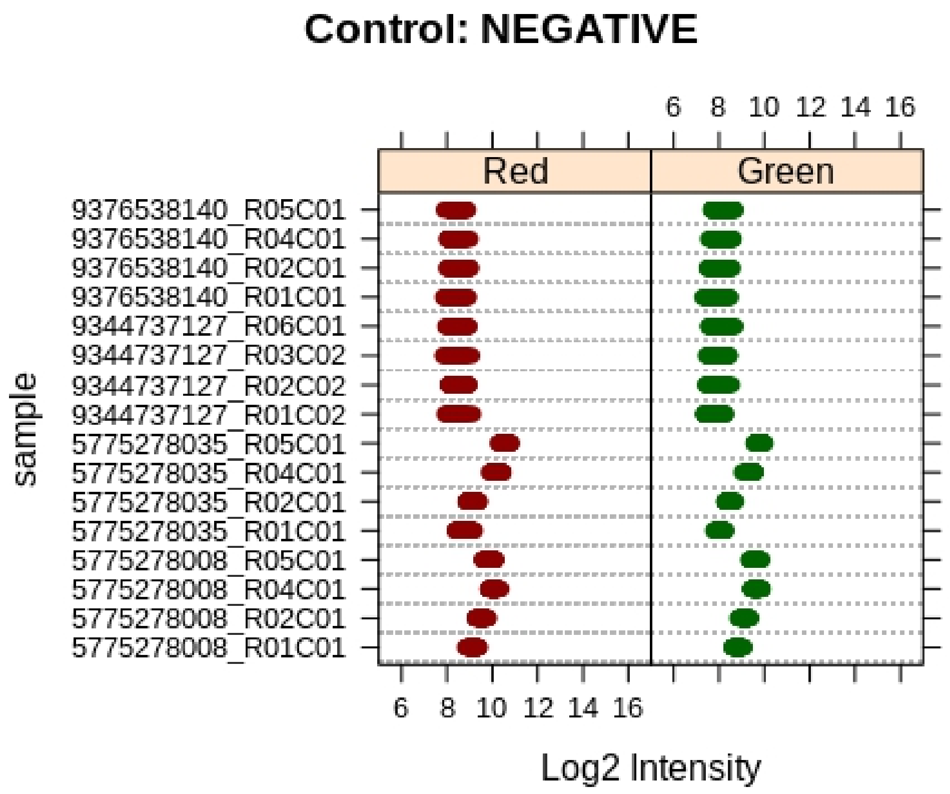
   getQC minfi: Estimates sample-specific quality control (QC) for methylation data. It gets the value of a DataFrame with two columns: mMed and uMed which are the chipwide medians of the Meth and Unmeth channels. Then it provides a plot where it is possible to observe that good samples cluster together, while failed samples tend to separate and have lower median intensities.



- check the intensity of negative controls using minfi:

```
####################################
### Control probes
####################################
# getProbeInfo returns the types and the numbers of control
    probes of each type. It can act both on RGChannelSet and
    on MethylSet classes.
getProbeInfo(RGset, type = "Control")

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

Internal control probes can be used to assess the quality of different sample preparation steps (bisulfite conversion, hybridization, etc.). The values of these control probes are stored in the initial RGChannelSet and can be plotted by using the function controlStripPlot and by specifying the control probe type that in this case are Negative controls



- Calculate detection pValues; for each sample, how many probes have a detection p-value higher than the threshold assigned to each student?

```r
### Detection p-value
#####################################
?detectionP
detP <- detectionP(RGset)
save(detP,file="detP.RData")
str(detP)
dim(detP)
head(detP)

failed <- detP>0.05
head(failed)
dim(failed)
table(failed)
summary(failed) # In this way we can now how many probes
    failed for each sample
```

we used the detectionP function of minfi: this function detect the p-values, for all probed genomic positions in every sample. Small p-values indicate a good position. Positions

SAMPLE	Failed positions
9344737127_R01C02	499
9344737127_R03C02	428
9344737127_R06C01	480
9344737127_R02C02	463
9376538140_R02C01	441
9376538140_R05C01	420
9376538140_R01C01	523
9376538140_R04C01	97

with non-significant p-values (here > 0.05) should not be trusted. So we identified failed positions defined as both the methylated and unmethylated channel reporting background signal levels.

- Create a vector of bad probes to be removed because they have a detection pValue higher that the assigned threshold (see above) in more than 1% of the samples.

```r
# We can remove the probes having > 1 % of samples with a
   detection p-value greater than 0.05
means_rows <- rowMeans(failed)
head(means_rows)
length(means_rows)
probes_to_be_removed <- means_rows[means_rows>0.01]
length(probes_to_be_removed)
head(probes_to_be_removed)
names_probes_to_be_removed <- names(probes_to_be_removed)
head(names_probes_to_be_removed)
str(names_probes_to_be_removed)
save(names_probes_to_be_removed,file="names_probes_to_be_
   removed.RData")
```

6. Calculate raw beta and M values and plot the densities of mean methylation values, dividing the samples in DS and WT .
Beta value is the percentage of methylation at the given CpG site; it is a continuous variable between 0 (absent methylation) and 1(completely methylated)

$$\beta = \frac{M}{M+U} \tag{1}$$

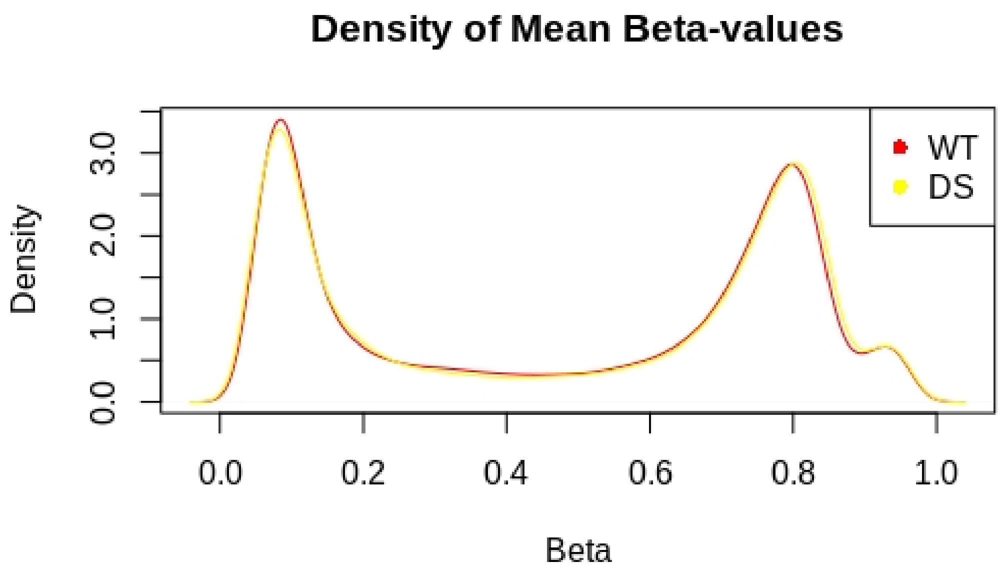
The M value is a continuous variable which can in principle take on any value on the real line.

$$M = \log_2 \frac{M}{U} \tag{2}$$

```r
#subset the beta and M values matrixes in order to retain DS or
   WT subjects
DS <- targets[targets$Group=="DS",]
DS <- droplevels(DS)
RGset_DS <- read.metharray.exp(targets = DS)
MSet_DS.raw <- preprocessRaw(RGset_DS)


WT <- targets[targets$Group=="WT",]
WT <- droplevels(WT)
RGset_WT <- read.metharray.exp(targets = WT)
MSet_WT.raw <- preprocessRaw(RGset_WT)
```

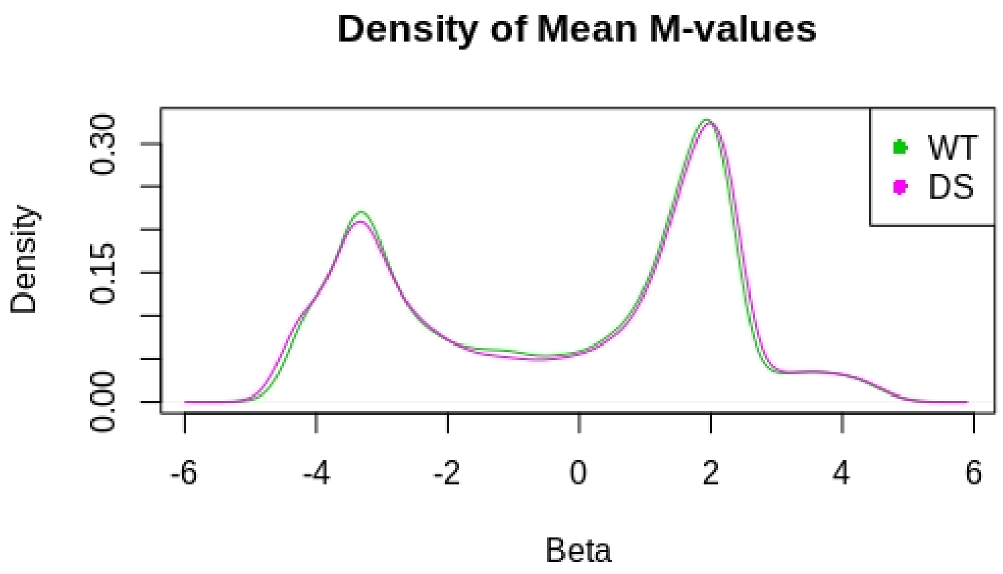
- Beta value:

```r
#Density plots of Beta :
# I want to plot the density distribution of mean beta for each
   CpG
# First, I have to calculate the mean of each row of the
   dataframe
# apply <- works on a dataframe or a matrix and returns a vector
?apply
beta_DS <- getBeta(MSet_DS.raw)
beta_WT <- getBeta(MSet_WT.raw)

mean_beta_DS <- apply(beta_DS,1,mean) # 1 indicates that I apply
    the functiont to each row
mean_beta_WT <- apply(beta_WT,1,mean)
d_mean_beta_DS <- density(mean_beta_DS,na.rm=T)
d_mean_beta_WT <- density(mean_beta_WT,na.rm=T)
par(mfrow=c(1,2))
plot(d_mean_beta_WT,col=2,main="Density of Mean Beta-values",
   xlab="Beta")
lines(d_mean_beta_DS,col=7)
legend('topright',pch=c(16,16),col=c(2,7),legend=c("WT","DS"))
```



- M value:

```
#Density plots of M values :
# I want to plot the density distribution of mean M values for
    each CpG
Mvalue_DS <- getM(MSet_DS.raw)
Mvalue_WT <- getM(MSet_WT.raw)
mean_M_DS <- apply(Mvalue_DS,1,mean)
mean_M_WT <- apply(Mvalue_WT,1,mean)
d_mean_M_DS <- density(mean_M_DS,na.rm=T)
d_mean_M_WT <- density(mean_M_WT,na.rm=T)
par(mfrow=c(1,2))
plot(d_mean_M_WT,col=3,main="Density of Mean M-values",xlab="
    Beta")
lines(d_mean_M_DS,col=6)
legend('topright',pch=c(16,16),col=c(3,6),legend=c("WT","DS"))
```

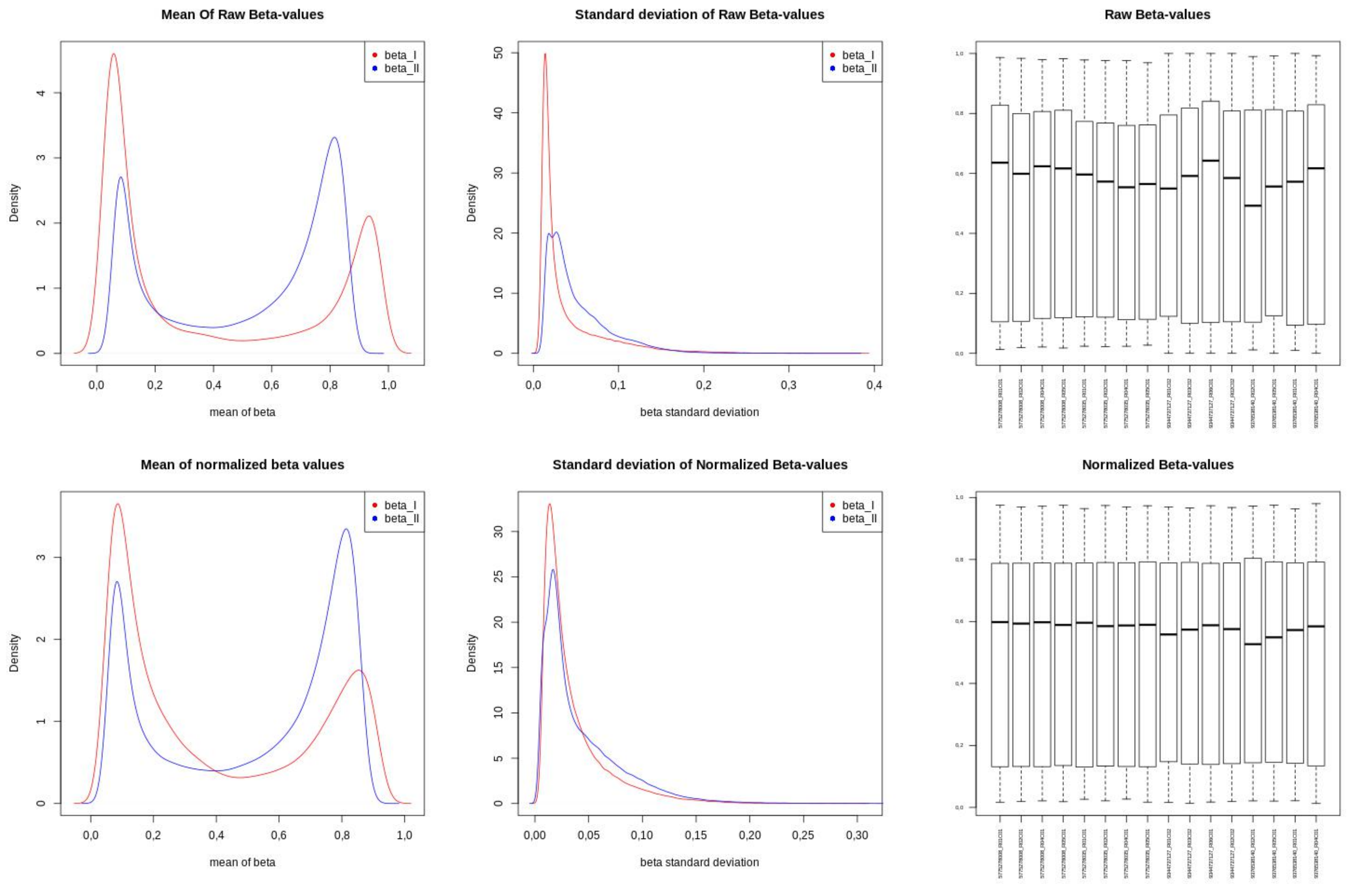


7. Compare raw data with normalized data using preprocessQuantile function. Produce a plot with 6 panels (2 rows, 3 columns) in which, for both raw and normalized data, you show the density plots of beta mean values according to the chemistry of the probes, the density plot of beta standard deviation values according to the chemistry of the probes and the boxplot of beta values. Provide a short comment regarding the changes you observe.

```
load('/home/maria/Desktop/DRD2/Illumina450Manifest_clean.RData')

dfI <- Illumina450Manifest_clean[Illumina450Manifest_clean$
    Infinium_Design_Type=="I",]
dfI <- droplevels(dfI)
dim(dfI)
str(dfI)
dfII <- Illumina450Manifest_clean[Illumina450Manifest_clean$
    Infinium_Design_Type=="II",]
dfII <- droplevels(dfII)


# I subset the dataframe beta in order to retain only the rows
    whose name is in the first column of dfI
beta_I <- beta[rownames(beta) %in% dfI$IlmnID,]
dim(beta_I)
# I subset the dataframe beta in order to retain only the rows
    whose name is in the first column of dfII
beta_II <- beta[rownames(beta) %in% dfII$IlmnID,]
dim(beta_II)

# I  want to compare the density distributions of type I and
    type II probes
# first, I have to calculate the mean of beta for each row
mean_of_beta_I <- apply(beta_I,1,mean)
mean_of_beta_II <- apply(beta_II,1,mean)

d_mean_of_beta_I <- density(mean_of_beta_I,na.rm=T)
d_mean_of_beta_II <- density(mean_of_beta_II,na.rm=T)

plot(d_mean_of_beta_I,main="Mean Of Raw Beta-values",col=2,xlab=
    "mean of beta")
lines(d_mean_of_beta_II,col=4)
legend('topright',pch=c(16,16),col=c(2,4),legend=c("beta_I","
    beta_II"))
#####
# For Raw data we have just prepared the density plot of mean
    betas...let's prepare the density plot of standard deviations
sd_of_beta_I <- apply(beta_I,1,sd)
sd_of_beta_II <- apply(beta_II,1,sd)
d_sd_of_beta_I <- density(sd_of_beta_I,na.rm=T)
d_sd_of_beta_II <- density(sd_of_beta_II,na.rm=T)

plot(d_sd_of_beta_I,main="Standard deviation of Raw Beta-values"
    ,col=2,xlab="beta standard deviation")
```

```
lines(d_sd_of_beta_II,col=4)
legend('topright',pch=c(16,16),col=c(2,4),legend=c("beta_I","beta_II"))

boxplot(beta, main="Raw Beta-values", las=2, cex.axis=0.5)

####################################

# preprocessQuantile (minfi; input: RGset or Mset; output: GenomicRatioSet)
?preprocessQuantile
RGset
preprocessQuantile_results <- preprocessQuantile(RGset)
str(preprocessQuantile_results)
class(preprocessQuantile_results)
preprocessQuantile_results
beta_preprocessQuantile <- getBeta(preprocessQuantile_results)
head(beta_preprocessQuantile)

beta_preprocessQuantile_I <- beta_preprocessQuantile[rownames(beta_preprocessQuantile) %in% dfI$IlmnID,]
beta_preprocessQuantile_II <- beta_preprocessQuantile[rownames(beta_preprocessQuantile) %in% dfII$IlmnID,]
mean_of_beta_preprocessQuantile_I <- apply(beta_preprocessQuantile_I,1,mean)
mean_of_beta_preprocessQuantile_II <- apply(beta_preprocessQuantile_II,1,mean)
d_mean_of_beta_preprocessQuantile_I <- density(mean_of_beta_preprocessQuantile_I,na.rm=T)
d_mean_of_beta_preprocessQuantile_II <- density(mean_of_beta_preprocessQuantile_II,na.rm=T)
sd_of_beta_preprocessQuantile_I <- apply(beta_preprocessQuantile_I,1,sd)
sd_of_beta_preprocessQuantile_II <- apply(beta_preprocessQuantile_II,1,sd)
d_sd_of_beta_preprocessQuantile_I <- density(sd_of_beta_preprocessQuantile_I,na.rm=T)
d_sd_of_beta_preprocessQuantile_II <- density(sd_of_beta_preprocessQuantile_II,na.rm=T)
par(mfrow=c(1,3))
jpeg('norm1.jpg')
plot(d_mean_of_beta_preprocessQuantile_I,na.rm=T,main="Mean of normalized beta values",col=2,xlab="mean of beta")
lines(d_mean_of_beta_preprocessQuantile_II,col=4)
legend('topright',pch=c(16,16),col=c(2,4),legend=c("beta_I","beta_II"))
```

```
dev.off()
jpeg('norm2.jpg')
plot(d_sd_of_beta_preprocessQuantile_I ,main="Standard deviation
     of Normalized Beta-values",col=2,xlab="beta standard
    deviation")
lines(d_sd_of_beta_preprocessQuantile_II,col=4)
legend('topright',pch=c(16,16),col=c(2,4),legend=c("beta_I","
    beta_II"))
dev.off()
jpeg('norm3.jpg')
boxplot(beta_preprocessQuantile, main="Normalized Beta-values",
    las=2, cex.axis=0.5)
dev.off()
```



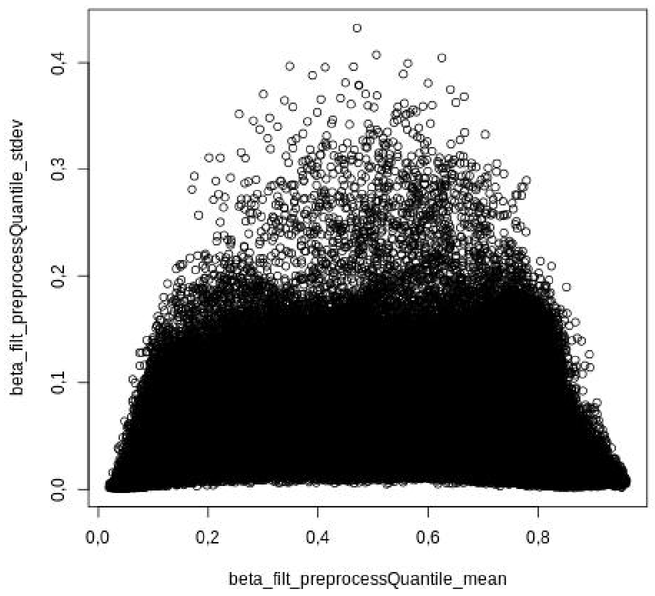
From these plots it can be appreciated how the normalization procedure removed technical variation between different array and different samples. As we can see from the plot, after applying preprocessQuantile normalization, the distributions of -values appear to look more similar than Raw Beta-values.

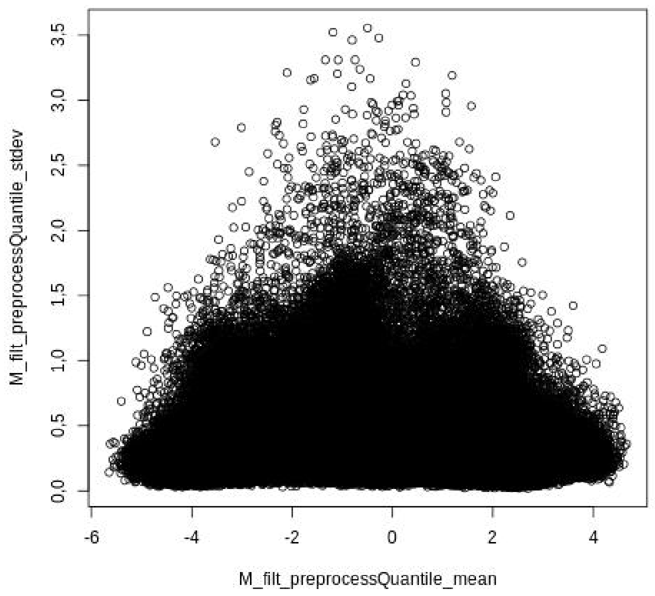
8. Consider the normalized beta and M values and remove the probes previously defined as bad according to the detection pValue.

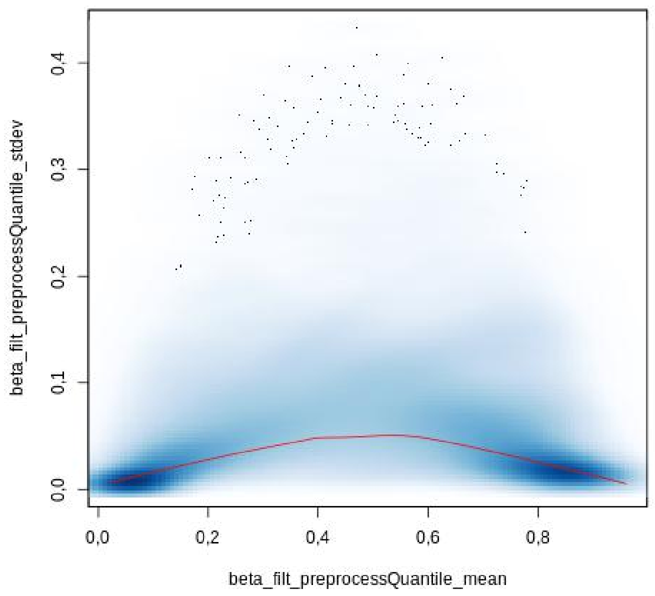
```
beta_preprocessQuantile <- getBeta(preprocessQuantile_results)
M_preprocessQuantile <- getM(preprocessQuantile_results)

summary(beta_preprocessQuantile)
summary(M_preprocessQuantile)
beta_filt_preprocessQuantile <- beta_preprocessQuantile[!
    rownames(beta_preprocessQuantile) %in% names_probes_to_be_
    removed,]
dim(beta_filt_preprocessQuantile)
M_filt_preprocessQuantile <- M_preprocessQuantile[!rownames(M_
    preprocessQuantile) %in% names_probes_to_be_removed,]
dim(M_filt_preprocessQuantile)
```

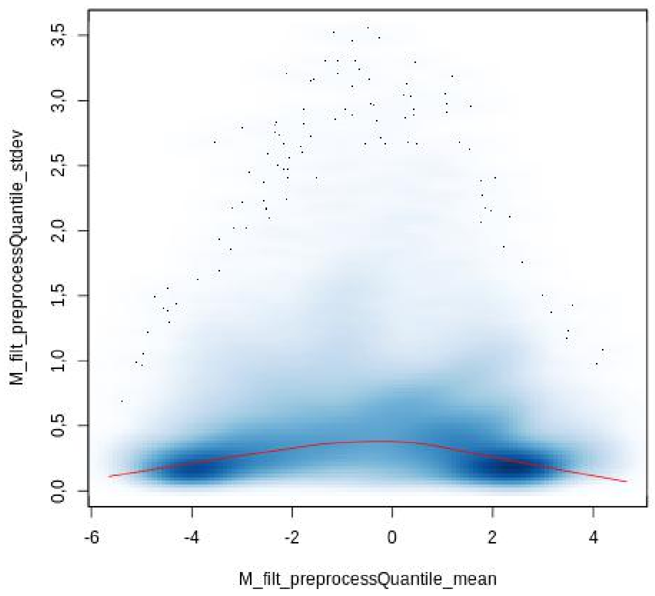
9. Check beta and M matrixes generated in step 8 for homo/heteroschedasticity; comment the plot. Optional: plot the lowess line.

```
# HOMOSCEDASTICITY and HETEROSCEDASTICITY

beta_filt_preprocessQuantile_mean <- apply(beta_filt_
    preprocessQuantile,1, mean)
beta_filt_preprocessQuantile_stdev <- apply(beta_filt_
    preprocessQuantile,1, sd)

M_filt_preprocessQuantile_mean <- apply(M_filt_
    preprocessQuantile,1, mean)
M_filt_preprocessQuantile_stdev <- apply(M_filt_
    preprocessQuantile,1, sd)
jpeg('betameandsd3.jpg')
plot(beta_filt_preprocessQuantile_mean, beta_filt_
    preprocessQuantile_stdev)
dev.off()
jpeg('Mmeandsd3.jpg')
plot(M_filt_preprocessQuantile_mean, M_filt_preprocessQuantile_
    stdev)
dev.off()

par(mfrow=c(1,2))
jpeg('LOWESS1.jpg')
smoothScatter(beta_filt_preprocessQuantile_mean, beta_filt_
    preprocessQuantile_stdev)
lines(lowess(beta_filt_preprocessQuantile_mean, beta_filt_
    preprocessQuantile_stdev), col="red") # lowess carries out a
    locally weighted regression of y on x
dev.off()
jpeg('LOWESS2.jpg')
smoothScatter(M_filt_preprocessQuantile_mean, M_filt_
    preprocessQuantile_stdev)
lines(lowess(M_filt_preprocessQuantile_mean, M_filt_
    preprocessQuantile_stdev), col="red") # lowess line (x,y)
dev.off()
```


beta_filt_preprocessQuantile_stdev
beta_filt_preprocessQuantile_mean
0,0
0,1
0,2
0,3
0,4
0,0
0,2
0,4
0,6
0,8


M_filt_preprocessQuantile_stdev
M_filt_preprocessQuantile_mean
0,0
0,5
1,0
1,5
2,0
2,5
3,0
3,5
-6
-4
-2
0
2
4

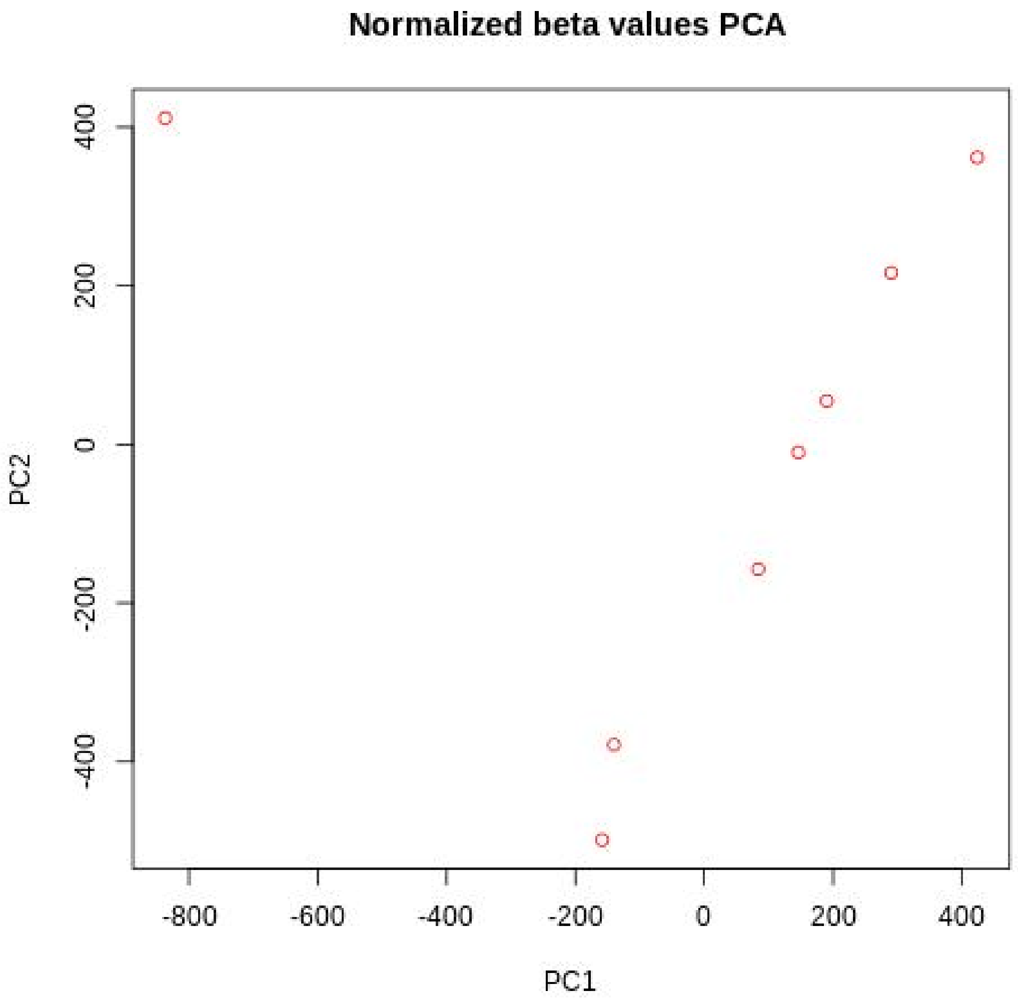

beta_filt_preprocessQuantile_stdev
beta_filt_preprocessQuantile_mean
0,0
0,1
0,2
0,3
0,4
0,0
0,2
0,4
0,6
0,8


M_filt_preprocessQuantile_stdev
M_filt_preprocessQuantile_mean
0,0
0,5
1,0
1,5
2,0
2,5
3,0
3,5
-6
-4
-2
0
2
4

This plots show Beta-value with a significant heteroscedasticity and M-values are approximately more homoscedastic.

10. Perform a PCA on the beta matrix generated in step 8 to check for batch effects. Comment the plot.

```r
###########################################
# PCA as diagnostic plots for outliers and batch effects

?prcomp
pca <- prcomp(t(beta_filt_preprocessQuantile),scale=T)
print(summary(pca)) # print variance accounted for each
    component
str(pca)
pca$x
jpeg('PCA.jpg')
plot(pca$x[,1],pca$x[,2],main="Normalized beta values PCA ",
    xlab="PC1", ylab="PC2",col="red")
dev.off()
```



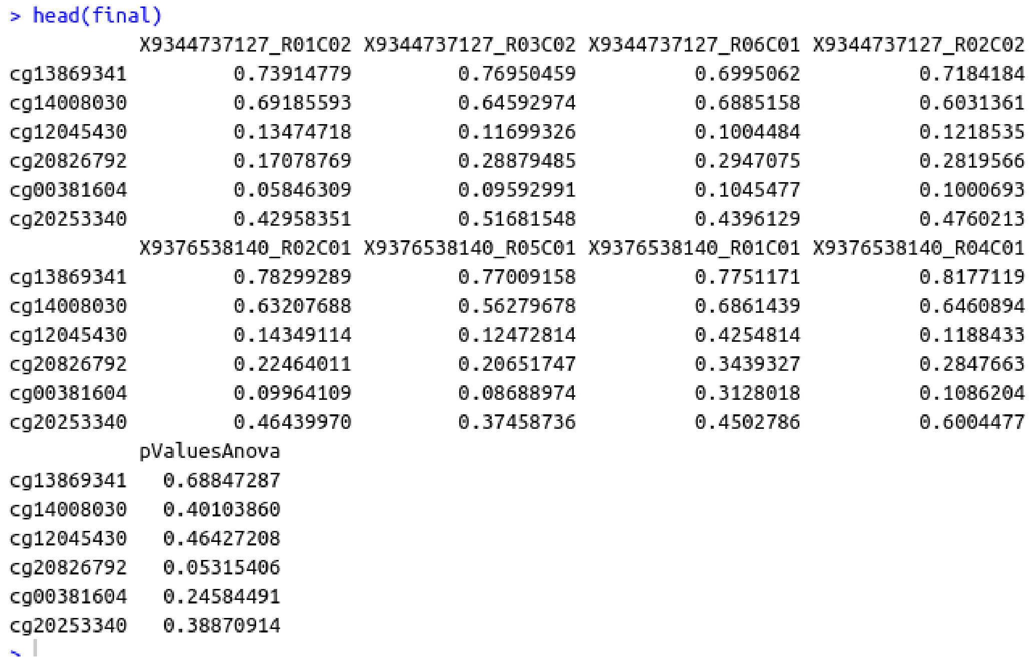
The distances among samples in a pca scores plot are related to their similarity with respect to the pattern described by the model and so this type of plots can be used for the identification of outliers and batch effects. Batch effects are technical sources of variation that have been added to the samples during handling. Here, from the obtained plot no obvious batch effect is observed.

11. Using the beta matrix generated in step 8, identify differentially methylated probes between group A and group B using Anova correcting for Female .

```
#####11#####
pheno <- read.csv("/home/maria/Desktop/DRD2/Input_data/
    SampleSheet.csv",sep=",")
```

```r
pheno
pheno$Group
pheno$Female <- factor(c("0","1"))
nlevels(pheno$Female)
pheno$Female
MYanovaFunction <- function(x) {
  anova_test <- aov(x~ pheno$Group+pheno$Female)
  return(summary(anova_test)[[1]][[5]][1])
}
pValuesAnova <- apply(beta_filt_preprocessQuantile,1,
   MYanovaFunction)
pValuesAnova
final <- data.frame(beta_filt_preprocessQuantile, pValuesAnova)
head(final)
```

Results :



```
> head(final)
           X9344737127_R01C02 X9344737127_R03C02 X9344737127_R06C01 X9344737127_R02C02
cg13869341         0.73914779         0.76950459          0.6995062          0.7184184
cg14008030         0.69185593         0.64592974          0.6885158          0.6031361
cg12045430         0.13474718         0.11699326          0.1004484          0.1218535
cg20826792         0.17078769         0.28879485          0.2947075          0.2819566
cg00381604         0.05846309         0.09592991          0.1045477          0.1000693
cg20253340         0.42958351         0.51681548          0.4396129          0.4760213
           X9376538140_R02C01 X9376538140_R05C01 X9376538140_R01C01 X9376538140_R04C01
cg13869341         0.78299289         0.77009158          0.7751171          0.8177119
cg14008030         0.63207688         0.56279678          0.6861439          0.6460894
cg12045430         0.14349114         0.12472814          0.4254814          0.1188433
cg20826792         0.22464011         0.20651747          0.3439327          0.2847663
cg00381604         0.09964109         0.08688974          0.3128018          0.1086204
cg20253340         0.46439970         0.37458736          0.4502786          0.6004477
           pValuesAnova
cg13869341   0.68847287
cg14008030   0.40103860
cg12045430   0.46427208
cg20826792   0.05315406
cg00381604   0.24584491
cg20253340   0.38870914
>
```

12. 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 after BH correction?

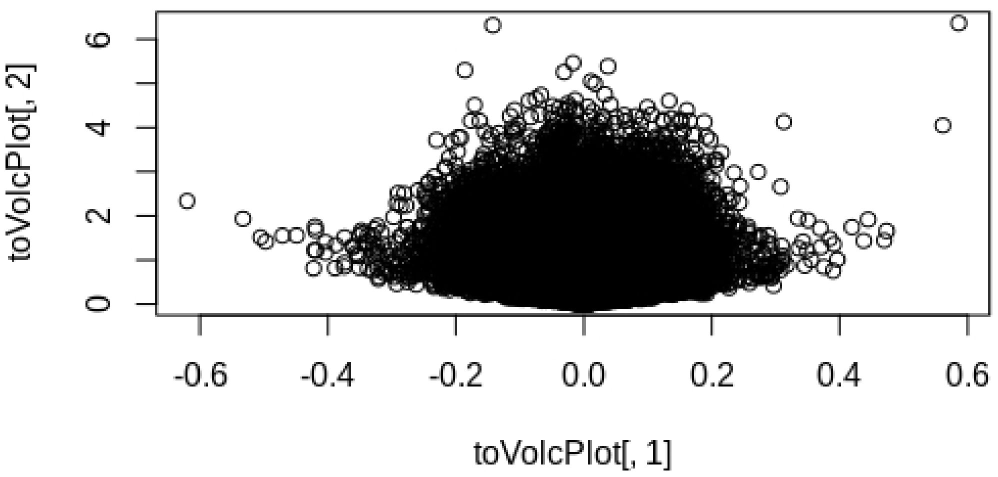
```r
 ###12###
length(pValuesAnova[pValuesAnova<=0.05])
pValuesAnovaBonf <- p.adjust(pValuesAnova,"bonferroni")
length(pValuesAnovaBonf[pValuesAnovaBonf<=0.05])
pValuesAnovaBH <- p.adjust(pValuesAnova,"BH")
```

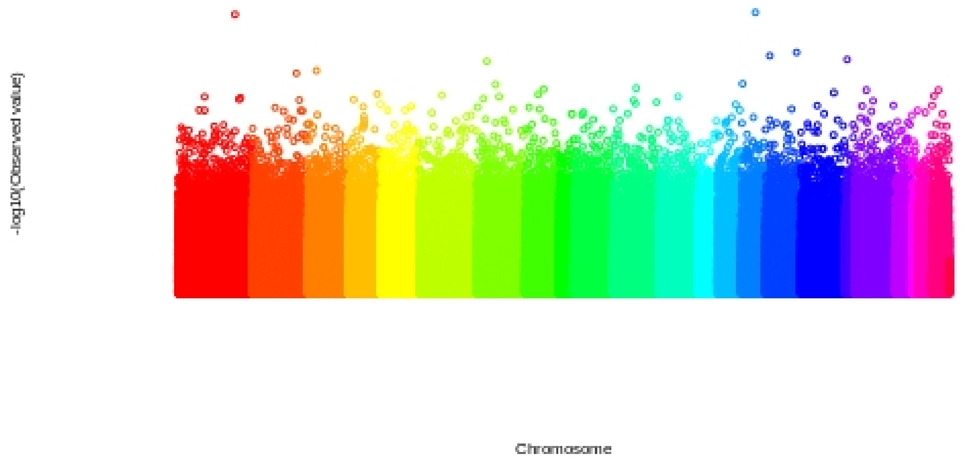
```
length(pValuesAnovaBH[pValuesAnovaBH<=0.05])
```

Probes resulted to be differentially methylated were 36693. But after both Bonferroni correction and BH correction they resulted to be 0.

13. Produce a Manhattan plot and a volcano plot of your data.

```
 # VOLCANO PLOTS
############################################
#For each probe of the microarray, we have to calculate the
   delta between mean group A values and mean group B values

beta <- final[,1:8]
beta_groupA <- beta[,pheno$Group=="A"]
mean_beta_groupA <- apply(beta_groupA,1,mean)
beta_groupB <- beta[,pheno$Group=="B"]
mean_beta_groupB <- apply(beta_groupB,1,mean)
delta <- mean_beta_groupB-mean_beta_groupA
head(delta)

toVolcPlot <- data.frame(delta, -log10(final$pValuesAnova))
head(toVolcPlot)
plot(toVolcPlot[,1], toVolcPlot[,2])

# MANHATTAN PLOTS
############################################
#manhattan plot shows significant methylation sites in the
   genome
install.packages("gap")
library(gap)
# First we have to annotate our dataframe, that is add genome
   annotation information for each cpg probe
final<- data.frame(rownames(final),final)
colnames(final)[1] <- "IlmnID"
annotated <- merge(final, Illumina450Manifest_clean,by="IlmnID")
save(annotated,file="final_annotated.RData")

db <- data.frame(annotated$CHR, annotated$MAPINFO, annotated$
   pValuesAnova)
levels(db$annotated.CHR)
db$annotated.CHR <- factor(db$annotated.CHR,levels=c("1","2","3"
   ,"4","5","6","7","8","9","10","11","12","13","14","15","16","
   17","18","19","20","21","22","X","Y"))
levels(db$annotated.CHR)
palette <- rainbow(24)
mhtplot(db,control=mht.control(colors=palette))
```

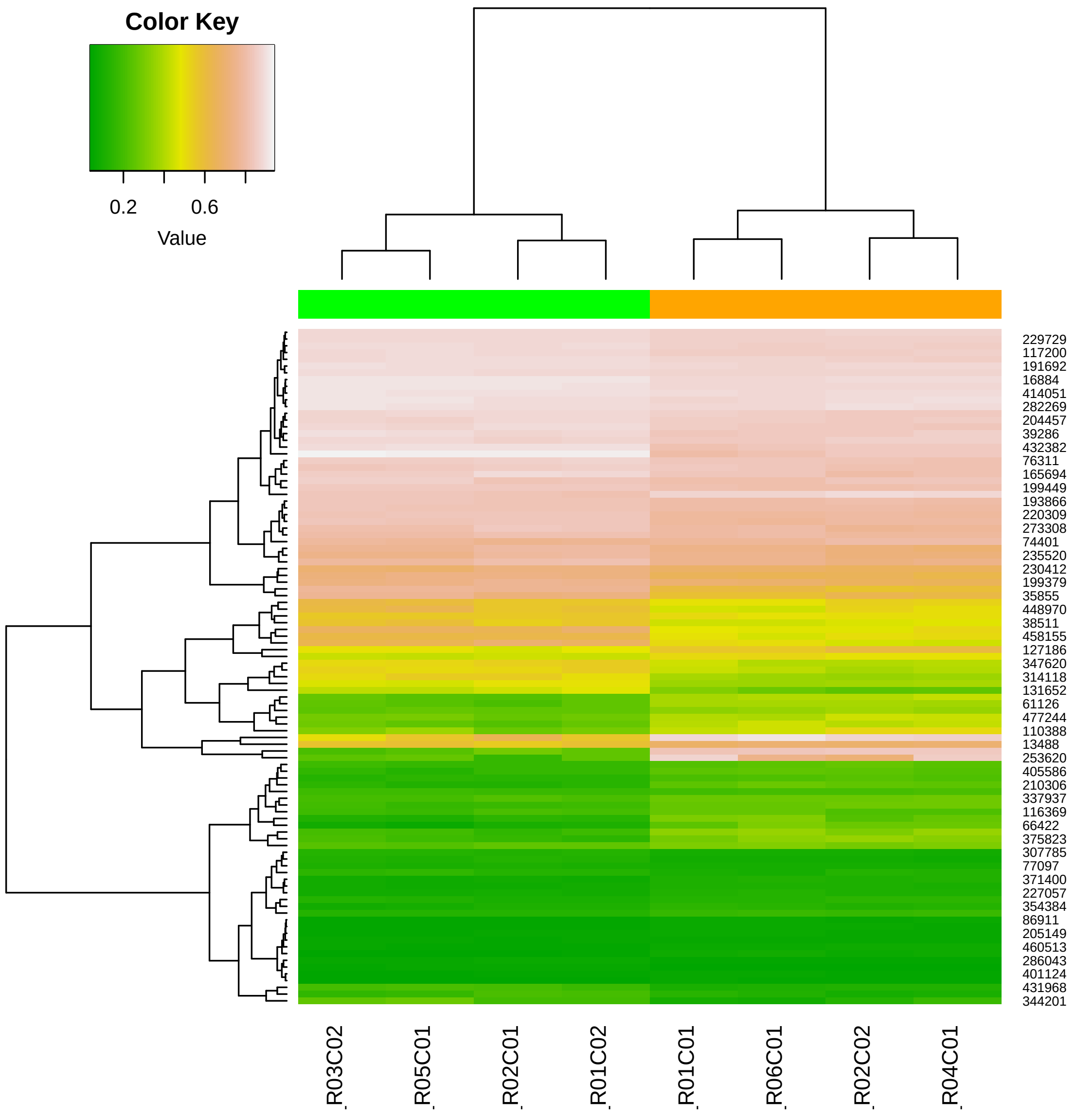




14. Produce an heatmap of the top 100 differentially mehtylated probes.

```
# HEATMAP

```

```
install.packages("gplots")
library(gplots)

# Heatmap on several probes is computationally demonading. We
    use just the top 100 most significant CpG probes
annotated <-annotated[order(annotated$pValuesAnova),]

matrix=as.matrix(annotated[1:100,2:9])
colorbar <- c("green","green","orange","orange","green","green",
    "orange","orange")
heatmap.2(matrix,col=terrain.colors(100),Rowv=T,Colv=T,
    dendrogram="both",key=T,ColSideColors=colorbar,density.info="
    none",trace="none",scale="none",symm=F)
```


Color Key
0.2
0.6
Value
229729
117200
191692
16884
414051
282269
204457
39286
432382
76311
165694
199449
193866
220309
273308
74401
235520
230412
199379
35855
448970
38511
458155
127186
347620
314118
131652
61126
477244
110388
13488
253620
405586
210306
337937
116369
66422
375823
307785
77097
371400
227057
354384
86911
205149
460513
286043
401124
431968
344201
R03C02
R05C01
R02C01
R01C02
R01C01
R06C01
R02C02
R04C01