Analytical Pipeline for the analysis and identification of DNA methylation locus with R code

Alessia Campo, Master degree in Bioinformatics, University of Bologna

SECTION 1: create an object called RGset storing the RGChannlSet object

Step 1 Set the working directory and load the minfi package to start the analysis. The .dat files storing the raw data are in the “Input_data” folder

```
setwd("/home/alessia/drd/projectdrd")
suppressMessages(library(minfi))
list.files("Input_data")
```

```
##  [1] "5775278051_R01C01_Grn.idat"  "5775278051_R01C01_Red.idat"
##  [3] "5775278051_R04C02_Grn.idat"  "5775278051_R04C02_Red.idat"
##  [5] "5775278078_R02C01_Grn.idat"  "5775278078_R02C01_Red.idat"
##  [7] "5775278078_R05C01_Grn.idat"  "5775278078_R05C01_Red.idat"
##  [9] "5775278078_R05C02_Grn.idat"  "5775278078_R05C02_Red.idat"
## [11] "5930514034_R01C02_Grn.idat"  "5930514034_R01C02_Red.idat"
## [13] "5930514035_R04C02_Grn.idat"  "5930514035_R04C02_Red.idat"
## [15] "5930514035_R06C02_Grn.idat"  "5930514035_R06C02_Red.idat"
## [17] "Samplesheet_report_2020.csv"
```

```
directory <-("Input_data")
directory
```

```
## [1] "Input_data"
```

Step 2 open the SampleSheet file containing the main information about the samples used in the analysis.

```
SAMPLE_SHEET <- read.csv("Input_data/Samplesheet_report_2020.csv",header=T, stringsAs
Factors = T)

targets <- read.metharray.sheet(directory)
```

```
## [read.metharray.sheet] Found the following CSV files:
```

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

```
targets
```

```
##   Sample_Name Group Age      Slide  Array                     Basename
## 1        1020    DS  29 5775278051 R01C01 Input_data/5775278051_R01C01
## 2        1036    DS  34 5775278051 R04C02 Input_data/5775278051_R04C02
## 3        3038    WT  46 5775278078 R02C01 Input_data/5775278078_R02C01
## 4        3042    WT  32 5775278078 R05C01 Input_data/5775278078_R05C01
## 5        3052    WT  31 5775278078 R05C02 Input_data/5775278078_R05C02
## 6        1016    DS  43 5930514034 R01C02 Input_data/5930514034_R01C02
## 7        1029    DS  32 5930514035 R04C02 Input_data/5930514035_R04C02
## 8        3029    WT  35 5930514035 R06C02 Input_data/5930514035_R06C02
```

Step 3 create an RGchannelSet object using as target the SampleSheet shown above by the use of **read.metharray.exp()** function (from minfi package) and save it in the working directory as .RData.

```
RGset1 <- read.metharray.exp(targets=targets)
summary(RGset1)
```

```
## [1] "RGChannelSet object of length 622399 with 0 metadata columns"
```

```
head(RGset1)
```

```
## class: RGChannelSet
## dim: 6 8
## metadata(0):
## assays(2): Green Red
## rownames(6): 10600313 10600322 ... 10600345 10600353
## rowData names(0):
## colnames(8): 5775278051_R01C01 5775278051_R04C02 ... 5930514035_R04C02
##   5930514035_R06C02
## colData names(7): Sample_Name Group ... Basename filenames
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
```

```
save(RGset1, file="RGset1.RData")
```

SECTION 2: Store the Red and Green fluorescences in two different datframes

Step 1 create two new dataframes to store the RED and GREEN fluorescences respectively

```
load("RGset1.RData")
Red_dataf <- data.frame(getRed(RGset1))
Green_dataf <- data.frame(getGreen(RGset1))
dim(Green_dataf)
```

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

Step 2 apply the head() function on the Red fluorescences dataframe to get an overview of the data and the way in which they are organized (the head() function will be used several times in these pipeline to visualize

objects)

```
head(Red_dataf)
```

```
##          X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01
## 10600313                816               1055                603
## 10600322               2269               2965               1840
## 10600328               1673               2500               1420
## 10600336              18318              21740              16924
## 10600345               4141               4575               3198
## 10600353               1433               1680               1228
##          X5775278078_R05C01 X5775278078_R05C02 X5930514034_R01C02
## 10600313                647                583                441
## 10600322               2103               2228               1383
## 10600328               1588               1600               3456
## 10600336              17981              16158              16169
## 10600345               3123               3233               3424
## 10600353               1211               1141               1164
##          X5930514035_R04C02 X5930514035_R06C02
## 10600313                463                606
## 10600322               1823               2315
## 10600328               2773               2540
## 10600336              18221              19048
## 10600345               3336               3595
## 10600353                975               1136
```

do the same for the Green fluorescences dataframe

```
head(Green_dataf)
```

```
##          X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01
## 10600313                373                494                278
## 10600322               7413              10620               6667
## 10600328               2369               2375               1995
## 10600336               1734               2060               1456
## 10600345               3400               4416               3590
## 10600353               4233               4873               3790
##          X5775278078_R05C01 X5775278078_R05C02 X5930514034_R01C02
## 10600313                360                321                210
## 10600322               8362               7963               6896
## 10600328               2179               2401               2954
## 10600336               1775               1757               1152
## 10600345               3665               3559               2975
## 10600353               3406               3693               2973
##          X5930514035_R04C02 X5930514035_R06C02
## 10600313                198                350
## 10600322               8344               9709
## 10600328               3222               2806
## 10600336               1702               1617
## 10600345               3516               4530
## 10600353               3371               3983
```

SECTION 3: starting from the Address

46801437 check what are the Red and Green fluorescence intensities for the address in the Red and Green dataframes just created.

Step 1 the address names in the dataframes correspond to the row names, so it is possible to obtain and visualize the intesities values corresponding to the adress across the samples in the following way:

```
Red_dataf[rownames(Red_dataf)=="46801437",]
```

```
##          X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01
## 46801437              14170              17209              13306
##          X5775278078_R05C01 X5775278078_R05C02 X5930514034_R01C02
## 46801437              14021              13331              15231
##          X5930514035_R04C02 X5930514035_R06C02
## 46801437              18049              17677
```

```
Green_dataf[rownames(Green_dataf)=="46801437",]
```

```
##          X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01
## 46801437               1309               1757               1162
##          X5775278078_R05C01 X5775278078_R05C02 X5930514034_R01C02
## 46801437               1347               1194               1022
##          X5930514035_R04C02 X5930514035_R06C02
## 46801437               1232               1290
```

Step 2 check what is the infinium chemestry of the probe for the same address: knowing that the Type I probes are related to both adress A and B and Type II probes are related only to a single address A we can firstly search for the Adress B in the Illumina 450K Manifest and obtain the following result:

```
load("Illumina450Manifest_clean.RData")
Illumina450Manifest_clean[Illumina450Manifest_clean$AddressB_ID=="46801437",]
```

```
0 rows | 1-7 of 33 columns
```

then search for the Adress A

```
Illumina450Manifest_clean[Illumina450Manifest_clean$AddressA_ID=="46801437",]
```

```
##            IlmnID       Name AddressA_ID
## 134322 cg21308020 cg21308020    46801437
##                                       AlleleA_ProbeSeq AddressB_ID
## 134322 AATCCTCTACTAACAAATCRAACTCAATATCCCCATTCCCTATTTTCTCC
##        AlleleB_ProbeSeq Infinium_Design_Type Next_Base Color_Channel
## 134322                                   II
##
Forward_Sequence
## 134322 AGGCAGCGGTGAGTCCTCTGCTAGCAGATCGGGCTCAATATCCCCATTCCCTGTTTTCTC[CG]GGCCCGCCCTC
CGCCTCTCAGGCGGCCGCCATGAAGATCCTCTGCCGCGGCTGCAGCCGG
##        Genome_Build CHR  MAPINFO
## 134322           37   4 99182242
##                                                SourceSeq Chromosome_36
## 134322 GTCCTCTGCTAGCAGATCGGGCTCAATATCCCCATTCCCTGTTTTCTCCG             4
##        Coordinate_36 Strand Probe_SNPs Probe_SNPs_10 Random_Loci Methyl27_Loci
## 134322      99401265      R               rs79567852          NA            NA
##                                         UCSC_RefGene_Name
## 134322 RAP1GDS1;RAP1GDS1;RAP1GDS1;RAP1GDS1;RAP1GDS1;RAP1GDS1
##                                                                UCSC_RefGene_Accession
## 134322 NM_001100430;NM_001100428;NM_001100429;NM_001100426;NM_021159;NM_001100427
##                                      UCSC_RefGene_Group  UCSC_CpG_Islands_Name
## 134322 TSS1500;TSS1500;TSS1500;TSS1500;TSS1500;TSS1500 chr4:99181509-99183199
##        Relation_to_UCSC_CpG_Island Phantom DMR Enhancer          HMM_Island
## 134322                      Island                  NA 4:99400416-99402464
##        Regulatory_Feature_Name Regulatory_Feature_Group DHS
## 134322     4:99182041-99182530      Promoter_Associated  NA
```

from this output it is clear that the probe is an Infinium Type II and no color needs to be specified

Step 3 Starting from the information retrieved in step 1 and 2, fill the table with the Red and Green fluorescence intensities and the type of probe

```
table <- matrix(c(14170 ,1309,"II", 17209,1757,"II", 13306,1162,"II", 14021,1347, "I
I",13331,1194   ,"II",15231, 1022,"II", 18049, 1232,"II",17677,
1290, "II"), ncol=3, byrow=T)
rownames(table) <- SAMPLE_SHEET$Basename
colnames(table) <- c("RED_fluor", "GREEN_fluor", "TYPE")
table <-as.table(table)
table
```

```
##                   RED_fluor GREEN_fluor TYPE
## 5775278051_R01C01 14170     1309        II
## 5775278051_R04C02 17209     1757        II
## 5775278078_R02C01 13306     1162        II
## 5775278078_R05C01 14021     1347        II
## 5775278078_R05C02 13331     1194        II
## 5930514034_R01C02 15231     1022        II
## 5930514035_R04C02 18049     1232        II
## 5930514035_R06C02 17677     1290        II
```

SECTION 4: create an MSet.raw object to extract methylatation and unmethylation

signals from the Red/Green fluorescences set and save it as .RData file

Step 1 Use the function **preprocessRaw()** to create the methyl set object containing the Beta and M values and save it as .RData file

```
MSet1.raw <- preprocessRaw(RGset1)
```

```
## Loading required package: IlluminaHumanMethylation450kmanifest
```

```
MSet1.raw
```

```
## class: MethylSet
## dim: 485512 8
## metadata(0):
## assays(2): Meth Unmeth
## rownames(485512): cg00050873 cg00212031 ... ch.22.47579720R
##   ch.22.48274842R
## rowData names(0):
## colnames(8): 5775278051_R01C01 5775278051_R04C02 ... 5930514035_R04C02
##   5930514035_R06C02
## colData names(7): Sample_Name Group ... Basename filenames
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
## Preprocessing
##   Method: Raw (no normalization or bg correction)
##   minfi version: 1.34.0
##   Manifest version: 0.4.0
```

```
dim(MSet1.raw)
```

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

```
save(MSet1.raw, file="MSet1.RData")
```

SECTION 5: perform quality checks

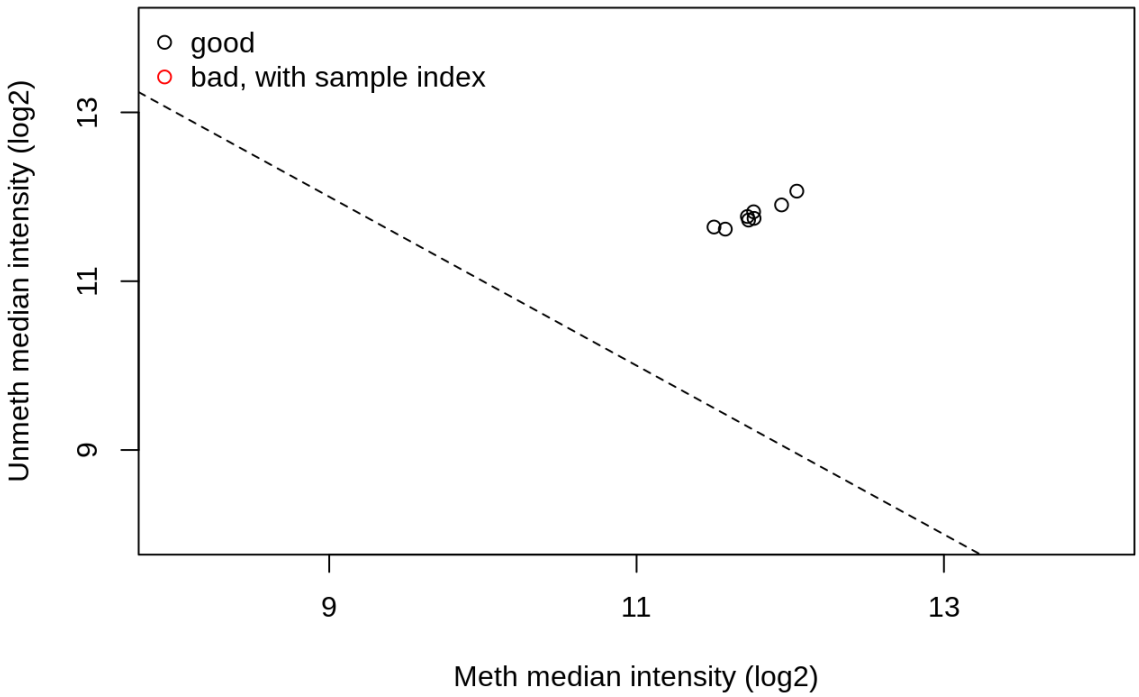
Step 1 in order to estimate the sample-specific quality control extract the median values for the methylation and unmethylation channels for each sample using the function **getQC()**

```
qc_medians <- getQC(MSet1.raw)
qc_medians
```

```
## DataFrame with 8 rows and 2 columns
##                         mMed      uMed
##                    <numeric> <numeric>
## 5775278051_R01C01   11.7616   11.8222
## 5775278051_R04C02   12.0427   12.0668
## 5775278078_R02C01   11.5774   11.6170
## 5775278078_R05C01   11.7645   11.7444
## 5775278078_R05C02   11.7288   11.7241
## 5930514034_R01C02   11.5038   11.6416
## 5930514035_R04C02   11.7211   11.7661
## 5930514035_R06C02   11.9436   11.9035
```

With the median values obtained build a QCplot

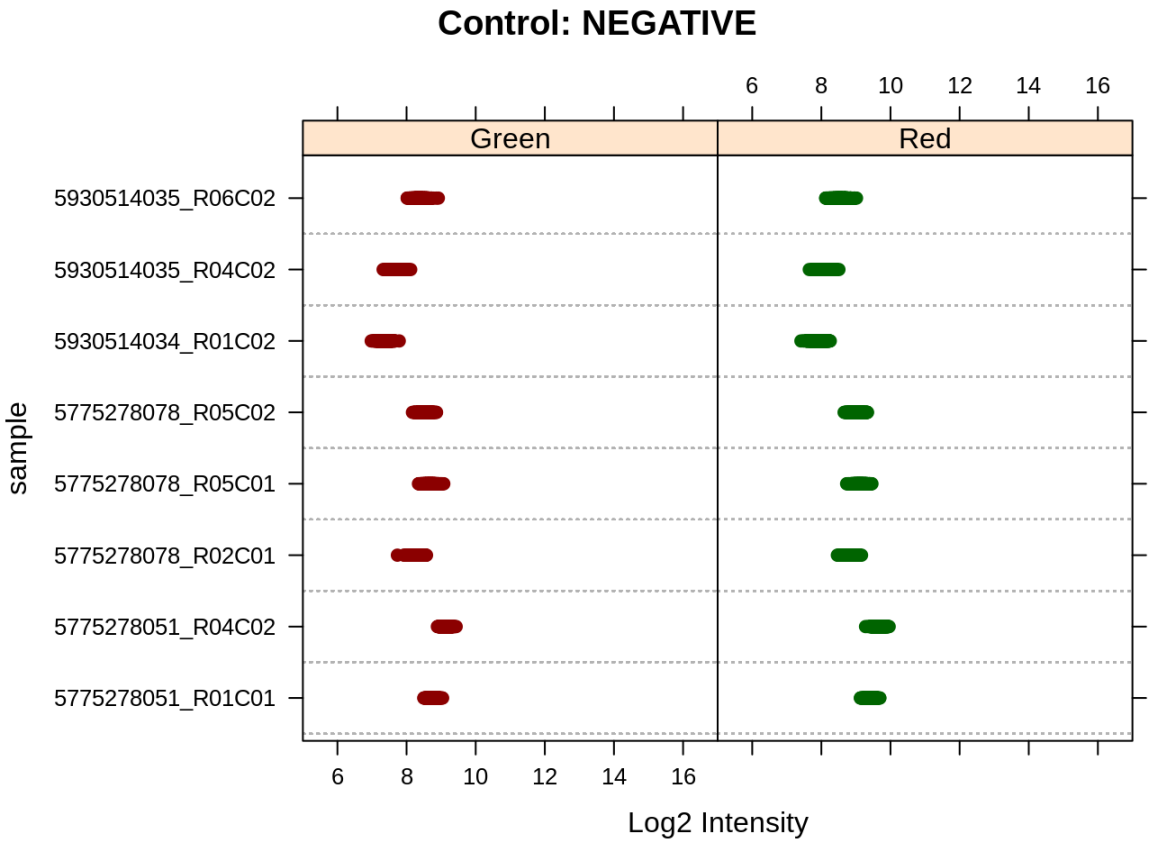
```
plotQC(qc_medians)
```



the QCplot reports high median values for methylation and unmethylation signals: all the points are above the diagonal and they can be rated as being good quality samples

Step 2 Among all the controls available in the array experiment, check and plot the negative control probe intensities from the RGset using the function **controlStripPlot()**

```
controlStripPlot(RGset1, controls ="NEGATIVE")
```



The first thing to note is that green and red colors are inverted in the strip plot probably due to a minfi package issue. However the correct analysis can be fixed by considering the green and red labels to be correct and the colors to be inverted. The instensities are reported in the log2 scale and both for red and green intensities they lie between 7 and 10 in the background signal range (below 10) indicating that negative control are all fine.

Step 3 for each sample calculate the detection p-values of each probe from the RGset using the function **detectionP()** and check how many of them are above the **threshold (=0.05)** to assess the failed positions

```
detP1 <- detectionP(RGset1)
head(detP1)
```

```
##            5775278051_R01C01 5775278051_R04C02 5775278078_R02C01
## cg00050873       0.00000e+00      0.000000e+00      0.000000e+00
## cg00212031       0.00000e+00      0.000000e+00      0.000000e+00
## cg00213748      4.06658e-113      1.139431e-84     3.522029e-195
## cg00214611       0.00000e+00      0.000000e+00      0.000000e+00
## cg00455876       0.00000e+00     8.148885e-301      0.000000e+00
## cg01707559       0.00000e+00      0.000000e+00      0.000000e+00
##            5775278078_R05C01 5775278078_R05C02 5930514034_R01C02
## cg00050873      0.000000e+00        0.0000e+00                 0
## cg00212031      0.000000e+00        0.0000e+00                 0
## cg00213748      5.119634e-63        1.7166e-87                 0
## cg00214611      0.000000e+00        0.0000e+00                 0
## cg00455876     1.307576e-275        0.0000e+00                 0
## cg01707559      0.000000e+00        0.0000e+00                 0
##            5930514035_R04C02 5930514035_R06C02
## cg00050873      0.000000e+00      0.000000e+00
## cg00212031      0.000000e+00      0.000000e+00
## cg00213748     4.314061e-227     2.379035e-197
## cg00214611      0.000000e+00      0.000000e+00
## cg00455876      0.000000e+00      0.000000e+00
## cg01707559      0.000000e+00      0.000000e+00
```

Determine how many failed positions occur for each sample

```
failed_pos <- detP1>0.05
summary(failed_pos)
```

```
##  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 results obtained fill a table which reports the number of failed positions for each sample

```
failed_tab <- matrix(c(247,210,264,413,385,91,46,115), ncol=1, byrow=T)
rownames(failed_tab) <- SAMPLE_SHEET$Basename
colnames(failed_tab) <- c("Failed positions")
failed_tab <-as.table(failed_tab)
failed_tab
```

```
##                   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
```

In addition the percentage of failed position (those position above the threshold 0.05) for each sample can be computed with the function **colMeans()** that calculates the ratio between failed positions (TRUE) and the total number of positions (TRUE+FALSE)

```
failed_within_sample <- colMeans(failed_pos)
failed_within_sample
```

```
## 5775278051_R01C01 5775278051_R04C02 5775278078_R02C01 5775278078_R05C01
##      5.087413e-04      4.325331e-04      5.437559e-04      8.506484e-04
## 5775278078_R05C02 5930514034_R01C02 5930514035_R04C02 5930514035_R06C02
##      7.929773e-04      1.874310e-04      9.474534e-05      2.368634e-04
```

Using the function **rowMeans()** it is possibile to check if there are failed positions (percentage) across all the samples for a given probe

```
failed_between_samples <- rowMeans(failed_pos)
head(failed_between_samples)
```

```
## cg00050873 cg00212031 cg00213748 cg00214611 cg00455876 cg01707559
##          0          0          0          0          0          0
```

check in how many samples the number of failed positions is less than 5% and decide to retain them

```
samples_retained <- failed_within_sample<0.05
length(samples_retained)
```

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

8/8 samples have failed positions below the 5% and they could be retained

check how many probes failed in less than 5% of samples

```
probes_retained <- failed_between_samples<0.05
summary(probes_retained)
```

```
##    Mode   FALSE    TRUE
## logical    1118  484394
```

NOTE:these last operations are useful in filtering procedures to manage restricted but more reliable sets of samples and probes for specific purposes. However in the whole procedure will be used the orginal set without restrictions

SECTION 6: Compute Beta and M values. Subsequently build and assess the density distributions plot for both M and Beta values

Step 1 extract the raw BETA from the MSet.raw with the **getBeta()** function

```
load("MSet1.RData")
beta_values <- getBeta(MSet1.raw)
head(beta_values)
```

```
##            5775278051_R01C01 5775278051_R04C02 5775278078_R02C01
## cg00050873        0.89768977        0.88526903        0.91002063
## cg00212031        0.06793670        0.08674123        0.05236115
## cg00213748        0.78107046        0.75359665        0.85311119
## cg00214611        0.05653951        0.07416197        0.04504615
## cg00455876        0.79688897        0.79345392        0.81265881
## cg01707559        0.06095205        0.06598391        0.06432003
##            5775278078_R05C01 5775278078_R05C02 5930514034_R01C02
## cg00050873        0.85944474        0.89039520        0.89226078
## cg00212031        0.05221462        0.08168594        0.02673267
## cg00213748        0.87509294        0.82262774        0.85504886
## cg00214611        0.06380711        0.04375164        0.03280879
## cg00455876        0.77596240        0.78732426        0.83219021
## cg01707559        0.07188831        0.07842939        0.07693851
##            5930514035_R04C02 5930514035_R06C02
## cg00050873        0.87229692        0.88248526
## cg00212031        0.05225601        0.07023848
## cg00213748        0.81156222        0.85151515
## cg00214611        0.04829622        0.04752655
## cg00455876        0.87056226        0.87620235
## cg01707559        0.03453142        0.05274206
```

Do the same for the raw M values using the function **getM()**

```
M_values <- getM(MSet1.raw)
summary(M_values)
```

```
##  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
```

NOTE: in the summary version there are min values equal to -Inf when the methylation value is equal to 0 and unmethylation value is >0 and max values equal to +Inf when methylation value >0 and unmethylation value =0

Step 2 Consider the two groups WT and DS associated with the sample basename reported in the original SampleSheet:

```
BASENAME_GROUP <- SAMPLE_SHEET[,c(2,6)]
BASENAME_GROUP
```

```
##   Group          Basename
## 1    DS 5775278051_R01C01
## 2    DS 5775278051_R04C02
## 3    WT 5775278078_R02C01
## 4    WT 5775278078_R05C01
## 5    WT 5775278078_R05C02
## 6    DS 5930514034_R01C02
## 7    DS 5930514035_R04C02
## 8    WT 5930514035_R06C02
```

Create two different sets containing the beta values according to the group classification of samples

```
DS <- SAMPLE_SHEET$Group=="DS"
DS_betavalue <- beta_values[,DS]
head(DS_betavalue)
```

```
##            5775278051_R01C01 5775278051_R04C02 5930514034_R01C02
## cg00050873         0.89768977         0.88526903         0.89226078
## cg00212031         0.06793670         0.08674123         0.02673267
## cg00213748         0.78107046         0.75359665         0.85504886
## cg00214611         0.05653951         0.07416197         0.03280879
## cg00455876         0.79688897         0.79345392         0.83219021
## cg01707559         0.06095205         0.06598391         0.07693851
##            5930514035_R04C02
## cg00050873         0.87229692
## cg00212031         0.05225601
## cg00213748         0.81156222
## cg00214611         0.04829622
## cg00455876         0.87056226
## cg01707559         0.03453142
```

```
WT <- SAMPLE_SHEET$Group=="WT"
WT_betavalue <- beta_values[, WT]
head(WT_betavalue)
```

```
##            5775278078_R02C01 5775278078_R05C01 5775278078_R05C02
## cg00050873         0.91002063         0.85944474         0.89039520
## cg00212031         0.05236115         0.05221462         0.08168594
## cg00213748         0.85311119         0.87509294         0.82262774
## cg00214611         0.04504615         0.06380711         0.04375164
## cg00455876         0.81265881         0.77596240         0.78732426
## cg01707559         0.06432003         0.07188831         0.07842939
##            5930514035_R06C02
## cg00050873         0.88248526
## cg00212031         0.07023848
## cg00213748         0.85151515
## cg00214611         0.04752655
## cg00455876         0.87620235
## cg01707559         0.05274206
```

Do the same for the M values:

```
DS_Mvalue <- M_values[,DS]
head(DS_Mvalue)
```

```
##            5775278051_R01C01 5775278051_R04C02 5930514034_R01C02
## cg00050873          3.133267          2.947861          3.049922
## cg00212031         -3.778165         -3.396234         -5.186160
## cg00213748          1.834986          1.612771          2.560440
## cg00214611         -4.060631         -3.642008         -4.881647
## cg00455876          1.972110          1.941683          2.310086
## cg01707559         -3.945452         -3.823261         -3.584649
##            5930514035_R04C02
## cg00050873          2.772026
## cg00212031         -4.180829
## cg00213748          2.106614
## cg00214611         -4.300531
## cg00455876          2.749689
## cg01707559         -4.805248
```

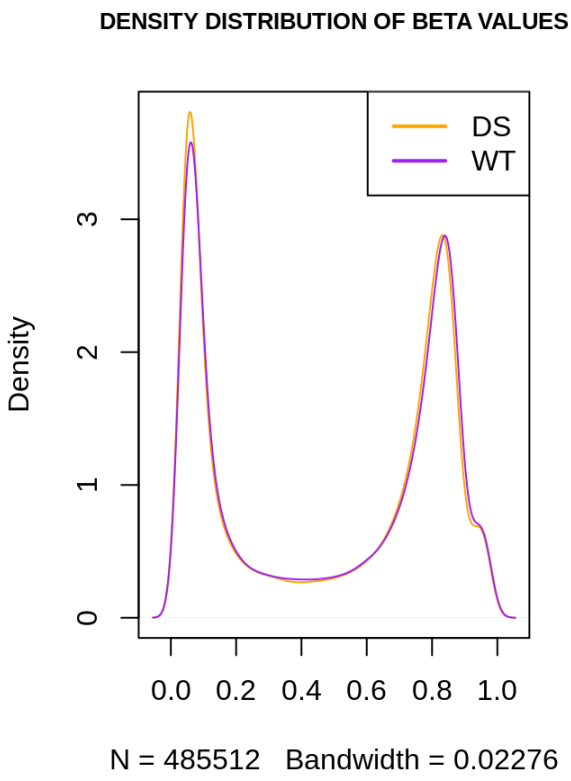
```
WT_Mvalue <- M_values[,WT]
head(WT_Mvalue)
```

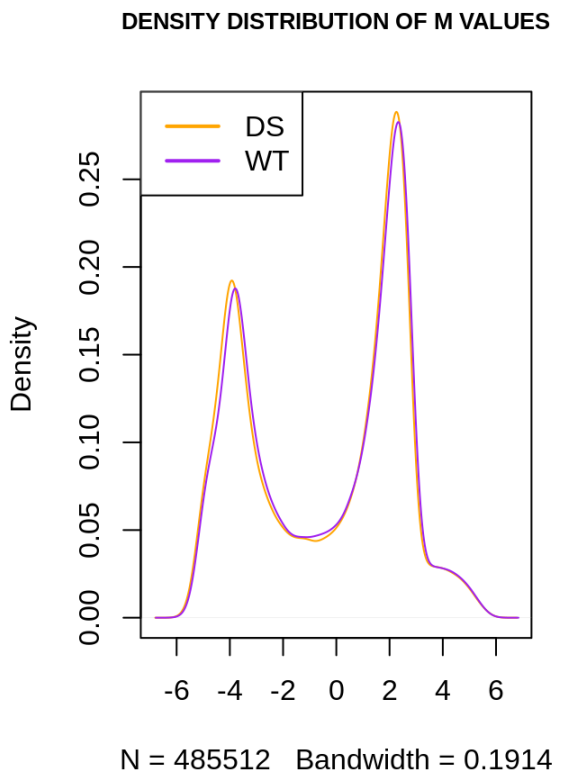
```
##            5775278078_R02C01 5775278078_R05C01 5775278078_R05C02
## cg00050873          3.338233          2.612267          3.022135
## cg00212031         -4.177769         -4.182035         -3.490828
## cg00213748          2.538009          2.808581          2.213459
## cg00214611         -4.405955         -3.875017         -4.449976
## cg00455876          2.116982          1.792246          1.888302
## cg01707559         -3.862675         -3.690469         -3.554628
##            5930514035_R06C02
## cg00050873          2.908730
## cg00212031         -3.726527
## cg00213748          2.519716
## cg00214611         -4.324873
## cg00455876          2.823280
## cg01707559         -4.166731
```

Step 3 Calculate the mean for both M and Beta values across the samples for WT and DS groups

```
mean_beta_DS <- apply(DS_betavalue,1, mean,na.rm=T)
mean_M_DS <- apply(DS_Mvalue,1,mean,na.rm=T)

mean_beta_WT <- apply(WT_betavalue,1,mean,na.rm=T)
mean_M_WT <- apply(WT_Mvalue,1,mean,na.rm=T)
```

Step 4 calculate the kernel density distribution for the beta mean values using the **density()** function and plot them. Do the same for the M means values.





It is possbile to note that the beta value distribution and the M value distribution follow a similar shape for both DS and WT samples. The M distribution clearly follows a typical bimodal distribution with negative values for the unmethylated mode and positive values for methylated mode, while the range of beta value is between O and 1 and it is more intuitive because it is directly related to the level (percenage) of methylation for a specific site.

SECTION 7: Normalization methods: compare raw data and normalized data according to the infinium chemestry (I, II)

Step 1 Firstly load the Illumina 450k Manifest and extract the type I and II probes.

```
load("Illumina450Manifest_clean")

typeI <- Illumina450Manifest_clean[Illumina450Manifest_clean$Infinium_Design_Type=="
I",]
typeI <- droplevels(typeI)
dim(typeI)
```

```
## [1] 135476     33
```

```
typeII <-  Illumina450Manifest_clean[Illumina450Manifest_clean$Infinium_Design_Type=
="II",]
typeII <- droplevels(typeII)
dim(typeII)
```

```
## [1] 350036     33
```

Step 2 Use the probe names extracted to divide the beta values according to the chemestry (Type I and Type II) mapping them into the dataframe containing the name of the probes and the beta values

```
beta_value_I <- beta_values[rownames(beta_values) %in% typeI$IlmnID,]
dim(beta_value_I)
```

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

```
beta_value_II <- beta_values[rownames(beta_values) %in% typeII$IlmnID,]
dim(beta_value_II)
```

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

This operation returns two matrices containing the beta values for each probe in the different samples separately for type I and II probes

Step 3 Compute the mean and the standard deviation for raw beta values of type I and II probes and the kernel densities

```
mean_beta_I<- apply(beta_value_I, 1,mean, na.rm=T)
mean_beta_II <- apply(beta_value_II,1,mean, na.rm=T)

sd_beta_I <- apply(beta_value_I,1,sd,na.rm=T)
sd_beta_II <- apply(beta_value_II,1,sd,na.rm=T)


den_mean_beta_I <- density(mean_beta_I)
den_mean_beta_II <- density(mean_beta_II)

den_sd_beta_I <- density(sd_beta_I)
den_sd_beta_II <- density(sd_beta_II)
```

Step 4 Normalise the beta values with the between-array normalisation method preprocessFunnorm()

```
load("RGset1.RData")
preprocessFunnorm_res <- preprocessFunnorm(RGset1)
```

```
## [preprocessFunnorm] Background and dye bias correction with noob
```

```
## Loading required package: IlluminaHumanMethylation450kanno.ilmn12.hg19
```

```
## [preprocessFunnorm] Mapping to genome
```

```
## [preprocessFunnorm] Quantile extraction
```

```
## Warning in .getSex(CN = CN, xIndex = xIndex, yIndex = yIndex, cutoff = cutoff):
## An inconsistency was encountered while determining sex. One possibility is
## that only one sex is present. We recommend further checks, for example with the
## plotSex function.
```

```
## [preprocessFunnorm] Normalization
```

```
preprocessFunnorm_res
```

```
## class: GenomicRatioSet
## dim: 485512 8
## metadata(0):
## assays(2): Beta CN
## rownames(485512): cg13869341 cg14008030 ... cg08265308 cg14273923
## rowData names(0):
## colnames(8): 5775278051_R01C01 5775278051_R04C02 ... 5930514035_R04C02
##   5930514035_R06C02
## colData names(10): Sample_Name Group ... yMed predictedSex
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
## Preprocessing
##   Method: NA
##   minfi version: NA
##   Manifest version: NA
```

Step 5 get the normalised beta values from the Genomic Ratio Set object and divide them accoring to the probe chemestry as done before

```
norm_beta_values<- getBeta(preprocessFunnorm_res)
dim(norm_beta_values)
```

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

```
norm_beta_I <- norm_beta_values[rownames(norm_beta_values) %in% typeI$IlmnID,]
norm_beta_II <- norm_beta_values[rownames(norm_beta_values) %in% typeII$IlmnID,]
dim(norm_beta_I)
```

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

```
dim(norm_beta_II)
```

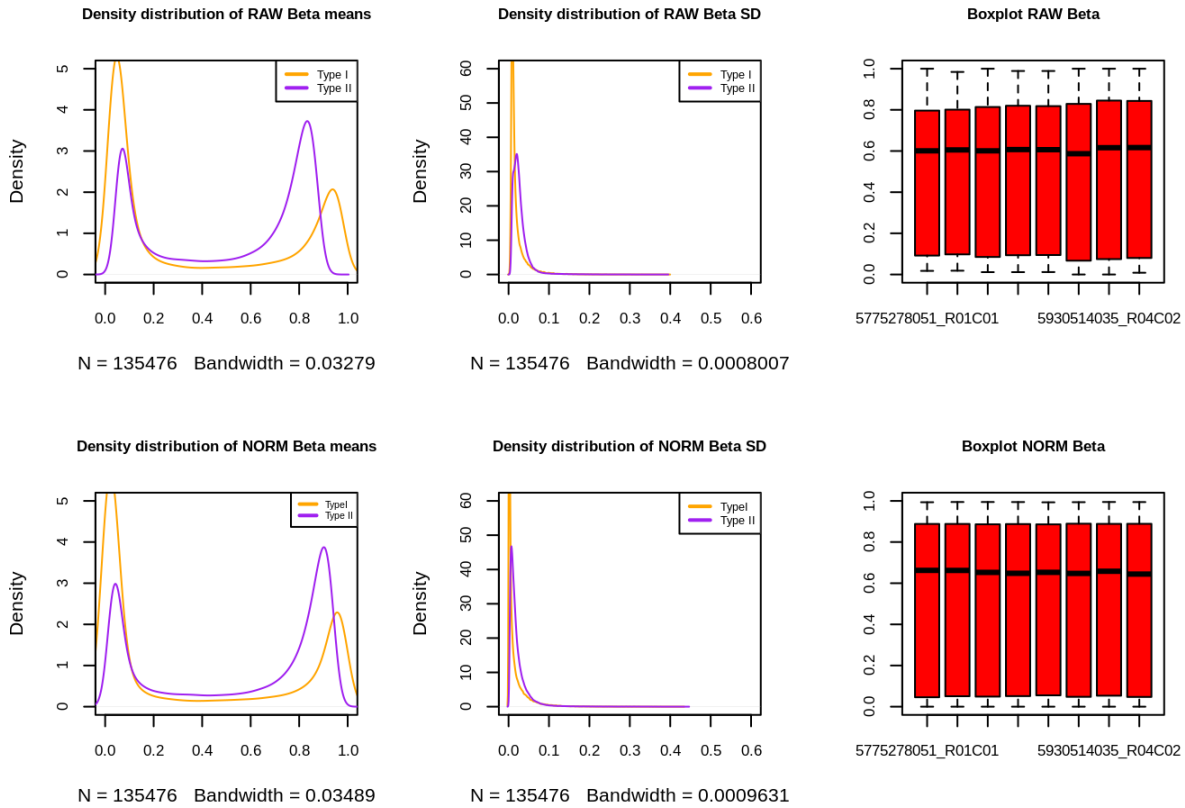
```
## [1] 350036      8
```

Step 5 compute the means, the standard deviations and the kernel density of the normilised beta values

```
norm_mean_beta_I <- apply(norm_beta_I,1 , mean)
norm_mean_beta_II <- apply(norm_beta_II,1,mean)

norm_sd_beta_I <- apply(norm_beta_I,1, sd, na.rm=T)
norm_sd_beta_II <- apply(norm_beta_II,1, sd, na.rm=T)

den_norm_mean_beta_I <- density(norm_mean_beta_I, na.rm=T)
den_norm_mean_beta_II <- density(norm_mean_beta_II, na.rm=T)

norm_den_sd_beta_I <- density(norm_sd_beta_I,na.rm=T)
norm_den_sd_beta_II <- density(norm_sd_beta_II,na.rm=T)
```

Step 6 Create a plot for the density distributions of the means and standard deviations and the boxplots for both raw and normalised data according to the probe chemestry

```
{par(mfrow=c(2,3))
plot(den_mean_beta_I, col="orange", main="Density distribution of RAW Beta means", ce
x.main=0.8, cex.axis=0.8, xlim=c(0,1),ylim=c(0,5))
lines(den_mean_beta_II, col="purple")
legend("topright", legend=c("Type I", "Type II"),col=c("orange","purple"),lwd=2,cex=
0.6)
plot(den_sd_beta_I, col="orange", main="Density distribution of RAW Beta SD", cex.mai
n=0.8,cex.axis=0.8, xlim=c(0,0.6), ylim=c(0,60))
lines(den_sd_beta_II, col="purple")
legend("topright", legend=c("Type I","Type II"), col=c("orange","purple"), lwd=2, cex
=0.6)
boxplot(beta_values, main="Boxplot RAW Beta",cex.main=0.8,cex.axis=0.8,col="red", yli
m=c(0,1))

plot(den_norm_mean_beta_I, col="orange", main="Density distribution of NORM Beta mean
s", cex.main=0.8, cex.axis=0.8, xlim=c(0,1), ylim=c(0,5))
lines(den_norm_mean_beta_II, col="purple")
legend("topright", legend=c("TypeI", "Type II"), col=c("orange","purple"),lwd=2,cex=
0.5)
plot(norm_den_sd_beta_I,col="orange", main="Density distribution of NORM Beta SD",ce
x.main=0.8,cex.axis=0.8, xlim=c(0,0.6), ylim=c(0,60))
lines(norm_den_sd_beta_II,col="purple")
legend("topright", legend=c("TypeI","Type II"), col=c("orange", "purple"), lwd=2, cex
=0.6)
boxplot(norm_beta_values, main="Boxplot NORM Beta", col="red",cex.main=0.8,cex.axis=
0.8, ylim=c(0,1))}
```



there some differences between raw and normalized data: the mean density distribution for raw data shows a higher distance between the two peaks of methylation mode for Type I and II, indeed the Type II peak is shifted

to the centre respect to the one of the Type I. In the normalized version they are more overlapped among each other; the SD density distribution for the raw data shows a higher peak for the Type II probes and a shorter peak for type I. However, this trend can be also observered in the normalised SD distribution. The decisive evaluation of the differences between raw and normalised data can be given by the inspection of the boxplots. The raw data boxplot shows a larger variability respect to the mean in some samples, instead the normalised data boxplot shows less variability among the samples. This observation confirms that the normalisation procedure is often necessary to reduce the variability of the dataset.

SECTION 8: perform Principal Component Analysis (PCA) on the normalised beta matrix generated in the previous step (Section 4; step 2) to inspect the presence of possible outliers

Step 1 The function **prcomp()** performs principal component analysis in a trasposed matrix of values, in this case the traspose matrix is obtained using the function **t()** on normalised beta values

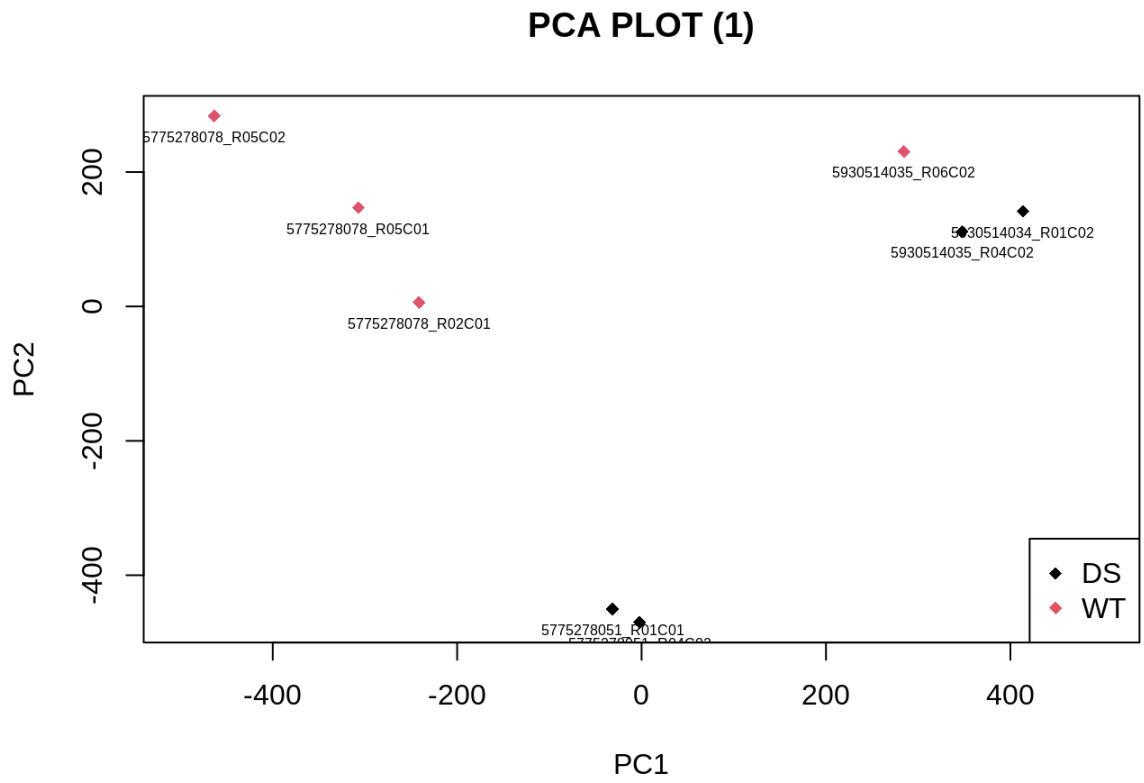
```
PCA <- prcomp(t(norm_beta_values), scale=T)
summary(PCA)
```

```
## Importance of components:
##                             PC1      PC2      PC3      PC4      PC5      PC6
## Standard deviation     325.3972 295.4418 260.8247 248.7267 237.9097 231.3976
## Proportion of Variance   0.2181   0.1798   0.1401   0.1274   0.1166   0.1103
## Cumulative Proportion    0.2181   0.3979   0.5380   0.6654   0.7820   0.8923
##                             PC7       PC8
## Standard deviation     228.6972 6.446e-12
## Proportion of Variance   0.1077 0.000e+00
## Cumulative Proportion    1.0000 1.000e+00
```

The function returns the proportion of the variability covered by each component and the cumuative percentage.The result suggests that the variability among the samples is described in a high-dimensionality data space since most of the variability is covered by more than two components (from PC1 to PC5-> 78% of variability)

Step 2 Plot the first two components considering the two phenogroup associated to the sample used in the experiment

```
{PCA$x

plot(PCA$x[,1], PCA$x[,2], cex=1, pch=18, ylab="PC2",xlab="PC1", col=SAMPLE_SHEET$Gro
up
      , xlim=c(-500,500), main="PCA PLOT (1)")
legend("bottomright",legend=levels(SAMPLE_SHEET$Group), col=c(1:nlevels(SAMPLE_SHEE
T$Group)), pch = 18)
  text(PCA$x[,1],PCA$x[,2], labels=rownames(PCA$x), pos=1, cex=0.5)}
```



The PCA plot shows that WT and DS are quite well separated and there is no presence of outliers.

NOTE: the PCA is also useful to detect the presence of some batch effects in the experiment. This type of assessment can be done following the same procedure but considering the slides as source of variability to build the plot.

SECTION 9: Identification of differentially methylated loci between the two phenogroup DS and WT

Step 1 Apply the **wilcox.test()** function that allow to apply the M-W non-parametric statistic test.It retunrs the p-values extracted from the beta value distribution of each probe across the samples

```
two_groups <- SAMPLE_SHEET$Group
MW_FUN <- function(x) {
  MW_test <- wilcox.test(x~two_groups)
  return(MW_test$p.value)}

MW_pvalues <- apply(norm_beta_values,1,MW_FUN)
summary(MW_pvalues)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02857 0.20000 0.48571 0.52682 0.88571 1.00000
```

Step 2 Rearrange the output building a dataframe which will contain all the normalised beta values and the associated p-values computed by the M-W test for each locus across the samples. The data can be ordered

according to p-values

```
pvalue_df <- data.frame(norm_beta_values, MW_pvalues)
pvalues_df_ordered <- pvalue_df[order(pvalue_df$MW_pvalues),]
head(pvalues_df_ordered)
```

```
##            X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01
## cg11954957          0.8237170          0.8052609          0.7743885
## cg16736630          0.2895560          0.2267553          0.5232336
## cg15903280          0.5497174          0.4187877          0.6195478
## cg09856436          0.5179173          0.5102878          0.5295284
## cg13856810          0.8574653          0.8461140          0.7951753
## cg02896266          0.7398698          0.7268130          0.7146548
##            X5775278078_R05C01 X5775278078_R05C02 X5930514034_R01C02
## cg11954957          0.7947203          0.7965254          0.8259956
## cg16736630          0.4096823          0.6150373          0.3902688
## cg15903280          0.6279177          0.7000623          0.5624398
## cg09856436          0.5354099          0.5710098          0.5061116
## cg13856810          0.7760182          0.8167775          0.8534069
## cg02896266          0.6718059          0.6839416          0.7383158
##            X5930514035_R04C02 X5930514035_R06C02 MW_pvalues
## cg11954957          0.8016213          0.8005788 0.02857143
## cg16736630          0.3282085          0.4160006 0.02857143
## cg15903280          0.5160919          0.5841023 0.02857143
## cg09856436          0.5224426          0.5246753 0.02857143
## cg13856810          0.8239305          0.8044273 0.02857143
## cg02896266          0.7247830          0.6897759 0.02857143
```

SECTION 10: multiple-testing corrections is necessary when the experiment provides the analysis and evaluation of too many data, as in this case. The Bonferroni and Benjamini-Hochberg correction methods are applied in this pipeline

Step 1 Apply the **p.adjust()** function that returns adjusted p-values using a specified method of correction

```
not_corrected_pvalues <-pvalues_df_ordered[,9]
bonferroni_pvalues <- p.adjust(not_corrected_pvalues, "bonferroni")
summary(bonferroni_pvalues)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
```

```
BH_pvalues <- p.adjust(not_corrected_pvalues, "BH")
summary(BH_pvalues)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4720  0.7554  0.9007  0.8558  0.9855  1.0000
```

The resulting corrected p-values in both cases seems to be not significant: the bonferroni correction returns all the p-values equal to 1; the BH correction returns also high corrected p-values respect to the level of confidence

Step 2 Update the dataframe with the corrected p-values

```
bonf_BH_corrected_df <- data.frame(pvalues_df_ordered,BH_pvalues, bonferroni_pvalues)
head(bonf_BH_corrected_df)
```

```
##            X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01
## cg11954957          0.8237170          0.8052609          0.7743885
## cg16736630          0.2895560          0.2267553          0.5232336
## cg15903280          0.5497174          0.4187877          0.6195478
## cg09856436          0.5179173          0.5102878          0.5295284
## cg13856810          0.8574653          0.8461140          0.7951753
## cg02896266          0.7398698          0.7268130          0.7146548
##            X5775278078_R05C01 X5775278078_R05C02 X5930514034_R01C02
## cg11954957          0.7947203          0.7965254          0.8259956
## cg16736630          0.4096823          0.6150373          0.3902688
## cg15903280          0.6279177          0.7000623          0.5624398
## cg09856436          0.5354099          0.5710098          0.5061116
## cg13856810          0.7760182          0.8167775          0.8534069
## cg02896266          0.6718059          0.6839416          0.7383158
##            X5930514035_R04C02 X5930514035_R06C02 MW_pvalues BH_pvalues
## cg11954957          0.8016213          0.8005788 0.02857143  0.4720216
## cg16736630          0.3282085          0.4160006 0.02857143  0.4720216
## cg15903280          0.5160919          0.5841023 0.02857143  0.4720216
## cg09856436          0.5224426          0.5246753 0.02857143  0.4720216
## cg13856810          0.8239305          0.8044273 0.02857143  0.4720216
## cg02896266          0.7247830          0.6897759 0.02857143  0.4720216
##            bonferroni_pvalues
## cg11954957                  1
## cg16736630                  1
## cg15903280                  1
## cg09856436                  1
## cg13856810                  1
## cg02896266                  1
```

Step 3 Set a threshold equal to **0.05** to assess how many probes result to be significnatly differentially methylated before and after the correction

```
dim(bonf_BH_corrected_df[bonf_BH_corrected_df$MW_pvalues<=0.05,])
```

```
## [1] 29388    11
```

```
dim(bonf_BH_corrected_df[bonf_BH_corrected_df$BH_pvalues<=0.05,])
```

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

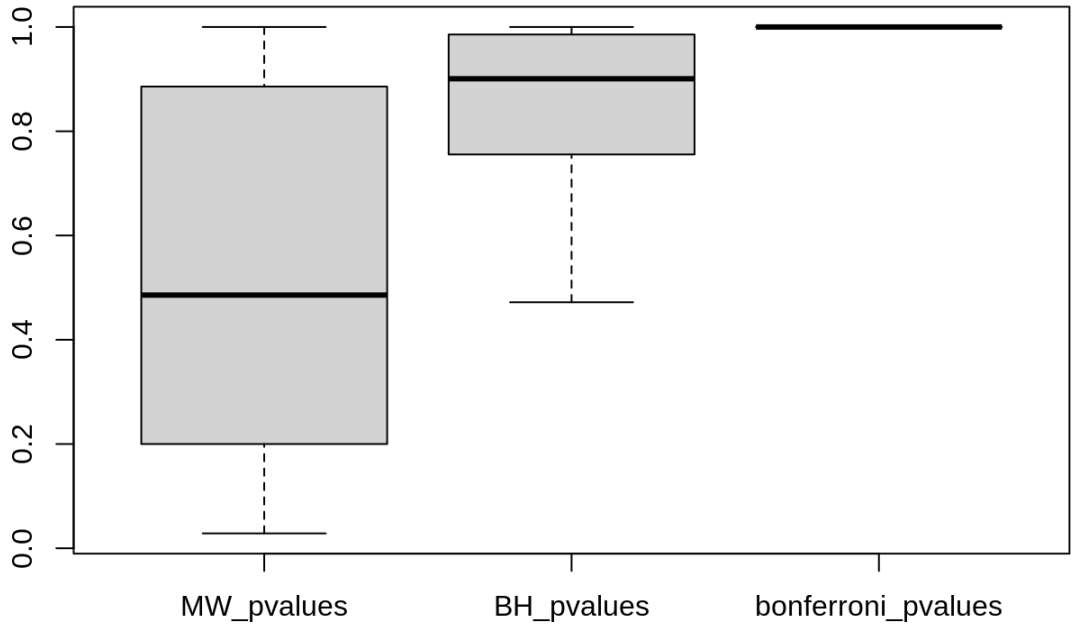
```
dim(bonf_BH_corrected_df[bonf_BH_corrected_df$bonferroni_pvalues<=0.05,])
```

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

The mutiple-testing correction methods led to no significant results. From the not corrected p-values, 29388 probes are below the threshold and can be considered significantly differentially methylated loci according to the M-W test.

Step 4 Produce the boxplot to visualize the distribution of the p-values and further consolidate what has been said before

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



SECTION 11 : Produce the Heatmap for the first 100 differentially methyated probes

Step 1 Use the gplots package and create an input object that contains the first 100 differentially methyated probes with a significant nominal p-value.

```
library(gplots)
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:IRanges':
##
##     space
```

```
## The following object is masked from 'package:S4Vectors':
##
##     space
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
input_heatmap <- as.matrix(bonf_BH_corrected_df[1:100,1:8])
head(input_heatmap)
```

```
##            X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01
## cg11954957          0.8237170          0.8052609          0.7743885
## cg16736630          0.2895560          0.2267553          0.5232336
## cg15903280          0.5497174          0.4187877          0.6195478
## cg09856436          0.5179173          0.5102878          0.5295284
## cg13856810          0.8574653          0.8461140          0.7951753
## cg02896266          0.7398698          0.7268130          0.7146548
##            X5775278078_R05C01 X5775278078_R05C02 X5930514034_R01C02
## cg11954957          0.7947203          0.7965254          0.8259956
## cg16736630          0.4096823          0.6150373          0.3902688
## cg15903280          0.6279177          0.7000623          0.5624398
## cg09856436          0.5354099          0.5710098          0.5061116
## cg13856810          0.7760182          0.8167775          0.8534069
## cg02896266          0.6718059          0.6839416          0.7383158
##            X5930514035_R04C02 X5930514035_R06C02
## cg11954957          0.8016213          0.8005788
## cg16736630          0.3282085          0.4160006
## cg15903280          0.5160919          0.5841023
## cg09856436          0.5224426          0.5246753
## cg13856810          0.8239305          0.8044273
## cg02896266          0.7247830          0.6897759
```

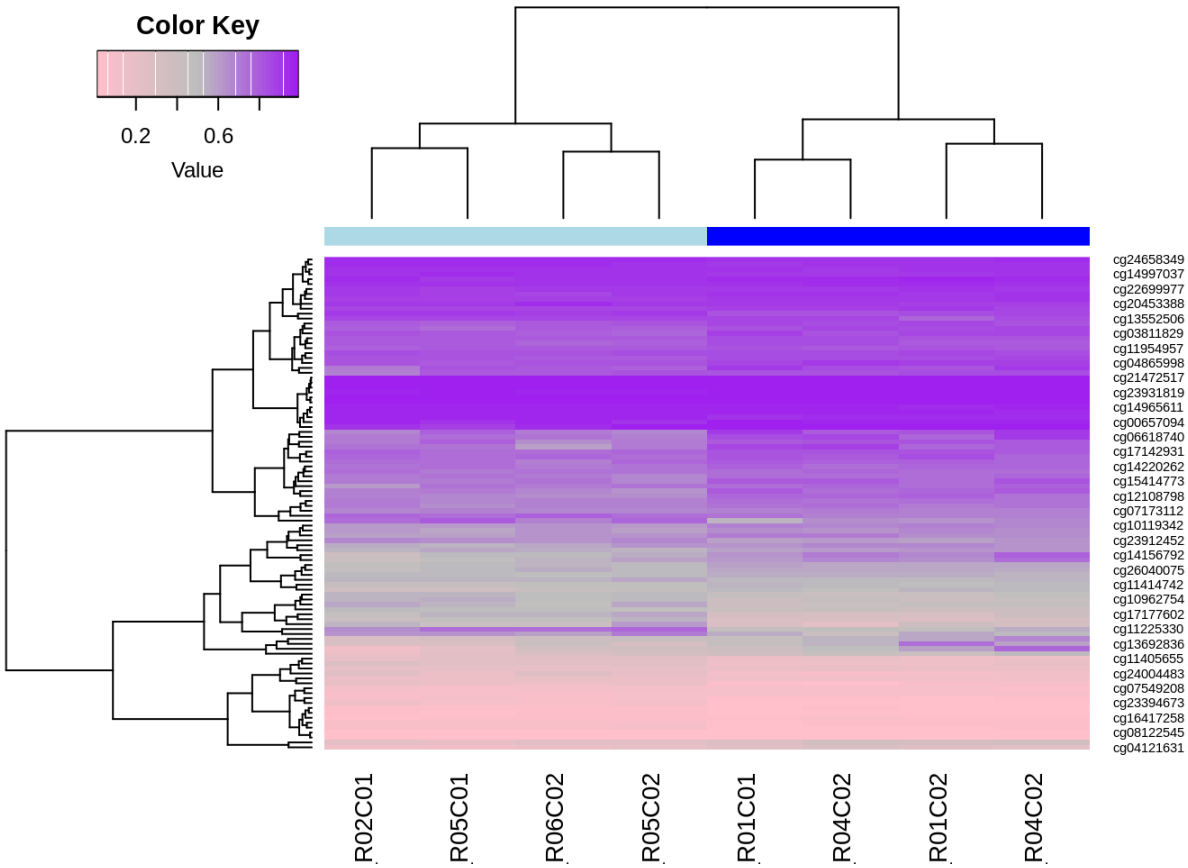
The matrix produced contains only the beta values of the first 100 probes tested to have a p-value lower than the settled threshold (0.05)

Step 2 Apply the heatmap.2() function to produce the heatmap of the input matrix values by setting the colors for the samples according to the phenogroup DS and WT. The linkage method and the distance metric used are the default ones: average linkage and euclidean distance

```
SAMPLE_SHEET$Group
```

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

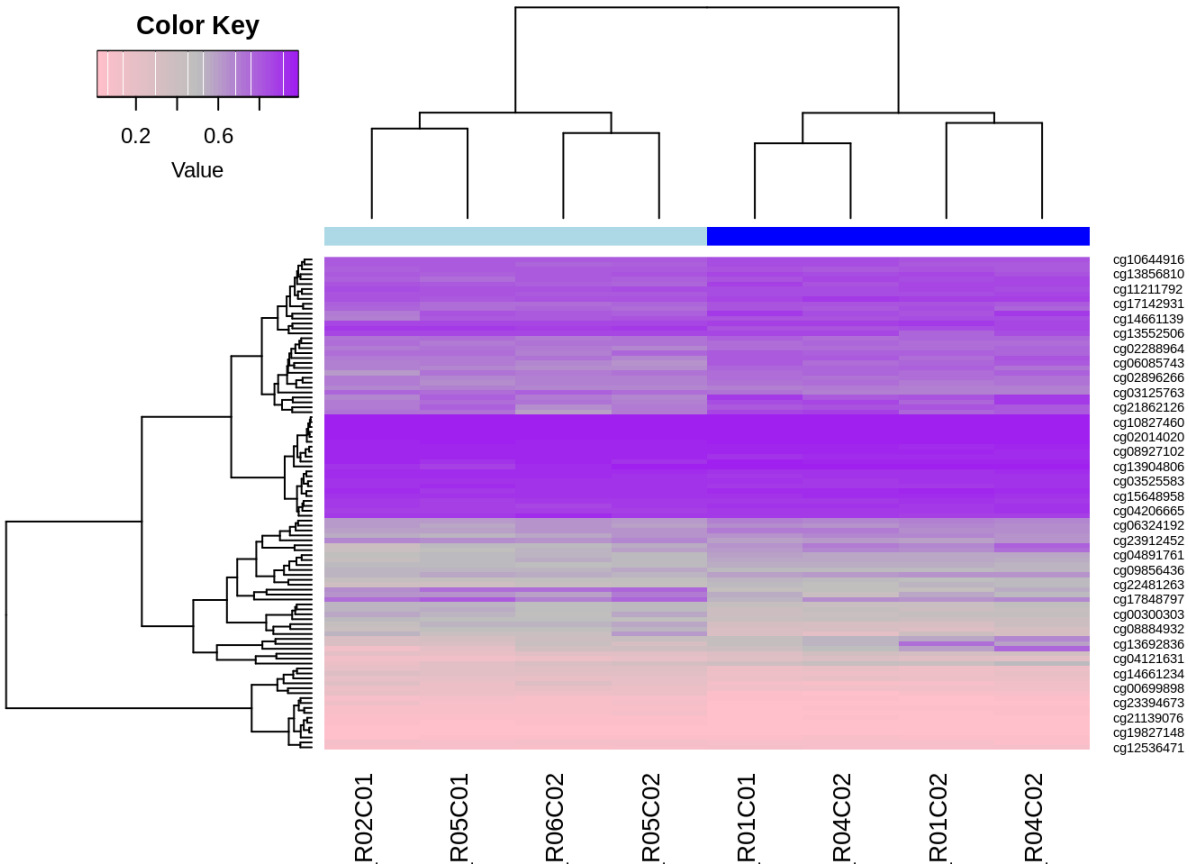
```
colorbar <- c("blue", "blue", "lightblue", "lightblue", "lightblue", "blue", "blue",
"lightblue")
color <- colorRampPalette(c( "pink","grey", "purple"))(50)
heatmap.2(input_heatmap, col=color , Rowv=T, Colv=T, dendrogram="both", key=T, ColSid
eColors=colorbar,density.info="none",trace="none",scale="none", symm=F)
```



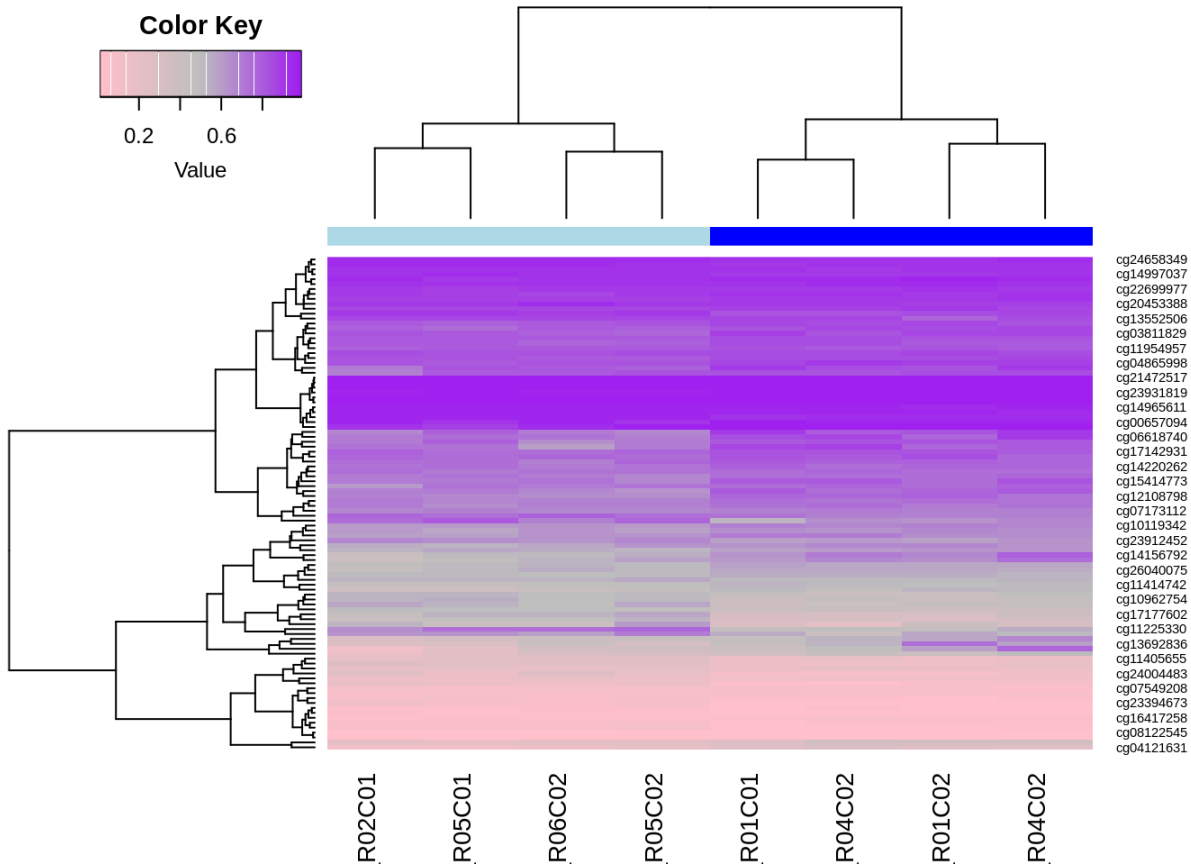
The computed heatmap shows that by the hierarchical clustering procedure WT and DS samples are well divided. Some probes (shown in the vertical side) have a higher level of methylation (iper-methylation) in the WT phenogroup samples than the other one and viceversa

The heatmap.2 function allows to easily manage and modify the default settings: the linkage method can be modified to produce the following heatmap with “single linkage” and then with “complete linkage”

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




```
heatmap.2(input_heatmap, col=color , Rowv=T, Colv=T, hclustfun=function(x) hclust(x,
method="complete"), dendrogram="both", key=T, ColSideColors=colorbar,density.info="no
ne",trace="none",scale="none", symm=F)
```



The three heatmaps produced using different linkage methods show slightly different outputs but all of them maintain the two phenogroups well clustered

SECTION 12: Produce a volcanoplot and a Manhattan plot from the results of the differential methylation analysis.

To obtain a volcanoplot it is necessary to compute the difference of the beta values averages for each probe in the two different pheno gorup (WT and DS)

Step 1 Create two matrices containing the beta values of the DS and WT groups

```
beta_allgroups <- bonf_BH_corrected_df[,1:8]
beta_groupDS <- beta_allgroups[,SAMPLE_SHEET$Group=="DS"]
beta_groupWT<- beta_allgroups[,SAMPLE_SHEET$Group=="WT"]
```

Step 2 Compute the mean of the two groups and the difference between the two means

```
mean_beta_groupDS <- apply(beta_groupDS,1,mean)
mean_beta_groupWT <- apply(beta_groupWT,1,mean)
delta <- mean_beta_groupDS- mean_beta_groupWT
summary(delta)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.5763406 -0.0061096 -0.0001211  0.0003977  0.0059492  0.6507866
```

Step 3 create a dataframe with the delta values and the -log10 of the nominal p-values from the M-W test for each probe

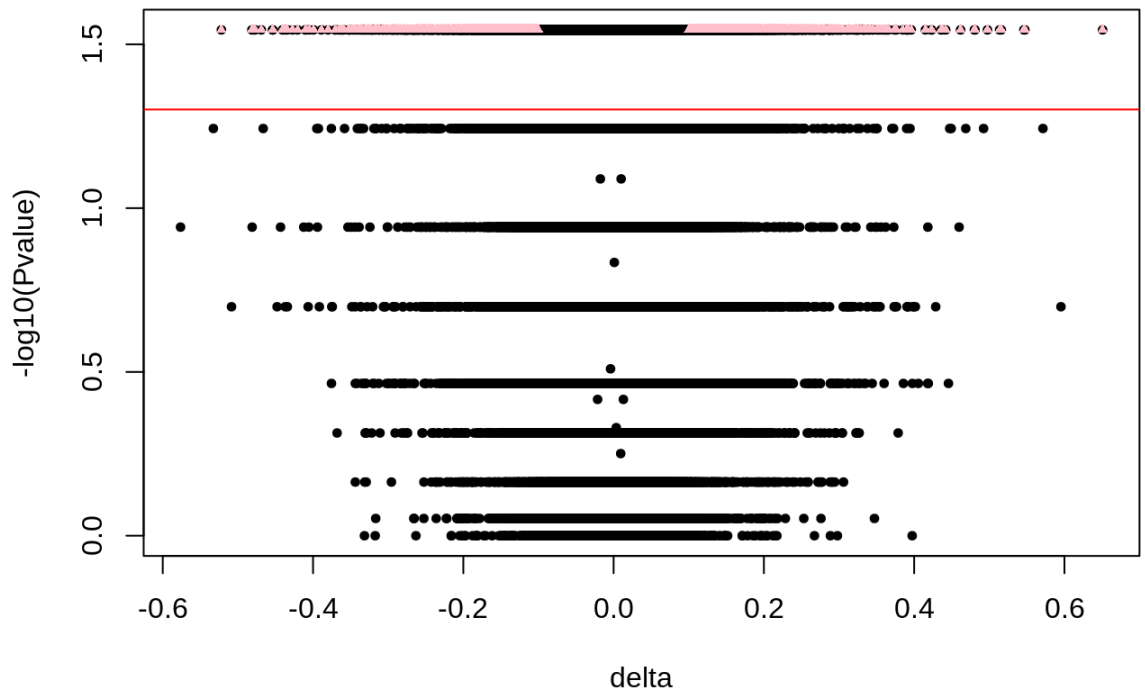
```
volcanop_dt <- data.frame(delta, -log10(bonf_BH_corrected_df$MW_pvalues))
head(volcanop_dt)
```

	delta <dbl>	X.log10.bonf_BH_corrected_df.MW_pvalues. <dbl>
cg11954957	0.02259544	1.544068
cg16736630	-0.18229126	1.544068
cg15903280	-0.12114830	1.544068
cg09856436	-0.02596603	1.544068
cg13856810	0.04712958	1.544068
cg02896266	0.04240087	1.544068

6 rows

Step 4 plot the values from the dataframe to produce the volcanoplot

```
{HighLight <- volcanop_dt[abs(volcanop_dt[,1])>0.1 & volcanop_dt[,2]>(-log10(0.05)),]
plot(volcanop_dt[,1],volcanop_dt[,2], pch=19,  cex=0.6 ,xlab="delta",ylab="-log10(Pva
lue)", col="black")
abline(a=-log10(0.05),b=0, col="red")
points(HighLight[,1], HighLight[,2], pch=17, cex=0.6, col="pink")}
```



only few probes are above the significance pvalue threshold of 0.05 (the red line) according to the nominal p-value. In addition those probes that are above the threshold and also have an absolute delta value greater than 0.1 are highlighted in pink. **NOTE:** the unusual volcanoplot produced in this case may be given by the fact that a non-parametric test has been applied for the detection of the pvalues

Step 4 In order to build the Manhattan plot the “gap” package is required. Then the **merge()** function can be used to enrich the dataframe containing the normilised beta values and the p-values (from the M-W test and the corrected ones) with other genome information retrieved by mapping them to the Illumina 450k Manifest according to the probe names **NOTE:** probe names correspond to rownames in the dataframe and to the “IlmnID” columns of the Manifest)

```
library(gap)
```

```
## gap version 1.2.2
```

```
load("Illumina450Manifest_clean.RData")

bonf_BH_corrected_df1 <- data.frame(rownames(bonf_BH_corrected_df),bonf_BH_corrected_
df)
colnames(bonf_BH_corrected_df1)[1] <- "IlmnID"


annotated_df <- merge(bonf_BH_corrected_df1, Illumina450Manifest_clean, by="IlmnID")
dim(annotated_df)
```

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

Step 5 Create a suitable input for the production of the manhattan plot: build a dataframe from the annotated_df which contains three main information (Chromosome number, position info and the p-values from the M-W test)

```
input_mht <- data.frame(annotated_df$CHR, annotated_df$MAPINFO, annotated_df$MW_pvalu
es)

head(input_mht)
```

```
##   annotated_df.CHR annotated_df.MAPINFO annotated_df.MW_pvalues
## 1               16             53468112              0.20000000
## 2                3             37459206              0.34285714
## 3                3            171916037              0.68571429
## 4                1             91194674              0.48571429
## 5                8             42263294              0.02857143
## 6               14             69341139              0.05714286
```

Order the levels according to the chromosome number

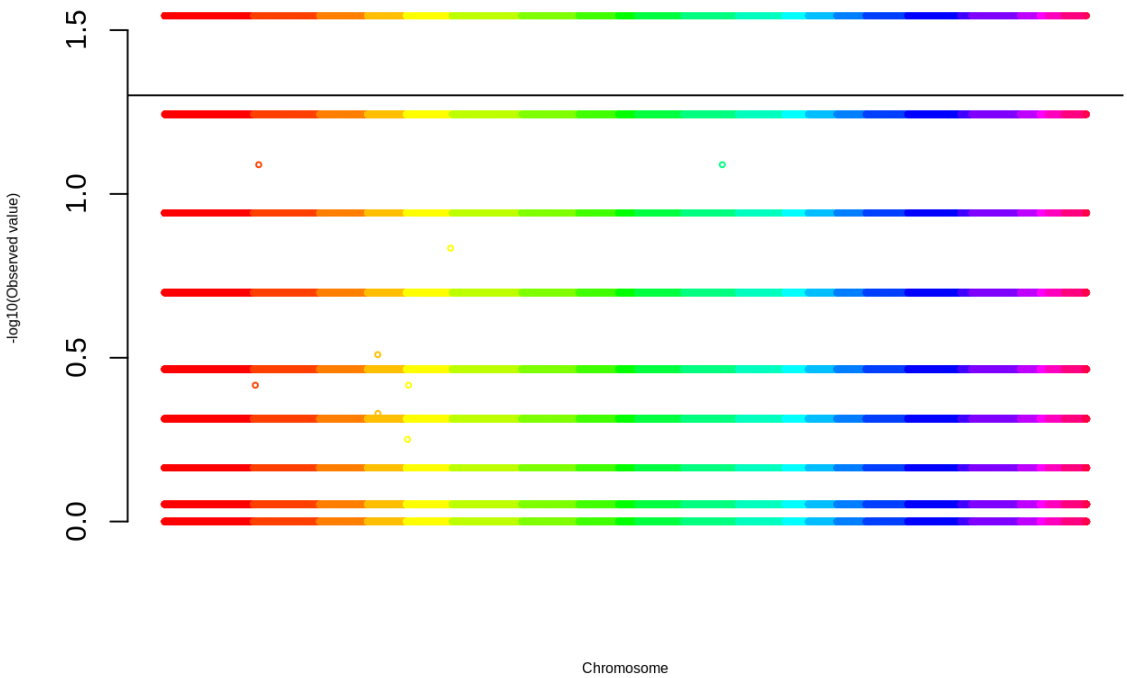
```
input_mht$annotated_df.CHR <-factor(input_mht$annotated_df.CHR, levels=c("1","2","
3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20","2
1","22","X","Y"))
```

Step 6 Use the function mhtplot() to build the Manhattan plot setting a suitable palette of colours for the rerpesentation of the data

```
palette <- rainbow(24)
palette
```

```
##  [1] "#FF0000" "#FF4000" "#FF8000" "#FFBF00" "#FFFF00" "#BFFF00" "#80FF00"
##  [8] "#40FF00" "#00FF00" "#00FF40" "#00FF80" "#00FFBF" "#00FFFF" "#00BFFF"
## [15] "#0080FF" "#0040FF" "#0000FF" "#4000FF" "#8000FF" "#BF00FF" "#FF00FF"
## [22] "#FF00BF" "#FF0080" "#FF0040"
```

```
{mhtplot(input_mht, control=mht.control(colors=palette))
axis(2)
abline(a=-log10(0.05),b=0)}
```



```
## Plotting points  1 - 46857
## Plotting points  46858 - 81667
## Plotting points  81668 - 106826
## Plotting points  106827 - 127290
## Plotting points  127291 - 151617
## Plotting points  151618 - 188228
## Plotting points  188229 - 218245
## Plotting points  218246 - 239195
## Plotting points  239196 - 249056
## Plotting points  249057 - 273444
## Plotting points  273445 - 302238
## Plotting points  302239 - 326777
## Plotting points  326778 - 339062
## Plotting points  339063 - 354140
## Plotting points  354141 - 369399
## Plotting points  369400 - 391368
## Plotting points  391369 - 419247
## Plotting points  419248 - 425169
## Plotting points  425170 - 450690
## Plotting points  450691 - 461069
## Plotting points  461070 - 465312
## Plotting points  465313 - 473864
## Plotting points  473865 - 485096
## Plotting points  485097 - 485512
```

The Manhattan plot is useful for the visualization of large dataset, as in this case. As seen in the volcanoplot, only few probes have p-values above the significance pvalue threshold (black line).

OPTIONAL SECTION

Biological background: the DS phenotype is caused by the trisomy 21 so it can be very useful to plot the denisty of the methylation values of the probes mapping on CHR21 in order to detect a possible correlation between the level of methylation of the CHR21 loci and the onset of the syndrome

Step 1 Use the dataframe created in the previous section to select only the probes that map onto the chromosome 21

```
head(annotated_df)
```

	IlmnID <chr>	X5775278051_R01C01 <dbl>	X5775278051_R04C02 <dbl>	X5775278078_R02C01 <dbl>	X5775278078
1	cg00000029	0.5919759	0.5442704	0.5263055	0
2	cg00000108	0.9512149	0.9503131	0.9431860	0
3	cg00000109	0.9044581	0.9126984	0.9007444	0
4	cg00000165	0.1205171	0.1034396	0.1089885	0
5	cg00000236	0.7579678	0.7601622	0.8312432	0
6	cg00000289	0.6781146	0.6977880	0.7253554	0

6 rows | 1-6 of 45 columns

```
CHR21_annotated <- annotated_df[annotated_df$CHR=="21",]
```

Step 2 map the probes names of the WT and DS dataframes containing the beta values and the probe names as rownames

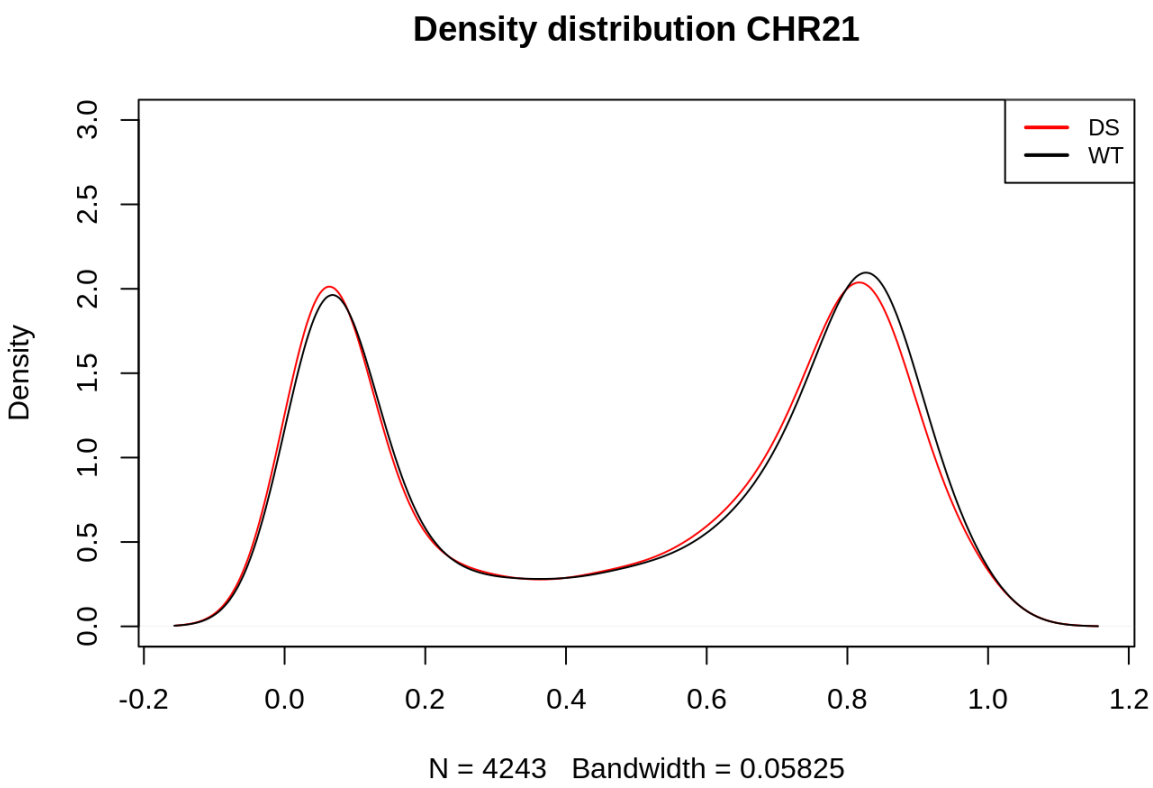
```
DS_betavalue_chr21 <- DS_betavalue[rownames(DS_betavalue) %in% CHR21_annotated$IlmnI
D,]
WT_betavalue_chr21 <- WT_betavalue[rownames(WT_betavalue) %in% CHR21_annotated$IlmnI
D,]
```

```
dim(WT_betavalue_chr21)
```

```
## [1] 4243    4
```

Step 3 compute the mean and the density of the beta values for the two groups and create the plot of the mean density distribution

```
mean_ds_21 <- apply(DS_betavalue_chr21,1,mean,na.rm=T)
mean_wt_21 <- apply(WT_betavalue_chr21,1,mean,na.rm=T)
den_ds_21 <- density(mean_ds_21)
den_wt_21 <- density(mean_wt_21 )

{plot(den_ds_21, col="red", main="Density distribution CHR21", ylim=c(0,3))
lines(den_wt_21, col="black")
legend("topright", legend=c("DS","WT"), col=c("red","black"), lwd=2, cex=0.8)}
```



The density distribution plot shows that there are no differences between the methylation and unmethylation

mode of the two phenogroups

Step 4 count how many probes can be defined significantly differentially methylated at the given threshold of 0.05

```
differential_meth <- CHR21_annotated[CHR21_annotated$MW_pvalues <=0.05,]
dim(differential_meth)
```

```
## [1] 485  44
```

The result shows that 485 probes are differentially methylated on chromosome 21 at signficance level threshold of 0.05 respect to the p-values obtained from the M-W test