A pipeline in R for the analysis of Infinium methylation array data

Saul Pierotti

June 14, 2020

This report was initially written as a Jupyter notebook, then migrated to Rmarkdown for compatibility issues with some packages. This report serves as a final assignment for the DNA and RNA dynamics course of the Bioinformatics master's degree at the University of Bologna (AA 2019/2020). The student-specific parameters to be used for the report (as detailed in the assignment) are reported below.

Parameter	Value
Student number	24
Step 3 address	59625465
Step 5 p-value threshold	0.01
Step 7 normalization	preprocessFunnorm
Step 9 test	Mann-Whitney test

Step 1

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

I first clean the workspace and then load the required libraries. I do not set the working directory since I will give full paths relative to the directory in which this notebook is run.

```
rm(list = ls())
```

```
require(minfi)
require(IlluminaHumanMethylation450kmanifest)
require(gplots)
require(gap)
require(amap)
require(magick)
data_dir <- './Input_data'
```

I load the sample sheet of the experiment using the `minfi` `read.metharray.sheet` function.

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

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

Now I load the raw experimental data relative to the sample sheet and I create the `RGChannelSet` object using the `read.metharray.exp` function.

```
RGSet <- read.metharray.exp(base = './Input_data', targets = targets)
dim(RGSet)
```

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

Step 2

Create the data frames Red and Green to store the red and green fluorescences respectively.

I use the `getGreen` and `getRed` functions on the `RGSet` for extracting data from the respective color channels. These data are then converted to 2 data frames of the same dimensionality (8*622399, 8 samples and 622399 probe addresses).

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

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

```
dim(Green)
```

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

Step 3

Fill the following table: what are the Red and Green fluorescences for the address assigned to you?

Optional: check in 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.

The student-specific probe address assigned for this step is `59625465`. First I note down which array and slide refer to which sample by reading the sample sheet.

```
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
```

Then I extract the red and green intensities for the assigned probe by slicing the respective data frames (`Red` and `Green`).

```
Red[rownames(Red) == "59625465", ]
```

```
##          X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01
## 59625465              11585              14282              11253
##          X5775278078_R05C01 X5775278078_R05C02 X5930514034_R01C02
## 59625465              11494              11254              11152
##          X5930514035_R04C02 X5930514035_R06C02
## 59625465              11625              13019
```

```
Green[rownames(Green) == "59625465", ]
```

```
##          X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01
## 59625465                935                845                666
##          X5775278078_R05C01 X5775278078_R05C02 X5930514034_R01C02
## 59625465                785                652                306
##          X5930514035_R04C02 X5930514035_R06C02
```

```
## 59625465                584                668
```

Now I load the manifest for the Illumina 450k Infinium array with the `getProbeInfo` function and check what is the chemistry of the assigned probe.

```
probe_info_I <- getProbeInfo(RGSet, type = 'I')
probe_info_II <- getProbeInfo(RGSet, type = 'II')
probe_info_I[probe_info_I$AddressA == 59625465, ]
```

```
## DataFrame with 0 rows and 8 columns
```

```
probe_info_I[probe_info_I$AddressB == 59625465, ]
```

```
## DataFrame with 0 rows and 8 columns
```

```
probe_info_II[probe_info_II$AddressA == 59625465, ]
```

```
## DataFrame with 1 row and 4 columns
##          Name    AddressA               ProbeSeqA      nCpG
##   <character> <character>          <DNAStringSet> <integer>
## 1  cg01644972    59625465 ATATCCTAAT...TACCTCAACC         1
```

I find that it is a type II probe, so there is no color to specify in the table. With all of these pieces of information, I can complete the table as assigned. I also included additional information for clarity and completeness.

Sample	Group	Slide	Array	Red fluorescence	Green fluorescence	Type	Color
1020	DS	5775278051	R01C01	11585	935	II	/
1036	DS	5775278051	R04C02	14282	845	II	/
3038	WT	5775278078	R02C01	11253	666	II	/
3042	WT	5775278078	R05C01	11494	785	II	/
3052	WT	5775278078	R05C02	11254	652	II	/
1016	DS	5930514034	R01C02	11152	306	II	/
1029	DS	5930514035	R04C02	11625	584	II	/
3029	WT	5930514035	R06C02	13019	668	II	/

Step 4

Create the object MSet.raw

I use the `preprocessRaw` function on the `RGSet` to get `MSet.raw`. This object contains the beta and M values for the dataset.

```
MSet.raw <- preprocessRaw(RGSet)
dim(MSet.raw)
```

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

Step 5

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

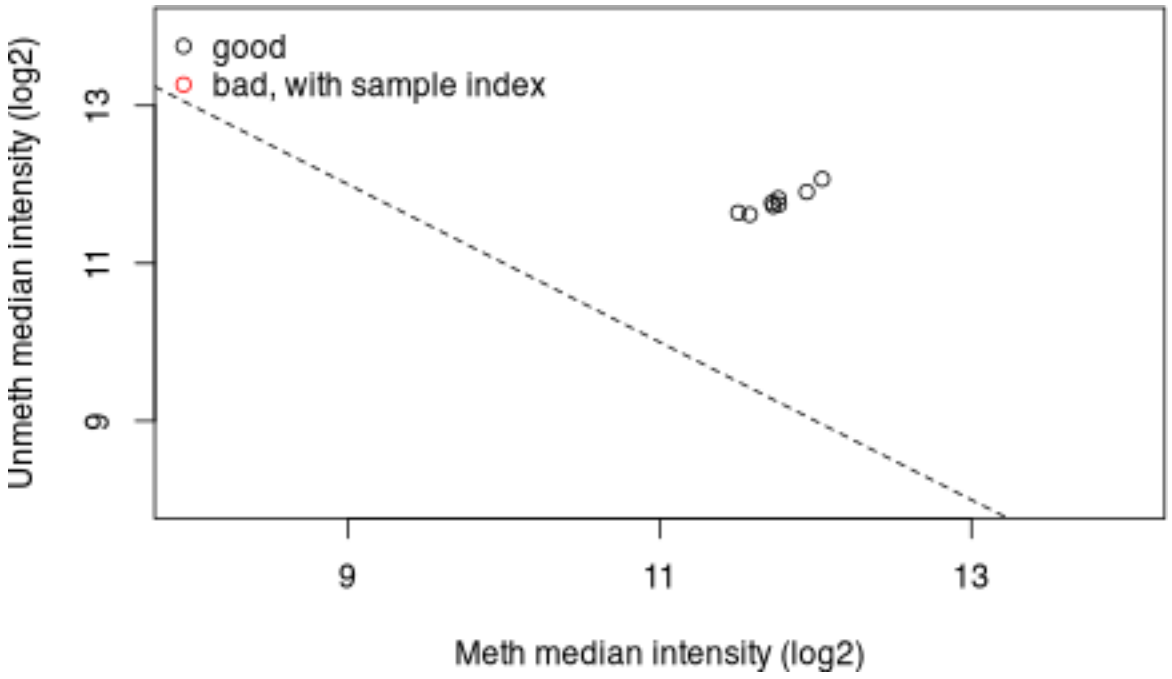
QCplot

check the intensity of negative controls using minfi

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

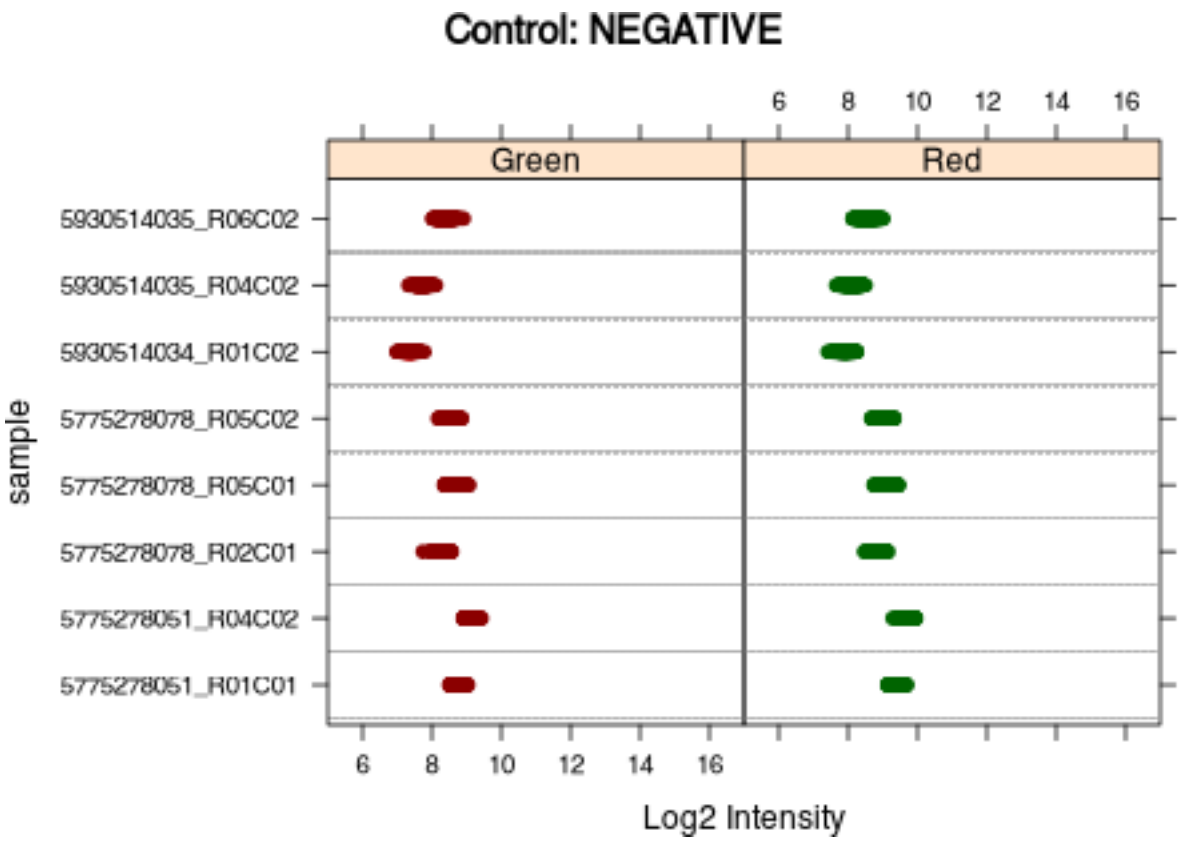
I first create a QC plot by getting the medians of the methylated and unmethylated channels for each sample with the `getQC` function and plotting them with the `plotQC` function.

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



Since all the samples have high median methylated and unmethylated signals, I can consider that all of them are of good quality. Now I plot the negative control probe intensities of the `RGSet` using the `controlStripPlot` function.

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



The represented colors are inverted (the red is shown in green and vice-versa). This was observed also during the lectures and it can be a bug of the package (`minfi`). During the lectures, we determined that the textual label is correct, while the color of the dots is inverted. The negative controls are all fine since their intensity is at background levels (below 10 in Log2 scale). We can notice how the negative control probes intensities show slight differences across slides in a manner that is fairly consistent for the 2 color channels.

Now I will determine the detection p-values for the probes using the `detectionP` function on the `RGSet`. I will consider as failed the probes with a detection p-value higher than the assigned student-specific threshold of 0.01.

```
detP <- detectionP(RGSet)
dim(detP)
```

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

```
failed <- detP > 0.01
table(failed)
```

```
## failed
##   FALSE    TRUE
## 3881919    2177
```

```
summary(failed)
```

```
##  5775278051_R01C01 5775278051_R04C02 5775278078_R02C01 5775278078_R05C01
##  Mode :logical     Mode :logical     Mode :logical     Mode :logical
##  FALSE:485189      FALSE:485252      FALSE:485200      FALSE:485027
##  TRUE :323         TRUE :260         TRUE :312         TRUE :485
##  5775278078_R05C02 5930514034_R01C02 5930514035_R04C02 5930514035_R06C02
##  Mode :logical     Mode :logical     Mode :logical     Mode :logical
```

```
##  FALSE:485047      FALSE:485389      FALSE:485452      FALSE:485363
##  TRUE :465         TRUE :123         TRUE :60          TRUE :149
```

From the last summary I note down the number of failed probes for each sample. This is the number of `TRUE` values in the `failed` data frame for each sample.

Sample	Group	Slide	Array	Failed probes (p-value > 0.01)
1020	DS	5775278051	R01C01	323
1036	DS	5775278051	R04C02	260
3038	WT	5775278078	R02C01	312
3042	WT	5775278078	R05C01	485
3052	WT	5775278078	R05C02	465
1016	DS	5930514034	R01C02	123
1029	DS	5930514035	R04C02	60
3029	WT	5930514035	R06C02	149

Even if not specified in the assignment, it can be useful to look for the percentages of failed probes in each sample.

```
sample_failed_probes_percentage <- colMeans(failed)
sample_failed_probes_percentage
```

```
## 5775278051_R01C01 5775278051_R04C02 5775278078_R02C01 5775278078_R05C01
##      0.0006652771      0.0005355171      0.0006426206      0.0009989454
## 5775278078_R05C02 5930514034_R01C02 5930514035_R04C02 5930514035_R06C02
##      0.0009577518      0.0002533408      0.0001235809      0.0003068925
```

All the samples seem to show an acceptable amount of failed probes. I can also check for probes that failed (according to the aforementioned criterion) on more than 1% of the samples (also this was not required in the assignment).

```
mean_failure_probes <- rowMeans(failed)
probes_to_be_retained <- mean_failure_probes < 0.01
summary(probes_to_be_retained)
```

```
##    Mode   FALSE    TRUE
## logical    1307  484205
```

```
probes_to_be_removed <- !probes_to_be_retained
names_probes_to_be_removed <- names(probes_to_be_removed)
```

1307 probes failed in more than 1% of the samples. I will save their name in a vector to eventually filter them out later, but I will retain them, for now, to avoid possible problems with processing due to missing probes.

Step 6

> Calculate raw beta and M values and plot the densities of mean methylation values, dividing the samples in `DS` and `WT` (suggestion: subset the beta and M values matrixes to retain `DS` or `WT` subjects and apply the function mean to the 2 subsets).

I first extract the beta and M values from `MSet.raw` using the `getBeta` and `getM` functions.

```
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
```

```
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
```

Now I subset the beta and M value matrices based on the group (`DS` or `WT`) of the sample.

```
isDS <- targets$Group == 'DS'
summary(isDS)
```

```
##    Mode   FALSE    TRUE
## logical       4       4
```

```
beta_DS <- beta[, isDS]
beta_WT <- beta[, !isDS]
M_DS <- M[, isDS]
M_WT <- M[, !isDS]
```

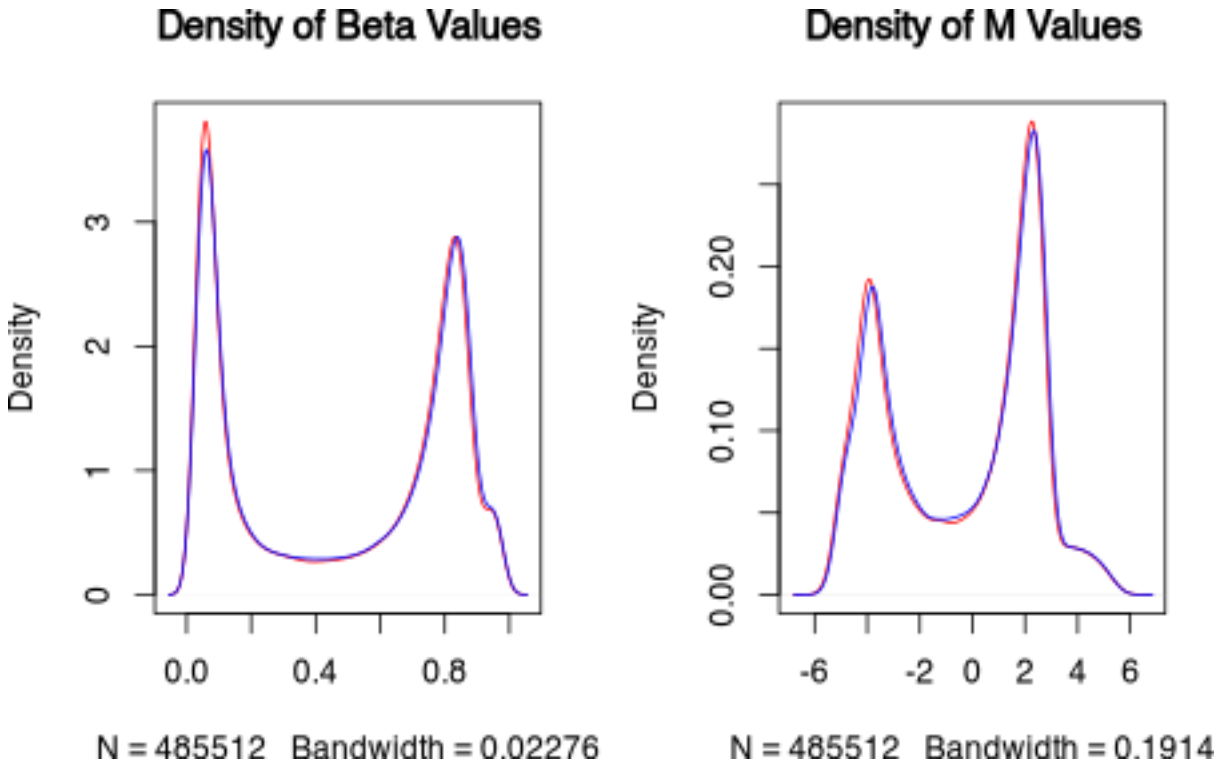
Now I calculate the mean beta and M values for each queried position across the 2 sample groups (across rows, `MARGIN=1`), discarding NA values (`na.rm=T`). I need to discard NA values since, where both the methylated and unmethylated signals are 0, the beta and M values are not defined.

```
beta_DS_mean <- apply(beta_DS, MARGIN = 1, mean, na.rm = T)
beta_WT_mean <- apply(beta_WT, MARGIN = 1, mean, na.rm = T)
M_DS_mean <- apply(M_DS, MARGIN = 1, mean, na.rm = T)
M_WT_mean <- apply(M_WT, MARGIN = 1, mean, na.rm = T)
```

Now I can calculate the density distribution of all of these means and plot them.

```
beta_DS_mean_d <- density(beta_DS_mean)
beta_WT_mean_d <- density(beta_WT_mean)
M_DS_mean_d <- density(M_DS_mean)
M_WT_mean_d <- density(M_WT_mean)
```

```
par(mfrow = c(1, 2))
plot(beta_DS_mean_d, main = "Density of Beta Values", col = "red")
lines(beta_WT_mean_d, main = "Density of Beta Values", col = "blue")
plot(M_DS_mean_d, main = "Density of M Values", col = "red")
lines(M_WT_mean_d, main = "Density of M Values", col = "blue")
```



The red lines represent the `DS` samples beta and M values mean densities while the blue lines represent those of the `WT` samples. We can see how in the 2 sample groups both beta and M values tend to have a very similar distribution. It is noticeable the presence of a small sub-peak to the right of the main methylated peak both in the beta and M value graphs. This is probably due to the misalignment of type I and type II peaks in the raw data.

Step 7

> Normalize the data using the function assigned to each student 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.
>
> Provide a short comment regarding the changes you observe.

I first extract the type I and type II probe names from the manifest of the array and use them to subset the beta values matrix.

```
type_I <- getProbeInfo(MSet.raw, type = 'I')$Name
type_II <- getProbeInfo(MSet.raw, type = 'II')$Name
beta_I <- beta[rownames(beta) %in% type_I, ]
beta_II <- beta[rownames(beta) %in% type_II, ]
dim(beta_I)
```

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

```
dim(beta_II)
```

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

I then calculate mean and standard deviation densities for type I and type II probes.

```
mean_beta_I <- apply(beta_I, 1, mean, na.rm = T)
mean_beta_II <- apply(beta_II, 1, mean, na.rm = T)
d_mean_beta_I <- density(mean_beta_I)
d_mean_beta_II <- density(mean_beta_II)
sd_beta_I <- apply(beta_I, 1, sd, na.rm = T)
sd_beta_II <- apply(beta_II, 1, sd, na.rm = T)
d_sd_beta_I <- density(sd_beta_I)
d_sd_beta_II <- density(sd_beta_II)
```

I normalize (between-array) the RGSet using the function assigned (`preprocessFunnorm`), which removes the variability explained by the control probes. This will produce a `GenomicRatioSet` object.

```
normalized_data <- preprocessFunnorm(RGSet)
normalized_data
```

```
## 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
```

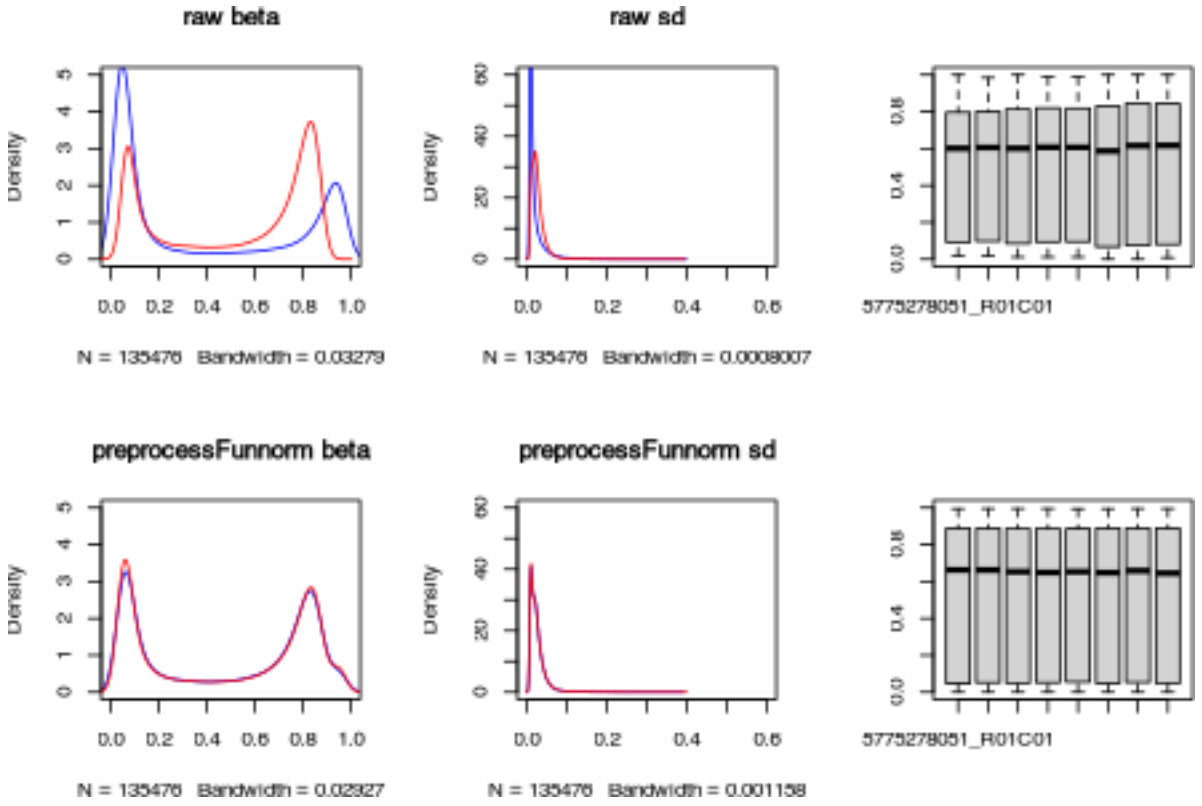
I can now extract the normalized beta values using the `getBeta` function and then separate type I and type II probes as done with the raw data. I also calculate mean and standard deviation densities as before.

```
beta_norm <- getBeta(normalized_data)
beta_I_norm <- beta[rownames(beta_norm) %in% type_I, ]
beta_II_norm <- beta[rownames(beta_norm) %in% type_II, ]
mean_beta_I_norm <- apply(beta_I_norm, 1, mean, na.rm = T)
mean_beta_II_norm <- apply(beta_II_norm, 1, mean, na.rm = T)
d_mean_beta_I_norm <- density(mean_beta_I_norm)
d_mean_beta_II_norm <- density(mean_beta_II_norm)
sd_beta_I_norm <- apply(beta_I_norm, 1, sd, na.rm = T)
```

```
sd_beta_II_norm <- apply(beta_II_norm, 1, sd, na.rm = T)
d_sd_beta_I_norm <- density(sd_beta_I_norm)
d_sd_beta_II_norm <- density(sd_beta_II_norm)
```

Finally, I can produce plots of mean beta value densities and standard deviation densities, differentiated by the chemistry of the probes, for raw and normalized data. I also produce a box plot of the beta values across samples for the raw and normalized data. I set the axes limits to make the plots easier to compare.

```
par(mfrow = c(2, 3))
plot(
  d_mean_beta_I,
  col = "blue",
  main = "raw beta",
  xlim = c(0, 1),
  ylim = c(0, 5)
)
lines(d_mean_beta_II, col = "red")
plot(
  d_sd_beta_I,
  col = "blue",
  main = "raw sd",
  xlim = c(0, 0.6),
  ylim = c(0, 60)
)
lines(d_sd_beta_II, col = "red")
boxplot(beta, ylim = c(0, 1))
plot(
  d_mean_beta_I_norm,
  col = "blue",
  main = "preprocessFunnorm beta",
  xlim = c(0, 1),
  ylim = c(0, 5)
)
lines(d_mean_beta_II_norm, col = "red")
plot(
  d_sd_beta_I_norm,
  col = "blue",
  main = "preprocessFunnorm sd",
  xlim = c(0, 0.6),
  ylim = c(0, 60)
)
lines(d_sd_beta_II_norm, col = "red")
boxplot(beta_norm, ylim = c(0, 1))
```



The blue lines refers to Infinium I probes, while the red lines refer to Infinium II probes. I can note in the boxplots how the distribution of beta values is almost identical across samples for the normalized data, while it is more variable for the raw data. Only the median in the normalized data seems to differ slightly among samples. The distribution of mean densities and standard deviation densities is almost equivalent among type I and II probes in the normalized data, while it is heavily different in the raw data. In the raw data, the density of standard deviations for type II probes tend to be shifted towards higher values compared to type I probes. The peaks on the mean density distribution for the type II probes are more shifted towards the center compared to type I probes in the raw data. All of these differences among chemistry types in the raw data are as expected: type II probes are more variable and they show a narrower range of beta values compared to type I probes.

Step 8

> Perform a PCA on the beta matrix generated in step 7.
>
> Comment the plot.

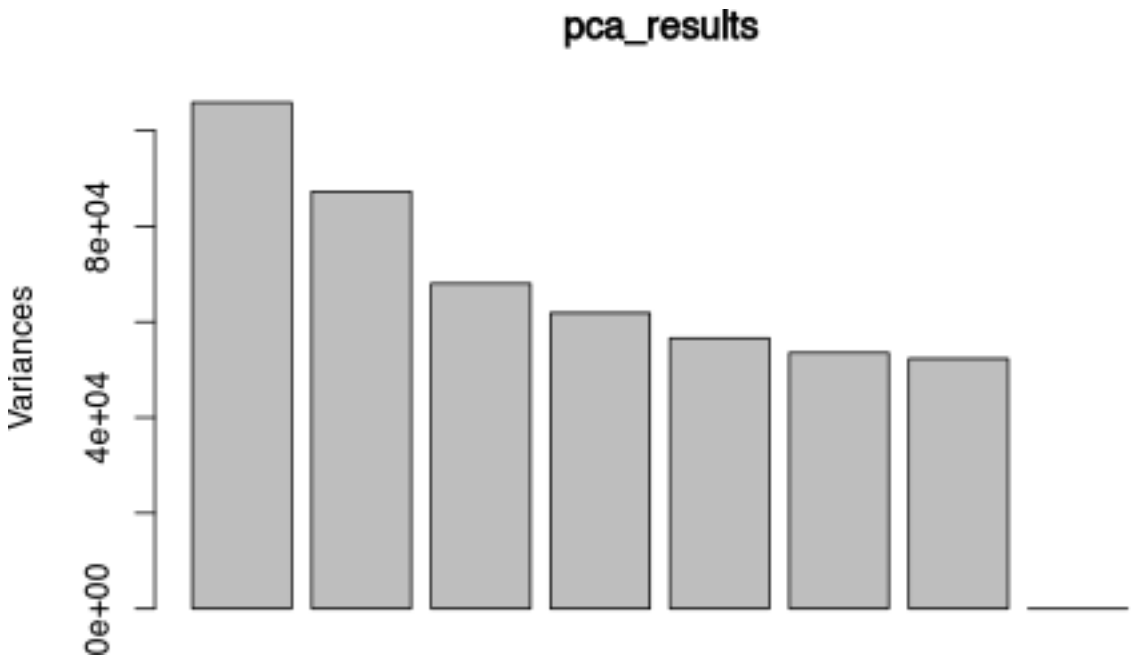
I want now to apply the `prcomp` function on the normalized beta values matrix obtained in step 7 to perform a PCA on it. The beta values matrix has one column per sample and one probe per row. Because of this, it must be transposed before applying the PCA (`t` function).

```
pca_results <- prcomp(t(beta_norm), scale = T)
summary(pca_results)
```

```
## 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
```

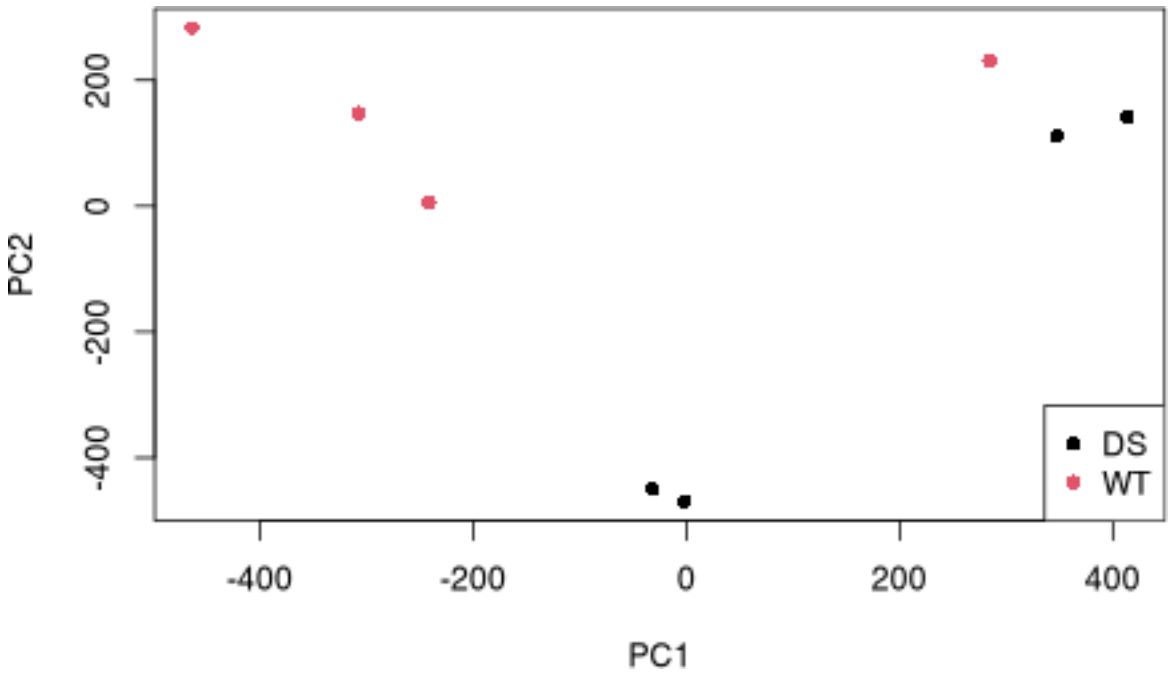
```
## Standard deviation     228.6972 6.435e-12
## Proportion of Variance   0.1077 0.000e+00
## Cumulative Proportion    1.0000 1.000e+00
```

```
plot(pca_results)
```



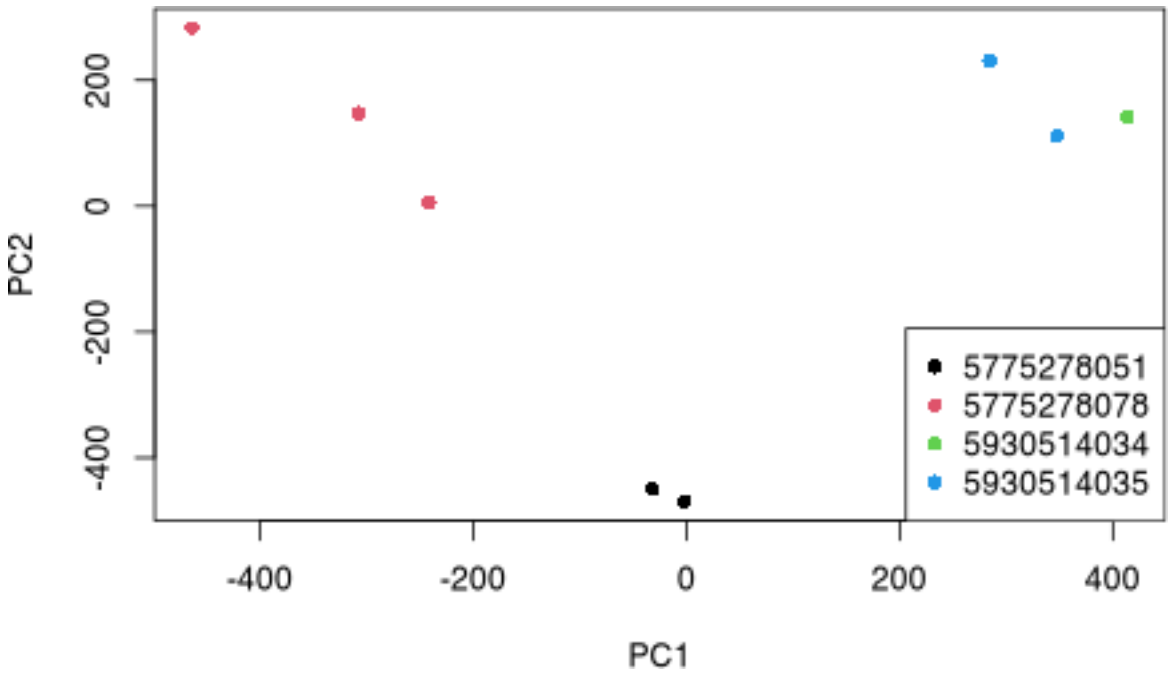
In the scree plot, it seems that the variance is quite uniformly distributed among the first 7 components. The 8th component contains really limited variance, and so the first 7 components can explain almost all of the variability of the dataset. The quite uniform distribution of variance among the first 7 components may mean that the samples differ sensibly on more than a couple of orthogonal axes. I can plot the first 2 to components to see if the `DS` and `WT` groups are separated according to those dimensions.

```
palette("default")
groups <- factor(targets$Group)
plot(
  pca_results$x[, 1],
  pca_results$x[, 2],
  pch = 16,
  col = groups,
  xlab = "PC1",
  ylab = "PC2"
)
legend(
  "bottomright",
  legend = levels(groups),
  col = c(1:nlevels(groups)),
  pch = 16
)
```



I can color the same plot according to the array used, to spot batch effects.

```
slides <- factor(targets$Slide)
plot(
  pca_results$x[, 1],
  pca_results$x[, 2],
  pch = 16,
  col = slides,
  xlab = "PC1",
  ylab = "PC2"
)
legend(
  "bottomright",
  legend = levels(slides),
  col = c(1:nlevels(slides)),
  pch = 16
)
```



From these plots, I can see how the 2 groups (`DS` and `WT`) seem distinct according to the first 2 principal components. A line of the type $y = x$ seems to well separate the 2 groups. `DS` samples seem confined to a region where PC1 is greater than PC2, while `WT` samples are confined in a region where PC2 is greater than PC1. The samples seem to be highly clustered according to the slide in which they were run (presumably due to batch effects). However, slide `5930514035`, which contains a sample from the `DS` group and a sample from the `WT` group, still respects the pattern of separation between sample groups. Randomization of the samples on the slides would have been advisable to allow to test for batch effects separately from sample groups. There seem to be no outliers among the samples.

Step 9

> Using the matrix of normalized beta values generated in step 7, identify differentially methylated probes between group `DS` and group `WT` using the functions assigned to each student.
>
> Note; it can take several minutes; if you encounter any problem you can run the differential methylated analysis only on a subset of probes (for example those on chromosome 1, 18 and 21)

The test assigned to me for the analysis of differentially methylated probes is the Mann-Whitney U test (Wilcoxon rank-sum test), which is implemented in the `wilcox.test` function. I apply it to the normalized beta values from step 7, using `DS` and `WT` as sample groups. I need to define a function to apply the test to each probe.

```
mann_whitney_all_rows <- function(x) {
  wilcox <- wilcox.test(x ~ groups)
  return(wilcox$p.value)
}

pval_raw <- apply(beta_norm, 1, mann_whitney_all_rows)
summary(pval_raw)
```

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

```
length(pval_raw)
```

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

I create then a data frame containing the beta values and the raw p-values for each probe and sort it from the smallest to the largest p-value.

```
beta_rawp_df <- data.frame(beta_norm, pval_raw)
beta_rawp_df_sorted <- beta_rawp_df[order(beta_rawp_df$pval_raw), ]
summary(beta_rawp_df_sorted)
```

```
##  X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01 X5775278078_R05C01
##  Min.   :0.00000    Min.   :0.00000    Min.   :0.00000    Min.   :0.0000
##  1st Qu.:0.04591    1st Qu.:0.05082    1st Qu.:0.04892    1st Qu.:0.0514
##  Median :0.66239    Median :0.66205    Median :0.65265    Median :0.6483
##  Mean   :0.50624    Mean   :0.50733    Mean   :0.50645    Mean   :0.5066
##  3rd Qu.:0.88802    3rd Qu.:0.88803    3rd Qu.:0.88615    3rd Qu.:0.8874
##  Max.   :0.99361    Max.   :0.99456    Max.   :0.99461    Max.   :0.9945
##  X5775278078_R05C02 X5930514034_R01C02 X5930514035_R04C02 X5930514035_R06C02
##  Min.   :0.00000    Min.   :0.00000    Min.   :0.0000     Min.   :0.00000
##  1st Qu.:0.05529    1st Qu.:0.04776    1st Qu.:0.0534     1st Qu.:0.04739
##  Median :0.65274    Median :0.64770    Median :0.6582     Median :0.64441
##  Mean   :0.50920    Mean   :0.50574    Mean   :0.5089     Mean   :0.50431
##  3rd Qu.:0.88597    3rd Qu.:0.88921    3rd Qu.:0.8882     3rd Qu.:0.88865
##  Max.   :0.99310    Max.   :0.99389    Max.   :0.9948     Max.   :0.99429
##     pval_raw
##  Min.   :0.02857
##  1st Qu.:0.20000
##  Median :0.48571
##  Mean   :0.52682
##  3rd Qu.:0.88571
##  Max.   :1.00000
```

Step 10

> 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?

For doing multiple-testing corrections, I will use the `p.adjust` function, specifying which correction I want to use. In this report, I will use the False Discovery Rate (`BH`) and the Bonferroni correction (`Bonf`). I will apply them to the sorted raw p-values sliced from the data frame of step 9.

```
raw_p_sorted <- beta_rawp_df_sorted[, 9]
summary(raw_p_sorted)
```

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

```
BH_p <- p.adjust(raw_p_sorted, "BH")
summary(BH_p)
```

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

```
Bonf_p <- p.adjust(raw_p_sorted, "bonferroni")
summary(Bonf_p)
```

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

We can note how the Bonferroni correction is too stringent, bringing all the p-values to 1. Also with the BH correction the smallest p-value is only 0.47, so definitively not significant. I can now count how many probes have a p-value below the threshold of 0.05.

```
diff_met_raw_p <- raw_p_sorted[raw_p_sorted <= 0.05]
length(diff_met_raw_p)
```

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

```
summary(diff_met_raw_p)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02857 0.02857 0.02857 0.02857 0.02857 0.02857
```

```
diff_met_BH <- BH_p[BH_p <= 0.05]
length(diff_met_BH)
```

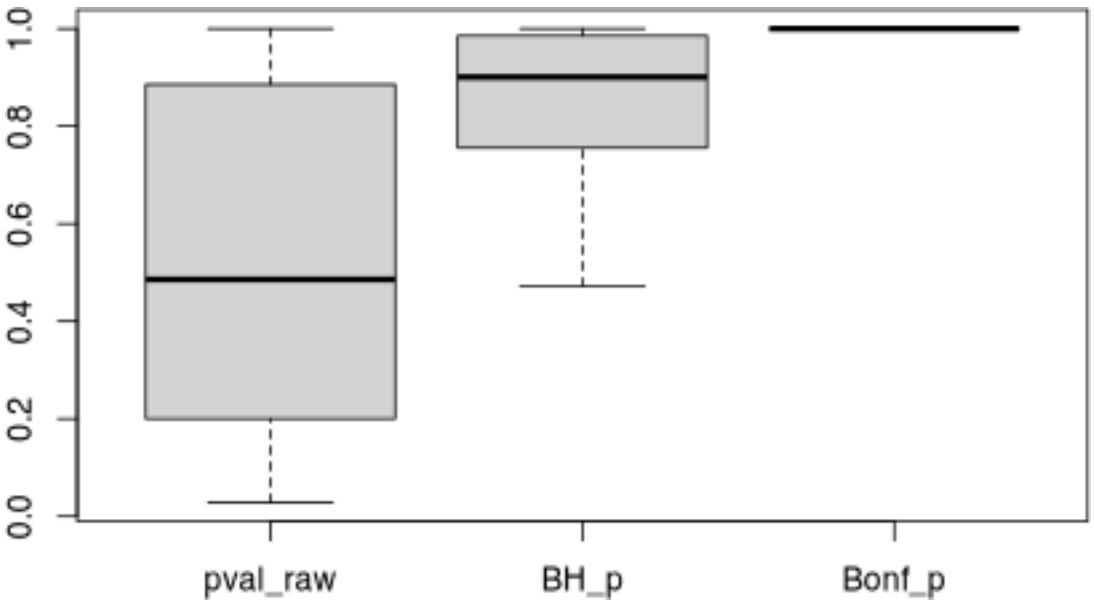
```
## [1] 0
```

```
diff_met_Bonf <- Bonf_p[Bonf_p <= 0.05]
length(diff_met_Bonf)
```

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

There are no significant probes after multiple test correction. 29388 probes have a raw p-value under 0.05. I can create a data frame including the beta values, raw p-values, and corrected p-values for all the probes. I will also create a boxplot of the raw and corrected p-values to look at their distribution.

```
complete_beta_p_df <- data.frame(beta_rawp_df_sorted, BH_p, Bonf_p)
boxplot(complete_beta_p_df[, 9:11])
```



Step 11

Produce a heatmap of the top 100 differentially methylated probes.

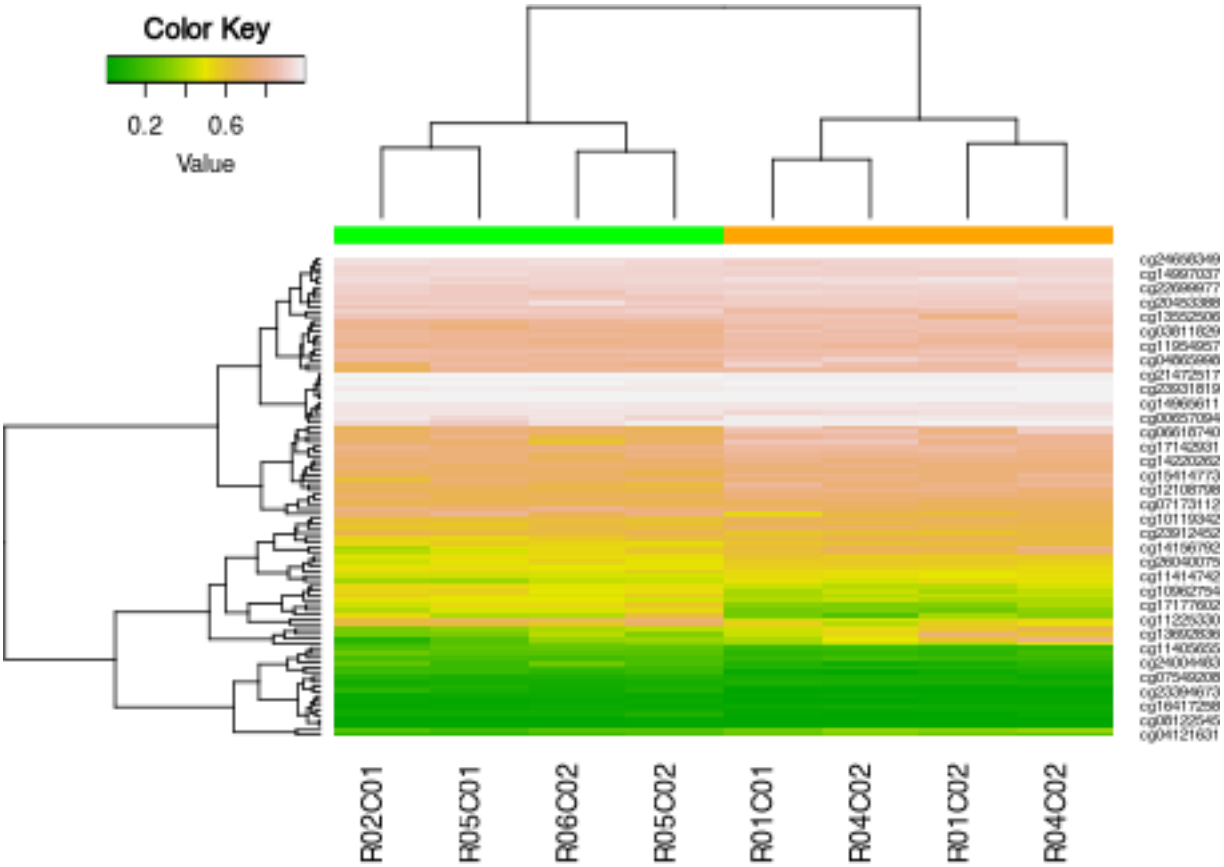
I will use the function `heatmap.2` to produce a heatmap of the top-100 differentially methylated probes (according to raw p-value). I first produce an input matrix by slicing the first 100 rows of the sorted data frame from step 10. I also slice the data frame by column to exclude p-values and retain only beta-values.

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

```
##  X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01 X5775278078_R05C01
##  Min.   :0.01368    Min.   :0.01535    Min.   :0.01664    Min.   :0.01643
##  1st Qu.:0.39971    1st Qu.:0.43013    1st Qu.:0.40007    1st Qu.:0.44578
##  Median :0.73226    Median :0.73165    Median :0.68598    Median :0.70404
##  Mean   :0.61602    Mean   :0.61925    Mean   :0.59610    Mean   :0.60328
##  3rd Qu.:0.86825    3rd Qu.:0.85724    3rd Qu.:0.82758    3rd Qu.:0.82634
##  Max.   :0.98686    Max.   :0.98686    Max.   :0.98738    Max.   :0.98721
##  X5775278078_R05C02 X5930514034_R01C02 X5930514035_R04C02 X5930514035_R06C02
##  Min.   :0.01682    Min.   :0.01471    Min.   :0.01465    Min.   :0.01553
##  1st Qu.:0.46763    1st Qu.:0.41911    1st Qu.:0.46339    1st Qu.:0.45596
##  Median :0.67895    Median :0.73837    Median :0.74613    Median :0.67529
##  Mean   :0.61333    Mean   :0.62000    Mean   :0.63148    Mean   :0.60584
##  3rd Qu.:0.82587    3rd Qu.:0.85565    3rd Qu.:0.87327    3rd Qu.:0.82153
##  Max.   :0.98757    Max.   :0.98618    Max.   :0.98696    Max.   :0.98703
```

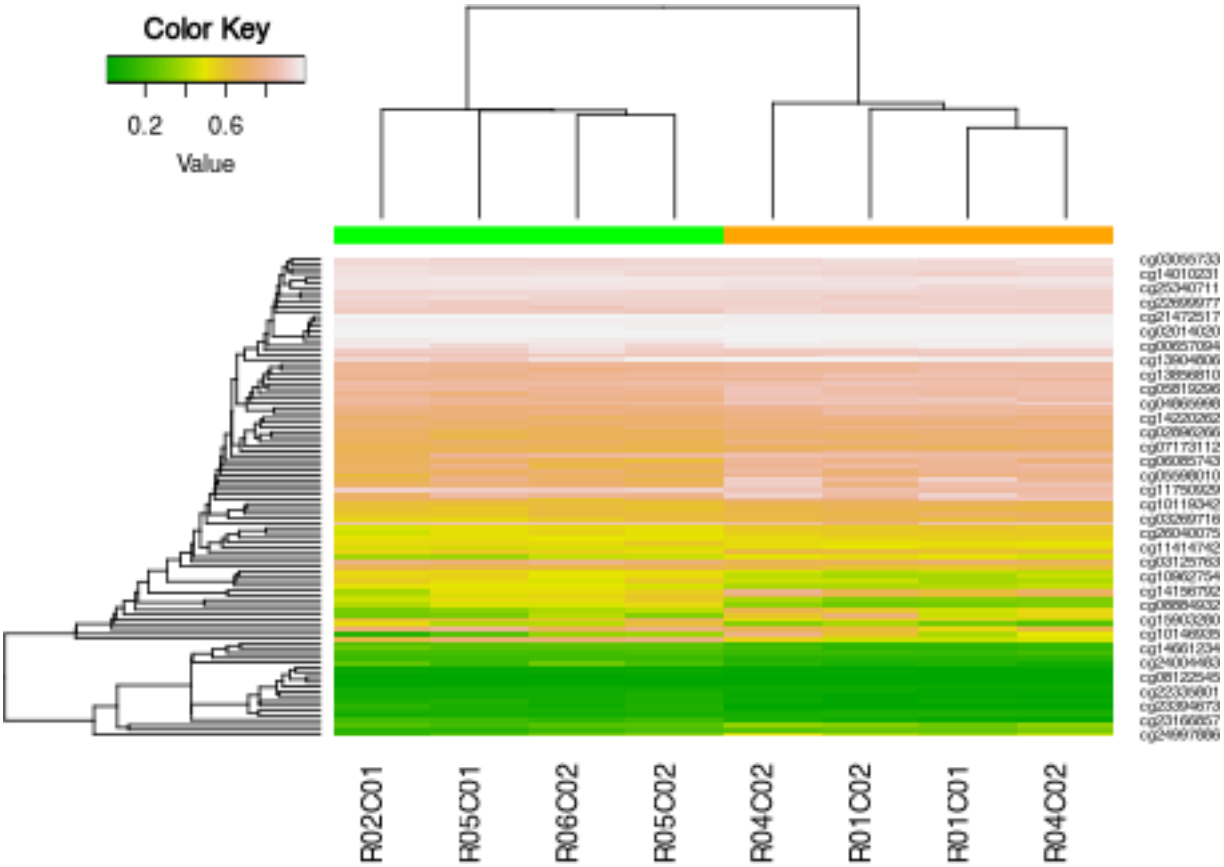
I will now produce the heatmap from this matrix, coloring the samples according to the group they belong to (`DS` or `WT`). For now, I use the default distance metric and linkage method (Euclidean distance and complete linkage).

```
DS_col <- 'orange'
WT_col <- 'green'
colorbar <-
  c(DS_col, DS_col, WT_col, WT_col, WT_col, DS_col, DS_col, WT_col)
heatmap.2(
  input_heatmap,
  col = terrain.colors(100),
  Rowv = T,
  Colv = T,
  dendrogram = "both",
  key = T,
  ColSideColors = colorbar,
  density.info = "none",
  trace = "none",
  scale = "none",
  symm = F
)
```

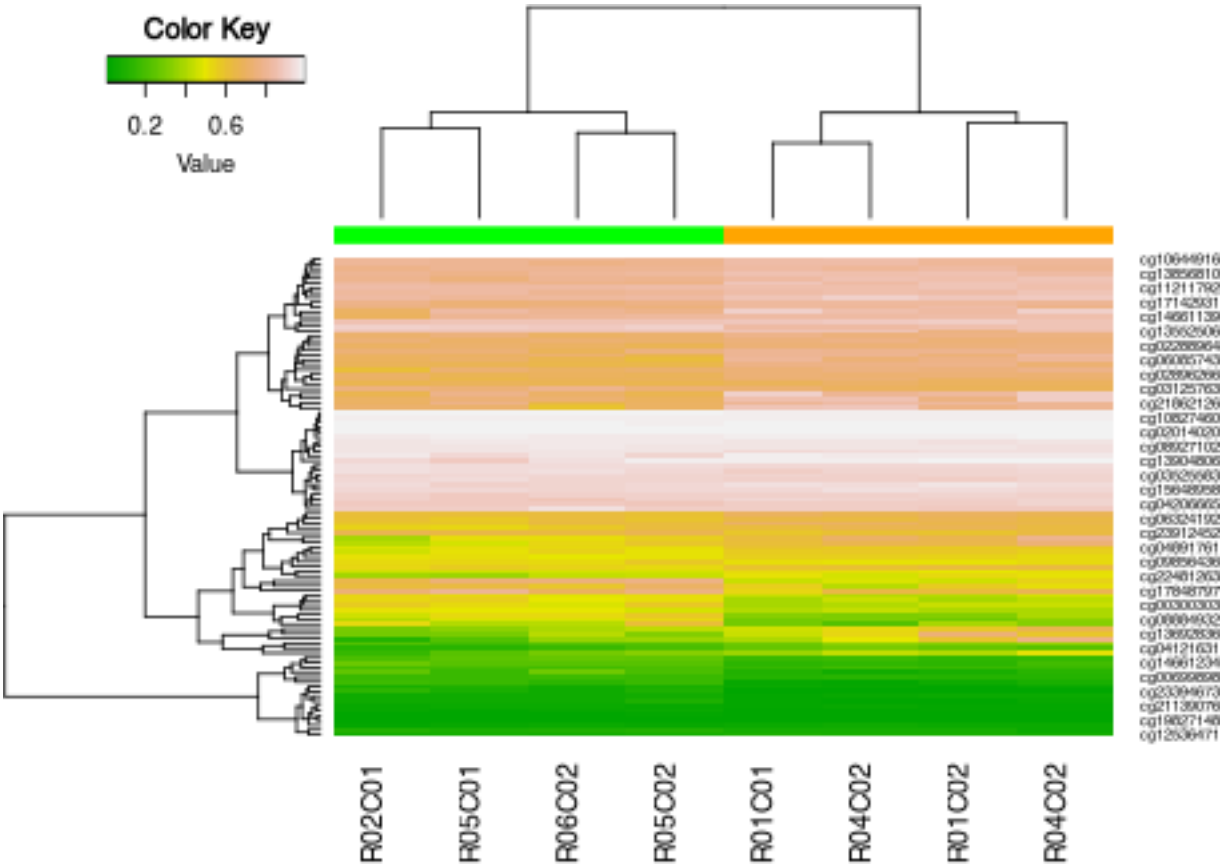


From the heatmap, we notice how the 2 sample groups are well clustered according to the methylation status of the top 100 probes. Some probes are more methylated in one of the groups, other probes in the other. I can repeat the heatmap using single and average linkage.

```
heatmap.2(
  input_heatmap,
  col = terrain.colors(100),
  Rowv = T,
  Colv = T,
  hclustfun = function(x)
    hclust(x, method = 'single'),
  dendrogram = "both",
  key = T,
  ColSideColors = colorbar,
  density.info = "none",
  trace = "none",
  scale = "none",
  symm = F
)
```

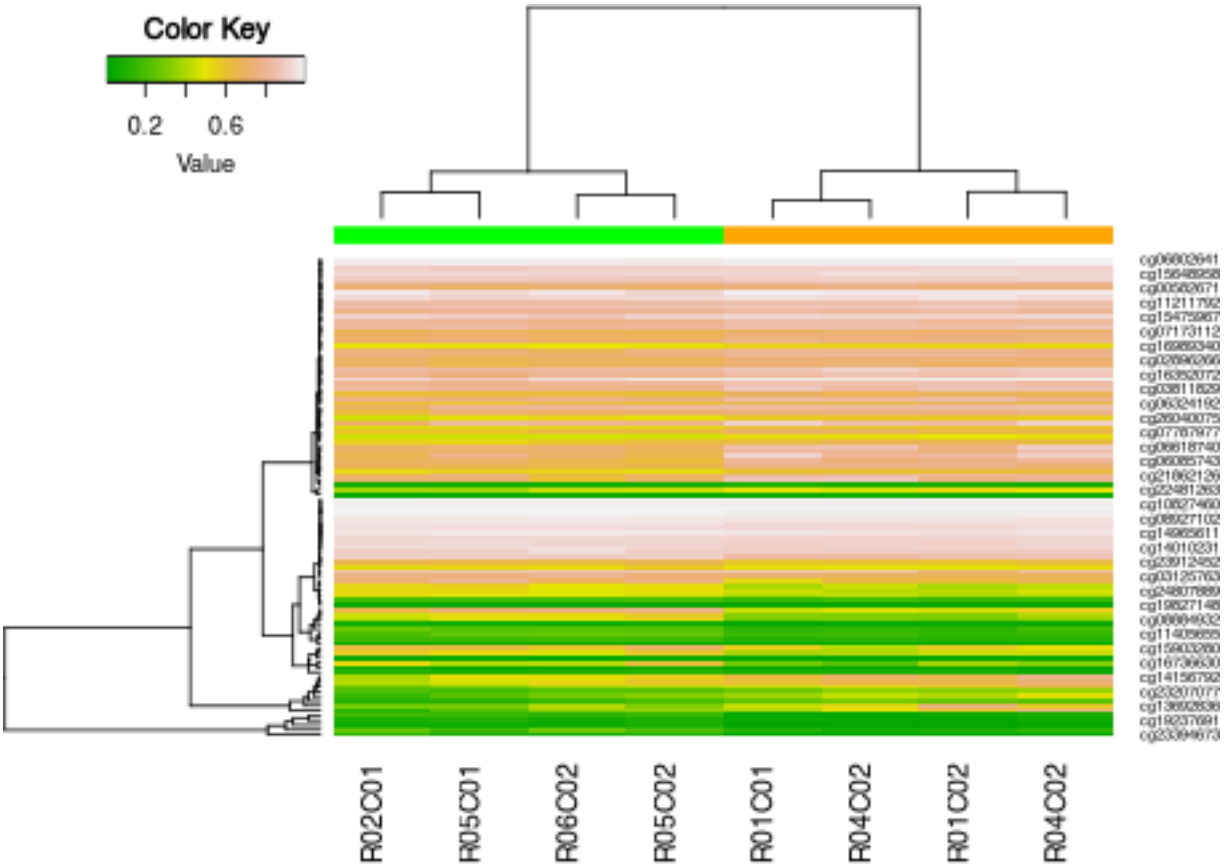



```
heatmap.2(
  input_heatmap,
  col = terrain.colors(100),
  Rowv = T,
  Colv = T,
  hclustfun = function(x)
    hclust(x, method = 'average'),
  dendrogram = "both",
  key = T,
  ColSideColors = colorbar,
  density.info = "none",
  trace = "none",
  scale = "none",
  symm = F
)
```

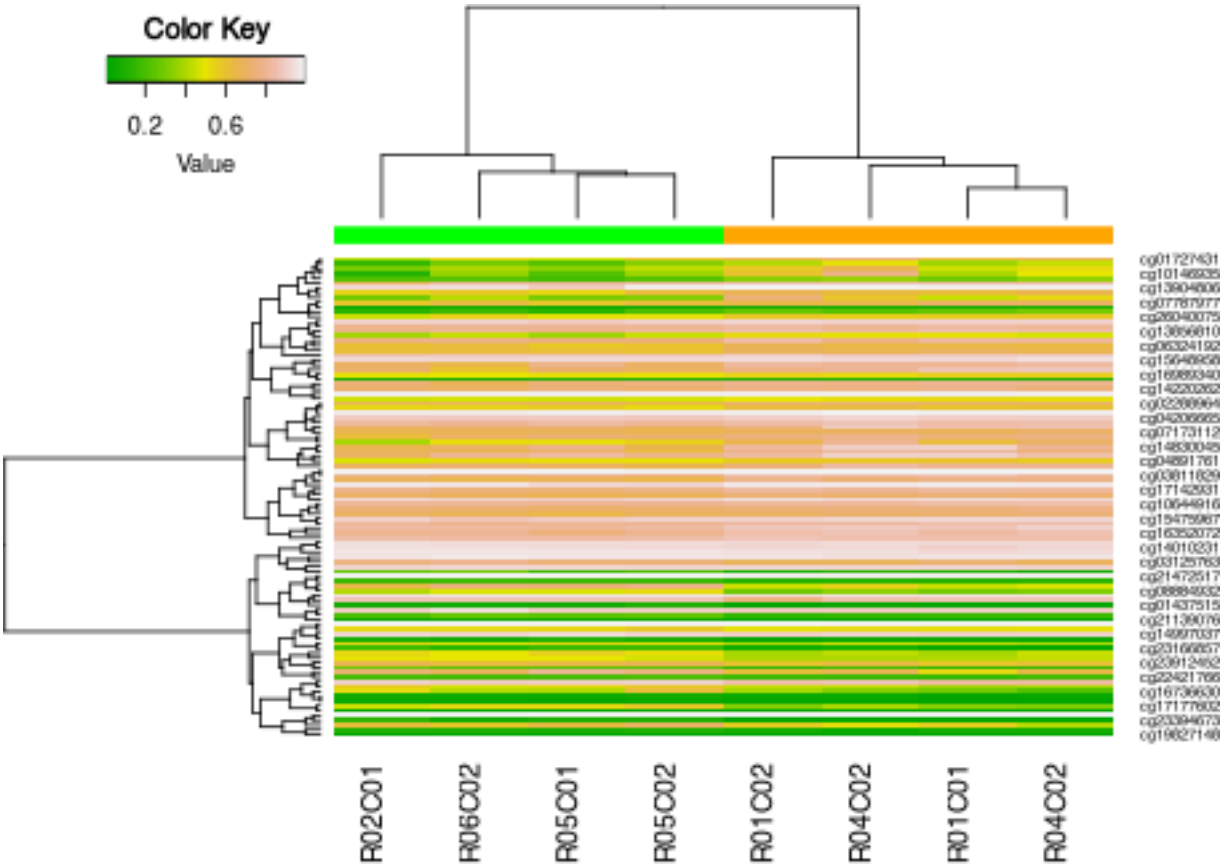


We can notice how the result is not so influenced by the linkage method used. The samples are well clustered in `WT` and `DS` with all the 3 methods. I will also perform hierarchical clustering using the Pearson and Spearman correlation coefficients as distance metrics, instead of Euclidean distance. This time I will use average linkage as a linkage method.

```
heatmap.2(
  input_heatmap,
  col = terrain.colors(100),
  Rowv = T,
  Colv = T,
  distfun = function(x)
    Dist(x, method = "pearson"),
  dendrogram = "both",
  key = T,
  ColSideColors = colorbar,
  density.info = "none",
  trace = "none",
  scale = "none",
  symm = F
)
```




```
heatmap.2(
  input_heatmap,
  col = terrain.colors(100),
  Rowv = T,
  Colv = T,
  distfun = function(x)
    Dist(x, method = "spearman"),
  dendrogram = "both",
  key = T,
  ColSideColors = colorbar,
  density.info = "none",
  trace = "none",
  scale = "none",
  symm = F
)
```



The terminal branches for both samples and probes seem much shorter when using the Pearson correlation coefficient as a distance metric. In general, using Pearson and Spearman correlation makes the profiles appear more closely related than using Euclidean distance.

Step 12

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

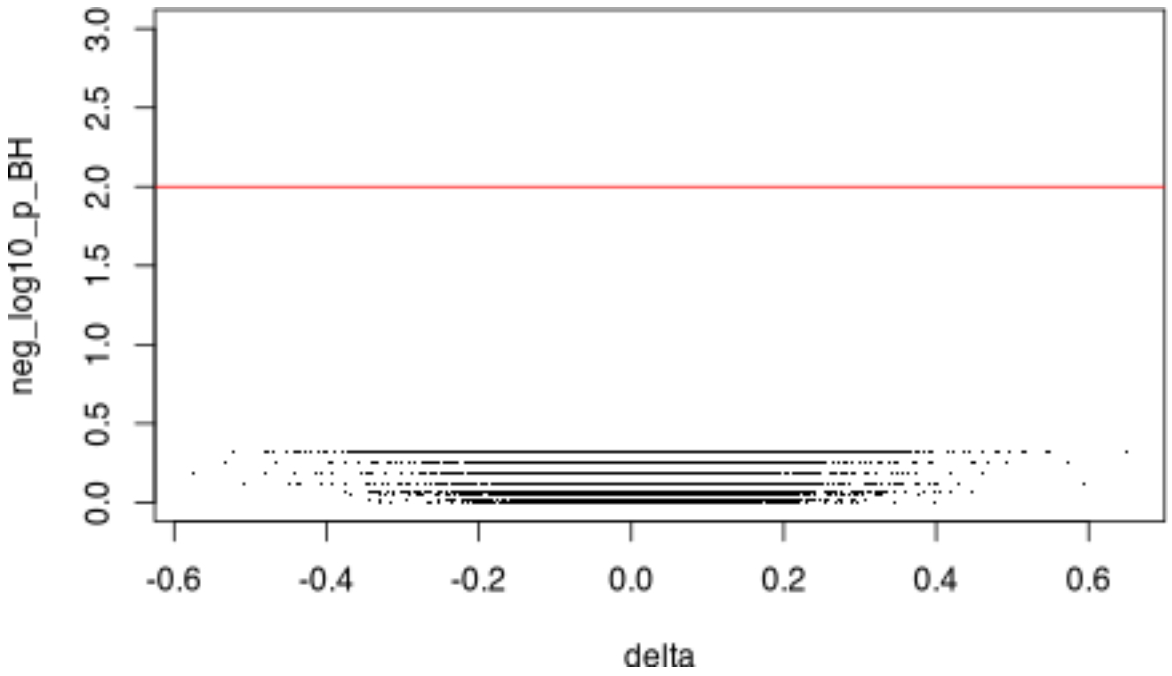
For obtaining a volcano plot, I need the difference in average methylation status for each probe among the 2 sample groups. Since this is not specified in the assignment, I will use the normalized beta values from step 7 in this section, and the `BH` p-values from step 10. Here I calculate the mean beta values for each probe in each sample group and their difference (across the 2 sample groups).

```
beta_norm_sorted <- complete_beta_p_df[, 1:8]
beta_norm_sorted_DS <- beta_norm_sorted[, groups == "DS"]
beta_norm_sorted_WT <- beta_norm_sorted[, groups == "WT"]
mean_beta_DS <- apply(beta_norm_sorted_DS, 1, mean)
mean_beta_WT <- apply(beta_norm_sorted_WT, 1, mean)
delta <- mean_beta_DS - mean_beta_WT
summary(delta)
```

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

At this point, `delta` contains the differential mean beta values between sample groups, for each position. I can now plot `delta` against the $-\log_{10}$ of the corrected p-values.

```
neg_log10_p_BH <- -log10(complete_beta_p_df[, 10])
plot(delta, neg_log10_p_BH, ylim = c(0, 3), pch = '.')
abline(a = -log10(0.01),
       b = 0,
       col = "red")
```



We can see how the p-values are binned at discrete levels (a consequence of the Mann-Whitney test) and nothing reaches the significance threshold of 0.01.

For obtaining the Manhattan plot, I first merge the data frame containing the p-values for each probe with the annotation of the probes, using the `merge` function and merging according to the probe name.

```
df_manhattan <-
  data.frame(rownames(complete_beta_p_df), complete_beta_p_df)
colnames(df_manhattan)[1] <- 'Name'
annotation <- getAnnotation(RGSet)
colnames(annotation)
```

```
##  [1] "chr"                      "pos"
##  [3] "strand"                   "Name"
##  [5] "AddressA"                 "AddressB"
##  [7] "ProbeSeqA"                "ProbeSeqB"
##  [9] "Type"                     "NextBase"
## [11] "Color"                    "Probe_rs"
## [13] "Probe_maf"                "CpG_rs"
## [15] "CpG_maf"                  "SBE_rs"
## [17] "SBE_maf"                  "Islands_Name"
## [19] "Relation_to_Island"       "Forward_Sequence"
## [21] "SourceSeq"                "Random_Loci"
## [23] "Methyl27_Loci"            "UCSC_RefGene_Name"
## [25] "UCSC_RefGene_Accession"   "UCSC_RefGene_Group"
## [27] "Phantom"                  "DMR"
## [29] "Enhancer"                 "HMM_Island"
## [31] "Regulatory_Feature_Name"  "Regulatory_Feature_Group"
```

```
## [33] "DHS"
```

```
complete_df_annotated <- merge(df_manhattan, annotation, by = "Name")
colnames(complete_df_annotated)
```

```
##  [1] "Name"                     "X5775278051_R01C01"
##  [3] "X5775278051_R04C02"       "X5775278078_R02C01"
##  [5] "X5775278078_R05C01"       "X5775278078_R05C02"
##  [7] "X5930514034_R01C02"       "X5930514035_R04C02"
##  [9] "X5930514035_R06C02"       "pval_raw"
## [11] "BH_p"                     "Bonf_p"
## [13] "chr"                      "pos"
## [15] "strand"                   "AddressA"
## [17] "AddressB"                 "ProbeSeqA"
## [19] "ProbeSeqB"                "Type"
## [21] "NextBase"                 "Color"
## [23] "Probe_rs"                 "Probe_maf"
## [25] "CpG_rs"                   "CpG_maf"
## [27] "SBE_rs"                   "SBE_maf"
## [29] "Islands_Name"             "Relation_to_Island"
## [31] "Forward_Sequence"         "SourceSeq"
## [33] "Random_Loci"              "Methyl27_Loci"
## [35] "UCSC_RefGene_Name"        "UCSC_RefGene_Accession"
## [37] "UCSC_RefGene_Group"       "Phantom"
## [39] "DMR"                      "Enhancer"
## [41] "HMM_Island"               "Regulatory_Feature_Name"
## [43] "Regulatory_Feature_Group" "DHS"
```

Now I extract only the columns needed for obtaining the Manhattan plot (chromosome, position, `BH` p-value). I also rename the columns for clarity and order the levels according to the chromosome number.

```
chr_vect <- factor(
  complete_df_annotated$chr,
  levels = c(
    "chr1",
    "chr2",
    "chr3",
    "chr4",
    "chr5",
    "chr6",
    "chr7",
    "chr8",
    "chr9",
    "chr10",
    "chr11",
    "chr12",
    "chr13",
    "chr14",
    "chr15",
    "chr16",
    "chr17",
    "chr18",
    "chr19",
    "chr20",
    "chr21",
    "chr22",
```

```
    "chrX",
    "chrY"
  )
)
input_manhattan <-
  data.frame(chr_vect,
             complete_df_annotated$pos,
             complete_df_annotated$BH_p)
colnames(input_manhattan) <- c('chr', 'pos', 'BH_pv
al')
str(input_manhattan)
```

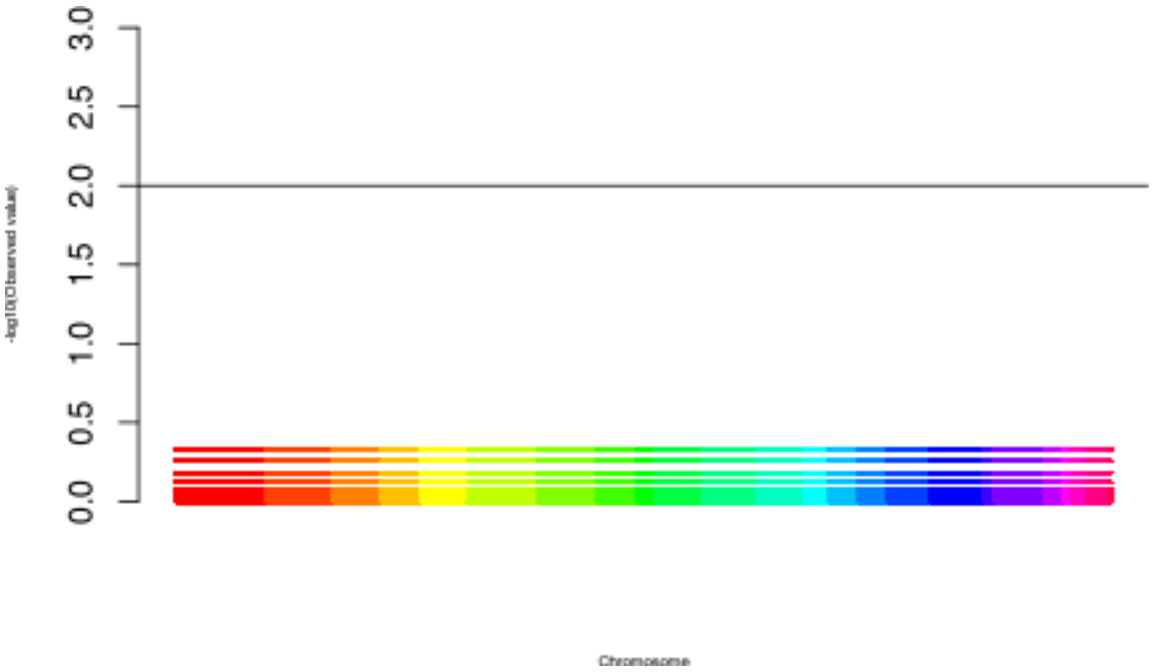
```
## 'data.frame':    485512 obs. of  3 variables:
##  $ chr      : Factor w/ 24 levels "chr1","chr2",..: 16 3 3 1 8 14 16 8 1 15 ...
##  $ pos      : int  53468112 37459206 171916037 91194674 42263294 69341139 28890100 41167802 230560793
##  $ BH_pv
## al: num  0.755 0.844 0.951 0.901 0.472 ...
```

Finally, I can obtain the Manhattan plot.

```
palette <- rainbow(24)
mhtplot(
  input_manhattan,
  control = mht.control(colors = palette),
  ylim = c(0, 3),
  pch = 16
)
```

```
## 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
```

```
axis(2, cex = 0.5)
abline(a = -log10(0.01), b = 0)
```



We can see how all the probes have a corrected p-value much higher than the significance threshold, and there are no significant differences across chromosomes. We can also notice the binned distribution of the Mann-Whitney p-values.

Both the volcano plot and the Manhattan plot have an appearance that is quite different from what I am used to. I suspect that this is due to the use of a non-parametric test (Mann-Whitney U test). To test this hypothesis I will here repeat the analysis using the Welch test.

The code for the Welch test:

```
t_test_all_rows <- function(x) {
  t_val <- t.test(x ~ targets$Group)
  return(t_val$p.value)
}

pval_raw_t <- apply(beta_norm, 1, t_test_all_rows)
summary(pval_raw_t)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1994  0.4399  0.4593  0.7107  1.0000
```

```
length(pval_raw_t)
```

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

The code for obtaining the joined dataset and for multiple testing correction:

```
beta_rawp_df_t <- data.frame(beta_norm, pval_raw_t)
beta_rawp_df_sorted_t <-
  beta_rawp_df_t[order(beta_rawp_df_t$pval_raw_t), ]
summary(beta_rawp_df_sorted_t)
```

```
##  X5775278051_R01C01 X5775278051_R04C02 X5775278078_R02C01 X5775278078_R05C01
##  Min.   :0.00000    Min.   :0.00000    Min.   :0.00000    Min.   :0.0000
##  1st Qu.:0.04591    1st Qu.:0.05082    1st Qu.:0.04892    1st Qu.:0.0514
##  Median :0.66239    Median :0.66205    Median :0.65265    Median :0.6483
```

```
##  Mean   :0.50624    Mean   :0.50733    Mean   :0.50645    Mean   :0.5066
##  3rd Qu.:0.88802    3rd Qu.:0.88803    3rd Qu.:0.88615    3rd Qu.:0.8874
##  Max.   :0.99361    Max.   :0.99456    Max.   :0.99461    Max.   :0.9945
##  X5775278078_R05C02 X5930514034_R01C02 X5930514035_R04C02 X5930514035_R06C02
##  Min.   :0.00000    Min.   :0.00000    Min.   :0.0000     Min.   :0.00000
##  1st Qu.:0.05529    1st Qu.:0.04776    1st Qu.:0.0534     1st Qu.:0.04739
##  Median :0.65274    Median :0.64770    Median :0.6582     Median :0.64441
##  Mean   :0.50920    Mean   :0.50574    Mean   :0.5089     Mean   :0.50431
##  3rd Qu.:0.88597    3rd Qu.:0.88921    3rd Qu.:0.8882     3rd Qu.:0.88865
##  Max.   :0.99310    Max.   :0.99389    Max.   :0.9948     Max.   :0.99429
##    pval_raw_t
##  Min.   :0.0000
##  1st Qu.:0.1994
##  Median :0.4399
##  Mean   :0.4593
##  3rd Qu.:0.7107
##  Max.   :1.0000
```

```
raw_p_sorted_t <- beta_rawp_df_sorted_t[, 9]
summary(raw_p_sorted_t)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1994  0.4399  0.4593  0.7107  1.0000
```

```
BH_p_t <- p.adjust(raw_p_sorted_t, "BH")
summary(BH_p)
```

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

```
Bonf_p_t <- p.adjust(raw_p_sorted_t, "bonferroni")
summary(Bonf_p_t)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0173  1.0000  1.0000  1.0000  1.0000  1.0000
```

```
complete_beta_p_df_t <-
  data.frame(beta_rawp_df_sorted_t, BH_p_t, Bonf_p_t)
```

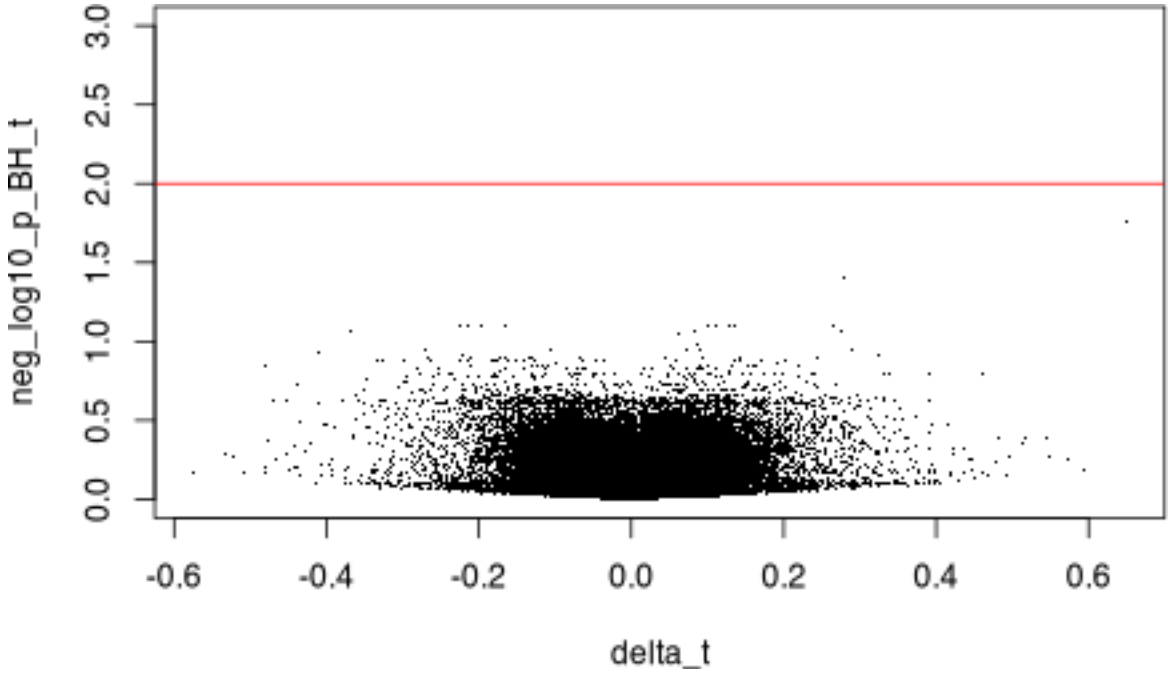
The code for the volcano plot:

```
beta_norm_sorted_t <- complete_beta_p_df_t[, 1:8]
beta_norm_sorted_DS_t <- beta_norm_sorted_t[, groups == "DS"]
beta_norm_sorted_WT_t <- beta_norm_sorted_t[, groups == "WT"]
mean_beta_DS_t <- apply(beta_norm_sorted_DS_t, 1, mean)
mean_beta_WT_t <- apply(beta_norm_sorted_WT_t, 1, mean)
delta_t <- mean_beta_DS_t - mean_beta_WT_t
summary(delta_t)
```

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

```
neg_log10_p_BH_t <- -log10(complete_beta_p_df_t[, 10])
plot(delta_t,
     neg_log10_p_BH_t,
     ylim = c(0, 3),
     pch = '.')
abline(a = -log10(0.01),
```

```
       b = 0,
       col = "red")
```



The code for the Manhattan plot:

```
df_manhattan_t <-
  data.frame(rownames(complete_beta_p_df_t), complete_beta_p_df_t)
colnames(df_manhattan_t)[1] <- 'Name'
annotation <- getAnnotation(RGSet)
colnames(annotation)
```

```
##  [1] "chr"                      "pos"
##  [3] "strand"                   "Name"
##  [5] "AddressA"                 "AddressB"
##  [7] "ProbeSeqA"                "ProbeSeqB"
##  [9] "Type"                     "NextBase"
## [11] "Color"                    "Probe_rs"
## [13] "Probe_maf"                "CpG_rs"
## [15] "CpG_maf"                  "SBE_rs"
## [17] "SBE_maf"                  "Islands_Name"
## [19] "Relation_to_Island"       "Forward_Sequence"
## [21] "SourceSeq"                "Random_Loci"
## [23] "Methyl27_Loci"            "UCSC_RefGene_Name"
## [25] "UCSC_RefGene_Accession"   "UCSC_RefGene_Group"
## [27] "Phantom"                  "DMR"
## [29] "Enhancer"                 "HMM_Island"
## [31] "Regulatory_Feature_Name"  "Regulatory_Feature_Group"
## [33] "DHS"
```

```
complete_df_annotated_t <-
  merge(df_manhattan_t, annotation, by = "Name")
colnames(complete_df_annotated_t)
```

```
##  [1] "Name"                     "X5775278051_R01C01"
```

```
##  [3] "X5775278051_R04C02"       "X5775278078_R02C01"
##  [5] "X5775278078_R05C01"       "X5775278078_R05C02"
##  [7] "X5930514034_R01C02"       "X5930514035_R04C02"
##  [9] "X5930514035_R06C02"       "pval_raw_t"
## [11] "BH_p_t"                   "Bonf_p_t"
## [13] "chr"                      "pos"
## [15] "strand"                   "AddressA"
## [17] "AddressB"                 "ProbeSeqA"
## [19] "ProbeSeqB"                "Type"
## [21] "NextBase"                 "Color"
## [23] "Probe_rs"                 "Probe_maf"
## [25] "CpG_rs"                   "CpG_maf"
## [27] "SBE_rs"                   "SBE_maf"
## [29] "Islands_Name"             "Relation_to_Island"
## [31] "Forward_Sequence"         "SourceSeq"
## [33] "Random_Loci"              "Methyl27_Loci"
## [35] "UCSC_RefGene_Name"        "UCSC_RefGene_Accession"
## [37] "UCSC_RefGene_Group"       "Phantom"
## [39] "DMR"                      "Enhancer"
## [41] "HMM_Island"               "Regulatory_Feature_Name"
## [43] "Regulatory_Feature_Group" "DHS"
```

```r
chr_vect_t <-
  factor(
    complete_df_annotated_t$chr,
    levels = c(
      "chr1",
      "chr2",
      "chr3",
      "chr4",
      "chr5",
      "chr6",
      "chr7",
      "chr8",
      "chr9",
      "chr10",
      "chr11",
      "chr12",
      "chr13",
      "chr14",
      "chr15",
      "chr16",
      "chr17",
      "chr18",
      "chr19",
      "chr20",
      "chr21",
      "chr22",
      "chrX",
      "chrY"
    )
  )
input_manhattan_t <-
  data.frame(chr_vect_t,
```

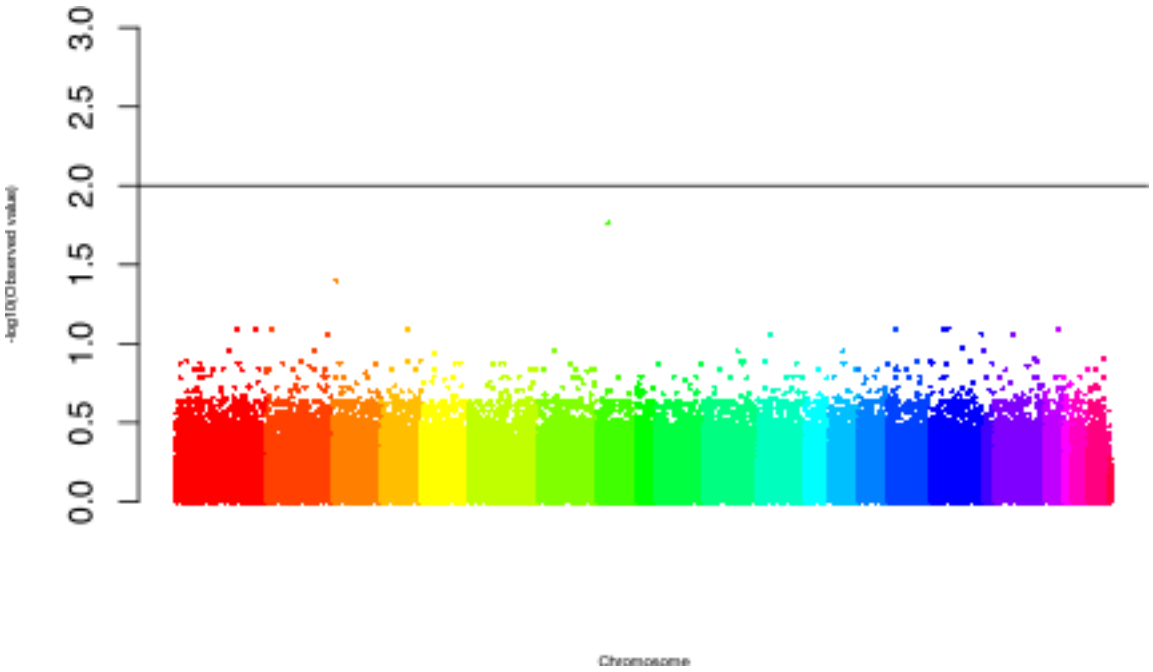
```
            complete_df_annotated_t$pos,
            complete_df_annotated_t$BH_p_t)
colnames(input_manhattan_t) <- c('chr', 'pos', 'BH_pval')
str(input_manhattan_t)
```

```
## 'data.frame':    485512 obs. of  3 variables:
##  $ chr    : Factor w/ 24 levels "chr1","chr2",..: 16 3 3 1 8 14 16 8 1 15 ...
##  $ pos    : int  53468112 37459206 171916037 91194674 42263294 69341139 28890100 41167802 230560793 1
##  $ BH_pval: num  0.779 0.829 0.861 0.871 0.716 ...
```

```
palette_t <- rainbow(24)
mhtplot(
  input_manhattan_t,
  control = mht.control(colors = palette_t),
  ylim = c(0, 3),
  pch = 16
)
```

```
## 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
```

```
axis(2, cex = 0.5)
abline(a = -log10(0.01), b = 0)
```



Now the volcano and Manhattan plots look more familiar. Therefore, I conclude that the peculiar appearance of the volcano and Manhattan plots is due to the use of a non-parametric test, which does not have enough statistical power to detect differentially methylated probes in this dataset.

Optional task

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

First I extract the rows relative to probes belonging to chromosome 21 from the complete dataset prepared in step 12 and I subset it according to the group of the samples.

```
chr21_complete <-
  complete_df_annotated[complete_df_annotated$chr == 'chr21', ]
chr21_beta <- chr21_complete[, 2:9]
chr21_beta_WT <- chr21_beta[, groups == 'WT']
chr21_beta_DS <- chr21_beta[, groups == 'DS']
dim(chr21_beta_DS)
```

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

```
dim(chr21_beta_WT)
```

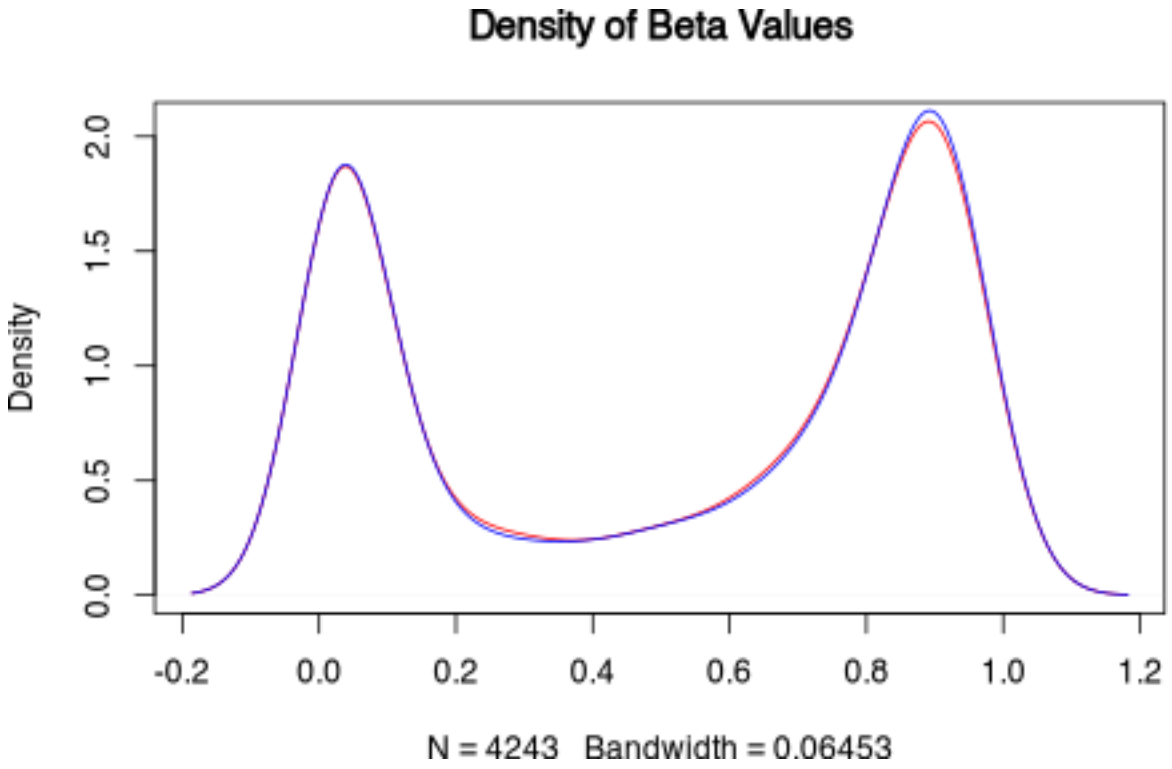
```
## [1] 4243    4
```

Now I calculate the mean beta values of each probe in each of the sample groups, and the density of mean beta values.

```
chr21_beta_DS_mean <- apply(chr21_beta_DS, MARGIN = 1, mean, na.rm = T)
chr21_beta_WT_mean <- apply(chr21_beta_WT, MARGIN = 1, mean, na.rm = T)
d_mean_chr21_beta_DS <- density(chr21_beta_DS_mean)
d_mean_chr21_beta_WT <- density(chr21_beta_WT_mean)
```

Finally, I plot the densities of mean beta values differentiated by sample group, only for probes on chromosome 21.

```
plot(d_mean_chr21_beta_DS, main = "Density of Beta Values", col = "red")
lines(d_mean_chr21_beta_WT, main = "Density of Beta Values", col = "blue")
```



The red line refers to the `DS` group, while the blue line refers to the `WT` group. I observe that there are no obviuos differences among `DS` and `WT` samples regarding the distribution of mean beta values for probes on chromosome 21. Only the methylated peak of beta values seems slightly more pronounced in the `WT` group.

Now I want to evaluate how many probes are differentially methylated on chromosome 21. I will just extract this information from the complete data frame obtained in the previous steps. I will use raw p-values here.

```
diff_met_chr21 <- chr21_complete[chr21_complete$pval_raw <= 0.01, ]
dim(diff_met_chr21)
```

```
## [1]  0 44
```

On chromosome 21, 44 probes are differentially methylated according to raw p-value adopting a 0.01 threshold. No significant p-values are present in the whole dataset after multiple testing correction, so there is no need to look for significant probes on chromosome 21.