A cross-package Bioconductor workflow for analysing methylation array data

***Jovana Maksimovic*[*], Belinda Phipson and Alicia Oshlack**

[*]jovana.maksimovic@mcri.edu.au
(mailto:jovana.maksimovic@mcri.edu.au)

20 December 2019

Abstract

Methylation in the human genome is known to be associated with development and disease. The Illumina Infinium methylation arrays are by far the most common way to interrogate methylation across the human genome. This paper provides a Bioconductor workflow using multiple packages for the analysis of methylation array data. Specifically, we demonstrate the steps involved in a typical differential methylation analysis pipeline including: quality control, filtering, normalization, data exploration and statistical testing for probe-wise differential methylation. We further outline other analyses such as differential methylation of regions, differential variability analysis, estimating cell type composition and gene ontology testing. Finally, we provide some examples of how to visualise methylation array data.

Contents

References

R version: R Under development (unstable) (2019-12-14 r77569)

Bioconductor version: 3.11

Package: 1.11.0

1 Introduction

DNA methylation, the addition of a methyl group to a CG dinucleotide of the DNA, is the most extensively studied epigenetic mark due to its role in both development and disease (Bird 2002; Laird 2003). Although DNA methylation can be measured in several ways, the epigenetics community has enthusiastically embraced the Illumina HumanMethylation450 (450k) array (Bibikova et al. 2011) as a cost-effective way to assay methylation across the human genome. More recently, Illumina has increased the genomic coverage of the platform to >850,000 sites with the release of their MethylationEPIC (850k) array. As methylation arrays are likely to remain popular for measuring methylation for the foreseeable future, it is necessary to provide robust workflows for methylation array analysis.

Measurement of DNA methylation by Infinium technology (Infinium I) was first employed by Illumina on the HumanMethylation27 (27k) array (Bibikova et al. 2009), which measured methylation at approximately 27,000 CpGs, primarily in gene promoters. Like bisulfite sequencing, the Infinium assay detects methylation status at single base resolution. However, due to its relatively limited coverage the array platform was not truly considered "genome-wide" until the arrival of the 450k array. The 450k array increased the genomic coverage of the platform to over 450,000 gene-centric sites by combining the original Infinium I assay with the novel Infinium II probes. Both assay types employ 50bp probes that query a [C/T] polymorphism created by bisulfite conversion of unmethylated cytosines in the genome, however, the Infinium I and II assays differ in the number of beads required to detect methylation at a single locus. Infinium I uses two bead types per CpG, one for each of the methylated and unmethylated states (Figure 1a). In contrast, the Infinium II design uses one bead type and the methylated state is determined at the single base extension step after hybridization (Figure 1b). The 850k array also uses a combination of the Infinium I and II assays but achieves additional coverage by increasing the size of each array; a 450k slide contains 12 arrays whilst the 850k has only 8.

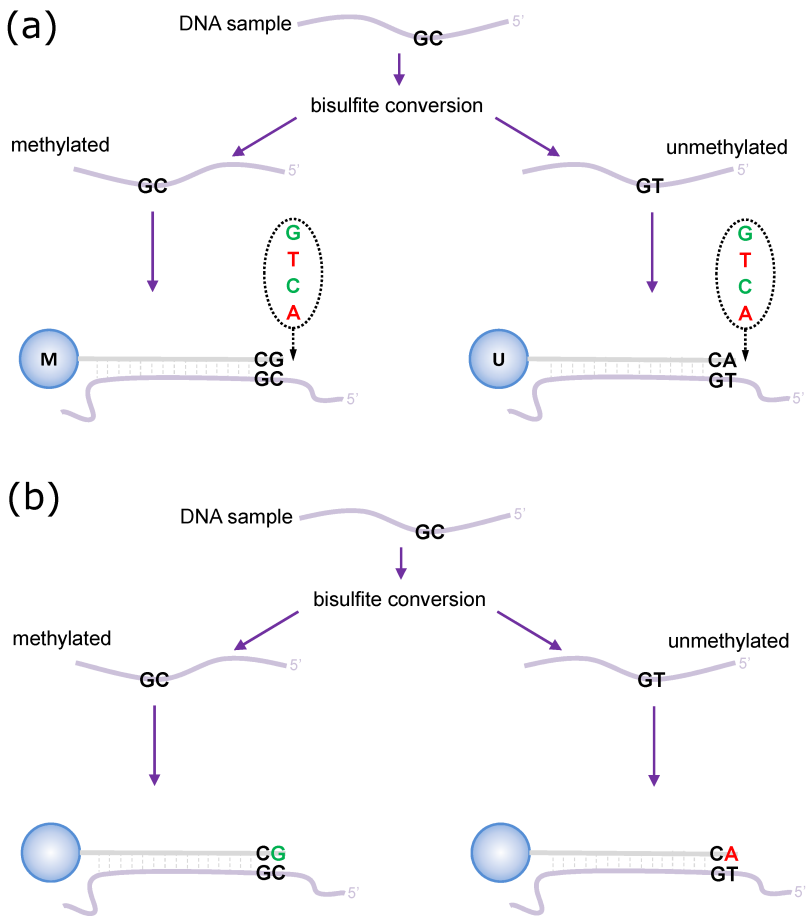

Figure 1: **Illumina Infinium HumanMethylation450 assay, reproduced from Maksimovic, Gordon and Oshlack 2012**
(a) Infinium I assay. Each individual CpG is interrogated using two bead types: methylated (M) and unmethylated (U). Both bead types will incorporate the same labeled nucleotide for the same target CpG, thereby producing the same color fluorescence. The nucleotide that is added is determined by the base downstream of the 'C' of the target CpG. The proportion of methylation can be calculated by comparing the intensities from the two different probes in the same color. (b) Infinium II assay. Each target CpG is interrogated using a single bead type. Methylation state is detected by single base extension at the position of the 'C' of the target CpG, which always results in the addition of a labeled 'G' or 'A' nucleotide, complementary to either the 'methylated' C or 'unmethylated' T, respectively. Each locus is detected in two colors, and methylation status is determined by comparing the two colors from the one position.

Regardless of the Illumina array version, for each CpG, there are two measurements: a methylated intensity (denoted by M) and an unmethylated intensity (denoted by U). These intensity values can be used to determine the proportion of methylation at each CpG locus. Methylation levels are commonly reported as either beta values ($\beta = M/(M + U)$) or M-values ($Mvalue = log2(M/U)$). For practical purposes, a small offset, α, can be added to the denominator of the β value equation to avoid dividing by small values, which is the default behaviour of the `getBeta` function in *minfi*. The default value for α is 100. It may also be desirable to add a small offset to the numerator and denominator when calculating M-values to avoid dividing by zero in rare cases, however the default `getM` function in *minfi* does not do this. Beta values and M-values are related through a logit transformation. Beta values are generally preferable for describing the level of methylation at a locus or for graphical presentation because

percentage methylation is easily interpretable. However, due to their distributional properties, M-values are more appropriate for statistical testing (Du et al. 2010).

In this workflow, we will provide examples of the steps involved in analysing methylation array data using R (R Core Team 2014) and Bioconductor (Huber et al. 2015), including: quality control, filtering, normalization, data exploration and probe-wise differential methylation analysis. We will also cover other approaches such as differential methylation analysis of regions, differential variability analysis, gene ontology analysis and estimating cell type composition. Finally, we will provide some examples of useful ways to visualise methylation array data.

2 Differential methylation analysis

2.1 Obtaining the data

The data required for this workflow has been bundled with the R package that contains this workflow document. Alternatively, it can be obtained from figshare (https://figshare.com/s/7a37f43c0ca2fec4669e). If you choose to download it seperately, once the data has been downloaded, it needs to be extracted from the archive. This will create a folder called `data`, which contains all the files necessary to execute the workflow.

Once the data has been downloaded and extracted, there should be a folder called `data` that contains all the files necessary to execute the workflow.

```
# set up a path to the data directory
dataDirectory <- system.file("extdata", package = "methylationA
rrayAnalysis")
# list the files
list.files(dataDirectory, recursive = TRUE)
```

```
##  [1] "48639-non-specific-probes-Illumina450k.csv"
##  [2] "5975827018/5975827018_R06C02_Grn.idat"
##  [3] "5975827018/5975827018_R06C02_Red.idat"
##  [4] "6264509100/6264509100_R01C01_Grn.idat"
##  [5] "6264509100/6264509100_R01C01_Red.idat"
##  [6] "6264509100/6264509100_R01C02_Grn.idat"
##  [7] "6264509100/6264509100_R01C02_Red.idat"
##  [8] "6264509100/6264509100_R02C01_Grn.idat"
##  [9] "6264509100/6264509100_R02C01_Red.idat"
## [10] "6264509100/6264509100_R02C02_Grn.idat"
## [11] "6264509100/6264509100_R02C02_Red.idat"
## [12] "6264509100/6264509100_R03C01_Grn.idat"
## [13] "6264509100/6264509100_R03C01_Red.idat"
## [14] "6264509100/6264509100_R03C02_Grn.idat"
## [15] "6264509100/6264509100_R03C02_Red.idat"
## [16] "6264509100/6264509100_R04C01_Grn.idat"
## [17] "6264509100/6264509100_R04C01_Red.idat"
## [18] "6264509100/6264509100_R04C02_Grn.idat"
## [19] "6264509100/6264509100_R04C02_Red.idat"
## [20] "6264509100/6264509100_R05C01_Grn.idat"
## [21] "6264509100/6264509100_R05C01_Red.idat"
## [22] "6264509100/6264509100_R05C02_Grn.idat"
## [23] "6264509100/6264509100_R05C02_Red.idat"
## [24] "6264509100/6264509100_R06C01_Grn.idat"
## [25] "6264509100/6264509100_R06C01_Red.idat"
## [26] "6264509100/6264509100_R06C02_Grn.idat"
## [27] "6264509100/6264509100_R06C02_Red.idat"
## [28] "SampleSheet.csv"
## [29] "ageData.RData"
## [30] "human_c2_v5.rdata"
## [31] "model-based-cpg-islands-hg19-chr17.txt"
## [32] "wgEncodeRegDnaseClusteredV3chr17.bed"
```

To demonstrate the various aspects of analysing methylation data, we will be using a small, publicly available 450k methylation dataset (GSE49667 (http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE49667))(Zhang et al. 2013). The dataset contains 10 samples in total: there are 4 different sorted T-cell types (naive, rTreg, act_naive, act_rTreg, collected from 3 different individuals (M28, M29, M30). For details describing sample collection and preparation, see Zhang et al. (2013). An additional `birth` sample (individual VICS-72098-18-B) is included from another study (GSE51180 (http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE51180))(Cruickshank et al. 2013) to illustrate approaches for identifying and excluding poor quality samples.

There are several R Bioconductor packages available that have been developed for analysing methylation array data, including *minfi* (Aryee et al. 2014), *missMethyl* (Phipson, Maksimovic, and Oshlack 2016), *wateRmelon* (Pidsley et al. 2013), *methylumi* (Davis et al. 2015), *ChAMP* (Morris et al. 2014) and *charm* (Aryee et al. 2011). Some of the packages, such as *minfi* and *methylumi* include a framework for reading in the raw data from IDAT files and various specialised objects for storing and manipulating the data throughout the course of an analysis. Other packages provide specialised analysis methods for normalisation and statistical testing that rely on either *minfi* or *methylumi* objects. It is possible to convert between *minfi* and *methylumi* data types, however, this is not always trivial. Thus, it is advisable to consider the methods that you are interested in using and the data types that are most appropriate

before you begin your analysis. Another popular method for analysing methylation array data is *limma* (Ritchie et al. 2015), which was originally developed for gene expression microarray analysis. As *limma* operates on a matrix of values, it is easily applied to any data that can be converted to a `matrix` in R. For a complete list of Bioconductor packages for analysing DNA methylation data, one can search for "DNAMethylation" in BiocViews ([https://www.bioconductor.org/packages/release/BiocViews.html#___DNAMethylation (https://www.bioconductor.org/packages/release/BiocViews.html#___DNAMethylation)] (https://www.bioconductor.org/packages/release/BiocViews.html#___DNAMethylation (https://www.bioconductor.org/packages/release/BiocViews.html#___DNAMethylation))) on the Bioconductor website (https://www.bioconductor.org/).

We will begin with an example of a **probe-wise** differential methylation analysis using *minfi* and *limma*. By **probe-wise** analysis we mean each individual CpG probe will be tested for differential methylation for the comparisons of interest and p-values and moderated t-statistics (Smyth 2004) will be generated for each CpG probe.

2.2 Loading the data

It is useful to begin an analysis in R by loading all the packages that are likely to be required.

```
# load packages required for analysis
library(knitr)
library(limma)
library(minfi)
library(IlluminaHumanMethylation450kanno.ilmn12.hg19)
library(IlluminaHumanMethylation450kmanifest)
library(RColorBrewer)
library(missMethyl)
library(minfiData)
library(Gviz)
library(DMRcate)
library(stringr)
```

The *minfi*, *IlluminaHumanMethylation450kanno.ilmn12.hg19*, *IlluminaHumanMethylation450kmanifest*, *missMethyl*, *minfiData* and *DMRcate* are methylation specific packages, while *RColorBrewer* and *Gviz* are visualisation packages. We use *limma* for testing differential methylation, and *matrixStats* and *stringr* have functions used in the workflow. The *IlluminaHumanMethylation450kmanifest* package provides the Illumina manifest as an R object which can easily be loaded into the environment. The manifest contains all of the annotation information for each of the CpG probes on the 450k array. This is useful for determining where any differentially methylated probes are located in a genomic context.

```
# get the 450k annotation data
ann450k <- getAnnotation(IlluminaHumanMethylation450kanno.ilmn1
2.hg19)
head(ann450k)
```

```
## DataFrame with 6 rows and 33 columns
##                    chr       pos      strand        Name
AddressA
##            <character> <integer> <character> <character> <ch
aracter>
## cg00050873        chrY   9363356           -  cg00050873
32735311
## cg00212031        chrY  21239348           -  cg00212031
29674443
## cg00213748        chrY   8148233           -  cg00213748
30703409
## cg00214611        chrY  15815688           -  cg00214611
69792329
## cg00455876        chrY   9385539           -  cg00455876
27653438
## cg01707559        chrY   6778695           +  cg01707559
45652402
##               AddressB
ProbeSeqA
##            <character>
<character>
## cg00050873    31717405 ACAAAAAAACAACACACAACTATAATAATTTTTAAAA
TAAATAAACCCCA
## cg00212031    38703326 CCCAATTAACCACAAAAACTAAACAAATTATACAATC
AAAAAAACATACA
## cg00213748    36767301 TTTTAACACCTAACACCATTTTAACAATAAAAATTCT
ACAAAAAAAAACA
## cg00214611    46723459 CTAACTTCCAAACCACACTTTATATACTAAACTACAA
TATAACACAAACA
## cg00455876    69732350 AACTCTAAACTACCCAACACAAACTCCAAAAACTTCT
CAAAAAAAACTCA
## cg01707559    64689504 ACAAATTAAAAACACTAAAACAAACACAACAACTACA
ACAACAAAAAACA
##                                                       ProbeSeq
B        Type
##                                                     <character
> <character>
## cg00050873 ACGAAAAAACAACGCACAACTATAATAATTTTTAAAATAAATAAACCCC
G           I
## cg00212031 CCCAATTAACCGCAAAAACTAAACAAATTATACGATCGAAAAAACGTAC
G           I
## cg00213748 TTTTAACGCCTAACACCGTTTTAACGATAAAAATTCTACAAAAAAAAAC
G           I
## cg00214611 CTAACTTCCGAACCGCGCTTTATATACTAAACTACAATATAACGCGAAC
G           I
## cg00455876 AACTCTAAACTACCCGACACAAACTCCAAAAACTTCTCGAAAAAAACTC
G           I
## cg01707559 GCGAATTAAAAACACTAAAACGAACGCGACGACTACAACGACAAAAAAC
G           I
##               NextBase       Color    Probe_rs Probe_maf
CpG_rs   CpG_maf
##            <character> <character> <character> <numeric> <ch
aracter> <numeric>
## cg00050873           A         Red          NA        NA
NA        NA
## cg00212031           T         Red          NA        NA
NA        NA
## cg00213748           A         Red          NA        NA
NA        NA
## cg00214611           A         Red          NA        NA
```

```
NA        NA
## cg00455876           A         Red          NA        NA
NA        NA
## cg01707559           A         Red          NA        NA
NA        NA
##                 SBE_rs   SBE_maf           Islands_Name Rela
tion_to_Island
##            <character> <numeric>            <character>
<character>
## cg00050873          NA        NA   chrY:9363680-9363943
N_Shore
## cg00212031          NA        NA chrY:21238448-21240005
Island
## cg00213748          NA        NA   chrY:8147877-8148210
S_Shore
## cg00214611          NA        NA chrY:15815488-15815779
Island
## cg00455876          NA        NA   chrY:9385471-9385777
Island
## cg01707559          NA        NA   chrY:6778574-6780028
Island
##
Forward_Sequence
##
<character>
## cg00050873 TATCTCTGTCTGGCGAGGAGGCAACGCACAACTGTGGTGGTTTTTGGAG
TGGGTGGACCC[CG]GCCAAGACGGCCTGGGCTGACCAGAGACGGGAGGCAGAAAAAGTGGGC
AGGTGGTTGCAG
## cg00212031 CCATTGGCCCGCCCCAGTTGGCCGCAGGGACTGAGCAAGTTATGCGGTC
GGGAAGACGTG[CG]TTAAAGGGCTGAAGGGGAGGGACGGAACTGACAGTCTCTGTGACAGCT
CTGAGGTGGGAG
## cg00213748 TCTGTGGGACCATTTTAACGCCTGGCACCGTTTTAACGATGGAGGTTCT
GCAGGAGGGGG[CG]ACCTGGGGTAGGAGGCGTGCTAGTGGTGGATGACATTGTGGCAGAGAT
GGAGGTGGTGGC
## cg00214611 GCGCCGGCAGGACTAGCTTCCGGGCCGCGCTTTGTGTGCTGGGCTGCAG
TGTGGCGCGGG[CG]AGGAAGCTGGTAGGGCGGTTGTCGCAAGCTCCAGCTGCAGCCTCCGCC
TACGTGAGAAGA
## cg00455876 CGCGTGTGCCTGGACTCTGAGCTACCCGGCACAAGCTCCAAGGGCTTCT
CGGAGGAGGCT[CG]GGGACGGAAGGCGTGGGGTGAGTGGGCTGGAGATGCAGGCGCGCCCGT
GGCTGTGCAGCC
## cg01707559 AGCGGCCGCTCCCAGTGGTGGTCACCGCCAGTGCCAATCCCTTGCGCCG
CCGTGCAGTCC[CG]CCCTCTGTCGCTGCAGCCGCCGCGCCCGCTCCAGTGCCCCCAATTCGC
GCTCGGGAGTGA
##                                                    SourceSe
q Random_Loci
##                                                   <character
> <character>
## cg00050873 CGGGGTCCACCCACTCCAAAAACCACCACAGTTGTGCGTTGCCTCCTCG
C
## cg00212031 CGCACGTCTTCCCGACCGCATAACTTGCTCAGTCCCTGCGGCCAACTGG
G
## cg00213748 CGCCCCCTCCTGCAGAACCTCCATCGTTAAAACGGTGCCAGGCGTTAAA
A
## cg00214611 CGCCCGCGCCACACTGCAGCCCAGCACACAAAGCGCGGCCCGGAAGCTA
G
## cg00455876 GACTCTGAGCTACCCGGCACAAGCTCCAAGGGCTTCTCGGAGGAGGCTC
G
## cg01707559 CGCCCTCTGTCGCTGCAGCCGCCGCGCCCGCTCCAGTGCCCCCAATTCG
C
##            Methyl27_Loci UCSC_RefGene_Name        UCSC_RefGe
ne_Accession
```

```
##             <character>       <character>
<character>
## cg00050873                 TSPY4;FAM197Y2        NM_0011644
71;NR_001553
## cg00212031                        TTTY14
NR_001543
## cg00213748
## cg00214611                  TMSB4Y;TMSB4Y           NM_0042
02;NM_004202
## cg00455876
## cg01707559              TBL1Y;TBL1Y;TBL1Y NM_134259;NM_0332
84;NM_134258
##             UCSC_RefGene_Group     Phantom         DMR    E
nhancer
##                    <character> <character> <character> <cha
racter>
## cg00050873         Body;TSS1500
## cg00212031              TSS200
## cg00213748
## cg00214611        1stExon;5'UTR
## cg00455876
## cg01707559 TSS200;TSS200;TSS200
##                     HMM_Island Regulatory_Feature_Name
##                    <character>             <character>
## cg00050873   Y:9973136-9976273
## cg00212031 Y:19697854-19699393
## cg00213748   Y:8207555-8208234
## cg00214611 Y:14324883-14325218     Y:15815422-15815706
## cg00455876   Y:9993394-9995882
## cg01707559   Y:6838022-6839951
##                          Regulatory_Feature_Group         DH
S
##                                     <character> <character
>
## cg00050873
## cg00212031
## cg00213748
## cg00214611 Promoter_Associated_Cell_type_specific
## cg00455876
## cg01707559
```

As for their many other BeadArray platforms, Illumina methylation data is usually obtained in the form of Intensity Data (IDAT) Files. This is a proprietary format that is output by the scanner and stores summary intensities for each probe on the array. However, there are Bioconductor packages available that facilitate the import of data from IDAT files into R (Smith et al. 2013). Typically, each IDAT file is approximately 8MB in size. The simplest way to import the raw methylation data into R is using the *minfi* function `read.metharray.sheet`, along with the path to the IDAT files and a sample sheet. The sample sheet is a CSV (comma-separated) file containing one line per sample, with a number of columns describing each sample. The format expected by the `read.metharray.sheet` function is based on the sample sheet file that usually accompanies Illumina methylation array data. It is also very similar to the targets file described by the *limma* package. Importing the sample sheet into R creates a `data.frame` with one row for each sample and several columns.

The `read.metharray.sheet` function uses the specified path and other information from the sample sheet to create a column called `Basename` which specifies the location of each individual IDAT file in the experiment.

```
# read in the sample sheet for the experiment
targets <- read.metharray.sheet(dataDirectory, pattern="SampleS
heet.csv")
targets
```

Now that we have imported the information about the samples and where the data is located, we can read the raw intensity signals into R from the IDAT files using the `read.metharray.exp` function. This creates an `RGChannelSet` object that contains all the raw intensity data, from both the red and green colour channels, for each of the samples. At this stage, it can be useful to rename the samples with more descriptive names.

```
# read in the raw data from the IDAT files
rgSet <- read.metharray.exp(targets=targets)
rgSet
```

```
## class: RGChannelSet
## dim: 622399 11
## metadata(0):
## assays(2): Green Red
## rownames(622399): 10600313 10600322 ... 74810490 74810492
## rowData names(0):
## colnames(11): 6264509100_R01C01 6264509100_R02C01 ... 626450
9100_R04C02
##   5975827018_R06C02
## colData names(10): Sample_Name Sample_Well ... Basename file
names
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
```

```
# give the samples descriptive names
targets$ID <- paste(targets$Sample_Group,targets$Sample_Name,se
p=".")
sampleNames(rgSet) <- targets$ID
rgSet
```

```
## class: RGChannelSet
## dim: 622399 11
## metadata(0):
## assays(2): Green Red
## rownames(622399): 10600313 10600322 ... 74810490 74810492
## rowData names(0):
## colnames(11): naive.1 rTreg.2 ... act_rTreg.10 birth.11
## colData names(10): Sample_Name Sample_Well ... Basename file
names
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
```

2.3 Quality control

Once the data has been imported into R, we can evaluate its quality. Firstly, we need to calculate detection p-values. We can generate a detection p-value for every CpG in every sample, which is indicative of the quality of the signal. The method used by *minfi* to calculate detection p-values compares the total signal \((M + U)\) for each probe to the background signal level, which is estimated from the negative control probes. Very small p-values are indicative of a reliable signal whilst large p-values, for example >0.01, generally indicate a poor quality signal.

Plotting the mean detection p-value for each sample allows us to gauge the general quality of the samples in terms of the overall signal reliability (Figure 2). Samples that have many failed probes will have relatively large mean detection p-values.

```
# calculate the detection p-values
detP <- detectionP(rgSet)
head(detP)
```

```
##                  naive.1       rTreg.2  act_naive.3     naiv
e.4   act_naive.5
## cg00050873  0.000000e+00  0.000000e+00 0.000000e+00 0.00000e
+00  0.000000e+00
## cg00212031  0.000000e+00  0.000000e+00 0.000000e+00 0.00000e
+00  0.000000e+00
## cg00213748  2.139652e-88  4.213813e-31 1.181802e-12 1.29802e
-47  8.255482e-15
## cg00214611  0.000000e+00  0.000000e+00 0.000000e+00 0.00000e
+00  0.000000e+00
## cg00455876 1.400696e-234 9.349236e-111 4.272105e-90 0.00000e
+00 3.347145e-268
## cg01707559  0.000000e+00  0.000000e+00 0.000000e+00 0.00000e
+00  0.000000e+00
##              act_rTreg.6      naive.7       rTreg.8  act_nai
ve.9  act_rTreg.10
## cg00050873  0.000000e+00  0.00000e+00  0.000000e+00 0.000000
e+00  0.000000e+00
## cg00212031  0.000000e+00  0.00000e+00  0.000000e+00 0.000000
e+00  0.000000e+00
## cg00213748  2.592206e-23  1.16160e-28  1.469801e-05 1.543654
e-21  1.365951e-08
## cg00214611  0.000000e+00  0.00000e+00  0.000000e+00 0.000000
e+00  0.000000e+00
## cg00455876 4.690740e-308 1.08647e-219 5.362780e-178 0.000000
e+00 7.950724e-295
## cg01707559  0.000000e+00  0.00000e+00  0.000000e+00 0.000000
e+00  0.000000e+00
##                 birth.11
## cg00050873  0.000000e+00
## cg00212031 2.638199e-237
## cg00213748  6.735224e-01
## cg00214611  7.344451e-01
## cg00455876 4.403634e-174
## cg01707559  0.000000e+00
```

```
# examine mean detection p-values across all samples to identif
y any failed samples
pal <- brewer.pal(8,"Dark2")
par(mfrow=c(1,2))
barplot(colMeans(detP), col=pal[factor(targets$Sample_Group)],
las=2,
        cex.names=0.8, ylab="Mean detection p-values")
abline(h=0.05,col="red")
legend("topleft", legend=levels(factor(targets$Sample_Group)),
fill=pal,
       bg="white")

barplot(colMeans(detP), col=pal[factor(targets$Sample_Group)],
las=2,
        cex.names=0.8, ylim=c(0,0.002), ylab="Mean detection p-
values")
abline(h=0.05,col="red")
legend("topleft", legend=levels(factor(targets$Sample_Group)),
fill=pal,
       bg="white")
```

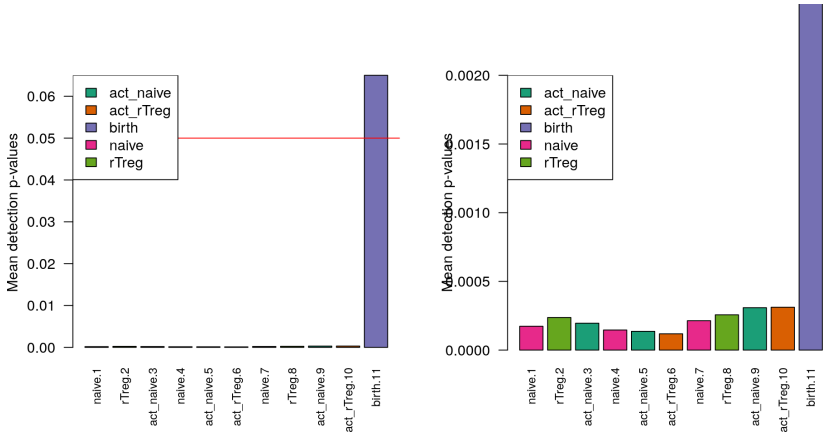

Figure 2: **Mean detection p-values summarise the quality of the signal across all the probes in each sample**
The plot on the right is a zoomed in version of the plot on the left.

The *minfi* `qcReport` function generates many other useful quality control plots. The *minfi* vignette (http://bioconductor.org/packages/release/bioc/vignettes/minfi/inst/doc/minfi.pdf) describes the various plots and how they should be interpreted in detail. Generally, samples that look poor based on mean detection p-value will also look poor using other metrics and it is usually advisable to exclude them from further analysis.

```
qcReport(rgSet, sampNames=targets$ID, sampGroups=targets$Sample
_Group,
         pdf="qcReport.pdf")
```

Poor quality samples can be easily excluded from the analysis using a detection p-value cutoff, for example >0.05. For this particular dataset, the `birth` sample shows a very high mean detection p-value, and hence it is excluded from subsequent analysis (Figure 2).

```
# remove poor quality samples
keep <- colMeans(detP) < 0.05
rgSet <- rgSet[,keep]
rgSet
```

```
## class: RGChannelSet
## dim: 622399 10
## metadata(0):
## assays(2): Green Red
## rownames(622399): 10600313 10600322 ... 74810490 74810492
## rowData names(0):
## colnames(10): naive.1 rTreg.2 ... act_naive.9 act_rTreg.10
## colData names(10): Sample_Name Sample_Well ... Basename file
names
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
```

```
# remove poor quality samples from targets data
targets <- targets[keep,]
targets[,1:5]
```

```
##    Sample_Name Sample_Well Sample_Source Sample_Group Sample
_Label
## 1            1          A1           M28        naive
naive
## 2            2          B1           M28        rTreg
rTreg
## 3            3          C1           M28    act_naive    act
_naive
## 4            4          D1           M29        naive
naive
## 5            5          E1           M29    act_naive    act
_naive
## 6            6          F1           M29    act_rTreg    act
_rTreg
## 7            7          G1           M30        naive
naive
## 8            8          H1           M30        rTreg
rTreg
## 9            9          A2           M30    act_naive    act
_naive
## 10          10          B2           M30    act_rTreg    act
_rTreg
```

```
# remove poor quality samples from detection p-value table
detP <- detP[,keep]
dim(detP)
```

```
## [1] 485512     10
```

2.4 Normalisation

To minimise the unwanted variation within and between samples, various data normalisations can be applied. Many different types of normalisation have been developed for methylation arrays and it is beyond the scope of

this workflow to compare and contrast all of them (Fortin et al. 2014; Wu et al. 2014; Sun et al. 2011; Wang et al. 2012; Maksimovic, Gordon, and Oshlack 2012; Mancuso et al. 2011; Touleimat and Tost 2012; Teschendorff et al. 2013; Pidsley et al. 2013; Triche et al. 2013). Several methods have been built into *minfi* and can be directly applied within its framework (Fortin et al. 2014; Triche et al. 2013; Maksimovic, Gordon, and Oshlack 2012; Touleimat and Tost 2012), whilst others are *methylumi*-specific or require custom data types (Wu et al. 2014; Sun et al. 2011; Wang et al. 2012; Mancuso et al. 2011; Teschendorff et al. 2013; Pidsley et al. 2013). Although there is no single normalisation method that is universally considered best, a recent study by Fortin et al. (2014) has suggested that a good rule of thumb within the *minfi* framework is that the `preprocessFunnorm` (Fortin et al. 2014) function is most appropriate for datasets with global methylation differences such as cancer/normal or vastly different tissue types, whilst the `preprocessQuantile` function (Touleimat and Tost 2012) is more suited for datasets where you do not expect global differences between your samples, for example a single tissue. Further discussion on appropriate choice of normalisation can be found in (Hicks and Irizarry 2015), and the accompanying *quantro* package includes data-driven tests for the assumptions of quantile normalisation. As we are comparing different blood cell types, which are globally relatively similar, we will apply the `preprocessQuantile` method to our data (Figure 3). This function implements a stratified quantile normalisation procedure which is applied to the methylated and unmethylated signal intensities separately, and takes into account the different probe types. Note that after normalisation, the data is housed in a `GenomicRatioSet` object. This is a much more compact representation of the data as the colour channel information has been discarded and the \(M\) and \(U\) intensity information has been converted to M-values and beta values, together with associated genomic coordinates. Note, running the `preprocessQuantile` function on this dataset produces the warning: **'An inconsistency was encountered while determining sex'** ; this can be ignored as it is due to all the samples being from male donors.

```
# normalize the data; this results in a GenomicRatioSet object
mSetSq <- preprocessQuantile(rgSet)


## [preprocessQuantile] Mapping to genome.


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


## [preprocessQuantile] Fixing outliers.


## [preprocessQuantile] Quantile normalizing.


# create a MethylSet object from the raw data for plotting
mSetRaw <- preprocessRaw(rgSet)
```

```
# visualise what the data looks like before and after normalisa
tion
par(mfrow=c(1,2))
densityPlot(rgSet, sampGroups=targets$Sample_Group,main="Raw",
legend=FALSE)
legend("top", legend = levels(factor(targets$Sample_Group)),
       text.col=brewer.pal(8,"Dark2"))
densityPlot(getBeta(mSetSq), sampGroups=targets$Sample_Group,
            main="Normalized", legend=FALSE)
legend("top", legend = levels(factor(targets$Sample_Group)),
       text.col=brewer.pal(8,"Dark2"))
```

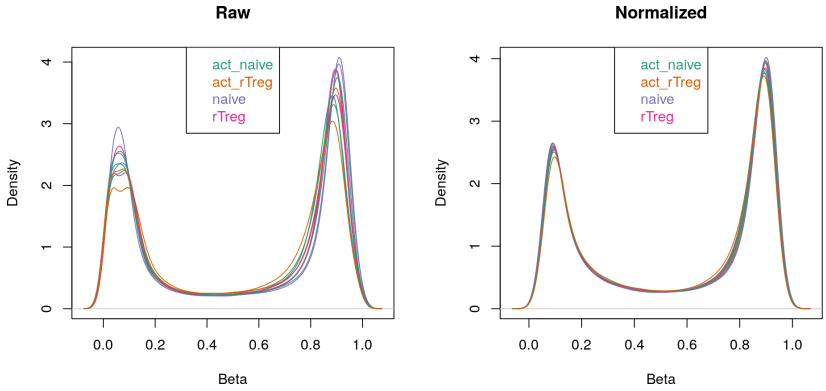

Figure 3: **The density plots show the distribution of the beta values for each sample before and after normalisation**

2.5 Data exploration

Multi-dimensional scaling (MDS) plots are excellent for visualising data, and are usually some of the first plots that should be made when exploring the data. MDS plots are based on principal components analysis and are an unsupervised method for looking at the similarities and differences between the various samples. Samples that are more similar to each other should cluster together, and samples that are very different should be further apart on the plot. Dimension one (or principal component one) captures the greatest source of variation in the data, dimension two captures the second greatest source of variation in the data and so on. Colouring the data points or labels by known factors of interest can often highlight exactly what the greatest sources of variation are in the data. It is also possible to use MDS plots to decipher sample mix-ups.

```
# MDS plots to look at largest sources of variation
par(mfrow=c(1,2))
plotMDS(getM(mSetSq), top=1000, gene.selection="common",
        col=pal[factor(targets$Sample_Group)])
legend("top", legend=levels(factor(targets$Sample_Group)), tex
t.col=pal,
       bg="white", cex=0.7)

plotMDS(getM(mSetSq), top=1000, gene.selection="common",
        col=pal[factor(targets$Sample_Source)])
legend("top", legend=levels(factor(targets$Sample_Source)), tex
t.col=pal,
       bg="white", cex=0.7)
```

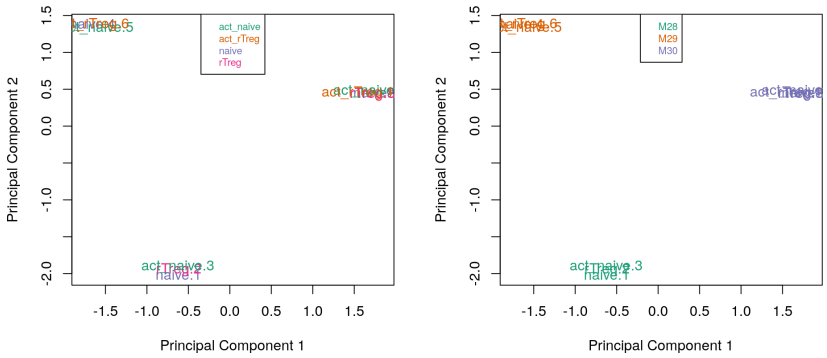

Figure 4: **Multi-dimensional scaling plots are a good way to visualise the relationships between the samples in an experiment**

Examining the MDS plots for this dataset demonstrates that the largest source of variation is the difference between individuals (Figure 4). The higher dimensions reveal that the differences between cell types are largely captured by the third and fourth principal components (Figure 5). This type of information is useful in that it can inform downstream analysis. If obvious sources of unwanted variation are revealed by the MDS plots, we can include them in our statistical model to account for them. In the case of this particular dataset, we will include individual to individual variation in our statistical model.

```
# Examine higher dimensions to look at other sources of variati
on
par(mfrow=c(1,3))
plotMDS(getM(mSetSq), top=1000, gene.selection="common",
        col=pal[factor(targets$Sample_Group)], dim=c(1,3))
legend("top", legend=levels(factor(targets$Sample_Group)), tex
t.col=pal,
       cex=0.7, bg="white")

plotMDS(getM(mSetSq), top=1000, gene.selection="common",
        col=pal[factor(targets$Sample_Group)], dim=c(2,3))
legend("topleft", legend=levels(factor(targets$Sample_Group)),
text.col=pal,
       cex=0.7, bg="white")

plotMDS(getM(mSetSq), top=1000, gene.selection="common",
        col=pal[factor(targets$Sample_Group)], dim=c(3,4))
legend("topright", legend=levels(factor(targets$Sample_Group)),
text.col=pal,
       cex=0.7, bg="white")
```

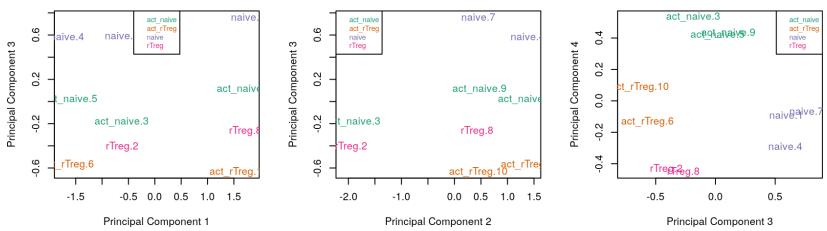

Figure 5: **Examining the higher dimensions of an MDS plot can reaveal significant sources of variation in the data**

2.6 Filtering

Poor performing probes are generally filtered out prior to differential methylation analysis. As the signal from these probes is unreliable, by removing them we perform fewer statistical tests and thus incur a reduced multiple testing penalty. We filter out probes that have failed in one or more samples based on detection p-value.

```
# ensure probes are in the same order in the mSetSq and detP ob
jects
detP <- detP[match(featureNames(mSetSq),rownames(detP)),]

# remove any probes that have failed in one or more samples
keep <- rowSums(detP < 0.01) == ncol(mSetSq)
table(keep)
```

```
## keep
##  FALSE   TRUE
##    977 484535
```

```
mSetSqFlt <- mSetSq[keep,]
mSetSqFlt
```

```
## class: GenomicRatioSet
## dim: 484535 10
## metadata(0):
## assays(2): M CN
## rownames(484535): cg13869341 cg14008030 ... cg08265308 cg142
73923
## rowData names(0):
## colnames(10): naive.1 rTreg.2 ... act_naive.9 act_rTreg.10
## colData names(13): Sample_Name Sample_Well ... yMed predicte
dSex
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
## Preprocessing
##   Method: Raw (no normalization or bg correction)
##   minfi version: 1.33.0
##   Manifest version: 0.4.0
```

Depending on the nature of your samples and your biological question you may also choose to filter out the probes from the X and Y chromosomes or probes that are known to have common SNPs at the CpG site. As the samples in this dataset were all derived from male donors, we will not be removing the sex chromosome probes as part of this analysis, however example code is provided below. A different dataset, which contains both male and female samples, is used to demonstrate a Differential Variability analysis and provides an example of when sex chromosome removal is necessary (Figure 13).

```
# if your data includes males and females, remove probes on the
sex chromosomes
keep <- !(featureNames(mSetSqFlt) %in% ann450k$Name[ann450k$chr
%in%
                                                    c("chr
X","chrY")])
table(keep)
mSetSqFlt <- mSetSqFlt[keep,]
```

There is a function in *minfi* that provides a simple interface for the removal of probes where common SNPs may affect the CpG. You can either remove all probes affected by SNPs (default), or only those with minor allele frequencies greater than a specified value.

```
# remove probes with SNPs at CpG site
mSetSqFlt <- dropLociWithSnps(mSetSqFlt)
mSetSqFlt
```

```
## class: GenomicRatioSet
## dim: 467351 10
## metadata(0):
## assays(2): M CN
## rownames(467351): cg13869341 cg14008030 ... cg08265308 cg142
73923
## rowData names(0):
## colnames(10): naive.1 rTreg.2 ... act_naive.9 act_rTreg.10
## colData names(13): Sample_Name Sample_Well ... yMed predicte
dSex
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
## Preprocessing
##   Method: Raw (no normalization or bg correction)
##   minfi version: 1.33.0
##   Manifest version: 0.4.0
```

We will also filter out probes that have shown to be cross-reactive, that is, probes that have been demonstrated to map to multiple places in the genome. This list was originally published by Chen et al. (2013) and can be obtained from the authors' website (http://www.sickkids.ca/MS-Office-Files/Research/Weksberg%20Lab/48639-non-specific-probes-Illumina450k.xlsx).

```
# exclude cross reactive probes
xReactiveProbes <- read.csv(file=paste(dataDirectory,
                                       "48639-non-specific-prob
es-Illumina450k.csv",
                                       sep="/"), stringsAsFacto
rs=FALSE)
keep <- !(featureNames(mSetSqFlt) %in% xReactiveProbes$TargetI
D)
table(keep)
```

```
## keep
##  FALSE   TRUE
##  27433 439918
```

```
mSetSqFlt <- mSetSqFlt[keep,]
mSetSqFlt
```

```
## class: GenomicRatioSet
## dim: 439918 10
## metadata(0):
## assays(2): M CN
## rownames(439918): cg13869341 cg24669183 ... cg08265308 cg142
73923
## rowData names(0):
## colnames(10): naive.1 rTreg.2 ... act_naive.9 act_rTreg.10
## colData names(13): Sample_Name Sample_Well ... yMed predicte
dSex
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
## Preprocessing
##   Method: Raw (no normalization or bg correction)
##   minfi version: 1.33.0
##   Manifest version: 0.4.0
```

Once the data has been filtered and normalised, it is often useful to re-examine the MDS plots to see if the relationship between the samples has changed. It is apparent from the new MDS plots that much of the inter-individual variation has been removed as this is no longer the first principal component (Figure 6), likely due to the removal of the SNP-affected CpG probes. However, the samples do still cluster by individual in the second dimension (Figure 6 and Figure 7) and thus a factor for individual should still be included in the model.

```
par(mfrow=c(1,2))
plotMDS(getM(mSetSqFlt), top=1000, gene.selection="common",
        col=pal[factor(targets$Sample_Group)], cex=0.8)
legend("right", legend=levels(factor(targets$Sample_Group)), te
xt.col=pal,
       cex=0.65, bg="white")

plotMDS(getM(mSetSqFlt), top=1000, gene.selection="common",
        col=pal[factor(targets$Sample_Source)])
legend("right", legend=levels(factor(targets$Sample_Source)), t
ext.col=pal,
       cex=0.7, bg="white")
```

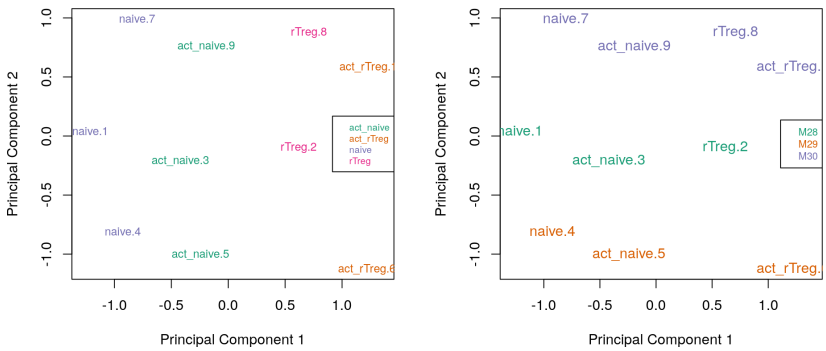

Figure 6: **Removing SNP-affected CpGs probes from the data changes the sample clustering in the MDS plots**

```
par(mfrow=c(1,3))
# Examine higher dimensions to look at other sources of variati
on
plotMDS(getM(mSetSqFlt), top=1000, gene.selection="common",
        col=pal[factor(targets$Sample_Source)], dim=c(1,3))
legend("right", legend=levels(factor(targets$Sample_Source)), t
ext.col=pal,
       cex=0.7, bg="white")

plotMDS(getM(mSetSqFlt), top=1000, gene.selection="common",
        col=pal[factor(targets$Sample_Source)], dim=c(2,3))
legend("topright", legend=levels(factor(targets$Sample_Sourc
e)), text.col=pal,
       cex=0.7, bg="white")

plotMDS(getM(mSetSqFlt), top=1000, gene.selection="common",
        col=pal[factor(targets$Sample_Source)], dim=c(3,4))
legend("right", legend=levels(factor(targets$Sample_Source)), t
ext.col=pal,
       cex=0.7, bg="white")
```

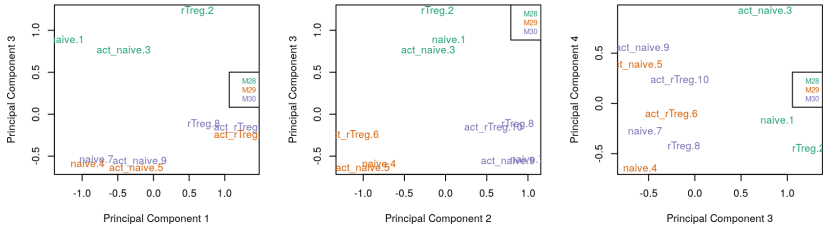

Figure 7: **Examining the higher dimensions of the MDS plots shows that significant inter-individual variation still exists in the second and third principal components**

The next step is to calculate M-values and beta values (Figure 8). As previously mentioned, M-values have nicer statistical properties and are thus better for use in statistical analysis of methylation data whilst beta values are easy to interpret and are thus better for displaying data. A detailed comparison of M-values and beta values was published by Du et al. (2010).

```
# calculate M-values for statistical analysis
mVals <- getM(mSetSqFlt)
head(mVals[,1:5])
```

```
##              naive.1   rTreg.2 act_naive.3   naive.4 act_nai
ve.5
## cg13869341  2.421276  2.515948    2.165745  2.286314    2.10
9441
## cg24669183  2.169414  2.235964    2.280734  1.632309    2.18
4435
## cg15560884  1.761176  1.577578    1.597503  1.777486    1.76
4999
## cg01014490 -3.504268 -3.825119   -5.384735 -4.537864   -4.29
6526
## cg17505339  3.082191  3.924931    4.163206  3.255373    3.65
4134
## cg11954957  1.546401  1.912204    1.727910  2.441267    1.61
8331
```

```
bVals <- getBeta(mSetSqFlt)
head(bVals[,1:5])
```

```
##               naive.1    rTreg.2 act_naive.3    naive.4 act_
naive.5
## cg13869341 0.84267937 0.85118462   0.8177504 0.82987650  0.8
1186174
## cg24669183 0.81812908 0.82489238   0.8293297 0.75610281  0.8
1967323
## cg15560884 0.77219626 0.74903910   0.7516263 0.77417882  0.7
7266205
## cg01014490 0.08098986 0.06590459   0.0233755 0.04127262  0.0
4842397
## cg17505339 0.89439216 0.93822870   0.9471357 0.90520570  0.9
2641305
## cg11954957 0.74495496 0.79008516   0.7681146 0.84450764  0.7
5431167
```

```
par(mfrow=c(1,2))
densityPlot(bVals, sampGroups=targets$Sample_Group, main="Beta
values",
            legend=FALSE, xlab="Beta values")
legend("top", legend = levels(factor(targets$Sample_Group)),
       text.col=brewer.pal(8,"Dark2"))
densityPlot(mVals, sampGroups=targets$Sample_Group, main="M-val
ues",
            legend=FALSE, xlab="M values")
legend("topleft", legend = levels(factor(targets$Sample_Grou
p)),
       text.col=brewer.pal(8,"Dark2"))
```

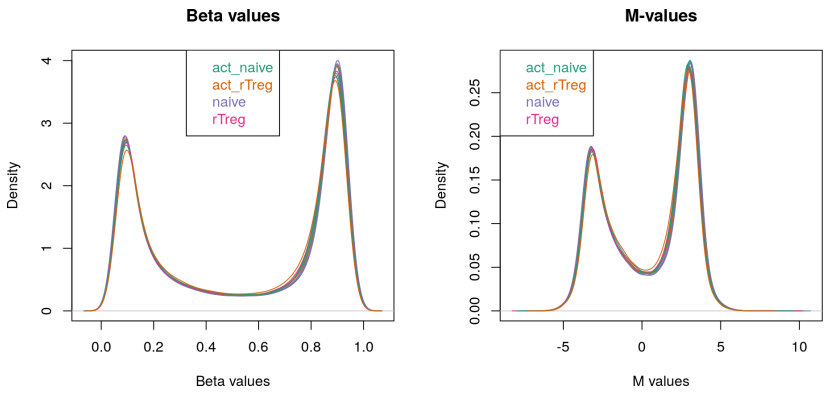

Figure 8: **The distributions of beta and M-values are quite different**
Beta values are constrained between 0 and 1 whilst M-values range between -Inf and Inf.

2.7 Probe-wise differential methylation analysis

The biological question of interest for this particular dataset is to discover differentially methylated probes between the different cell types. However, as was apparent in the MDS plots, there is another factor that we need to take into account when we perform the statistical analysis. In the `targets` file, there is a column called `Sample_Source`, which refers to the individuals that the samples were collected from. In this dataset, each of the individuals contributes more than one cell type. For example, individual M28 contributes `naive`, `rTreg` and `act_naive` samples. Hence, when we specify our design matrix, we need to include two

factors: individual and cell type. This style of analysis is called a paired analysis; differences between cell types are calculated *within* each individual, and then these differences are averaged *across* individuals to determine whether there is an overall significant difference in the mean methylation level for each CpG site. The *limma* User's Guide (https://bioconductor.org/packages/release/bioc/vignettes/limma/inst/doc/usersguide.pdf) extensively covers the different types of designs that are commonly used for microarray experiments and how to analyse them in R.

We are interested in pairwise comparisons between the four cell types, taking into account individual to individual variation. We perform this analysis on the matrix of M-values in *limma*, obtaining moderated t-statistics and associated p-values for each CpG site. A convenient way to set up the model when the user has many comparisons of interest that they would like to test is to use a contrasts matrix in conjunction with the design matrix. A contrasts matrix will take linear combinations of the columns of the design matrix corresponding to the comparisons of interest.

Since we are performing hundreds of thousands of hypothesis tests, we need to adjust the p-values for multiple testing. A common procedure for assessing how statistically significant a change in mean levels is between two groups when a very large number of tests is being performed is to assign a cut-off on the false discovery rate (Benjamini and Hochberg 1995), rather than on the unadjusted p-value. Typically 5% FDR is used, and this is interpreted as the researcher willing to accept that from the list of significant differentially methylated CpG sites, 5% will be false discoveries. If the p-values are not adjusted for multiple testing, the number of false discoveries will be unacceptably high. For this dataset, assuming a Type I error rate of 5%, we would expect to see 0.05*439918=21996 statistical significant results for a given comparison, even if there were truly no differentially methylated CpG sites.

Based on a false discovery rate of 5%, there are 3021 significantly differentially methylated CpGs in the `naïve` vs `rTreg` comparison, while `rTreg` vs `act_rTreg` doesn't show any significant differential methylation.

```
# this is the factor of interest
cellType <- factor(targets$Sample_Group)
# this is the individual effect that we need to account for
individual <- factor(targets$Sample_Source)

# use the above to create a design matrix
design <- model.matrix(~0+cellType+individual, data=targets)
colnames(design) <- c(levels(cellType),levels(individual)[-1])

# fit the linear model
fit <- lmFit(mVals, design)
# create a contrast matrix for specific comparisons
contMatrix <- makeContrasts(naive-rTreg,
                            naive-act_naive,
                            rTreg-act_rTreg,
                            act_naive-act_rTreg,
                            levels=design)
contMatrix
```

```
##            Contrasts
## Levels      naive - rTreg naive - act_naive rTreg - act_rTre
g
##   act_naive             0               -1
0
##   act_rTreg             0                0               -
1
##   naive                 1                1
0
##   rTreg                -1                0
1
##   M29                   0                0
0
##   M30                   0                0
0
##            Contrasts
## Levels      act_naive - act_rTreg
##   act_naive                     1
##   act_rTreg                    -1
##   naive                         0
##   rTreg                         0
##   M29                           0
##   M30                           0


# fit the contrasts
fit2 <- contrasts.fit(fit, contMatrix)
fit2 <- eBayes(fit2)

# look at the numbers of DM CpGs at FDR < 0.05
summary(decideTests(fit2))


##        naive - rTreg naive - act_naive rTreg - act_rTreg act
_naive - act_rTreg
## Down            1618               400                 0
559
## NotSig        436895            439291            439918
438440
## Up              1405               227                 0
919
```

We can extract the tables of differentially expressed CpGs for each comparison, ordered by B-statistic by default, using the `topTable` function in *limma*. The B-statistic is the log-odds of differential methylation, first published by Lonnstedt and Speed (Lonnstedt and Speed 2002). To order by p-value, the user can specify `sort.by="p"`; and in most cases, the ordering based on the p-value and ordering based on the B-statistic will be identical.The results of the analysis for the first comparison, `naive` vs. `rTreg`, can be saved as a `data.frame` by setting `coef=1`. The `coef` parameter explicitly refers to the column in the contrasts matrix which corresponds to the comparison of interest.

```
# get the table of results for the first contrast (naive - rTre
g)
ann450kSub <- ann450k[match(rownames(mVals),ann450k$Name),
                      c(1:4,12:19,24:ncol(ann450k))]
DMPs <- topTable(fit2, num=Inf, coef=1, genelist=ann450kSub)
head(DMPs)
```

```
##              chr       pos strand       Name Probe_rs Probe_
maf CpG_rs CpG_maf
## cg07499259  chr1  12188502      + cg07499259     <NA>
NA   <NA>      NA
## cg26992245  chr8  29848579      - cg26992245     <NA>
NA   <NA>      NA
## cg09747445 chr15  70387268      - cg09747445     <NA>
NA   <NA>      NA
## cg18808929  chr8  61825469      - cg18808929     <NA>
NA   <NA>      NA
## cg25015733  chr2  99342986      - cg25015733     <NA>
NA   <NA>      NA
## cg21179654  chr3 114057297      + cg21179654     <NA>
NA   <NA>      NA
##            SBE_rs SBE_maf            Islands_Name Relation_t
o_Island
## cg07499259   <NA>      NA
OpenSea
## cg26992245   <NA>      NA
OpenSea
## cg09747445   <NA>      NA chr15:70387929-70393206
N_Shore
## cg18808929   <NA>      NA  chr8:61822358-61823028
S_Shelf
## cg25015733   <NA>      NA  chr2:99346882-99348177
N_Shelf
## cg21179654   <NA>      NA
OpenSea
##                                        UCSC_RefGene_Name
## cg07499259                               TNFRSF8;TNFRSF8
## cg26992245
## cg09747445                                TLE3;TLE3;TLE3
## cg18808929
## cg25015733                                         MGAT4A
## cg21179654 ZBTB20;ZBTB20;ZBTB20;ZBTB20;ZBTB20;ZBTB20;ZBTB20
##
UCSC_RefGene_Accession
## cg07499259
NM_152942;NM_001243
## cg26992245
## cg09747445
NM_001105192;NM_020908;NM_005078
## cg18808929
## cg25015733
NM_012214
## cg21179654 NM_001164343;NM_001164346;NM_001164345;NM_0011643
42;NM_001164344;NM_001164347;NM_015642
##                                  UCSC_RefGene_Group Phantom
DMR Enhancer
## cg07499259                              5'UTR;Body
## cg26992245
TRUE
## cg09747445                           Body;Body;Body
## cg18808929
TRUE
## cg25015733                                  5'UTR
## cg21179654 3'UTR;3'UTR;3'UTR;3'UTR;3'UTR;3'UTR;3'UTR
##                     HMM_Island Regulatory_Feature_Name
## cg07499259 1:12111023-12111225
## cg26992245
```

```
## cg09747445
## cg18808929
## cg25015733
## cg21179654                        3:114057192-114057775
##                   Regulatory_Feature_Group DHS     logFC
AveExpr         t
## cg07499259                                       3.654104  2.
46652171  18.73082
## cg26992245                                       4.450696 -0.
09180715  18.32680
## cg09747445                                      -3.337299 -0.
25201484 -18.24369
## cg18808929                                      -2.990263  0.
77522878 -17.90079
## cg25015733                                      -3.054336  0.
83280190 -17.32537
## cg21179654 Unclassified_Cell_type_specific       2.859016  1.
32460816  17.27702
##                 P.Value   adj.P.Val        B
## cg07499259 7.258963e-08 0.005062817 7.454265
## cg26992245 8.603867e-08 0.005062817 7.360267
## cg09747445 8.913981e-08 0.005062817 7.340431
## cg18808929 1.033329e-07 0.005062817 7.256713
## cg25015733 1.332317e-07 0.005062817 7.109143
## cg21179654 1.361568e-07 0.005062817 7.096322
```

The resulting `data.frame` can easily be written to a CSV file, which can be opened in Excel.

```
write.table(DMPs, file="DMPs.csv", sep=",", row.names=FALSE)
```

It is always useful to plot sample-wise methylation levels for the top differentially methylated CpG sites to quickly ensure the results make sense (Figure 9). If the plots do not look as expected, it is usually an indication of an error in the code, or in setting up the design matrix. It is easier to interpret methylation levels on the beta value scale, so although the analysis is performed on the M-value scale, we visualise data on the beta value scale. The `plotCpg` function in *minfi* is a convenient way to plot the sample-wise beta values stratified by the grouping variable.

```
# plot the top 4 most significantly differentially methylated C
pGs
par(mfrow=c(2,2))
sapply(rownames(DMPs)[1:4], function(cpg){
  plotCpg(bVals, cpg=cpg, pheno=targets$Sample_Group, ylab = "B
eta values")
})
```

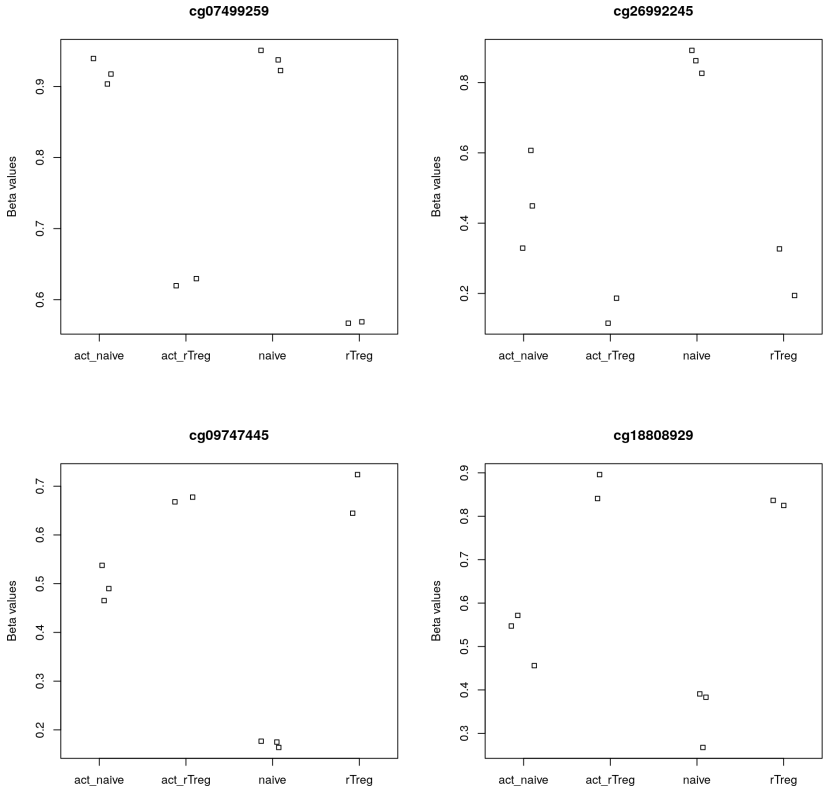

Figure 9: **Plotting the top few differentially methylated CpGs is a good way to check whether the results make sense**

```
## $cg07499259
## NULL
##
## $cg26992245
## NULL
##
## $cg09747445
## NULL
##
## $cg18808929
## NULL
```

2.8 Differential methylation analysis of regions

Although performing a *probe-wise* analysis is useful and informative, sometimes we are interested in knowing whether several proximal CpGs are concordantly differentially methylated, that is, we want to identify differentially methylated *regions*. There are several Bioconductor packages that have functions for identifying differentially methylated regions from 450k data. Some of the most popular are the `dmrFind` function in the charm (http://www.bioconductor.org/packages/release/bioc/html/charm.html) package, which has been somewhat superseded for 450k arrays by the `bumphunter` function in minfi (http://bioconductor.org/packages/release/bioc/html/minfi.html)(Jaffe et al. 2012; Aryee et al. 2014), and, the recently published `dmrcate` in the DMRcate (https://www.bioconductor.org/packages/release/bioc/html/DMRcate.html) package (Peters et al. 2015). They are each based on different statistical methods. In our experience, the `bumphunter` and `dmrFind` functions can

be somewhat slow to run unless you have the computer infrastructure to parallelise them, as they use permutations to assign significance. In this workflow, we will perform an analysis using the `dmrcate`. As it is based on *limma*, we can directly use the `design` and `contMatrix` we previously defined.

Firstly, our matrix of M-values is annotated with the relevant information about the probes such as their genomic position, gene annotation, etc. By default, this is done using the `ilmn12.hg19` annotation, but this can be substituted for any argument compatible with the interface provided by the *minfi* package. The *limma* pipeline is then used for differential methylation analysis to calculate moderated t-statistics.

```
myAnnotation <- cpg.annotate(object = mVals, datatype = "arra
y", what = "M",
                             analysis.type = "differential", de
sign = design,
                             contrasts = TRUE, cont.matrix = co
ntMatrix,
                             coef = "naive - rTreg", arraytype
= "450K")

## Your contrast returned 3023 individually significant probes.
We recommend the default setting of pcutoff in dmrcate().

str(myAnnotation)
```

```
## Formal class 'CpGannotated' [package "DMRcate"] with 1 slot
##   ..@ ranges:Formal class 'GRanges' [package "GenomicRange
s"] with 7 slots
##   .. .. ..@ seqnames       :Formal class 'Rle' [package "S4V
ectors"] with 4 slots
##   .. .. .. .. ..@ values         : Factor w/ 24 levels "chr
1","chr2",..: 1 2 3 4 5 6 7 8 9 10 ...
##   .. .. .. .. ..@ lengths        : int [1:24] 42733 31682 23
086 18431 22093 32652 26437 18956 8961 22163 ...
##   .. .. .. .. ..@ elementMetadata: NULL
##   .. .. .. .. ..@ metadata       : list()
##   .. .. ..@ ranges         :Formal class 'IRanges' [package
"IRanges"] with 6 slots
##   .. .. .. .. ..@ start          : int [1:439918] 15865 5342
42 710097 714177 720865 758829 763119 779995 805102 805338 ...
##   .. .. .. .. ..@ width          : int [1:439918] 1 1 1 1 1
1 1 1 1 1 ...
##   .. .. .. .. ..@ NAMES          : chr [1:439918] "cg1386934
1" "cg24669183" "cg15560884" "cg01014490" ...
##   .. .. .. .. ..@ elementType    : chr "ANY"
##   .. .. .. .. ..@ elementMetadata: NULL
##   .. .. .. .. ..@ metadata       : list()
##   .. .. ..@ strand         :Formal class 'Rle' [package "S4V
ectors"] with 4 slots
##   .. .. .. .. ..@ values         : Factor w/ 3 levels "+","-
","*": 3
##   .. .. .. .. ..@ lengths        : int 439918
##   .. .. .. .. ..@ elementMetadata: NULL
##   .. .. .. .. ..@ metadata       : list()
##   .. .. ..@ seqinfo        :Formal class 'Seqinfo' [package
"GenomeInfoDb"] with 4 slots
##   .. .. .. .. ..@ seqnames   : chr [1:24] "chr1" "chr2" "chr
3" "chr4" ...
##   .. .. .. .. ..@ seqlengths : int [1:24] NA NA NA NA NA NA
NA NA NA NA ...
##   .. .. .. .. ..@ is_circular: logi [1:24] NA NA NA NA NA NA
...
##   .. .. .. .. ..@ genome     : chr [1:24] NA NA NA NA ...
##   .. .. ..@ elementMetadata:Formal class 'DFrame' [package
"S4Vectors"] with 6 slots
##   .. .. .. .. ..@ rownames       : NULL
##   .. .. .. .. ..@ nrows          : int 439918
##   .. .. .. .. ..@ listData       :List of 4
##   .. .. .. .. .. ..$ stat   : num [1:439918] 0.0489 -2.0773
0.7711 -0.0304 -0.764 ...
##   .. .. .. .. .. ..$ diff   : num [1:439918] 0.00039 -0.0453
4 0.01594 0.00251 -0.00869 ...
##   .. .. .. .. .. ..$ ind.fdr: num [1:439918] 0.994 0.565 0.8
72 0.997 0.873 ...
##   .. .. .. .. .. ..$ is.sig : logi [1:439918] FALSE FALSE FA
LSE FALSE FALSE FALSE ...
##   .. .. .. .. ..@ elementType    : chr "ANY"
##   .. .. .. .. ..@ elementMetadata: NULL
##   .. .. .. .. ..@ metadata       : list()
##   .. .. ..@ elementType    : chr "ANY"
##   .. .. ..@ metadata       : list()
```

Once we have the relevant statistics for the individual CpGs, we can then use the `dmrcate` function to combine them to identify differentially methylated regions. The main output table `DMRs$results` contains all of

the regions found, along with their genomic annotations and p-values.

```
#endif /* NEWSTUFF */
DMRs <- dmrcate(myAnnotation, lambda=1000, C=2)
results.ranges <- extractRanges(DMRs)
results.ranges
```

```
## GRanges object with 545 ranges and 8 metadata columns:
##         seqnames              ranges strand |   no.cpgs min_
smoothed_fdr
##            <Rle>           <IRanges>  <Rle> | <integer>
<numeric>
##     [1]    chr17   57915665-57918682      * |        12
4.94393e-91
##     [2]     chr3 114012316-114012912      * |         5
1.63019e-180
##     [3]    chr18   21452730-21453131      * |         7
5.77246e-115
##     [4]    chr17   74639731-74640078      * |         6
9.62833e-90
##     [5]     chrX   49121205-49122718      * |         6
6.75742e-84
##     ...      ...                 ...    ... .       ...
...
##   [541]     chr2   43454761-43455103      * |        14
1.29083e-25
##   [542]     chr6   31832650-31833452      * |        18
2.46781e-28
##   [543]     chrX   43741310-43742501      * |         9
5.27008e-62
##   [544]     chr6 144385771-144387124      * |        22
2.85245e-60
##   [545]     chr2   25141532-25142229      * |         8
4.31468e-25
##            Stouffer      HMFDR      Fisher   maxdiff   meand
iff
##           <numeric>  <numeric>   <numeric> <numeric>  <numer
ic>
##     [1] 6.60666e-10 0.02351388 6.57544e-08  0.398286   0.313
161
##     [2] 1.51038e-07 0.00707524 1.39235e-06  0.543428   0.425
162
##     [3] 7.65545e-07 0.01239758 1.85082e-06 -0.386747  -0.254
609
##     [4] 1.52368e-07 0.01403323 2.60145e-06 -0.252864  -0.195
190
##     [5] 2.92694e-07 0.01163337 3.55872e-06  0.452909   0.300
624
##     ...         ...        ...         ...       ...
...
##   [541]    0.967707  0.1532340    0.620701 -0.218836 -0.0427
390
##   [542]    0.886282  0.2310998    0.647328  0.153367  0.0490
080
##   [543]    0.914407  0.0431746    0.655714  0.413832  0.0558
174
##   [544]    0.996631  0.0796880    0.690460  0.325422  0.0449
451
##   [545]    0.992418  0.0567769    0.748698  0.282058  0.0314
244
##
overlapping.genes
##
<character>
##     [1]
SNORA69, VMP1, MIR21
##     [2]                                          TIGIT, SN
```

```
ORA33, SNORA81, SNORD66, SNORD2, SNORD5, SNORD63, SNORD61, SNOR
A24, Metazoa_SRP, SNORA18, U4
##     [3]
LAMA3, SNORD23
##     [4]
SNORA69, ST6GALNAC1
##     [5]
FOXP3
##     ...
...
##   [541] SNORA73, SNORA64, SNORD75, SNORA74, snR65, 5S_rRNA,
SCARNA6, SNORD39, SNORD18, SNORA36, THADA, SNORA75, SNORA48, SN
ORD56, SNORA43, SNORA1, Vault
##   [542]
SNORA38, SLC44A4, SNORA20
##   [543]
MAOB
##   [544]
SNORD28, SNORA20
##   [545]                                    SNORA73, SNORA64,
SNORA74, snR65, SCARNA6, SNORD39, SNORD18, ADCY3, SNORA75, SNOR
A48, SNORD56, SNORA43, SNORA1
##   -------
##   seqinfo: 23 sequences from an unspecified genome; no seqle
ngths
```

As for the probe-wise analysis, it is advisable to visualise the results to ensure that they make sense. The regions can easily be viewed using the `DMR.plot` function provided in the *DMRcate* package (Figure 10).

```
# set up the grouping variables and colours
groups <- pal[1:length(unique(targets$Sample_Group))]
names(groups) <- levels(factor(targets$Sample_Group))
cols <- groups[as.character(factor(targets$Sample_Group))]


# draw the plot for the top DMR
par(mfrow=c(1,1))
DMR.plot(ranges = results.ranges, dmr = 2, CpGs = bVals, phen.c
ol = cols,
         what = "Beta", arraytype = "450K", genome = "hg19")


## snapshotDate(): 2019-12-17


## see ?DMRcatedata and browseVignettes('DMRcatedata') for docu
mentation


## loading from cache
```

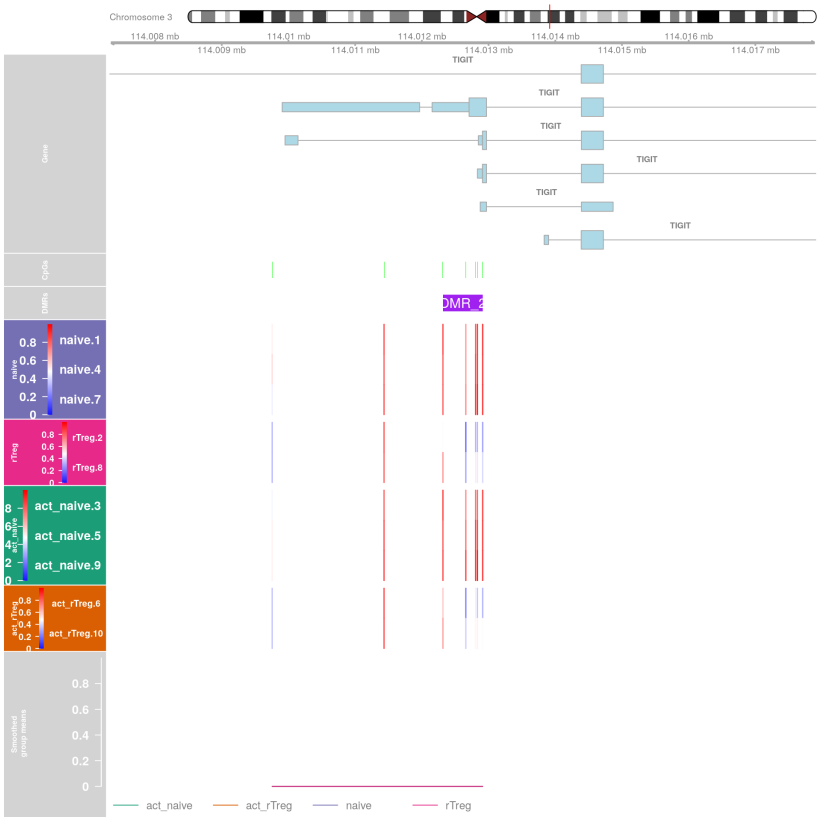

Figure 10: **DMRcate allows you to quickly visualise DMRs in their genomic context**
By default, the plot shows the location of the DMR in the genome, the position of any genes that are nearby, the base pair positions of the CpG probes, the methylation levels of the individual samples as a heatmap and the mean methylation levels for the various sample groups in the experiment. This plot shows one of the DMRs identified by the DMRcate analysis.

2.9 Customising visualisations of methylation data

The *Gviz* package offers powerful functionality for plotting methylation data in its genomic context. The package vignette (https://bioconductor.org/packages/release/bioc/vignettes/Gviz/inst/doc/Gviz.pdf) is very extensive and covers the various types of plots that can be produced using the *Gviz* framework. We will plot one of the differentially methylated regions from the *DMRcate* analysis to demonstrate the type of visualisations that can be created (Figure 11).

We will first set up the genomic region we would like to plot by extracting the genomic coordinates of one of the differentially methylated regions.

```
# indicate which genome is being used
gen <- "hg19"
# the index of the DMR that we will plot
dmrIndex <- 1
# extract chromosome number and location from DMR results
chrom <- as.character(seqnames(results.ranges[dmrIndex]))
start <- as.numeric(start(results.ranges[dmrIndex]))
end <- as.numeric(end(results.ranges[dmrIndex]))
# add 25% extra space to plot
minbase <- start - (0.25*(end-start))
maxbase <- end + (0.25*(end-start))
```

Next, we will add some genomic annotations of interest such as the locations of CpG islands and DNAseI hypersensitive sites; this can be any feature or genomic annotation of interest that you have data available for. The CpG islands data was generated using the method published by Wu et al. (2010); the DNaseI hypersensitive site data was obtained from the UCSC Genome Browser (https://genome.ucsc.edu/cgi-bin/hgTables).

```
# CpG islands
islandHMM <- read.csv(paste0(dataDirectory,
                             "/model-based-cpg-islands-hg19-chr
17.txt"),
                      sep="\t", stringsAsFactors=FALSE, header=
FALSE)
head(islandHMM)
```

```
##                V1     V2     V3   V4  V5   V6    V7    V8
## 1 chr17_ctg5_hap1   8935  10075 1141 129  815 0.714 0.887
## 2 chr17_ctg5_hap1  64252  64478  227  30  165 0.727 1.014
## 3 chr17_ctg5_hap1  87730  89480 1751 135 1194 0.682 0.663
## 4 chr17_ctg5_hap1  98265  98591  327  29  226 0.691 0.744
## 5 chr17_ctg5_hap1 120763 125451 4689 359 3032 0.647 0.733
## 6 chr17_ctg5_hap1 146257 146607  351  19  231 0.658 0.500
```

```
islandData <- GRanges(seqnames=Rle(islandHMM[,1]),
                      ranges=IRanges(start=islandHMM[,2], end=i
slandHMM[,3]),
                      strand=Rle(strand(rep("*",nrow(islandHM
M)))))
islandData
```

```
## GRanges object with 3456 ranges and 0 metadata columns:
##                  seqnames            ranges strand
##                     <Rle>         <IRanges>  <Rle>
##      [1] chr17_ctg5_hap1        8935-10075      *
##      [2] chr17_ctg5_hap1       64252-64478      *
##      [3] chr17_ctg5_hap1       87730-89480      *
##      [4] chr17_ctg5_hap1       98265-98591      *
##      [5] chr17_ctg5_hap1     120763-125451      *
##      ...             ...               ...    ...
##   [3452]           chr17 81147380-81147511      *
##   [3453]           chr17 81147844-81148321      *
##   [3454]           chr17 81152612-81153665      *
##   [3455]           chr17 81156194-81156512      *
##   [3456]           chr17 81162945-81165532      *
##   -------
##   seqinfo: 5 sequences from an unspecified genome; no seqlen
gths
```

```
# DNAseI hypersensitive sites
dnase <- read.csv(paste0(dataDirectory,"/wgEncodeRegDnaseCluste
redV3chr17.bed"),
                  sep="\t",stringsAsFactors=FALSE,header=FALSE)
head(dnase)
```

```
##      V1   V2   V3 V4  V5 V6
V7
## 1 chr17  125  335  7 444  7                    84,83,88,90,7
7,87,89,
## 2 chr17  685  835  1 150  1
80,
## 3 chr17 2440 2675 13 410 13 0,30,102,104,38,47,61,68,122,1,5
1,73,75,
## 4 chr17 3020 3170  1 247  1
120,
## 5 chr17 3740 3890  2 161  2
71,73,
## 6 chr17 5520 6110  4 241  5                         17,19,2
5,16,16,
##                                              V8
## 1                  328,208,444,218,109,171,191,
## 2                                          150,
## 3 204,410,301,206,46,48,84,164,85,12,98,215,146,
## 4                                          247,
## 5                                     108,161,
## 6                             241,185,239,26,52,
```

```
dnaseData <- GRanges(seqnames=dnase[,1],
                     ranges=IRanges(start=dnase[,2], end=dnase
[,3]),
                     strand=Rle(rep("*",nrow(dnase))),
                     data=dnase[,5])
dnaseData
```

```
## GRanges object with 74282 ranges and 1 metadata column:
##           seqnames            ranges strand |      data
##              <Rle>         <IRanges>  <Rle> | <integer>
##       [1]    chr17           125-335      * |       444
##       [2]    chr17           685-835      * |       150
##       [3]    chr17          2440-2675      * |       410
##       [4]    chr17          3020-3170      * |       247
##       [5]    chr17          3740-3890      * |       161
##       ...      ...                 ...    ... .       ...
##   [74278]    chr17 81153140-81153350      * |       574
##   [74279]    chr17 81153580-81153810      * |       208
##   [74280]    chr17 81185540-81185750      * |       326
##   [74281]    chr17 81188880-81189090      * |       209
##   [74282]    chr17 81194900-81195115      * |       185
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqleng
ths
```

Now, set up the ideogram, genome and RefSeq tracks that will provide context for our methylation data.

```
iTrack <- IdeogramTrack(genome = gen, chromosome = chrom, name
="")
gTrack <- GenomeAxisTrack(col="black", cex=1, name="", fontcolo
r="black")
rTrack <- UcscTrack(genome=gen, chromosome=chrom, track="NCBI R
efSeq",
                    from=minbase, to=maxbase, trackType="GeneRe
gionTrack",
                    rstarts="exonStarts", rends="exonEnds", gen
e="name",
                    symbol="name2", transcript="name", strand
="strand",
                    fill="darkblue",stacking="squish", name="Re
fSeq",
                    showId=TRUE, geneSymbol=TRUE)
```

Ensure that the methylation data is ordered by chromosome and base position.

```
ann450kOrd <- ann450kSub[order(ann450kSub$chr,ann450kSub$pos),]
head(ann450kOrd)
```

```
## DataFrame with 6 rows and 22 columns
##                    chr       pos      strand        Name
Probe_rs Probe_maf
##            <character> <integer> <character> <character> <ch
aracter> <numeric>
## cg13869341        chr1     15865           +  cg13869341
NA        NA
## cg24669183        chr1    534242           -  cg24669183   r
s6680725  0.108100
## cg15560884        chr1    710097           +  cg15560884
NA        NA
## cg01014490        chr1    714177           -  cg01014490
NA        NA
## cg17505339        chr1    720865           -  cg17505339
NA        NA
## cg11954957        chr1    758829           +  cg11954957 rs1
15498424  0.029514
##                 CpG_rs   CpG_maf      SBE_rs   SBE_maf
Islands_Name
##            <character> <numeric> <character> <numeric>
<character>
## cg13869341          NA        NA          NA        NA
## cg24669183          NA        NA          NA        NA chr1:
533219-534114
## cg15560884          NA        NA          NA        NA chr1:
713984-714547
## cg01014490          NA        NA          NA        NA chr1:
713984-714547
## cg17505339          NA        NA          NA        NA
## cg11954957          NA        NA          NA        NA chr1:
762416-763445
##            Relation_to_Island UCSC_RefGene_Name UCSC_RefGene
_Accession
##                   <character>       <character>            <
character>
## cg13869341            OpenSea            WASH5P
NR_024540
## cg24669183            S_Shore
## cg15560884            N_Shelf
## cg01014490             Island
## cg17505339            OpenSea
## cg11954957            N_Shelf
##            UCSC_RefGene_Group     Phantom         DMR    Enh
ancer
##                   <character> <character> <character> <chara
cter>
## cg13869341              Body
## cg24669183
## cg15560884
## cg01014490
## cg17505339
## cg11954957
##                 HMM_Island Regulatory_Feature_Name Regulator
y_Feature_Group
##                <character>            <character>
<character>
## cg13869341
## cg24669183 1:523025-524193
## cg15560884
## cg01014490 1:703784-704410         1:713802-715219      Prom
```

```
oter_Associated
## cg17505339
## cg11954957
##                     DHS
##             <character>
## cg13869341
## cg24669183
## cg15560884
## cg01014490
## cg17505339
## cg11954957
```

```
bValsOrd <- bVals[match(ann450kOrd$Name,rownames(bVals)),]
head(bValsOrd)
```

```
##               naive.1    rTreg.2 act_naive.3    naive.4 act_
naive.5 act_rTreg.6
## cg13869341 0.84267937 0.85118462   0.8177504 0.82987650  0.8
1186174   0.8090798
## cg24669183 0.81812908 0.82489238   0.8293297 0.75610281  0.8
1967323   0.8187838
## cg15560884 0.77219626 0.74903910   0.7516263 0.77417882  0.7
7266205   0.7721528
## cg01014490 0.08098986 0.06590459   0.0233755 0.04127262  0.0
4842397   0.0644404
## cg17505339 0.89439216 0.93822870   0.9471357 0.90520570  0.9
2641305   0.9286016
## cg11954957 0.74495496 0.79008516   0.7681146 0.84450764  0.7
5431167   0.8116911
##               naive.7    rTreg.8 act_naive.9 act_rTreg.10
## cg13869341 0.8891851 0.88537940  0.90916748   0.88334231
## cg24669183 0.7903763 0.85304116  0.80930568   0.80979554
## cg15560884 0.7658623 0.75909061  0.78099397   0.78569274
## cg01014490 0.0245281 0.02832358  0.07740468   0.04640659
## cg17505339 0.8889361 0.87205348  0.90099782   0.93508348
## cg11954957 0.7832207 0.84929777  0.84719430   0.83350220
```

Create the data tracks using the appropriate track type for each data type.

```
# create genomic ranges object from methylation data
cpgData <- GRanges(seqnames=Rle(ann450kOrd$chr),
                   ranges=IRanges(start=ann450kOrd$pos, end=ann
450kOrd$pos),
                   strand=Rle(rep("*",nrow(ann450kOrd))),
                   betas=bValsOrd)
# extract data on CpGs in DMR
cpgData <- subsetByOverlaps(cpgData, results.ranges[dmrIndex])

# methylation data track
methTrack <- DataTrack(range=cpgData, groups=targets$Sample_Gro
up,genome = gen,
                       chromosome=chrom, ylim=c(-0.05,1.05), co
l=pal,
                       type=c("a","p"), name="DNA Meth.\n(beta
value)",
                       background.panel="white", legend=TRUE, c
ex.title=0.8,
                       cex.axis=0.8, cex.legend=0.8)
# CpG island track
islandTrack <- AnnotationTrack(range=islandData, genome=gen, na
me="CpG Is.",
                               chromosome=chrom,fill="darkgree
n")

# DNaseI hypersensitive site data track
dnaseTrack <- DataTrack(range=dnaseData, genome=gen, name="DNAs
eI",
                        type="gradient", chromosome=chrom)

# DMR position data track
dmrTrack <- AnnotationTrack(start=start, end=end, genome=gen, n
ame="DMR",
                            chromosome=chrom,fill="darkred")
```

Set up the track list and indicate the relative sizes of the different tracks. Finally, draw the plot using the `plotTracks` function (Figure 11).

```
tracks <- list(iTrack, gTrack, methTrack, dmrTrack, islandTrac
k, dnaseTrack,
               rTrack)
sizes <- c(2,2,5,2,2,2,3) # set up the relative sizes of the tr
acks
plotTracks(tracks, from=minbase, to=maxbase, showTitle=TRUE, ad
d53=TRUE,
           add35=TRUE, grid=TRUE, lty.grid=3, sizes = sizes, le
ngth(tracks))
```

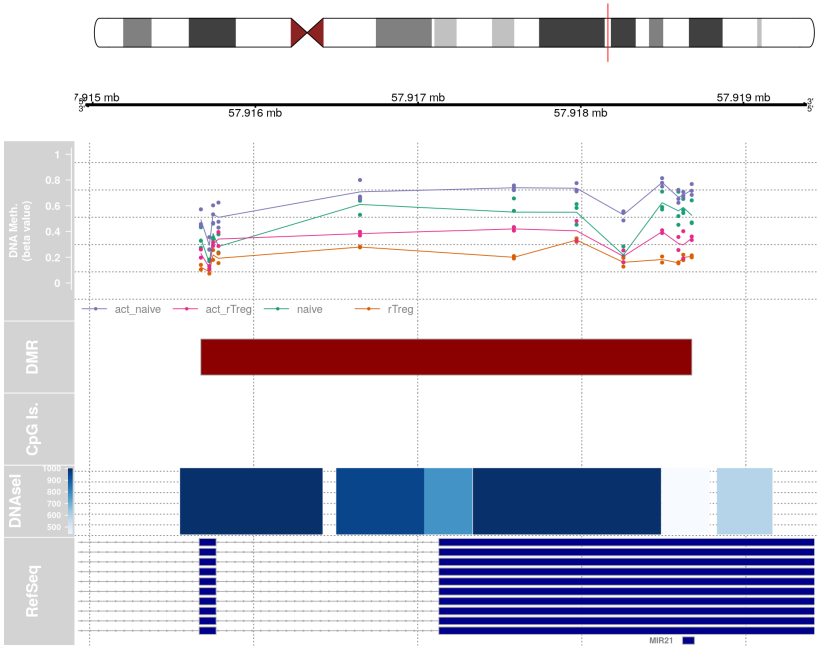

Figure 11: **The Gviz package provides extensive functionality for customising plots of genomic regions**
This plot shows one of the DMRs identified by the DMRcate analysis.

3 Additional analyses

3.1 Gene ontology testing

Once you have performed a differential methylation analysis, there may be a very long list of significant CpG sites to interpret. One question a researcher may have is, "which gene pathways are over-represented for differentially methylated CpGs?" In some cases it is relatively straightforward to link the top differentially methylated CpGs to genes that make biological sense in terms of the cell types or samples being studied, but there may be many thousands of CpGs significantly differentially methylated. In order to gain an understanding of the biological processes that the differentially methylated CpGs may be involved in, we can perform gene ontology or KEGG pathway analysis using the `gometh` function in the *missMethyl* package (Phipson, Maksimovic, and Oshlack 2016).

Let us consider the first comparison, naive vs rTreg, with the results of the analysis in the `DMPs` table. The `gometh` function takes as input a character vector of the names (e.g. cg20832020) of the significant CpG sites, and optionally, a character vector of all CpGs tested. This is recommended particularly if extensive filtering of the CpGs has been performed prior to analysis. For gene ontology testing (default), the user can specify `collection="GO"`. For testing KEGG pathways, specify `collection="KEGG"`. In the `DMPs` table, the `Name` column corresponds to the CpG name. We will select all CpG sites that have adjusted p-value of less than 0.05.

```
# Get the significant CpG sites at less than 5% FDR
sigCpGs <- DMPs$Name[DMPs$adj.P.Val<0.05]
# First 10 significant CpGs
sigCpGs[1:10]
```

```
##  [1] "cg07499259" "cg26992245" "cg09747445" "cg18808929" "cg
25015733"
##  [6] "cg21179654" "cg26280976" "cg16943019" "cg10898310" "cg
25130381"


# Total number of significant CpGs at 5% FDR
length(sigCpGs)


## [1] 3023


# Get all the CpG sites used in the analysis to form the backgr
ound
all <- DMPs$Name
# Total number of CpG sites tested
length(all)


## [1] 439918
```

The `gometh` function takes into account the varying numbers of CpGs associated with each gene on the Illumina methylation arrays. For the 450k array, the numbers of CpGs mapping to genes can vary from as few as 1 to as many as 1200. The genes that have more CpGs associated with them will have a higher probability of being identified as differentially methylated compared to genes with fewer CpGs. We can look at this bias in the data by specifying `plot=TRUE` in the call to `gometh` (Figure 12).

```
par(mfrow=c(1,1))
gst <- gometh(sig.cpg=sigCpGs, all.cpg=all, plot.bias=TRUE)
```

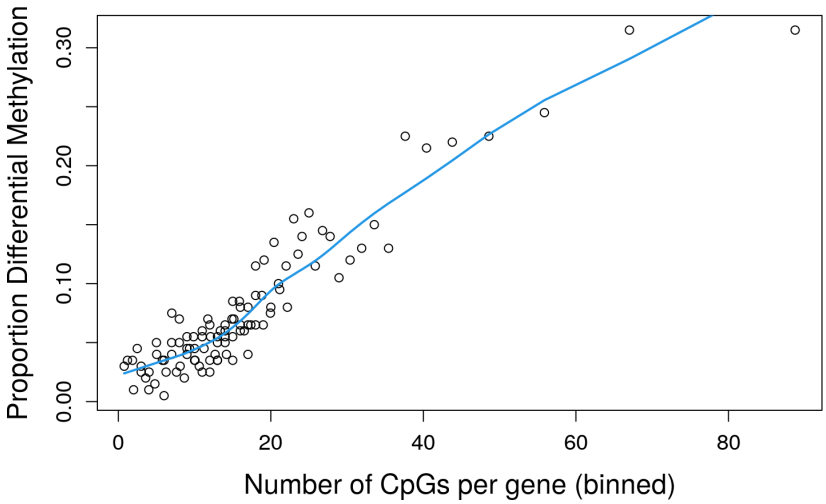

Figure 12: **Bias resulting from different numbers of CpG probes in different genes**

The `gst` object is a `data.frame` with each row corresponding to the GO category being tested. Note that the warning regarding multiple symbols will always be displayed as there are genes that have more than one alias, however it is not a cause for concern.

The top 20 gene ontology categories can be displayed using the `topGSA` function. For KEGG pathway analysis, the `topGSA` function will also display the top 20 enriched pathways.

```
# Top 10 GO categories
topGSA(gst, number=10)
```

```
##            ONTOLOGY
## GO:0006954       BP
## GO:0009897       CC
## GO:0005515       MF
## GO:0043123       BP
## GO:0042110       BP
## GO:0007165       BP
## GO:0032088       BP
## GO:0050853       BP
## GO:0006955       BP
## GO:0007166       BP
##
TERM    N
## GO:0006954                                              inflamma
tory response  366
## GO:0009897                                  external side of pl
asma membrane  364
## GO:0005515                                                  pr
otein binding 9690
## GO:0043123     positive regulation of I-kappaB kinase/NF-kap
paB signaling  179
## GO:0042110                                                T ce
ll activation   42
## GO:0007165                                              signal
transduction  973
## GO:0032088 negative regulation of NF-kappaB transcription fa
ctor activity   84
## GO:0050853                                 B cell receptor sign
aling pathway  103
## GO:0006955                                                  im
mune response  420
## GO:0007166                           cell surface receptor sign
aling pathway  206
##                DE          P.DE           FDR
## GO:0006954  58.50 7.903340e-09 8.598932e-05
## GO:0009897  54.00 9.485336e-09 8.598932e-05
## GO:0005515 934.25 1.147985e-07 6.938041e-04
## GO:0043123  35.00 1.736824e-06 7.872589e-03
## GO:0042110  15.00 2.457257e-06 8.910506e-03
## GO:0007165 130.50 5.142009e-06 1.553829e-02
## GO:0032088  20.50 9.255092e-06 2.397201e-02
## GO:0050853  14.00 1.747414e-05 3.960295e-02
## GO:0006955  43.50 2.087561e-05 4.205508e-02
## GO:0007166  34.00 3.413760e-05 6.060344e-02
```

From the output we can see many of the top GO categories correspond to immune system and T cell processes, which is unsurprising as the cell types being studied form part of the immune system. Typically, we consider GO categories that have associated false discovery rates of less than 5% to be statistically significant. If there aren't any categories that achieve this significance it may be useful to scan the top 5 or 10 highly ranked GO categories to gain some insight into the biological system.

The `gometh` function only tests GO and KEGG pathways. For a more generalised version of gene set testing for methylation data where the user can specify the gene set to be tested, the `gsameth` function can be

used. To display the top 20 pathways, `topGSA` can be called. `gsameth` accepts a single gene set, or a list of gene sets. The gene identifiers in the gene set must be Entrez Gene IDs. To demonstrate `gsameth`, we are using the curated genesets (C2) from the Broad Institute Molecular signatures database (http://software.broadinstitute.org/gsea/msigdb). These can be downloaded as an `RData` object from the WEHI Bioinformatics website (http://bioinf.wehi.edu.au/software/MSigDB/).

```
# load Broad human curated (C2) gene sets
load(paste(dataDirectory,"human_c2_v5.rdata",sep="/"))
# perform the gene set test(s)
gsa <- gsameth(sig.cpg=sigCpGs, all.cpg=all, collection=Hs.c2)

# top 10 gene sets
topGSA(gsa, number=10)
```

```
##                                                N        DE
P.DE
## ZHENG_BOUND_BY_FOXP3                         491 136.50000
3.091746e-28
## JAATINEN_HEMATOPOIETIC_STEM_CELL_DN          226  57.83333
1.637910e-15
## MARTENS_BOUND_BY_PML_RARA_FUSION             456 104.50000
7.963170e-14
## SMID_BREAST_CANCER_NORMAL_LIKE_UP            476  90.00000
4.102500e-13
## PILON_KLF1_TARGETS_DN                       1972 258.00000
1.325550e-12
## MARSON_BOUND_BY_FOXP3_UNSTIMULATED          1229 165.00000
3.458846e-11
## LEE_EARLY_T_LYMPHOCYTE_DN                     57  25.00000
5.662466e-11
## ZHENG_FOXP3_TARGETS_IN_THYMUS_UP             196  51.00000
1.648924e-10
## DEURIG_T_CELL_PROLYMPHOCYTIC_LEUKEMIA_DN     320  62.50000
4.458658e-10
## BOSCO_ALLERGEN_INDUCED_TH2_ASSOCIATED_MODULE 151  39.50000
1.466495e-09
##                                                       FDR
## ZHENG_BOUND_BY_FOXP3                         1.460850e-24
## JAATINEN_HEMATOPOIETIC_STEM_CELL_DN          3.869561e-12
## MARTENS_BOUND_BY_PML_RARA_FUSION             1.254199e-10
## SMID_BREAST_CANCER_NORMAL_LIKE_UP            4.846078e-10
## PILON_KLF1_TARGETS_DN                        1.252645e-09
## MARSON_BOUND_BY_FOXP3_UNSTIMULATED           2.723841e-08
## LEE_EARLY_T_LYMPHOCYTE_DN                    3.822165e-08
## ZHENG_FOXP3_TARGETS_IN_THYMUS_UP             9.738958e-08
## DEURIG_T_CELL_PROLYMPHOCYTIC_LEUKEMIA_DN     2.340796e-07
## BOSCO_ALLERGEN_INDUCED_TH2_ASSOCIATED_MODULE 6.929188e-07
```

While gene set testing is useful for providing some biological insight in terms of what pathways might be affected by abberant methylation, care should be taken not to over-interpret the results. Gene set testing should be used for the purpose of providing some biological insight that ideally would be tested and validated in further laboratory experiments. It is important to keep in mind that we are not observing gene level activity such as in RNA-Seq experiments, and that we have had to take an extra step to associate CpGs with genes.

3.2 Differential variability

Rather than testing for differences in mean methylation, we may be interested in testing for differences between group variances. For example, it has been hypothesised that highly variable CpGs in cancer may contribute to tumour heterogeneity (Hansen et al. 2011). Hence we may be interested in CpG sites that are consistently methylated in one group, but variably methylated in another group.

Sample size is an important consideration when testing for differentially variable CpG sites. In order to get an accurate estimate of the group variances, larger sample sizes are required than for estimating group means. A good rule of thumb is to have at least ten samples in each group (Phipson and Oshlack 2014). To demonstrate testing for differentially variable CpG sites, we will use a publicly available dataset on ageing GSE30870 (http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE30870), where whole blood samples were collected from 18 centenarians and 18 newborns and profiled for methylation on the 450k array (Heyn et al. 2012). The data (`age.rgSet`) and sample information (`age.targets`) have been included as an R data object in both the workflow package or the data archive you downloaded from figshare (https://figshare.com/s/7a37f43c0ca2fec4669e). We can load the data using the `load` command, after which it needs to be normalised and filtered as previously described.

```
load(file.path(dataDirectory,"ageData.RData"))

# calculate detection p-values
age.detP <- detectionP(age.rgSet)

# pre-process the data after excluding poor quality samples
age.mSetSq <- preprocessQuantile(age.rgSet)


## [preprocessQuantile] Mapping to genome.


## [preprocessQuantile] Fixing outliers.


## [preprocessQuantile] Quantile normalizing.
```

```
# add sex information to targets information
age.targets$Sex <- getSex(age.mSetSq)$predictedSex

# ensure probes are in the same order in the mSetSq and detP ob
jects
age.detP <- age.detP[match(featureNames(age.mSetSq),rownames(ag
e.detP)),]
# remove poor quality probes
keep <- rowSums(age.detP < 0.01) == ncol(age.detP)
age.mSetSqFlt <- age.mSetSq[keep,]

# remove probes with SNPs at CpG or single base extension (SBE)
site
age.mSetSqFlt <- dropLociWithSnps(age.mSetSqFlt, snps = c("Cp
G", "SBE"))

# remove cross-reactive probes
keep <- !(featureNames(age.mSetSqFlt) %in% xReactiveProbes$Targ
etID)
age.mSetSqFlt <- age.mSetSqFlt[keep,]
```

As this dataset contains samples from both males and females, we can use it to demonstrate the effect of removing sex chromosome probes on the data. The MDS plots below show the relationship between the samples in the ageing dataset before and after sex chromosome probe removal (Figure 13). It is apparent that before the removal of sex chromosome probes, the sample cluster based on sex in the second principal component. When the sex chromosome probes are removed, age is the largest source of variation present and the male and female samples no longer form separate clusters.

```
# tag sex chromosome probes for removal
keep <- !(featureNames(age.mSetSqFlt) %in% ann450k$Name[ann450k
$chr %in%
                                                        c
("chrX","chrY")])

age.pal <- brewer.pal(8,"Set1")
par(mfrow=c(1,2))
plotMDS(getM(age.mSetSqFlt), top=1000, gene.selection="common",
        col=age.pal[factor(age.targets$Sample_Group)], labels=a
ge.targets$Sex,
        main="With Sex CHR Probes")
legend("topleft", legend=levels(factor(age.targets$Sample_Grou
p)),
       text.col=age.pal)

plotMDS(getM(age.mSetSqFlt[keep,]), top=1000, gene.selection="c
ommon",
        col=age.pal[factor(age.targets$Sample_Group)], labels=a
ge.targets$Sex,
        main="Without Sex CHR Probes")
legend("top", legend=levels(factor(age.targets$Sample_Group)),
       text.col=age.pal)
```