Analyze Illumina Infinium methylation microarray data

Pan Du[‡,*] Gang Feng[‡,†] Spencer Huang[‡,‡] Warren A. Kibbe[‡,§] Simon Lin[‡,¶]

October 1, 2012

[‡]Biomedical Informatics Center
Northwestern University, Chicago, IL, 60611, USA

Contents

1 Overview **2**

2 Major classes of Illumina methylation microarray data **2**

3 Data preprocessing **3**
3.1 Read the methylation microarray data 3
3.2 Example methylation datasets 3
3.3 Check data distribution 6
3.4 Check color balance 8
3.5 Quality assessment based on the distribution of CpG-site Intensity 16
3.6 Color balance adjustment 22
3.7 Background level correction 24
3.8 Data normalization 28
3.9 Modeling the methylation status 32
3.10 Use user provided preprocessing functions 41
3.11 Options to separately process each color channel 42
3.12 About the detection p-value of a CpG site 42

4 Gene annotation **43**

5 Performance comparison **43**

6 A use case: from raw data to functional analysis **43**
6.1 Preprocessing the Illumina Infinium Methylation microarray data 43
6.2 Identify differentially methylated genes and Functional analysis . 44
6.3 GEO submission of the methylation data 44

[*]dupan (at) northwestern.edu
[†]g-feng (at) northwestern.edu
[‡]huangcc (at) northwestern.edu
[§]wakibbe (at) northwestern.edu
[¶]s-lin2 (at) northwestern.edu

7 Session Info **45**

8 Acknowledgement **45**

9 References **45**

1 Overview

This is a tutorial of processing Illumina Infinium methylation microarray data using *lumi* package. For the processing of Illumina GoldenGate methylation microarray data, please check the *methylumi* package. The tutorial will briefly describe the class structure, supported methods and functions in quality and color balance assessment, color balance adjustment, background adjustment, normalization and modeling the methylation status. It will also illustrate why we suggest using M-value instead of Beta-value in the statistics analysis.

2 Major classes of Illumina methylation microarray data

By default, Illumina suggests using the Beta-value to quantify the methylation levels. The Beta-value is a ratio between Illumina methylated probe intensity and total probe intensities (sum of methylated and unmethylated probe intensities). It is in the range of 0 and 1, which can be interpreted as the percentage of methylation. However, we have shown the Beta-value method has severe heteroscedasticity in the low and high methylation range, which imposes serious challenges in applying many statistic models [1]. The M-value, which is the log2 ratio of methylated probe intensity and unmethylated probe intensity, is another method used to measure the methylation level. We have shown the M-value method is approximately homoscedastic in the entire methylation range. As a result, it is more statistically valid in differential and other statistic analysis [1].

For this reason, we defined a new *MethyLumiM* class. *MethyLumiM* is a class inherited from *ExpressionSet* class. The "assayData" slot includes three required data matrix, which are "exprs", "methylated" and "unmethylated". The "exprs" is a matrix of M-values, which is the log2 ratio of methylated and unmethylated probe intensities; "methylated" and "unmethylated" are intensity matrix measured by methylated and unmethylalted probes of Illumina Infinium methylation microarray. "detection" is an optional matrix in "assayData" slot. It is the detection p-value outputted by Illumina GenomeStudio software. To keep the control data information, which is mainly used for the quality control purpose, the *MethyLumiM* class has a "controlData" slot. The "controlData" slot can be either NULL or a *MethyLumiQC* object. Users are able to plot the control data using `plotControlData` function.

We use the `methylumiR` function in *methylumi* package to read in the Illumina methylation data, which was outputted by Illumina GenomeStudio or BeadStudio software. The `methylumiR` function returns a *MethyLumiSet* class object. For convenience, we wrote a coerce function to map from *MethyLumi*, *MethyLumiSet* or other *eSet* inherited objects to *MethyLumiM* class object.

And We wrote a `lumiMethyR` function as a wrap function of `methylumiR` and coerce the returned object as a *MethyLumiM* class object.

3 Data preprocessing

The first thing is to load the *lumi* package.

```
> library(lumi)
```

3.1 Read the methylation microarray data

We define a `lumiMethyR` function to read the Illumina methylation data, which is a wrap of the `methylumiR` function in *methylumi* package. The `lumiMethyR` function coerces the returned object (in *MethyLumiSet* class) as a *MethyLumiM* class object. The annotation library is not required if the GenomeStudio output data has already included the "COLOR_CHANNEL" column.

```
> ## specify the file name
> # fileName <- 'Example_Illumina_Methylation_profile.txt'
> ## load the data
> # example.lumiMethy <- lumiMethyR(fileName, lib="IlluminaHumanMethylation27k.db")
```

3.2 Example methylation datasets

The *lumi* package includes two example datasets. One is a control and treatment dataset (included in `example.lumiMethy` data object), which is measured in two batches. Another one is a titration dataset (included in `example.methyTitration` data object). To save storage space, only a randomly selected subset rows (CpG sites) were kept in the example datasets.

The control and treatment dataset (`example.lumiMethy`) includes four control and four treatment samples together with their technique replicates. The original samples and technique replicates were measured in two batches. Figure 1 and 2 show the overall sample relations of the control and treatment dataset in a PCA plot and Hierarchical cluster tree. We can see the batch difference is the major effect of sample differences.

```
> ## load example data (a methyLumiM object)
> data(example.lumiMethy)
> ## summary of the example data
> example.lumiMethy
```

```
MethyLumiM (storageMode: lockedEnvironment)
assayData: 5000 features, 16 samples
  element names: detection, exprs, methylated, unmethylated
protocolData: none
phenoData
  sampleNames: Treat1 Treat2 ... Ctrl4.rep (16 total)
  varLabels: sampleID label
  varMetadata: labelDescription
featureData
```

```
> plotSampleRelation(example.lumiMethy, method='mds', cv.Th=0)
```

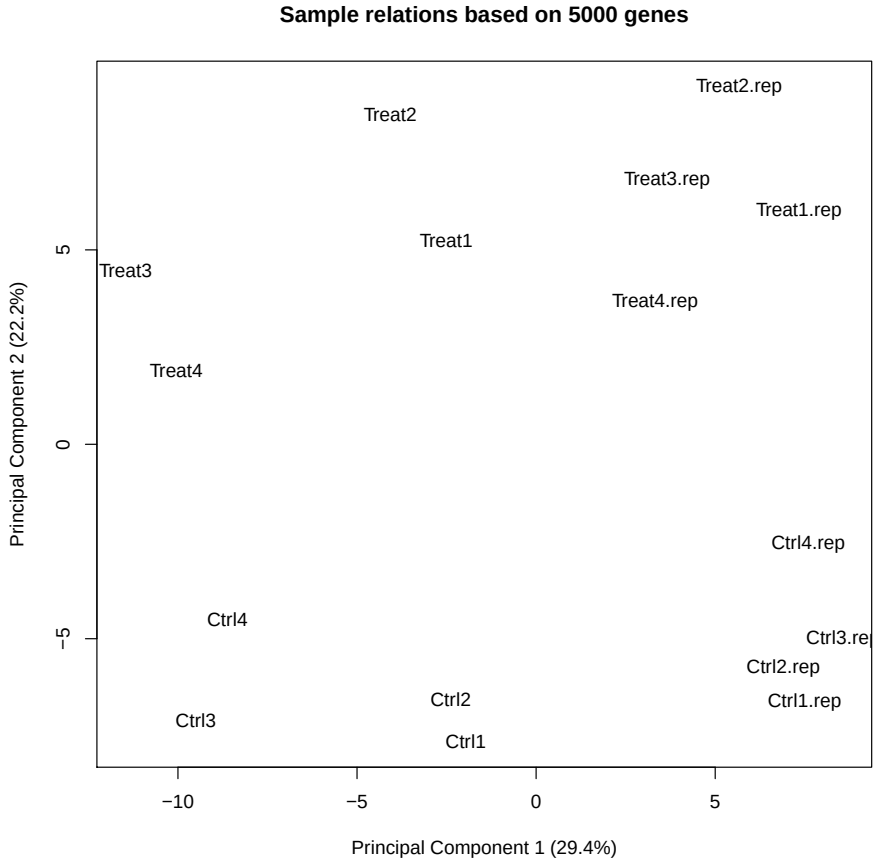

Figure 1: Overall sample relations in example.lumiMethy dataset

```
  featureNames: cg00002426 cg00012386 ... cg27662379 (5000 total)
  fvarLabels: CHR COLOR_CHANNEL
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
```

```
> ## print sample Names
> sampleNames(example.lumiMethy)
```

```
 [1] "Treat1"     "Treat2"     "Treat3"     "Treat4"     "Ctrl1"
 [6] "Ctrl2"      "Ctrl3"      "Ctrl4"      "Treat1.rep" "Treat2.rep"
[11] "Treat3.rep" "Treat4.rep" "Ctrl1.rep"  "Ctrl2.rep"  "Ctrl3.rep"
[16] "Ctrl4.rep"
```

The methylation titration dataset (`example.methyTitration`) includes 8 samples: "A1", "A2", "B1", "B2", "C1", "C2", "D" and "E". They are mixtures of Sample A (a B-lymphocyte sample) and Sample B (is (a colon cancer sample

```
> plotSampleRelation(example.lumiMethy, method='cluster', cv.Th=0)
```

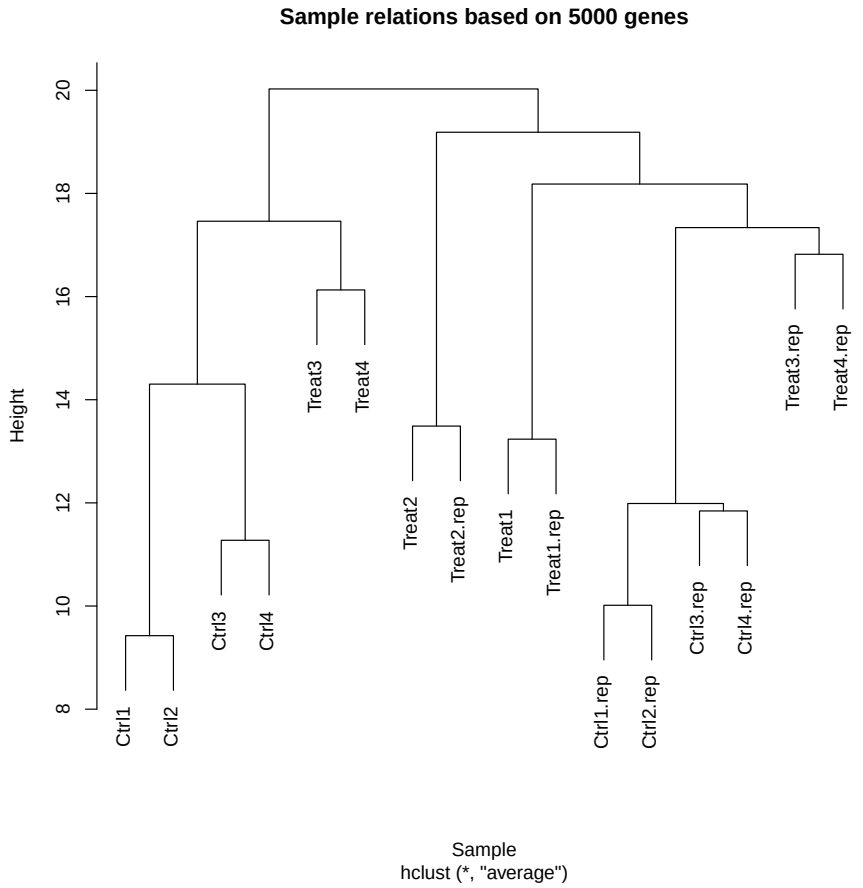

Figure 2: Overall sample relations shown as a hierarchical tree

from a female donor) at five different titration ratios: 100:0 (A), 90:10 (C), 75:25 (D), 50:50 (E) and 0:100 (B). Sample A, B and C have technique replicates. Figure 3 shows the overall sample relations of the titration dataset in a PCA plot.

```
> ## load the tritration data (a methyLumiM object)
> data(example.methyTitration)
> ## summary of the example data
> example.methyTitration

MethyLumiM (storageMode: lockedEnvironment)
assayData: 10000 features, 8 samples
  element names: detection, exprs, methylated, unmethylated
protocolData: none
phenoData
  sampleNames: A1 B1 ... E (8 total)
  varLabels: sampleID label
  varMetadata: labelDescription
featureData
  featureNames: cg09234859 cg22654935 ... cg04283392 (10000 total)
  fvarLabels: TargetID SYMBOL COLOR_CHANNEL CHR
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:

> ## print sample Names
> sampleNames(example.methyTitration)

[1] "A1" "B1" "C1" "A2" "B2" "C2" "D"  "E"
```

3.3 Check data distribution

Similar with expression microarray data, the *lumi* defines density and boxplot to check the overall distribution of the data. Figure 4 and 5 show the density of the example datasets. We can see the sample distributions can be quite different from each other. Because the density plot of M-values usually includes two modes, using the traditional boxplot cannot accurately represent the distribution of the data. We used another type of boxplot (method signature is *MethyLumiM* class) which is capable to show data with multiple modes. It uses different color levels to represent the M-values in different levels of HDRs (highest density regions) as specified in "prob" (probability of density coverage). The M-values locating outside the range of the maximum probability specified in "prob" are plotted as dots, which is similar with the outliers in the regular boxplot. By default, parameter "prob" is set as c(seq(10,90, by=10), 95). This means the M-values outside of 95% of HDR are plotted as outliers. Figure 6 shows the boxplot of M-values of example methylation data. The color depth represents the height of density. The short horizontal bar at the position with the deepest color corresponds to the maximum position of density distribution of each sample. Based on Figure 6, we can see the distribution difference across samples. Because the methylation levels (measured by M-values or Beta-values)

```
> plotSampleRelation(example.methyTitration, method='mds', cv.Th=0)
```

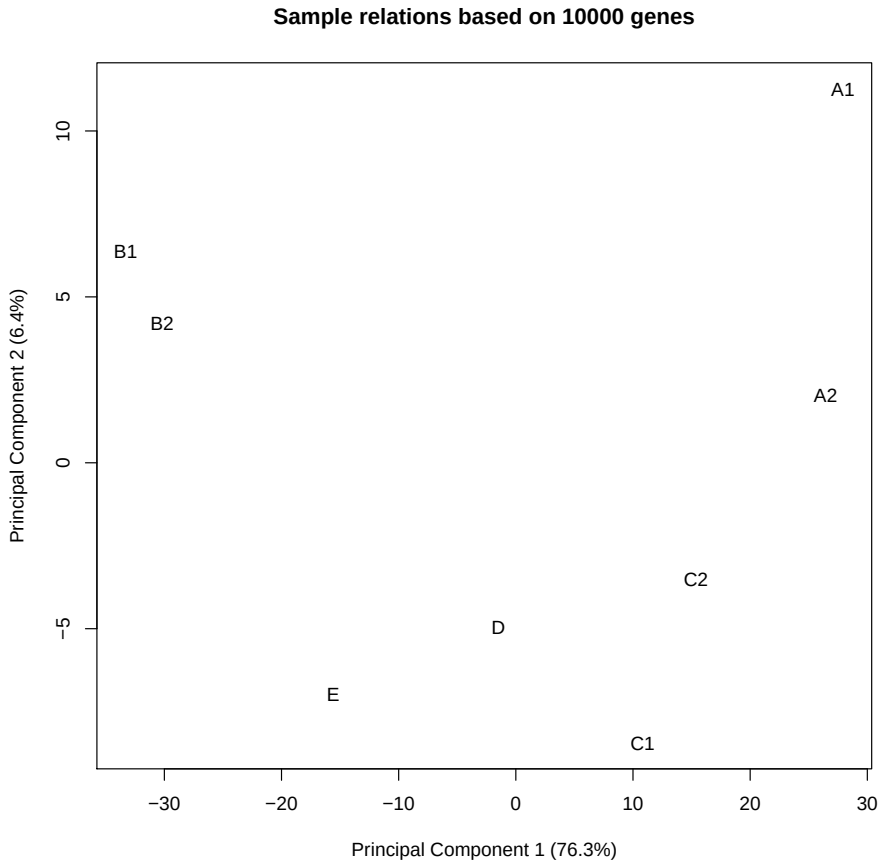

Figure 3: Overall sample relations of the titration dataset

```
> ## plot the density
> density(example.methyTitration, xlab="M-value")
```

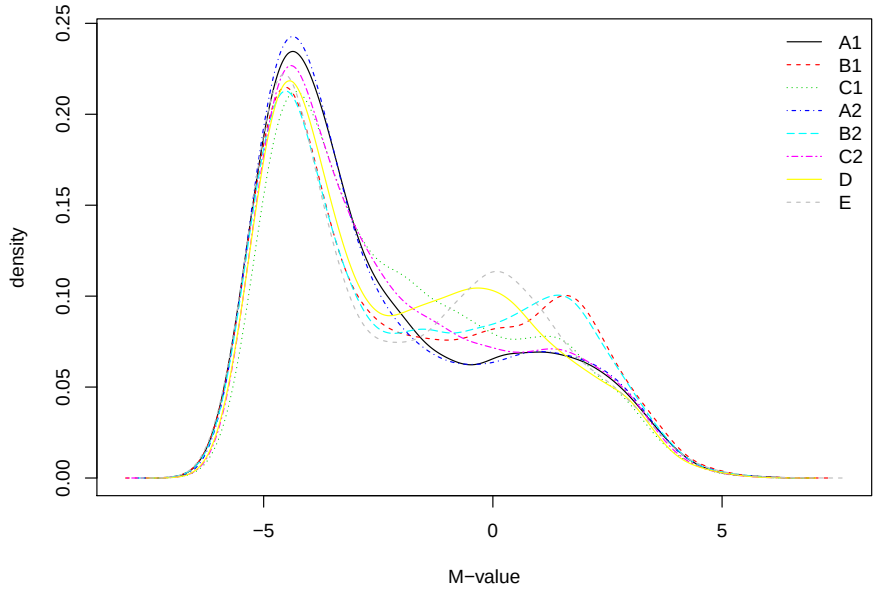

Figure 4: Density plot of M-value methylation levels of titration data before normalization

can have big changes across different conditions and treatment, only the methylation level distributions of technique or biological replicates are expected to have consistency. Therefore, we cannot claim a sample has quality issue if its distribution is quite different from others.

3.4 Check color balance

Based on the current Infinium assay design, Illumina uses two colors to label the final extended base following the hybridization of mehtylated or unmethylated probes. As a result, some of the CpG sites are measured in the red channel (final extended bases are A or T), whereas others are measured in the green channel (final extended bases are C or G). Note that the methylated and unmethylated probes of the same CpG site have the same color. Due to the difference in labeling efficiency and scanning properties of two color channels, the intensities measured in two color channels might be imbalanced. Because the methylation level is estimated based on the ratio of methylated and unmethylated probe intensities, many probe specific variations (like binding affinity and color balance) can be greatly reduced. However, because of the non-linearity of color effects, especially this effects will be different across different experiment conditions, color imbalance is not ignorable if the color effects are quite different across samples. Therefore, color balance adjustment will be important if the

```
> ## specify the colors of control and treatment samples
> sampleColor <- rep(1, ncol(example.lumiMethy))
> sampleColor[grep("Treat", sampleNames(example.lumiMethy))] <- 2
> density(example.lumiMethy, col=sampleColor, xlab="M-value")        ## plot the density
```

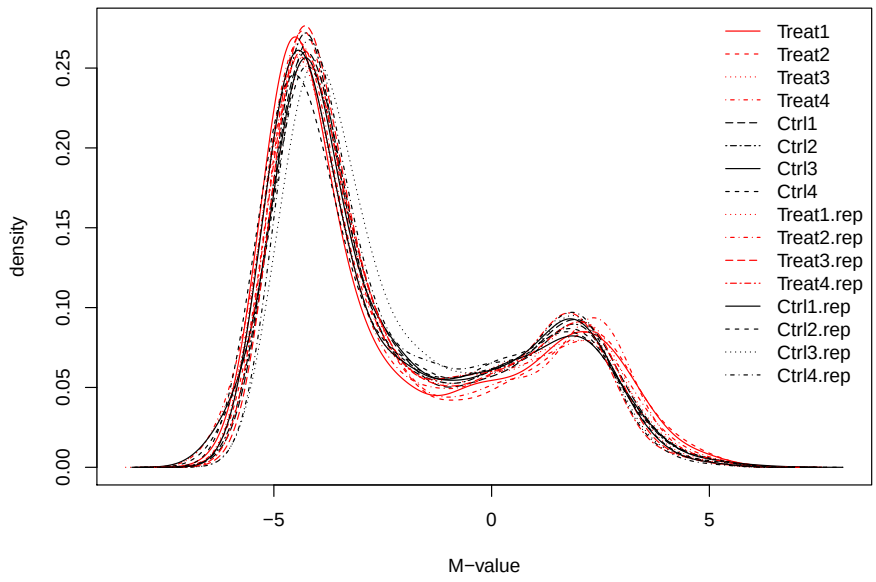

Figure 5: Density plot of M-value methylation levels before normalization

```
> ## Because the distribution of M-value has two modes, we use a boxplot different from re
> boxplot(example.lumiMethy)
```

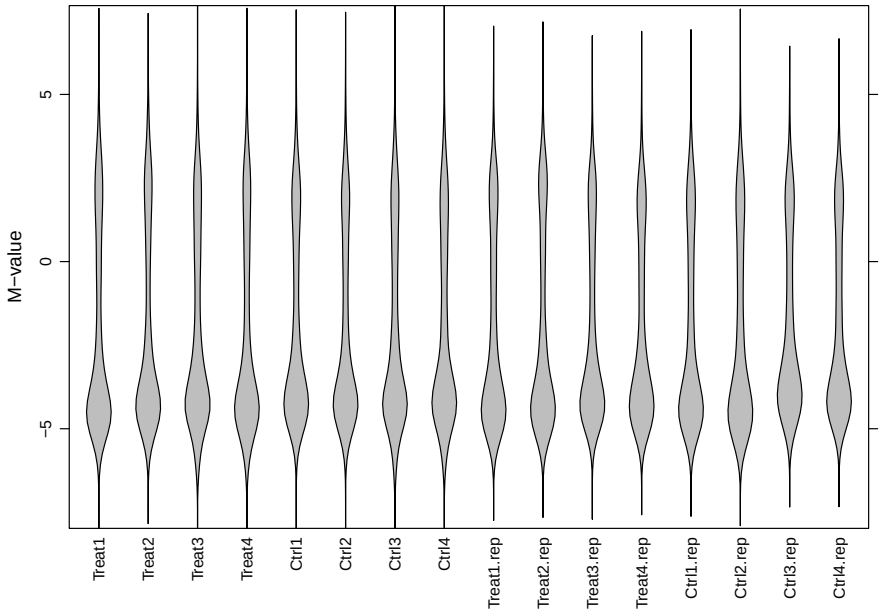

Figure 6: Multi-mode boxplot of M-value methylation levels before normalization

```
> plotColorBias1D(example.lumiMethy)
```

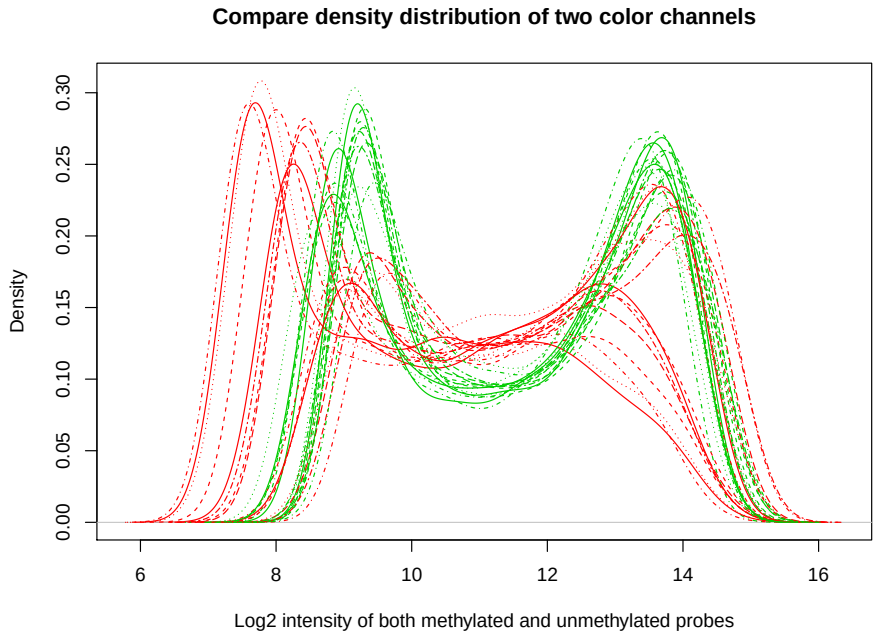

Figure 7: Density plot of two color channels (both methylated and unmethylated probe intensities)

color effect is very inconsistent across samples. Moreover, the inconsistence of color balance is another indicator of sample quality problems. So checking color balance is another important measure of QC.

The *lumi* package provides three different types of plotting functions to check color balance. Function `plotColorBias1D` plots the density of two color channels separately and shows them in red and green colors. By default, it pools both methylated and unmethylated probe intensities together, as shown in Figure 7. It can also separately plot the methylated and unmethylated probe intensities, as shown in Figure 8 and 9, or plot the CpG-site Intensity (details shown in the next subsection).

Similarly, we defined boxplot function, which plot two color channels separately. Function `boxplotColorBias` separately plots boxplot of the red and green color channels. Like `plotColorBias1D`, we can select to plot methylated and unmethylated probe intensities, as shown in Figure 10 and 11. Function `colorBiasSummary` can produce a quantile CpG-site Intensity summary of each sample. It has the same options of "channel" parameter.

```
> ## summary of color balance information of individual samples
> colorBiasSummary(example.lumiMethy[,1:8], channel='methy')

$red
        Treat1 Treat2 Treat3 Treat4  Ctrl1  Ctrl2  Ctrl3  Ctrl4
```

```
> plotColorBias1D(example.lumiMethy, channel='methy')
```

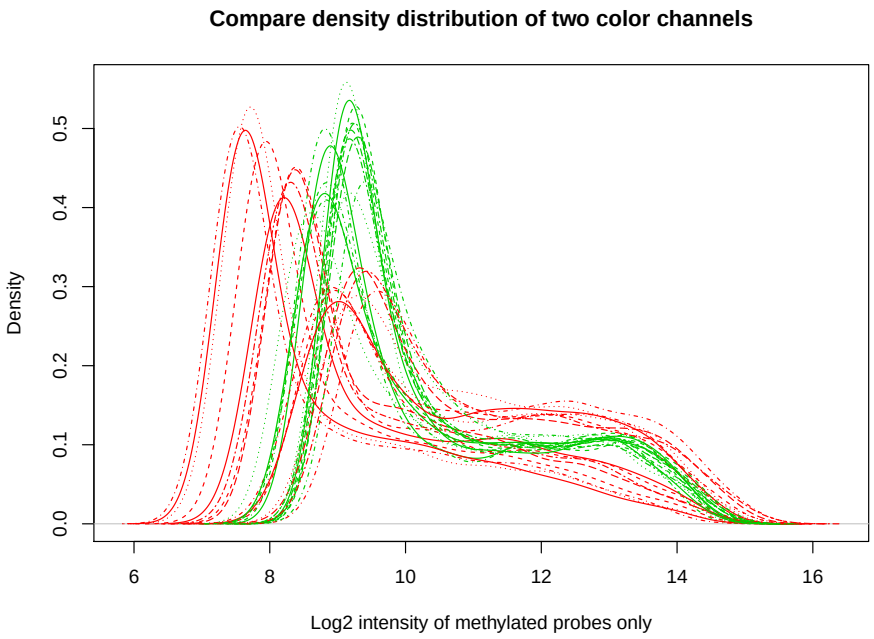

Figure 8: Density plot of two color channels (methylated probe intensities)

```
> plotColorBias1D(example.lumiMethy, channel='unmethy')
```

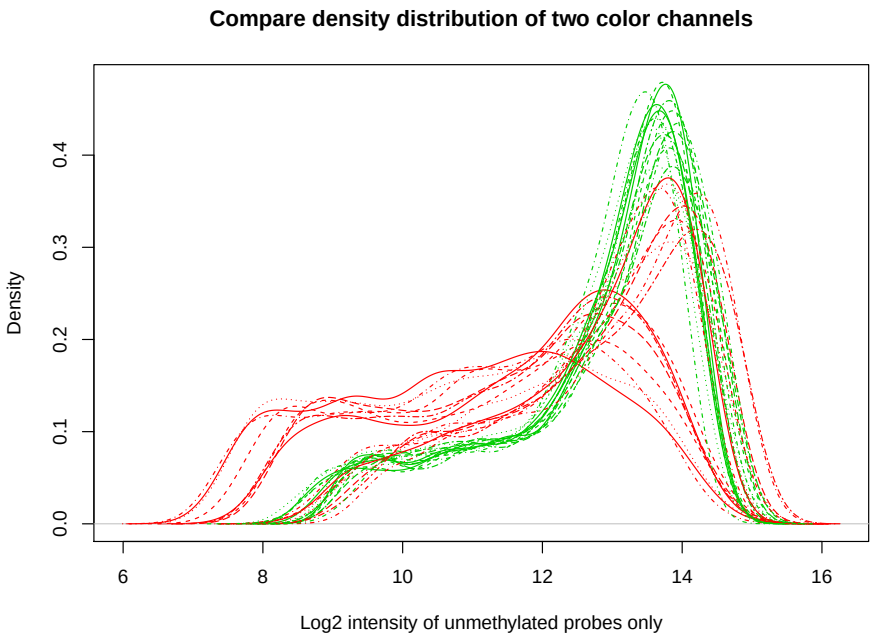

Figure 9: Density plot of two color channels (unmethylated probe intensities)

```
> boxplotColorBias(example.lumiMethy, channel='methy')
```

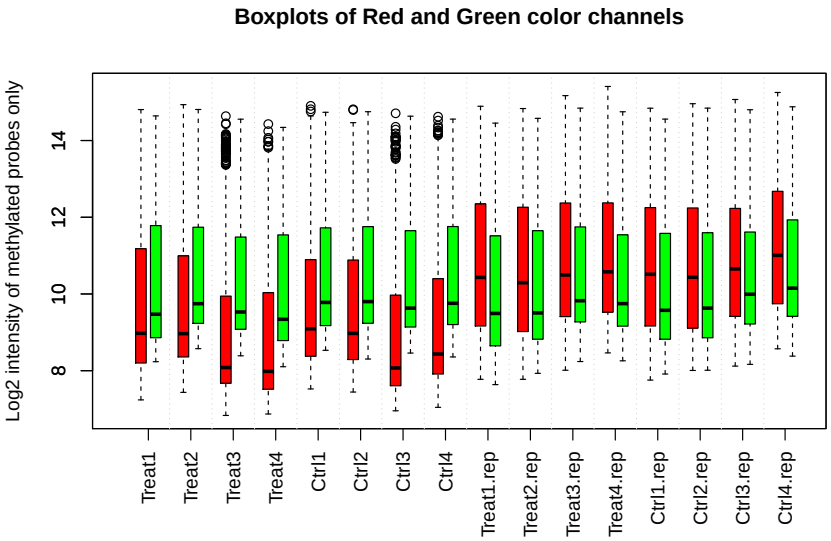

Figure 10: Box plot of two color channels (methylated probe intensities)

```
> boxplotColorBias(example.lumiMethy, channel='unmethy')
```

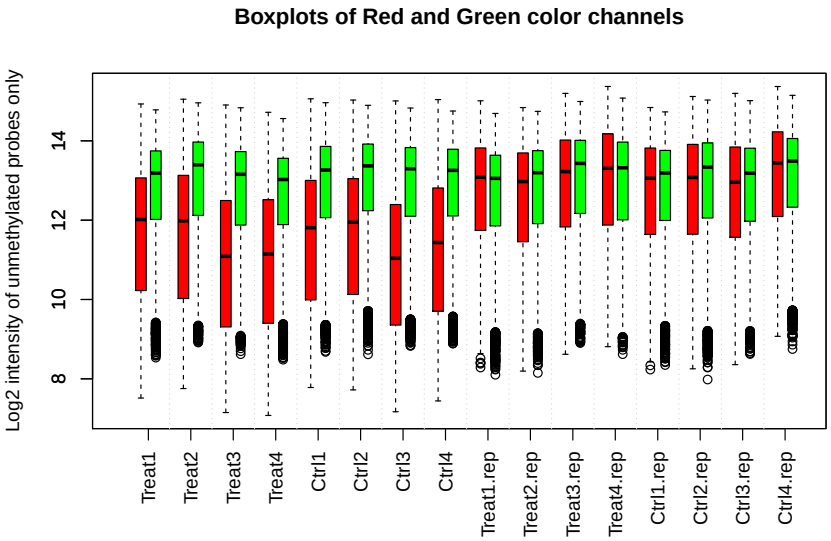

Figure 11: Box plot of two color channels (unmethylated probe intensities)

```
Min.     7.238  7.435  6.833  6.870  7.524  7.443  6.954  7.044
1st Qu.  8.200  8.358  7.672  7.516  8.375  8.290  7.607  7.913
Median   8.974  8.966  8.082  7.983  9.093  8.972  8.071  8.435
Mean     9.746  9.780  8.944  8.889  9.753  9.707  8.920  9.277
3rd Qu. 11.180 10.990  9.944 10.030 10.890 10.880  9.970 10.400
Max.    14.810 14.940 14.640 14.430 14.900 14.810 14.710 14.620

$green
        Treat1 Treat2 Treat3 Treat4  Ctrl1  Ctrl2  Ctrl3  Ctrl4
Min.     8.234  8.574  8.388  8.103  8.531  8.304  8.459  8.358
1st Qu.  8.860  9.234  9.079  8.785  9.175  9.236  9.138  9.202
Median   9.472  9.747  9.529  9.340  9.780  9.801  9.631  9.757
Mean    10.320 10.530 10.320 10.150 10.500 10.540 10.420 10.500
3rd Qu. 11.780 11.740 11.480 11.540 11.720 11.760 11.650 11.760
Max.    14.640 14.810 14.560 14.340 14.740 14.750 14.630 14.560
```

Function `plotColorBias2D` separately plots the methylated and unmethylated probe intensities in a 2-D scatter plot, and shows the interrogated CpG sites in red and green dots based on their color channels. Each time, `plotColorBias2D` can only plot one sample. By comparing the 2D scatter plots, we can easily see the details of color balance difference between samples. Figure 12 shows one example of color imbalance.

3.5 Quality assessment based on the distribution of CpG-site Intensity

Apart from color balance assessment, we propose to assess the sample quality based on the across sample distribution of the measured CpG site intensities. We defined the intensity of a measured CpG site, CpG-site Intensity, as the sum of methylated probe intensity and unmethylated probe intensity. As a single CpG site on one chromosome can only have two methylation status, if methylated probe for this CpG site has higher intensity, then the corresponding unmethylated probe should have lower intensity because only one probe (either methylated or unmethylated probe) can be bound on a particular CpG site on one copy of chromosome. That means the CpG-site Intensity is in proportional to the total copies of CpG sites. It also means that the methylation level (ratio between methylated and unmethylated probe intensities) changes should not have big effects on the CpG-site Intensity. Therefore, the CpG-site Intensity distribution of methylation microarray should still be similar across samples in different conditions if they have only methylation level difference, and we can check the sample quality based on CpG-site Intensity distribution. Figure 13 and 14 show the boxplot and density plot of two color channels. We can see it is much more obvious when using CpG-site Intensity to check color imbalance. Figure 15 and 16 show the CpG-site Intensity distribution of the example data. Because this example data has severe color imbalance, we can see the CpG-site Intensity distributions are similar but still have many differences. We can see the improvements after color balance adjustment.

We can also use other QC plots for expression data to check the sample consistency of methylation data base on CpG-site Intensity. Figure **??** shows pairwise plot of CpG-site Intensities of four samples, Ctrl1, Ctrl1.rep, Treat1

```
> plotColorBias2D(example.lumiMethy, selSample=1, cex=2)
```

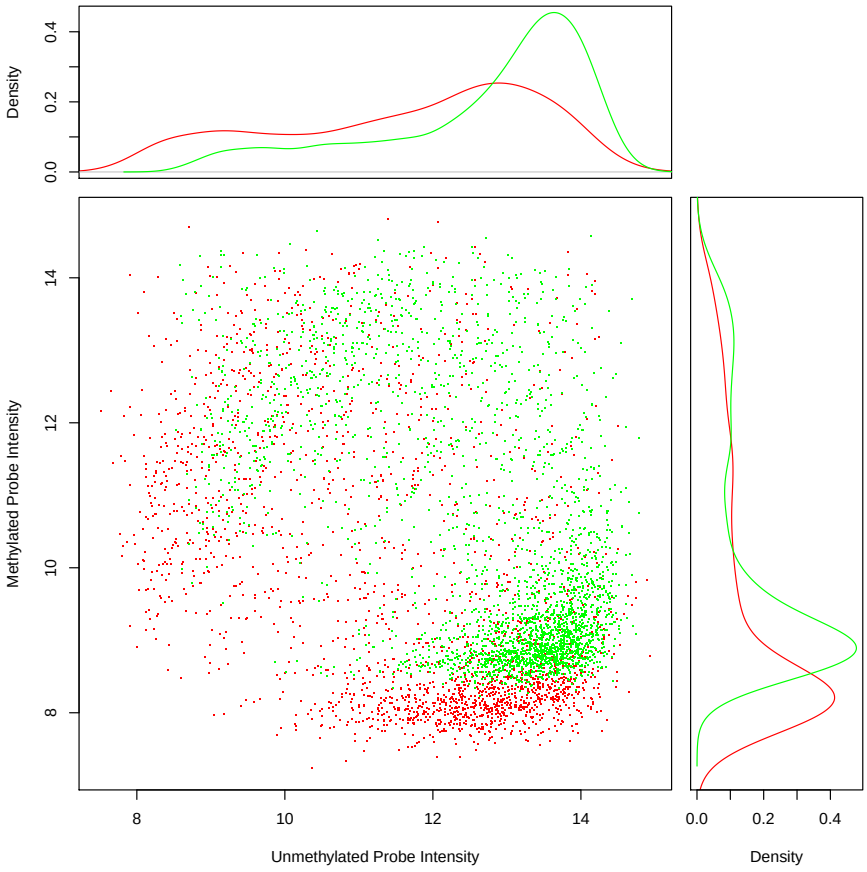

Figure 12: Scatter plot of methylated and unmethylated probe intensities to show color imbalance (example 1)

```
> boxplotColorBias(example.lumiMethy, channel='sum')
```

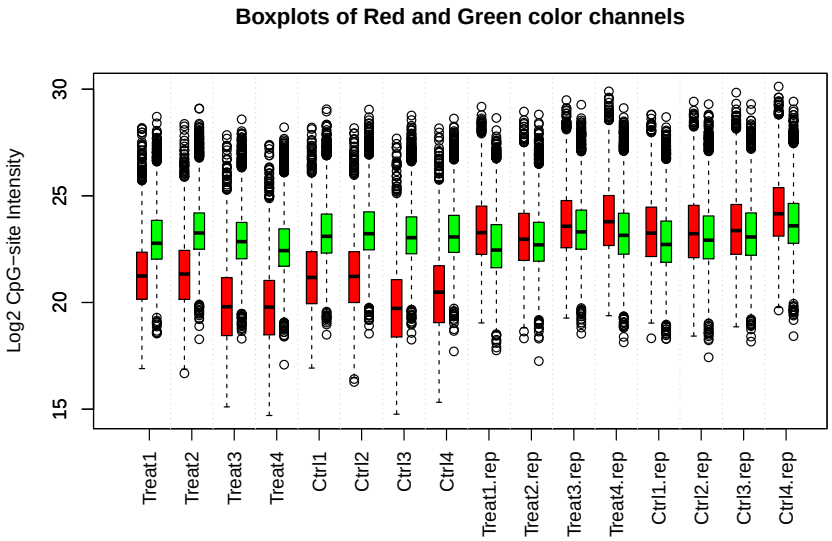

Figure 13: Box plot of of CpG-site Intensity (two color channels were plotted separately)

```
> plotColorBias1D(example.lumiMethy, channel='sum')
```

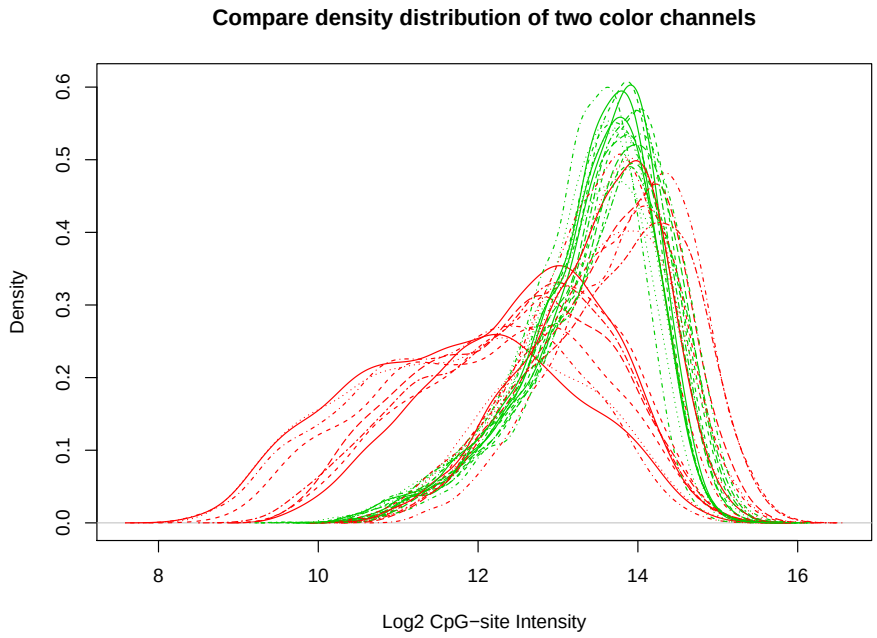

Figure 14: Density plot of CpG-site Intensity (two color channels were plotted separately)

```
> density(estimateIntensity(example.lumiMethy), xlab="log2(CpG-site Intensity)")
```

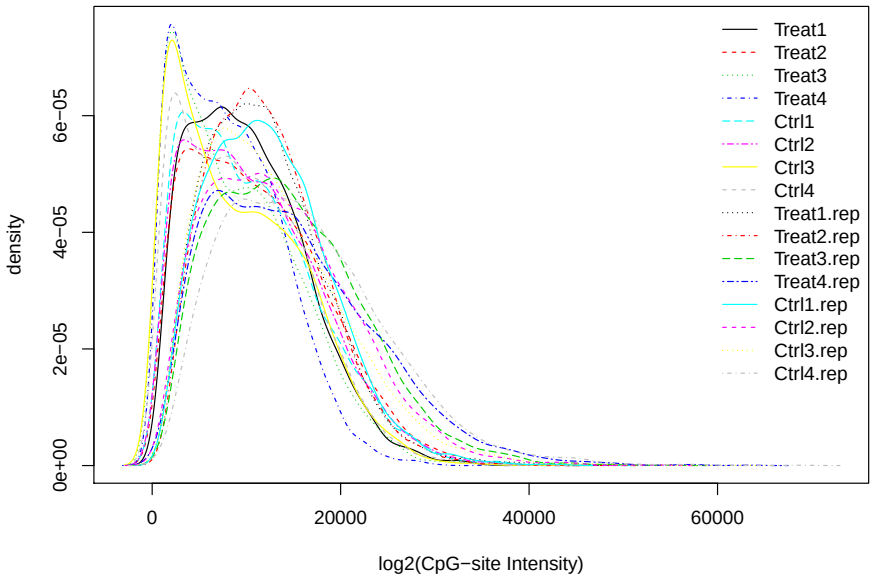

Figure 15: Density plot of CpG-site Intensity

```
> boxplot(estimateIntensity(example.lumiMethy))
```

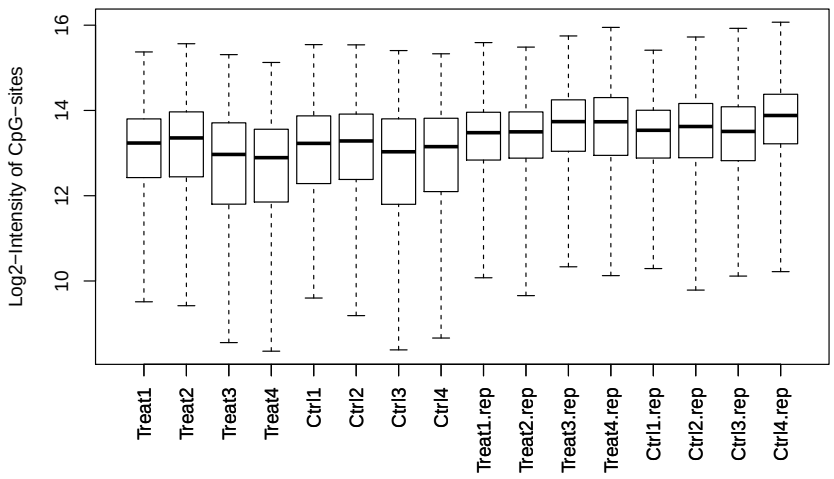

Figure 16: Boxplot of CpG-site Intensity

and Treat1.rep. Ctrl1, Ctrl1.rep and Treat1, Treat1.rep are technique replicates. We can clearly see two bands in the pairwise plot, which were caused by the difference of color imbalance between two batches.

3.6 Color balance adjustment

Based on the color balance assessment plots shown in previous two sections, we can see there are clear color balance differences between two batches in the example control and treatment dataset. That means we need to perform color balance adjustment before further analysis. Alternatively, we can separately process two color channels, but will course many inconveniences in the following up analysis. For example, we have to use different p-value or fold-change thresholds in identifying the differentially methylated CpG sites. Moreover, considering we may not have access to the color channel information later on, performing color balance adjustment first will make following up analysis much more convenient.

The *lumi* package provides `lumiMethyC` function for color balance adjustment. The basic idea of color balance adjustment is to treat it as the normalization between two color channels. Function `lumiMethyC` provides two color balance adjustment methods: "quantile" and "ssn". Because two color channels have different number of probes and not match each other, we cannot directly apply regular "quantile" normalization used in expression microarray analysis. To solve this problem, we designed a `smoothQuantileNormalization` method. Basically, it requires the quantile normalization should be smooth. By doing this, the `smoothQuantileNormalization` can further resolve one major withdraw of regular quantile normalization, i.e., quantile normalization doesn't work well in the areas with sparse density, like two extremes of data distribution. Figure 18 and 19 show density and boxplot of two color channels after color balance adjustment using smooth quantile normalization. Figure 20 shows the scatter plots after color balance adjustment using smooth quantile normalization, we can see the distribution becomes very similar between two color channels. And the change of pair plot after color balance is more obvious by comparing Figure 21 and Figure 17.

```
> ## summary of color balance information of individual samples
> lumiMethy.c.adj <- lumiMethyC(example.lumiMethy)

Perform quantile color balance adjustment ...
Processing sample ...
Processing sample ...
Processing sample ...
Processing sample ...
Processing sample ...
Processing sample ...
Processing sample ...
Processing sample ...
Processing sample ...
Processing sample ...
Processing sample ...
Processing sample ...
```

```
> ## get the color channel information
> colorChannel <- as.character(pData(featureData(example.lumiMethy))[, "COLOR_CHANNEL"])
> ## replace the "Red" and "Grn" as color names defined in R
> colorChannel[colorChannel == 'Red'] <- 'red'
> colorChannel[colorChannel == 'Grn'] <- 'green'
> ## select a subet of sample for pair plot
> selSample <- c( "Ctrl1", "Ctrl1.rep", "Treat1", "Treat1.rep")
> ## plot pair plot with the dots in scatter plot colored based on the color channels
> pairs(estimateIntensity(example.lumiMethy[, selSample]), dotColor= colorChannel, main="P
```

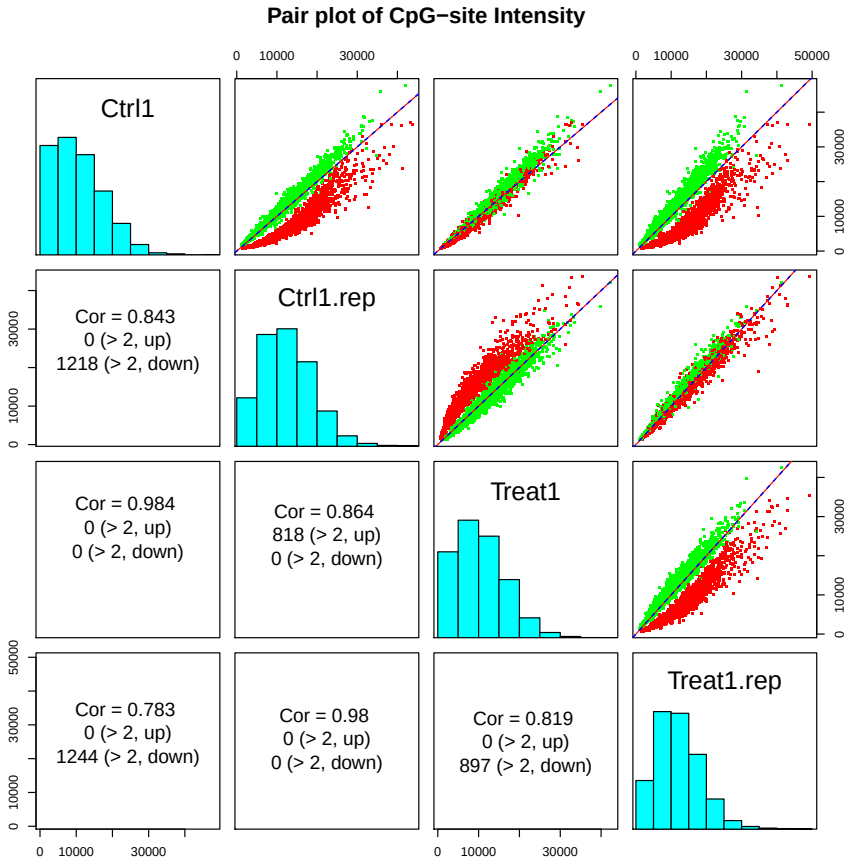

Figure 17: Pair plot of CpG-site Intensity with colors

```
> plotColorBias1D(lumiMethy.c.adj, channel='sum')
```

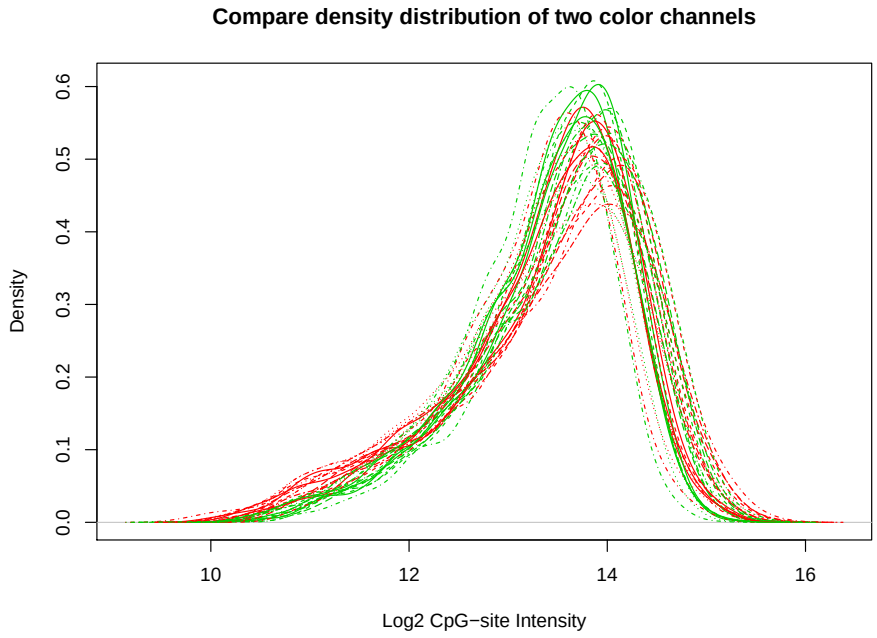

Figure 18: Density plot of CpG-site Intensity (two color channels were plotted separately) after color balance adjustment

```
Processing sample ...
Processing sample ...
Processing sample ...
Processing sample ...
```

3.7 Background level correction

Illumina provides negative control probes for the estimation of background levels. The information of the control probe information is kept in the controlData slot, which is a *MethyLumiQC* object. When the controlData includes the negative control probe information, the background estimation will be the median of the negative control probes. Red and Green color channels will be estimated separately. However, in many cases the information of negative control probes usually was not outputted together with the methylation data. As a result, we have to find other methods to estimate the background levels of individual samples. As Illumina methylation has two color channels, the background levels of two color channel can be different due to color imbalance. So color balance adjustment is suggested to be performed before background adjustment, or background estimation should be separately performed in each color channel.

The estimation of background level of Infinium methylaton microarray is based on the assumption that the lots of CpG sites are unmethylated, which

```
> boxplotColorBias(lumiMethy.c.adj, channel='sum')
```

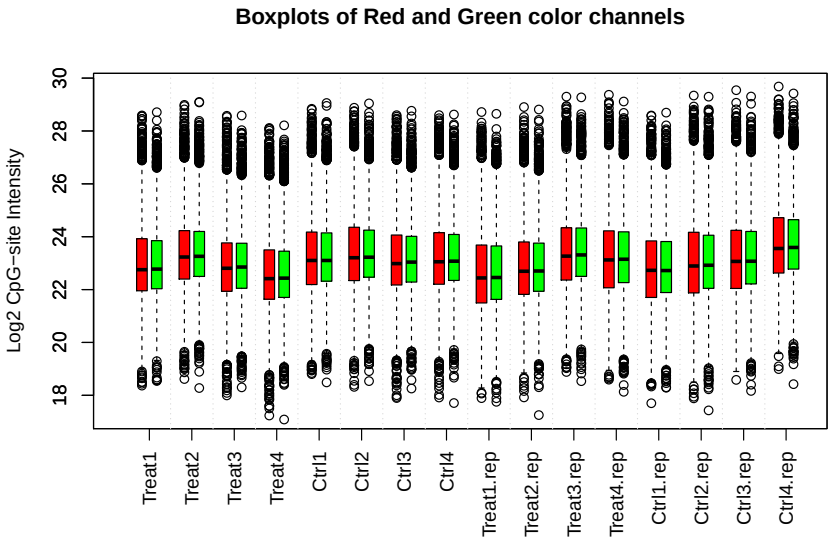

Figure 19: Box plot of of CpG-site Intensity (two color channels were plotted separately) after color balance adjustment

```
> ## plot the color balance adjusted scatter plot of two color channels
> plotColorBias2D(lumiMethy.c.adj, selSample=1, cex=2)
```

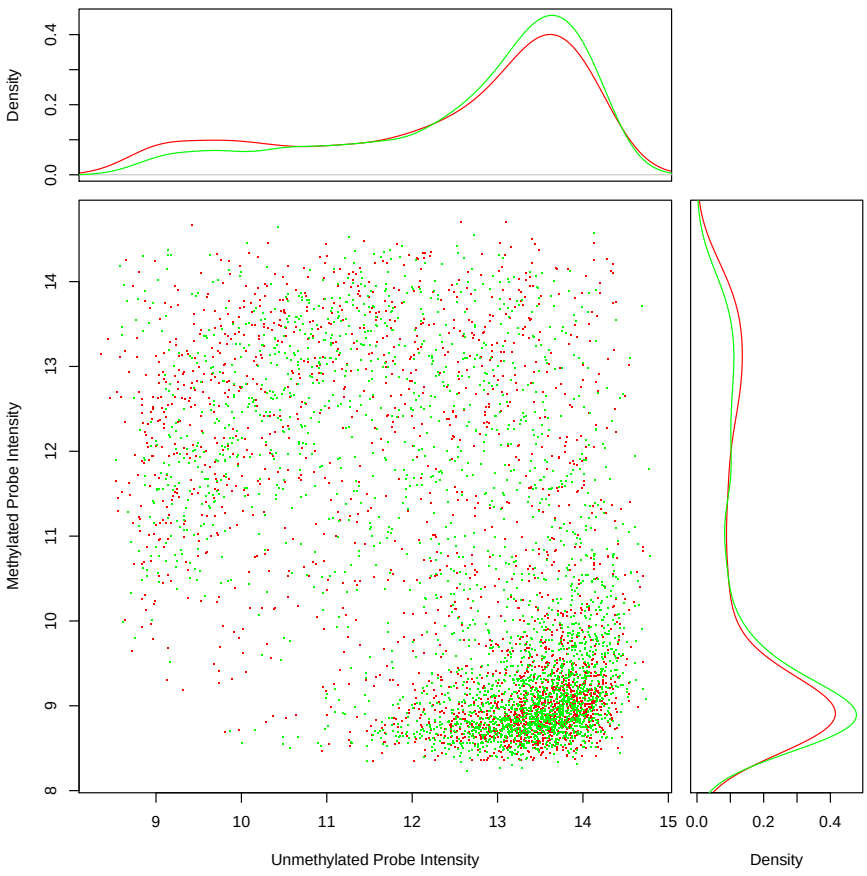

Figure 20: Scatter plot of methylated and unmethylated probe intensities after color balance adjustment (example 1)

```
> ## plot pairwise plot after color balance adjustment
> pairs(estimateIntensity(lumiMethy.c.adj[, selSample]), dotColor= colorChannel, main="Pai
```

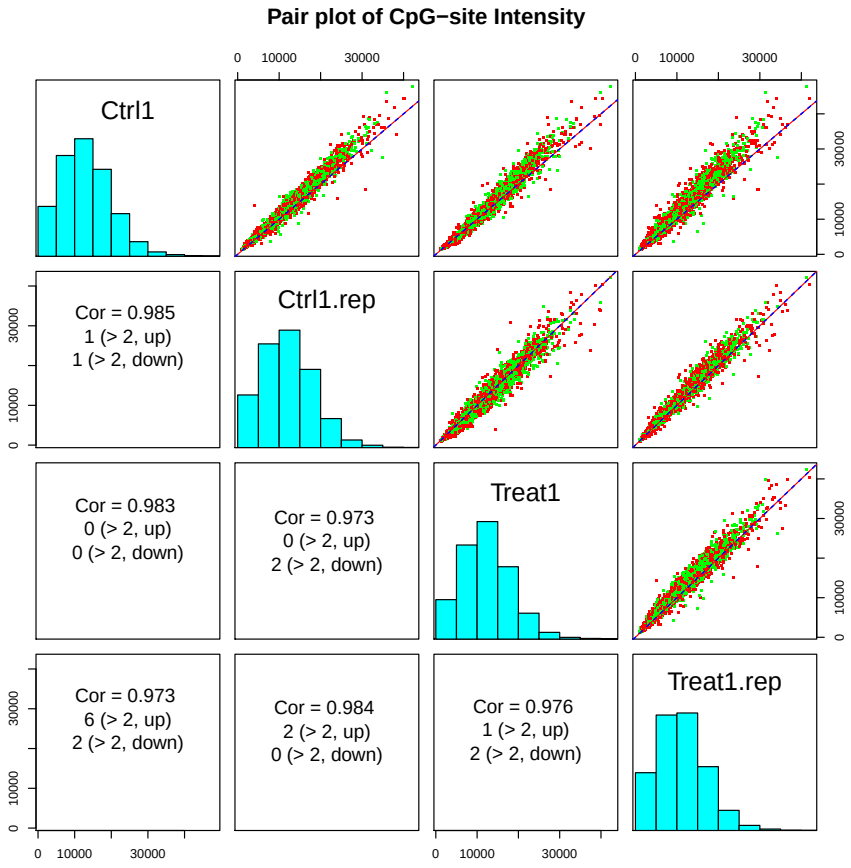

Figure 21: Pair plot of CpG-site Intensity with colors

results in a density mode of the intensities measured by methylated probes. The position of this mode represents the background level. Figure 22 shows the background mode of methylated probe data of first five example samples. We can see the differences between two color channels. Function `estimateMethylationBG` estimates the background level of each sample. Function `bgAdjustMethylation` adjusts the background levels based on the estimation returned by `estimateMethylationBG`. The `lumiMehtyB` function is a general function for background correction, which by default call `estimateMethylationBG` function. If we directly perform background adjustment on the raw data, we have to perform background adjustment separately on two color channels, which can be done by setting the parameter "separateColor" of `lumiMethyB` as TRUE. Figure 23 shows the density plot of methylated probes after background correction with two color channels processed separately. We can see the density modes of two color channels overlapping each other after background correction performed separately performed in each color channel. Alternatively, we can perform background adjustment after color balance adjustment, as shown in Figure 24.

```
> ##separately adjust backgrounds of two color channels
> lumiMethy.b.adj <- lumiMethyB(example.lumiMethy, method="bgAdjust2C", separateColor=TRUE

Perform bgAdjust2C background correction ...

> ##background adjustment of individual samples
> lumiMethy.bc.adj <- lumiMethyB(lumiMethy.c.adj, method="bgAdjust2C")

Perform bgAdjust2C background correction ...
```

3.8 Data normalization

Because the total amount of CpG methylation can differ significantly from sample to sample in different conditions, many assumptions used by mRNA expression microarrays are not valid in processing DNA methylation data. So directly applying the normalization methods designed for expression microarray to the methylation data, like M-value or Beta-value, is inappropriate. To circumvent this difficulty, we perform normalization at the probe level, i.e., normalize the intensities of methylated and unmethylated probes instead of the summarized methylation levels.

The *lumi* package provides `lumiMethyN` function for probe level normalization. Currently, it supports two methods, "ssn" and "quantile". Users can also provide there own normalization methods, as long as it imports and outputs a data matrix. See subsection " Use user provided preprocessing functions" of for more details. The "ssn" (simple scaling normalization) method first estimates the background level use function `estimateMethylationBG`, then scale the background subtracted data to the same average intensity level, and finally adjust the background level to a user specified level. The assumption of "quantile" normalization is that the intensity distribution of the pooled methylated and unmethylated probes, as shown in Figure 7, are the similar for different samples. The quantile normalization itself is the same as expression microarrays.

```
> ## plot the background mode of methylated probe data of first five example samples
> plotColorBias1D(example.lumiMethy[,1:5], channel='methy', xlim=c(-1000,5000), logMode=FA
```

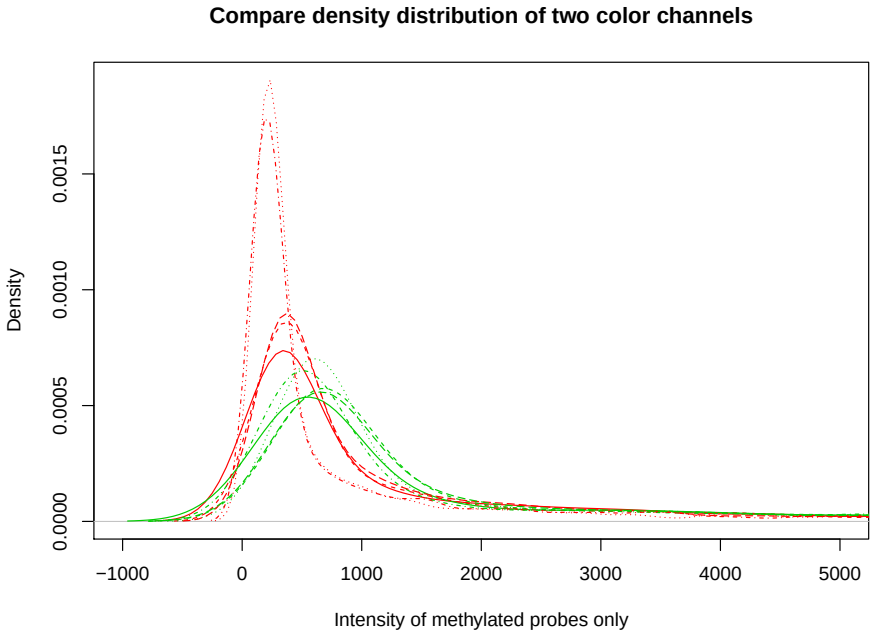

Figure 22: Background mode shown in the density plot of methylated probes

```
> ## plot the background mode of methylated probe data of first five example samples
> plotColorBias1D(lumiMethy.b.adj [,1:5], channel='methy', xlim=c(-1000,5000), logMode=FAL
```

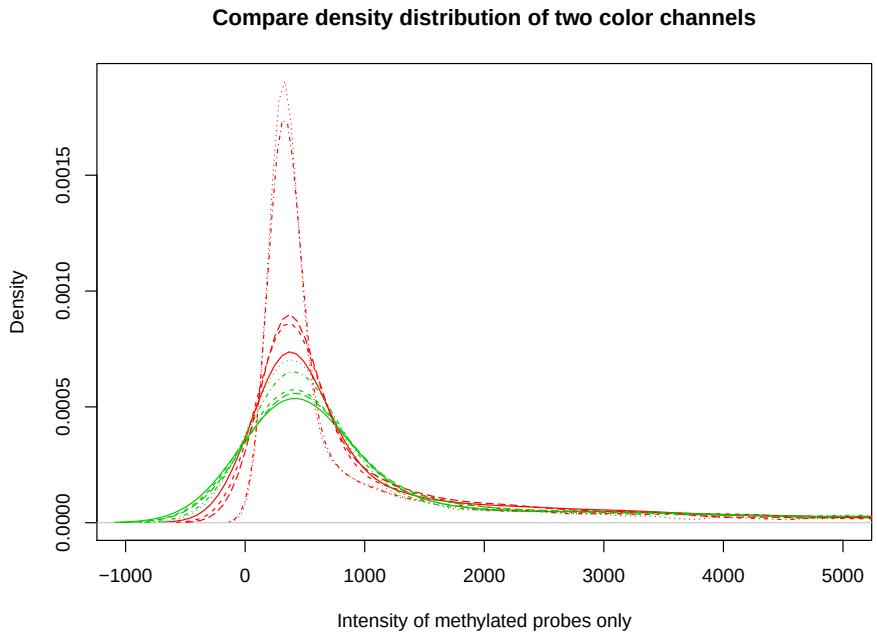

Figure 23: Background mode shown in the density plot of methylated probes after background adjustment (separately in two color channel)

```
> ## plot the background mode of methylated probe data of first five example samples
> plotColorBias1D(lumiMethy.bc.adj [,1:5], channel='methy', xlim=c(-1000,5000), logMode=FA
```

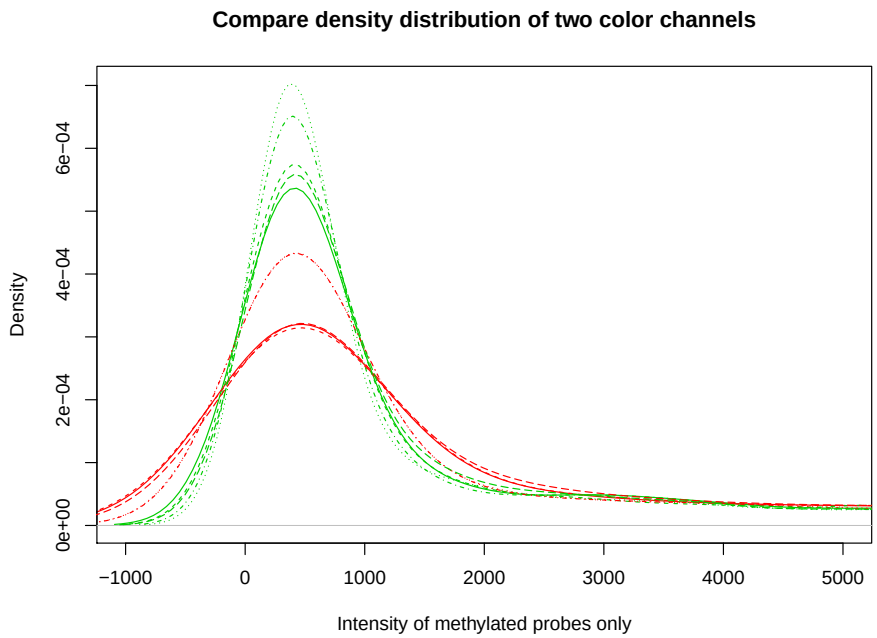

Figure 24: Background mode shown in the density plot of methylated probes after background and color adjustment

To check the effectiveness of normalization, we first check the sample relations after SSN normalization, as shown in Figure 25. We can see most of the technique replicates are clustered together after normalization. Comparing with the sample relations of the raw data shown in Figure 2, the improvement is obvious. Figure 26 and 27 show the density plots of methylation levels after SSN and quantile normalization, separately. The red and black colors represent "Treatment" and "Control" samples, respectively. We can see the mode positions of normalized data are more consistent across samples while keeping the methylation differences in the middle methylation range.

Figure 28 and 29 show the density plot of CpG-site Intensity after SSN and quantile normalization, respectively. Comparing with the density plot of CpG-site Intensity before preprocessing, shown in Figure 15, we can see they are much more consistent across samples. We can see the distribution of CpG-site Intensity of quantile normalized data is more consistent than SSN normalized data. This is because the assumption of quantile normalization is more aggressive than SSN, it assumes the distribution of pooled intensities of methylated and unmethylated probes are the same. The selection of SSN and quantile normalization is similar with the case in expression microarray. It depends on the data itself and the purpose of analysis. Figure 30 shows the color bias boxplot after preprocessing. We can see they are very consistent across samples. Finally, we show the pair plot of M-value after preprocessing. We expect there is little difference between technique replicates, but more difference between "Treatment" and "Control" samples, this is exactly what is shown in Figure 31. All of these results show the color balance adjustment plus normalization is effective.

```
> ## Perform SSN normalization based on color balance adjusted data
> lumiMethy.c.ssn <- lumiMethyN(lumiMethy.c.adj, method='ssn')

Perform ssn normalization ...

> ## Perform quantile normalization based on color balance adjusted data
> lumiMethy.c.quantile <- lumiMethyN(lumiMethy.c.adj, method='quantile')

Perform quantile normalization ...
```

3.9 Modeling the methylation status

Comparing with the Beta-value, we cannot directly know the methylation status of the CpG site based on the M-value. By checking the density plots of M-values, as shown in Figure 29 and Figure 28, we can see the density plot has two obvious modes, with one corresponds to the unmethylated CpG sites and another one corresponds to the methylated CpG sites. Based on this observation, we developed an algorithm to estimate the methylation status by fitting a two component Gamma mixture model. Function `gammaFitEM` estimates the parameters of gamma mixture models using EM algorithm. Figure 32 shows the fitted two component gamma mixture model in comparison with the histogram of the first sample in example.lumiMethy dataset.

Function `methylationCall` estimates the methylation status based on the fitting results of `gammaFitEM`. Three methylation statuses are reported: "Unmethy" (unmethylation posterior probability > unmethylation threshold), "Methy"

```
> plotSampleRelation(lumiMethy.c.ssn, method='cluster', cv.Th=0)
```

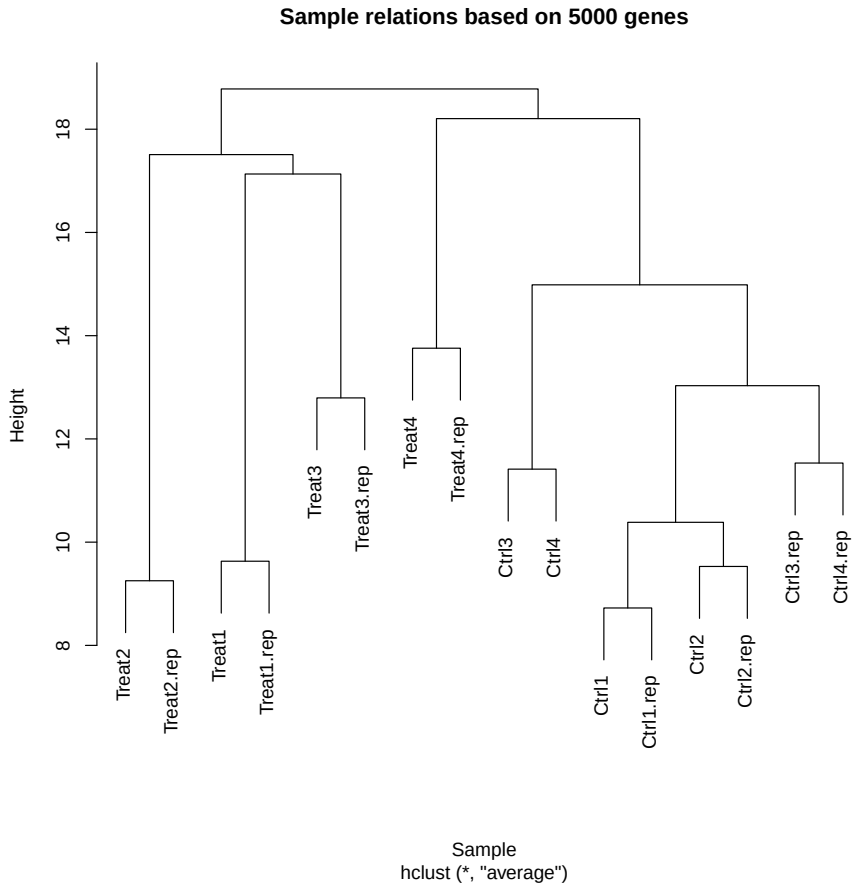

Figure 25: Overall sample relations shown as a hierarchical tree

```
> ## plot the density of M-values after SSN normalization
> density(lumiMethy.c.ssn, col= sampleColor, main="Density plot after SSN normalization")
```

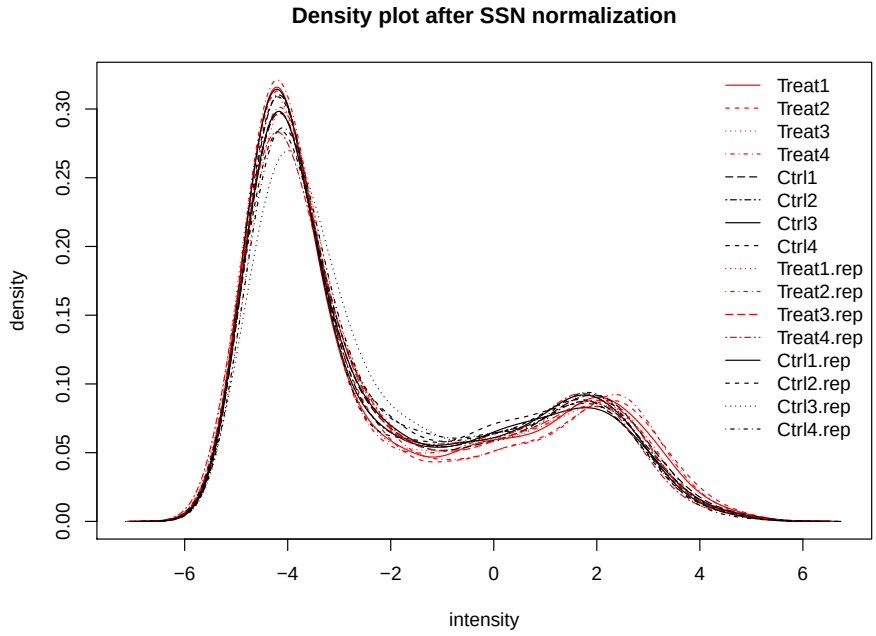

Figure 26: Density plot of M-values after SSN normalization

```
> ## plot the density of M-values after quantile normalization
> density(lumiMethy.c.quantile, col= sampleColor, main="Density plot after quantile normal
```

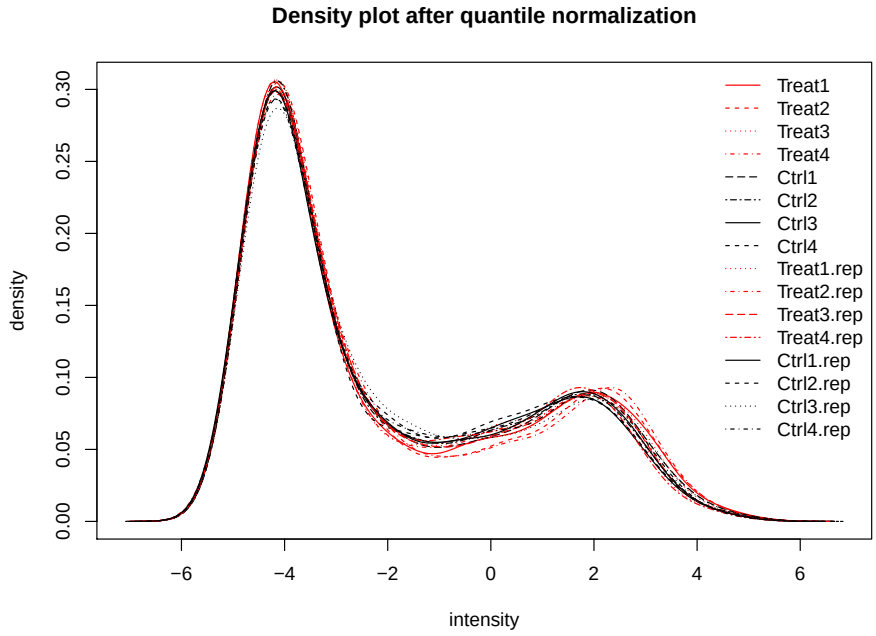

Figure 27: Density plot of M-values after quantile normalization

```
> density(estimateIntensity(lumiMethy.c.ssn), col= sampleColor,  xlab="log2(CpG-site Inten
```

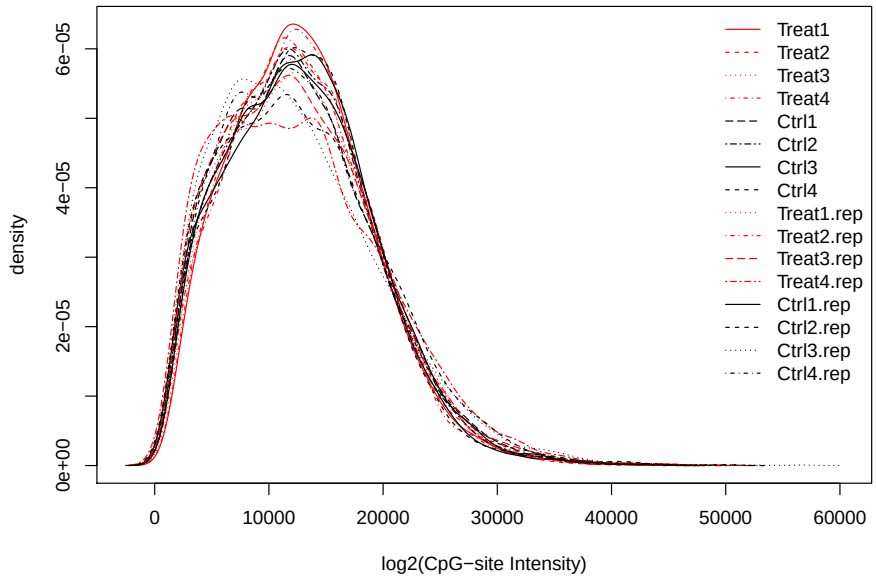

Figure 28: Density plot of CpG-site Intensity after quantile SSN normalization

```
> density(estimateIntensity(lumiMethy.c.quantile), col= sampleColor,  xlab="log2(CpG-site
```

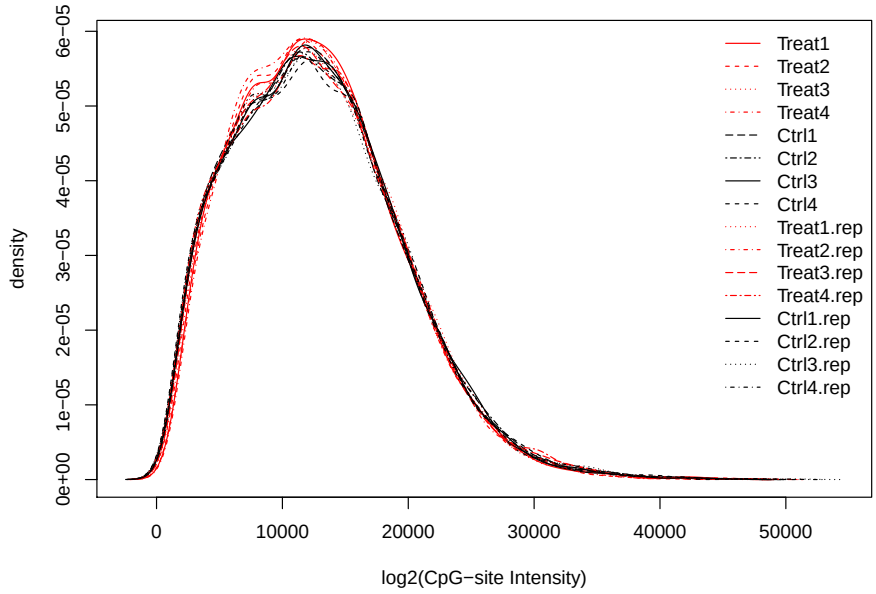

Figure 29: Density plot of CpG-site Intensity after quantile normalization

```
> boxplotColorBias(lumiMethy.c.quantile, channel='sum')
```

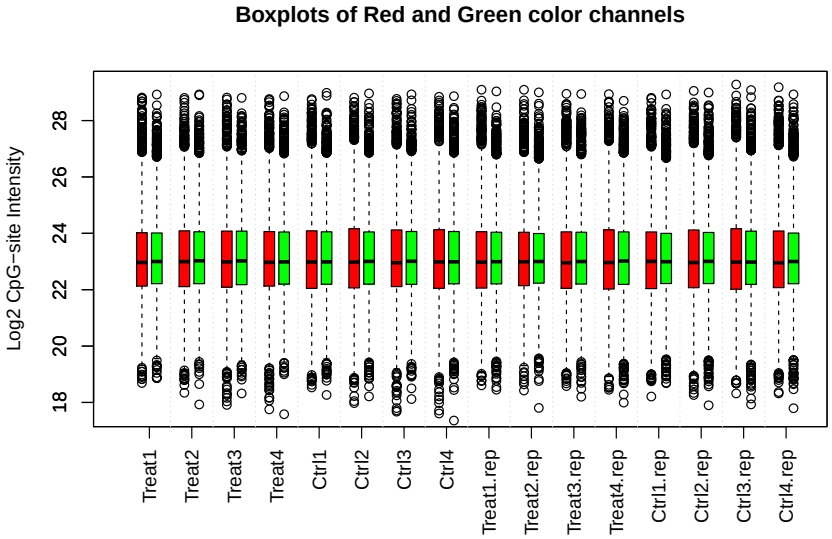

Figure 30: Box plot of of CpG-site Intensity (two color channels were plotted separately) after normalization

```
> ## select a subet of sample for pair plot
> selSample <- c( "Ctrl1", "Ctrl1.rep", "Treat1", "Treat1.rep")
> ## plot pair plot with the dots in scatter plot colored based on the color channels
> pairs(lumiMethy.c.quantile[, selSample], dotColor= colorChannel, main="Pair plot of M-va
```

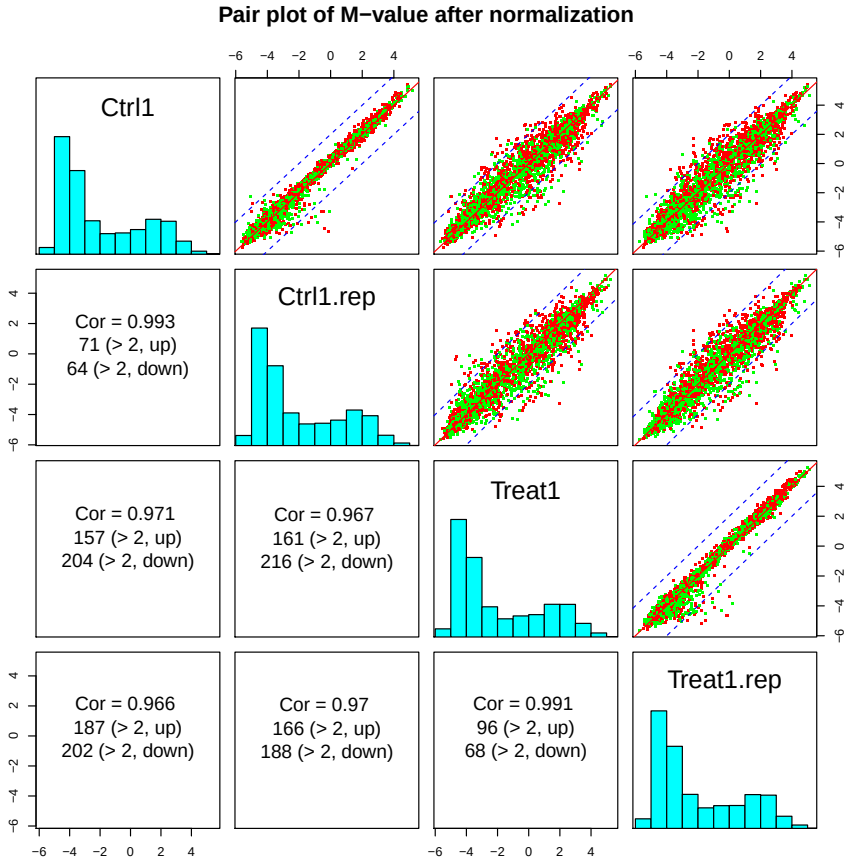

Figure 31: Pair plot of M-value after normalization

```
> ## Fit the two component gamma mixture model of the first sample of example.lumiMethy
> fittedGamma <- gammaFitEM(exprs(example.lumiMethy)[,1], plotMode=TRUE)
```

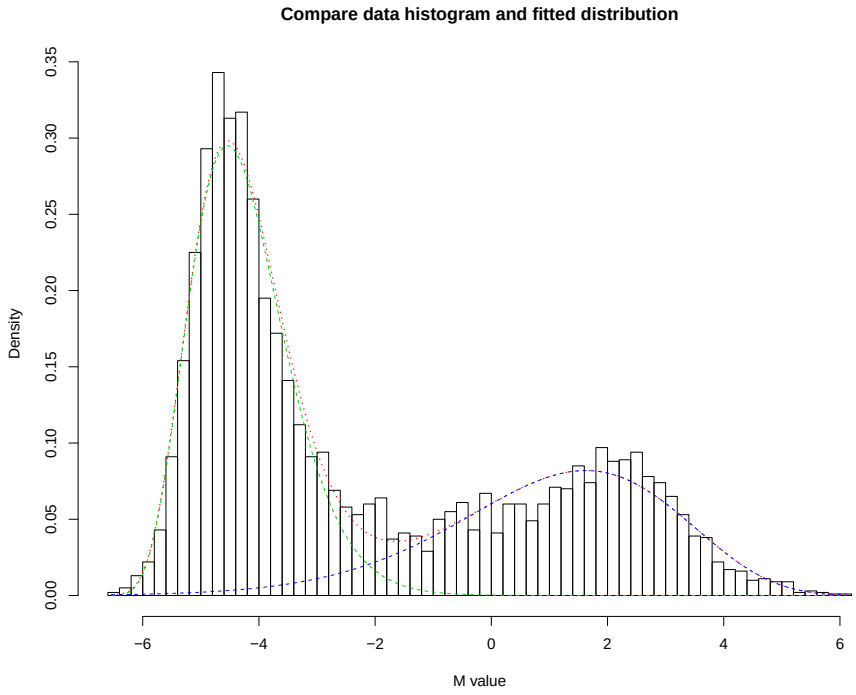

Figure 32: Compare the fitted two component gamma mixture model with the histogram of the first sample of example.lumiMethy

(methylation posterior probability > methylation threshold), or "Margin". By default, both unmethylation and methylation probability thresholds are 0.95. Following is the summary of methylation status of the first sample in example.lumiMethy dataset:

```
> ## estimate the methylation status based on the results of gammaFitEM
> methyCall <- methylationCall(fittedGamma)
> table(methyCall)

methyCall
 Margin   Methy Unmethy
    806    1648    2546
```

The inputs of both `gammaFitEM` and `methylationCall` are a M-value vector of a particular sample. Function `lumiMethyStatus` is a wrap function of `methylationCall` to process a *MethyLumiM* object. It returns a methylation status matrix of the same dimension as the input *MethyLumiM* object. The methylation probability matrix is kept as the "probability" attributes of the result. Following is an example:

```
> ## estimate the methylation status of a LumiMethyM object
> methyCall <- lumiMethyStatus(example.lumiMethy[,1:4])
> head(methyCall)

           Treat1    Treat2    Treat3    Treat4
cg00002426 "Margin"  "Margin"  "Methy"   "Methy"
cg00012386 "Unmethy" "Unmethy" "Unmethy" "Unmethy"
cg00013618 "Methy"   "Methy"   "Methy"   "Methy"
cg00014837 "Methy"   "Methy"   "Methy"   "Methy"
cg00020533 "Methy"   "Methy"   "Methy"   "Methy"
cg00021527 "Unmethy" "Unmethy" "Unmethy" "Unmethy"

> ## retrieve the methylation probability matrix
> methyProb <- attr(methyCall, "probability")
> head(methyProb)

               Treat1      Treat2      Treat3      Treat4
cg00002426 0.553664242 0.910504727 0.987548469 0.992710117
cg00012386 0.005614807 0.005282840 0.004704675 0.004626131
cg00013618 0.999981923 0.999963027 0.999983664 0.999988660
cg00014837 0.999965767 0.999363128 0.999738191 0.999680610
cg00020533 0.999999231 0.999990911 0.999657947 0.999927978
cg00021527 0.008324763 0.005820892 0.004675040 0.004762027
```

3.10 Use user provided preprocessing functions

For convenience to the users, the user can specify their own preprocessing function when call `lumiMethyB`, `lumiMethyC` and `lumiMethyN`. The input and output of the user provided function should be a intensity matrix (pool of methylated and unmethylated probe intensities). Following are some examples of using user defined preprocessing functions.

```
> ## suppose "userB" is a user defined background adjustment method
>  lumiMethy.b.adj <- lumiMethyB(example.lumiMethy, method=userB, separateColor=TRUE) # no
> ## suppose "userC" is a user defined color balance adjustment method
>  lumiMethy.c.adj <- lumiMethyC(example.lumiMethy, method=userC, separateColor=TRUE) # no
> ## suppose "userN" is a user defined probe level normalization method
>  lumiMethy.c.n <- lumiMethyN(lumiMethy.c.adj, method= userN, separateColor=TRUE) # not R
```

3.11 Options to separately process each color channel

Functions `lumiMethyB`, `lumiMethyC` and `lumiMethyN` also has the option to separately process in each color channel. The presumption is that the data (*MethyLumiM* object) should include the color channel column (with column name "COLOR_CHANNEL") in the featureData and the green and red colors are represented as "Red" and "Grn". If the GenomeStudio (BeadStudio) output the annotation information together with the data (we recommend it), the `lumiMethyR` function will automatically add the annotation information, including color channel information, to the featureData slot. If the data does not include color channel information, users can add it use function `addAnnotationInfo`.

```
> ## retrieve the featureData information
> ff <- pData(featureData(example.lumiMethy))
> ## show the color channel information
> head(ff)
```

```
           CHR COLOR_CHANNEL
cg00002426   3           Red
cg00012386   1           Red
cg00013618  22           Grn
cg00014837  12           Red
cg00020533   6           Red
cg00021527  17           Red
```

```
>
> ## add user provided color channel information if it is not existed in the featureData
> # example.lumiMethy <- addAnnotationInfo(example.lumiMethy, lib="IlluminaHumanMethylatio
```

Here are examples of separately processing each color channel:

```
> ## suppose "userB" is a user defined background adjustment method
>  lumiMethy.b.adj <- lumiMethyB(example.lumiMethy,  separateColor=TRUE)
> ## suppose "userC" is a user defined color balance adjustment method
>  lumiMethy.c.adj <- lumiMethyC(example.lumiMethy,  separateColor=TRUE)
> ## suppose "userN" is a user defined probe level normalization method
>  lumiMethy.c.n <- lumiMethyN(lumiMethy.c.adj,  separateColor=TRUE)
```

3.12 About the detection p-value of a CpG site

The detection p-values of methylation arrays reflect the strength of DNA hybridization over the background (comparing the CpG-intensity with the intensities of negative control probes). Because all genomic DNA is expect existing

in a normal diploid sample, almost all detection p-values are significant. Non-significant detection p-values usually means bad probe design, bad hybridization or possible chromosome abnormalities (like mutations, indels) at the probe matching locations. Given one sample in the titration data set as an example, the 99 percentile of the -log10(detection p-value) is 28.3, which is extraordinarily significant p-value. And there are only 27 probes with detection p-value worse than 0.0001. This is in great contrast to expression microarrays, where up to 40 percent or more of the genes might not be expressed in a given sample, and filtering based on detection p-values can dramatically reduce the false-positive rate for the expression data. However, removing the bad quality CpG measurements is still an important step during preprocessing of Illumina methylation data. Same as Illumina Expression data, we can use function `detectionCall` to estimate the detection call. Please check the help of `detectionCall` for more details of its usage.

```
> ## Estimate the detection call of a CpG site
> presentCount <- detectionCall(example.lumiMethy)
```

4 Gene annotation

The annotation package, *IlluminaHumanMethylation27k.db*, of the HumanMethylation27 BeadChip can be downloaded from Bioconductor. The usage of this package is the same as the expression microarrays.

5 Performance comparison

To be added in the near future ...

6 A use case: from raw data to functional analysis

6.1 Preprocessing the Illumina Infinium Methylation microarray data

```
> library(lumi)
> ## specify the file name
> # fileName <- 'Example_Illumina_Methylation_profile.txt'
> ## load the data
> # example.lumiMethy <- lumiMethyR(fileName, lib="IlluminaHumanMethylation27k.db")
>
> ## Quality and color balance assessment
> data(example.lumiMethy)
> ## summary of the example data
> example.lumiMethy
> ## preprocessing and quality control after normalization
> plotColorBias1D(example.lumiMethy, channel='sum')
> boxplotColorBias(example.lumiMethy, channel='sum')
> ## select interested sample to further check color balance in 2D scatter plot
```

```
> plotColorBias2D(example.lumiMethy, selSample=1)
> ## Color balance adjustment between two color channels
> lumiMethy.c.adj <- lumiMethyC(example.lumiMethy)
> ## Check color balance after color balance adjustment
> boxplotColorBias(lumiMethy.c.adj, channel='sum')
> ## Background adjustment is skipped because the SSN normalization includes background ad
>
> ## Normalization
> ## Perform SSN normalization based on color balance adjusted data
> lumiMethy.c.ssn <- lumiMethyN(lumiMethy.c.adj, method='ssn')
> ## Or we can perform quantile normalization based on color balance adjusted data
> # lumiMethy.c.q <- lumiMethyN(lumiMethy.c.adj, method='quantile')
>
> ## plot the density of M-values after SSN normalization
> density(lumiMethy.c.ssn, main="Density plot of M-value after SSN normalization")
> ## comparing with the density of M-values before normalization
> density(example.lumiMethy, main="Density plot of M-value of the raw data")
> ## output the normlized M-value as a Tab-separated text file
> write.exprs(lumiMethy.c.ssn, file='processedMethylationExampleData.txt')
```

6.2 Identify differentially methylated genes and Functional analysis

The identification of differentially methylated genes and Functional analysis are the same as expression data, shown in the lumi.pdf. Here we will not repeat this part.

6.3 GEO submission of the methylation data

The submission of methylation data is similar with expression microarray data. The submission file will be in the SOFT format. So users can submit all the data in a batch. We also use `produceGEOSampleInfoTemplate` to produce a template of sample information with some default fillings. Some fields have been filed in with default settings. Users should fill in or modify the detailed sample descriptions by referring some previous submissions. No blank fields are allowed. Users are also required to fill in the "Sample_platform_id" by checking information of the GEO Illumina platform. `produceMethylationGEOSubmissionFile` is the main function of produce GEO submission file including both normalized and raw methylation data information in the SOFT format. By default, the R objects, methyLumiM, methyLumiM.raw and sampleInfo, will be saved in a file named 'supplementaryData.Rdata'. Users can include this R data file as a GEO supplementary data file.

```
## Produce the sample information template
> produceGEOSampleInfoTemplate(methyLumiM, fileName = "GEOsampleInfo.txt")
## After editing the 'GEOsampleInfo.txt' by filling in sample information
> produceMethylationGEOSubmissionFile(methyLumiM, methyLumiM.raw, sampleInfo='GEOsampleInf
```

7 Session Info

```
> toLatex(sessionInfo())
```

- R version 2.15.1 (2012-06-22), `i386-pc-mingw32`
- Locale: `LC_COLLATE=C`, `LC_CTYPE=English_United States.1252`, `LC_MONETARY=English_United States.1252`, `LC_NUMERIC=C`, `LC_TIME=English_United States.1252`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: AnnotationDbi 1.20.0, Biobase 2.18.0, BiocGenerics 0.4.0, DBI 0.2-5, RColorBrewer 1.0-5, RSQLite 0.11.2, annotate 1.35.3, genefilter 1.40.0, limma 3.14.0, lumi 2.10.0, lumiBarnes 1.3.8, lumiHumanAll.db 1.18.0, lumiHumanIDMapping 1.10.0, nleqslv 1.9.4, org.Hs.eg.db 2.8.0, vsn 3.26.0
- Loaded via a namespace (and not attached): BiocInstaller 1.8.0, IRanges 1.16.0, KernSmooth 2.23-8, MASS 7.3-21, Matrix 1.0-9, XML 3.9-4.1, affy 1.36.0, affyio 1.26.0, colorspace 1.1-1, grid 2.15.1, lattice 0.20-10, methylumi 2.4.0, mgcv 1.7-21, nlme 3.1-104, parallel 2.15.1, preprocessCore 1.20.0, splines 2.15.1, stats4 2.15.1, survival 2.36-14, tools 2.15.1, xtable 1.7-0, zlibbioc 1.4.0

8 Acknowledgement

We appreciate Dr. Sean Davis from NIH developed methylumi package and built IlluminaHumanMethylation27k.db package for us!

This work was supported in part by the NIH award 1RC1ES018461-01 (to Dr. Lifang Hou), P30CA060553 and UL1RR025741. We would like to thank Lifang Hou for supporting this work, Xiao Zhang for preparing the titration samples, Nadereh Jafari and Vivi Frangidakis for conducting the Illumina BeadChip experiments.

9 References

1. Du P, Kibbe WA and Lin SM: "lumi: a Bioconductor package for processing Illumina microarray" Bioinformatics 2008 24(13):1547-1548

2. Du P, Zhang X, Huang CC, Jafari N, Kibbe WA, Hou L, and Lin SM: "Comparison of Beta-value and M-value methods for quantifying methylation levels by microarray analysis", BMC Bioinformatics 2010, 11:587