minfi and **shinyMethyl**: a winning pair of R-packages for the analysis of methylation data

shinyMethyl's author: Jean-Philippe Fortin

minfi's authors: Kasper D. Hansen Martin J. Aryee Rafael Irizarry

Tutorial BioC 2013

1 Introduction

The goal of the tutorial is to introduce the Bioconductor community to the `minfi` package [1], an R-package that provides tools for analyzing Illumina's Methylation array, and `shinyMethyl` [2], a complementary `shiny` application to interactively explore methylation data.

We will go through an almost complete list of the `minfi`'s functions, starting by reading the input data (IDAT files) of a small dataset, and ending with a list of differentially methylated loci. We will cover quality control assessments, within-array and between-array normalization, gender prediction, differential analysis of unique locations and bump hunting. But before that, let's start by a quick review of the 450k methylation array.

450k Array design and terminology

Each sample is measured on a single array, in two different color channels (red and green). As the name of the platform indicates, each array measures more than 450,000 CpG positions. For each CpG, we have two measurements: a *methylated* intensity and an *unmethylated* intensity. Depending on the probe design, the signals are reported in different colors:

For **Type I** design, both signals are measured in the same color: one probe for the methylated signal and one probe for the unmethylated signal.

For **Type II** design, only one probe is used. The **Green** intensity measures the *methylated signal*, and the **Red** intensity measures the *unmethylated signal*.

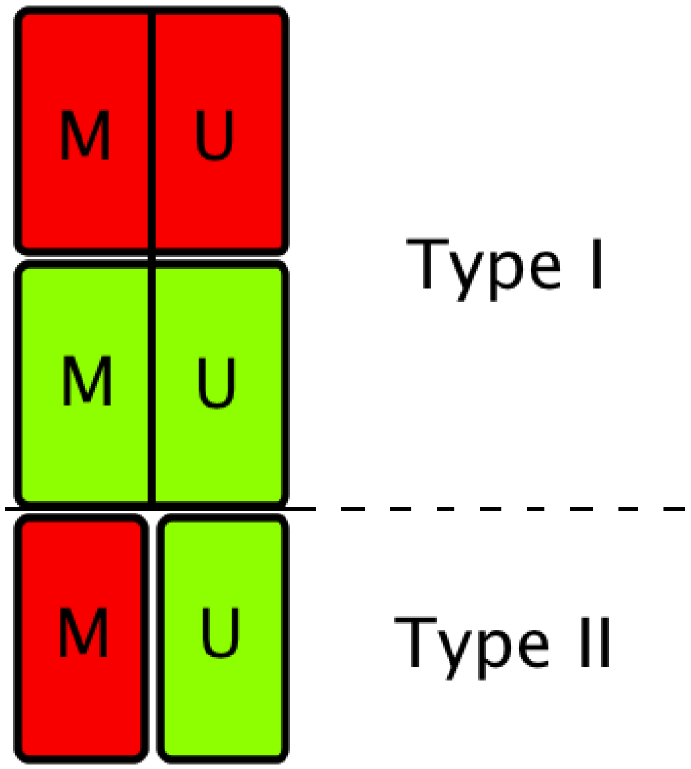


Figure 1: Probe design of the 450k array

Some definitions

Beta-value:

$$\beta = \frac{M}{M + U + 100}$$

where M and U denote the methylated and unmethylated signals respectively.

M-Value:

$$Mval = \log\left(\frac{M}{U}\right)$$

DMP: Differentially methylated position: single genomic position that has a different methylated level in two different groups of samples (or conditions)

DMR: Differentially methylated region: when consecutive genomic locations are differentially methylated in the same direction.

Array: One sample

Slide: Physical slide containing 12 arrays (6×2 grid)

Plate: Physical plate containing at most 8 slides (96 arrays). For this tutorial, we use **batch** and plate interchangeably.

2 Reading Data

The starting point of `minfi` is the reading of the .IDAT file with the built-in function`read.450k.exp`. Several options are available: the user can specify the sample filenames to be read in along with the directory path, or can specify the directory that contains the files. In the latter case, all the files with the extension .IDAT located in the directory will be loaded into R. The user can also read in a sample sheet, and then use the sample sheet to load the data into a `RGChannelSet`. For more information, see the `minfi` vignette. Here, we will load the dataset containing 6 samples from the `minfiData` package using the sample sheet provided within the package:

```
> require(minfi)
> require(minfiData)
```

```
> baseDir <- system.file("extdata",package="minfiData")
> targets <- read.450k.sheet(baseDir)
```

```
[read.450k.sheet] Found the following CSV files:
[1] "/Users/Jean-Philippe/Library/R/3.0/library/minfiData/extdata/SampleSheet.csv"
```

```
> targets

  Sample_Name Sample_Well Sample_Plate Sample_Group Pool_ID person age sex
1    GroupA_3          H5           NA       GroupA      NA    id3  83   M
2    GroupA_2          D5           NA       GroupA      NA    id2  58   F
3    GroupB_3          C6           NA       GroupB      NA    id3  83   M
4    GroupB_1          F7           NA       GroupB      NA    id1  75   F
5    GroupA_1          G7           NA       GroupA      NA    id1  75   F
6    GroupB_2          H7           NA       GroupB      NA    id2  58   F
  status  Array      Slide
1 normal R02C02 5723646052
2 normal R04C01 5723646052
3 cancer R05C02 5723646052
4 cancer R04C02 5723646053
5 normal R05C02 5723646053
6 cancer R06C02 5723646053
                                                                                             Basename
1 /Users/Jean-Philippe/Library/R/3.0/library/minfiData/extdata/5723646052/5723646052_R02C02
2 /Users/Jean-Philippe/Library/R/3.0/library/minfiData/extdata/5723646052/5723646052_R04C01
3 /Users/Jean-Philippe/Library/R/3.0/library/minfiData/extdata/5723646052/5723646052_R05C02
4 /Users/Jean-Philippe/Library/R/3.0/library/minfiData/extdata/5723646053/5723646053_R04C02
5 /Users/Jean-Philippe/Library/R/3.0/library/minfiData/extdata/5723646053/5723646053_R05C02
6 /Users/Jean-Philippe/Library/R/3.0/library/minfiData/extdata/5723646053/5723646053_R06C02

> RGSet <- read.450k.exp(base = baseDir, targets = targets)
```

RGSet RGSet is a `RGChannelSet` object, which contains the raw data from the IDAT files: green intensities and red intensities. Let's extract the phenotype data:

```
> phenoData <- pData(RGSet)
> phenoData[,1:6]

                  Sample_Name Sample_Well Sample_Plate Sample_Group Pool_ID
5723646052_R02C02    GroupA_3          H5           NA       GroupA      NA
5723646052_R04C01    GroupA_2          D5           NA       GroupA      NA
5723646052_R05C02    GroupB_3          C6           NA       GroupB      NA
5723646053_R04C02    GroupB_1          F7           NA       GroupB      NA
5723646053_R05C02    GroupA_1          G7           NA       GroupA      NA
5723646053_R06C02    GroupB_2          H7           NA       GroupB      NA
                  person
5723646052_R02C02    id3
5723646052_R04C01    id2
5723646052_R05C02    id3
5723646053_R04C02    id1
5723646053_R05C02    id1
5723646053_R06C02    id2
```

This is nothing else than the sample information contained in the sample sheet. The `RGChannelSet` contains also a manifest object of the 450k array describing the design of the array:

```
> manifest <- getManifest(RGSet)
> manifest
```

```
IlluminaMethylationManifest object
Annotation
  array: IlluminaHumanMethylation450k
Number of type I probes: 135476
Number of type II probes: 350036
Number of control probes: 850
Number of SNP type I probes: 25
Number of SNP type II probes: 40
```

```
> head(getProbeInfo(manifest))
```

```
DataFrame with 6 rows and 8 columns
         Name    AddressA    AddressB       Color       NextBase
  <character> <character> <character> <character> <DNAStringSet>
1  cg00050873    32735311    31717405         Red              A
2  cg00212031    29674443    38703326         Red              T
3  cg00213748    30703409    36767301         Red              A
4  cg00214611    69792329    46723459         Red              A
5  cg00455876    27653438    69732350         Red              A
6  cg01707559    45652402    64689504         Red              A
                                           ProbeSeqA
                                      <DNAStringSet>
1 ACAAAAAAACAACACACAACTATAATAATTTTTAAAATAAATAAACCCCA
2 CCCAATTAACCACAAAAACTAAACAAATTATACAATCAAAAAAACATACA
3 TTTTAACACCTAACACCATTTTAACAATAAAAATTCTACAAAAAAAAACA
4 CTAACTTCCAAACCACACTTTATATACTAAACTACAATATAACACAAACA
5 AACTCTAAACTACCCAACACAAACTCCAAAAACTTCTCAAAAAAAACTCA
6 ACAAATTAAAAACACTAAAACAAACACAACAACTACAACAACAAAAAACA
                                           ProbeSeqB      nCpG
                                      <DNAStringSet> <integer>
1 ACGAAAAAAACAACGCACAACTATAATAATTTTTAAAATAAATAAACCCCG         2
2 CCCAATTAACCGCAAAAACTAAACAAATTATACGATCGAAAAAAACGTACG         4
3 TTTTAACGCCTAACACCGTTTTAACGATAAAAATTCTACAAAAAAAAACG         3
4 CTAACTTCCGAACCGCGCTTTATATACTAAACTACAATATAACGCGAACG         5
5 AACTCTAAACTACCCGACACAAACTCCAAAAACTTCTCGAAAAAAAACTCG         2
6 GCGAATTAAAAACACTAAAACGAACGCGACGACTACAACGACAAAAAACG         6
```

Let’s extract the names of the probes of Type I design:

```
> typeIProbes <- getProbeInfo(manifest, type = "I")$Name
> head(typeIProbes)
```

```
[1] "cg00050873" "cg00212031" "cg00213748" "cg00214611" "cg00455876"
[6] "cg01707559"
```

The 450 array contains also 65 SNP probes that do not interrogate methylation. They can be used as control probes to check whether or not samples have been mixed up:

```
> snpProbesI <- getProbeInfo(manifest, type = "SnpI")$Name
> snpProbesII <- getProbeInfo(manifest, type = "SnpI")$Name
> head(snpProbesI)
```

```
[1] "rs10796216" "rs715359"   "rs1040870"  "rs10936224" "rs213028"
[6] "rs2385226"
```

3 Quality control

`minfi` provides several functions and diagnostic plots to assess quality of the methylation samples. As a starting point, we suggest to look at the function `detectionP()` which identifies failed positions defined as both the methylated and unmethylated channel reporting background signal levels:

```
> detP <- detectionP(RGSet)
> failed <- detP > 0.01
> head(failed, n=3)
```

```
           5723646052_R02C02 5723646052_R04C01 5723646052_R05C02
cg00050873             FALSE              TRUE             FALSE
cg00212031             FALSE              TRUE             FALSE
cg00213748             FALSE              TRUE              TRUE
           5723646053_R04C02 5723646053_R05C02 5723646053_R06C02
cg00050873              TRUE              TRUE              TRUE
cg00212031              TRUE              TRUE              TRUE
cg00213748              TRUE              TRUE              TRUE
```

To see the fraction of failed positions per sample:

```
> colMeans(failed)
```

```
5723646052_R02C02 5723646052_R04C01 5723646052_R05C02 5723646053_R04C02
     0.0009927664      0.0032769530      0.0092397304      0.0042244064
5723646053_R05C02 5723646053_R06C02
     0.0034561453      0.0348374499
```

and to see how many positions failed in $> 50\%$ of the samples:

```
> sum(rowMeans(failed)>0.5)
```

```
[1] 1047
```

A simple way to quickly check if a sample failed is to look at the log median intensity in both the methylated and unmethylated channels. When plotting the U channel against the M channel, it has been observed that good samples cluster together, while failed samples tend to separate and to have lower median intensities.
But wait, so far we only have green and red intensities. We need the methylated and unmethylated signals; the function `preprocessRaw` is done for that. It takes as input a `RGChannelSet`, convert the red and green intensities to methylated and unmethylated signals according to the probe design stored in the manifest object, and returns the converted signals in a new object of class `MethylSet`.

```
> MSet <- preprocessRaw(RGSet)
> MSet
```

```
MethylSet (storageMode: lockedEnvironment)
assayData: 485512 features, 6 samples
  element names: Meth, Unmeth
phenoData
  sampleNames: 5723646052_R02C02 5723646052_R04C01 ...
    5723646053_R06C02 (6 total)
  varLabels: Sample_Name Sample_Well ... filenames (13 total)
  varMetadata: labelDescription
Annotation
  array: IlluminaHumanMethylation450k
  annotation: ilmn.v1.2
Preprocessing
  Method: Raw (no normalization or bg correction)
  minfi version: 1.7.8
  Manifest version: 0.4.0
```

To access the methylated and unmethylated intensities:

```
> head(getMeth(MSet)[,1:3])
```

```
           5723646052_R02C02 5723646052_R04C01 5723646052_R05C02
cg00050873             22041               588             20505
cg00212031               679               569               439
cg00213748              1620               421               707
cg00214611               449               614               343
cg00455876              5921               398              3257
cg01707559              1238               646               637
```

```
> head(getUnmeth(MSet)[,1:3])
```

```
           5723646052_R02C02 5723646052_R04C01 5723646052_R05C02
cg00050873              1945               433              1012
cg00212031              6567               300              2689
cg00213748               384               461               295
```

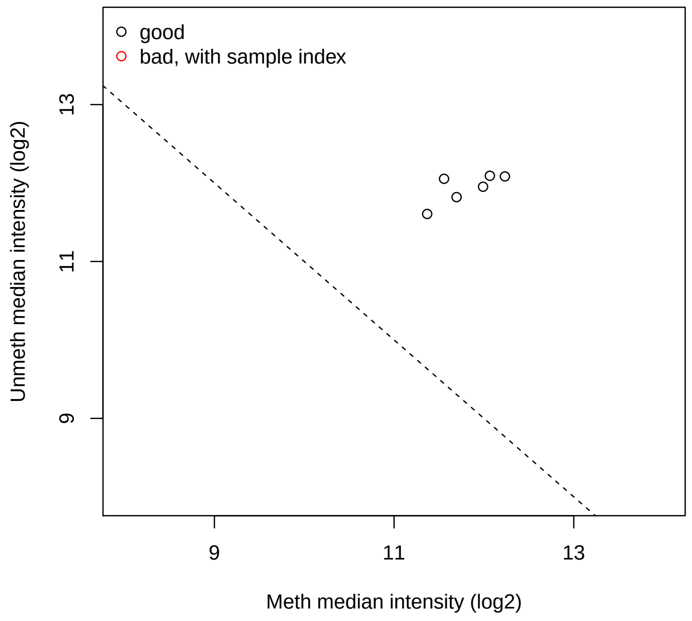
```
cg00214611                 4869                183                1655
cg00455876                 1655                792                1060
cg01707559                12227               1009                7414
```

We will talk more about the `MethylSet` later. The functions `getQC` and `plotQC` are designed to extract the quality control information from the `MethylSet`:

```
> qc <- getQC(MSet)
> head(qc)
```

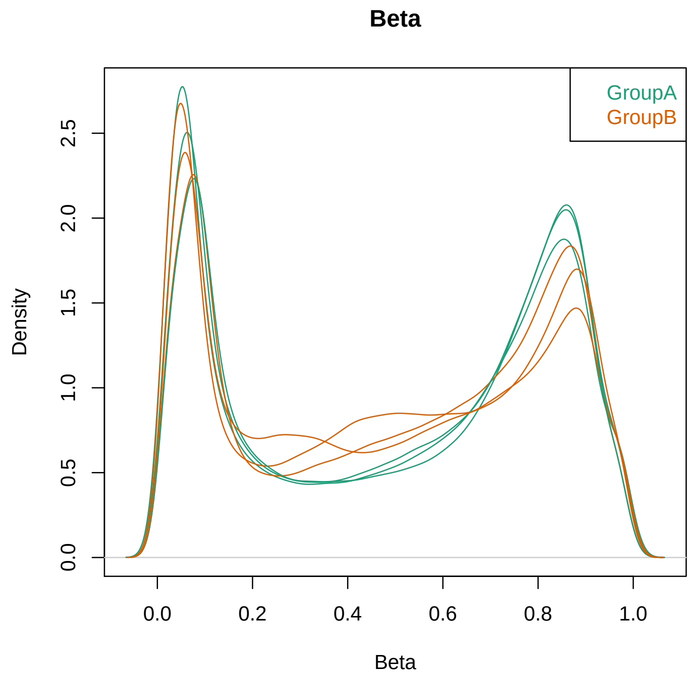
```
DataFrame with 6 rows and 2 columns
                          mMed      uMed
                     <numeric> <numeric>
5723646052_R02C02  11.69566  11.82058
5723646052_R04C01  11.99046  11.95274
5723646052_R05C02  11.55603  12.05393
5723646053_R04C02  12.06609  12.09276
5723646053_R05C02  12.23332  12.08448
5723646053_R06C02  11.36851  11.60594
```

```
> plotQC(qc)
```



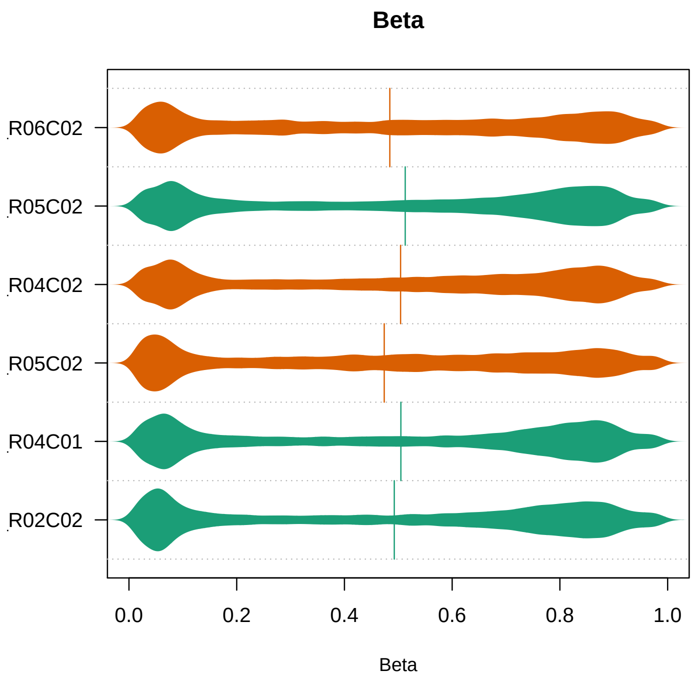
To further explore the quality of the samples, it is useful to look at the Beta value densities of the samples, with the option to color the densities by group (e.g. processing data:

```
> densityPlot(RGSet, sampGroups = phenoData$Sample_Group, main= "Beta", xlab = "Beta")
```



or density bean plots:

```
> densityBeanPlot(RGSet, sampGroups = phenoData$Sample_Group)
```



Finally, the 450k array contains several internal control probes that can be used to assess the quality control of different sample preparation steps (bisulfite conversion, hybridization, etc.). The values of these control probes can be plotted by using the function `controlStripPlot` and by specifying the control probe type:

```
> controlStripPlot(RGSet, controls="BISULFITE CONVERSION II")
```

All the plots above can be exported into a pdf file in one step using the function `qcReport`:

```
> qcReport(RGSet, pdf= "qcReport.pdf")
```

When the number of samples becomes large, it becomes difficult to assess quality control using the generated plots above. This is where `shinyMethyl` comes into play and complements `minfi`.

4 Starting with shinyMethyl

In the current version of shinyMethyl, the `RGChannelSet` is the class object from which summary statistics are extracted for visualization of the data using the function `extractFromRGSet450k()`:

```
> source("/data/minfiLab/extractFromRGSet450k.R")
> extractedDataMinfiLab <- extractFromRGSet450k(RGSet,
+  file = "extractedDataMinfiLab.Rda")
```

The file containing the necessary information for `shinyMethyl` is saved in the current working directory. The function reduces a `RGChannelSet` object of a large size to a small object (a few MB in size). The extracted object can be loaded into a personal laptop, and `shinyMethyl` can run from there. It gives users the chance to explore their genomic data comfortably, without any delay due to computational complexity.

There are 3 simple steps to launch `shinyMethyl`:

1. Extract the data summaries from an `RGChannelSet` using `extractFrom-RGSet450k()`
2. Drag the extracted .Rda file in the `shinyMethyl` directory
3. Lauch `shinyMethyl` in R by using the function `runApp("./shinyMethyl")`

For today's tutorial, we have previously extracted the following data from another dataset containing 260 samples (HNSCC data from TCGA [3]). The file is called `extractedData.Rd` and is already located in `/home/data/minfiLab/shinyMethyl`. Please DO NOT modify it. For this tutorial, we need an extra function to make the server work on the Amazon cluster:

```
> source("/data/minfiLab/myRunApp.R")
```

Now we are ready to launch `shinyMethyl`:

```
> myRunApp(appDir = "/data/minfiLab/shinyMethyl")
```

5 Preprocessing and normalization

Different preprocessing steps options are available in `minfi`

`preprocessRaw()`

As seen before, it converts a `RGChannelSet` to a `MethylSet` by converting the Red and Green channels into a matrix of methylated signals and a matrix of unmethylated signals. No "normalization" is performed.

preprocessIllumina()

Convert a `RGChannelSet` to a `MethylSet` by implementing the preprocessing choices as available in Genome Studio: background subtraction and control normalization. Both of them are optional and turning them off is equivalent to raw preprocessing (`preprocessRaw`):

```
> MSet.illumina <- preprocessIllumina(RGSet, bg.correct = TRUE,
+                                 normalize = "controls")
```

preprocessSWAN()

Perform Subset-quantile within array normalization (SWAN) [4], a within-array normalization correction for the technical differences between the Type I and Type II array designs. The algorithm matches the Beta-value distributions of the Type I and Type II probes by applying a within-array quantile normalization separately for different subsets of probes (divided by CpG content). The input of SWAN is a `MethylSet`, and the function returns a `MethylSet` as well. If an `RGChannelSet` is provided instead, the function will first call `preprocessRaw` on the `RGChannelSet`, and then apply the SWAN normalization. We recommend setting a seed (using `set.seed`) before using `preprocessSWAN` to ensure that the normalized intensities will be reproducible.

```
> MSet.swan <- preprocessSWAN(RGSet)
```

Before discussing the main preprocessing function of `minfi`, we need to discuss in more depth the different storage objects of `minfi`.

6 MethylSet and RatioSet

As said before, the `MethylSet` contains the methylated and unmethylated signals, but no methylation level statistic such as Beta-values of M-values. We can extract the Beta-values and the M-values from a `MethylSet` using `getBeta` and `getM`:

```
> getBeta(MSet, type = "Illumina")[1:4,1:3]

           5723646052_R02C02 5723646052_R04C01 5723646052_R05C02
cg00050873        0.91509591         0.5245317         0.9485590
cg00212031        0.09243126         0.5872033         0.1359975
cg00213748        0.76996198         0.4287169         0.6415608
cg00214611        0.08287191         0.6845039         0.1634890
```

```
> getM(MSet)[1:4,1:3]

           5723646052_R02C02 5723646052_R04C01 5723646052_R05C02
cg00050873          3.502348         0.4414491          4.340695
cg00212031         -3.273751         0.9234662         -2.614777
cg00213748          2.076816        -0.1309465          1.260995
cg00214611         -3.438838         1.7463950         -2.270551
```

Note that the option `type = "Illumina"` means that a constant of 100 is added in the denominator. Similarly, as seen before, `getMeth` and `getUnmeth` return methylated and unmethylated intensities respectively .

The `RatioSet` class is designed to store Beta values and/or M-values instead of the methylated and unmethylated signals. An optional copy number matrix, `CN`, can be also stored. Mapping a `MethylSet` to a `RatioSet` is irreversible, i.e.

one cannot technically retrieve the raw methylated and unmethylated signals from a `RatioSet`. A `RatioSet` can be created with the function `ratioConvert`:

```
> ratioSet <- ratioConvert(MSet, what = "both", keepCN = TRUE)
> ratioSet

RatioSet (storageMode: lockedEnvironment)
assayData: 485512 features, 6 samples
  element names: Beta, CN, M
phenoData
  sampleNames: 5723646052_R02C02 5723646052_R04C01 ...
    5723646053_R06C02 (6 total)
  varLabels: Sample_Name Sample_Well ... filenames (13 total)
  varMetadata: labelDescription
Annotation
  array: IlluminaHumanMethylation450k
  annotation: ilmn.v1.2
Preprocessing
  Method: Raw (no normalization or bg correction)
  minfi version: 1.7.8
  Manifest version: 0.4.0
```

The function `getBeta`, `getM` and `getCN` return respectively the beta-values matrix, m-values matrix and a matrix of copy number estimates.

7 Map to Genome

The `RatioSet` is the object containing the final Beta-values and/or M-values for further analysis. It is also possible to associate genomic coordinates to the probes by using the function `mapToGenome`, which will transform the `RatioSet` object to a `GenomicRatioSet` object (class holding M or/and Beta values together with associated genomic coordinates). It is possible to choose the genome build version, and to merge the manifest object with the genomic locations by letting `mergeManifest` to be `TRUE`.

```
> gRatioSet <- mapToGenome(ratioSet, genomeBuild = "hg19", mergeManifest = TRUE)
> gRatioSet

class: GenomicRatioSet
dim: 485512 6
exptData(0):
assays(3): Beta M CN
rownames(485512): cg13869341 cg14008030 ... cg08265308 cg14273923
rowData metadata column names(9): Name AddressA ... nCpG Type
colnames(6): 5723646052_R02C02 5723646052_R04C01 ... 5723646053_R05C02
  5723646053_R06C02
colData names(13): Sample_Name Sample_Well ... Basename filenames
```

```
Annotation
  array: IlluminaHumanMethylation450k
  annotation: ilmn.v1.2
Preprocessing
  Method: Raw (no normalization or bg correction)
  minfi version: 1.7.8
  Manifest version: 0.4.0
```

```
> showClass("GenomicRatioSet")
```

```
Class "GenomicRatioSet" [package "minfi"]

Slots:

Name:            preprocessMethod                 annotation
Class:                  character                  character

Name:                   exptData                    rowData
Class:                SimpleList GenomicRangesORGRangesList

Name:                    colData                     assays
Class:                 DataFrame                     Assays

Extends: "SummarizedExperiment"
```

Note that the `GenomicRatioSet` extends the class `SummarizedExperiment`. Here are the main accessors functions to access the data:

```
> getBeta(gRatioSet)
> getM(gRatioSet)
> getCN(gRatioSet)
```

```
> sampleNames <- sampleNames(gRatioSet)
> probeNames <- featureNames(gRatioSet)
> pheno <- pData(gRatioSet)
```

To extract the genomic locations:

```
> gRanges <- rowData(gRatioSet)
> head(gRanges, n= 3)
```

```
GRanges with 3 ranges and 9 metadata columns:
             seqnames         ranges strand |        Name    AddressA
                <Rle>      <IRanges>  <Rle> | <character> <character>
  cg13869341     chr1 [15865, 15865]      * |  cg13869341    62703328
  cg14008030     chr1 [18827, 18827]      * |  cg14008030    27651330
  cg12045430     chr1 [29407, 29407]      * |  cg12045430    25703424
                AddressB       Color       NextBase
```

```
           <character> <character> <DNAStringSet>
cg13869341    16661461         Red             A
cg14008030        <NA>        <NA>              
cg12045430    34666387         Red             A
                                                   ProbeSeqA
                                              <DNAStringSet>
cg13869341 CCAATAACTAACCACTCTACTAAAATCCATCCACCAAACTAAAAACATCA
cg14008030 ACTCRAAATTTACTCAATAAACCRTTCAATATATACAAAAACAATTCCCC
cg12045430 AAAAAAAACACAATAAAAAACAAACAACAACATTAAAACCCAAAAACACA
                                                   ProbeSeqB      nCpG
                                              <DNAStringSet> <integer>
cg13869341 CCGATAACTAACCACTCTACTAAAATCCATCCGCCAAACTAAAAACATCG         2
cg14008030                                                           2
cg12045430 GAAAAAAACGCAATAAAAAACGAACGACGACGTTAAAACCCGAAAACGCG         7
                  Type
           <character>
cg13869341           I
cg14008030          II
cg12045430           I
---
seqlengths:
  chr1  chr2  chr3  chr4  chr5  chr6 ... chr19 chr20 chr21 chr22  chrX  chrY
    NA    NA    NA    NA    NA    NA ...    NA    NA    NA    NA    NA    NA
```

8 Sex prediction

By looking at the median total intensity of the X chromosome-mapped probes, denoted $med(X)$, and the median total intensity of the Y-chromosome-mapped probes, denoted $med(Y)$, one can observe two different clusters of points corresponding to which gender the samples belong to. To predict the gender, `minfi` separate the points by using a cutoff on $\log_2 med(Y) - \log_2 med(Y)$. The default cutoff is -2. Since the algorithm needs to map probes to the X-chr and to the Y-chr, the input of the function `getSex()` needs to be a `GenomicMethylSet` or a `GenomicRatioSet`.

```
> predictedSex <- getSex(gRatioSet, cutoff = -2)$predictedSex
> head(predictedSex)
```

```
[1] "M" "F" "M" "F" "F" "F"
```

To choose the cutoff to separate the two gender clusters, one can plot $med(Y)$ against $med(Y)$ with the function `plotSex`:

```
> plotSex(getSex(gRatioSet, cutoff = -2))
```

Furthermore, `shinyMethyl` has a panel which allow the user to choose interactively the cutoff.

Remark: the function does not handle datasets with only females or only males

9 preprocessQuantile()

The function `preprocessQuantile()` is a useful one-step command to convert directly a basic `RGChannelSet` to a `GenomicRatioSet`, i.e. convert red and green intensities to final beta values, M values, copy number estimates and genomic locations. It also suggests a between-array normalization called "Stratified Subset Quantile Normalization". The function has several internal steps:

1) Fix outliers of both the methylated and unmethylated channels (optional)
2) Remove bad samples using quality control (optional)
3) Map to the genome using `mapToGenome`
4) Perform stratified subset quantile normalization (optional)
5) Predict the sex if not provided using the function `getSex`

We have covered all steps mentioned above except the stratified subset quantile normalization (SSQN). What SSQN does is

- Perform the quantile normalization on the methylated and unmethylated channels separately
- The quantile normalization is applied separately on 3 different subsets of probes: autosomal probes, X-chr probes and Y-chr probes. For the X-chr and Y-chr probes, samples are quantile normalized separately by sex.
- If `stratified = TRUE`, the subset of autosomal probes is further divided into three subsets: probes mapping to CpG islands, probes mapping to CpG shores, and the rest of the probes. Quantile normalization is then applied separately for each subset of probes

```
> gRatioSet.quantile <- preprocessQuantile(RGSet, fixOutliers = TRUE,
+ removeBadSamples = TRUE, badSampleCutoff = 10.5,
+ quantileNormalize = TRUE, stratified = TRUE,
+ mergeManifest = FALSE, sex = NULL)
```

```
[preprocessQuantile] Mapping to genome.
[preprocessQuantile] Fixing outliers.
[preprocessQuantile] Quantile normalizing.
```

Now, we have covered all the preprocessing steps included in `minfi`. We have a `GenomicRatioSet`, and we are ready to start a statistical analysis of the data.

10 dmpFinder: to find differentially methylated positions (DMPs)

To find differentially methylated positions with respect to a phenotype of interest, the function `dmpFinder` is our friend. The phenotype may be categorical (e.g. cancer vs. normal) or continuous (e.g. blood pressure). Let's apply `dmrFinder` on the genomicRatioSet.quantile that we have created above, with the predicted sex as the phenotype:

```
> beta <- getBeta(gRatioSet.quantile)
> predictedSex <- pData(gRatioSet.quantile)$predictedSex
> dmp <- dmpFinder(beta, pheno = predictedSex  , type = "categorical")
> head(dmp)
```

```
           intercept        f         pval        qval
cg14928964 0.6267611 25241.22 9.414902e-09 0.001840777
cg26010707 0.4258709 20569.70 1.417603e-08 0.001840777
cg22484503 0.5228533 19957.59 1.505879e-08 0.001840777
cg01771673 0.6362476 19165.32 1.632932e-08 0.001840777
cg11854877 0.5623077 17787.24 1.895707e-08 0.001840777
cg24779040 0.5850674 15312.92 2.557680e-08 0.002069641
```

11 Bumphunter: to find differentially methylated regions (DMRs)

The `bumphunter` function in `minfi` is a version of the bump hunting algorithm [5] adapted to the 450k array, relying on the `bumphunter` function implemented in the eponym package `bumphunter` [6].

Instead of looking for association between a single genomic location and a phenotype of interest, `bumphunter` looks for association with **genomic regions**. In the context of the 450k array, the algorithm first defines *clusters* of probes. Clusters are simply groups of probes such that two consecutive probe locations in the cluster are not separated by more than some distance `mapGap`. To find the differentially methylated regions, the following statistical model is used.

Underlying statistical model for bumphunter

(statistical model described in [6]) We assume that we have data Y_{ij}, here Beta-values of M-values, where i represents biological replicate, and l_j represents a genomic location. The basis statistical model is

$$Y_{ij} = \beta_0(l_j) + \beta_1(l_j)X_j + \epsilon_{ij}$$

with i representing subject, l_j representing the jth location, X_j is the covariate of interest (for example $X_j = 1$ for cases and $X_j = 0$ otherwise), ϵ_{ij} is measurement error, $\beta_0(l)$ is a baseline function, and $\beta_1(l)$ is the parameter of interest, which is a function of location. We assume that $\beta_1(l)$ will be equal to zero over most of the genome, and we want to identify segments where $\beta_1 \neq 0$, which we call **bumps**.

We want to share information between nearby locations, typically through some form of smoothing. As explained in [[[]], the algorithm can be summarized into the following main steps:

A. Compute the coefficients $\beta_1(l_j)$ at each genomic location l_j.

B. Optional smoothing of the coefficients $\beta_1(l_j)$ across genomic locations within a cluster of probes

C. Define a candidate region as a region for which$\beta_1(l_j)$ exceeds a given predefined threshold for all probes within the region

D. Test for significance of the candidate regions using a permutation scheme

Since the permutation test is expensive, parallel computation is implemented in the `bumphunter` function. The `foreach` package allows different parallel "back-ends" that will distribute the computation across multiple cores in a single machine, or across machines in a cluster.

```
> pheno <- pData(gRatioSet.quantile)$age
> designMatrix <- model.matrix(~ pheno)
> dmrs <- bumphunter(gRatioSet.quantile, design = designMatrix, cutoff = 0.05, B=1)
> names(dmrs)

[1] "table"            "coef"             "fitted"           "pvaluesMarginal"
[5] "null"             "algorithm"

> head(dmrs$table, n = 3)

       chr     start       end      value     area cluster indexStart indexEnd
18657 chrX 153774721 153776358 -0.1267772 2.662320  204615     484928   484948
17769 chrX  47004051  47005131 -0.1321849 2.379329  201679     476641   476658
17986 chrX  68835489  68837158 -0.1366464 2.186342  202426     478793   478808
       L clusterL p.value fwer p.valueArea fwerArea
18657 21       21       0    0           0        0
17769 18       18       0    0           0        0
17986 16       16       0    0           0        0
```

References

[1] Hansen, K.D. and Aryee, M. *minfi: Analyze Illumina's 450k methylation arrays. R package version 1.7.8*

[2] *shinyMethyl at shinyMethyl.com*

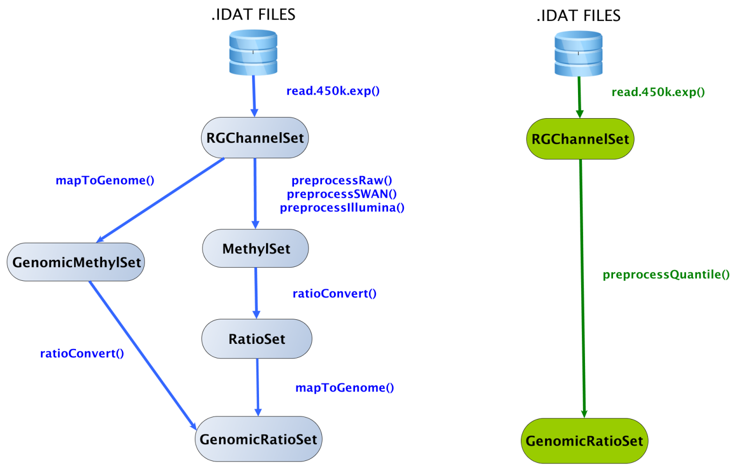


Figure 2: Flow chart of the `minfi`'s functions

[3] The Cancer Genome Atlas (TCGA): data portal at https://tcga-data.nci.nih.gov/tcga/

[4] Maksimovic, J., Gordon, L., Oshlack, A. *SWAN: subset quantile Within-Array Normalization for Illumina Infinium HumanMethylation450 BeadChips* Genome Biology, 13(6) R44, 2012

[5] Jaffe, A. E, Murakami, P. Lee, H. Leek, J.T., Fallin, M.D., Feinberg, A.P., Irizarry, R. *Bump hunting to identify differentially methylated regions in epidemiology studies* International Journal of Epidemiology, 41(1): 200-209, 2012

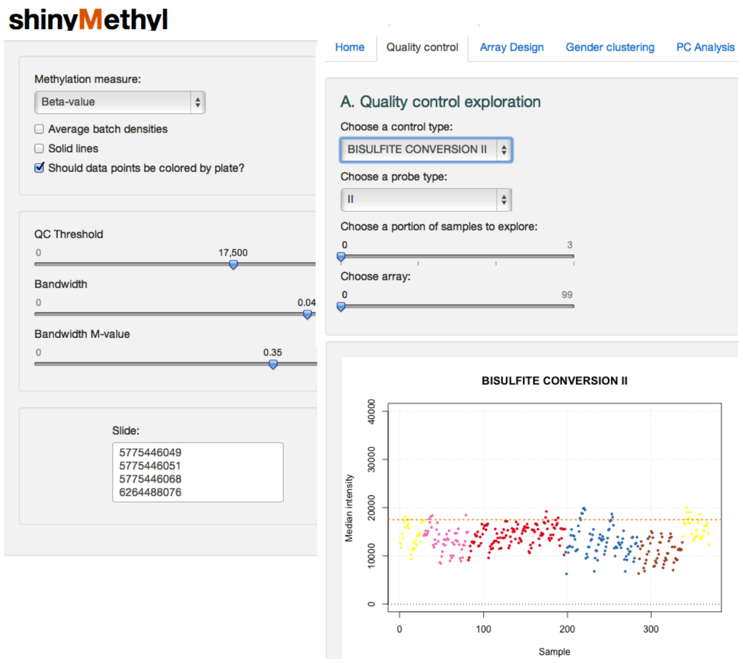
[6] Irizarry, A.R., Aryee, M., Corrada Bravo, H., Hansen, K.D., Jaffee, H.A *bumphunter: Bump Hunter, R package version 1.1.10*

12 Exercises

1) Before processing a RGChannelSet further, coudl you remove the probes which failed more than 50% of the samples in the example dataset?

2) For the top loci that we found differentially methylated for the predicted sex, could you tell if those loci are mostly mapped to the X and Y chromosomes?

3) It is known that the Beta-value distribution of the Type I probes is different from the Beta value distribution of the Type II probes. Can you verify this with by plotting the Beta-value distribution density for each type separately?


shinyMethyl

Methylation measure: Beta-value

☐ Average batch densities
☐ Solid lines
☑ Should data points be colored by plate?

QC Threshold: 0 — 17,500

Bandwidth: 0 — 0.04

Bandwidth M-value: 0 — 0.35

Slide:
5775446049
5775446051
5775446068
6264488076

A. Quality control exploration

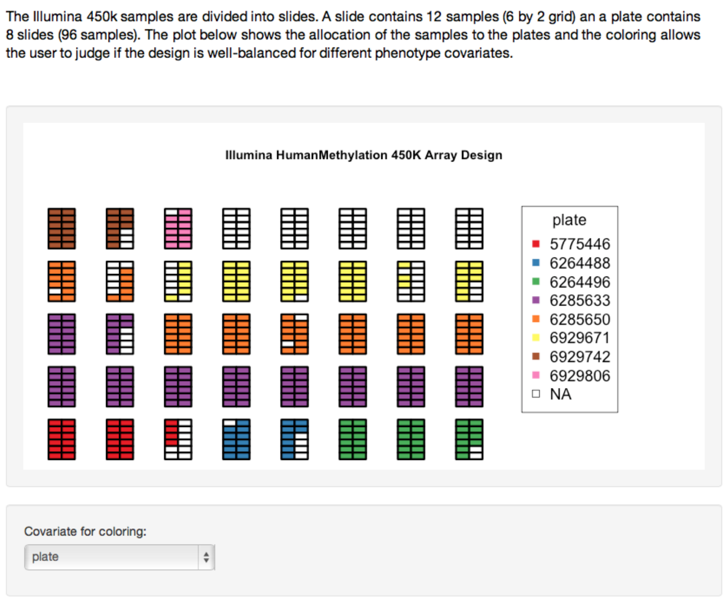
Choose a control type: BISULFITE CONVERSION II

Choose a probe type: II

Choose a portion of samples to explore: 0 — 3

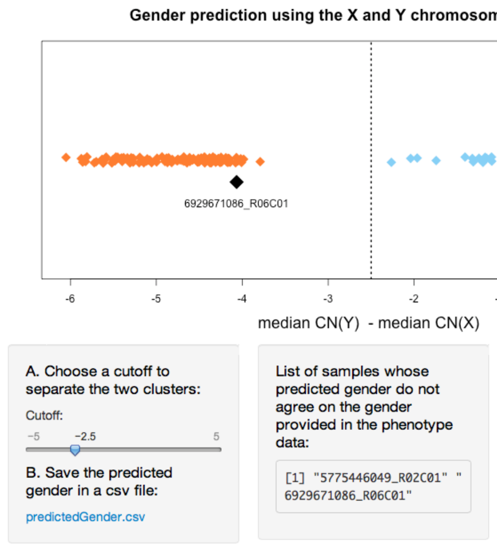
Choose array: 0 — 99

Figure 3: Quality control panel in `shinyMethyl`


The Illumina 450k samples are divided into slides. A slide contains 12 samples (6 by 2 grid) an a plate contains 8 slides (96 samples). The plot below shows the allocation of the samples to the plates and the coloring allows the user to judge if the design is well-balanced for different phenotype covariates.

Covariate for coloring: plate

Figure 4: Array design panel in `shinyMethyl`



A. Choose a cutoff to separate the two clusters:

Cutoff: −2.5

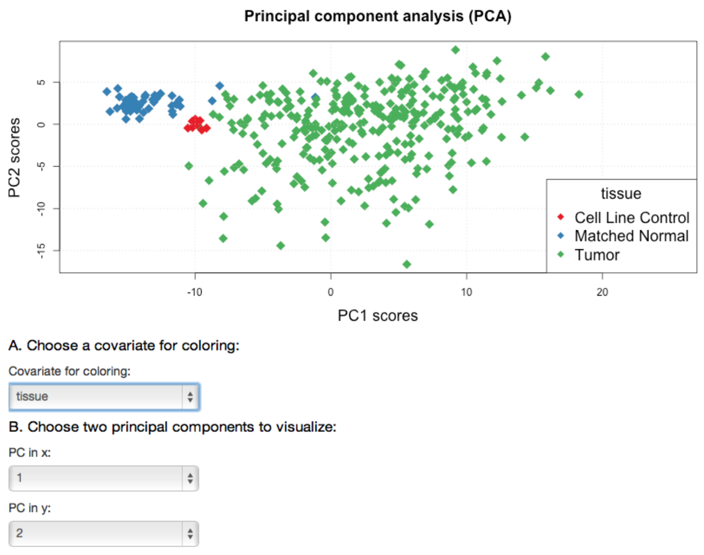
B. Save the predicted gender in a csv file:

predictedGender.csv

List of samples whose predicted gender do not agree on the gender provided in the phenotype data:

```
[1] "5775446049_R02C01" "
6929671086_R06C01"
```

Figure 5: Gender prediction panel of shinyMethyl



A. Choose a covariate for coloring:

Covariate for coloring: tissue

B. Choose two principal components to visualize:

PC in x: 1

PC in y: 2

Figure 6: Principal component analysis (PCA) panel in `shinyMethyl`