

DNA Methylation Practical 1 - Cleaning

Allan McRae
a.mcrae@uq.edu.au

Sonia Shah
sonia.shah@uq.edu.au

Overview

This practical will cover

- Loading DNA methylation array data into R
- Data quality control
- Creating a cleaned data-set

Example Data-set

Inoshita M, Numata S, Tajima A, Kinoshita M et al. **Sex differences of leukocytes DNA methylation adjusted for estimated cellular proportions.** *Biology of Sex Differences* 2015 6:11

Available on GEO (Gene Expression Omnibus) with accession GSE67393

- 116 samples (53 female, 63 male)
- Raw idat files from Illumina Infinium 450k Human Methylation Beadchip
- Peripheral leukocytes from healthy controls

Installing R packages

```
source("http://bioconductor.org/biocLite.R")
biocLite("illuminaio")
biocLite("limma")
biocLite("IlluminaHumanMethylation450kmanifest")
biocLite("IlluminaHumanMethylation450kanno.ilmn12.hg19")
biocLite("CopyNumber450kData")
biocLite("DNAcopy")
install.packages("markdown")
install.packages("knitr")
install.packages("devtools")
library(devtools)
install_github("perishky/meffil")
install.packages("qqman")
```

Meffil

“Efficient algorithms for analyzing DNA methylation data generated using Infinium HumanMethylation450 or MethylationEPIC BeadChips”

```
library(meffil)  
packageVersion("meffil")
```

Ensure the meffil version is at least 1.0.0 due to a bug in earlier versions

You can optionally control the number of CPU cores meffil uses when reading/analysing data

```
options(mc.cores = 1)
```

Reading Data

Meffil requires a sample sheet to read in data files. It can automatically create one using:

```
samplesheet =  
  meffil.create.samplesheet("/path/to/idat/files")  
head(samplesheet)
```

We will supplement the sample sheet with known sex values

```
sex = read.table("sex.txt")  
samplesheet$Sex =  
  sex[match(samplesheet$Sample_Name, sex[,1]),2]  
head(samplesheet)
```

Reading Data

It will be very useful later if the “Slide”, “sentry_row”, etc variables are recognised as a Factor in R. Check this with:

```
str(samplesheet)
```

If any variables apart from “Sample_Name” and “Basename” are not the Factor type, fix this with:

```
samplesheet$Slide = as.factor(samplesheet$Slide)

samplesheet$sentry_row =
                        as.factor(samplesheet$sentry_row)

...

```

Reading Data

When reading in data in Meffil, it is useful to specify what cell types the data come from. This can be used for QC.

```
meffil.list.cell.type.references()
```

We are using whole blood, so will select the “blood gse35069 complete” reference when reading the data

```
qc.objects <- meffil.qc(samplesheet,  
                        cell.type.reference="blood gse35069 complete",  
                        verbose=TRUE)
```

```
save(qc.objects, file="qc.objects.Robj")
```

```
save(samplesheet, file="samplesheet.Robj")
```


Matching Genotypes

It is useful to use the genotypes on the methylation array data to match against SNP data from other sources. This detects potential sample mix-ups.

We do not have a complete genotype file for this data, but the following would extract the necessary genotypes:

(EXAMPLE ONLY)

```
writeLines(meffil.snp.names(), con="snp-names.txt")  
  
# outside of R  
plink2 --bfile dataset --extract snp-names.txt \  
      --recode A --out genotypes
```

Matching Genotypes

Read in the pre-generated genotype file using:

```
genotypes <- meffil.extract.genotypes("genotypes.raw")  
  
genotypes <- genotypes[,  
  match(samplesheet$Sample_Name, colnames(genotypes))]
```

The second command ensures that the genotypes read in from the “genotypes.raw” file are in the same order as those in the sample sheet.

Setting QC parameters

We will use the parameters suggested in the Meffil manual:

```
qc.parameters <- meffil.qc.parameters(  
  beadnum.samples.threshold = 0.1,  
  detectionp.samples.threshold = 0.1,  
  detectionp.cpgs.threshold = 0.1,  
  beadnum.cpgs.threshold = 0.1,  
  sex.outlier.sd = 5,  
  snp.concordance.threshold = 0.95,  
  sample.genotype.concordance.threshold = 0.8  
)
```

Setting QC parameters

beadnum.samples.threshold = fraction of probes that failed the threshold of 3 beads.

detectionp.samples.threshold = fraction of probes that failed a detection.pvalue threshold of 0.01.

beadnum.cpgs.threshold = fraction of samples that failed the threshold of 3 beads.

detectionp.cpgs.threshold = fraction of samples that failed the detection.pvalue threshold of 0.01.

sex.outlier.sd = number of standard deviations to determine whether sample is sex outlier

snp.concordance.threshold = concordance threshold to include snps to calculate sample concordance

sample.genotype.concordance.threshold = concordance threshold to determine whether sample is outlier

Run QC and Generate Report

The following commands run the QC pipeline, save the output and generate a report. Finally we list which samples fail QC and the reason why.

```
qc.summary <- meffil.qc.summary(  
  qc.objects,  
  parameters = qc.parameters,  
  genotypes=genotypes  
)  
  
save(qc.summary, file="qcsummary.Robj")  
  
meffil.qc.report(qc.summary,  
                output.file="qc-report.html")  
  
outlier = qc.summary$bad.samples
```

Remove Bad Samples

Bad samples are removed from the dataset. Probes failing QC metrics are filtered out later on

```
length(qc.objects)

qc.objects <- meffil.remove.samples(qc.objects,
                                   outlier$sample.name)

length(qc.objects)

save(qc.objects, file="qc.objects.clean.Robj")
```

Re-run the QC on the Clean Data

It is important to re-run the QC to ensure the correct samples were removed and that no further issues are found

```
qc.summary <- meffil.qc.summary(  
  qc.objects,  
  parameters = qc.parameters,  
  genotypes=genotypes  
)  
  
save(qc.summary, file="qcsummary.clean.Robj")  
  
meffil.qc.report(qc.summary,  
  output.file="qc-report.clean.html")
```