Working with Illumina 450k Methylation Arrays

Tim Triche, Jr. & Sean Davis

April 27, 2020

Contents

1 Creating a MethyLumiSet object from IDATs

This also happens to be the first step in the TCGA processing pipeline. The complete pipeline is available on GitHub as the EGC.tools project. Ten samples from the TCGA breast cancer (BRCA) project are included in the TCGAMethylation450k package, which should be installed for this step.

```
suppressPackageStartupMessages(require('methylumi'))
suppressPackageStartupMessages(require('TCGAMethylation450k'))
suppressPackageStartupMessages(require('FDb.InfiniumMethylation.hg19'))
```

```
## read in 10 BRCA IDATs
idatPath <- system.file('extdata/idat',package='TCGAMethylation450k')
mset450k <- methylumIDAT(getBarcodes(path=idatPath), idatPath=idatPath)

## 0 HumanMethylation27 samples found
## 10 HumanMethylation450 samples found
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Warning in readChar(con, nchars = n):  truncating string with embedded nuls
## Attempting to extract protocolData() from list...
## Determining chip type from IDAT protocolData...

sampleNames(mset450k) <- paste0('TCGA', seq_along(sampleNames(mset450k)))
show(mset450k)

##
## Object Information:
## MethyLumiSet (storageMode: lockedEnvironment)
## assayData: 485577 features, 10 samples
##   element names: betas, methylated, methylated.OOB, pvals, unmethylated, unmethylated.OOB
## protocolData: none
## phenoData
##   sampleNames: TCGA1 TCGA2 ... TCGA10 (10
##     total)
##   varLabels: barcode
##   varMetadata: labelDescription
## featureData
##   featureNames: cg00000029 cg00000108 ...
##     rs9839873 (485577 total)
##   fvarLabels: Probe_ID DESIGN COLOR_CHANNEL
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
```

```
## Annotation: IlluminaHumanMethylation450k
## Major Operation History:
##             submitted            finished
## 1 2020-04-27 21:25:17 2020-04-27 21:25:41
## 2 2020-04-27 21:25:43 2020-04-27 21:25:44
##                                                                      command
## 1 methylumIDAT(barcodes = getBarcodes(path = idatPath), idatPath = idatPath)
## 2                                            Subset of 485577 features.
```

Note that the default is to collect opposite-channel fluorescence from Type I methylation probes (which are paired and designed to fluoresce in one channel) in the matrices 'methylated.OOB' and 'unmethylated.OOB' (OOB, as in out-of-band) for use in background correction and perhaps additional steps. This also allows a user to coerce the resulting object into minfi's RGChannelSet if desired since all of the signal information in the IDATs is thus retained.

2 Negative and normalization controls

Plot the negative and normalization controls:

```
library(ggplot2)
## for larger datasets, the by.type argument be set to FALSE
## positional effects will manifest as a wave-like pattern
p <- qc.probe.plot(mset450k, by.type=TRUE)
print(p)
```

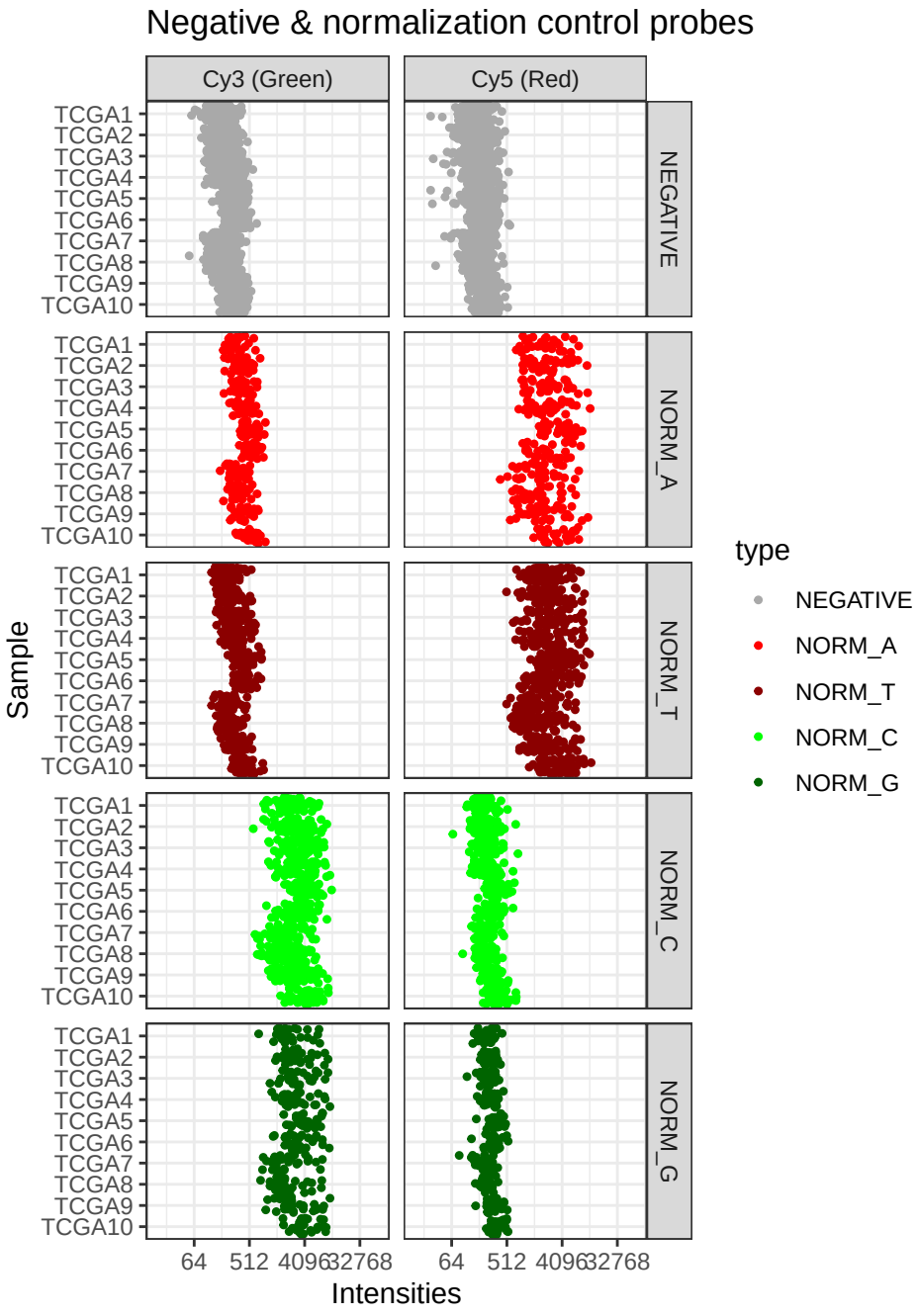

Figure 1: Some of the controls on the 450k chip

3 Preprocessing the raw data

After importing the data from IDATs, the next step is to background correct and dye bias equalize the data. The default for background correction is a normal- exponential model which uses the out-of-band intensities as control probes. Dye bias correction is performed by picking the least-biased sample, and using it as a reference for red:green intensity ratio adjustments based on the normalization controls. Other approaches to preprocessing (as implemented in minfi and lumi) include various flavors of quantile normalization and smoothing spline fits.

After preprocessing, we can reduce the size of the resulting MethyLumiSet substantially by dropping the out-of-band intensities with stripOOB(). This frees up some memory, but precludes later coercion to an RGChannelSet.

```
mset450k.proc <- stripOOB(normalizeMethyLumiSet(methylumi.bgcorr(mset450k)))

## Background mean & SD estimated from 178406 probes

## Loading required package:  MASS
##
## Attaching package:  'MASS'
## The following object is masked from 'package:AnnotationDbi':
##
##    select

## Background mean & SD estimated from 92596 probes

## Normalizing via Illumina controls...
## Using sample number 5 as reference level...
```

Now we compare the post-processing controls with those from figure 1.

```
library(ggplot2)
p2 <- qc.probe.plot(mset450k.proc, by.type=TRUE)
print(p2)
```

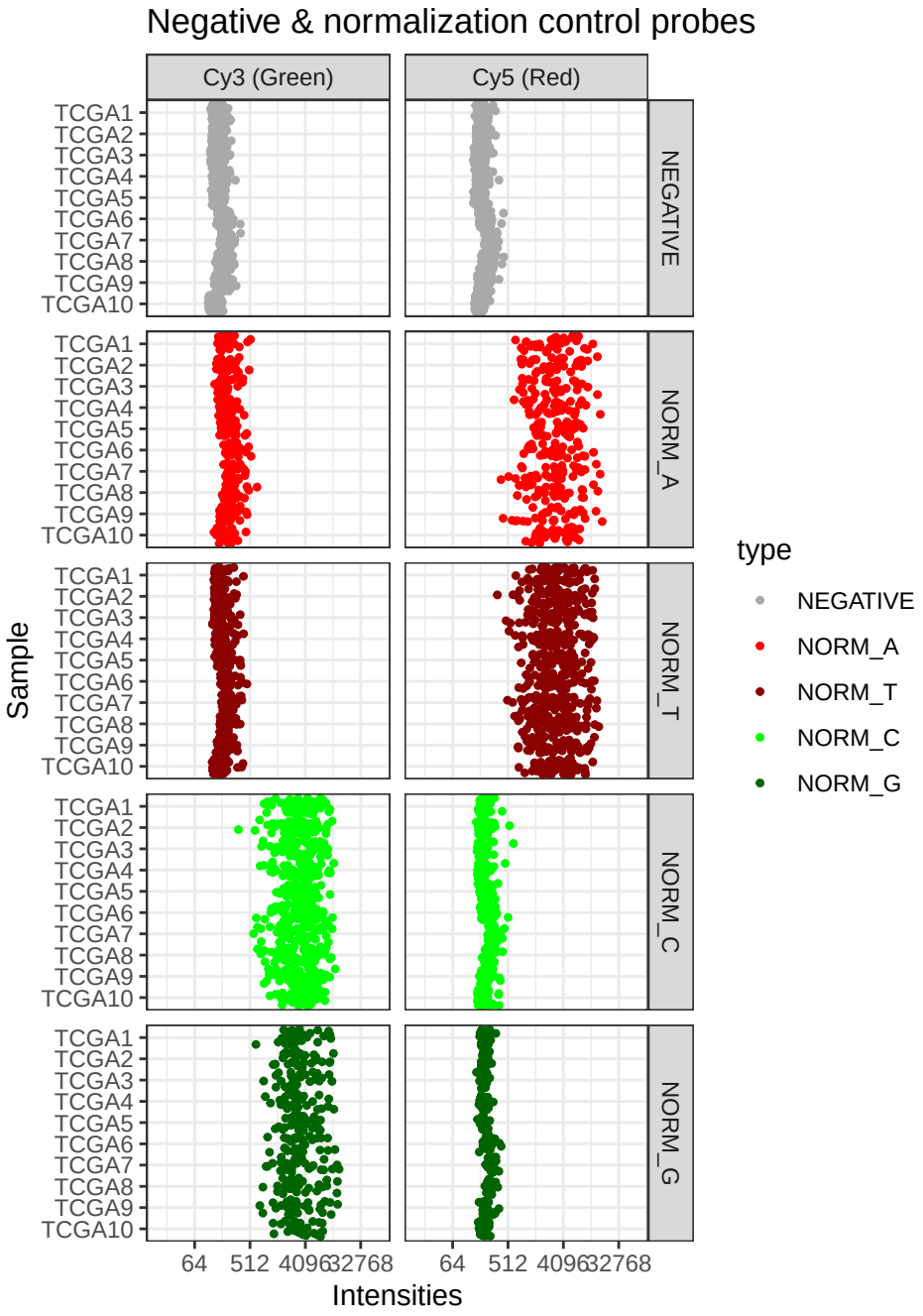

Figure 2: Controls after preprocessing

4 Coercions to other data structures

Coercions are provided to and from various data structures in the lumi and minfi packages. Each provides various functionality and exhibits different design decisions. One may be more appropriate than the other for some needs. Preprocessing in methylumi retains SNP probes, which can identify label swaps, but is less efficient than preprocessing in minfi, and cannot use shinyMethyl.

Coercing to lumi (e.g. for lumiMethyN or similar):

```
suppressPackageStartupMessages(require(lumi))

## No methods found in package 'RSQLite' for request:  'dbListFields' when loading 'lumi'

mset450k.lumi <- as(mset450k.proc, 'MethyLumiM')
show(mset450k.lumi)

## MethyLumiM (storageMode: lockedEnvironment)
## assayData: 485577 features, 10 samples
##   element names: detection, exprs, methylated, unmethylated
## protocolData: none
## phenoData
##   sampleNames: TCGA1 TCGA2 ... TCGA10 (10
##     total)
##   varLabels: barcode mu.Cy3 ... offset.Cy5
##     (9 total)
##   varMetadata: labelDescription
## featureData
##   featureNames: cg00000029 cg00000108 ...
##     rs9839873 (485577 total)
##   fvarLabels: Probe_ID DESIGN COLOR_CHANNEL
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation: IlluminaHumanMethylation450k
```

Coercing back to a MethyLumiSet:

```
mset450k.andBack <- as(mset450k.lumi, 'MethyLumiSet')
show(mset450k.andBack)
```

```
##
## Object Information:
## MethyLumiSet (storageMode: lockedEnvironment)
## assayData: 485577 features, 10 samples
##   element names: betas, methylated, pvals, unmethylated
## protocolData: none
## phenoData
##   sampleNames: TCGA1 TCGA2 ... TCGA10 (10
##     total)
##   varLabels: barcode mu.Cy3 ... offset.Cy5
##     (9 total)
##   varMetadata: labelDescription
## featureData
##   featureNames: cg00000029 cg00000108 ...
##     rs9839873 (485577 total)
##   fvarLabels: Probe_ID DESIGN COLOR_CHANNEL
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation: IlluminaHumanMethylation450k
## Major Operation History:
##             submitted            finished
## 1 2020-04-27 21:25:17 2020-04-27 21:25:41
## 2 2020-04-27 21:25:43 2020-04-27 21:25:44
## 3 2020-04-27 21:26:15            21:26:34
## 4 2020-04-27 21:26:34            21:26:38
## 5 2020-04-27 21:26:38 2020-04-27 21:26:38
## 6 2020-04-27 21:26:51 2020-04-27 21:26:52
##                                                                          command
## 1 methylumIDAT(barcodes = getBarcodes(path = idatPath), idatPath = idatPath)
## 2                                                 Subset of 485577 features.
## 3                                           methylumi.bgcorr(x = mset450k)
## 4                                              normalizeViaControls(x = x)
## 5       stripOOB(object = normalizeMethyLumiSet(methylumi.bgcorr(mset450k)))
## 6                                                  asMethod(from = object)
##   lumiVersion
## 1        <NA>
## 2        <NA>
## 3        <NA>
## 4        <NA>
## 5        <NA>
## 6      2.35.0
```

MethyLumiSet objects with OOB matrices can be coerced to RGChannelSet objects for further processing using functions found in the minfi or ChAMP packages.

```
suppressPackageStartupMessages(require(FDb.InfiniumMethylation.hg19))
rgSet450k <- as(mset450k, 'RGChannelSet')

## Loading required package:  rtracklayer
## Fetching coordinates for hg19...

show(rgSet450k)

## class: RGChannelSet
## dim: 621928 10
## metadata(0):
## assays(2): Green Red
## rownames(621928): 14782418 12709357 ...
##   28673402 13742412
## rowData names(0):
## colnames(10): TCGA1 TCGA2 ... TCGA9 TCGA10
## colData names(1): barcode
## Annotation
##   array: IlluminaHumanMethylation450k
```

The above will not work for the processed data, but only because we called stripOOB() on the resulting object to reduce its size. If you plan on using a preprocessed MethyLumiSet in minfi for further processing, don't strip it.

The GenomicMethylSet and GenomicRatioSet classes in minfi inherit from the RangedSummarizedExperiment class, which has some particularly useful features:

```
suppressPackageStartupMessages(require(minfi))
suppressPackageStartupMessages(require(IlluminaHumanMethylation450kanno.ilmn12.hg19))
grSet450k <- mapToGenome(mset450k.andBack)

sexChroms <- GRanges( seqnames=c('chrX','chrY'),
                      IRanges(start=c(1, 1),
                              end=c(155270560, 59373566)),
                      strand=c('*','*') )
summary(subsetByOverlaps(grSet450k, sexChroms))

## [1] "GenomicMethylSet object of length 11648 with 0 metadata columns"

dim(subsetByOverlaps(grSet450k, sexChroms))

## [1] 11648    10
```

These SummarizedExperiment-derived objects can be subsetted by nearly anything that has an interval-based representation. Here we extract some promoters, but one could just as easily use AnnotationHub resources to find CTCF peaks or, say, H3K4me1 peaks in ChIP-seq data (often associated with transcriptional enhancers; the presence or absence of DNA methylation may help determine their activity).

```
## perhaps more topical:
suppressPackageStartupMessages(require(TxDb.Hsapiens.UCSC.hg19.knownGene))
suppressPackageStartupMessages(require(Homo.sapiens))
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene

KDM6AEntrezID=org.Hs.egSYMBOL2EG[['KDM6A']]
txs.KDM6A <- transcriptsBy(txdb, 'gene')[[KDM6AEntrezID]]
tss.KDM6A <- unique(resize(txs.KDM6A, 1, fix='start')) ## two start sites
promoters.KDM6A <- flank(tss.KDM6A, 100) ## an arbitrary distance upstream
show( subsetByOverlaps(grSet450k, promoters.KDM6A) ) ## probes in this window

## class: GenomicMethylSet
## dim: 6 10
## metadata(0):
## assays(2): Meth Unmeth
## rownames(6): cg14384228 cg07167981 ...
##   cg17824914 cg06877198
## rowData names(0):
## colnames(10): TCGA1 TCGA2 ... TCGA9 TCGA10
## colData names(9): barcode mu.Cy3 ...
##   alpha.Cy5 offset.Cy5
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
## Preprocessing
##   Method: methylumi, background corrected, dye bias equalized
##   minfi version: 1.35.0
##   Manifest version: 0.4
```

Consult the AnnotationHub package vignette for some other possibilities. If you are unfamiliar with the powerful GenomicRanges and GenomicFeatures packages, you may want to familiarize yourself with them as well.

5 sessionInfo

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

- R version 4.0.0 (2020-04-24), `x86_64-w64-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`
- Running under: `Windows Server 2012 R2 x64 (build 9600)`
- Matrix products: default
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.51.0, Biobase 2.49.0, BiocGenerics 0.35.0, Biostrings 2.57.0, DelayedArray 0.15.0, FDb.InfiniumMethylation.hg19 2.2.0, GO.db 3.10.0, GenomeInfoDb 1.25.0, GenomicFeatures 1.41.0, GenomicRanges 1.41.0, Homo.sapiens 1.3.1, IRanges 2.23.0, IlluminaHumanMethylation450kanno.ilmn12.hg19 0.6.0, MASS 7.3-51.6, OrganismDbi 1.31.0, S4Vectors 0.27.0, SummarizedExperiment 1.19.0, TCGAMethylation450k 1.23.0, TxDb.Hsapiens.UCSC.hg19.knownGene 3.2.2, XVector 0.29.0, bumphunter 1.31.0, foreach 1.5.0, ggplot2 3.3.0, iterators 1.0.12, knitr 1.28, lattice 0.20-41, limma 3.45.0, locfit 1.5-9.4, lumi 2.41.0, matrixStats 0.56.0, methylumi 2.35.0, minfi 1.35.0, org.Hs.eg.db 3.10.0, reshape2 1.4.4, rtracklayer 1.49.0, scales 1.1.0, xtable 1.8-4
- Loaded via a namespace (and not attached): BiocFileCache 1.13.0, BiocManager 1.30.10, BiocParallel 1.23.0, DBI 1.1.0, DelayedMatrixStats 1.11.0, GEOquery 2.57.0, GenomeInfoDbData 1.2.3, GenomicAlignments 1.25.0, HDF5Array 1.17.0, KernSmooth 2.23-17, Matrix 1.2-18, R6 2.4.1, RBGL 1.65.0, RColorBrewer 1.1-2, RCurl 1.98-1.2, RSQLite 2.2.0, Rcpp 1.0.4.6, Rhdf5lib 1.11.0, Rsamtools 2.5.0, XML 3.99-0.3, affy 1.67.0, affyio 1.59.0, annotate 1.67.0, askpass 1.1, assertthat 0.2.1, base64 2.0, beanplot 1.2, biomaRt 2.45.0, bit 1.1-15.2, bit64 0.9-7, bitops 1.0-6, blob 1.2.1, codetools 0.2-16, colorspace 1.4-1, compiler 4.0.0, crayon 1.3.4, curl 4.3, data.table 1.12.8, dbplyr 1.4.3, digest 0.6.25, doRNG 1.8.2, dplyr 0.8.5, ellipsis 0.3.0, evaluate 0.14, farver 2.0.3, genefilter 1.71.0, glue 1.4.0, graph 1.67.0, grid 4.0.0, gtable 0.3.0, highr 0.8, hms 0.5.3, httr 1.4.1, illuminaio 0.31.0, lifecycle 0.2.0, magrittr 1.5, mclust 5.4.6, memoise 1.1.0, mgcv 1.8-31, multtest 2.45.0, munsell 0.5.0, nleqslv 3.3.2, nlme 3.1-147, nor1mix 1.3-0, openssl 1.4.1, pillar 1.4.3, pkgconfig 2.0.3, plyr 1.8.6, preprocessCore 1.51.0, prettyunits 1.1.1, progress 1.2.2, purrr 0.3.4, quadprog 1.5-8, rappdirs 0.3.1, readr 1.3.1, reshape 0.8.8, rhdf5 2.33.0,

rlang 0.4.5, rngtools 1.5, scrime 1.3.5, siggenes 1.63.0, splines 4.0.0, stringi 1.4.6, stringr 1.4.0, survival 3.1-12, tibble 3.0.1, tidyr 1.0.2, tidyselect 1.0.0, tools 4.0.0, vctrs 0.2.4, withr 2.2.0, xfun 0.13, xml2 1.3.2, zlibbioc 1.35.0