The minfi User's Guide

Kasper D. Hansen, Jean-Phillipe Fortin

1 May 2017

Abstract

A comprehensive guide to using the minfi package for analyzing DNA methylation microarrays from Illumina.vignettes.

Package version: minfi 1.22.1

Contents

Introduction

The *minfi* package provides tools for analyzing Illumina's Methylation arrays, specifically the 450k and EPIC (also known as the 850k) arrays. We have partial support for the older 27k array.

The tasks addressed in this package include preprocessing, QC assessments, identification of interesting methylation loci and plotting functionality. Analyzing these types of arrays is ongoing research in ours and other groups.

The input data to this package are IDAT files, representing two different color channels prior to normalization. This is the most complete data type, because IDAT files includes measurements on control probes. It is possible to use Genome Studio

files together with the data structures contained in this package, but only some functionality is available because Genome Studio output does not contain control probe information. In addition, usually Genome Studio output is normalized using the methods implemented in Genome Studio and these are generally considered inferior.

Citing the minfi package

The MINFI package contains methods which are described across multiple manuscripts, by different non-overlapping authors. This makes citing the package a bit difficult, so here is a guide.

- If you are using MINFI in a publication, please cite (Aryee et al. 2014). This publication includes details on sex estimation using `getSex()`, quality control using `getQC()`, quantile normalization using `preprocessQuantile()`, bump hunting using `bumphunter()` and block finding using `blockFinder()`.
- If you are using MINFI to analyze EPIC or 27k arrays, please cite (Fortin, Triche, and Hansen 2016). The publication includes details on `convertArray()` and `combineArrays()`, extending NOOB to work in single-sample mode as well as using `estimateCellCounts()` with reference data from the 450k array to estimate cell type composition for EPIC data.
- If you are using `preprocessQuantile()`, it would be considerate to also cite (Touleimat and Tost 2012), since this publication describes a method essentially identical to `preprocessQuantile()`.
- If you are using `bumphunter()`, you should also cite the original bump hunter publication (Jaffe et al. 2012).
- If you are using SWAN normalization as implemented in `preprocessSWAN()` please cite (Maksimovic, Gordon, and Oshlack 2012).
- If you are using noob background correction as implemented in `preprocessNoob()`, please cite (Triche et al. 2013).
- If you are using functional normalization as implemented in `preprocessFunnorm()`, please cite (Fortin et al. 2014). The default in `preprocessFunnorm()` is to do noob background correction. If this is used, please also cite (Triche et al. 2013).
- If you are estimating A/B compartments as implemented in `compartments()` and `extractAB()`, please cite (Fortin and Hansen 2015).

If you're using Bibtex, you can get the citations in this format by

```
toBibtex(citation("minfi"))
```

Terminology

The literature is often a bit unspecific wrt. the terminology for the DNA methylation microarrays.

For the 450k microarray, each sample is measured on a single array, in two different color channels (red and green). Each array measures roughly 450,000 CpG positions. Each CpG is associated with two measurements: a methylated measurement and an "un"-methylated measurement. These two values can be measured in one of two ways: using a "Type I" design or a "Type II design". CpGs measured using a Type I design are measured using a single color, with two different probes in the same color channel providing the methylated and the unmethylated measurements. CpGs measured using a Type II design are measured using a single probe, and two different colors provide the methylated and the unmethylated measurements. Practically, this implies that on this array there is **not** a one-to-one correspondence between probes and CpG positions. We have therefore tried to be precise about this and we refer to a "locus" (or "CpG") when we refer to a single-base genomic locus, and we differentiate this from a "probe". The previous generation 27k methylation array uses only the Type I design, and the EPIC arrays uses both Type I and Type II.

Differences in DNA methylation between samples can either be at a single CpG which is called a differentially methylated position (DMP), or at a regional level which is called a differentially methylated region (DMR).

Physically, each sample is measured on a single "array". For the 450k design, there are 12 arrays on a single physical "slide" (organized in a 6 by 2 grid). Slides are organized into "plates" containing at most 8 slides (96 arrays). The EPIC array has 8 arrays per slide and 64 arrays per plate.

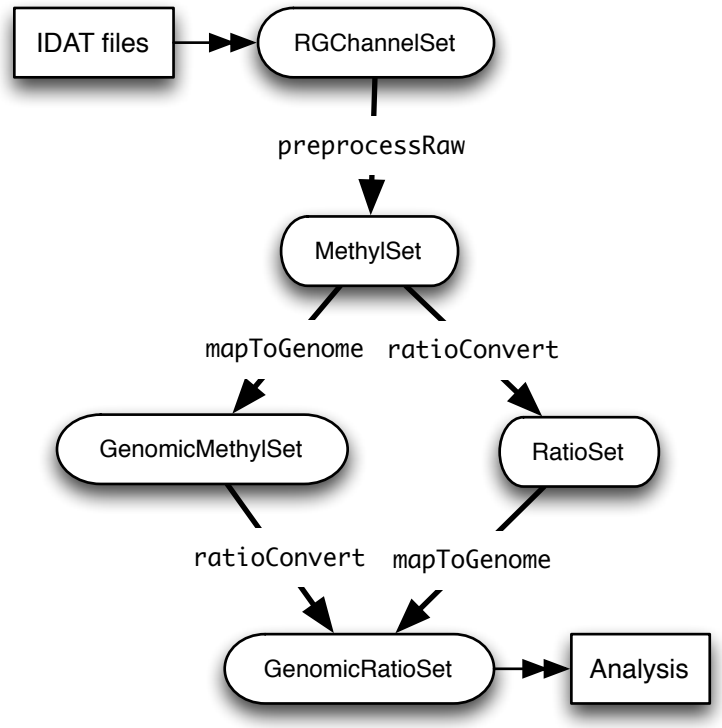

Figure 1: The class hierarchy of minfi

Dependencies

This document has the following dependencies

```
library(minfi)
library(minfiData)
```

minfi classes

The MINFI package is designed to be very flexible for methods developers. This flexibility comes at a cost for users; they need to understand a few different data classes:

- `RGChannelSet` : raw data from the IDAT files; this data is organized at the probe (not CpG locus) level. This data has two channels: Red and Green.
- `MethylSet` : data organized by the CpG locus level, but not mapped to a genome. This data has two channels: Meth (methylated) and Unmeth (unmethylated).
- `RatioSet` : data organized by the CpG locus level, but not mapped to a genome. The data has at least one of two channels: Beta and/or M (logratio of Beta). It may optionally include a CN channel (copy number).
- `GenomicMethylSet` : like a `MethylSet`, but mapped to a genome.
- `GenomicRatioSet` : like a `RatioSet`, but mapped to the genome.

A class hierarchy is as follows

To make this more clear, let us look at the example data called `RGsetEx` from the *minfiData* package. This is the the number of features and and classes as we move through the class hierarchy (code not run):

```
RGsetEx
## class: RGChannelSet
## dim: 622399 6
## metadata(0):
## assays(2): Green Red
## rownames(622399): 10600313 10600322 ... 74810490 74810492
## rowData names(0):
## colnames(6): 5723646052_R02C02 5723646052_R04C01 ...
##   5723646053_R05C02 5723646053_R06C02
## colData names(13): Sample_Name Sample_Well ... Basename filenames
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
## RGsetEx: RGChannelSet, 622,399 features
MsetEx <- preprocessRaw(RGsetEx)
## MsetEx: MethylSet, 485,512 features
GMsetEx <- mapToGenome(MsetEx)
## GMsetEx: GenomicMethylSet, 485,512 features
```

Note how the number of features changes. In the `RGChannelSet` a feature is a probe (which is different from a CpG locus, see the Terminology section). In the `MethylSet` each feature is now a methylation locus, and it has fewer features because some loci are measured using two probes. Finally, the `GenomicMethylSet` has the same size as the `MethylSet`, but it could in principle be smaller in case the annotation you use says that some probes do not map to the genome (in this case hg19).

Finally we can convert to a `RatioSet` by `ratioConvert()`. The two functions `ratioConvert()` and `mapToGenome()` commute, as shown by the class hierarchy diagram above. Many preprocessing functions goes through several steps in the diagram, for example if the function needs to know the genomic location of the probes (several preprocessing functions handle probes measured on the sex chromosomes in a different way).

Reading data

This package supports analysis of IDAT files, containing the summarized bead information.

In our experience, most labs use a "Sample Sheet" CSV file to describe the layout of the experiment. This is based on a sample sheet file provided by Illumina. Our pipeline assumes the existence of such a file(s), but it is relatively easy to create such a file using for example Excel, if it is not available.

We use an example dataset with 6 samples, spread across two slides. First we obtain the system path to the IDAT files; this requires a bit since the data comes from an installed package

```
baseDir <- system.file("extdata", package = "minfiData")
list.files(baseDir)
## [1] "5723646052"      "5723646053"      "SampleSheet.csv"
```

This shows the typical layout of 450k data: each "slide" (containing 12 arrays, see Termiology) is stored in a separate directory, with a numeric name. The top level directory contains the sample sheet file. Inside the slide directories we find the IDAT files (and possible a number of JPG images or other files):

```
list.files(file.path(baseDir, "5723646052"))
## [1] "5723646052_R02C02_Grn.idat" "5723646052_R02C02_Red.idat"
## [3] "5723646052_R04C01_Grn.idat" "5723646052_R04C01_Red.idat"
## [5] "5723646052_R05C02_Grn.idat" "5723646052_R05C02_Red.idat"
```

The files for each array has another numeric number and consists of a Red and a Grn (Green) IDAT file. Note that for

this example data, each slide contains only 3 arrays and not 12. This was done because of file size limitations and because we only need 6 arrays to illustrate the package's functionality.

First we read the sample sheet. We provide a convenience function for reading in this file `read.metharray.sheet()`. This function has a couple of attractive bells and whistles. Let us look at the output

```
targets <- read.metharray.sheet(baseDir)
## [read.metharray.sheet] Found the following CSV files:
## [1] "/home/biocbuild/bbs-3.5-bioc/R/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 /home/biocbuild/bbs-3.5-bioc/R/library/minfiData/extdata/5723646052/5723646052_R02C02
## 2 /home/biocbuild/bbs-3.5-bioc/R/library/minfiData/extdata/5723646052/5723646052_R04C01
## 3 /home/biocbuild/bbs-3.5-bioc/R/library/minfiData/extdata/5723646052/5723646052_R05C02
## 4 /home/biocbuild/bbs-3.5-bioc/R/library/minfiData/extdata/5723646053/5723646053_R04C02
## 5 /home/biocbuild/bbs-3.5-bioc/R/library/minfiData/extdata/5723646053/5723646053_R05C02
## 6 /home/biocbuild/bbs-3.5-bioc/R/library/minfiData/extdata/5723646053/5723646053_R06C02
```

First the output: this is just a `data.frame`. It contains a column `Basename` that describes the location of the IDAT file corresponding to the sample, as well as two columns `Array` and `Slide`. In the sample sheet provided by Illumina, these two columns are named `Sentrix_Position` and `Sentrix_ID`, but we rename them. We provide more detail on the use of this function below. The `Basename` column tend to be too large for display, here it is simplified relative to `baseDir`:

```
sub(baseDir, "", targets$Basename)
## [1] "/5723646052/5723646052_R02C02" "/5723646052/5723646052_R04C01"
## [3] "/5723646052/5723646052_R05C02" "/5723646053/5723646053_R04C02"
## [5] "/5723646053/5723646053_R05C02" "/5723646053/5723646053_R06C02"
```

(This is just for display purposes).

With this `data.frame`, it is easy to read in the data

```
RGset <- read.metharray.exp(targets = targets)
```

Let us look at the associated pheno data, which is really just the information contained in the targets object above.

```
RGset
## class: RGChannelSet
## dim: 622399 6
## metadata(0):
## assays(2): Green Red
## rownames(622399): 10600313 10600322 ... 74810490 74810492
## rowData names(0):
## colnames(6): 5723646052_R02C02 5723646052_R04C01 ...
```

```
##   5723646053_R05C02 5723646053_R06C02
## colData names(13): Sample_Name Sample_Well ... Basename filenames
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
pd <- pData(RGset)
pd[,1:4]
## DataFrame with 6 rows and 4 columns
##                   Sample_Name Sample_Well Sample_Plate Sample_Group
##                   <character> <character>  <character>  <character>
## 5723646052_R02C02    GroupA_3          H5           NA       GroupA
## 5723646052_R04C01    GroupA_2          D5           NA       GroupA
## 5723646052_R05C02    GroupB_3          C6           NA       GroupB
## 5723646053_R04C02    GroupB_1          F7           NA       GroupB
## 5723646053_R05C02    GroupA_1          G7           NA       GroupA
## 5723646053_R06C02    GroupB_2          H7           NA       GroupB
```

The `read.metharray.exp()` function also makes it possible to read in an entire directory or directory tree (with `recursive` set to `TRUE`) by using the function just with the argument `base` and `targets=NULL`, like

```
RGset2 <- read.metharray.exp(file.path(baseDir, "5723646052"))
RGset3 <- read.metharray.exp(baseDir, recursive = TRUE)
```

Advanced notes on Reading Data

The only important column in sheet `data.frame` used in the `targets` argument for the `read.metharray.exp()` function is a column names `Basename`. Typically, such an object would also have columns named `Array`, `Slide`, and (optionally) `Plate`.

We used sheet data files build on top of the Sample Sheet data file provided by Illumina. This is a CSV file, with a header. In this case we assume that the phenotype data starts after a line beginning with `[Data]` (or that there is no header present).

It is also easy to read a sample sheet manually, using the function `read.csv()`. Here, we know that we want to skip the first 7 lines of the file.

```
targets2 <- read.csv(file.path(baseDir, "SampleSheet.csv"),
                     stringsAsFactors = FALSE, skip = 7)
targets2
##   Sample_Name Sample_Well Sample_Plate Sample_Group Pool_ID Sentrix_ID
## 1    GroupA_3          H5           NA       GroupA      NA 5723646052
## 2    GroupA_2          D5           NA       GroupA      NA 5723646052
## 3    GroupB_3          C6           NA       GroupB      NA 5723646052
## 4    GroupB_1          F7           NA       GroupB      NA 5723646053
## 5    GroupA_1          G7           NA       GroupA      NA 5723646053
## 6    GroupB_2          H7           NA       GroupB      NA 5723646053
##   Sentrix_Position person age sex status
## 1           R02C02    id3  83   M normal
## 2           R04C01    id2  58   F normal
## 3           R05C02    id3  83   M cancer
## 4           R04C02    id1  75   F cancer
## 5           R05C02    id1  75   F normal
## 6           R06C02    id2  58   F cancer
```

We now need to populate a `Basename` column. On possible approach is the following

```
targets2$Basename <- file.path(baseDir, targets2$Sentrix_ID,
                               paste0(targets2$Sentrix_ID,
                                      targets2$Sentrix_Position))
```

Finally, MINFI contains a file-based parser: `read.metharray()`. The return object represents the red and the green channel measurements of the samples. A useful function that we get from the package *Biobase* is `combine()` that combines ("adds") two sets of samples. This allows the user to manually build up an `RGChannelSet`.

Manifest / annotation

What everyone needs to know

For a methylation array, we have two types of annotation packages: "manifest" packages which contains the array design and "annotation" packages which contains information about where the methylation loci are located on the genome, which genomic features they map to and possible whether they overlap any known SNPs.

You can see which packages are being used by

```
annotation(RGsetEx)
##                           array                      annotation
## "IlluminaHumanMethylation450k"                  "ilmn12.hg19"
```

Advanced discussion

This discussion is intended for package developers or users who want to understand the internals of MINFI.

A set of 450k data files will initially be read into an `RGChannelSet`, representing the raw intensities as two matrices: one being the green channel and one being the red channel. This is a class which is very similar to an `ExpressionSet` or an `NChannelSet`. The `RGChannelSet` is, together with a `IlluminaMethylationManifest` object, preprocessed into a `MethylSet`. The `IlluminaMethylationManifest` object contains the array design, and describes how probes and color channels are paired together to measure the methylation level at a specific CpG. The object also contains information about control probes (also known as QC probes). The `MethylSet` contains normalized data and essentially consists of two matrices containing the methylated and the unmethylated evidence for each CpG. Only the `RGChannelSet` contains information about the control probes.

The process described in the previous paragraph is very similar to the paradigm for analyzing Affymetrix expression arrays using the *affy* package (an `AffyBatch` is preprocessed into an `ExpressionSet` using array design information stored in a CDF environment (package)).

Quality control

- use shinyMethyl
- minfiQC
- getSex checks
- mds plots
- plot conversion probes

Preprocessing

Preprocessing in MINFI is done by a series of functions with names like `preprocessXX`. Different functions has different classes as output.

Currently, we have

- `preprocessRaw` : No processing.
- `preprocessIllumina` : Illumina preprocessing, as performed by Genome Studio (reverse engineered by us).
- `preprocessSWAN` : SWAN normalization, described in (Maksimovic, Gordon, and Oshlack 2012).
- `preprocessQuantile` : Quantile normalization (adapted to DNA methylation arrays), described in (Touleimat and Tost 2012, Aryee et al. (2014))
- `preprocessNoob` : Noob preprocessing, described in (Triche et al. 2013).
- `preprocessFunnorm` : Functional normalization as described in (Fortin et al. 2014).

FIXME: discuss literature

SNPs and other issues

- SNPs
- cross reactive probes
- remove bad probes
- detection P
- Gap Hunting

Identifying differentially methylated regions

- DMP finding
- Bump hunting
- Block finding
- plot DMRs with annotation.

Correcting for cell type heterogenity

- estimateCellTypes
- reference packages

Other stuff

- Horvath age estimation
- other packages

Sessioninfo

- R version 3.4.0 (2017-04-21), `x86_64-pc-linux-gnu`

- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8, LC_COLLATE=C, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_US.UTF-8, LC_PAPER=en_US.UTF-8, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=en_US.UTF-8, LC_IDENTIFICATION=C
- Running under: Ubuntu 16.04.2 LTS
- Matrix products: default
- BLAS: /home/biocbuild/bbs-3.5-bioc/R/lib/libRblas.so
- LAPACK: /home/biocbuild/bbs-3.5-bioc/R/lib/libRlapack.so
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: Biobase 2.36.0, BiocGenerics 0.22.0, BiocStyle 2.4.0, Biostrings 2.44.0, DelayedArray 0.2.0, GenomeInfoDb 1.12.0, GenomicRanges 1.28.0, IRanges 2.10.0, IlluminaHumanMethylation450kanno.ilmn12.hg19 0.6.0, IlluminaHumanMethylation450kmanifest 0.4.0, S4Vectors 0.14.0, SummarizedExperiment 1.6.0, XVector 0.16.0, bumphunter 1.16.0, foreach 1.4.3, iterators 1.0.8, locfit 1.5-9.1, matrixStats 0.52.2, minfi 1.22.1, minfiData 0.22.0
- Loaded via a namespace (and not attached): AnnotationDbi 1.38.0, BiocParallel 1.10.0, DBI 0.6-1, GEOquery 2.42.0, GenomeInfoDbData 0.99.0, GenomicAlignments 1.12.0, GenomicFeatures 1.28.0, MASS 7.3-47, Matrix 1.2-10, R6 2.2.0, RColorBrewer 1.1-2, RCurl 1.95-4.8, RSQLite 1.1-2, Rcpp 0.12.10, Rsamtools 1.28.0, XML 3.98-1.6, annotate 1.54.0, backports 1.0.5, base64 2.0, beanplot 1.2, biomaRt 2.32.0, bitops 1.0-6, codetools 0.2-15, compiler 3.4.0, data.table 1.10.4, digest 0.6.12, doRNG 1.6.6, evaluate 0.10, genefilter 1.58.0, grid 3.4.0, htmltools 0.3.6, httr 1.2.1, illuminaio 0.18.0, knitr 1.15.1, lattice 0.20-35, limma 3.32.2, magrittr 1.5, mclust 5.2.3, memoise 1.1.0, multtest 2.32.0, nlme 3.1-131, nor1mix 1.2-2, openssl 0.9.6, pkgmaker 0.22, plyr 1.8.4, preprocessCore 1.38.0, quadprog 1.5-5, registry 0.3, reshape 0.8.6, rmarkdown 1.5, rngtools 1.2.4, rprojroot 1.2, rtracklayer 1.36.0, siggenes 1.50.0, splines 3.4.0, stringi 1.1.5, stringr 1.2.0, survival 2.41-3, tools 3.4.0, xtable 1.8-2, yaml 2.1.14, zlibbioc 1.22.0

References

Aryee, Martin J, Andrew E Jaffe, Hector Corrada Bravo, Christine Ladd-Acosta, Andrew P Feinberg, Kasper D Hansen, and Rafael A Irizarry. 2014. "Minfi: a flexible and comprehensive Bioconductor package for the analysis of Infinium DNA methylation microarrays." *Bioinformatics* 30 (10): 1363–9. doi:10.1093/bioinformatics/btu049.

Fortin, Jean-Philippe, and Kasper D Hansen. 2015. "Reconstructing A/B compartments as revealed by Hi-C using long-range correlations in epigenetic data." *Genome Biology* 16: 180. doi:10.1186/s13059-015-0741-y.

Fortin, Jean-Philippe, Aurélie Labbe, Mathieu Lemire, Brent W Zanke, Thomas J Hudson, Elana J Frtig, Celia MT Greenwood, and Kasper D Hansen. 2014. "Functional normalization of 450k methylation array data improves replication in large cancer studies." *Genome Biology* 15 (11): 503. doi:10.1186/s13059-014-0503-2.

Fortin, Jean-Philippe, Timothy J Triche, and Kasper D Hansen. 2016. "Preprocessing, Normalization and Integration of the Illumina HumanMethylationEPIC Array." *BioRxiv*. doi:10.1101/065490.

Jaffe, Andrew E, Peter Murakami, Hwajin Lee, Jeffrey T Leek, M Daniele Fallin, Andrew P Feinberg, and Rafael A Irizarry. 2012. "Bump Hunting to Identify Differentially Methylated Regions in Epigenetic Epidemiology Studies." *International Journal of Epidemiology* 41 (1): 200–209. doi:10.1093/ije/dyr238.

Maksimovic, Jovana, Lavinia Gordon, and Alicia Oshlack. 2012. "SWAN: Subset quantile Within-Array Normalization for Illumina Infinium HumanMethylation450 BeadChips." *Genome Biology* 13 (6): R44. doi:10.1186/gb-2012-13-6-r44.

Touleimat, Nizar, and Jörg Tost. 2012. "Complete Pipeline for Infinium() Human Methylation 450K BeadChip Data Processing Using Subset Quantile Normalization for Accurate DNA Methylation Estimation." *Epigenomics* 4 (3): 325–41. doi:10.2217/epi.12.21.

Triche, Timothy J, Daniel J Weisenberger, David Van Den Berg, Peter W Laird, and Kimberly D Siegmund. 2013. "Low-level processing of Illumina Infinium DNA Methylation BeadArrays." *Nucleic Acids Research* 41 (7): e90. doi:10.1093/nar/gkt090.