Normalizing Illumina Infinium Human Methylation 450k for multiple cell types with funtooNorm

Celia Greenwood[1,2], Stepan Grinek[1], Raphael Poujol[1], Maxime Turgeon[2], Kathleen Oros Klein[1]

[1] Lady Davis Institute of Research, Montreal Quebec. [2] McGill University, Montreal Quebec.

2017-10-02

Contents

Introduction

The `funtooNorm` package provides a function for normalization of Illumina Infinium Human Methylation 450 BeadChip (Illumina 450K) data when there are samples from multiple tissues or cell types. The algorithm in this package represents an extension of a normalization method introduced recently by (Fortin et al. 2014; Aryee et al. 2014). In addition to the percentile-specific adjustments implemented in funNorm, funtooNorm is specifically designed to allow flexibility in the adjustments for different cell types or tissue types. Percentiles of methylation levels may vary across cell types and hence the original funNorm may not be ideal if applied to all samples simultaneously. Normalization separately for each cell type may introduce unwanted variability if sample sizes are small. Therefore, this algorithm provides flexibility while optimizing the sample size used to estimate the corrections.

Note that the current version of the package does not do a good job of normalizing the Y chromosome probes; the funNorm method performs better. In a subsequent version of the package we will address this issue.

Normalization

Terminology

As described in the `minfi` vignette, the 450k array contains two types of probes:

```
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.
```

Therefore, we separate the 6 types of signals in our method : *AIGrn*, *BIGrn*, *AIRed*, *BIRed*, *AII* and *BII*, where the **A** (methylated) and **B** (unmethylated) are on **Green** or **Red** channel depending on the 3 types : **Type I Red**, **Type I Green** or **Type II**. We will be careful to use position when referring to a CpG and not probe since the number of probes per position depends on the probe type of these position.

Loading Raw Data

The first step is to create a new SampleSet for your data. The SampleSet is an object of class S3, defined for the purpose of running the funtooNorm algorithm. The SampleSet object contains the chip data and the cell type for each sample. There are two ways to load your data into R, depending on whether your files our in .csv or .idat format.

1. **csv files**: csv files are generated from the *Genome Studio* software. After loading the data into Genome Studio, export the control probes file and the signal intensity file. These csv files will contain the sample names as well as column headers. Both files should contain the exact same samples in the same order.

   Use the function `fromGenStudFiles`. The first argument should be the control probe file, the second the signal file and last argument should be a vector containing the cell types. In order to avoid any assignment error, the vector elements should be named with the exact same sample labels as the Genome Studio files.

2. **idat files**: Using the *minfi* package, create a *RGChannelSet* object containing all your samples and use the function *fromRGChannelSet* to create your *SampleSet*. Please refer to the *minfi* vignette on how to create a *RGChannelSet*. The phenotype data of your object should contain a column name *cell_type*, you can access it using the `pData` function

There must be at least two different cell or tissue types in the data or the program will terminate with a warning.

Example

We have provided a small data set containing $N = 6$ samples from the Illumina 450K to demonstrate `funtooNorm`. Since the sample are all the same cell type, this example will not produce meaningful results.

```
require(funtooNorm)
require(minfiData)
# We randomly assign cell types for the purpose of this example.
pData(RGsetEx)$cell_type <- rep(c("type1","type2"),3)
mySampleSet=fromRGChannelSet(RGsetEx)
```

The above script prepares your sampleSet object and your data is ready for normalization. Raw beta values can be extracted at this point. get the Beta value before normalization.

```
origBeta <- getRawBeta(mySampleSet)
origBeta[1:3,1:3]
```

```
##            5723646052_R02C02 5723646052_R04C01 5723646052_R05C02
## cg00008945        0.01681778         0.4511749        0.06168549
## cg00026186        0.01123767         0.5445124        0.02027104
## cg00072288        0.02706664         0.5885229        0.01681831
```

Before normalizing with funtooNorm you need to choose the ideal number of components, `ncmp`, for your data. We have set 4 as the default value for `ncmp`.

Choice of the number of components can be facilitated by examining a series of fits with different numbers of components: Calling the `plotValidationGraph` function with `type.fits = "PCR"` produces a set of plots, showing the root mean squared errors from cross-validated fits, for different numbers of components, separately for each type of signal AIGrn, BIGrn, AIRed, BIRed, AII, and BII. The ideal `ncmp` would be the smallest value where the cross-validated root mean squared error is fairly small across all the quantiles.

Figure 1. Cross-validated root mean squared errors across percentiles of the signal distributions for different numbers of PCR components. signal A and B; probe type I red, probe type I green and probe type II.

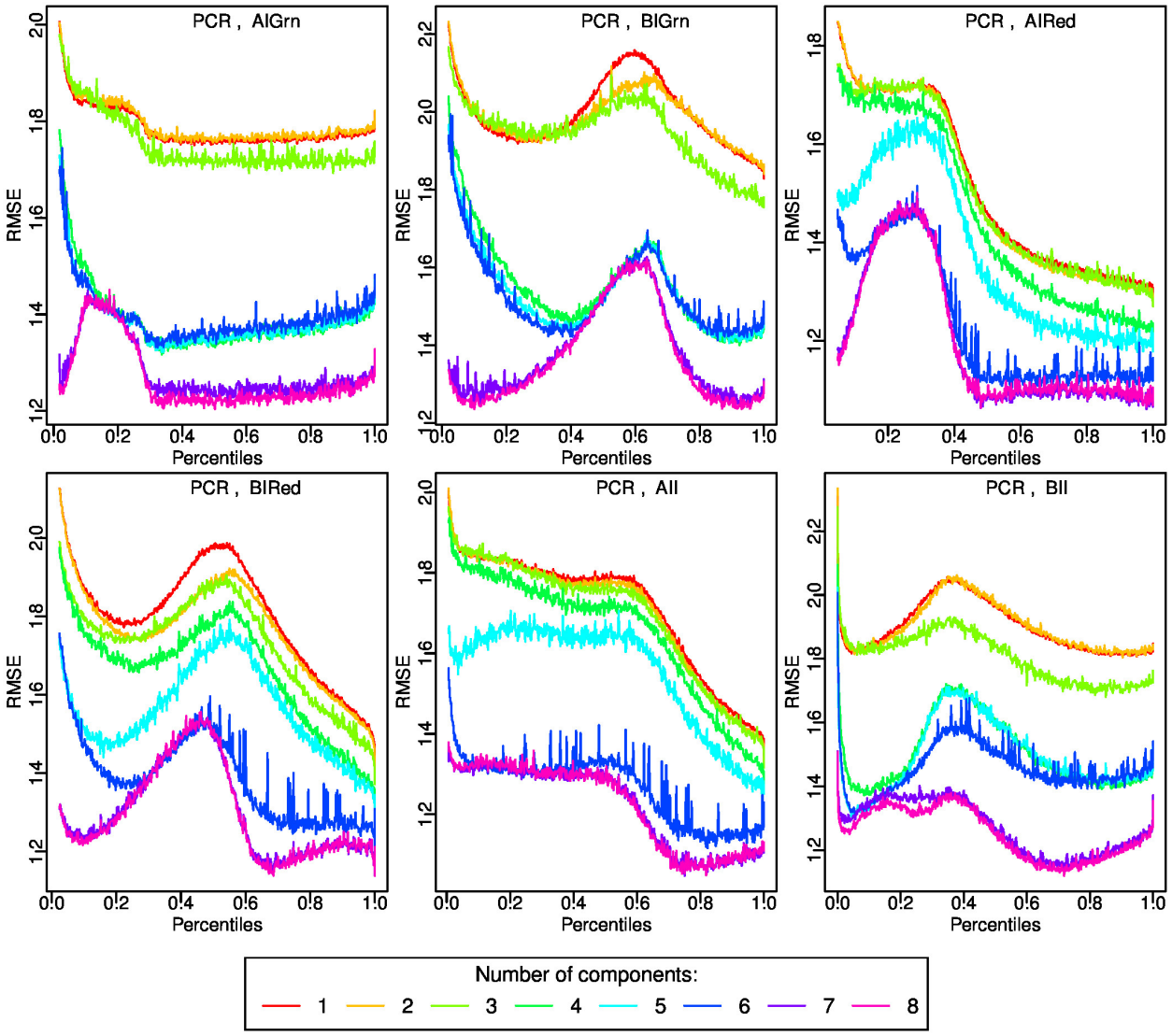

Figure 1: valPCR

By default, `funtooNorm` will perform 10-fold cross-validation, but this can be changed with the parameter `ncv.fold`. Since this step can be very lengthy, we advice you to set the output of your plot to a pdf file: `file = "validationcurve.pdf"`. The default fit type is *PCR*, you can change it with `type.fits="PLS"`.

```
    plotValidationGraph(mySampleSet, type.fits="PCR")
```

```
## pdf
##   2
```

Below is a basic call to the function `funtooNorm`: `funtooNorm` will fit either principal component regression (PCR) or partial least squares regression (PLS) by specifying `type.fits="PCR"` or `type.fits="PLS"`. The default is set to **PCR**, to match funNorm. An important user-chosen parameter is `ncmp`, the number of components to be included in either of these two models; these components are calculated from the control probe data and cell type data.

```
    mySampleSet=funtooNorm(mySampleSet,type.fits="PCR",ncmp=3)
    mySampleSet
```

```
## SampleSet object built from  minfi
## Data:  485577 positions and  6 samples
##    cell type: type1 type2
##    528 quantiles
## funtooNorm Normalization was applied
```

```
    normBeta <- getNormBeta(mySampleSet)
    normBeta[1:3,1:3]
```

```
##            5723646052_R02C02 5723646052_R04C01 5723646052_R05C02
## cg00008945        0.02311210         0.4619684        0.06307128
## cg00026186        0.01359405         0.5495726        0.02009510
## cg00072288        0.03179749         0.5957099        0.01635315
```

To assess the performance of the normalization, if technical replicates are available, one can use a measure of intra-replicate differences **M**, described in (Oros Klein et al. 2016). We provide a function `agreement` to implement calculation of this measure. The function requires two arguments: a matrix of beta values and a vector of individual ID's. Technical replicates will have identical individual ID's. The returned value of **M** is expected to be more similar for the data after normalization:

```
    #technical replicates are fictional, just for demonstration purposes.
    agreement(origBeta, c(1:5,5)) # M for data before the normalization
```

```
## [1] 0.0281463
```

```
    agreement(normBeta, c(1:5,5)) # M for data after normalization
```

```
## [1] 0.02351814
```

FuntooNorm and the minfi package

The `minfi` package (Aryee et al. 2014) contains several tools for analyzing and visualizing Illumina's 450k array data. This section shows how to incorporate the `funtooNorm` and `minfi` packages.

Using `minfi` to find differentially methylated CpG

Here we demonstrate the use of `dmpFinder` on the M values: $log(Meth/nmeth)$

```
library(minfi)

    age=pData(RGsetEx)$age
```

```
    dmp=dmpFinder(getNormM(mySampleSet), age, type="continuous")
    dmp[1:2,]
```

```
##            intercept       beta        t         pval      qval
## cg08422471  1.412644 0.02978811 36.31539 3.432385e-06 0.9999887
## cg19137649 -3.784871 0.07245625 31.37784 6.147863e-06 0.9999887
```

Creating GenomicRatioSet

Since normalization is rarely the final goal, this section illustrates how to convert the output of `funtooNorm` (the `funtooNorm` object created in section 2.3) to a `GenomicRatioSet` object, so that it can be used by other tools in `minfi::bumphunter` or `minfi::blockFinder`.

First we extract the phenotype data provided in the RGset from the initial `minfi` example. Then we create the GenomicRatioSet which includes the normalized values and the phenotype data.

```
phenoData <- pData(RGsetEx)[,c("age","sex","status")]
genomerange <- getGRanges(mySampleSet)
grs <- GenomicRatioSet(gr=genomerange,
                       Beta=normBeta,
                       preprocessMethod="funtooNorm",
                       metadata=list(pData=phenoData))
grs
```

```
## class: GenomicRatioSet
## dim: 485512 6
## metadata(1): pData
## assays(1): Beta
## rownames(485512): cg00008945 cg00026186 ... cg27611726 cg27636129
## rowData names(0):
## colnames(6): 5723646052_R02C02 5723646052_R04C01 ...
##   5723646053_R05C02 5723646053_R06C02
## colData names(0):
## Annotation
##   array:
## Preprocessing
##   Method: NA
##   minfi version: NA
##   Manifest version: NA
```

The default print method of a `GenomicRatioSet` object shows basic information of that object. In this example things were kept simple in order to show the bare necessities.

```
## R Under development (unstable) (2017-09-18 r73311)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.3 LTS
##
## Matrix products: default
## BLAS: /usr/lib/libblas/libblas.so.3.6.0
## LAPACK: /usr/lib/lapack/liblapack.so.3.6.0
##
## locale:
##  [1] LC_CTYPE=en_CA.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_CA.UTF-8        LC_COLLATE=en_CA.UTF-8
##  [5] LC_MONETARY=en_CA.UTF-8    LC_MESSAGES=en_CA.UTF-8
```

```
##  [7] LC_PAPER=en_CA.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_CA.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] minfiData_0.23.0
##  [2] IlluminaHumanMethylation450kanno.ilmn12.hg19_0.6.0
##  [3] IlluminaHumanMethylation450kmanifest_0.4.0
##  [4] funtooNorm_1.1.1
##  [5] minfi_1.23.3
##  [6] bumphunter_1.17.2
##  [7] locfit_1.5-9.1
##  [8] iterators_1.0.8
##  [9] foreach_1.4.3
## [10] Biostrings_2.45.4
## [11] XVector_0.17.1
## [12] SummarizedExperiment_1.7.8
## [13] DelayedArray_0.3.19
## [14] matrixStats_0.52.2
## [15] Biobase_2.37.2
## [16] GenomicRanges_1.29.14
## [17] GenomeInfoDb_1.13.4
## [18] IRanges_2.11.16
## [19] S4Vectors_0.15.8
## [20] BiocGenerics_0.23.1
##
## loaded via a namespace (and not attached):
##  [1] httr_1.3.1               nor1mix_1.2-3
##  [3] bit64_0.9-7              splines_3.5.0
##  [5] assertthat_0.2.0         doRNG_1.6.6
##  [7] blob_1.1.0               GenomeInfoDbData_0.99.1
##  [9] Rsamtools_1.29.1         yaml_2.1.14
## [11] progress_1.1.2           RSQLite_2.0
## [13] backports_1.1.0          lattice_0.20-35
## [15] quadprog_1.5-5           limma_3.33.9
## [17] digest_0.6.12            RColorBrewer_1.1-2
## [19] htmltools_0.3.6          preprocessCore_1.39.3
## [21] Matrix_1.2-11            plyr_1.8.4
## [23] GEOquery_2.43.0          siggenes_1.51.0
## [25] XML_3.98-1.9             biomaRt_2.33.4
## [27] genefilter_1.59.0        zlibbioc_1.23.0
## [29] xtable_1.8-2             BiocParallel_1.11.8
## [31] annotate_1.55.0          openssl_0.9.7
## [33] tibble_1.3.4             beanplot_1.2
## [35] pkgmaker_0.22            GenomicFeatures_1.29.9
## [37] survival_2.41-3          magrittr_1.5
## [39] mclust_5.3               memoise_1.1.0
## [41] evaluate_0.10.1          nlme_3.1-131
## [43] MASS_7.3-47              data.table_1.10.4
## [45] tools_3.5.0              registry_0.3
```

```
## [47] prettyunits_1.0.2        stringr_1.2.0
## [49] rngtools_1.2.4           AnnotationDbi_1.39.3
## [51] base64_2.0               pls_2.6-0
## [53] compiler_3.5.0           rlang_0.1.2
## [55] grid_3.5.0               RCurl_1.95-4.8
## [57] bitops_1.0-6             rmarkdown_1.6
## [59] codetools_0.2-15         multtest_2.33.0
## [61] DBI_0.7                  reshape_0.8.7
## [63] R6_2.2.2                 illuminaio_0.19.0
## [65] GenomicAlignments_1.13.6 knitr_1.17
## [67] rtracklayer_1.37.3       bit_1.1-12
## [69] rprojroot_1.2            stringi_1.1.5
## [71] Rcpp_0.12.12
```

References

Aryee, M. J., A. E. Jaffe, H. Corrada-Bravo, C. Ladd-Acosta, A. P. Feinberg, K. D. Hansen, and R. A. Irizarry. 2014. "Minfi: A Flexible and Comprehensive Bioconductor Package for the Analysis of Infinium Dna Methylation Microarrays." Journal Article. *Bioinformatics* 30 (10): 1363–9. doi:10.1093/bioinformatics/btu049.

Fortin, J. P., A. Labbe, M. Lemire, B. W. Zanke, T. J. Hudson, E. J. Fertig, C. M. Greenwood, and K. D. Hansen. 2014. "Functional Normalization of 450k Methylation Array Data Improves Replication in Large Cancer Studies." Journal Article. *Genome Biol* 15 (12): 503. doi:10.1186/s13059-014-0503-2.

Oros Klein, K., S. Grinek, S. Bernatsky, L. Bouchard, A. Ciampi, I. Colmegna, J. P. Fortin, et al. 2016. "FuntooNorm: An R Package for Normalization of Dna Methylation Data When There Are Multiple Cell or Tissue Types." Journal Article. *Bioinformatics* 32 (4): 593–5. doi:10.1093/bioinformatics/btv615.