Using funtooNorm

Celia Greenwood, Raphaël Poujol, Stepan Grinek, Kathleen Oros Klein

March 28, 2016

1 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 [**?**, **?**]. 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.

2 Package use

2.1 Terminology

As the `minfi` vignette describes nicely, 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 dissociate 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 carefully talk about position when refering to a CpG and not about probe since the number of probes per position depends on the probe type of these position.

The **beta** value is computed with a default offset of `100` like the ILLUMINA standard but the offset can be easily change.

2.2 Reading Data

The package uses a `SampleSet` on which you can apply several functions. This SampleSet will contain your chip data and a matching cell type for each sample. The first step is to create a new SampleSet for your data. There are two ways to load your data to the package. Using the output of `GenomeStudio` or using raw `IDAT` files and the `minfi` package:

- `GenomeStudio:` In order to create a SampleSet from GenomeStudio output, use the function `fromGenStudFiles`. You should first generate the control probes file and the signal intensity file after loading your data in Genome Studio. Those 'csv' files will contain sample names as column headers. Both should contain the exact same samples in the same order. The last argument is 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 in the Genome Studio files.
- Directly `From IDAT:` Using the `minfi` package, you should 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 via `pData()`.

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

2.3 DataSet

We are using here the small data set containing $N = 6$ samples from the ILLUMINA 450K to demonstrate the usage of the package. Since the sample are not from different cell type, this example will not produce meaningfull results.

```
> require(funtooNorm)
> require(minfiData)
> # Here some fictive cell types are given for demonstration
> pData(RGsetEx)$cell_type <- rep(c("type1","type2"),3)
> mySampleSet=fromRGChannelSet(RGsetEx)
```

Now you have the sampleSet ready for normalisation. From here you can already get the Beta value before normalization. To change the default ILLUMINA offset, you can choose `offset=100`

```
>     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 normalizingwith funtooNorm you need to choose the ideal number of components 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. By looking at figure **??** the goal is to choose the smallest value of `ncmp` where the cross-validated root mean squared error is fairly small across all the quantiles. By default, funtooNorm will perform 10-fold cross-validation, but this can be changed with the parameter `ncv.fold`. Since this step can be vert long, we advice you o set the o ouput 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")
```

Below is a basic call to normalize this sample data set: 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:
   528 quantiles
funtooNorm Normalization was applied
>     normBeta <- getNormBeta(mySampleSet)
>     normBeta[1:3,1:3]
```

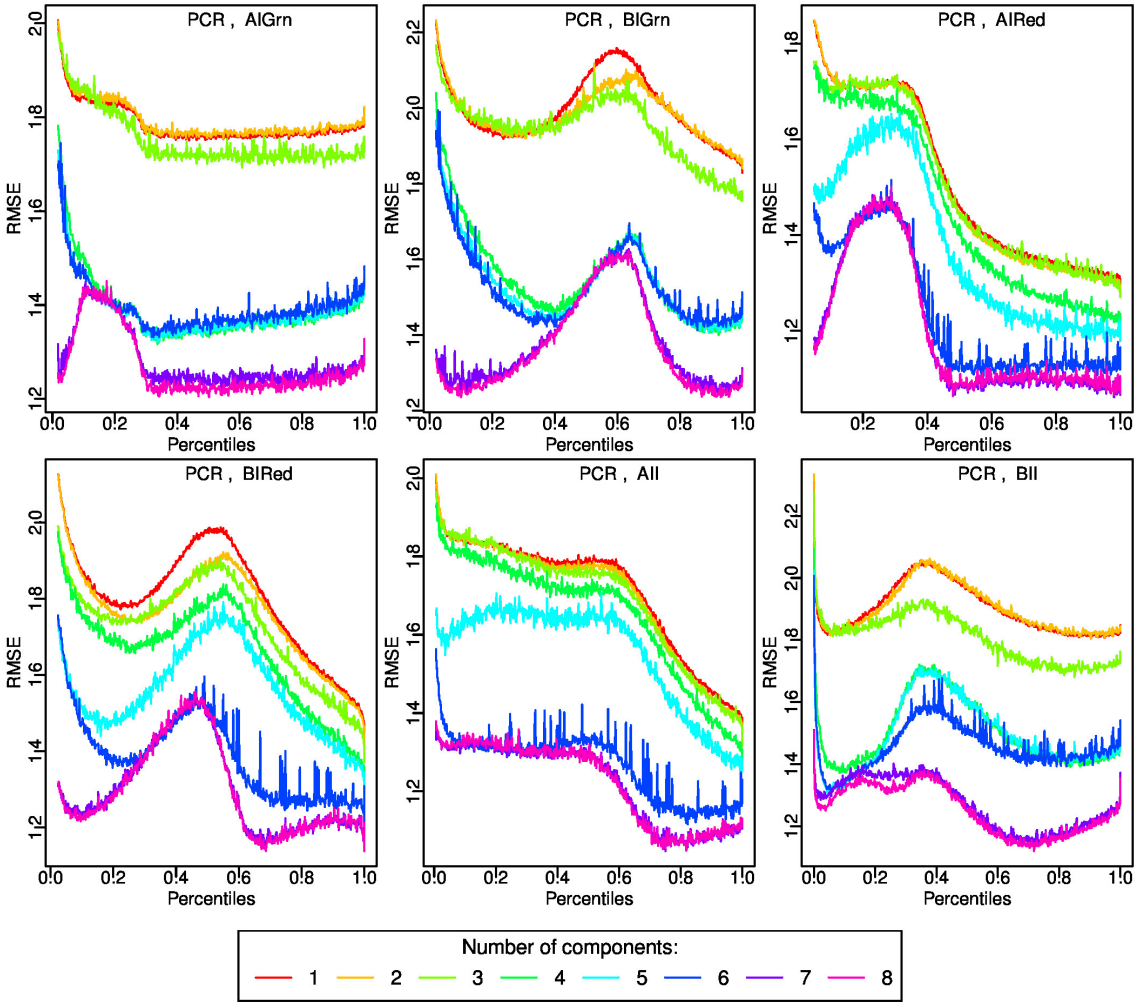

Figure 1: Cross-validated root mean squared errors across percentiles of the signal distributions for different numbers of PCR components. Top: signal A; Bottom: signal B; Left: probe type I red; Middle: probe type I green; Right: probe type II.

```
           5723646052_R02C02 5723646052_R04C01 5723646052_R05C02
cg00008945        0.01749639         0.4876271        0.06235737
cg00026186        0.01149692         0.5487109        0.01992674
cg00072288        0.02693873         0.5890657        0.01544062
```

To assess the performance of the normalization, one can use a measure of intra-replicate differences `M`, described in [funtooNorm]. We provide a function `agreement` implementing this measure. It takes as arguments a matrix of beta values and a vector of individual ID's. For the function to work some elements of the individual ID's vector should be identical, indicating technical replicates. The returned value of `M` is expected to be more similar for the data before and after normalization:

```
>     # Again, technical replicates are fictive, just for demonstration
>     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] 0.02796826
```

3 FuntooNorm and the minfi package

The `minfi` package [**?**] contains several tools for analyzing and visualizing Illumina's 450k array data. This section shows some interoperability of this package with the `funtooNorm` package.

```
> library(minfi)
```

3.1 Using minfi to find differentially methylated CpG

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

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

           intercept        beta         t         pval      qval
cg18762849  2.402393  0.03393986  41.00073 2.114777e-06 0.9941387
cg10426951  3.316091 -0.03065413 -31.48570 6.064335e-06 0.9941387
```

3.2 Creating GenomicRatioSet

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

A `GenomicRatioSet` object requires some phenotype information, so the following extract age sex and disease status from the inital `minfi` example.

```
> phenoData <- pData(RGsetEx)[,c("age","sex","status")]

> genomerange <- getGRanges(mySampleSet)
> grs <- GenomicRatioSet(gr=genomerange,
+                        Beta=normBeta,
+                        preprocessMethod="funtooNorm",
+                        pData=phenoData)
>
```

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.

```
> grs

class: GenomicRatioSet
dim: 485512 6
metadata(0):
assays(1): Beta
rownames(485512): cg00008945 cg00026186 ... cg27611726 cg27636129
rowRanges metadata column names(0):
colnames(6): 5723646052_R02C02 5723646052_R04C01 ... 5723646053_R05C02
  5723646053_R06C02
colData names(3): age sex status
Annotation
  array:
Preprocessing
  Method: NA
  minfi version: NA
  Manifest version: NA
```

4 EPIC dataset

The last version of this code is previous to the release of monfi that include

References

[1] Fortin, J.-P., et al. (2014). Functional normalization of 450K methylation array data improves replication in large cancer studies. *Genome Biology*, 15: p. 503.

[2] Aryee, M.J., et al. (2014). Minfi: a flexible and comprehensive Bioconductor package for the analysis of Infinium DNA methylation microarrays. *Bioinformatics*, 30(10): p. 1363-9.

[3] Smith M., et al. (2013). illuminaio: An open source IDAT parsing tool for Illumina microarrays. *F1000Research*, 2:264, 2013.

[4] Kathleen Oros Klein et al. (2015). *funtooNorm:* An improvement of the funNorm normalization method for methylation data from multiple cell or tissue types. Manuscript submitted.