# shinyMethyl: interactive visualization of Illumina 450K methylation arrays

Jean-Philippe Fortin, Kasper Daniel Hansen

April 24, 2017

## 1 Introduction

Up to now, more than 10,000 methylation samples from the state-of-the-art 450K microarray have been made available through The Cancer Genome Atlas portal [1] and the Gene Expression Omnibus (GEO) [2]. Large-scale comparison studies, for instance between cancers or tissues, become possible epigenome-widely. These large studies often require a substantial amount of time spent on preprocessing the data and performing quality control. For such studies, it is not rare to encounter significant batch effects, and those can have a dramatic impact on the validity of the biological results [3, 4]. With that in mind, we developed *shinyMethyl* to make the preprocessing of large 450K datasets intuitive, enjoyable and reproducible. *shinyMethyl* is an interactive visualization tool for Illumina 450K methylation array data based on the packages *minfi* and *shiny* [5, 6].

A few mouse clicks allow the user to appreciate insightful biological inter-array differences on a large scale. The goal of *shinyMethyl* is two-fold: (1) summarize a high-dimensional 450K array experiment into an exportable small-sized R object and (2) launch an interactive visualization tool for quality control assessment as well as exploration of global methylation patterns associated with different phenotypes.

Because a picture is worth a thousand words, we have implemented an example online of the interactive interface: http://spark.rstudio.com/jfortin/shinyMethyl/

## 2 Example dataset

To take a quick look at how the interactive interface of *shinyMethyl* works, we have included an example dataset in the companion package *shinyMethylData*. The dataset contains the extracted data of 369 Head and Neck cancer samples downloaded from The Cancer Genome Atlas (TCGA) data portal [1]: 310 tumor samples, 50 matched normals and 9 replicates of a control cell line. The first `shinyMethylSet` object (see Section 3 for the definition of a `shinyMethylSet` object) was created from the raw data (no normalization) and is stored under the name `summary.tcga.raw.rda`; the second `shinyMethylSet` object was created from a `GenomicRatioSet` containing the normalized data and the file is stored under

the name `summary.tcga.norm.rda`. The samples were normalized using functional normalization, a preprocessing procedure that we recently developed for heterogeneous methylation data [7].

To launch *shinyMethyl* with this TCGA dataset, simply type the following commands in a fresh *R* session:

```
library(shinyMethyl)
library(shinyMethylData)
runShinyMethyl(summary.tcga.raw, summary.tcga.norm)
```

*comment: The interactive interface will take a few seconds to be launched in your default HTML browser.*

# 3 Creating your own dataset visualization

In this section we describe how to launch an interactive visualization for your dataset. If you know and already have an `RGChannelSet` for your data, go to section a). If not, go to section b).

## a) For users familiar with RGChannelSet objects

If you are familiar with `RGChannelSet` objects from the *minfi* package, and if you already have an `RGChannelSet` for your experiment, you can launch *shinyMethyl* in two steps. For this tutorial purpose we will use the `RGChannelSet` stored in the package *minfiData* under the name `RGsetEx`:

```
library(minfiData)
```

We first summarize the 450K experiment with `shinySummarize`

```
library(shinyMethyl)
summary <- shinySummarize(RGsetEx)
```

```
## [shinySummarize] Extracting Red and Green channels
## [shinySummarize] Raw preprocessing
## [shinySummarize] Mapping to genome
## [shinySummarize] Computing quantiles
## [shinySummarize] Computing principal components
```

and then launch the interface with `runShinyMethyl`:

```
runShinyMethyl(summary)
```

To learn how to create an `RGChannelSet` object, go to section **b**.

## b) To create an RGChannelSet object

An `RGChannelSet` is an object defined in *minfi* containing the raw intensities of the green and red channels of your 450K experiment. To create an `RGChannelSet`, you will need to have the raw files of the experiment with extension .IDAT (we refer to those as .IDAT files). In case you do not have these files, you might want to ask your collaborators or your processing core if they have those. You absolutely need them to both use the packages *minfi* and *shinyMethyl*. The vignette in *minfi* describes carefully how to read the data in for different scenarios and how to construct an `RGChannelSet`. Here, we show a quick way to create an `RGChannelSet` from the .IDAT files contained in the package *minfiData*.

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

We need to tell *R* which directory contains the .IDAT files and the experiment sheet:

```
baseDir <- system.file("extdata", package = "minfiData")
# baseDir <- "/home/yourDirectoryPath"
```

We also need to read in the experiment sheet:

```
targets <- read.450k.sheet(baseDir)
head(targets)
```

Finally, we construct the `RGChannelSet` using `read.450k.exp`:

```
RGSet <- read.450k.exp(base = baseDir, targets = targets)
```

The function `pData()` in *minfi* allows to see the phenotype data of the samples:

```
pd <- pData(RGSet)
head(pd)
```

Please see **c** to create a `shinyMethylSet` necessary to launch *shinyMethyl*.

## c) To create a shinyMethylSet object

*shinyMethyl* requires that you have already created an `RGChannelSet`. From the `RGChannelSet` created in the previous section, we create a `shinyMethylSet` by using the command `shinySummarize`

```
myShinyMethylSet <- shinySummarize(RGSet)
```

Please see **d** to launch `shinyMethyl`

## d) To launch the interactive interface

To launch a *shinyMethyl* session, simply pass your `shinyMethylSet` object to the `runShinyMethyl` function as follows:

```
runShinyMethyl(myShinyMethylSet)
```

# 4 How to use the different shinyMethyl panels

The different figures at the end of the vignette explain how to use each of the *shinyMethyl* panels.

# 5 Advanced option: visualization of normalized data

*shinyMethyl* also offers the possibility to visualize normalized data that are stored in a `GenomicRatioSet` object. For instance, suppose we normalize the data by using the quantile normalization algorithm implemented in *minfi* (this function returns a `GenomicRatioSet` object by default):

```
GRSet.norm <- preprocessQuantile(RGSet)
```

We can then create two separate `shinyMethylSet` objects corresponding to the raw and normalized data respectively:

```
summary   <- shinySummarize(RGSset)
summary.norm <- shinySummarize(GRSet.norm)
```

To launch the *shinyMethyl* interface, use `runShinyMethyl` with the first argument being the `shinyMethylSet` extracted from the raw data and the second argument being the `shinyMethylSet` extracted from the normalized data as follows:

```
runShinyMethyl(summary, summary.norm)
```

# 6 What does a shinyMethylSet contain?

A `shinyMethylSet` object contains several summary data from a 450K experiment: the names of the samples, a data frame for the phenotype, a list of quantiles for the M and Beta values, a list of quantiles for the methylated and unmethylated channels intensities and a list of quantiles for the copy numbers, the green and red intensities of different control probes, and the principal component analysis (PCA) results performed on the Beta values. One can access the different summaries by using the slot operator `@`. The slot names can be obtained with the function `slotNames` as follows:

```
library(shinyMethyl)
library(shinyMethylData)
slotNames(summary.tcga.raw)
```

```
##  [1] "sampleNames"     "phenotype"       "mQuantiles"      "betaQuantiles"
##  [5] "methQuantiles"   "unmethQuantiles" "cnQuantiles"     "greenControls"
```

```
##  [9] "redControls"     "pca"             "originObject"    "array"
```

For instance, one can retrieve the phenotype by

```
head(summary.tcga.raw@phenotype)
```

```
##                    gender caseControlStatus  plate position
## 5775446049_R01C01   MALE            Normal 577544   R01C01
## 5775446049_R01C02 FEMALE            Normal 577544   R01C02
## 5775446049_R02C01   MALE             Tumor 577544   R02C01
## 5775446049_R02C02   MALE             Tumor 577544   R02C02
## 5775446049_R03C01   MALE             Tumor 577544   R03C01
## 5775446049_R03C02 FEMALE             Tumor 577544   R03C02
```

*comment: shinyMethyl also contain different accessor functions to access the slots. Please see the manual for more information.*

## Session info

Here is the output of `sessionInfo` on the system on which this document was compiled:

- 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, 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 1.22.0, minfiData 0.21.1, shiny 1.0.2, shinyMethyl 1.12.0, shinyMethylData 0.109.0
- Loaded via a namespace (and not attached): AnnotationDbi 1.38.0, BiocParallel 1.10.0, BiocStyle 2.4.0, 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 1.2-9, 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, highr 0.6, htmltools 0.3.5, httpuv 1.3.3, httr 1.2.1, illuminaio 0.18.0, knitr 1.15.1, lattice 0.20-35, limma 3.32.0, magrittr 1.5, mclust 5.2.3, memoise 1.1.0, mime 0.5, 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 1.4, 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

[1] The Cancer Genome Atlas. *Data portal*, 2014. Online. URL: https://tcga-data.nci.nih.gov/tcga/.

[2] R. Edgar, M. Domrachev, and A. E. Lash. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res.*, 30(1):207–210, Jan 2002.

[3] Jeffrey T Leek, Robert B Scharpf, Héctor Corrada Bravo, David Simcha, Benjamin Langmead, W Evan Johnson, Donald Geman, Keith Baggerly, and Rafael A Irizarry. Tackling the widespread and critical impact of batch effects in high-throughput data. *Nature Reviews Genetics*, 11(10):733–739, 2010. doi:10.1038/nrg2825.

[4] Kristin N Harper, Brandilyn A Peters, and Mary V Gamble. Batch effects and pathway analysis: two potential perils in cancer studies involving dna methylation array analysis. *Cancer Epidemiol Biomarkers Prev*, 22(6):1052–60, 2013. doi:10.1158/1055-9965.EPI-13-0114.

[5] Martin J Aryee, Andrew E Jaffe, Hector Corrada Bravo, Christine Ladd-Acosta, Andrew P Feinberg, Kasper D Hansen, and Rafael A Irizarry. Minfi: a flexible and comprehensive Bioconductor package for the analysis of Infinium DNA methylation microarrays. *Bioinformatics*, 30(10):1363–1369, 2014. doi:10.1093/bioinformatics/btu049, PMID:24478339.

[6] RStudio and Inc. *shiny: Web Application Framework for R*, 2014. R package version 0.9.1. URL: http://CRAN.R-project.org/package=shiny.

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

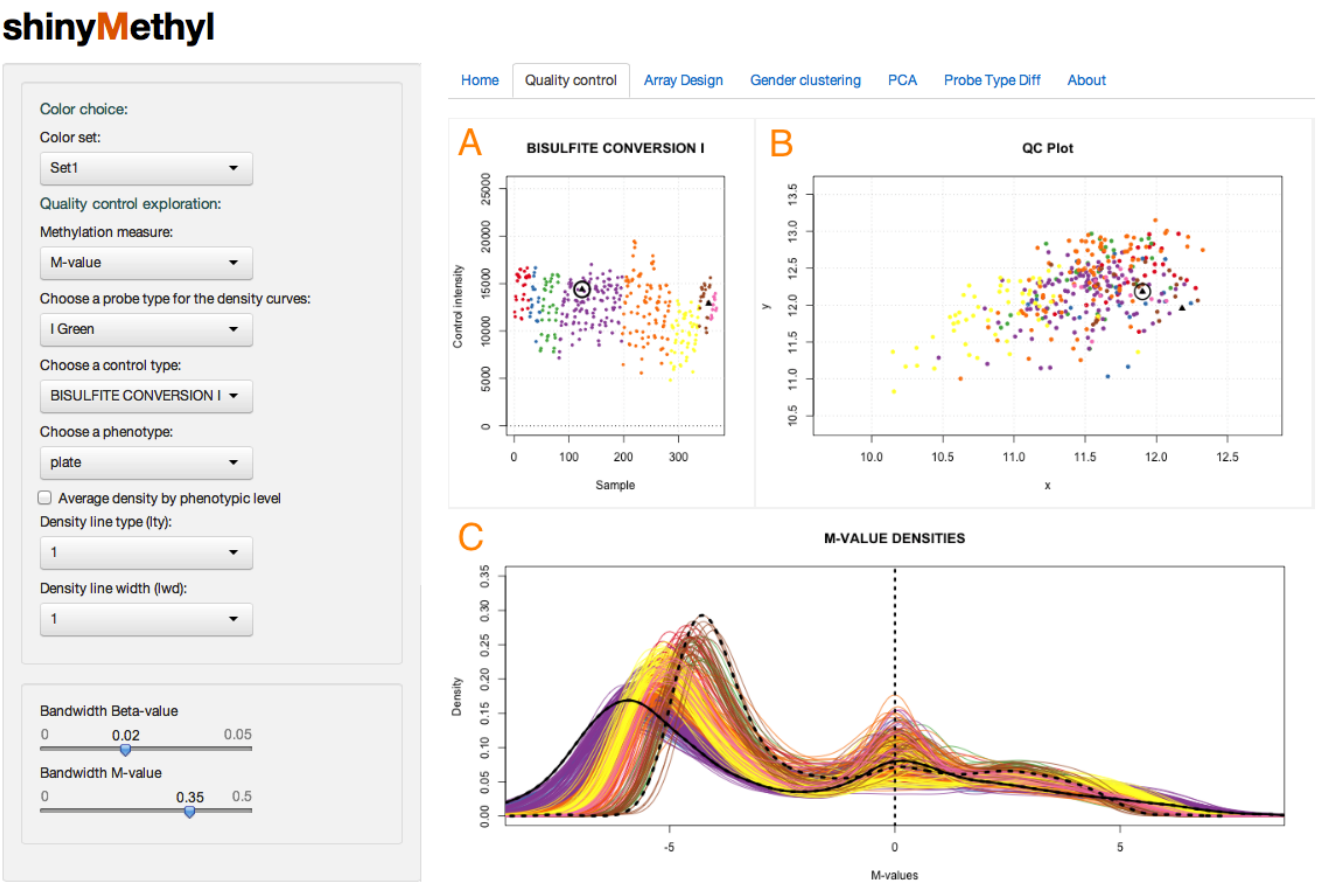


Figure 1: **Example of interactive visualization and quality control assessment** The three plots react simultaneously to the user mouse clicks and selected samples are represented in black. In this scenario, colors represent batch, but colors can be chosen to reflect the phenotype of the samples as well via the left-hand-side panel. The three different plots are: A) Plot of the quality controls probes reacting to the left-hand-side panel; the current plot shows the bisulfite conversion probes intensities. B) Quality control plot as implemented in *minfi*: the median intensity of the M channel against the median intensity of the U channel. Samples with bad quality would cluster away from the cloud shown in the current plot. For this dataset, all samples look good. C) Densities of the methylation intensities (can be chosen to be Beta-values or M-values, and can be chosen by probe type). The current plot shows the M-value densities for Infinium I probes, for the raw data. The dashed and solid lines in black correspond to the two samples selected by the user and match to the dots circled in black in the left-hand plots. The left-hand-side panel allows users to select different tuning parameters for the plots, as well as different phenotypes for the colors. The user can click on the samples that seem to have low quality, and can download the names of the samples in a csv file for further analysis (not shown in the screenshot).

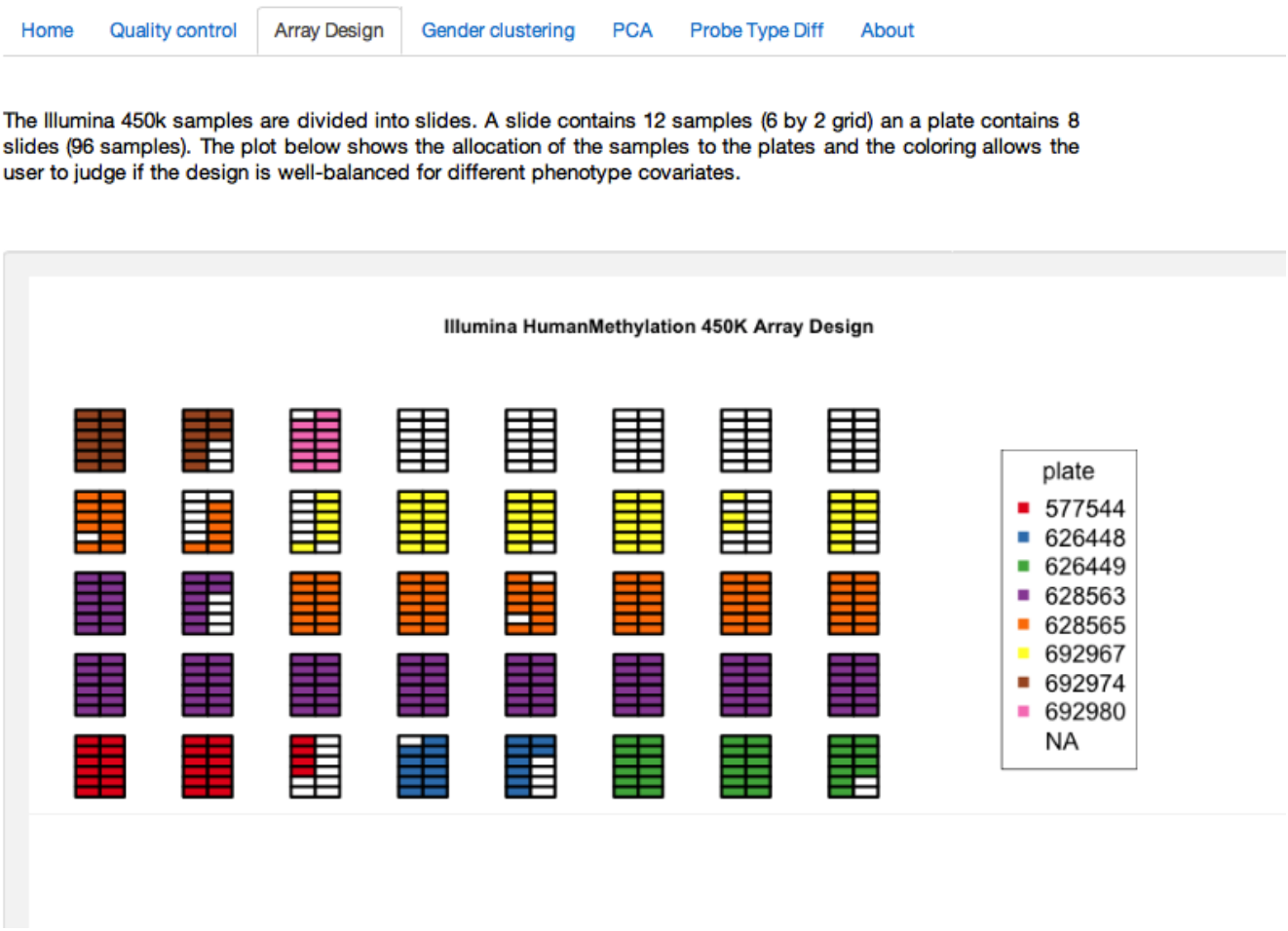


Figure 2: **Array design panel** The plot represents the physical slides (6 $\times$ 2 samples) on which the samples were assayed in the machine. The user can select the phenotype to color the samples. This plot allows to explore the quality of the randomization of the samples for a given phenotype.

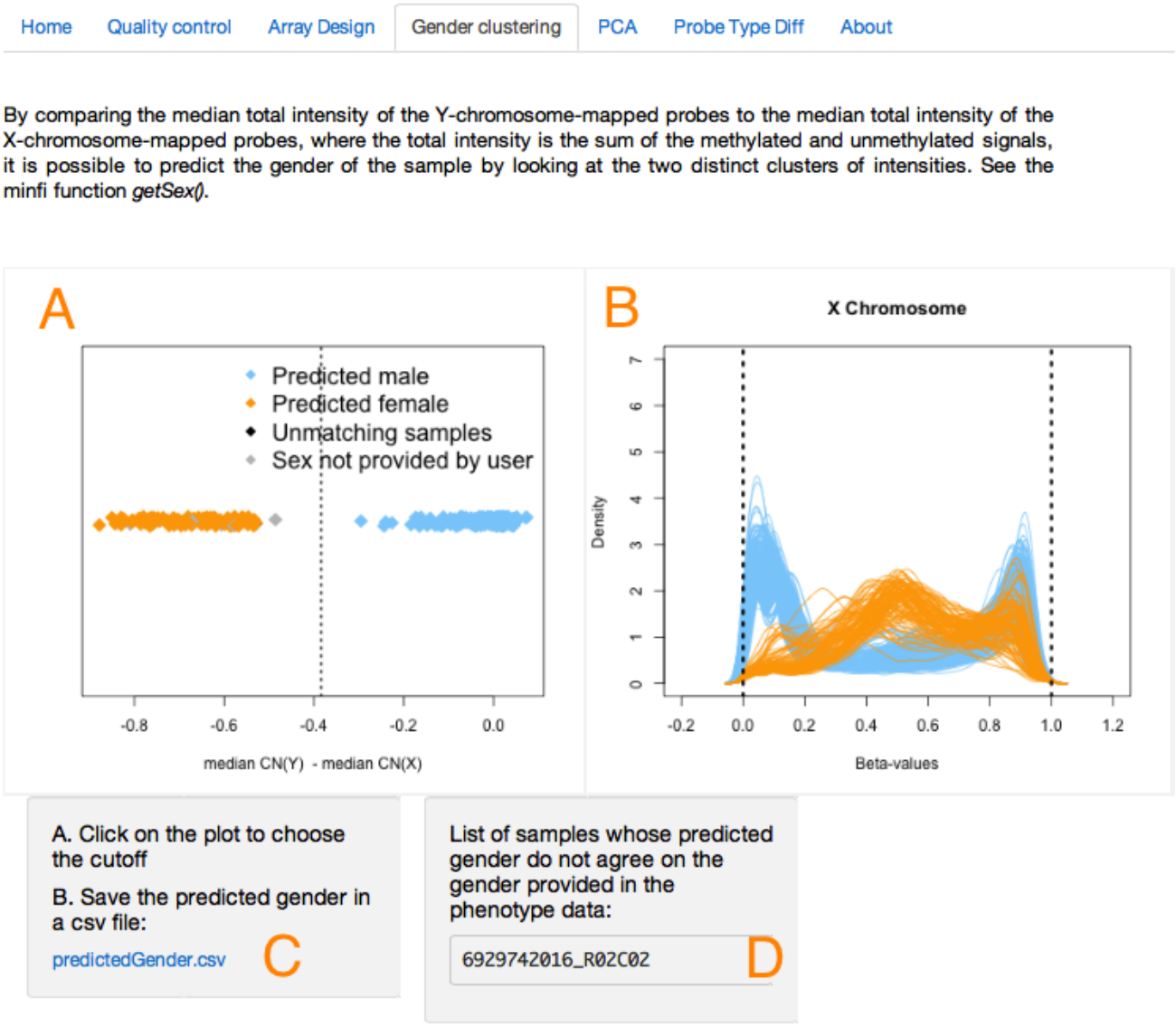


Figure 3: **Sex prediction algorithm panel** The difference of the median copy number intensity for the Y chromosome and the median copy number intensity for the X chromosome can be used to separate males and females. In A), the user can select the vertical cutoff (dashed line) manually with the mouse to separate the two clusters (orange for females, blue for males). Corresponding Beta-value densities appear in B) for further validation. The predicted sex can be downloaded in a csv file in C), and samples for which the predicted sex differs from the sex provided in the phenotype will appear in D).


Figure 4: **Principal component analysis (PCA) panel.** The user can select the principal components to visualize (PC1 and PC2 are shown in the current plot) and can choose the phenotype for the coloring. In the present plot, one can observe that the first two principal components distinguish tumor samples from normal samples for the TCGA example dataset (see example dataset section).

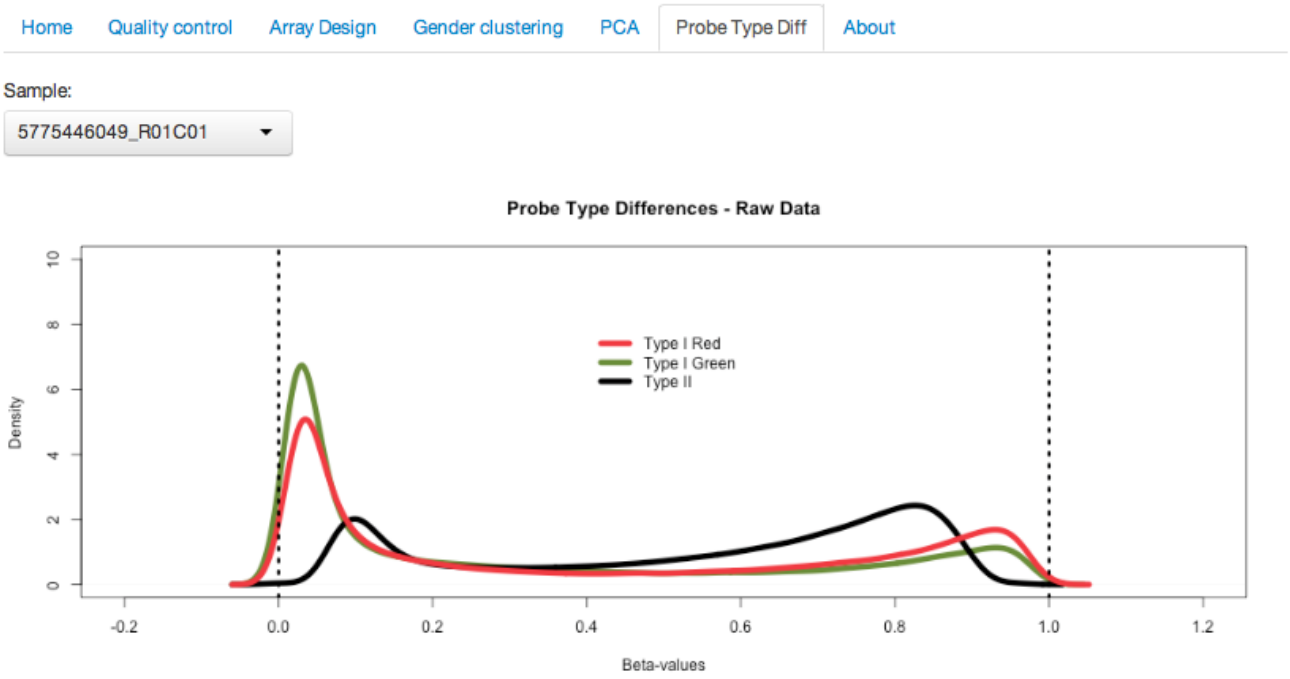


Figure 5: **Type I/II probe bias** The distributional differences between the Type I and Type II probes can be observed for each sample (selected by the user).

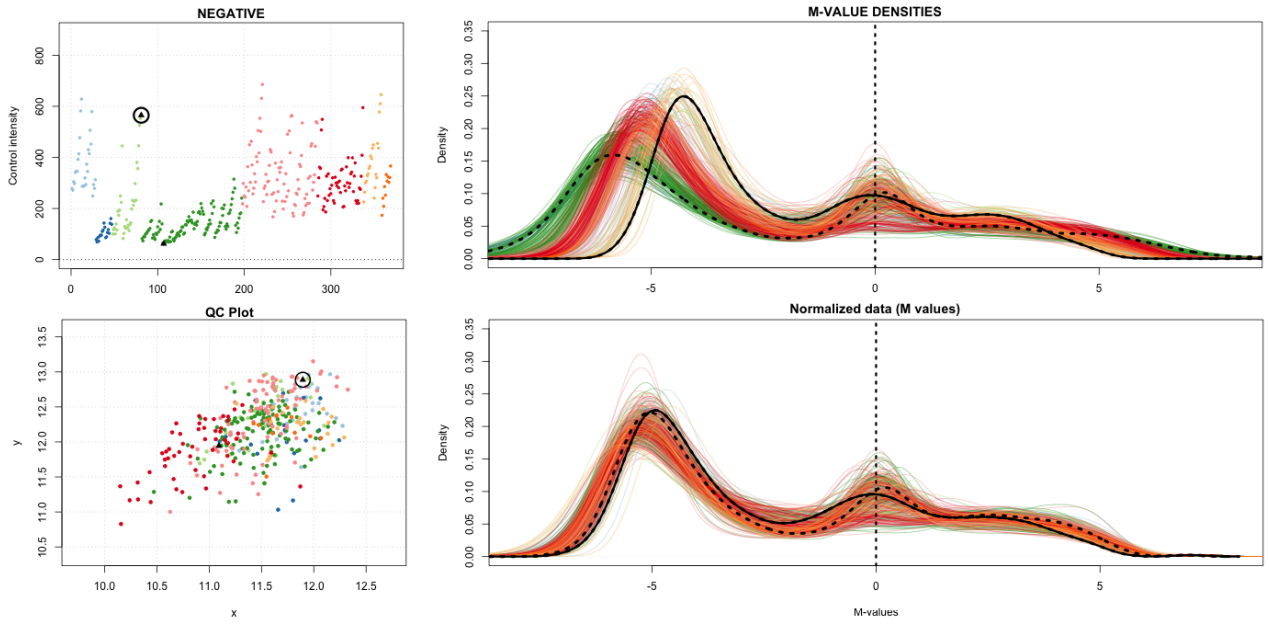


Figure 6: **Comparison of raw and normalized data** As discussed in the Advanced option, normalized data can be added as well to the visualization interface. In the present plot, the top plot at the right shows the raw densities while the bottom plot shows the densities after functional normalization [7].

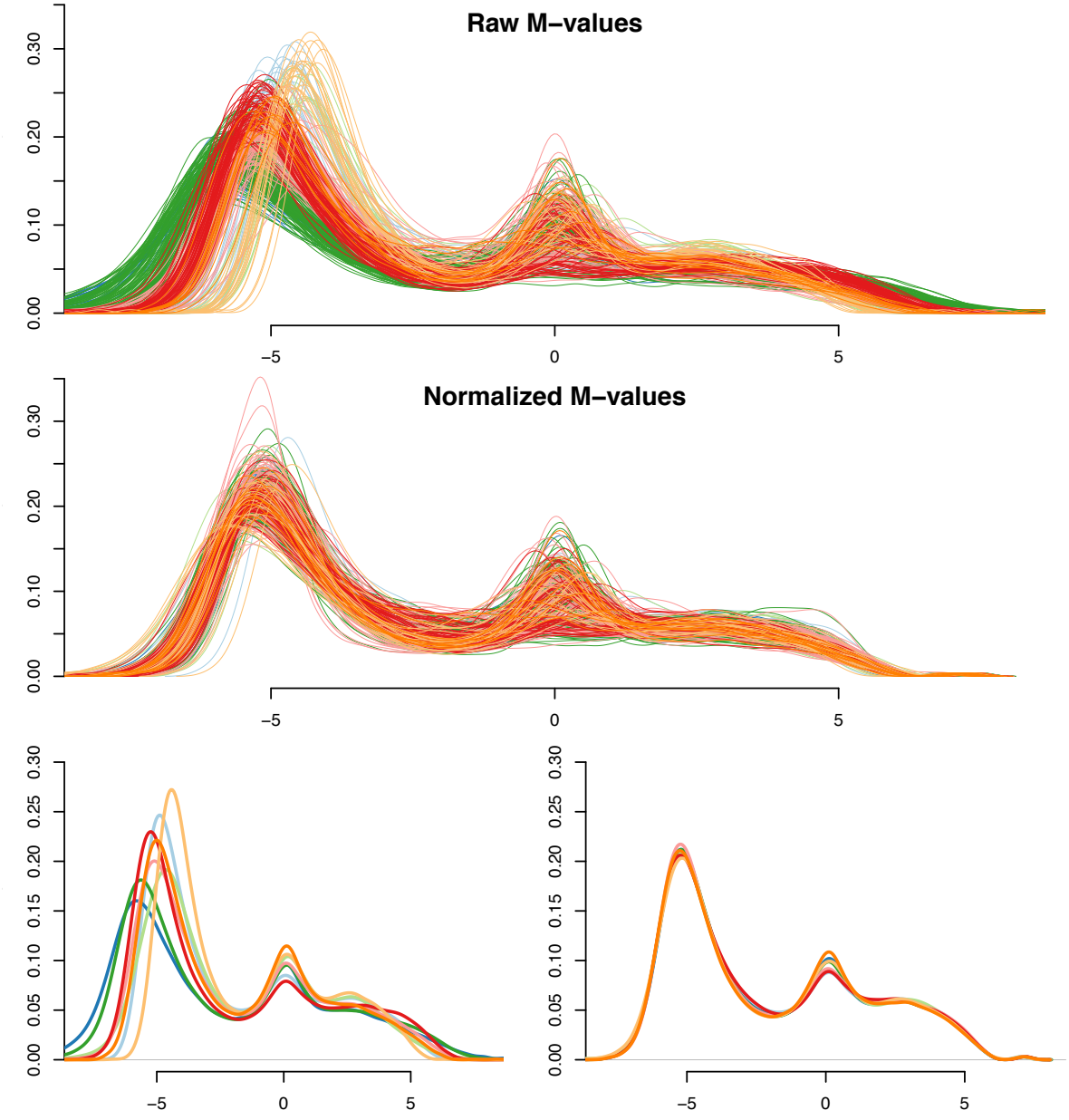


Figure 7: **Visualization of batch effects in the TCGA HNSCC dataset** In the first two plots are shown the densities of the M-values for Type I green probes before and after functional normalization [7] as presented in the *shinyMethyl* interactive interface. Each curve represents one sample and different colors represent different batches. The last two plots show the average density for each plate before and after normalization. One can observe that functional normalization removed significantly global batch effects.

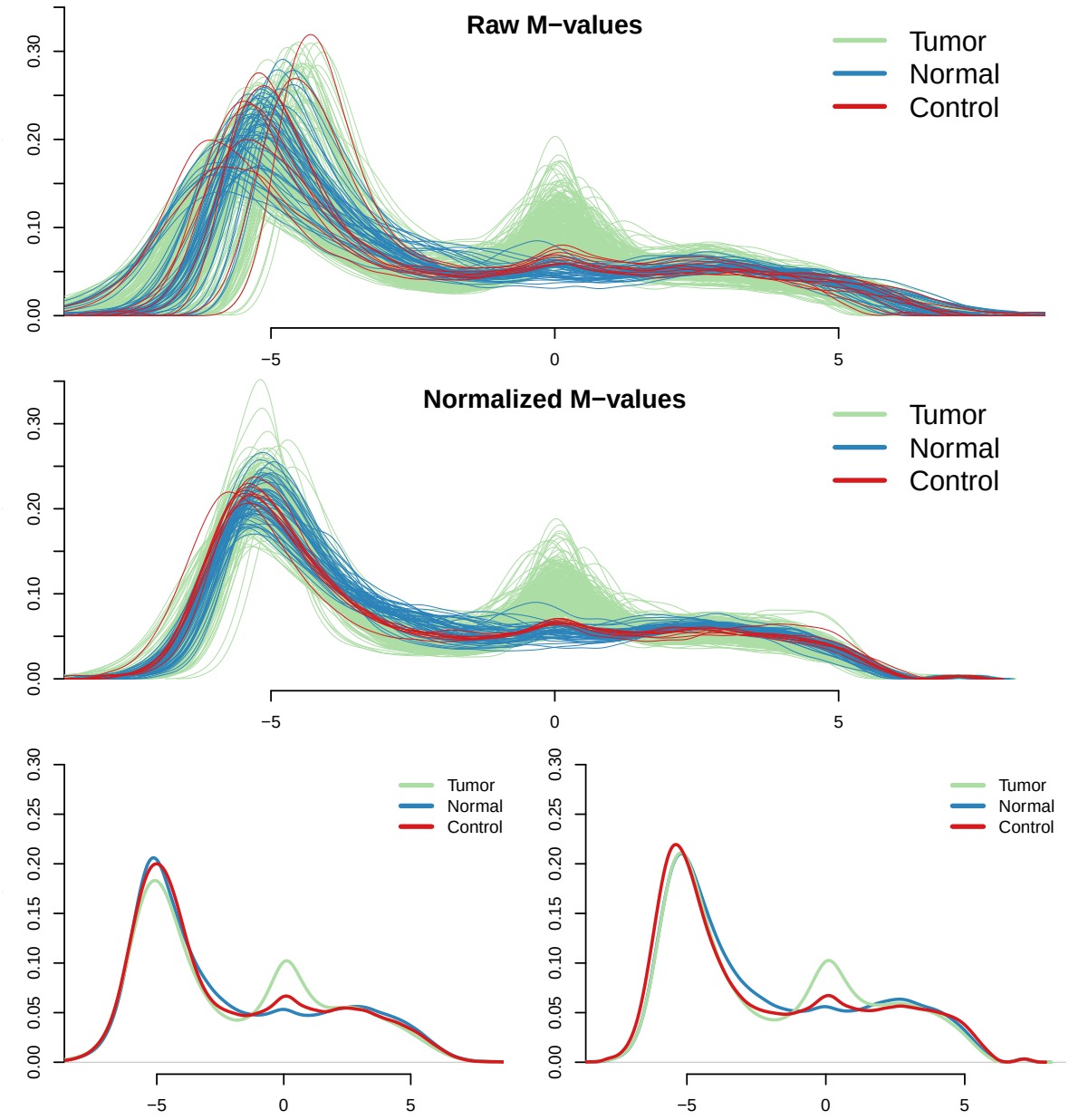


Figure 8: **Visualization of cancer/normal differences in the TCGA HNSCC dataset**In the first two plots are shown the densities of the M-values for Type I green probes before (a) and after (b) functional normalization [7] as presented in the *shinyMethyl* interactive interface. Green and blue densities represent tumor and normal samples respectively, and red densities represent 9 technical replicates of a control cell line. The last two plots show the average density for each sample group before and after normalization. Functional normalization preserves the expected marginal differences between normal and cancer, while reducing the variation between the technical controls (red lines).