The quantro user's guide

Stephanie C. Hicks shicks@jimmy.harvard.edu
Rafael A. Irizarry rafa@jimmy.harvard.edu

Modified: April 7, 2017. Compiled: April 24, 2017

Contents

1 Introduction

Multi-sample normalization techniques such as quantile normalization [1, 2] have become a standard and essential part of analysis pipelines for high-throughput data. Although it was originally developed for gene expression microarrays, it is now used across many different high-throughput applications including genotyping arrays, DNA Methylation, RNA Sequencing (RNA-Seq) and Chromatin Immunoprecipitation Sequencing (ChIP-Seq). These techniques transform the original raw data to remove unwanted technical variation. However, quantile normalization and other global normalization methods rely on assumptions about the data generation process that are not appropriate in some context. Until now, it has been left to the researcher to check for the appropriateness of these assumptions.

Quantile normalization assumes that the statistical distribution of each sample is the same. Normalization is achieved by forcing the observed distributions to be the same and the average distribution, obtained by taking the average of each quantile across samples, is used as the reference. This method has worked very well in practice but note that when the assumptions are not met, global changes in distribution, that may be of biological interest, will be wiped out and features that are not different across samples can be artificially induced. These types of assumptions are justified in many

biomedical applications, for example in gene expression studies in which only a minority of genes are expected to be differentially expressed. However, if, for example, a substantially higher percentage of genes are expected to be expressed in only one group of samples, it may not be appropriate to use global adjustment methods.

The quantro R-package can be used to test for global differences between groups of distributions which asses whether global normalization methods such as quantile normalization should be applied. Our method uses the raw unprocessed high-throughput data to test for global differences in the distributions across a set of groups. The main function `quantro()` will perform two tests:

1. An ANOVA to test if the medians of the distributions are different across groups. Differences across groups could be attributed to unwanted technical variation (such as batch effects) or real global biological variation. This is a helpful step for the user to verify if there is any technical variation unaccounted for.
2. A test for global differences between the distributions across groups which returns a test statistic called `quantroStat`. This test statistic is a ratio of two variances and is similar to the idea of ANOVA. The main idea is to compare the variability of distributions within groups relative to between groups. If the variability between groups is sufficiently larger than the variability within groups, then this suggests global adjustment methods may not be appropriate. As a default, we perform this test on the median normalized data, but the user may change this option.

2 Getting Started

Load the package in R

```
library(quantro)
```

3 Data

3.1 flowSorted Data Example

To explore how to use `quantro()`, we use the `FlowSorted.DLPFC.450k` data package in Bioconductor [3]. This data set contains raw data objects of 58 Illumina 450K DNA methylation microarrays, formatted as `RGset` objects. The samples represent two different cellular populations of brain tissues on the same 29 individuals extracted using flow sorting. For more information on this data set, please see the FlowSorted.DLPFC.450k User's Guide. For the purposes of this vignette, a MethylSet object from the `minfi` Bioconductor package [4] was created which is a subset of the rows from the original `FlowSorted.DLPFC.450k` data set. This MethylSet object is found in the /data folder and the script to create the object is found in /inst.

Here we will explore the distributions of these two cellular populations of brain tissue (`NeuN_pos` and `NeuN_neg`) and then test if there are global differences in the distributions across groups. First, load the MethylSet object (`flowSorted`) and compute the Beta values using the function `getBeta()` in the `minfi` Bioconductor package. We use an offset of 100 as this is the default used by Illumina.

```
library(quantro)
library(minfi)
data(flowSorted)
p <- getBeta(flowSorted, offset = 100)
pd <- pData(flowSorted)
dim(p)
```

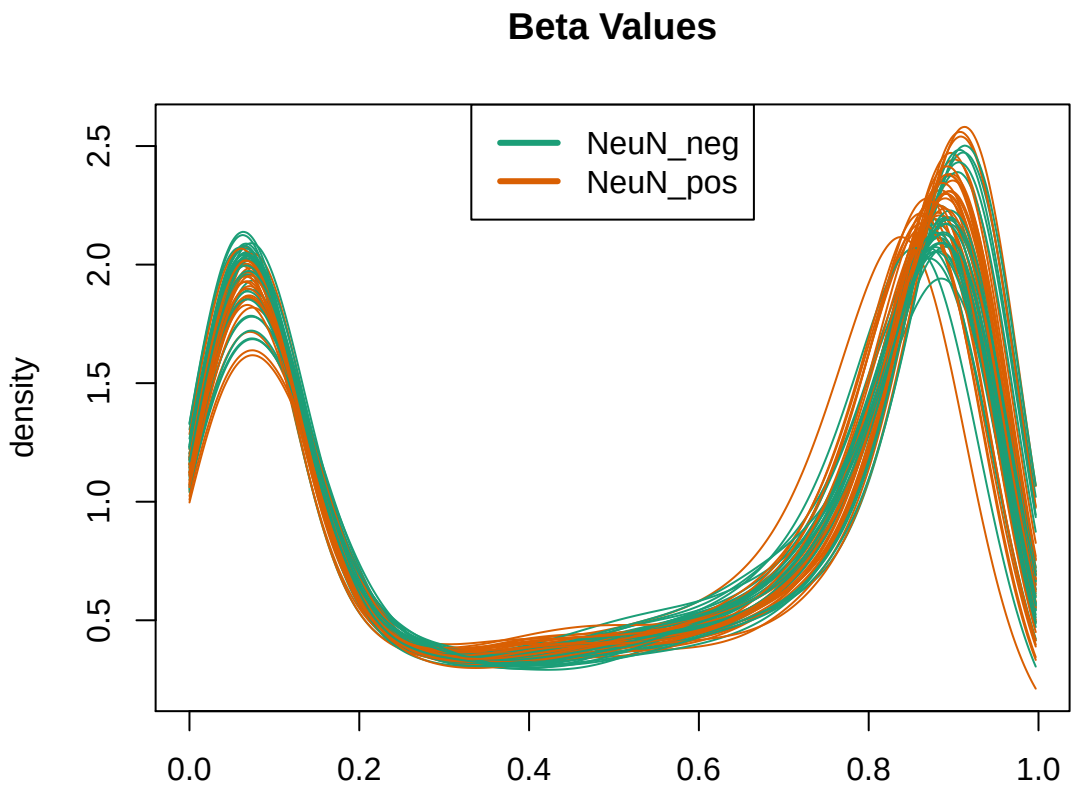
```
## [1] 10000    58

head(pd)

## DataFrame with 6 rows and 11 columns
##        Sample_Name    SampleID    CellType Sentrix.Barcode Sample.Section        diag
##        <character> <character> <character>       <numeric>    <character> <character>
## 813_N        813_N         813    NeuN_pos      7766130090         R02C01     Control
## 1740_N      1740_N        1740    NeuN_pos      7766130090         R02C02     Control
## 1740_G      1740_G        1740    NeuN_neg      7766130090         R04C01     Control
## 1228_G      1228_G        1228    NeuN_neg      7766130090         R04C02     Control
## 813_G        813_G         813    NeuN_neg      7766130090         R06C01     Control
## 1228_N      1228_N        1228    NeuN_pos      7766130090         R06C02     Control
##                sex   ethnicity       age       PMI
##        <character> <character> <integer> <integer>
## 813_N       Female   Caucasian        30        14
## 1740_N      Female     African        13        17
## 1740_G      Female     African        13        17
## 1228_G        Male   Caucasian        47        13
## 813_G       Female   Caucasian        30        14
## 1228_N        Male   Caucasian        47        13
##                                                                  BasePath
##                                                               <character>
## 813_N  /dcs01/lieber/ajaffe/450k/Zack_450k/IDAT/7766130090/7766130090_R02C01
## 1740_N /dcs01/lieber/ajaffe/450k/Zack_450k/IDAT/7766130090/7766130090_R02C02
## 1740_G /dcs01/lieber/ajaffe/450k/Zack_450k/IDAT/7766130090/7766130090_R04C01
## 1228_G /dcs01/lieber/ajaffe/450k/Zack_450k/IDAT/7766130090/7766130090_R04C02
## 813_G  /dcs01/lieber/ajaffe/450k/Zack_450k/IDAT/7766130090/7766130090_R06C01
## 1228_N /dcs01/lieber/ajaffe/450k/Zack_450k/IDAT/7766130090/7766130090_R06C02
```

3.2 Plot distributions

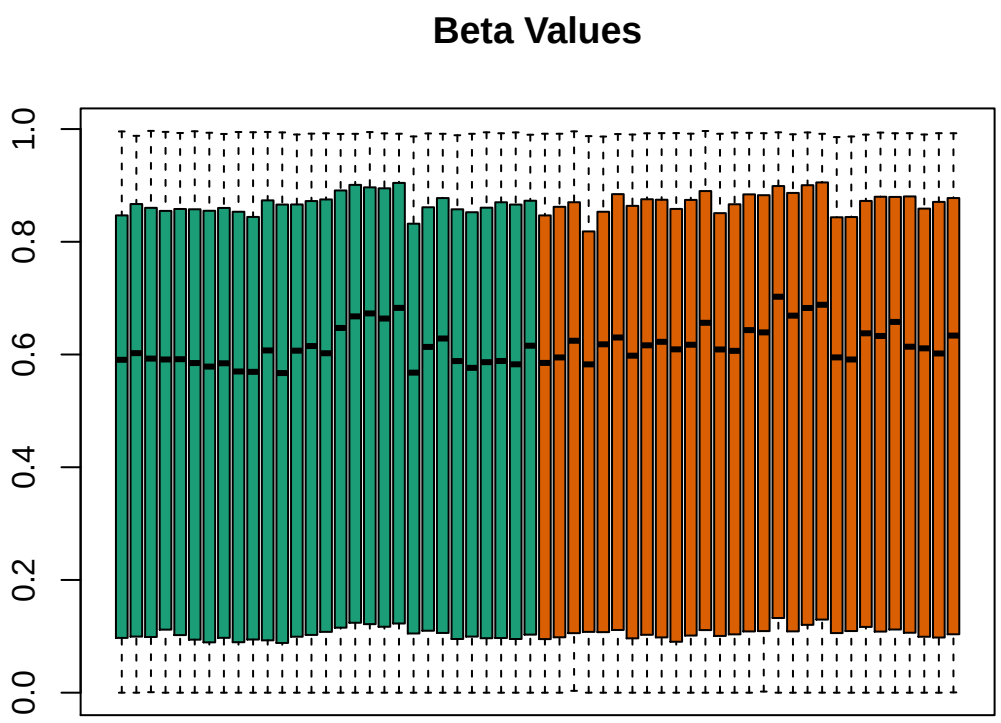
quantro contains two functions to view the distributions of the samples of interest: `matdensity()` and `matboxplot()`. `matdensity()` computes the density for each sample (columns) and uses the `matplot()` function to plot all the densities. `matboxplot()` orders and colors the samples by a group level variable. These two functions use the `RColorBrewer` package and the brewer palettes can be changed using the arguments `brewer.n` and `brewer.name`.

The distributions of the two groups of cellular populations are shown here. The `NeuN_neg` samples are colored in green and the `NeuN_pos` are colored in red.

```
matdensity(p, groupFactor = pd$CellType, xlab = " ", ylab = "density",
           main = "Beta Values", brewer.n = 8, brewer.name = "Dark2")
legend('top', c("NeuN_neg", "NeuN_pos"), col = c(1, 2), lty = 1, lwd = 3)
```




```
matboxplot(p, groupFactor = pd$CellType, xaxt = "n", main = "Beta Values")
```



4 Using the quantro() function

4.1 Input for quantro()

The quantro() function must have two objects as input:

- an object which is a data frame or matrix with observations (e.g. probes or genes) on the rows and samples as the columns.
- a groupFactor which represents the group level information about each sample. For example if the samples represent tumor and normal samples, provide quantro() with a factor representing which columns in the object are normal and tumor samples.

4.2 Running quantro()

In this example, the groups we are interested in comparing are contained in the CellType column in the pd dataset. To run the quantro() function, input the data object and the object containing the phenotypic data. Here we use the flowSorted data set as an example.

```
qtest <- quantro(object = p, groupFactor = pd$CellType)

## [quantro] Average medians of the distributions are
##                         not equal across groups.

## [quantro] Calculating the quantro test statistic.

## [quantro] No permutation testing performed.
##                           Use B > 0 for permutation testing.

qtest

## quantro: Test for global differences in distributions
##    nGroups:  2
##    nTotSamples:  58
##    nSamplesinGroups:  29 29
##    anovaPval:  0.01206
##    quantroStat:  8.80735
##    quantroPvalPerm:  Use B > 0 for permutation testing.
```

The details related to the experiment can be extracted using the summary accessor function:

```
summary(qtest)

## $nGroups
## [1] 2
##
## $nTotSamples
## [1] 58
##
## $nSamplesinGroups
## NeuN_neg NeuN_pos
##       29       29
```

To asssess if the medians of the distributions different across groups, we perform an ANOVA on the medians from the samples. Those results can be found using anova:

```
anova(qtest)

## Analysis of Variance Table
##
## Response: objectMedians
##             Df   Sum Sq   Mean Sq F value  Pr(>F)
## groupFactor  1 0.006919 0.0069194  6.7327 0.01206 *
## Residuals   56 0.057553 0.0010277
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The full output can be seen The test statistic produced from quantro() testing for global differences between distributions is given by quantroStat:

```
quantroStat(qtest)

## [1] 8.807348
```

4.3 eSets

quantro() also can accept objects that inherit eSets such as an ExpressionSet or MethylSet. The groupFactor must still be provided.

```
is(flowSorted, "MethylSet")

## [1] TRUE

qtest <- quantro(flowSorted, groupFactor = pData(flowSorted)$CellType)

## [quantro] Average medians of the distributions are
##                          not equal across groups.

## [quantro] Calculating the quantro test statistic.

## [quantro] No permutation testing performed.
##                            Use B > 0 for permutation testing.

qtest

## quantro: Test for global differences in distributions
##    nGroups:  2
##    nTotSamples:  58
##    nSamplesinGroups:  29 29
##    anovaPval:  0.01206
##    quantroStat:  8.80735
##    quantroPvalPerm:  Use B > 0 for permutation testing.
```

4.4 Output from quantro()

Elements in the S4 object from quantro() include:

Element	Description
summary	Returns a list of three elements related to a summary of the experiment: nGroups: number of groups nTotSamples: total number of samples nSamplesinGroups: number of samples in each group
anova	Results from an ANOVA to test if the average medians of the distributions are different across groups
MSbetween	Mean squared error between groups
MSwithin	Mean squared error within groups
quantroStat	A test statistic which is a ratio of the mean squared error between groups of distributions (MSbetween) to the mean squared error within groups of distributions (MSwithin)
quantroStatPerm	If B is not equal to 0, then a permutation test was performed to assess the statistical significance of quantroStat. These are the test statistics resulting from the permuted samples
quantroPvalPerm	If B is not equal to 0, then this is the p-value associated with the proportion of times the test statistics resulting from the permuted samples were larger than quantroStat

5 Assessing the statistical significance

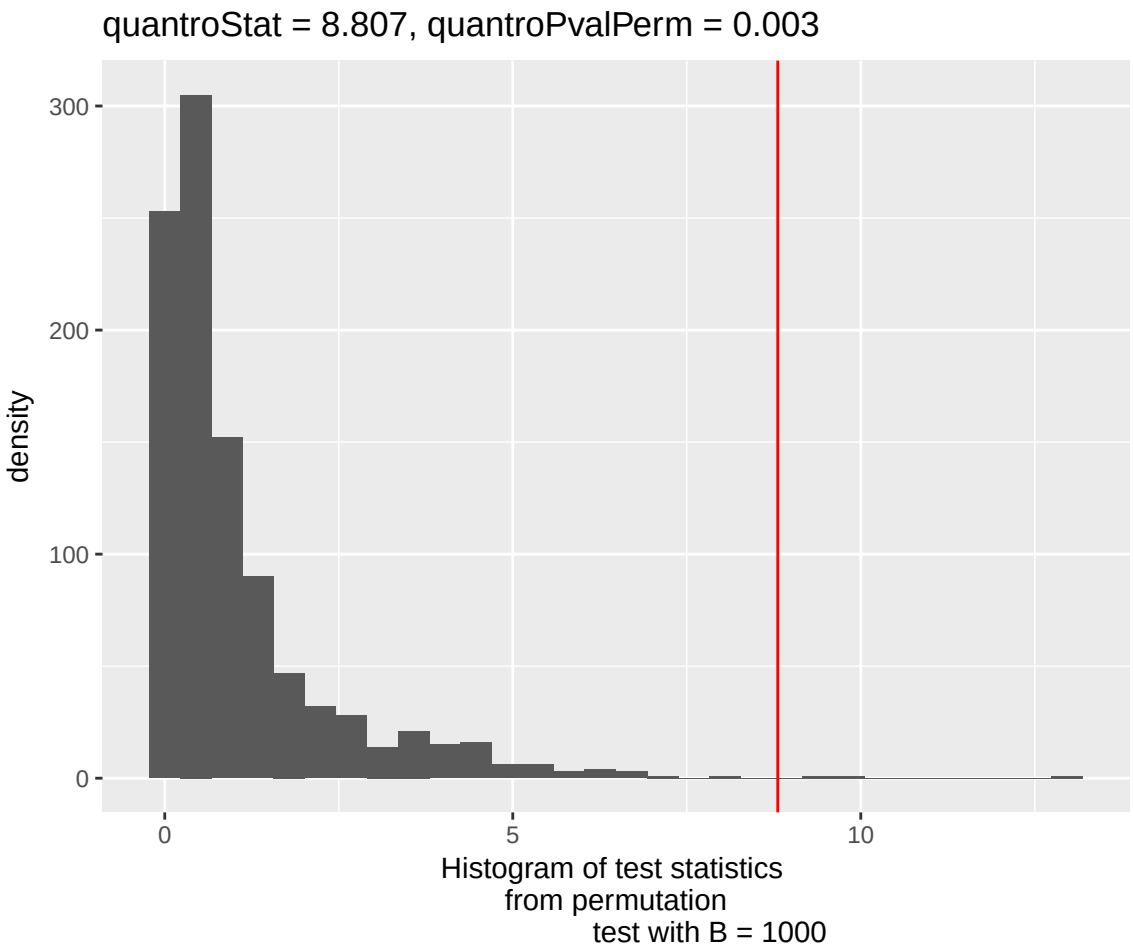
To assess statistical significance of the test statistic, we use permutation testing. We use the `foreach` package which distribute the computations across multiple cross in a single machine or across multiple machines in a cluster. The user must pick how many permutations to perform where `B` is the number of permutations. At each permutation of the samples, a test statistic is calculated. The proportion of test statistics (`quantroStatPerm`) that are larger than the `quantroStat` is reported as the `quantroPvalPerm`. To use the `foreach` package, we first register a backend, in this case a machine with 1 cores.

```
library(doParallel)
registerDoParallel(cores=1)
qtestPerm <- quantro(p, groupFactor = pd$CellType, B = 1000)

## [quantro] Average medians of the distributions are
##                         not equal across groups.

## [quantro] Calculating the quantro test statistic.

## [quantro] Starting permutation testing.

## [quantro] Using a single core (backend:  doParallelMC,
##                         version:  1.0.10).

qtestPerm

## quantro: Test for global differences in distributions
##    nGroups:  2
##    nTotSamples:  58
##    nSamplesinGroups:  29 29
##    anovaPval:  0.01206
##    quantroStat:  8.80735
##    quantroPvalPerm:  0.003
```

6 Visualizing the statistical significance from permutation tests

If permutation testing was used (i.e. specifying $B > 0$), then there is a second function in the package called `quantroPlot()` which will plot the test statistics of the permuted samples. The plot is a histogram of the null test statistics `quantroStatPerm` from `quantro()` and the red line is the observed test statistic `quantroStat` from `quantro()`.

```
quantroPlot(qtestPerm)
```



Additional options in the `quantroPlot()` function include:

Element	Description
xLab	the x-axis label
yLab	the y-axis label
mainLab	title of the histogram
binWidth	change the binwidth

7 SessionInfo

```
sessionInfo()

## R version 3.4.0 (2017-04-21)
## Platform: x86_64-pc-linux-gnu (64-bit)
## 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
##
## locale:
```

```
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
##  [4] LC_COLLATE=C               LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
## [10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets  methods
## [9] base
##
## other attached packages:
##  [1] doParallel_1.0.10          minfi_1.22.0               bumphunter_1.16.0
##  [4] locfit_1.5-9.1             iterators_1.0.8            foreach_1.4.3
##  [7] Biostrings_2.44.0          XVector_0.16.0             SummarizedExperiment_1.6.0
## [10] DelayedArray_0.2.0         matrixStats_0.52.2         Biobase_2.36.0
## [13] GenomicRanges_1.28.0       GenomeInfoDb_1.12.0        IRanges_2.10.0
## [16] S4Vectors_0.14.0           BiocGenerics_0.22.0        quantro_1.10.0
##
## loaded via a namespace (and not attached):
##  [1] httr_1.2.1               nor1mix_1.2-2            splines_3.4.0
##  [4] highr_0.6                doRNG_1.6.6              GenomeInfoDbData_0.99.0
##  [7] Rsamtools_1.28.0         yaml_2.1.14              RSQLite_1.1-2
## [10] backports_1.0.5          lattice_0.20-35          quadprog_1.5-5
## [13] limma_3.32.0             digest_0.6.12            RColorBrewer_1.1-2
## [16] colorspace_1.3-2         htmltools_0.3.5          preprocessCore_1.38.0
## [19] Matrix_1.2-9             plyr_1.8.4               GEOquery_2.42.0
## [22] siggenes_1.50.0          XML_3.98-1.6             biomaRt_2.32.0
## [25] genefilter_1.58.0        zlibbioc_1.22.0          xtable_1.8-2
## [28] scales_0.4.1             BiocParallel_1.10.0      tibble_1.3.0
## [31] openssl_0.9.6            annotate_1.54.0          beanplot_1.2
## [34] pkgmaker_0.22            ggplot2_2.2.1            GenomicFeatures_1.28.0
## [37] lazyeval_0.2.0           survival_2.41-3          magrittr_1.5
## [40] mclust_5.2.3             memoise_1.1.0            evaluate_0.10
## [43] nlme_3.1-131             MASS_7.3-47              data.table_1.10.4
## [46] tools_3.4.0              registry_0.3             BiocStyle_2.4.0
## [49] stringr_1.2.0            munsell_0.4.3            rngtools_1.2.4
## [52] AnnotationDbi_1.38.0     base64_2.0               compiler_3.4.0
## [55] grid_3.4.0               RCurl_1.95-4.8           labeling_0.3
## [58] bitops_1.0-6             rmarkdown_1.4            gtable_0.2.0
## [61] codetools_0.2-15         multtest_2.32.0          DBI_0.6-1
## [64] reshape_0.8.6            R6_2.2.0                 illuminaio_0.18.0
## [67] GenomicAlignments_1.12.0 knitr_1.15.1             rtracklayer_1.36.0
## [70] rprojroot_1.2            stringi_1.1.5            Rcpp_0.12.10
```

References

[1] B M Bolstad, R A Irizarry, M Astrand, and T P Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–93, Jan 2003.

[2] Rafael A Irizarry, Bridget Hobbs, Francois Collin, Yasmin D Beazer-Barclay, Kristen J Antonellis, Uwe Scherf, and Terence P Speed. Exploration, normalization, and summaries of high density oligonucleotide array probe level data.

Biostatistics, 4(2):249–64, Apr 2003. doi:10.1093/biostatistics/4.2.249.

[3] A J Jaffe and Z A Kaminsky. *FlowSorted.DLPFC.450k: Illumina HumanMethylation data on sorted frontal cortex cell populations*, r package version 1.0.0 edition.

[4] 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):1363–9, May 2014. doi:10.1093/bioinformatics/btu049.