Detecting allele-specific events from ChIP-seq data

Ines de Santiago, Wei Liu, Ke Yuan, Bruce Ponder, Kerstin Meyer, Florian Markowetz

Edited: 2016; Compiled: June 13, 2016

Contents

1 Introduction

Allele-specific binding (ASB) measurements of transcription-factor binding from ChIP-seq data have provided important insights into the allelic effects of non-coding variants and its contribution to phenotypic diversity. However, such approaches are designed to examine the allelic imbalances in diploid samples and do not address copy number differences between the two alleles, a known phenotypical feature of cancer cells.

BaalChIP (Bayesian Analysis of Allelic imbalances from ChIP-seq data) tests the differential read counts of the alleles at each heterozygous variant using a Bayesian framework to account for the background allele composition and other sources of bias that might influence the overall ChIP-seq read count, and takes advantage of the fact that multiple transcription factor ChIP-seq data may be available for the same variant to improve ASB detection (Figure 1).

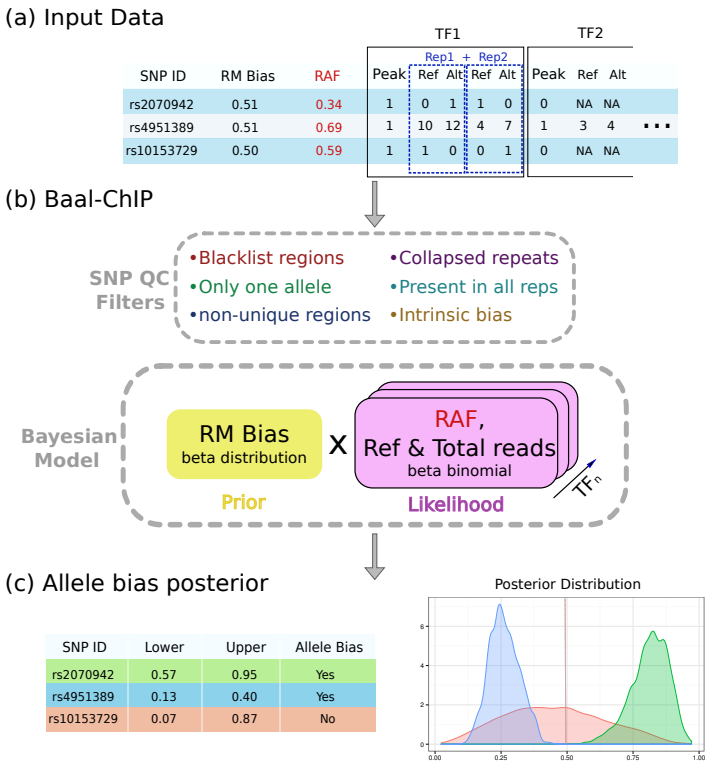


Figure 1: Description of Baal-ChIP model frame work. (a) Input data: the reference mapping (RM) bias and reference allele frequency (RAF) have been included in the input data. The first column of peak file is the binary data used to state the calling peaks and the other columns are ChIP-seq reads count. (b) Baal-ChIP package has two modules: SNP QC filters to remove the false identified SNPs causing by technical biases and beta-binomial Bayesian model to consider RM and RAF bias. (c) Model output: the output from Baal-ChIP is posterior distribution for each SNP and user can use defined threshold to identify the SNPs with allele bias (default value for threshold is 0.5)

2 A sample session

This section offers a quick example of how to use *BaalChIP* to identify ASB events with correction for relative allele frequency.

2.1 Example datasets in this vignette

The example dataset in this vignette contains ChIP-seq data obtained for two cell lines: A cancer cell-line (MCF7) and a normal cell line (GM12891). For each cell line, ChIP-seq data exists for four transcription factors and two biological replicates for each of the transcription factors. This example dataset also contains the B-allele frequency (BAF) scores obtained using Genome-Wide Human SNP Array 6.0 Affymetrix microarrays. In this example dataset, the BAF scores are used to correct the allelic read counts.

Note that the example data in this vignette does not reveal real biology and was build only for demonstration purposes.

2.2 Reading in data

The first thing to do is to read some data.

The metadata and all files necessary for this example are available in the extra subdirectory of the BaalChIP package directory; you can make this your working directory by entering:

```
library(BaalChIP)
setwd(system.file("test",package="BaalChIP"))
```

The first step is to construct a *BaalChIP* object:

```
samplesheet <- "exampleChIP.tsv"
hets <- c("MCF7"="MCF7_hetSNP.txt", "GM12891"="GM12891_hetSNP.txt")
res <- new("BaalChIP", samplesheet=samplesheet, hets=hets)
```

2.3 BaalChIP analysis

The BaalChIP analysis can either be run with various commands (e.g. `alleleCounts`, `QCfilter`, `mergePerGroup`, `filter1allele`) one at the time, or use the function `BaalChIP.run` which will run a typical analysis.

Given a new BaalChIP object, to run a BaalChIP analysis and identify allele-specific binding events, type:

```
res <- new("BaalChIP", samplesheet=samplesheet, hets=hets)
res <- BaalChIP.run(res)
```

If you wish to have more control over the input options, the same analysis above can be performed with various commands as follows:

```
#load data
data(blacklist_hg19)
data(pickrell2011cov1_hg19)
data(UniqueMappability50bp_hg19)

#run one at the time
res <- alleleCounts(res, min_base_quality=10, min_mapq=15)
res <- QCfilter(res, RegionsToFilter=c("blacklist_hg19", "pickrell2011cov1_hg19"),
                RegionsToKeep="UniqueMappability50bp_hg19")
res <- mergePerGroup(res)
res <- filter1allele(res)
res <- getASB(res, Iter=5000, conf_level=0.95, RMcorrection = TRUE, RAFcorrection=TRUE)
```

The following sections describe these steps in more detail.

3 Notes on data entry

3.1 The samplesheet

In order to run BaalChIP, one needs to generate a sample sheet describing the samples and the groups within each study. This file should be saved as a tab-delimited file. The extension of this file is not important, for example it can be `.txt` as long as it is a tab-delimited file. A .tsv sample sheet has been included in this vignette and can be assessed as follows:

```
setwd(system.file("test",package="BaalChIP"))
samplesheet <- read.delim("exampleChIP.tsv")
samplesheet
```

```
##    group_name target replicate_number                       bam_name
## 1        MCF7   cFOS                1    bamFiles/MCF7_cFOS_Rep1.bam
## 2        MCF7   cFOS                2    bamFiles/MCF7_cFOS_Rep2.bam
## 3        MCF7   cMYC                1    bamFiles/MCF7_cMYC_Rep1.bam
## 4        MCF7   cMYC                2    bamFiles/MCF7_cMYC_Rep2.bam
## 5        MCF7   POL2                1    bamFiles/MCF7_POL2_Rep1.bam
## 6        MCF7   POL2                2    bamFiles/MCF7_POL2_Rep2.bam
## 7        MCF7  STAT3                1   bamFiles/MCF7_STAT3_Rep1.bam
## 8        MCF7  STAT3                2   bamFiles/MCF7_STAT3_Rep2.bam
## 9     GM12891   POL2                1 bamFiles/GM12891_POL2_Rep1.bam
## 10    GM12891   POL2                2 bamFiles/GM12891_POL2_Rep2.bam
## 11    GM12891   PAX5                1 bamFiles/GM12891_PAX5_Rep1.bam
## 12    GM12891   PAX5                2 bamFiles/GM12891_PAX5_Rep2.bam
## 13    GM12891    PU1                1  bamFiles/GM12891_PU1_Rep1.bam
## 14    GM12891    PU1                2  bamFiles/GM12891_PU1_Rep2.bam
## 15    GM12891   TAF1                1 bamFiles/GM12891_TAF1_Rep1.bam
## 16    GM12891   TAF1                2 bamFiles/GM12891_TAF1_Rep2.bam
##                     bed_name
## 1     bedFiles/MCF7_cFOS.bed
## 2     bedFiles/MCF7_cFOS.bed
## 3     bedFiles/MCF7_cMYC.bed
## 4    bedFiles//MCF7_cMYC.bed
## 5     bedFiles/MCF7_POL2.bed
## 6     bedFiles/MCF7_POL2.bed
## 7    bedFiles/MCF7_STAT3.bed
## 8    bedFiles/MCF7_STAT3.bed
## 9  bedFiles/GM12891_POL2.bed
## 10 bedFiles/GM12891_POL2.bed
## 11 bedFiles/GM12891_PAX5.bed
## 12 bedFiles/GM12891_PAX5.bed
## 13  bedFiles/GM12891_PU1.bed
## 14  bedFiles/GM12891_PU1.bed
## 15 bedFiles/GM12891_TAF1.bed
## 16 bedFiles/GM12891_TAF1.bed
```

This sample sheet details the metadata for ChIP-seq studies in MCF7 and GM12891 cell lines. For each study, ChIP-seq data exists for four transcription factors (`target`). The first column `group name` identifies the group label of each study (MCF7, GM12891). The column `replicate number` shows that there are two biological replicates for each ChIP-seq factor. The sample sheet also contains file paths to the BAM files (`bam name`) with the aligned reads and the BED files (`bed name`) with the genomic regions of signal enrichment that the user is interested in (typically these are the ChIP-seq peaks files).

3.2 The hets files

BaalChIP requires a variant file containing the list of heterozygous variants to be analysed. As an example, a small set of heterozygous variants for each cell line has been included in this vignette and can be assessed as follows:

```
setwd(system.file("test",package="BaalChIP"))
head(read.delim("MCF7_hetSNP.txt"))
```

```
##           ID CHROM       POS REF ALT       RAF
## 1 rs10169169  chr2 191412889   T   G 0.4870296
## 2  rs1021813  chr3  59413060   T   C 0.4689580
## 3  rs1025641 chr10 128307192   T   C 0.4077530
```

```
## 4 rs10444404 chr12  15114751   T   G 0.5195654
## 5  rs1048347 chr10 124096061   A   C 0.4852518
## 6 rs10495062  chr1 217804955   T   C 0.3654244
```

The information in the variant file should include an `ID` column with a unique identifier string per variant, the (1-based) genomic coordinates `CHROM, POS`, and the A,C,G,T bases for the reference `REF` and the non-reference alternate `ALT` allele.

The final column `RAF` consists of a value ranging from 0 to 1 for each variant denoting the relative allele frequency. A value between 0.5 and 1 denotes a bias to the reference allele, and a value between 0 and 0.5 a bias to the alternate allele. This column is optional, and will not be necessary if we ask BaalChIP to calculate the RAF values from the input gDNA libraries. If both gDNA and RAF values are missing BaalChIP will still run but will not correct for relative allele frequency (copy-number) bias.

4 Constructing a BaalChIP object

The first step is to generate a BaalChIP object. The function `new` accepts a samplesheet and a named vector containing the filenames for the variant files to be used. The names in the vector should correspond to `group_name` strings in the `.csv` samplesheet, in this case it should be MCF7 and GM12891.

```
samplesheet <- "example.tsv"
hets <- c("MCF7"="MCF7_hetSNP.txt", "GM12891"="GM12891_hetSNP.txt")
res <- new("BaalChIP", samplesheet=samplesheet, hets=hets)
```

the samplesheet is saved in the `samples` slot of a BaalChIP object:

```
res@samples
```

4.1 Obtaining allele-specific counts for BAM files

The next step is to compute, for each variant the number of reads carrying the reference (REF) and alternative (ALT) alleles. The `alleleCounts` function will read and scan all BAM files within the `samples` slot of a BaalChIP object and compute the read coverage at each allele. Allele counts are computed using the `pileup` function and the `PileupParam` constructor of the *Rsamtools* package (Morgan et al., 2016). For each BAM file, it will only consider heterozygous SNPs overlapping the genomic regions in the corresponding BED files. Two arguments can be manipulated by the user: the `min_mapq` refers to the minimum MAPQ value for an alignment to be included in pileup (default is 15); and the `min_base_quality` which refers to the minimum QUAL value for each nucleotide in an alignment (default is 10).

```
res <- alleleCounts(res, min_base_quality=10, min_mapq=15)
res
```

4.2 QCfilter: A filter to exclude SNPs in regions of known problematic read alignment

After computing the read counts per allele, the next step in the BaalChIP pipeline is an extensive quality control step to consider technical biases that may contribute to the false identification of regulatory SNPs.

The function `QCfilter` is used to excluded sites susceptible to allelic mapping bias in regions of known problematic read alignment (Pickrell et al., 2011; Consortium, 2012). This function accepts two arguments: RegionsToFilter with the genomic regions to be excluded, and RegionsToKeep with the genomic regions to be kept.

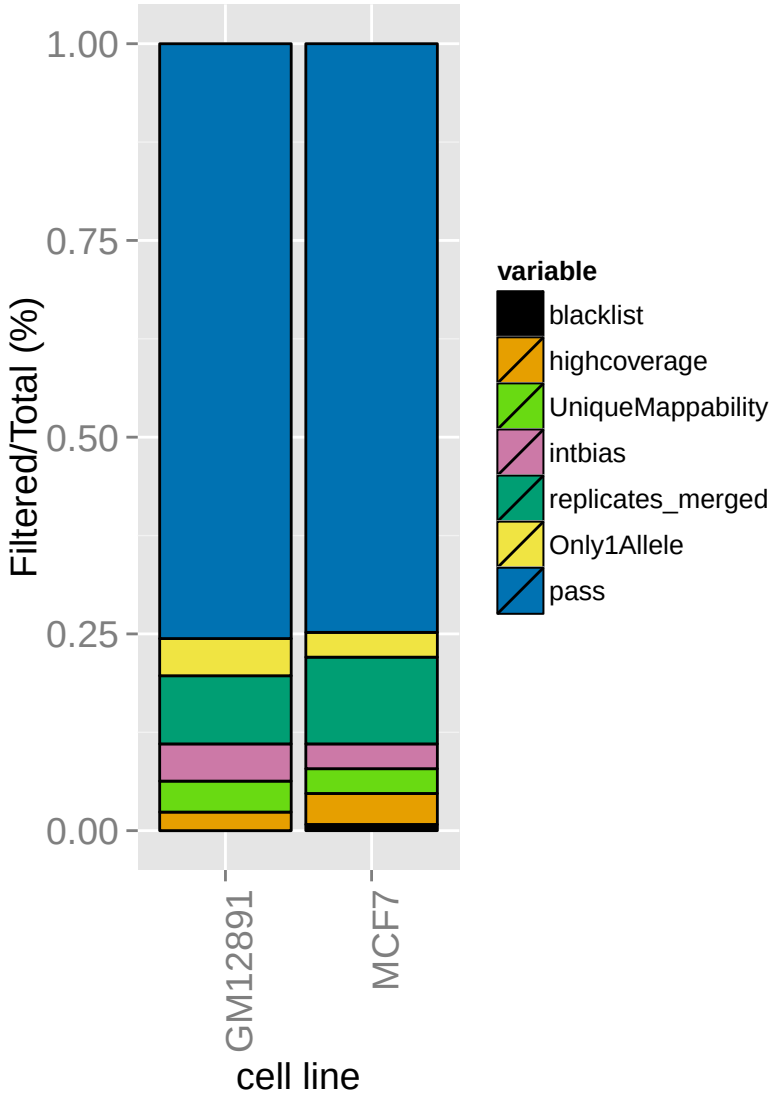
Three datafiles are included with BaalChIP package for the human genome:

(1) blacklisted regions downloaded from the UCSC Genome Browser (mappability track; release 3, October 2011; thewgEncodeDacMapabilityConsensusExcludable and wgEncodeDukeMapabilityRegionsExcludable tables), (2) non-unique regions selected from DUKE uniqueness mappability track of the UCSC genome browser (release 3, October 2011; wgEncodeCrgMapabilityAlign50mer table), and (3) collapsed repeat regions downloaded from Pickrell et al., 2010 at the 0.1% threshold.

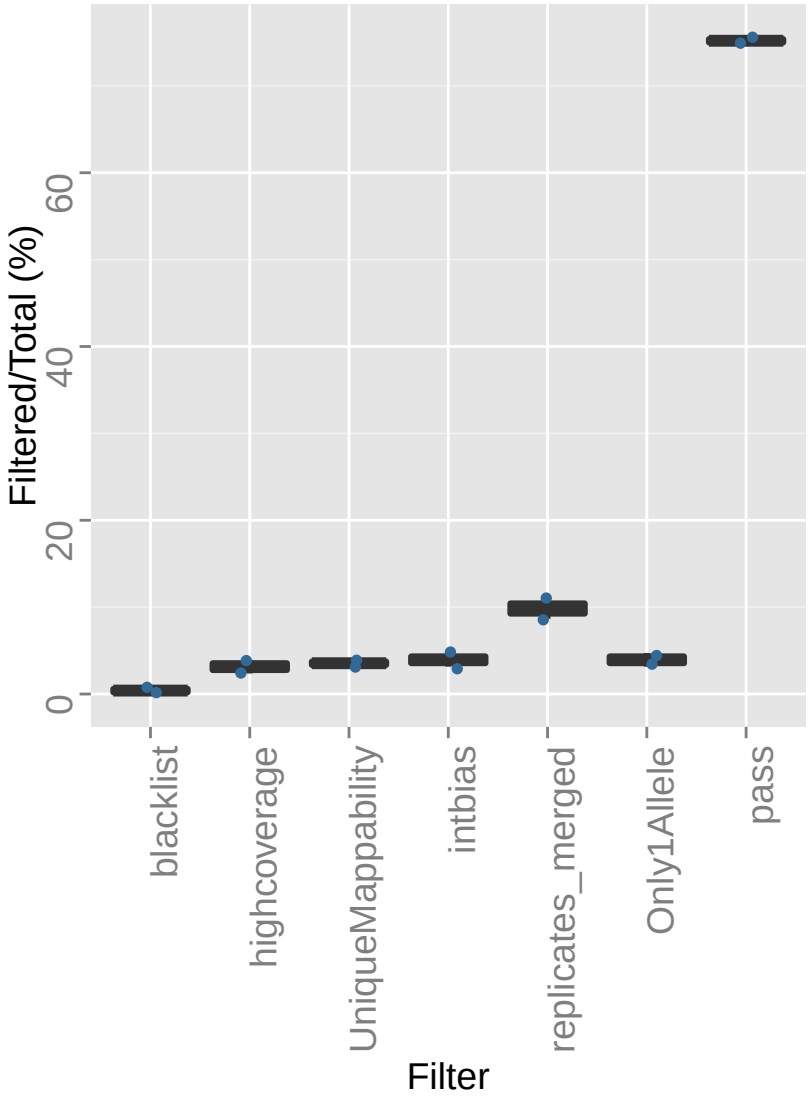
```
data(blacklist_hg19)
data(pickrell2011cov1_hg19)
data(UniqueMappability50bp_hg19)

res <- QCfilter(res,
                RegionsToFilter=list("blacklist"=blacklist_hg19, "highcoverage"=pickrell2011cov1_hg19),
                RegionsToKeep=list("UniqueMappability"=UniqueMappability50bp_hg19))
res <- mergePerGroup(res)
res <- filter1allele(res)
```

5 Summarizing and plotting QC data

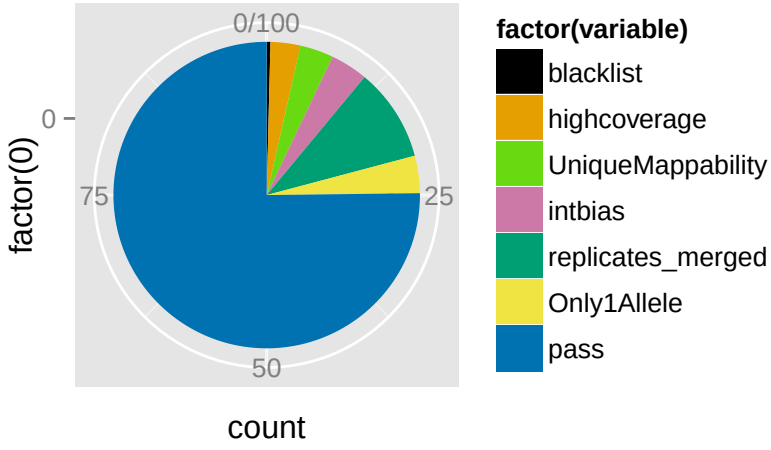
```
plotQC(res, "barplot_per_group")
```




```
plotQC(res, "boxplot_per_filter")
```




```
plotQC(res, "overall_pie")
```




```
summaryQC(res)
```

```
## $filtering_stats
##         blacklist highcoverage UniqueMappability intbias replicates_merged Only1Allele
## MCF7            1            5                 4       4                14           4
## GM12891         0            3                 5       6                11           6
##         pass
## MCF7      95
## GM12891   96
##
## $average_stats
##            variable value.mean       perc
## 1         blacklist        0.5  0.3937008
## 2      highcoverage        4.0  3.1496063
## 3 UniqueMappability        4.5  3.5433071
## 4           intbias        5.0  3.9370079
## 5 replicates_merged       12.5  9.8425197
## 6       Only1Allele        5.0  3.9370079
```

```
## 7              pass       95.5 75.1968504
```

The function BaalChIP.report outputs a table with all assayed variants and with additional information about their ASB status:

```
result <- BaalChIP.report(res)
head(result[["MCF7"]])
```

```
##           ID CHROM       POS REF ALT REF.counts ALT.counts Total.counts         AR
## 1 rs10169169  chr2 191412889   T   G          4         19           23 0.17391304
## 2  rs1021813  chr3  59413060   T   C          3         18           21 0.14285714
## 3 rs10444404 chr12  15114751   T   G          1         14           15 0.06666667
## 4 rs10495062  chr1 217804955   T   C          2         13           15 0.13333333
## 5 rs10502400 chr18  10353940   A   G          4         17           21 0.19047619
## 6 rs10512030  chr9  76484346   T   C          1         15           16 0.06250000
##      RMbias       RAF Bayes_lower Bayes_upper Corrected.AR isASB
## 1 0.4946235 0.4870296  0.09275682   0.3900729    0.2414148  TRUE
## 2 0.4946235 0.4689580  0.07535283   0.3808823    0.2281176  TRUE
## 3 0.4946235 0.5195654  0.02347753   0.2964224    0.1599500  TRUE
## 4 0.4946235 0.3654244  0.08792807   0.5073374    0.2976327 FALSE
## 5 0.4946235 0.4670719  0.10302328   0.4302656    0.2666444 FALSE
## 6 0.4946235 0.4328198  0.02752114   0.3427208    0.1851210  TRUE
```

Acknowledgements

We thank Thomas Carroll and Gordon Brown for suggestions and advice about ChIP-seq data analysis, and the ENCODE Consortium and the ENCODE production laboratories for generating the datasets used in this study. We thank the "Breast Cancer Research Foundation" for financial support of this work.

Session Info

- R version 3.1.3 (2015-03-09), `x86_64-apple-darwin10.8.0`
- Locale: `en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, stats4, utils
- Other packages: BaalChIP 0.0.3, BiocGenerics 0.12.1, Biostrings 2.34.1, doBy 4.5-13, GenomeInfoDb 1.2.5, GenomicAlignments 1.2.2, GenomicRanges 1.18.4, ggplot2 1.0.1, IRanges 2.0.1, knitr 1.10.5, reshape2 1.4.1, Rsamtools 1.18.3, S4Vectors 0.4.0, survival 2.38-1, XVector 0.6.0
- Loaded via a namespace (and not attached): base64enc 0.1-2, BatchJobs 1.6, BBmisc 1.9, BiocParallel 1.0.3, BiocStyle 1.4.1, bitops 1.0-6, brew 1.0-6, checkmate 1.5.2, codetools 0.2-11, colorspace 1.2-6, DBI 0.3.1, digest 0.6.8, evaluate 0.7, fail 1.2, foreach 1.4.2, formatR 1.2, grid 3.1.3, gtable 0.1.2, highr 0.5, iterators 1.0.7, labeling 0.3, lattice 0.20-31, magrittr 1.5, MASS 7.3-40, Matrix 1.2-0, munsell 0.4.2, plyr 1.8.2, proto 0.3-10, Rcpp 0.11.6, RSQLite 1.0.0, scales 0.2.4, sendmailR 1.2-1, splines 3.1.3, stringi 0.4-1, stringr 1.0.0, tools 3.1.3, zlibbioc 1.12.0

References

Consortium EP. 2012. An integrated encyclopedia of dna elements in the human genome. Nature 489: 5774.

Degner JF, Marioni JC, Pai AA, Pickrell JK, Nkadori E, Gilad Y, and Pritchard JK. 2009. Effect of read-mapping biases on detecting allele-specific expression from rna-sequencing data. Bioinformatics 25: 32073212.

Morgan M, Pags H, Obenchain V and Hayden N (2016). Rsamtools: Binary alignment (BAM), FASTA, variant call (BCF), and tabix file import. R package version 1.24.0, http://bioconductor.org/packages/release/bioc/html/Rsamtools.html.

Pickrell JK, Marioni JC, Pai AA, Degner JF, Engelhardt BE, Nkadori E, Veyrieras JB, Stephens M, Gilad Y, and Pritchard JK. 2010. Understanding mechanisms underlying human gene expression variation with rna sequencing. Nature 464: 768772.

R Core Team. 2014. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.