# Detecting allele-specific events from ChIP-seq data

Ines de Santiago, Wei Liu, Ke Yuan, Florian Markowetz

Edited: 2015; Compiled: June 12, 2015

## 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 (Baysian Anaysis of Allelic imbalances from ChIP-seq data) tests the differential read counts of the alleles at each heterozygous variant using the quantitative information of ChIP-seq read counts at the reference and alternative alleles and accomodating the information about the allele presence and other sources of ChIP-seq mapping biases.

## 2 Example

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

## 2.1 Identifying allele-specific binding (ASB) events from ChIP-seq data obtained from cancer cells

The example dataset 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.

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"))
```

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

The first step is to contruct a BaalChIP object:

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

Given a new BaalChIP object, allele-specific binding events can be identified as follows:

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

#run example
res <- alleleCounts(res, min_base_quality=10, min_mapq=15)
res <- QCfilter(res,
                RegionsToFilter=list("blacklist"=blacklist_hg19,
                                     "highcoverage"=pickrell2011cov1_hg19),
                RegionsToKeep=list("UniqueMappability"=UniqueMappability50bp_hg19))
res <- mergePerGroup(res)
res <- filter1allele(res)
res <- getASB(res, Iter=5000, conf_level=0.95)
```

If you trust the package defaults, the first four steps can be replaced by a wrapper function, making BaalChIP workflow possible to run a 3-step script:

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

The following sections describe these steps in more detail.

## 2.2 Data preparation

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. 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("example.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 previously called peaks.

Note that the sample sheet should be saved as a .csv file, you do not have to first load it into a data frame (the filename is passed directly to BaalChIP).

In addition to the sample sheet, BaalChIP requires a 'variant file' containning 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:

```
head(read.delim("MCF7_hetSNP.txt"))
head(read.delim("GM12891_hetSNP.txt"))
```

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, if missing BaalChIP will still run but will not correct for relative allele frequency (copy-number) bias.

## 2.3 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$_n ames tringsinthe.csvsamplesheet, inthi$

```
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
```

## 2.4 Obtainning allele-specific counts for BAM files

The next step is to compute the read coverage at each allele. BaalChIP will read in the information whithin the samples slot of a BaalChIP object and it will primarly find all variants overlapping peaks. Then, for each variant, computes the number of reads carrying the reference (REF) and alternative (ALT) alleles.

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

## 2.5 QC: Filtering variants that may be problematic for allele-specific detection

BaalChIP contains an extensive set of filters...

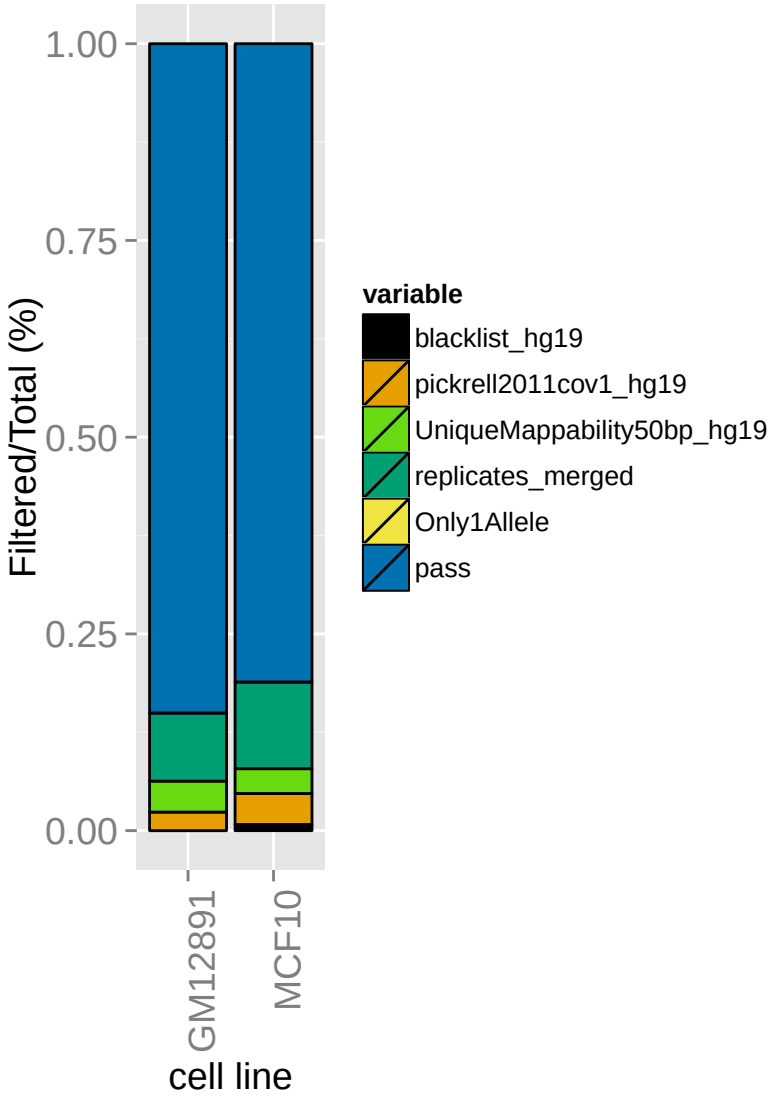
```
res <- QCfilter(res,
                RegionsToFilter=list("blacklist"=blacklist_hg19, "highcoverage"=pickrell2011cov1_hg19),
                RegionsToKeep=list("UniqueMappability"=UniqueMappability50bp_hg19))
res <- mergePerGroup(res)
res <- filter1allele(res)
```

## 2.6 Allele counts and QC in 1-step script

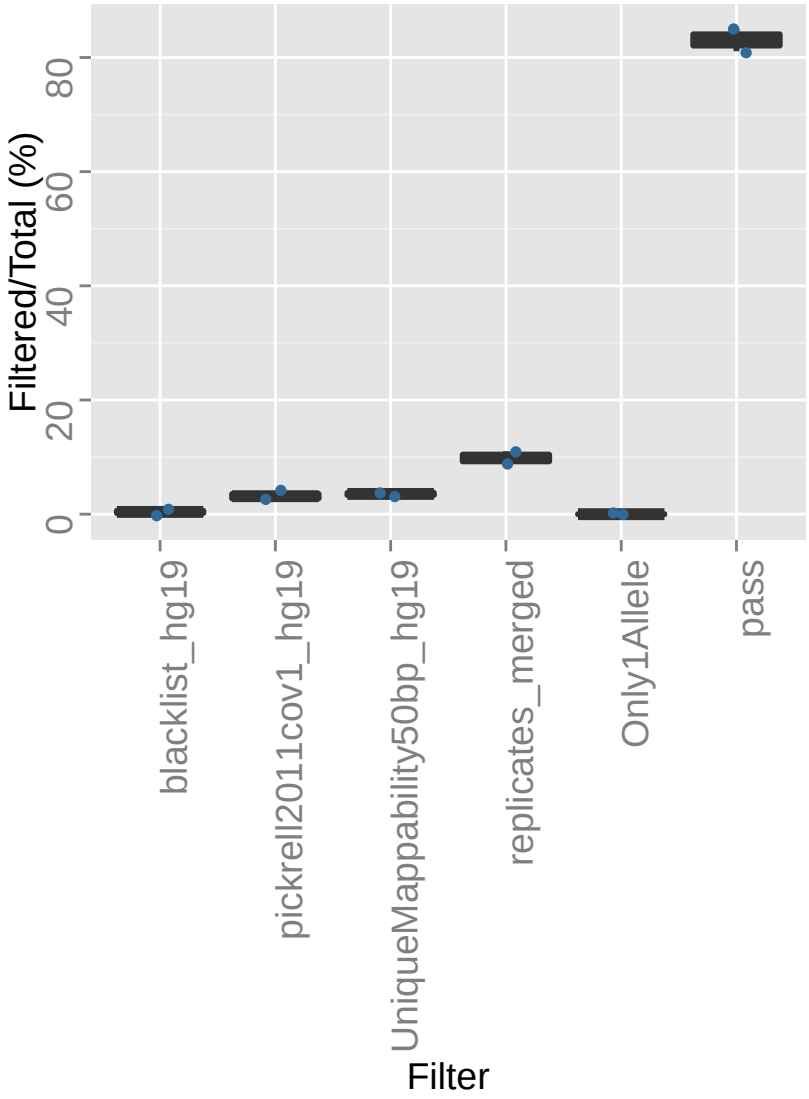
```
res <- new("BaalChIP", samplesheet, hets)
res <- BaalChIP.QC(res)
```

## 2.7 summarysing and plotting QC data

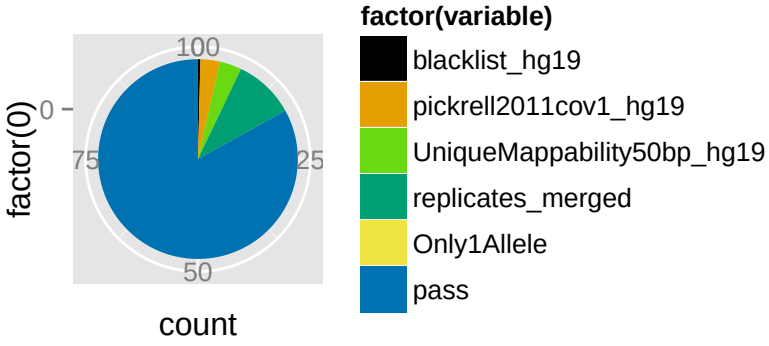
```
data(baalObject)
plotQC(res, "barplot_per_group")
```




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




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




```
summaryQC(res)
```

```
## $filtering_stats
##         blacklist_hg19 pickrell2011cov1_hg19 UniqueMappability50bp_hg19 replicates_merged
## MCF10                1                     5                          4                14
## GM12891              0                     3                          5                11
##         Only1Allele pass
## MCF10             0  103
## GM12891           0  108
##
## $average_stats
##                    variable value.mean       perc
## 1             blacklist_hg19        0.5  0.3937008
## 2      pickrell2011cov1_hg19        4.0  3.1496063
## 3 UniqueMappability50bp_hg19        4.5  3.5433071
## 4          replicates_merged       12.5  9.8425197
## 5                Only1Allele        0.0  0.0000000
## 6                       pass      105.5 83.0708661
```

## 2.8 Identifying allele-specific binding events

### 2.8.1 with RAF copy-number correction

```
res1 <- getASB(res, Iter=5000, conf_level=0.95, RAFcorrection=TRUE)
result_Corrected <- BaalChIP.report(res1)
summaryASB(res1)
```

### 2.8.2 without RAF copy-number correction

```
res2 <- getASB(res, Iter=5000, conf_level=0.95, RAFcorrection=FALSE)
result_NOTcorrected <- BaalChIP.report(res2)
summaryASB(res2)
```

# 3 Acknowledgements

We thank Thomas Carroll and Gordon Brown for suggestions and advice about ChIP-seq data analysis.

# 4 Session Info

```
toLatex(sessionInfo())
```

- 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 0.0.1, 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 1.5.3, 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