Package 'BaalChIP'

November 11, 2026

Title BaalChIP: Bayesian analysis of allele-specific transcription factor binding in cancer genomes

Description The package offers functions to process multiple ChIP-seq BAM files and detect allele-specific events. Computes allele counts at individual variants (SNPs/SNVs), implements extensive QC steps to remove problematic variants, and utilizes a bayesian framework to identify statistically significant allele- specific events. BaalChIP is able to account for copy number differences between the two alleles, a known phenotypical feature of cancer samples.

Version 1.38.0

Author Ines de Santiago, Wei Liu, Ke Yuan, Martin O'Reilly, Chandra SR Chilamakuri, Bruce Ponder, Kerstin Meyer, Florian Markowetz

Maintainer Ines de Santiago <inesdesantiago@gmail.com>

Depends R (>= 3.3.1), GenomicRanges, IRanges, Rsamtools,

Imports GenomicAlignments, GenomeInfoDb, doParallel, parallel, doBy, reshape2, scales, coda, foreach, ggplot2, methods, utils, graphics, stats

Suggests RUnit, BiocGenerics, knitr, rmarkdown, BiocStyle

VignetteBuilder knitr

License Artistic-2.0

LazyData true

RoxygenNote 7.1.1

biocViews Software, ChIPSeq, Bayesian, Sequencing

git_url https://git.bioconductor.org/packages/BaalChIP

git_branch RELEASE_3_23

git_last_commit e83afac

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

adjustmentBaalPlot . . . 2
alleleCounts . . . 3
BaalChIP . . . 4
BaalChIP.get . . . 6
BaalChIP.report . . . 7
BaalChIP.run . . . 9
BaalObject . . . 10
blacklist_hg19 . . . 10
ENCODEexample . . . 11
FAIREexample . . . 11
filter1allele . . . 12
filterIntbias . . . 13
getASB . . . 15
mergePerGroup . . . 16
pickrell2011cov1_hg19 . . . 18
plotQC . . . 18
plotSimul . . . 19
QCfilter . . . 20
summaryASB . . . 21
summaryQC . . . 22
UniqueMappability50bp_hg19 . . . 23

Index **24**

`adjustmentBaalPlot` *Method adjustmentBaalPlot*

Description

Produces a density plot of the distribution of allelic ratios (REF/TOTAL) before and after BaalChIP adjustment for RM and RAF biases.

Usage

```
adjustmentBaalPlot(.Object, col = c("green3", "gray50"))

## S4 method for signature 'BaalChIP'
adjustmentBaalPlot(.Object, col = c("green3", "gray50"))
```

Arguments

`.Object`	An object of the `BaalChIP` class.
`col`	A character vector indicating the colours for the density plot (default is c('green3','gray50')).

Value

A plot

Author(s)

Ines de Santiago

See Also

BaalChIP.report, summaryQC

Examples

```
data('BaalObject')
adjustmentBaalPlot(BaalObject)
adjustmentBaalPlot(BaalObject, col=c('blue','pink'))
```

---

alleleCounts *Method alleleCounts*

Description

Generates allele-specific read count data from each BAM ChIP-seq dataset for each variant.

Usage

```
alleleCounts(.Object, min_base_quality = 10, min_mapq = 15, verbose = TRUE)

## S4 method for signature 'BaalChIP'
alleleCounts(.Object, min_base_quality = 10, min_mapq = 15, verbose = TRUE)
```

Arguments

.Object	An object of the BaalChIP class.
min_base_quality	A numeric value indicating the minimum read base quality below which the base is ignored when summarizing pileup information (default 10).
min_mapq	A numeric value indicating the minimum mapping quality (MAPQ) below which the entire read is ignored (default 15).
verbose	logical. If TRUE reports extra information on the process

Details

Utilizes the information within the samples slot of a BaalChIP object. Will primarily find all variants overlapping peaks. Then, for each variant, computes the number of reads carrying the reference (REF) and alternative (ALT) alleles.

Value

An updated BaalChIP object with the slot alleleCounts containing a list of GRanges objects.

Note

BaalChIP computes allelic counts at each variant position with Rsamtools pileup function. The algorithm follows pileup::Rsamtools by automatically excluding reads flagged as unmapped, secondary, duplicate, or not passing quality controls.

Author(s)

Ines de Santiago

See Also

BaalChIP.get

Examples

```
samplesheet <- system.file("test", "exampleChIP.tsv", package = "BaalChIP")
hets <- c("MCF7"= system.file("test", "MCF7_hetSNP.txt", package = "BaalChIP"),
          "GM12891"= system.file("test", "GM12891_hetSNP.txt", package = "BaalChIP"))
res <- BaalChIP(samplesheet=samplesheet, hets=hets)
res <- alleleCounts(res, min_base_quality=10, min_mapq=15)

#retrieve alleleCounts:
counts <- BaalChIP.get(res, 'alleleCountsPerBam')

#alleleCounts are grouped by bam_name and group_name:
names(counts)
names(counts[['MCF7']])

#check out the result for one of the bam files:
counts[['MCF7']][[1]]
```

BaalChIP *BaalChIP-class*

Description

This S4 class includes a series of methods for detecting allele-specific events from multiple ChIP-seq datasets.

Usage

```
BaalChIP(samplesheet = NULL, hets = NULL, CorrectWithgDNA = list())
```

Arguments

`samplesheet` A character string indicating the filename for a `.tsv` file. Column names in the `.tsv` file should include:

- `group_name`: identifier string to group samples together
- `target`: identifier string for factor (transcription factor, protein)
- `replicate_number`: replicate number of sample
- `bam_name`: file path for BAM file containing aligned reads for ChIP sample. If duplicated reads are flaged they will not be included in the allelic count data
- `bed_name`: path for BED file containing peaks for ChIP sample
- `SampleID`: identifier string for sample. If not given will use <group_name>_<target>_<replicate_num

`hets` A named vector with filenames for the `.txt` variant files to be used. The names in the vector should correspond to `group_name` strings in the `.tsv` samplesheet. Columns names in the `.txt` file should include:

- ID: unique identifier string per variant. Identifiers have to be unique, and no more than one identifier should be present per data record. If there is no identifier available, then use an arbritary name to name each variant
- CHROM: chromosome identifier from the reference genome per variant (same genome build as BAM and BED files provided)
- POS: the reference position (1-based)
- REF: reference base. Each base must be one of A,C,G,T in uppercase. Multiple bases are not permitted
- ALT: alternate non-reference base. Each base must be one of A,C,G,T in uppercase. Multiple bases are not permitted
- RAF: [Optional] a value ranging from 0 to 1 for each variant denoting the relative allele frequency (RAF). 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. If neither RAF or `CorrectWithgDNA` are given, BaalChIP will not correct for relative allele frequency (copy-number) bias. If both RAF and `CorrectWithgDNA` are given, BaalChIP will give priority to the RAF values of the 'hets' files and will use these values to correct for relative allele frequency (copy-number) bias.

`CorrectWithgDNA`
An optional named list with comple file paths for the `.bam` gDNA files to be used. The names in the list should correspond to `group_name` strings in the `.tsv` samplesheet. Allelic read counts from all gDNA files are pooled together to generate the Reference Allelic Ratios (RAF) directly from input data. If missing, BaalChIP will try to read the background allelic ratios from the information in the RAF column of the 'hets' files indicated by the `hets` parameter. If both `RAF` and `CorrectWithgDNA` are missing, BaalChIP will not correct for relative allele frequency (copy-number) bias.

Value

.Object An object of the `BaalChIP` class.

Author(s)

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

Examples

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

BaalChIP.get *Method BaalChIP.get*

Description

Get information from individual slots in a BaalChIP object.

Usage

```
BaalChIP.get(
  .Object,
 what = c("samples", "param", "alleleCountsPerBam", "mergedCounts", "assayedVar",
    "biasTable")
)

## S4 method for signature 'BaalChIP'
BaalChIP.get(
  .Object,
 what = c("samples", "param", "alleleCountsPerBam", "mergedCounts", "assayedVar")
)
```

Arguments

.Object	An object of the BaalChIP class.
what	a single character value specifying which information should be retrieved. Options: 'samples', 'param', 'alleleCountsPerBam', 'mergedCounts', 'assayedVar', 'biasTable'.

Value

The slot content from an object of the BaalChIP class.

Author(s)

Ines de Santiago

Examples

```
data('BaalObject')

#samples data spreadsheet and hets:
BaalChIP.get(BaalObject,what='samples')

#parameters used within run:
BaalChIP.get(BaalObject,what='param')

#retrieve a GRanges list with allele-specific read counts per BAM file:
counts <- BaalChIP.get(BaalObject,what='alleleCountsPerBam')
counts[['MCF7']][[1]]

#retrieve a data.frame with allele-specific read counts per group:
counts <- BaalChIP.get(BaalObject,what='mergedCounts')
head(counts[[1]])
```

`BaalChIP.report` *Method BaalChIP.report*

Description

Generates a data.frame per group with all variants and a label for all identified allele-specific binding (ASB) variants.

Usage

```
BaalChIP.report(.Object)

## S4 method for signature 'BaalChIP'
BaalChIP.report(.Object)
```

Arguments

`.Object`	An object of the `BaalChIP` class.

Details

The reported data frame contains the following columns:

- ID: unique identifier string per analysed variant.
- CHROM: chromosome identifier from the reference genome per variant.
- POS: the reference position (1-based).
- REF: reference base. Each base must be one of A,C,G,T in uppercase.
- ALT: alternate non-reference base. Each base must be one of A,C,G,T in uppercase.
- REF.counts: pooled counts of all reads with the reference allele.
- ALT.counts: pooled counts of all reads with the non-reference allele.

- Total.counts: pooled counts of all reads (REF + ALT).
- AR: allelic ratio calculated directly from sequencing reads (REF / TOTAL).
- RMbias: numerical value indicating the value estimated and applied by BaalChIP for the reference mapping bias. 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 alternative allele.
- RAF: numerical value indicating the value applied by BaalChIP for the relative allele frequency (RAF) bias correction. 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 alternative allele.
- Bayes_lower: lower interval for the estimated allelic ratio (allelic ratio is given by REF / TOTAL).
- Bayes_upper: upper interval for the estimated allelic ratio (allelic ratio is given by REF / TOTAL).
- Corrected.AR: average estimated allelic ratio (average between Bayes_lower and Bayes_upper). 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 alternative allele.
- isASB: logical value indicating BaalChIP's classification of variants into allele-specific.

Value

A named list, with a data.frame per group.

Author(s)

Ines de Santiago

See Also

summaryASB, getASB

Examples

```
data('BaalObject')
report <- BaalChIP.report(BaalObject)

#the reported list is grouped by group_name:
names(report)

#check out the report for one of the groups:
head(report[['MCF7']])
```

BaalChIP.run *Method BaalChIP.run*

Description

BaalChIP.run is a wrapper convenience function, to compute allele counts and perform quality controls in one step. This function will use the package's defaults.

Usage

```
BaalChIP.run(.Object, cores = 4, verbose = TRUE)

## S4 method for signature 'BaalChIP'
BaalChIP.run(.Object, cores = 4, verbose = TRUE)
```

Arguments

.Object	An object of the BaalChIP class.
cores	number of cores for parallel computing (default is 4).
verbose	logical. If TRUE reports extra information on the process

Details

This function is a wrapper of the following functions: alleleCounts, QCfilter, mergePerGroup, filter1allele, getASB

Value

An object of the BaalChIP class.

Author(s)

Ines de Santiago

See Also

summaryQC, plotQC

Examples

```
samplesheet <- system.file("test", "exampleChIP.tsv", package = "BaalChIP")
hets <- c("MCF7"= system.file("test", "MCF7_hetSNP.txt", package = "BaalChIP"),
          "GM12891"= system.file("test", "GM12891_hetSNP.txt", package = "BaalChIP"))
res <- BaalChIP(samplesheet=samplesheet, hets=hets)
res <- BaalChIP.run(res, cores=2)

#summary of the QC step
summaryQC(res)
```

```
#summary of the ASB step
summaryASB(res)
```

`BaalObject` *BaalObject example dataset*

Description

BaalObject example dataset

Format

A BaalChIP-class object

Author(s)

Ines de Santiago <ines.desantiago@cruk.cam.ac.uk>

`blacklist_hg19` *Blacklisted genomic regions*

Description

A GRanges object containing blacklisted regions identified by the ENCODE and modENCODE consortia. These correspond to artifact regions that tend to show artificially high signal (excessive unstructured anomalous reads mapping). Selected from mappability track of the UCSC genome browser (hg19, wgEncodeDacMapabilityConsensusExcludable and wgEncodeDukeMapabilityRegionsExcludable tables).

Code used to retrieve these regions:

- curl http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeMapability/wgEncodeDacMapabilityCor > hg19_DACExcludable.txt

- cat hg19_DUKEExcludable.txt hg19_DACExcludable.txt | grep -v "^#" | cut -f 2,3,4,5,6,7 | sort -k1,1 -k2,2n | mergeBed -nms -i stdin > hg19_DUKE_DAC.bed

Used as 'RegionsToFilter' within the QCfilter function so that variants overlapping these regions will be removed.

Format

A GRanges object of 1378 ranges.

Details

Note that these blacklists are applicable to functional genomic data (e.g. ChIP-seq, MNase-seq, DNase-seq, FAIRE-seq) of short reads (20-100bp reads). These are not applicable to RNA-seq or other transcriptome data types.

Author(s)

Ines de Santiago <ines.desantiago@cruk.cam.ac.uk>

ENCODEexample *ENCODEexample example dataset*

Description

ENCODEexample example dataset

Format

A BaalChIP-class object

Author(s)

Ines de Santiago <ines.desantiago@cruk.cam.ac.uk>

Source

Downloaded from supplementary data of BaalChIP paper

FAIREexample *FAIREexample example dataset*

Description

FAIREexample example dataset

Format

A BaalChIP-class object

Author(s)

Ines de Santiago <ines.desantiago@cruk.cam.ac.uk>

Source

Downloaded from supplementary data of BaalChIP paper

filter1allele *Method filter1allele*

Description

Filters the data frame available within a BaalChIP object (slot mergedCounts). This filter ignores variants for which only one allele is observed after pooling ChIP-seq reads from all datasets.

Usage

```
filter1allele(.Object)

## S4 method for signature 'BaalChIP'
filter1allele(.Object)
```

Arguments

.Object	An object of the BaalChIP class.

Value

An updated BaalChIP object with the slot mergedCounts containing a data.frame of merged samples per group with variants that pass the filter.

Author(s)

Ines de Santiago

See Also

BaalChIP.get, plotQC, summaryQC

Examples

```
samplesheet <- system.file("test", "exampleChIP.tsv", package = "BaalChIP")
hets <- c("MCF7"= system.file("test", "MCF7_hetSNP.txt", package = "BaalChIP"),
         "GM12891"= system.file("test", "GM12891_hetSNP.txt", package = "BaalChIP"))
res <- BaalChIP(samplesheet=samplesheet, hets=hets)
res <- alleleCounts(res, min_base_quality=10, min_mapq=15)
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)
```

```
#retrieve mergedCounts:
counts <- BaalChIP.get(res, 'mergedCounts')

#mergedCounts are grouped by group_name:
names(counts)
sapply(counts, dim)

#check out the result for one of the groups:
head(counts[[1]])
```

`filterIntbias` *Method filterIntbias*

Description

Filters the data frame available within a BaalChIP object (slot `alleleCounts`). This filter performs simulations of reads of the same length as the original ChIP-seq reads, aligns the simulated reads to the genome, calculates the allelic ratios for each variant and finally ignores those variants for which the allelic ratio (REF/TOTAL) is different than 0.5.

Usage

```
filterIntbias(
  .Object,
  simul_output = NULL,
  tmpfile_prefix = NULL,
  simulation_script = "local",
  alignmentSimulArgs = NULL,
  skipScriptRun = FALSE,
  verbose = TRUE
)

## S4 method for signature 'BaalChIP'
filterIntbias(
  .Object,
  simul_output = NULL,
  tmpfile_prefix = NULL,
  simulation_script = "local",
  alignmentSimulArgs = NULL,
  skipScriptRun = FALSE,
  verbose = TRUE
)
```

Arguments

`.Object`	An object of the BaalChIP class.

`simul_output` a non-empty character vector giving the directory of where to save the FASTQ and BAM files generated by the function. If NULL, a random directory under the current working directory will be generated.

`tmpfile_prefix` an optional character vector giving the initial part of the name of the FASTQ and BAM files generated by the function. If NULL, a random name will be generated.

`simulation_script`
the file path for simulation script containing the instructions of simulation and alignment commands. If NULL, the default simulation script distributed with BaalChIP ('extra/simulation_run.sh') will be used.

`alignmentSimulArgs`
a character vector with arguments to the simulation script. If NULL no arguments are passed.

`skipScriptRun` a logical value indicating if simulation BAM files should not be generated. If TRUE BaalChIP will look for the BAM files in the 'simul_output/temp_prefix' (default is FALSE).

`verbose` logical. If TRUE reports extra information on the process

Value

An updated `BaalChIP` object with the slot `alleleCounts` containing a list of GRanges objects that pass filters.

Author(s)

Ines de Santiago

See Also

`BaalChIP.get`, `plotQC`, `summaryQC`

Examples

```
samplesheet <- system.file("test", "exampleChIP.tsv", package = "BaalChIP")
hets <- c("MCF7"= system.file("test", "MCF7_hetSNP.txt", package = "BaalChIP"),
         "GM12891"= system.file("test", "GM12891_hetSNP.txt", package = "BaalChIP"))
res <- BaalChIP(samplesheet=samplesheet, hets=hets)
res <- alleleCounts(res, min_base_quality=10, min_mapq=15)
skipScriptRun=TRUE #For demonstration purposes only (read details in vignette)
res <- filterIntbias(res,
      simul_output=system.file('test/simuloutput',package='BaalChIP'),
      tmpfile_prefix='c67c6ec6c433', skipScriptRun=TRUE)

#check results
plotSimul(res)
summaryQC(res)
```

getASB *Method getASB*

Description

getASB identifies allele-specific binding events using a bayesian framework.

Usage

```
getASB(
  .Object,
  Iter = 5000,
  conf_level = 0.95,
  cores = 4,
  RMcorrection = TRUE,
  RAFcorrection = TRUE,
  verbose = TRUE
)

## S4 method for signature 'BaalChIP'
getASB(
  .Object,
  Iter = 5000,
  conf_level = 0.95,
  cores = 4,
  RMcorrection = TRUE,
  RAFcorrection = TRUE,
  verbose = TRUE
)
```

Arguments

.Object	An object of the BaalChIP class.
Iter	Maximum number of iterations (default 5000).
conf_level	Confidence interval in the estimated allelic ratio (default 0.95).
cores	number of cores for parallel computing (default is 4).
RMcorrection	Logical value indicating if reference mapping (RM) bias should be applied (default TRUE). If FALSE will not correct for reference allele mapping bias. If TRUE will estimate the RM bias from the overall reference allele proportion.
RAFcorrection	Logical value indicating if relative allele frequency (RAF) bias correction should be applied (default TRUE). If TRUE will read RAF values for each variant from hets files (RAF column name). If FALSE will not correct for relative allele frequency bias.
verbose	logical. If TRUE reports extra information on the process

Value

An updated BaalChIP object with the slot ASB containing variants identified as allele-specific.

Author(s)

Wei Liu, Ke Yuan, Ines de Santiago

See Also

summaryASB, BaalChIP.report

Examples

```
samplesheet <- system.file("test", "exampleChIP.tsv", package = "BaalChIP")
hets <- c("MCF7"= system.file("test", "MCF7_hetSNP.txt", package = "BaalChIP"),
          "GM12891"= system.file("test", "GM12891_hetSNP.txt", package = "BaalChIP"))
res <- BaalChIP(samplesheet=samplesheet, hets=hets)
res <- alleleCounts(res, min_base_quality=10, min_mapq=15)
res <- mergePerGroup(res)
res <- getASB(res, cores=2)

#summary - number of significant ASB variants
summaryASB(res)

#report result
res <- BaalChIP.report(res)
```

---

mergePerGroup *Method mergePerGroup*

Description

Merges all ChIP-seq datasets within a group of samples creating a data.frame that contains allele-specific read count data for all variants that need to be analysed.

Usage

```
mergePerGroup(.Object)

## S4 method for signature 'BaalChIP'
mergePerGroup(.Object)
```

Arguments

.Object	An object of the BaalChIP class.

Details

if QCfilter has been applied, will use the most up-to-date variant set available for each individual BAM file (after QC). Missing values are allowed for heterozygous variants that are not available (e.g. do not pass filter for a particular ChIP-seq dataset).

Value

An updated BaalChIP object with the slot mergedCounts containing a data.frame of merged samples per group.

Author(s)

Ines de Santiago

See Also

BaalChIP.get, plotQC, summaryQC

Examples

```
samplesheet <- system.file("test", "exampleChIP.tsv", package = "BaalChIP")
hets <- c("MCF7"= system.file("test", "MCF7_hetSNP.txt", package = "BaalChIP"),
          "GM12891"= system.file("test", "GM12891_hetSNP.txt", package = "BaalChIP"))
res <- BaalChIP(samplesheet=samplesheet, hets=hets)
res <- alleleCounts(res, min_base_quality=10, min_mapq=15)
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)

#retrieve mergedCounts:
counts <- BaalChIP.get(res, 'mergedCounts')

#mergedCounts are grouped by group_name:
names(counts)
sapply(counts, dim)

#check out the result for one of the groups:
head(counts[[1]])
```

pickrell2011cov1_hg19 *Genomic regions of collapsed repeats*

Description

A GRanges object containing collapsed repeat regions at the 0.1% threshold (hg19 reference). Used as 'RegionsToFilter' within the QCfilter function so that variants overlapping these regions will be removed.

Format

A GRanges object of 34359 ranges.

Author(s)

Ines de Santiago <ines.desantiago@cruk.cam.ac.uk>

Source

File available as supplementary data: Pickrell2011_seq.cov1.bed (http://www.ncbi.nlm.nih.gov/pubmed/21690102)

References

Pickrell et al., 2011 (http://www.ncbi.nlm.nih.gov/pubmed/21690102)

plotQC *Method plotQC*

Description

Produces different plots of QC results.

Usage

```
plotQC(.Object, what = "barplot_per_group", addlegend = TRUE, plot = TRUE)

## S4 method for signature 'BaalChIP'
plotQC(.Object, what = "barplot_per_group", addlegend = TRUE, plot = TRUE)
```

Arguments

.Object	An object of the BaalChIP class.
what	A single character value indicating the type of plot. Options: • overall_pie: plots the average percentage of variants in each filter category (averaged across all groups analysed). • boxplot_per_filter: plots the number of variants that were filtered out per filter category. • barplot_per_group: plots the number of variants that were filtered out per group.
addlegend	A logical value indicating if legend should be included in the plot (default TRUE).
plot	a logical value to whether it should plot (TRUE) or not (FALSE). Default is TRUE.

Value

A plot

Author(s)

Ines de Santiago

See Also

BaalChIP.run, summaryQC

Examples

```
data('BaalObject')
plotQC(BaalObject, what = 'overall_pie')
plotQC(BaalObject, what =  'boxplot_per_filter', addlegend=FALSE)
plotQC(BaalObject, what =  'barplot_per_group')
```

---

plotSimul *Method plotSimul*

Description

Produces a plot of the proportion of variants that displayed the correct number of mapped simulated reads.

Usage

```
plotSimul(.Object, plot = TRUE)

## S4 method for signature 'BaalChIP'
plotSimul(.Object, plot = TRUE)
```

Arguments

.Object	An object of the BaalChIP class.
plot	a logical value to whether it should plot (TRUE) or not (FALSE). Default is TRUE.

Value

A plot

Author(s)

Ines de Santiago

Examples

```
data('BaalObject')
plotSimul(BaalObject)
```

QCfilter *Method QCfilter*

Description

Quality control step for removing variants that may be problematic for identification of allele-specific events.

Usage

```
QCfilter(.Object, RegionsToFilter = NULL, RegionsToKeep = NULL, verbose = TRUE)

## S4 method for signature 'BaalChIP'
QCfilter(.Object, RegionsToFilter = NULL, RegionsToKeep = NULL, verbose = TRUE)
```

Arguments

.Object	An object of the BaalChIP class.
RegionsToFilter	a named list of GRanges objects. Variants overlapping these regions will be removed.
RegionsToKeep	a named list of GRanges objects. Works in an oposite way to 'RegionstoFilter', variants NOT overlapping these regions will be removed.
verbose	logical. If TRUE reports extra information on the process

Value

An updated BaalChIP object with the slot alleleCounts containing a list of GRanges objects that pass filters.

Author(s)

Ines de Santiago

See Also

BaalChIP.get, plotQC, summaryQC

Examples

```
samplesheet <- system.file("test", "exampleChIP.tsv", package = "BaalChIP")
hets <- c("MCF7"= system.file("test", "MCF7_hetSNP.txt", package = "BaalChIP"),
         "GM12891"= system.file("test", "GM12891_hetSNP.txt", package = "BaalChIP"))
res <- BaalChIP(samplesheet=samplesheet, hets=hets)
res <- alleleCounts(res, min_base_quality=10, min_mapq=15)
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))

#check results
plotQC(res,'barplot')
summaryQC(res)
```

---

summaryASB *Method summaryASB*

Description

Generates summary of ASB test result.

Usage

```
summaryASB(.Object)

## S4 method for signature 'BaalChIP'
summaryASB(.Object)
```

Arguments

.Object	An object of the BaalChIP class.

Value

A matrix containing the total number of allele-specific variants (TOTAL) and the number of variants allele-specific for the reference (REF) and alternate alleles (ALT).

Author(s)

Ines de Santiago

See Also

getASB, BaalChIP.report

Examples

```
data('BaalObject')
summaryASB(BaalObject)
```

---

summaryQC *Method summaryQC*

Description

Generates summary of QC result.

Usage

```
summaryQC(.Object)

## S4 method for signature 'BaalChIP'
summaryQC(.Object)
```

Arguments

.Object	An object of the BaalChIP class.

Value

A list with two elements:

- filtering_stats containing the number of variants that were filtered out in each filter category and the total number that 'pass' all filters.
- average_stats containing the average number and average percentage of variants in each filter category, averaged across all analysed groups.

Author(s)

Ines de Santiago

See Also

BaalChIP.run, plotQC

Examples

```
data('BaalObject')
summaryQC(BaalObject)
```

UniqueMappability50bp_hg19

Genomic regions of unique mappability

Description

A GRanges object containing unique regions with genomic mappability score equal to 1. Selected from DUKE uniqueness mappability track of the UCSC genome browser (hg19, wgEncodeCrgMapabilityAlign50mer table).

Code used to retrieve these regions:

- curl http://hgdownload.cse.ucsc.edu/gbdb/hg19/bbi/wgEncodeCrgMapabilityAlign50mer.bw > wgEncodeCrgMapabilityAlign50mer.bw
- ./bigWigToBedGraph wgEncodeCrgMapabilityAlign50mer.bw wgEncodeCrgMapabilityAlign50mer.bedgraph
- awk ' if ($4 >= 1) print $0 ' wgEncodeCrgMapabilityAlign50mer.bedgraph > wgEncodeCrgMapabilityAlign50mer_UNIQUEregions.bedgraph

Used as 'RegionsToKeep' within the QCfilter function so that variants NOT overlapping these regions will be removed.

Format

A GRanges object of 9831690 ranges.

Details

These regions are not applicable to longer reads (> 50bp).

Author(s)

Ines de Santiago <ines.desantiago@cruk.cam.ac.uk>

Source

Downloaded from http://hgdownload.cse.ucsc.edu/gbdb/hg19/bbi/wgEncodeCrgMapabilityAlign50mer.bw.

Index

∗ **BaalObject**
BaalObject, 10
∗ **ENCODEexample**
ENCODEexample, 11
∗ **FAIREexample**
FAIREexample, 11
∗ **UniqueMappability50bp_hg19**
UniqueMappability50bp_hg19, 23
∗ **blacklist_hg19**
blacklist_hg19, 10
∗ **pickrell2011cov1_hg19**
pickrell2011cov1_hg19, 18

adjustmentBaalPlot, 2
adjustmentBaalPlot,BaalChIP-method *(adjustmentBaalPlot)*, 2
alleleCounts, 3, *9*
alleleCounts,BaalChIP-method *(alleleCounts)*, 3

BaalChIP, *2–4*, 4, *5–7*, *9*, *12–17*, *19–22*
BaalChIP.get, *4*, 6, *12*, *14*, *17*, *21*
BaalChIP.get,BaalChIP-method *(BaalChIP.get)*, 6
BaalChIP.report, *3*, 7, *16*, *22*
BaalChIP.report,BaalChIP-method *(BaalChIP.report)*, 7
BaalChIP.run, 9, *19*, *22*
BaalChIP.run,BaalChIP-method *(BaalChIP.run)*, 9
BaalObject, 10
blacklist_hg19, 10

ENCODEexample, 11

FAIREexample, 11
filter1allele, *9*, 12
filter1allele,BaalChIP-method *(filter1allele)*, 12
filterIntbias, 13
filterIntbias,BaalChIP-method *(filterIntbias)*, 13

getASB, *8*, *9*, 15, *22*
getASB,BaalChIP-method *(getASB)*, 15

mergePerGroup, *9*, 16
mergePerGroup,BaalChIP-method *(mergePerGroup)*, 16

pickrell2011cov1_hg19, 18
plotQC, *9*, *12*, *14*, *17*, 18, *21*, *22*
plotQC,BaalChIP-method *(plotQC)*, 18
plotSimul, 19
plotSimul,BaalChIP-method *(plotSimul)*, 19

QCfilter, *9*, 20
QCfilter,BaalChIP-method *(QCfilter)*, 20

summaryASB, *8*, *16*, 21
summaryASB,BaalChIP-method *(summaryASB)*, 21
summaryQC, *3*, *9*, *12*, *14*, *17*, *19*, *21*, 22
summaryQC,BaalChIP-method *(summaryQC)*, 22

UniqueMappability50bp_hg19, 23