# Package 'BayesPeak'

October 17, 2017

**Version** 1.28.0

**Date** 2009-11-04

**Title** Bayesian Analysis of ChIP-seq Data

**Author** Christiana Spyrou, Jonathan Cairns, Rory Stark, Andy Lynch, Simon Tavar\{\}\{\}'{e},

**Maintainer** Jonathan Cairns <jonathan.cairns@babraham.ac.uk>

**Depends** R (>= 2.14), IRanges

**Imports** IRanges, graphics

**Suggests** BiocStyle, parallel

**Description** This package is an implementation of the BayesPeak algorithm for peak-calling in ChIP-seq data.

**License** GPL (>= 2)

**biocViews** ChIPSeq

**NeedsCompilation** yes

## R topics documented:

bayespeak *BayesPeak - Bayesian analysis of ChIP-seq data*

## Description

BayesPeak - Bayesian analysis of ChIP-seq data. This function divides the genome into jobs, and performs the BayesPeak algorithm on each using a C backend. The jobs can be performed in parallel, using the package `parallel`. Results are returned in R.

## Usage

```
bayespeak(treatment, control, chr = NULL, start,
end, bin.size = 100L, iterations = 10000L,
repeat.offset = TRUE, into.jobs = TRUE, job.size = 6E6L,
job.overlap = 20L, use.multicore = FALSE,
mc.cores = getOption("mc.cores", 1), snow.cluster,
prior = c(5, 5, 10, 5, 25, 4, 0.5, 5),
report.p.samples = TRUE)
```

## Arguments

`treatment, control`

These arguments should contain the treated ChIP-seq data and the control data, respectively.

Each of these arguments can be:

- a path to a .bed file (this file will be read in as per `read.bed`).
- OR a `data.frame`, which should have columns `"chr", "start", "end", "strand"`.
- OR a `RangedData` object. This object is expected to be split into spaces by chromosome, and should have a data track labelled "strand".

The `control` argument is entirely optional. (Mathematically, leaving this argument out is equivalent to setting gamma = 1 in the model.)

Strand information is expected to be given as "+" or "-".

`chr` Character vector, specifying which chromosomes to restrict analysis to. Chromosome names must be specified exactly as they appear in the treatment and control arguments.

If left as the default value `chr = NULL`, then BayesPeak will find all chromosomes present in the `treatment` file.

`start, end` Numeric. Locations on the chromosome to start and end at, respectively. If unspecified, then the algorithm will start and end at the minimum and maximum reads found in the data, respectively.

`bin.size` Numeric. Reads are collected into bins. This parameter controls the width of each bin. The bin size is related to the mean fragment length in the library being sequenced, and thus a smaller mean fragment may merit a smaller bin size - please see Spyrou et al. (2009) for more information.

`iterations` Numeric. Number of iterations to run the Monte Carlo analysis for.

`repeat.offset` Logical. If `TRUE`, the algorithm is run a second time, this time with the bins offset by `floor(window/2)`.

`into.jobs` Logical. By default, BayesPeak will divide a large region into smaller jobs and analyse each one separately. To prevent this behaviour, set `into.chunks = FALSE`. This may put BayesPeak at increased risk of overflow and underflow issues, and will additionally prevent usage of the parallel processing options.

`job.size` Numeric. The size of the jobs in base pairs, as described above.

`job.overlap` Numeric. Jobs are expanded to overlap each other. This is prevent peaks on the boundary between two jobs being missed. `job.overlap` corresponds to the number of bins by which each job is expanded.

`use.multicore` Logical. If `use.multicore = TRUE`, then the individual chunks will be processed in parallel, using the `mclapply` function.

`mc.cores` Numeric. The number of cores to be used for parallel processing. This argument is passed directly to the `mclapply` function.

`snow.cluster` Cluster object. A cluster to be used for parallel processing, as per the `snow` package. A cluster can be created via the `makeCluster` function.

`prior` Numeric. A vector, specifying the prior on the hyperparameters as follows. We have lambda_0 ~ gamma(alpha_0, beta_0) and lambda_1 ~ gamma(alpha_1, beta_1). Additionally, we have that alpha_0, alpha_1, beta_0, beta_1 all have gamma priors. This argument should be c(alpha_0 shape, alpha_0 scale, beta_0 shape, beta_1 scale, alpha_1 shape, alpha_1 scale, beta_1 shape, beta_1 scale).

`report.p.samples` Logical. If FALSE, do not collect information required for the parameter samples reported in the output. Thus, output\$p.samples will be an empty list. If this information is not required, setting this parameter to FALSE will reduce memory usage.

## Details

BayesPeak uses a fully Bayesian hidden Markov model to detect enriched locations in the genome. The structure accommodates the natural features of the Solexa/Illumina sequencing data and allows for overdispersion in the abundance of reads in different regions. Markov chain Monte Carlo algorithms are applied to estimate the posterior distributions of the model parameters, and posterior probabilities are provided for the sites of interest.

## Value

A list of 4 objects:

- peaks: A `data.frame` corresponding to the bins that BayesPeak has identified as potentially being enriched. `chr, start, end` give the genomic co-ordinates of the bin. `PP` refers to the posterior probability of the bin being enriched. `job` is the number of the job within which the bin was called, which corresponds to a row in the QC data.frame (see below).
- QC: details of each individual job, listed in columns as follows:
  - `calls` is the number of potentially enriched bins identified in a job (i.e. bins with `PP` > 0.01).
  - `score` is simply the proportion of potentially enriched bins with a PP value above 0.5. Intuitively, a larger score is "better", as it indicates that more of the PP values have tended to 0 or 1.
  - `chr, start, end` are the genomic co-ordinates of the job.
  - We report the average value, across iterations of the algorithm, of the important parameters `p, theta, lambda0, lambda1, gamma` and the average log likelihood `loglhood`.

  - var is the variance of the bin counts.
  - autocorr is an estimate of the first order autocorrelation of bin counts.
  - status indicates whether the job was normal, or offset by half a bin width.
- call: the line of code used to run BayesPeak.
- p.samples: A list of matrix objects, each containing parameter samples from the MCMC runs. p.samples[[i]] corresponds to the samples taken in job i. Samples are taken every 10 iterations, with the first half of the run being discarded, and to avoid using too much memory, not all parameters are given. This output can be used to assess convergence e.g. using the CRAN packages coda or boa. (see the vignette - vignette("BayesPeak"))

Note that the raw output of this function is not intended to be used directly as results - the output should be summarized using the summarize.peaks function before using it in later analysis.

## Author(s)

Christiana Spyrou and Jonathan Cairns

## References

Spyrou C, Stark R, Lynch AG, Tavare S BayesPeak: Bayesian analysis of ChIP-seq data, BMC Bioinformatics 2009, 10:299 doi:10.1186/1471-2105-10-299

## See Also

read.bed, summarize.peaks.

## Examples

```
dir <- system.file("extdata", package="BayesPeak")
treatment <- file.path(dir, "H3K4me3reduced.bed")
input <- file.path(dir, "Inputreduced.bed")

##look at specific region 92-95Mb on chromosome 16
##(we've used half the number of iterations here to reduce the time this example takes)
raw.output <- bayespeak(treatment, input, chr = "chr16", start = 9.2E7, end = 9.5E7, iterations = 5000L, use.m
output <- summarize.peaks(raw.output)
output

## Not run:
##analyse all data in file
raw.output.wg <- bayespeak(treatment, input, use.multicore = TRUE)
output <- summarize.peaks(raw.output.wg)

## End(Not run)
```

---

# overfittingDiagnostics

*Overfitting diagnostic functions*

---

## Description

Three functions that provide diagnostic plots and tools to mitigate the effects of overfitting.

## Usage

```
plot.overfitdiag(x, whatX = "lambda1", whatY = "score",
logX = TRUE, logY = FALSE,
main = "Overfitting diagnostic", ...)
identify.overfitdiag(x, whatX = "lambda1", whatY = "score",
logX = TRUE, logY = FALSE,
main = "Overfitting diagnostic", ...)
region.overfitdiag(x, whatX = "lambda1", whatY = "score",
logX = TRUE, logY = FALSE,
main = "Overfitting diagnostic", ...)
```

## Arguments

| | |
|---|---|
| `x` | Raw output from the `bayespeak` function. |
| `whatX, whatY` | Character. The quantities to plot on the X and Y axes. Common choices would be `"lambda1"`, `"score"`, `"calls"`. Any choice in `names(raw.output$QC)` is, in theory, acceptable (except for `"chr"` and `"status"`, which do not correspond to numeric quantities). |
| `logX, logY` | Logical. If TRUE, the quantity on the corresponding axis undergoes a log transformation before being plotted. |
| `main` | Title of plot (corresponds to `main` argument in `plot` function). |
| `...` | Further arguments.<br>• `plot.overfitdiag` passes these through to `plot`.<br>• `identify.overfitdiag` passes these through to `identify`.<br>• `region.overfitdiag` passes these through to `plot.overfitdiag`. |

## Details

These three functions are used to investigate the prevalence of overfitting in a data set, and to aid selection of sensible criteria for performing overfitting corrections.

`plot.overfitdiag` provides a scatterplot of the key parameters associated with jobs. Please see section 9 of the vignette for an description of how to interpret this information.

`identify.overfitdiag` is used after a call `plot.overfitdiag`, with the same arguments, to find out which job was plotted at a particular location. The interface is operated in the same manner as `identify` - left-click on the plot to label the job closest to that point, and right-click on the plot to end this process.

`region.overfitdiag` is used to define an overfit region on the plot, and return the jobs in the region. The function is used in the same manner as `locator` - left-click on the plot to define the vertices of a polygon, and then right click anywhere to close the polygon (there is no need to left-click on the first vertex again). The area selected will be filled in with red hatching. The function then returns the IDs of the jobs in the hatched area. Typically, this output will be used as an `exclude.jobs` argument in `summarize.peaks`

## Value

All three functions output to the active graphical device. In addition, `identify.overfitdiag` and `region.overfitdiag` return integer vectors corresponding to the jobs selected on the plot.

## Author(s)

Jonathan Cairns

## Examples

```
data(raw.output)

plot.overfitdiag(raw.output)

##recreate figures in vignette
plot.overfitdiag(raw.output, whatX="calls", logX = TRUE, whatY = "lambda1", logY = TRUE)
plot.overfitdiag(raw.output, whatX="calls", logX = TRUE, whatY = "score", logY = TRUE)

## Not run:

##identify particular jobs in the plot
plot.overfitdiag(raw.output, whatX="calls", logX = TRUE, whatY = "score", logY = TRUE)
identify.overfitdiag(raw.output, whatX="calls", logX = TRUE, whatY = "score", logY = TRUE)

##define an overfit region
##left-click to define the polygon vertices, right-click to close the polygon
sel <- region.overfitdiag(raw.output, whatX="calls", logX = TRUE, whatY = "score", logY = TRUE)
output <- summarize.peaks(raw.output, exclude.jobs = sel)


## End(Not run)
```

---

# plot.bed *Plot bed file.*

---

## Description

Plot the distribution of reads in a .bed file.

## Usage

```
plot.bed(x, chr, start, end, strand = "+", bin = 50L, ...)
```

## Arguments

| | |
|---|---|
| x | RangedData (from the IRanges package) with a value column entitled "strand". The .bed file to plot. This could have been read in with read.bed, or alternatively by the import function in the rtracklayer library. |
| chr | Character. Chromosome to plot - should be exactly the same as a space in bed. |
| start, end | Integer. Start and end locations on chromosome. |
| strand | Character. Strand to plot - usually either "+" or "-". If the bed argument uses a different convention in its strand column, this can be used instead. |
| bin | Integer. The width of bin to plot. |
| ... | Additional arguments, passed to hist. |

## Details

This function takes the reads in a bed file, and plots a histogram of their locations. This allows us to see the peaks present in the data.

## Value

Plots a histogram on the active graphical device.

## Author(s)

Jonathan Cairns

## References

Spyrou C, Stark R, Lynch AG, Tavare S BayesPeak: Bayesian analysis of ChIP-seq data, BMC Bioinformatics 2009, 10:299 doi:10.1186/1471-2105-10-299

## See Also

read.bed

## Examples

```
dir <- system.file("extdata", package="BayesPeak")
treatment <- file.path(dir, "H3K4me3reduced.bed")
bed <- read.bed(treatment)

##look at region chr16:91000000-94000000
plot.bed(bed, "chr16", 9.1E7, 9.4E7)
```

---

# plot.job *Plot the called peaks in a job.*

---

## Description

Plot the distribution of reads in a .bed file, with BayesPeak's calls highlighted.

## Usage

```
plot.job(x, raw.out, job, strand = "+", threshold = 0.5,
xlim = c(0,1), highlight = TRUE, col.un = "grey",
col.enr = "blue", bin = 100L, ...)
```

## Arguments

| | |
|---|---|
| x | RangedData (from the IRanges package) with a value column entitled "strand". The .bed file to plot. This could have been read in with read.bed, or alternatively by the import function in the rtracklayer library. |
| raw.out | Raw output from the bayespeak function. |
| job | Integer. The number of the job to plot. |
| strand | Character. Strand to plot - usually either "+" or "-". If the bed argument uses a different convention in its strand column, this can be used instead. |
| threshold | Numeric. Bins with a PP higher than this value will be classed as enriched. |

| | |
|---|---|
| xlim | Numeric vector. This controls which part of the job is plotted. For example, c(0.5,1) would plot the last half of the job, whereas c(0.25,0.75) would plot the central half. |
| highlight | Logical. FIXME |
| col.un | The colour used to plot counts in unenriched bins. |
| col.enr | The colour used to plot counts in enriched bins. |
| bin | What sized bin should be used? Currently, this value should be the same as the value used in bayespeak. The function will behave strangely if this parameter is altered, particularly to pathological values. |
| ... | Additional arguments to be passed through to hist. |

## Details

Similar to plot.bed, plot.job takes the reads in a bed file, and plots a histogram of their locations - i.e. plots the bin counts. It then goes on to highlight the histogram bins that have been made in the raw.output from bayespeak.

It is worth bearing in mind that BayesPeak takes the information on both strands into account when calling peaks, and therefore judgements based on a one-stranded view of the data should be treated with caution. For a better picture of what is going on, both strands should be viewed simultaneously, as is done in the examples below.

## Value

Plots a histogram on the active graphical device.

## Author(s)

Jonathan Cairns

## References

Spyrou C, Stark R, Lynch AG, Tavare S BayesPeak: Bayesian analysis of ChIP-seq data, BMC Bioinformatics 2009, 10:299 doi:10.1186/1471-2105-10-299

## See Also

bayespeak, read.bed, plot.bed.

## Examples

```
##get the ChIP .bed file
dir <- system.file("extdata", package="BayesPeak")
treatment <- file.path(dir, "H3K4me3reduced.bed")
bed <- read.bed(treatment)

##get the corresponding raw.output object
data(raw.output.H3K4me3)

##plot job 1, + and - strand
par(mfrow = c(2,1))
plot.job(bed, raw.output.H3K4me3, 1)
plot.job(bed, raw.output.H3K4me3, 1, "-")
```

```
##zoom in for a closer look...
plot.job(bed, raw.output.H3K4me3, 1, xlim = c(0.58,0.6))
plot.job(bed, raw.output.H3K4me3, 1, "-", xlim = c(0.58,0.6))
```

---

plot.PP *Job PP plot.*

---

## Description

Plot the PP profile of a particular job.

## Usage

```
plot.PP(x, job, breaks = 150L, ...)
```

## Arguments

| | |
|---|---|
| x | Raw output from the bayespeak function. |
| job | Integer. Which of the jobs in the output should have its PP values plotted? |
| breaks | Integer. Analogous to the breaks argument in hist. |
| ... | Additional arguments passed to the hist function. |

## Details

plot.PP plots a histogram of the PP values returned in a particular BayesPeak job. This can be used to identify overfitting in a particular job. It is not suitable for identifying the prevalence of overfitting in all of the jobs in a genome-wide analysis - for that, please see plot.overfitdiag.

## Value

Plots a histogram on the active graphical device.

## Author(s)

Jonathan Cairns

## References

Spyrou C, Stark R, Lynch AG, Tavare S BayesPeak: Bayesian analysis of ChIP-seq data, BMC Bioinformatics 2009, 10:299 doi:10.1186/1471-2105-10-299

## See Also

bayespeak.

## Examples

```
##recreation of the plots in the vignette

data(raw.output) ##output from bayespeak()
plot.PP(raw.output, 324, ylim = c(0,50))
plot.PP(raw.output, 325, ylim = c(0,50))
```

---

`raw.output` *Example raw.output object*

---

## Description

This data set is an example of the output obtained from the bayespeak() function, in particular as an example of job parameters. The ChIP-seq experiment in question investigates ER binding in cells from the MCF7 cell line.

To keep the size of BayesPeak down, raw.output$peaks has been truncated - only the peaks on chromosome 16 are given. raw.output$QC has not been truncated in any way.

## Usage

```
raw.output
```

## Format

A list of 3 objects. See bayespeak for more details.

## References

Many thanks to Dr. Jason Carroll's group for permission to use this data set.

---

`raw.output.H3K4me3` *Example raw.output object from H3K4me3 data*

---

## Description

This data set is an example of the output obtained from the bayespeak function. The ChIP-seq experiment in question investigates H3K4me4 methylation data, in a short region of mouse chromosome 16.

The bed files used to generate this object are included in the BayesPeak package - please see the examples section in the bayespeak help file for the code that can be used to recreate this object.

The raw output has not been truncated in any way.

## Usage

```
raw.output.H3K4me3
```

### Format

A list of 3 objects. See bayespeak for more details.

### References

Many thanks to Dr. Duncan Odom's group for permission to use this data set.

---

`read.bed` *BayesPeak - Bayesian analysis of ChIP-seq data*

---

### Description

Read a .bed file into a data frame, but only the chr, start, end and strand columns.

### Usage

```
read.bed(filename, chr)
```

### Arguments

| | |
|---|---|
| `filename` | Character - The path to the .bed file in question. |
| `chr` | Character vector, specifying which chromosomes to read in. Chromosome names must be specified exactly as they appear in the .bed files.<br>If `chr` is missing, then `read.bed` will read in the entire data set. |

### Details

The purpose of this function is to extract 4 columns from a bed file: chromosome, start, end and strand. These are assumed to be in columns 1, 2, 3 and 6 respectively.

If the first line begins with "track" then it will be skipped.

The strand sense is expected to be given as "+"/"-".

### Value

A RangedData object, split into spaces by chromosome. This object has a "strand" data track.

See the IRanges package vignette for more information.

### Author(s)

Jonathan Cairns

### References

UCSC BED format FAQ - http://genome.ucsc.edu/FAQ/FAQformat.html#format1

### See Also

bayespeak.

## Examples

```
dir <- system.file("extdata", package="BayesPeak")
file <- file.path(dir, "H3K4me3reduced.bed")

treatment <- read.bed(file)
treatment
```

---

`summarize.peaks` *Summarize Peaks*

---

## Description

Summarize Peaks - Combine the potentially enriched bins found by BayesPeak into contiguous peaks, and associate each with a posterior probability. `summarise.peaks` is an alias for `summarize.peaks`

## Usage

```
summarize.peaks(x, threshold = 0.5, method = c("lowerbound", "max"), exclude.jobs = NULL)
summarise.peaks(x, threshold = 0.5, method = c("lowerbound", "max"), exclude.jobs = NULL)
```

## Arguments

| | |
|---|---|
| `x` | Raw output from the function bayespeak. |
| `threshold` | Numeric vector. `threshold` must have length equal to either 1 or `nrow(x$QC)` (i.e. the number of jobs).<br>• If `threshold` is of length 1, then for each job, all bins with a posterior probability (PP) lower than `threshold` will be discarded before summarising.<br>• If `threshold` is of length `nrow(x$QC)`, then jobs are taken to have separate thresholds - in other words, bins in job i will be discarded if they have a PP less than threshold[i]. Note that this behaviour is irrespective of how many jobs are excluded (see the exclude.jobs argument below) - excluded jobs are still assigned a PP threshold, which is essentially ignored. |
| `method` | The method used to combine the posterior probabilities of multiple peaks. Current methods are:<br>• `lowerbound`: A lower bound is found for the posterior probability of the region containing a peak. In contiguous regions of moderately high probability, this method should report a fairer value than `method = max`. Suppose we have a set of $n$ non-intersecting calls within our region, with posterior probabilities $p_1$ to $p_n$ respectively of containing peaks. Then the probability of there being a peak in this region is at least $1 - \prod_{i=1}^{n}(1 - p_i)$. We maximise this over all possible sets of non-intersecting calls within the region. (Usually, this will simply be a choice between exclusively using the offset or the non-offset analyses.)<br>• `max`: Combined region has probability equal to the maximum posterior probability over all the peaks it contains. |
| `exclude.jobs` | A vector of integers, denoting jobs to be excluded from later analysis. Alternatively, a logical vector (to be passed through `which()`). |

## Value

A RangedData object corresponding to the peaks called - each range has an associated PP (Posterior Probability) value.

## Author(s)

Jonathan Cairns

## See Also

bayespeak.

## Examples

```
dir <- system.file("extdata", package="BayesPeak")
treatment <- file.path(dir, "H3K4me3reduced.bed")
input <- file.path(dir, "Inputreduced.bed")

##look at specific region 92-95Mb on chromosome 16
##(we've used half the number of iterations here to reduce the time this example takes)
raw.output <- bayespeak(treatment, input, chr = "chr16", start = 9.2E7, end = 9.5E7, iterations = 5000L, use.m
output <- summarize.peaks(raw.output)
output

##higher threshold
output.ht <- summarize.peaks(raw.output, threshold = 0.9)
output.ht
```

# Index