Package 'GreyListChIP'

November 29, 2024

Type Package

Version 1.38.0

Title Grey Lists -- Mask Artefact Regions Based on ChIP Inputs

Date 2020-07-29

Description Identify regions of ChIP experiments with high signal in the input, that lead to spurious peaks during peak calling. Remove reads aligning to these regions prior to peak calling, for cleaner ChIP analysis.

License Artistic-2.0

LazyLoad yes

Depends R (>= 4.0), methods, GenomicRanges

Imports GenomicAlignments, BSgenome, Rsamtools, rtracklayer, MASS, parallel, GenomeInfoDb, SummarizedExperiment, stats, utils

Suggests BiocStyle, BiocGenerics, RUnit, BSgenome.Hsapiens.UCSC.hg19

biocViews ChIPSeq, Alignment, Preprocessing, DifferentialPeakCalling, Sequencing, GenomeAnnotation, Coverage

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

git_branch RELEASE_3_20

git_last_commit e147efc

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-11-28

Author Matt Eldridge [cre], Gord Brown [aut]

Maintainer Matt Eldridge <matthew.eldridge@cruk.cam.ac.uk>

Contents

`calcThreshold-methods` *Calculate Read Count Threshold*

Description

Based on the counts from `countReads`, sample counts from the set several times, estimate the parameters of the negative binomial distribution for each sample, then calculate the mean of the parameters (*size* and *mu*). Use these values to calculate the read count threshold, given the specified p-value threshold.

Usage

```
calcThreshold(obj, reps=100, sampleSize=30000, p=0.99,cores=1)
```

Arguments

`obj`	A `GreyList` object for which to calculate the threshold.
`reps`	The number of times to sample bins and estimate the parameters of the negative binomial distribution.
`sampleSize`	The number of bins to sample on each repetition.
`p`	The p-value threshold for marking bins as "grey".
`cores`	The number of CPU cores (parallel threads) to use when sampling repeatedly from the set of counts

Details

This method samples from the set of counts generated during the `countReads` step. Each sample is fitted to the negative binomial distribution, and the parameters estimated. The means of the `mu` and `size` parameters is calculated, then used to choose a read count threshold, given the p-value cutoff provided. If `cores` is given, the process will use that many cores to parallelize the parameter estimation.

Value

The modified `GreyList` object, with the threshold added.

Author(s)

Gord Brown

References

Venables, W. N. and Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth edition. Springer.

Examples

```
# Load a pre-built R object with counts.
data(greyList)

# Calculate the threshold:
gl <- calcThreshold(greyList,reps=10,sampleSize=1000,p=0.99,cores=1)
```

`ce10.blacklist` *A* `GRanges` *object representing ENCODE signal artefact or "black list" regions.*

Description

These objects are made from the so-called black list or signal artefact regions defined by Anshul Kundaje and Alan Boyle as part of the ENCODE and modENCODE projects. They are regions which show a signal in essentially any ChIP or similar experiment. For this reason it is useful to remove reads aligning to these regions before carrying out any sort of functional genomics analysis.

These objects can be merged with `GreyList` objects to perform grey list and black list filtering in one step. They are included in the package as a convenience, with the permission of the creators.

The package includes black lists for worm (ce10, ce11), fly (dm3, dm6), human (hg19, grch37, hg38, grch38), and mouse (mm9, mm10). (Note that hg19 and grch37 differ only in chromosome naming, and likewise for hg38 and grch38.)

Usage

```
data(ce10.blacklist,package=GreyListChIP)
data(ce11.blacklist,package=GreyListChIP)
data(dm3.blacklist,package=GreyListChIP)
data(dm6.blacklist,package=GreyListChIP)
data(grch37.blacklist,package=GreyListChIP)
data(hg19.blacklist,package=GreyListChIP)
data(grch38.blacklist,package=GreyListChIP)
data(hg38.blacklist,package=GreyListChIP)
data(mm9.blacklist,package=GreyListChIP)
data(mm10.blacklist,package=GreyListChIP)
```

Format

An S4 `GRanges` object.

Value

A set of intervals defining the black list regions for the named genome.

Source

https://sites.google.com/site/anshulkundaje/projects/blacklists

References

Amemiya HM, Kundaje A, Boyle AP. The ENCODE blacklist: identification of problematic regions of the genome. Sci Rep. 2019 Dec; 9(1) 9354 DOI: 10.1038/s41598-019-45839-z.

ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. Nature. 2012 Sep 6;489(7414):57-74. doi: 10.1038/nature11247.

`countReads-methods` *Count reads from a* `BamFile`

Description

Given the tiling of the genome created when the `GreyList` object was created (or replaced via `getKaryotype`), count reads overlapping the bins, in preparation for estimating the threshold for grey-listing bins.

Usage

```
countReads(obj,bamFile,yieldSize=NA_integer_)
```

Arguments

`obj`	A `GreyList` object on which to count reads.
`bamFile`	A `BamFile` from which to count reads.
`yieldSize`	Number of records to yield each time the BAM file is read.

Details

This method counts reads contained within the bins that make up the genome tiling. Bins are overlapping (by default 1Kb bins at 512b intervals) so reads are counted once for each bin that wholly contains them.

Setting the `yieldSize` can help control the memory usage. If unset, a value of 1,000,000 is used by the `summarizeOverlaps` function from the `GenomicAlignments` package that this method calls. Setting the `yieldSize` to a lower value will reduce the memory requirement at the expense of longer run times.

Value

The modified `GreyList` object, with added counts.

Author(s)

Gord Brown

See Also

GreyList, BamFile

Examples

```
# Load a pre-built GreyList object.
data(greyList)

path <- system.file("extra", package="GreyListChIP")
## Not run: fn <- file.path(path,"sample_chr21.bam")
## Not run: gl <- countReads(greyList,fn)
```

---

getKaryotype-methods *Replace the karyotype of a* GreyList *object*

Description

Though a BSgenome object (or a karyotype file) is supplied when the GreyList object is created, it is conceivable that the user might want to replace it. This method allows that.

Usage

```
getKaryotype(obj, genome, tileSize=1024)
```

Arguments

obj	A GreyList object.
genome	A BSgenome object, from which to take the karyotype.
tileSize	The size in nucleotides of each tile. Overlapping tiles will be generated, spaced at 1/2 the width of the tiles.

Value

Returns the GreyList object with a new genome and tiling.

Author(s)

Gord Brown

See Also

GreyList, BSgenome

Examples

```
# Load a pre-built GreyList objct.
data(greyList)
library(BSgenome.Hsapiens.UCSC.hg19)

# Replace the karyotype, updating the genome tiling.
## Not run: gl <- getKaryotype(greyList,BSgenome.Hsapiens.UCSC.hg19)
```

greyList *A sample* GreyList *object for use in examples.*

Description

This is a sample GreyList object, covering only human chromosome 21 (from genome version hg19). The input library used to generate this grey list can be found in the European Nucleotide Archive, under accession number ERR336953.

The library was made from a culture of the MCF-7 cell line, bought from ATCC. The library was sequenced to a depth of 35,716,191 reads on an Illumina Genome Analyzer IIx. The reads were aligned to human reference genome hg19 (GRCh37) using BWA version 0.7.5a with default parameters. Approximately 96% of reads aligned to the reference genome. Reads aligning to chromosome 21 were extracted using `Samtools`. The chromosome 21-only karyotype file was created by deleting all lines except chromosome 21, in a file generated by `fetchChromSizes` as described in the vignette. This package was then used to create the GreyList sample object.

When printed, the object displays several important slots in the object (if they have been filled with calculated values). For example, this object has all its slots filled, indicating that the analysis is complete:

```
GreyList on karyotype file karyotype_chr21.txt
  tiles: 94004
  files: jc899_chr21.bam
  size (mean): 0.370332362145541
  mu (mean): 10.2330008719269
  params: reps=10, sample size=1000, p-value=0.99
  threshold: 81
  regions: 118
  coverage: 4.45%
```

The fields are described in the class's documentation, but briefly, we can see:

1. the name of the karyotype file (or `BSgenome` object),
2. the number of tiles (overlapping by 1/2 the tile width),
3. the BAM file(s) used for read counting (currently only 1 is allowed),
4. the two estimated parameters of the negative binomial distribution, `size` and `mu`,
5. the input parameters,
6. the calculated read depth cutoff (over 1kb tiles),
7. the number of distinct regions, and
8. the percentage coverage of the reference genome.

The fact that all the fields are present indicates that the regions have been generated; otherwise fields still without values would be omitted. Of course any stage can be re-run with different parameters.

Usage

```
data(greyList)
```

Format

An S4 GreyList object.

Value

A GreyList object named greyList.

GreyList-class "GreyList" *Objects*

Description

Regions of high signal in the input samples of a ChIP experiment can lead to artefacts in peak calling. This class generates "grey lists" of such regions, for use in filtering reads before peak calling (or filtering peaks after peak calling, though it is generally safer to filter first).

Objects from the Class

Objects can be created by calls of the form new("GreyList", genome, ...), where genome is a "BSgenome" object describing a genome, such as BSgenome.Hsapiens.UCSC.hg19. Alternatively, a karyotype file can be provided explictly: new("GreyList", karyoFile=fn, ...). Either genome or karyoFile must be provided; if both are present, the BSgenome object takes precedence. Alternatively, an explicit list of regions may be provided as a GRanges object.

Slots

genome: The BSgenome object corresponding with the genome the reads are aligned to

karyotype: The Seqinfo object from the BSgenome object, or made from the karyo_file

karyo_file: The name of a file containing chromosome sizes for the reference genome of interest, one per line, as "chromName chromLength" pairs.

genomeRegions: a GRanges object that defines which regions of the genome should be used to build the grey list. This is to allow the list to be built on just part of the genome.

tiles: A GRanges object with an overlapping tiling of the genome (by default 1Kb tiles every 512b).

counts: A numeric vector holding the counts corresponding to the tiling and the BAM file provided.

files: A vector of BAM filenames that were used to generate the counts (currently only accepts one).

size_param: The computed estimates of the "size" parameter of the negative binomial distribution, estimated by MASS::fitdistr from repeated sampling from the counts.

size_stderr: The standard errors of the "size" parameters, as estimated by MASS::fitdistr.

size_mean: The mean of the "size" estimates.

mu_param: Computed estimates of the "mu" parameter of the negative binomial distribution, estimated by MASS::fitdistr from repeated sampling from the counts.

mu_stderr: The standard errors of the "mu" parameter.

mu_mean: The mean of the "mu" estimates.

reps: How many samples from the counts were taken.

sample_size: How many values were sampled from the counts, for each estimate of "size" and "mu".

pvalue: The requested p-value threshold.

`threshold`: The calculated threshold, based on the p-value.

`max_gap`: The largest gap to consider when merging nearby regions (i.e. if there are "grey" regions up to this many nucleotides apart, merge them into one long region).

`regions`: A GRanges object defining the final grey list regions.

`coverage`: The percentage of the genome covered by the grey list regions.

Methods

calcThreshold `signature(obj = "GreyList")`: Calculate the cutoff for reads in bins, based on fitting the counts to a negative binomial distribution.

countReads `signature(obj = "GreyList")`: Count reads in bins across the genome.

export `signature(object = "GreyList", con = "character", format = "missing")`: Write the grey list to a file.

initialize `signature(.Object = "GreyList")`: Create an initial object (invoked automatically by `new("GreyList",...)`).

loadKaryotype `signature(obj = "GreyList")`: Load a genome description from a file. The file format is one line per chromosome, with the name of the chromosome followed by white space followed by an integer indicating the length of the chromosome.

getKaryotype `signature(obj = "GreyList")`: Get the karyotype of a genome from a BSgenome object.

setRegions `signature(obj = "GreyList")`: Set the region(s) of a genome to use in making the GreyList object.

makeGreyList `signature(obj = "GreyList")`: Compute the actual grey list, after calculating the threshold.

show `signature(object = "GreyList")`: Display the grey list.

Author(s)

Gord Brown (<gdbzork@gmail.com>)

See Also

BSgenome, Seqinfo, GRanges

Examples

```
showClass("GreyList")

# Load a karyotype file:
path <- system.file("extra", package="GreyListChIP")
fn <- file.path(path,"karyotype_chr21.txt")

# Create a GreyList object:
gl <- new("GreyList",karyoFile=fn)
```

greyListBS *Construct a grey list with default arguments*

Description

This function is a convenience function, wrapping several steps of grey list construction into one step. If you are content to accept the package's defaults, and are using a BSgenome object to supply the karyotype, this function might be of use to you.

Usage

```
greyListBS(genome, bam)
```

Arguments

genome	a BSgenome object, for the relevant genome.
bam	a BAM file to use for making the grey list.

Value

An object of class GreyList.

Author(s)

Gord Brown

See Also

GreyList

Examples

```
# If you want to accept the defaults for everything, you can create the
# GreyList in one step using a BSgenome object:
library(BSgenome.Hsapiens.UCSC.hg19)
path <- system.file("extra", package="GreyListChIP")
## Not run: fn <- file.path(path,"sample_chr21.bam")
## Not run: gl <- greyListBS(BSgenome.Hsapiens.UCSC.hg19,fn)
```

loadKaryotype-methods *Load a karyotype from a file*

Description

Load a karyotype from a file, (re)generate the tiling, for a GreyList object.

Usage

```
loadKaryotype(obj, karyoFile, tileSize=1024)
```

Arguments

obj	A GreyList object to set a new karyotype for.
karyoFile	A text file describing a genome's karyotype. The format is one line per chromosome (or contig or whatever), with the name of the chromosome, some white space, and an integer giving the length of the chromosome in nucleotides.
tileSize	The width of tiles on which to count. Tiles will be placed every tileSize/2 nucleotides, to catch regions of high signal that might otherwise be split across (non-overlapping) tiles and hence missed.

Value

Returns the GreyList object with a new genome and tiling, loaded from the provided file.

Author(s)

Gord Brown

See Also

GreyList

Examples

```
# load a pre-built GreyList object:
data(greyList)

# Get a karyotype file:
path <- system.file("extra", package="GreyListChIP")
fn <- file.path(path,"karyotype_chr21.txt")

# Replace the karyotype in the GreyList:
gl <- loadKaryotype(greyList,fn)
```

---

makeGreyList-methods *Generate a grey list from a GreyList object*

Description

Create the actual grey list, based on the threshold calculated by calcThreshold and the read counts from countReads.

Usage

```
makeGreyList(obj,maxGap=16384)
```

Arguments

obj	The GreyList object to create the list for.
maxGap	If the distance between neighbouring grey regions is less than or equal to maxGap, the regions will be merged into one big region.

Details

Create the grey list as a GRanges object. Merge grey regions if they are separated by up to maxGap bp.

Value

The modified GreyList, with the regions added.

Author(s)

Gord Brown

See Also

GreyList, GRanges

Examples

```
# load a pre-built GreyList object:
data(greyList)

# calculate the actual regions:
gl <- makeGreyList(greyList)
```

---

setRegions-methods *Replace the set of regions of a GreyList object*

Description

Though a BSgenome object (or a karyotype file, or a GRanges) is supplied when the GreyList object is created, it is conceivable that the user might want to replace it. This method allows that.

Usage

```
setRegions(obj, regions, tileSize=1024)
```

Arguments

obj	A GreyList object.
regions	A GRanges object, from which to take the karyotype.
tileSize	The size in nucleotides of each tile. Overlapping tiles will be generated, spaced at 1/2 the width of the tiles.

Value

Returns the GreyList object with a new tiling.

Author(s)

Gord Brown

See Also

GreyList, GRanges

Examples

```
# Load a pre-built GreyList objct.
data(greyList)

# Replace the karyotype, updating the genome tiling.
regions=GRanges(seqnames=Rle(c('chr21','chr21','chr22')),
                ranges=IRanges(c(1,20000,30000),end=c(10000,30000,40000)))
gl <- setRegions(greyList,regions)
```

Index