Generating Grey Lists from Input Libraries

Gord Brown

Edited: 2014-12-29; Compiled: October 1, 2016

Contents

1 Introduction

Many cell lines and tumour samples show anomalous signal in the input or control sample in some regions. These regions also show high signal in the corresponding ChIPs. Peak callers are not, in general, well-behaved in these regions, tending to call many spurious peaks. The purpose of this package is to identify those regions, so that reads in those regions may be removed prior to peak calling, allowing for more accurate insert size estimation and reducing the number of false-positive peaks.

As part of the ENCODE project, Anshul Kundaje identified regions that show enrichment in ChIP experiments independent of what factor is being ChIPped, or what cell line the sample comes from[**?**, **?**]. He called those regions "signal artefact" regions, or colloquially "black lists". We call our lists of high signal grey lists, to distinguish them from ENCODE's black lists, because they are not universal, but rather cell line (or sample) specific, and because they can be tuned depending on the stringency required, and so that we can make jokes about having 50 shades of grey lists[**?**].

This vignette summarizes the construction of grey lists.

2 Generating a Grey List

Generating a grey list involves

1. generating a tiling of the genome,
2. counting reads from a BAM file for the tiling,
3. sampling from the counts and fitting the samples to the negative binomial distribution to calculate the read count threshold,
4. filtering the tiling to identify regions of high signal, then
5. exporting the resulting set to a bed file.

First create the `GreyList` object (this karyotype file includes just human chromosome 21, from reference genome version GRCh37):

```
> library(GreyListChIP)
> path <- system.file("extra", package="GreyListChIP")
> fn <- file.path(path,"karyotype_chr21.txt")
> gl <- new("GreyList",karyoFile=fn)
```

Normally the next step would be to count reads:

```
> gl <- countReads(gl,"myBamFile.bam")
```

but to save time we'll generate some fake data:

```
> gl@counts <- rnbinom(length(gl@tiles),size=1.08,mu=11.54)
```

Now calculate the threshold. The defaults are `reps=100,sampleSize=30000,p=0.99` but for demonstration purposes we'll use smaller values for faster results:

```
> gl <- calcThreshold(gl,reps=10,sampleSize=1000,p=0.99,cores=1)
```

This method fits the sample(s) to the negative binomial distribution, then uses the estimated parameters to identify a read-count threshold[**?**, **?**].

Now generate the grey list itself:

```
> gl <- makeGreyList(gl,maxGap=16384)
```

```
coverage: 2930685 bp (6.09%)
```

```
> gl
```

```
GreyList on karyotype file karyotype_chr21.txt
  tiles: 94004
  size (mean): 1.1016070689771
  mu (mean): 11.3234076282851
  params: reps=10, sample size=1000, p-value=0.99
  threshold: 52
  regions: 706
  coverage: 6.09%
```

(The coverage is higher than normal due to the counts being fake. Normally a threshold of p=0.99 leads to coverage of about 1% of the genome.)

And export it to a file:

```
> export(gl,con="myGreyList.bed")
```

And that's it. If you are happy to accept the package's defaults, you can generate the list in one step (not counting the export step):

```
> library(BSgenome.Hsapiens.UCSC.hg19)
> gl <- greyListBS(BSgenome.Hsapiens.UCSC.hg19,"myBamFile.bam")
> export(gl,con="myGreyList.bed")
```

3 Sample Data

A sample `GreyList` object named `gl` can be obtained, once the package is attached, via:

```
> # Load a pre-built GreyList object named "gl"
> data(greyList)
> print(gl)
```

```
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%
```

This sample object covers only human chromosome 21 (from genome version hg19). The read counts are from an MCF7 input library constructed in the Carroll Lab of Cancer Research UK's Cambridge Institute. See the `greyList` man page for details of this object.

4 Obtaining Karyotypes

If a `BSgenome` object exists for your reference genome of interest, the karyotype is usually most easily obtained via that object. See the `BSgenome` package documentation for a list of available reference genomes[**?**].

Otherwise, if the reference genome is available via the UCSC Genome Browser[**?**], karyotype files can be obtained using the `fetchChromSizes` utility available on the Genome Browser's software download page[**?**].

Failing that, a karyotype file can be constructed by hand using a text editor. The file format is given in the `loadKaryotype` documentation. All that is needed is the names of the chromosomes, (exactly) matching the names in the BAM file, and their lengths in base pairs.

5 Acknowledgements

Thanks to Rory Stark and Tom Carroll for suggestions, advice and encouragement.

6 Session Info

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

- R version 3.3.1 (2016-06-21), `x86_64-apple-darwin13.4.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: BiocGenerics 0.18.0, GenomeInfoDb 1.8.7, GenomicRanges 1.24.3, GreyListChIP 1.4.1, IRanges 2.6.1, S4Vectors 0.10.3
- Loaded via a namespace (and not attached): BSgenome 1.40.1, Biobase 2.32.0, BiocParallel 1.6.6, BiocStyle 2.0.3, Biostrings 2.40.2, GenomicAlignments 1.8.4, MASS 7.3-45, RCurl 1.95-4.8, Rsamtools 1.24.0, SummarizedExperiment 1.2.3, XML 3.98-1.4, XVector 0.12.1, bitops 1.0-6, rtracklayer 1.32.2, tools 3.3.1, zlibbioc 1.18.0