Rcade: R-based analysis of ChIP-seq And Differential Expression data

Jonathan Cairns

April 24, 2017

Contents

1 Introduction

Rcade is a tool that analyses ChIP-seq data and couples the results to an existing Differential Expression (DE) analysis. A key application of *Rcade* is in inferring the direct targets of a transcription factor (TF) - these targets should exhibit TF binding activity, and their expression levels should change in response to a perturbation of the TF.

The ChIP-seq analysis element of *Rcade* is performed through methods from the *baySeq* package, with respect to a user-defined universe of potential binding sites. This means that *Rcade* avoids the noise issues associated with peak-calling and focuses instead on robust quantification of binding activity. Any universe can be selected, and *Rcade* provides functionality to accommodate the common case where bins are defined relative to genomic annotation features.

In some situations, it may be appropriate to define the binding site universe based on a set of peak-calls from other data sets. However, it is inappropriate to use peak-calls from the ChIP-seq data used in the *Rcade* analysis - such an analysis would be prone to confirmation bias.

Rcade uses a fully Bayesian modelling approach. In particular, it uses log-odds values, or B-values, in both its input and output. The log-odds value is related to the posterior probability (PP) of an event, as per the formula $B = \log\left(\frac{PP}{1-PP}\right)$. PP-values should not be confused with the frequentist concept of *p*-values.

2 Model

Rcade perform analysis on any set of genes, with each gene uniquely identified by a gene ID. In the below example, we use Ensembl gene IDs.

Each gene is assumed to have some number of associated binding sites, each of which can be active or inactive as inferred from the ChIP-seq data. Additionally, every gene has one or more expression values, each of which is either DE or not DE under some perturbation - for example, knockdown or stimulation of a TF of interest. It is assumed that, conditional on a gene having both a ChIP-seq signal and a DE signature, the ChIP-seq and expression data associated with that gene are independent. All pairwise interactions between ChIP and DE data are considered for a given gene.

The "gene ID" need not literally refer to genes. For example, a user with transcript-specific expression data could use transcript IDs instead.

3 Simple workflow

In this vignette, we will use the example data provided in the *Rcade* package. These data were obtained from two sources, each pertaining to the transcription factor STAT1. All of the experiments were performed with cells from the HeLa cell line. The two data sets are:

Differential Expression data from Array Express,
`http://www.ebi.ac.uk/arrayexpress`, under accession number E-GEOD-11299. In this experiment, HeLa cells were stimulated with IFNγ, and a time course microarray experiment was performed. We have assessed DE status between the 0h and 6h time points.

STAT1 ChIP-seq data from the Snyder lab, as part of the ENCODE consortium
Input DCC accession numbers: wgEncodeEH000611 and wgEncodeEH000612
ChIP DCC accession number: wgEncodeEH000614
Here, HeLa cells were stimulated with IFNγ for 30 minutes, then cells were harvested and STAT1 ChIP-seq was performed.

To keep the size of the package down, all of these files have been truncated. Thus, they only contain data pertinent to a handful of selected genes.

The location of these data files will vary from system to system. To find the data directory on your computer, use the following code:

```
> dir <- file.path(system.file("extdata", package="Rcade"), "STAT1")
> dir
```

3.1 Preliminary

Load the *Rcade* package:

```
> library(Rcade)
```

To perform an *Rcade* analysis, you will need the following input data:

3.1.1 DE summary

DE matrix: Bayesian information about the DE status of each gene. Any Bayesian source can be used for this purpose:

- *limma*: Full DE results obtained with `topTable(..., number=Inf)`
- *baySeq*.

At time of writing, *edgeR* and *DEseq* do not supply Bayesian output.
The DE data must contain the following fields:
geneID - gene IDs used to link DE results to genes.
logFC - The log fold change associated with each gene.
B - B values (log-odds).

```
> DE <- read.csv(file.path(dir, "DE.csv"))
```

DElookup: We also need to tell *Rcade* which columns of the DE matrix are important. This is done using an object of the form `list(RcadeField1=DEdataField1, ...)`. You may omit the name `RcadeField1` if it is the same as `DEdataField1`. Any fields that are specified in addition to the three required fields above will not be manipulated in the analysis, but will be carried through and appear in the output.

```
> DElookup <- list(GeneID="ENSG", logFC="logFC", B="B",
+ "Genes.Location", "Symbol")
```

3.1.2 ChIP-seq data

.bam and .bai files: Sets of aligned reads. These reads should have already undergone sequence-level pre-processing, such as any read trimming and adaptor removal that may be required. Moreover, they should have appropriate index files - for example, using the `indexBam` function in the package *Rbamtools*. Example .bam and .bai files are provided in the package:

```
> dir(dir, pattern = ".bam")
[1] "wgEncodeSydhTfbsHelas3InputIfng30StdAlnRep1.bam"
[2] "wgEncodeSydhTfbsHelas3InputIfng30StdAlnRep1.bam.bai"
[3] "wgEncodeSydhTfbsHelas3InputStdAlnRep1.bam"
[4] "wgEncodeSydhTfbsHelas3InputStdAlnRep1.bam.bai"
[5] "wgEncodeSydhTfbsHelas3Stat1Ifng30StdAlnRep1.bam"
[6] "wgEncodeSydhTfbsHelas3Stat1Ifng30StdAlnRep1.bam.bai"
[7] "wgEncodeSydhTfbsHelas3Stat1Ifng30StdAlnRep2.bam"
[8] "wgEncodeSydhTfbsHelas3Stat1Ifng30StdAlnRep2.bam.bai"
```

Targets information: A matrix containing information about the .Bam files to be used in the analysis.
Required fields:
fileID – ID associated with the file.
sampleID – ID associated with the sample that was sequenced. Technical replicates of the same population should have the same sampleID.
factor – The antibody used in the experiment. Control files should be labelled "Input".
filepath – The .bam file's name/path.
Optional fields:

shift – Half of the fragment length. That is, before counting, *Rcade* will shift reads on the positive strand forwards by `shift`, and reads on the negative strand backwards by `shift`.

```
> targets <- read.csv(file.path(dir, "targets.csv"), as.is = TRUE)
> targets
  fileID sampleID factor                                          filepath shift
1 Input1   Input1  Input wgEncodeSydhTfbsHelas3InputIfng30StdAlnRep1.bam     0
2 Input2   Input2  Input       wgEncodeSydhTfbsHelas3InputStdAlnRep1.bam     0
3  ChIP1    ChIP1  Stat1 wgEncodeSydhTfbsHelas3Stat1Ifng30StdAlnRep1.bam     0
4  ChIP2    ChIP2  Stat1 wgEncodeSydhTfbsHelas3Stat1Ifng30StdAlnRep2.bam     0
```

3.1.3 Annotation information

In the ChIP-seq analysis, *Rcade* performs its analysis based on the counts in user-defined bin regions. These regions are specified with a `GRanges` object from the *GenomicRanges* package.

A common requirement is to define bins about genomic annotation features: *Rcade* provides simple functionality to generate an appropriate `GRanges` object, through the `defineBins()` function. Since STAT1 is a promoter-bound TF, we define bins about Ensembl-derived transcription start sites.

In this vignette, we use a very reduced annotation file as follows:

```
> anno <- read.csv(file.path(dir, "anno.csv"))
> anno <- anno[order(anno$chromosome_name),]
> colnames(anno) <- c("ENSG","chr","start","end","str")
```

Only use the preceding code to recreate the analysis in this vignette – do not use it at any other point! When analysing genome-wide data, use full annotation information – for example, download full transcript annotation information from Ensembl using *biomaRt*, as follows:

```
> library(biomaRt)
> anno <- getBM(
+                 attributes= c("ensembl_gene_id", "chromosome_name",
+                         "transcript_start", "transcript_end", "strand"),
+                 mart= useDataset("hsapiens_gene_ensembl", useMart("ensembl"))
+         )
> ##order, to reduce size of ChIPannoZones object later
> anno <- anno[order(anno$chromosome_name),]
> ##use appropriate column names
> colnames(anno) <- c("ENSG","chr","start","end","str")
```

We define bins based on the annotation information through the `defineBins()` function, as follows:

```
> ChIPannoZones <- defineBins(anno, zone=c(-1500, 1500), geneID="ENSG")
```

The `zone=c(-1500,1500)` argument defines the zone of interest: it starts 1500bp $5'$ of each TSS (`-1500`), and ends 1500bp $3'$ of each TSS (`1500`). This zone was chosen based on mapping peak-calls to TSSs using the *ChIPpeakAnno* package, and plotting the positional distribution of these peak-calls.

The object *ChIPannoZones* can now be used in the *Rcade* analysis.

3.1.4 Prior specification

By default, *Rcade*'s prior belief is that each gene's DE and ChIP-seq statuses are independent. This is unlikely to be true in real data, as genes with ChIP-seq signal are more likely to be DE than other genes.

Thus, we should give *Rcade* appropriate priors, if possible. For example, the prior dependency can be specified as follows:

```
> DE.prior = 0.01
> prior.mode = "keepChIP"
> prior = c("D|C" = 0.05, "D|notC" = 0.005)
```

We specify `DE.prior = 0.01` because that is the prior probability used by *limma*, the package that our DE analysis was performed with. The remaining settings ensure that genes with ChIP-seq signal have a higher than average probability of DE (`"D|C" = 0.05`), whereas genes without ChIP-seq signal have a lower that average probability of DE (`"D|notC" = 0.005`).

(The values `0.05` and `0.005` were selected arbitrarily, before analysis. An advanced user might select `prior` in a less arbitrary manner – for example, by looking at the overlap between ChIP-seq and DE in "similar" datasets. We do not go into further details of such an analysis here.)

If you do not supply this information, you may get unreasonably small B values at the end of the analysis.

3.2 Analysis function

We can parallelize *Rcade* by supplying a compute cluster (though this is optional):

```
> library(parallel)
> cl <- makeCluster(4, "SOCK")
```

If you don't have a compute cluster or a multicore machine, you can disable parallel processing:

```
> cl <- NULL
```

We now process the data with the `RcadeAnalysis()` function, obtaining an *Rcade* object:

```
> Rcade <- RcadeAnalysis(DE, ChIPannoZones, annoZoneGeneidName="ENSG",
+          ChIPtargets=targets, ChIPfileDir = dir,
+          cl=cl, DE.prior=DE.prior, prior.mode=prior.mode, prior=prior,
+          DElookup=DElookup)
```

.

```
> Rcade
```

```
Rcade: 57 rows, 15 columns.
DE: 59 rows, 13 columns.
ChIP: 1 track(s):
[[1]]
annoZones: GRanges with 48 ranges and 1 elementMetadata col.
shift:0
counts: 48 rows, 4 columns.
summary: 48 rows, 4 columns.

Metadata:
ChIPtargets: 4 rows, 5 columns.
fileDir:/tmp/RtmpAJ4EqG/Rinst67154faacce3/Rcade/extdata/STAT1
```

The *Rcade* object stores information from the DE analysis:

```
> x <- getDE(Rcade)
```

It also contains the ChIP-seq analysis:

```
> x <- getChIP(Rcade)
```

and the Rcade table obtained from linking the ChIP-seq analysis with the DE data:

```
> x <- getRcade(Rcade)
```

4 Plotting, QC

We can perform Principle Component Analysis on the counts with the `plotPCA` function. Usually, one would expect similar samples to cluster together on this plot, and so we can identify samples that do not fit this assumption - further investigation is required to determine the significance of such a finding. (Output is shown in Figure 1.)

```
> plotPCA(Rcade)
```

The MM plot shows log-ratios from DE plotted against log-ratios from the ChIP-seq. The colour of each point corresponds to the probability of both ChIP and DE being present. (Output is shown in Figure 2.)

```
> plotMM(Rcade)
```

The function `plotBBB()` plots log-odds values for ChIP-seq, DE, and combined ChIP-seq/DE analysis together. The package *rgl* is required for this 3D plot. (Output is shown in Figure 3.)

```
> library(rgl)
> plotBBB(Rcade)
```

5 Output

To export the Rcade results to disk in a user-friendly .csv format, use `exportRcade()` function as follows:

```
> exportRcade(Rcade, directory="RcadeOutput")
```

Usually, the file of interest is "DEandChIP.csv", which contains the genes most likely to have both DE and ChIP signals. A full explanation of all of the files can be found in the help page:

```
> ?exportRcade
```

By default, Rcade outputs the top 1000 geneIDs for each hypothesis. To increase the number of geneIDs, use a larger value for cutoffArg:

```
> exportRcade(Rcade, directory="RcadeOutput", cutoffArg=2000)
```

Alternative cutoff methods are described in the `exportRcade` help page - for example,

```
> exportRcade(Rcade, directory="RcadeOutput", cutoffMode="B", cutoffArg=0)
```

6 Questions/Bug reports

If you have any questions about *Rcade*, or encounter a bug, please post a message on the Bioconductor support site at https://support.bioconductor.org/. This will help anybody who later searches for help on the same query.

7 Session Info

```
> sessionInfo()
```

```
R version 3.4.0 (2017-04-21)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.2 LTS

Matrix products: default
BLAS: /home/biocbuild/bbs-3.5-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.5-bioc/R/lib/libRlapack.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C                 LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=C               LC_MONETARY=en_US.UTF-8      LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                    LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] parallel  stats4    stats     graphics  grDevices utils     datasets  methods
[9] base

other attached packages:
 [1] Rcade_1.18.0         baySeq_2.10.0        abind_1.4-5          Rsamtools_1.28.0
 [5] Biostrings_2.44.0    XVector_0.16.0       GenomicRanges_1.28.0 GenomeInfoDb_1.12.0
 [9] IRanges_2.10.0       S4Vectors_0.14.0     BiocGenerics_0.22.0

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.10               compiler_3.4.0             bitops_1.0-6
 [4] tools_3.4.0                zlibbioc_1.22.0            digest_0.6.12
 [7] jsonlite_1.4               evaluate_0.10              lattice_0.20-35
[10] Matrix_1.2-9               DelayedArray_0.2.0         shiny_1.0.2
[13] yaml_2.1.14                GenomeInfoDbData_0.99.0    stringr_1.2.0
[16] knitr_1.15.1               htmlwidgets_0.8            locfit_1.5-9.1
[19] rprojroot_1.2              grid_3.4.0                 Biobase_2.36.0
[22] R6_2.2.0                   plotrix_3.6-4              rgl_0.98.1
[25] BiocParallel_1.10.0        rmarkdown_1.4              limma_3.32.0
[28] magrittr_1.5               edgeR_3.18.0               matrixStats_0.52.2
[31] GenomicAlignments_1.12.0   backports_1.0.5            htmltools_0.3.5
[34] SummarizedExperiment_1.6.0 xtable_1.8-2               BiocStyle_2.4.0
[37] mime_0.5                   httpuv_1.3.3               stringi_1.1.5
[40] RCurl_1.95-4.8
```

8 Acknowledgements

Differential Expression data from Array Express, `http://www.ebi.ac.uk/arrayexpress`, under accession number E-GEOD-11299.

STAT1 ChIP-seq data from the Snyder lab, as part of the ENCODE consortium

Input DCC accession numbers: wgEncodeEH000611 and wgEncodeEH000612
ChIP DCC accession number: wgEncodeEH000614

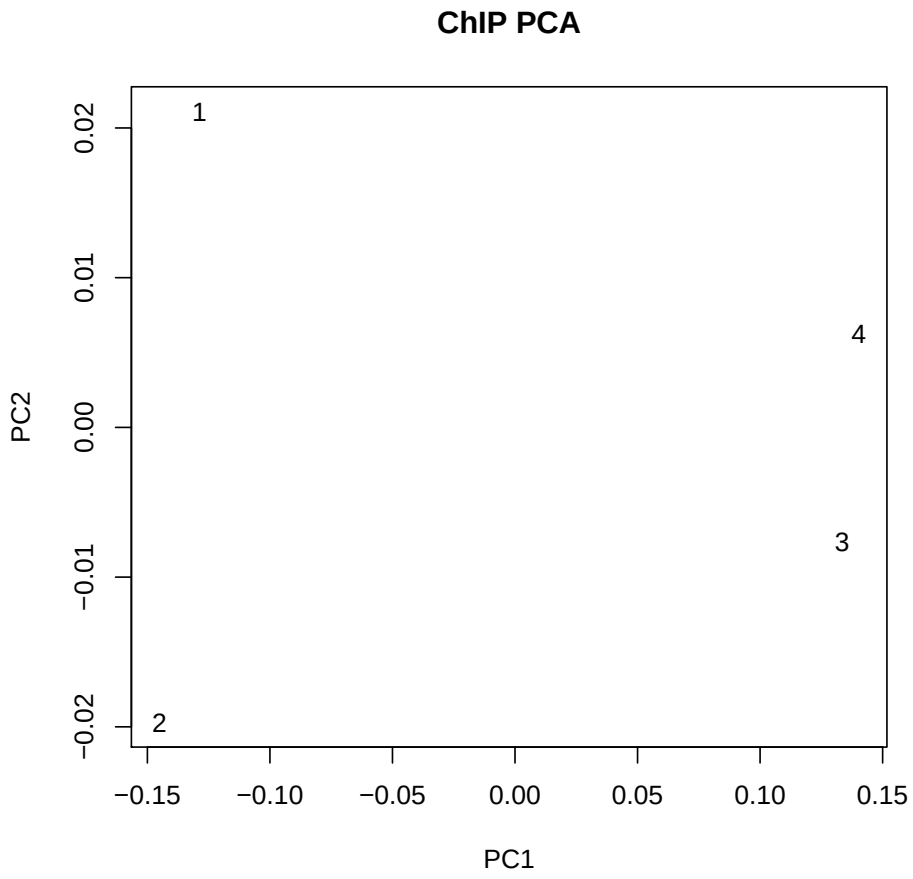

Figure 1: Output from `plotPCA(Rcade)`.

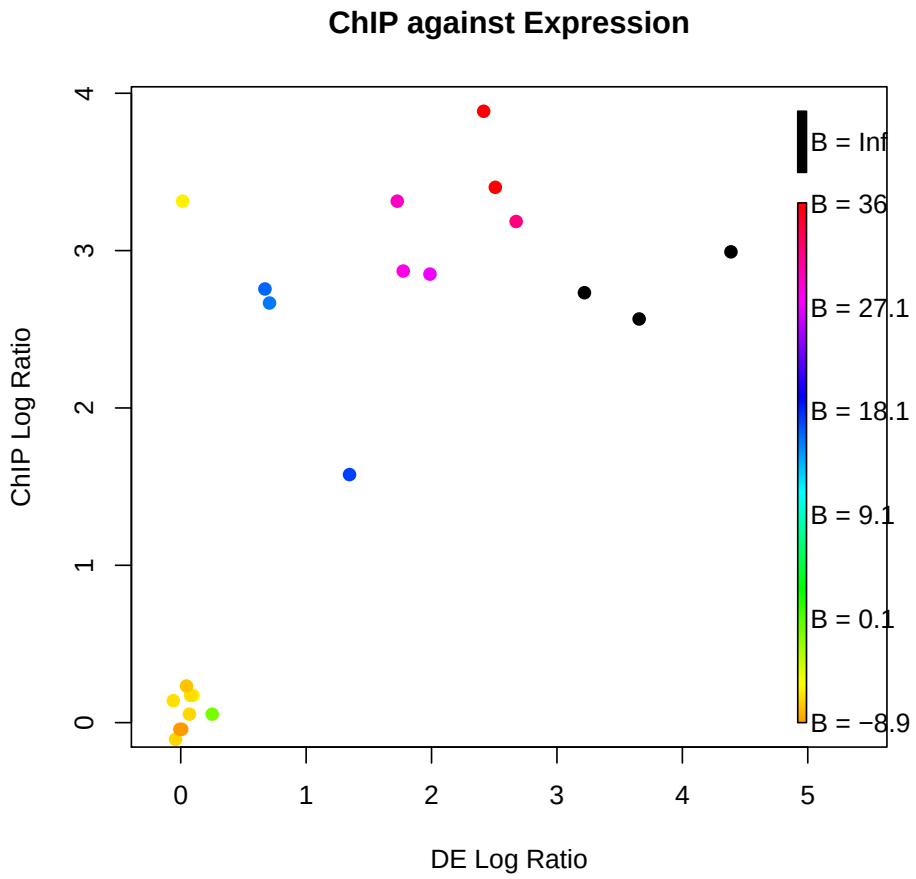

Figure 2: Output from `plotMM(Rcade)`.

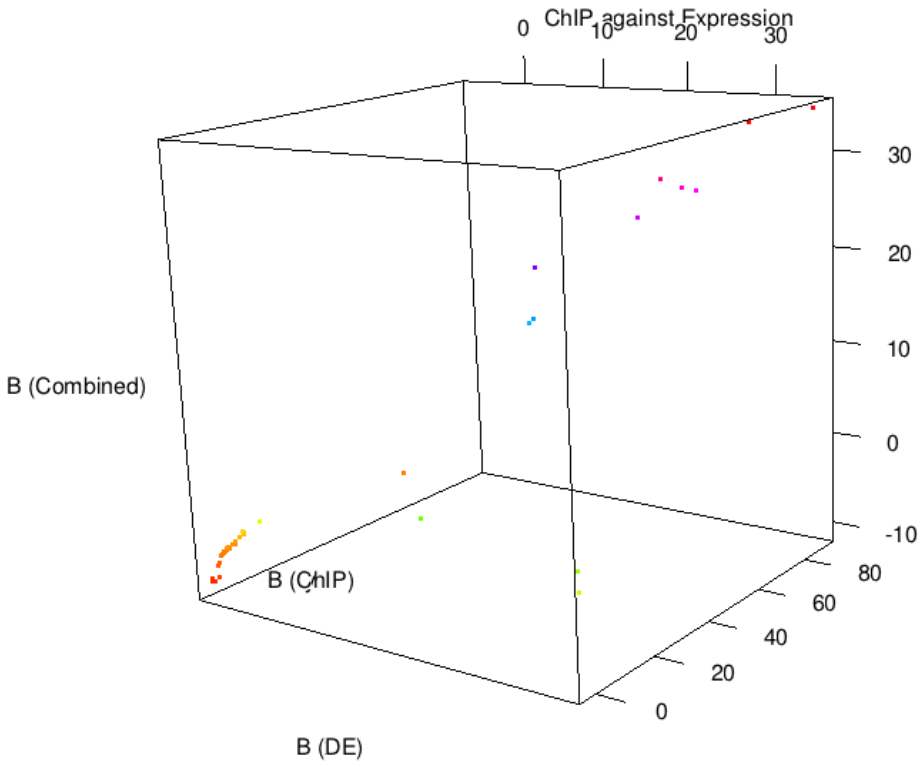

Figure 3: Output from `plotBBB(Rcade)`, subsequently saved to disk with the command `rgl.snapshot(filename="plotBBB.png")`