ChIPComp: A novel statistical method for quantitative comparison of multiple ChIP-seq datasets

Li Chen, Chi Wang, Zhaohui Qin, Hao Wu

April 24, 2017

Contents

1 Introduction

This vignette introduces the use of the Bioconductor package `ChIPComp`, which is designed for differential binding sites analyses based on high-throughput sequencing data. The core of `ChIPComp` is a new procedure to incorporate the control sequencing data in a linear model framework. `ChIPComp` focus on analyzing the DBS (transcription factor binding or histone modifications) generated by peak-calling software between two treatment conditions. Since an increasing number of ChIP experiments are investigating the same type of binding event (protein-DNA binding or histone modification) under different treatment conditions (cell lines), `ChIPComp` is to address how significant difference each binding site is between two treatment conditions by considering the control sequencing data. Compared with existing methods, ChIPComp provides excellent statistical and computational performance. Currently, `ChIPComp` only supports the situation when replicates are available for each treatment condition.

2 Overview

Here below is the `ChIPComp` work flow

1. Detect binding sites: The first step is to detect binding sites (transcription factor binding or histone modifications) for each ChIP sequencing data using existing peak-calling software.

2. Merge binding sites: Binding sites from all replicates in two treatment conditions are merged into one set of binding sites. In the process, common binding sites are also recorded.

3. Count reads: Both ChIP read counts and smoothing control read counts are calculated for each merged binding site.

4. Perform Hypothesis testing: We fit the model and perform hypothesis testing on each merged binding site.

3 Example

To utilize the ChIPComp software, we need a data frame that represents the ChIP experiment information. We also need a design matrix retrieved from ChIP experiment to fit the linear model. ChIPComp provides two ways to obtain the configuration data frame and the design matrix.
The first way is to enter ChIPComp experiment information into one csv file as an input for function makeConf. The configuration data frame and design matrix are the output of makeConf, for example,

```
> library(ChIPComp)
> confs=makeConf(system.file("extdata", "conf.csv", package="ChIPComp"))
> conf=confs$conf
> design=confs$design
```

Another way is to define a configuration data frame and design matrix manually, for example,

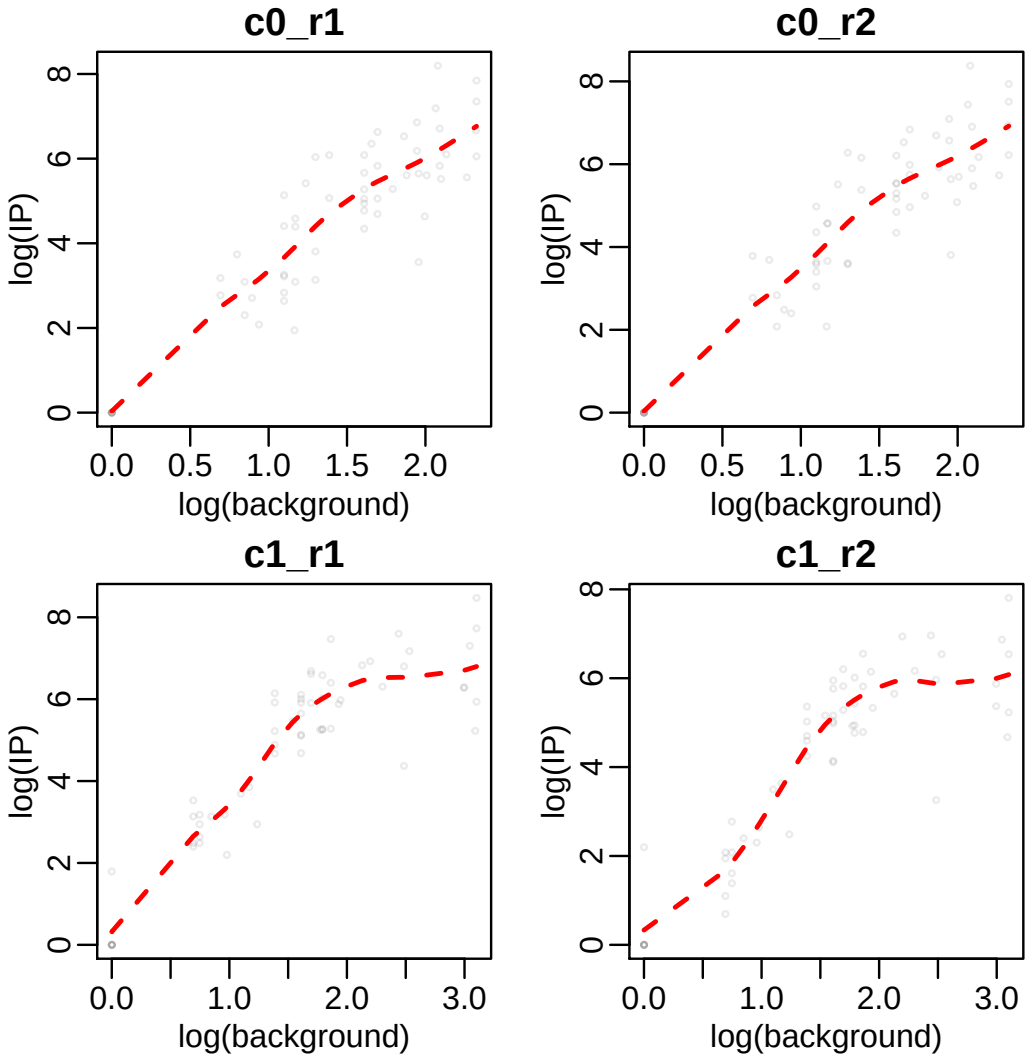
```
> conf=data.frame(
+ SampleID=1:4,
+ condition=c("Helas3","Helas3","K562","K562"),
+ factor=c("H3k27ac","H3k27ac","H3k27ac","H3k27ac"),
+ ipReads=system.file("extdata",c("Helas3.ip1.bed","Helas3.ip2.bed","K562.ip1.bed","K562.ip2.bed"),package=
+ ctReads=system.file("extdata",c("Helas3.ct.bed","Helas3.ct.bed","K562.ct.bed","K562.ct.bed"),package="Ch
+ peaks=system.file("extdata",c("Helas3.peak.bed","Helas3.peak.bed","K562.peak.bed","K562.peak.bed"),packa
+ )
> design=as.data.frame(lapply(conf[,c("condition","factor")],as.numeric))-1
> design=as.data.frame(model.matrix(~condition,design))
```

Once we have the configuration data frame and design matrix, we could merge binding sites, detect common binding sites and calculate read counts for each merged binding site.

```
> countSet=makeCountSet(conf,design,filetype="bed", species="hg19",binsize=1000)
```

Currently, if filetype is "bam", it is not necessary to specify species. However, if filetype is "bed", we need to specify species either "hg19" or "mm9". We could explore the correlation between ChIP sample and control sample.

```
> plot(countSet)
```



Finally, we perform hypothesis testing on each binding site and print the top differential binding sites.

```
> countSet=ChIPComp(countSet)
> print(countSet)
```

```
     chr     start       end ip_c0_r1 ip_c0_r2 ip_c1_r1 ip_c1_r2     ct_c0_r1
24 chr17  41462793  41468134     3626     4356       23        9 7.000000e+00
47 chr15  75309902  75327463      572      684      530      214 4.239901e+00
45  chr7 127281200 127558864      785     1030     4789     2459 9.204285e+00
8   chrX 153029492 153033027       44       36      746      336 2.666667e+00
2  chr19  16995963  17005821      283      280     1994     1054 6.076023e+00
53  chr9  34663975  34667049      418      531      235      173 2.666667e+00
14  chr1  36770095  36773714      194      250       18       11 4.000000e+00
28 chr14  55213729  55245520      249      237     2274      690 7.158971e+00
56  chr5 141389307 141389421        0        0        5        8 6.725728e-05
50  chr4 166244574 166254595      258      307      898      388 8.625731e+00
       ct_c0_r2     ct_c1_r1     ct_c1_r2 commonPeak  pvalue.wald prob.post
24 7.000000e+00 1.608187e+00 1.608187e+00          1 0.000000e+00 1.0000000
47 4.239901e+00 1.900000e+01 1.900000e+01          1 0.000000e+00 0.9999912
45 9.204285e+00 2.125000e+01 2.125000e+01          1 0.000000e+00 0.9999165
8  2.666667e+00 4.444444e+00 4.444444e+00          1 1.059262e-10 0.9928872
2  6.076023e+00 1.047368e+01 1.047368e+01          1 2.220446e-16 0.9912081
```

```
53 2.666667e+00 3.666667e+00 3.666667e+00          1 2.886580e-15 0.9857750
14 4.000000e+00 2.444444e+00 2.444444e+00          1 1.565828e-09 0.9772349
28 7.158971e+00 2.125000e+01 2.125000e+01          1 2.657446e-09 0.9600753
56 6.725728e-05 1.922092e-04 1.922092e-04          0 1.284754e-02 0.9359081
50 8.625731e+00 1.100000e+01 1.100000e+01          1 4.046903e-10 0.7351641
```

For the example data in the package, we collect 50 common binding sites between H3K27ac Helas3 and K562 cell lines and 10 unique binding sites for each cell line. Therefore, there are 60 binding sites for each cell line. We also extract the ChIP and control counts for each binding site in each condition. The configuration csv file, read bed files and peak bed files are stored in `inst/extdata` directory. The data frame that contains all binding sites and read counts have been pre-calculated and saved as a `ChIPComp` object `seqData` in `data` directory.

```
> data(seqData)
```

4 Session info

Here is the output of `sessionInfo` on the system on which this document was compiled:

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

- R version 3.4.0 (2017-04-21), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- 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`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: BiocGenerics 0.22.0, ChIPComp 1.6.0, GenomeInfoDb 1.12.0, GenomicRanges 1.28.0, IRanges 2.10.0, S4Vectors 0.14.0, rtracklayer 1.36.0
- Loaded via a namespace (and not attached): BSgenome 1.44.0, BSgenome.Hsapiens.UCSC.hg19 1.4.0, BSgenome.Mmusculus.UCSC.mm9 1.4.0, Biobase 2.36.0, BiocParallel 1.10.0, BiocStyle 2.4.0, Biostrings 2.44.0, DelayedArray 0.2.0, GenomeInfoDbData 0.99.0, GenomicAlignments 1.12.0, Matrix 1.2-9, RCurl 1.95-4.8, Rcpp 0.12.10, Rsamtools 1.28.0, SummarizedExperiment 1.6.0, XML 3.98-1.6, XVector 0.16.0, backports 1.0.5, bitops 1.0-6, compiler 3.4.0, digest 0.6.12, evaluate 0.10, grid 3.4.0, htmltools 0.3.5, knitr 1.15.1, lattice 0.20-35, limma 3.32.0, magrittr 1.5, matrixStats 0.52.2, rmarkdown 1.4, rprojroot 1.2, stringi 1.1.5, stringr 1.2.0, tools 3.4.0, yaml 2.1.14, zlibbioc 1.22.0