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

Li Chen, Chi Wang, Zhaohui Qin, Hao Wu

April 15, 2025

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
```

`ChIPComp` defines a configuration data frame and design matrix manually, for example,

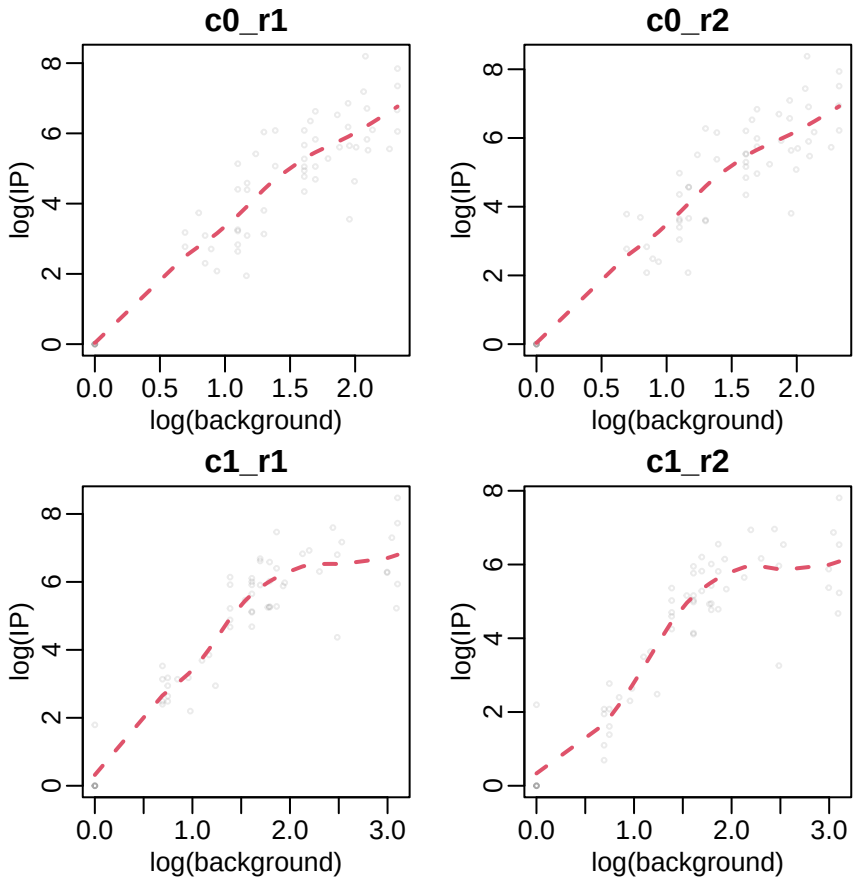
```
> conf=data.frame(
+ SampleID=1:4,
+ condition=c("Helas3","Helas3","K562","K562"),
+ factor=c("H3k27ac","H3k27ac","H3k27ac","H3k27ac"),
+ ipReads=system.file("extdata",c("Helas3.ip1.bed","Helas3.ip2.bed","K562.ip1.bed","K562.ip2.bed"),package="C
+ ctReads=system.file("extdata",c("Helas3.ct.bed","Helas3.ct.bed","K562.ct.bed","K562.ct.bed"),package="ChIPC
+ peaks=system.file("extdata",c("Helas3.peak.bed","Helas3.peak.bed","K562.peak.bed","K562.peak.bed"),package=
+ )
> conf$condition=factor(conf$condition)
> conf$factor=factor(conf$factor)
> 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 `file type` 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	4.239897e+00
45	chr7	127281200	127558864	785	1030	4789	2459	9.203767e+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	7.158938e+00
56	chr5	141389307	141389421	0	0	5	8	6.522411e-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	4.239897e+00	1.900000e+01	1.900000e+01	1	0.000000e+00	0.9999912
45	9.203767e+00	2.125000e+01	2.125000e+01	1	0.000000e+00	0.9999163
8	2.666667e+00	4.444444e+00	4.444444e+00	1	1.059270e-10	0.9928872
2	6.076023e+00	1.047368e+01	1.047368e+01	1	2.220446e-16	0.9912102
53	2.666667e+00	3.666667e+00	3.666667e+00	1	2.886580e-15	0.9857747
14	4.000000e+00	2.444444e+00	2.444444e+00	1	1.566118e-09	0.9772333

```
28 7.158938e+00 2.125000e+01 2.125000e+01          1 2.656539e-09 0.9600801
56 6.522411e-05 1.863988e-04 1.863988e-04          0 1.284723e-02 0.9359093
50 8.625731e+00 1.100000e+01 1.100000e+01          1 4.042460e-10 0.7352204
```

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 4.5.0 RC (2025-04-04 r88126), `x86_64-apple-darwin20`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Time zone: `America/New_York`
- TZcode source: `internal`
- Running under: `macOS Monterey 12.7.6`
- Matrix products: default
- BLAS: `/Library/Frameworks/R.framework/Versions/4.5-x86_64/Resources/lib/libRblas.0.dylib`
- LAPACK: `/Library/Frameworks/R.framework/Versions/4.5-x86_64/Resources/lib/libRlapack.dylib` ; LAPACK version3.12.1
- Base packages: base, datasets, grDevices, graphics, methods, stats, stats4, utils
- Other packages: BiocGenerics 0.54.0, ChIPComp 1.38.0, GenomeInfoDb 1.44.0, GenomicRanges 1.60.0, IRanges 2.42.0, S4Vectors 0.46.0, generics 0.1.3, rtracklayer 1.68.0
- Loaded via a namespace (and not attached): BSgenome 1.76.0, BSgenome.Hsapiens.UCSC.hg19 1.4.3, BSgenome.Mmusculus.UCSC.mm9 1.4.0, Biobase 2.68.0, BiocIO 1.18.0, BiocManager 1.30.25, BiocParallel 1.42.0, BiocStyle 2.36.0, Biostrings 2.76.0, DelayedArray 0.34.0, GenomeInfoDbData 1.2.14, GenomicAlignments 1.44.0, Matrix 1.7-3, MatrixGenerics 1.20.0, R6 2.6.1, RCurl 1.98-1.17, Rsamtools 2.24.0, S4Arrays 1.8.0, SparseArray 1.8.0, SummarizedExperiment 1.38.0, UCSC.utils 1.4.0, XML 3.99-0.18, XVector 0.48.0, abind 1.4-8, bitops 1.0-9, cli 3.6.4, codetools 0.2-20, compiler 4.5.0, crayon 1.5.3, curl 6.2.2, digest 0.6.37, evaluate 1.0.3, fastmap 1.2.0, grid 4.5.0, htmltools 0.5.8.1, httr 1.4.7, jsonlite 2.0.0, knitr 1.50, lattice 0.22-7, limma 3.64.0, matrixStats 1.5.0, parallel 4.5.0, restfulr 0.0.15, rjson 0.2.23, rlang 1.1.6, rmarkdown 2.29, statmod 1.5.0, tools 4.5.0, xfun 0.52, yaml 2.3.10

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