# The *ComplexDiff* user's guide

*Mingxiang Teng* mxteng@jimmy.harvard.edu
*Rafael A. Irizarry* rafa@jimmy.harvard.edu
*Department of Biostatistics, Dana-Farber Cancer Institute &*
*Harvard T.H. Chan School of Public Health, Boston, MA, USA*

*2016-12-14*

## Contents



## 1 Introduction

Identify protein binding differences using ChIP-Seq data involves comparing signal significances of protein binding across biological conditions. Two main strategies are widely accepted to accomplish this task. One strategy focuses on potential peak regions in compared samples, and applies peak calling followed by differential binding significance analysis. The other tests differential binding significance for genome-wide bins/windows followed by mergeing operation on significant windows to nominate report regions.

A particular type of ChIP-Seq data, mainly in protein complex studies, has a bimodel distribution of changes between compared conditions. For this data, exsiting methods need to be improved at least in two ways to fit the analysis: proper normalization and low coverage consideration. Here, we introduce *ComplexDiff* package to differential binding analysis in protein complex ChIP-Seq studies.

## 2 Getting Started

Load the package into R.

```
library(ComplexDiff)
```

# 3 Preparing Inputs

The input of this package includes a number of ChIP-Seq bam or bigwig files and corresponding meta information, *i.e.* sample conditions and batches. Exprienced users can also provide self-generated count matrix for genomic regions, and skip read counting using bam or bigwig files.

# 4 Differential Binding Regions

The differential binding analysis introduced in this package, mainly contains three steps: reads counting for genomic regions (*regionReads*), size factor estimation for normalization (*sizeFac*) and calling for differential binding regions (*diffRegions*). In addition, this package provides a separate function to help identify binding types (namely *unimodel* and *bimodel*) for a pair of compared ChIP-Seq samples. And another version of differential bidning calling is also provided with permutation analysis. For algorithms details, please refer to our manuscript (Teng 2016).

## 4.1 Count reads for genomic regions

We illustrate this step by counting reads on a pair of bam files built-in this package. The built-in bam files are generated from protein complex ChIP-Seq data stored in GEO databased with accession id GSM1645714 and GSM1645715. Only a small portion of reads on chromosome 1 are stored in these bam files. While chr1 is the only chromosome shown in the headers of these bam files, genomic regions are automatically generated only for chr1, followed by reads counting on these regions.

```
bams <- c(system.file("extdata", "control.bam", package="ComplexDiff"),
          system.file("extdata", "treated.bam", package="ComplexDiff"))
rc <- regionReads(bams)
```

```
names(rc)
## [1] "count"   "regions"
rc$regions
## GRanges object with 830836 ranges and 0 metadata columns:
##            seqnames                ranges strand
##               <Rle>             <IRanges>  <Rle>
##        [1]     chr1           [   1,  300]      *
##        [2]     chr1           [ 301,  600]      *
##        [3]     chr1           [ 601,  900]      *
##        [4]     chr1           [ 901, 1200]      *
##        [5]     chr1           [1201, 1500]      *
##        ...      ...                    ...    ...
##   [830832]     chr1 [249249301, 249249600]      *
##   [830833]     chr1 [249249601, 249249900]      *
##   [830834]     chr1 [249249901, 249250200]      *
##   [830835]     chr1 [249250201, 249250500]      *
##   [830836]     chr1 [249250501, 249250621]      *
##   -------
##   seqinfo: 1 sequence from an unspecified genome
```

To completely show the whole algorithms in this package, we further saved count matrix based on chr10 alignment reads of the same samples to perform downsteam analysis. First load this data by:

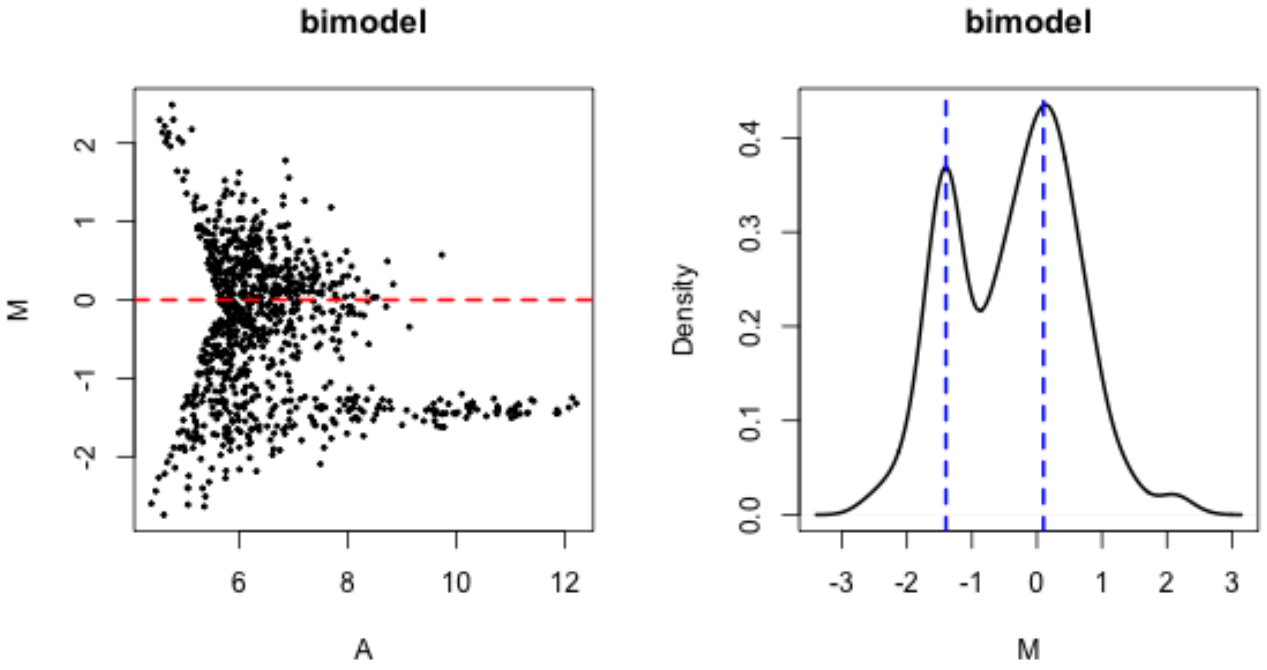
```
data(complex)
names(complex)
## [1] "counts" "bins"
```

```
complex$bins
## GRanges object with 451783 ranges and 0 metadata columns:
##            seqnames                ranges strand
##               <Rle>             <IRanges>  <Rle>
##        [1]    chr10           [   1,  300]      *
##        [2]    chr10           [ 301,  600]      *
##        [3]    chr10           [ 601,  900]      *
##        [4]    chr10           [ 901, 1200]      *
##        [5]    chr10           [1201, 1500]      *
##        ...      ...                    ...    ...
##   [451779]    chr10 [135533401, 135533700]      *
##   [451780]    chr10 [135533701, 135534000]      *
##   [451781]    chr10 [135534001, 135534300]      *
##   [451782]    chr10 [135534301, 135534600]      *
##   [451783]    chr10 [135534601, 135534747]      *
##   -------
##   seqinfo: 25 sequences from an unspecified genome
```

## 4.2 Determine protein binding type

Determine binding types of this protein complex ChIP-Seq data with following code. The default of this function generates two plots. For this data, a bimodel distribution of fold changes without normalization can be clearly visualized based on which binding type 'bimodel' will be concluded.

```
chipType(complex$counts)
## [1] "bimodel"
```



## 4.3 Estimate normalization size factors

In practice, users can skip binding type determination and directly estimate size factors once reads are counted from regions, as binding type determination is built into this step as well. As shown in the function help page, the size factors are calcualted pair-by-pair with all samples referring to the first sample in the count matrix.

Use following code to estimated size factors. Please also read the help page of this function to properly provide values for parameter *fold* and *h*. The returning values include two parts: size factors and binding types of samples by comparing to the first sample. When replicates are provided, replicates of the same condition should have the same binding type, either 'unimodel' or 'bimodel'. Inconsistent binding types of replicates also can happen for various experimental issues. Nevertheless, the strategy of kernal density bump hunting generates robust estimation of size factors regardless which binding type being concluded.

```
sizefac <- sizeFac(complex$counts)
sizefac
## $sizefac
## [1] 1.4504009 0.5495991
##
## $type
## [1] "control" "bimodel"
```

## 4.4 Identify differential binding regions

To call for differential binding regions, one can choose to use with or without permutation analysis, since permutation may take some time to accomplish. The algorithm detail of this step, please refer to function help page and our manuscript (Teng 2016).

Use following code for identification without permutation. Here, we use simple labels (*ctr* and *tre*) to represent experimental information of two samples.

```
meta <- data.frame(cond=c("ctr","tre"))
dr <- diffRegions(complex$counts,complex$bins,meta,design=~cond,
                  sizefac$sizefac)
dr
## GRanges object with 3769 ranges and 3 metadata columns:
##          seqnames                ranges strand |               stat
##             <Rle>             <IRanges>  <Rle> |          <numeric>
##      [1]    chr10 [ 63785101,  63786900]      * |  0.917431913225877
##      [2]    chr10 [116096401, 116098200]      * |  0.759628991925164
##      [3]    chr10 [ 59383501,  59385000]      * |  0.798277044014643
##      [4]    chr10 [ 52687201,  52688400]      * |  0.782135996148531
##      [5]    chr10 [ 98261701,  98262900]      * |  0.559358434988937
##      ...      ...                   ...    ... .                ...
##   [3765]    chr10 [ 39120301,  39120600]      * | -0.196059515158732
##   [3766]    chr10 [ 62366401,  62366700]      * |  0.192539317564189
##   [3767]    chr10 [131822701, 131823000]      * | -0.191914245318879
##   [3768]    chr10 [ 39121201,  39121500]      * | -0.191012873989451
##   [3769]    chr10 [ 84488401,  84488700]      * |   0.18743248541297
##                      log2fc            pvalue
##                   <numeric>         <numeric>
##      [1]   2.00171621009385  0.35891635613454
##      [2]   1.64114383185474 0.447476384290381
##      [3]   1.72678912557971 0.424709736555485
##      [4]   1.69098409537382 0.434134654077988
##      [5]   1.19543762060796 0.575917122378524
```

```
##      ...                 ...                 ...
##   [3765] -0.414357993665161 0.844563583453475
##   [3766]  0.410961755774115 0.847319773041433
##   [3767] -0.408133041890141 0.847809378705296
##   [3768] -0.403728144738157 0.848515506879191
##   [3769]  0.397975655016623 0.851321554230418
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

Alternatively, permutation can be added into analysis using following code. For this illustration examples, it will give the same results due to no permutation will be performed without replciates.

```
meta <- data.frame(cond=c("ctr","tre"))
dr <- diffRegionsWithPerm(complex$counts,complex$bins,meta,design=~cond,
                 sizefac$sizefac)
## No permutation performed due to not enough samples!
## Differential analysis of real data......done!
dr
## GRanges object with 3769 ranges and 3 metadata columns:
##          seqnames                ranges strand |               stat
##             <Rle>             <IRanges>  <Rle> |          <numeric>
##      [1]    chr10 [ 63785101,  63786900]      * |  0.917431913225877
##      [2]    chr10 [116096401, 116098200]      * |  0.759628991925164
##      [3]    chr10 [ 59383501,  59385000]      * |  0.798277044014643
##      [4]    chr10 [ 52687201,  52688400]      * |  0.782135996148531
##      [5]    chr10 [ 98261701,  98262900]      * |  0.559358434988937
##      ...      ...                   ...    ... .                 ...
##   [3765]    chr10 [ 39120301,  39120600]      * | -0.196059515158732
##   [3766]    chr10 [ 62366401,  62366700]      * |  0.192539317564189
##   [3767]    chr10 [131822701, 131823000]      * | -0.191914245318879
##   [3768]    chr10 [ 39121201,  39121500]      * | -0.191012873989451
##   [3769]    chr10 [ 84488401,  84488700]      * |   0.18743248541297
##                      log2fc            pvalue
##                   <numeric>         <numeric>
##      [1]   2.00171621009385  0.35891635613454
##      [2]   1.64114383185474 0.447476384290381
##      [3]   1.72678912557971 0.424709736555485
##      [4]   1.69098409537382 0.434134654077988
##      [5]   1.19543762060796 0.575917122378524
##      ...                 ...               ...
##   [3765] -0.414357993665161 0.844563583453475
##   [3766]  0.410961755774115 0.847319773041433
##   [3767] -0.408133041890141 0.847809378705296
##   [3768] -0.403728144738157 0.848515506879191
##   [3769]  0.397975655016623 0.851321554230418
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

## 5 Summary

In this guide, we illustrated the useage of functions provided in this package. They are particularly designed for ChIP-Seq experimental cases where changes between conditions have a bimodel distribution, as we shown for protein complex data.

It is also applicable for normal ChIP-Seq experiments.

## References

Teng, M et al. 2016. "ComplexDiff: Differential Binding Estimation for Protein Complexes." *In Preparation*.