Exploring Epigenomic Datasets by ChIPseeker

Qianwen Wang,[1,2] Ming Li,[1,2] Tianzhi Wu,[1] Li Zhan,[1] Lin Li,[1] Meijun Chen,[1] Wenqin Xie,[1] Zijing Xie,[1] Erqiang Hu,[1] Shuangbin Xu,[1] and Guangchuang Yu[1,3]

[1]Department of Bioinformatics, School of Basic Medical Sciences, Southern Medical University, Guangzhou, China
[2]These authors contributed equally to this work
[3]Corresponding author: *gcyu1@smu.edu.cn*

Published in the Bioinformatics section

In many aspects of life, epigenetics, or the altering of phenotype without changes in sequences, play an essential role in biological function. A vast number of epigenomic datasets are emerging as a result of the advent of next-generation sequencing. Annotation, comparison, visualization, and interpretation of epigenomic datasets remain key aspects of computational biology. ChIPseeker is a Bioconductor package for performing these analyses among variable epigenomic datasets. The fundamental functions of ChIPseeker, including data preparation, annotation, comparison, and visualization, are explained in this article. ChIPseeker is a freely available open-source package that may be found at *https://www.bioconductor.org/packages/ChIPseeker*.

Keywords: annotation • ChIPseeker • comparison • epigenetic • visualization

How to cite this article:
Wang, Q., Li, M., Wu, T., Zhan, L., Li, L., Chen, M., Xie, W., Xie, Z., Hu, E., Xu, S., & Yu, G. (2022). Exploring epigenomic datasets by ChIPseeker. *Current Protocols*, *2,* e585. doi: 10.1002/cpz1.585

INTRODUCTION

Epigenetics, which refers to the alteration of phenotype without changes in sequences, play essential biological functions in many aspects of life (Allis & Jenuwein, 2016). With the rapidly increasing diversity of epigenetic experimental assays using next-generation sequencing (NGS) technologies, these assays have become widely used to address biological questions (Klemm, Shipony, & Greenleaf, 2019; Zhao, Song, Liu, Song, & Yi, 2020). As a result, many laboratories have adopted the technology and are now facing the massive issue of manipulating NGS epigenomic datasets. To add to the burden, researchers are frequently asked to compare their innovative experimental results to publicly available sequencing data. In our experience, the lack of user-friendly software, as

CURRENT PROTOCOLS
A Wiley Brand
Current Protocols e585, Volume 2
Published in Wiley Online Library (wileyonlinelibrary.com).
doi: 10.1002/cpz1.585

well as thorough documentation and explanation of the various procedures, deter biologists from participating in the analysis of public or private epigenomic datasets. To solve these issues, we created ChIPseeker, a powerful R program that researchers can use to perform downstream analyses and visualization of epigenomic datasets (Yu, Wang, & He, 2015).

This article describes detailed protocols for using ChIPseeker in various tasks, including (1) ChIPseeker and epigenomic datasets preparation; (2) annotation of epigenomic datasets; (3) comparison of epigenomic datasets; (4) visualization of annotated results; (5) functional analysis of epigenomic datasets; (6) genome-wide and locus-specific distribution of epigenomic datasets; (7) heatmaps and metaplots of epigenomic datasets. A GitHub repository containing all of the documented code used in this article and expanded R Markdown scripts can also be found at *https://github.com/YuLab-SMU/ChIPseeker_current_protocols*.

STRATEGIC PLANNING

The following protocols are all run in R/Rstudio. To improve readability, names of commands, functions, or parameters are formatted in a distinct font, e.g., `exampleCode`. The following protocols are all run in R/Rstudio. In this manuscript, we are presenting the functions of ChIPseeker in version 1.32.1. To improve the readability, we attached the simplified storyline as our manuscript in the supplementary file.

BASIC PROTOCOL 1

ChIPseeker AND EPIGENOMIC DATASET PREPARATION

In this protocol, we will go through the system and environment requirements for installing ChIPseeker, as well as how to load the epigenomic datasets into ChIPseeker.

Briefly, ChIPseeker is an R package that can be downloaded and deployed on the most commonly used operating systems, including Linux, macOS, and Windows. The core function of this protocol is to save any kind of epigenomic dataset in Browser Extensible Data (BED)/bedGraph format (Kent et al., 2002) to a GRanges object (Lawrence et al., 2013). Users must provide the following information in the BED/bedGraph files: chr, chromStart, chromEnd, and the value that is associated with all the bases between the chromStart and chromEnd positions. Since the bedGraph format is a variant of BED, we will use BED format to perform the demonstration in this manuscript.

Necessary Resources

Hardware

A modern computing environment capable of running R/Rstudio $>$ 3.5.0

Software

R/Rstudio and the R package ChIPseeker. The latest ChIPseeker stable release version for Windows, Mac OS X, and Unix/Linux is available from Bioconductor (*https://www.bioconductor.org/packages/ChIPseeker*). The latest development branch is available from GitHub (*https://github.com/YuLab-SMU/ChIPseeker*).

Files

All types of epigenomic datasets in BED/bedGraph format. The details of BED/bedGraph format can be found on the UCSC website (*https://genome.ucsc.edu/FAQ/FAQformat.html*). The example datasets are available at *https://github.com/YuLab-SMU/ChIPseeker_current_protocols*.

1. Installation of ChIPseeker package.

   R can be downloaded and installed from *https://cran.r-project.org*. Alternatively, users could download and install Rstudio from *https://www.rstudio.com*. Then, open

R/Rstudio and install BiocManager. BiocManager is an R package that allows you to install and update Bioconductor packages.

```
install.package("BiocManager")
```

Using BiocManager to install ChIPseeker:

```
BiocManager::install("ChIPseeker")
```

Loading the ChIPseeker library:

```
library("ChIPseeker")
```

2. Loading epigenomic datasets.
All kinds of epigenomics datasets in BED/bedGraph format could be loaded locally. As an example, we have used re-analyzed CTCF ChIP-seq results in BED format, which were deposited in the NCBI database under accession number GSE52457 (Dixon et al., 2015). The data can be downloaded in R with the following command:

```
download.file("https://raw.githubusercontent.com/YuLab-
   SMU/ChIPseeker_current_protocols/master/CP_demo_data/CTCF_H1.test.bed",
        destfile = "CTCF_H1.test.bed")
```

Generally, the header information is absent in BED format, so we could load the normal BED file by `readPeakFile()`.

```
ChIPseq_CTCF_demo = readPeakFile("CTCF_H1.test.bed",
                                 header = FALSE)
```

Then we could rename the columns to their corresponding names after loading the BED file. You will get the results as shown in Figure 1A.

```
ChIPseq_CTCF_demo$CTCF_peaks = ChIPseq_CTCF_demo$V4
ChIPseq_CTCF_demo$level = ChIPseq_CTCF_demo$V5
ChIPseq_CTCF_demo$V4 = ChIPseq_CTCF_demo$V5 = NULL
ChIPseq_CTCF_demo
```

We could also add the header information into BED files and load them with header parameter. Here we have prepared the demo files of ChIP-seq (H3K4me1 and H3K4me3), Methyl-seq, DNase-seq, and smRNA datasets that were deposited under the accession number GSE16256 (Lister et al., 2009). These files could be downloaded by the following commands:

```
download.file("https://raw.githubusercontent.com/YuLab-
SMU/ChIPseeker_current_protocols/master/CP_demo_data/H3K4me1_H1.test.bed",
              destfile        =                "H3K4me1_H1.test.bed")
download.file("https://raw.githubusercontent.com/YuLab-
SMU/ChIPseeker_current_protocols/master/CP_demo_data/H3K4me3_H1.test.bed",
              destfile        =                "H3K4me3_H1.test.bed")
download.file("https://raw.githubusercontent.com/YuLab-
SMU/ChIPseeker_current_protocols/master/CP_demo_data/DHSs_H1.test.bed",
              destfile        =                "DHSs_H1.test.bed")
download.file("https://raw.githubusercontent.com/YuLab-
SMU/ChIPseeker_current_protocols/master/CP_demo_data/DNAmeth_H1.test.bed",
              destfile        =                "DNAmeth_H1.test.bed")
download.file("https://raw.githubusercontent.com/YuLab-
SMU/ChIPseeker_current_protocols/master/CP_demo_data/smRNA_H1.test.bed",
              destfile = "smRNA_H1.test.bed")
```

And then we could read these files with the `readPeakFile()` function with the parameter of `header=TRUE`.

A

```
## GRanges object with 1000 ranges and 2 metadata columns:
##          seqnames              ranges strand |       CTCF_peaks       level
##             <Rle>           <IRanges>  <Rle> |         <factor>   <numeric>
##      [1]     chr5     2156742-2156776      * | CTCF_merge_71009          70
##      [2]    chr12 128241396-128241477      * | CTCF_merge_22514 69.08333333
##      [3]     chr7   29811228-29811262      * | CTCF_merge_84581          70
##      [4]    chr12   24075670-24075701      * | CTCF_merge_18909          68
##      [5]    chr18   23857660-23857693      * | CTCF_merge_40461          70
##      ...      ...                 ...    ... .              ...         ...
##    [996]    chr17     5746531-5746563      * | CTCF_merge_36592          70
##    [997]     chr2       608916-608949      * | CTCF_merge_44717          69
##    [998]     chr4 116086644-116086677      * | CTCF_merge_68591          70
##    [999]     chr4 187291000-187291033      * | CTCF_merge_70585          70
##   [1000]    chr11   61262245-61262298      * | CTCF_merge_15171          70
##   -------
##   seqinfo: 22 sequences from an unspecified genome; no seqlengths
```

B

```
## GRanges object with 1000 ranges and 2 metadata columns:
##          seqnames    ranges strand |             m5C     level
##             <Rle> <IRanges>  <Rle> |        <factor> <integer>
##      [1]    chr17  76995109      * | dnameth_1158899         1
##      [2]     chr2  41333577      * | dnameth_1385974         6
##      [3]    chr19  33169954      * | dnameth_1285728         7
##      [4]    chr17  58796413      * | dnameth_1127090         5
##      [5]     chr4   7515464      * | dnameth_1941102         1
##      ...      ...       ...    ... .              ...       ...
##    [996]     chr5  11165409      * | dnameth_2120600         5
##    [997]    chr10  69610558      * |  dnameth_294571         9
##    [998]     chr2 206263019      * | dnameth_1511933         1
##    [999]     chr5  96441230      * | dnameth_2185369         4
##   [1000]     chr4  69863434      * | dnameth_1996883         6
##   -------
##   seqinfo: 22 sequences from an unspecified genome; no seqlengths
```

C

```
##                organism genomeVersion Freq
## 1    Anolis carolinensis       anoCar2    7
## 2             Bos taurus       bosTau4    2
## 3             Bos taurus       bosTau6   33
## 4             Bos taurus       bosTau7    2
## 5 Canis lupus familiaris       canFam3   10
```

D

```
## Annotated peaks generated by ChIPseeker
## 1000/1000  peaks were annotated
## Genomic Annotation Summary:
##              Feature Frequency
## 9    Promoter (<=1kb)       2.3
## 10   Promoter (1-2kb)       2.5
## 11   Promoter (2-3kb)       2.3
## 4             5' UTR       0.1
## 3             3' UTR       0.9
## 1           1st Exon       0.2
## 7          Other Exon       1.5
## 2           1st Intron      11.8
## 8        Other Intron      27.1
## 6  Downstream (<=300)       0.1
## 5   Distal Intergenic      51.2
```

Figure 1 Screenshot of GRange object of ChIPseq_CTCF_demo (**A**) and Methylseq_demo (**B**). (**C**) Five of the species in pre-prepared NCBI database in ChIPseeker. (**D**) Screenshot of annotated results of ChIPseq_CTCF_demo_anno_default.

```
ChIPseq_H3K4me1_demo1 = readPeakFile("H3K4me1_H1.test.bed",
                        header=TRUE)
ChIPseq_H3K4me3_demo2 = readPeakFile("H3K4me3_H1.test.bed",
                        header=TRUE)
DNaseseq_demo = readPeakFile("DHSs_H1.test.bed",
                header=TRUE)
Methylseq_demo = readPeakFile("DNAmeth_H1.test.bed",
                 header=TRUE)
```

A

```
##       series_id       gsm           organism                   title
## 11764  GSE43512 GSM1064688 Anolis carolinensis    Lizard testes Bio-CAP
## 11780  GSE43512 GSM1064689 Anolis carolinensis     Lizard liver Bio-CAP
## 19867  GSE64055 GSM1833394 Anolis carolinensis    Acar_Forelimb_H3K27ac
## 19869  GSE64055 GSM1833396 Anolis carolinensis    Acar_Hindlimb_H3K27ac
## 19871  GSE64055 GSM1833398 Anolis carolinensis Acar_Hemiphallus_H3K27ac
## 70432  GSE97451 GSM2564799 Anolis carolinensis              NSR_Testis
## 70433  GSE97451 GSM2564798 Anolis carolinensis               NSR_Brain
##                                                                                                  supplementary_file
## 11764              ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM1064nnn/GSM1064688/suppl/GSM1064688_ac_testes_nmi.bed.gz
## 11780               ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM1064nnn/GSM1064689/suppl/GSM1064689_ac_liver_nmi.bed.gz
## 19867    ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM1833nnn/GSM1833394/suppl/GSM1833394_fl_h3k27ac_ts7_8_anoCar2.bed.gz
## 19869    ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM1833nnn/GSM1833396/suppl/GSM1833396_hl_h3k27ac_ts7_8_anoCar2.bed.gz
## 19871   ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM1833nnn/GSM1833398/suppl/GSM1833398_hp_h3k27ac_ts9_11_anoCar2.bed.gz
## 70432  ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM2564nnn/GSM2564799/suppl/GSM2564799_brain_all_smooth.ucsc.bedGraph.gz
## 70433 ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM2564nnn/GSM2564798/suppl/GSM2564798_testis_all_smooth.ucsc.bedGraph.gz
##       genomeVersion pubmed_id
## 11764       anoCar2  23467541
## 11780       anoCar2  23467541
## 19867       anoCar2      <NA>
## 19869       anoCar2      <NA>
## 19871       anoCar2      <NA>
## 70432       anoCar2  28538188
## 70433       anoCar2  28538188
```

B

```
## GRanges object with 5 ranges and 11 metadata columns:
##       seqnames              ranges strand |       CTCF_peaks       level
##          <Rle>           <IRanges>  <Rle> |         <factor>   <numeric>
##   [1]     chr5     2156742-2156776      * | CTCF_merge_71009          70
##   [2]    chr12 128241396-128241477      * | CTCF_merge_22514 69.08333333
##   [3]     chr7   29811228-29811262      * | CTCF_merge_84581          70
##   [4]    chr12   24075670-24075701      * | CTCF_merge_18909          68
##   [5]    chr18   23857660-23857693      * | CTCF_merge_40461          70
##                                   annotation   geneChr geneStart   geneEnd
##                                  <character> <integer> <integer> <integer>
##   [1]                       Distal Intergenic         5   1877541   1887293
##   [2]                       Distal Intergenic        12 128366162 128383184
##   [3]                       Distal Intergenic         7  29846170  29956682
##   [4] Intron (uc001rfw.3/6660, intron 1 of 14)        12  23685231  24102637
##   [5] Intron (uc002kvs.4/6875, intron 5 of 15)        18  23806409  23971650
##       geneLength geneStrand      geneId transcriptId distanceToTSS
##        <integer>  <integer> <character>  <character>     <numeric>
##   [1]       9753          2       50805   uc031sir.1       -269449
##   [2]      17023          1      400087   uc001uhp.2       -124685
##   [3]     110513          1      644150   uc022aaz.1        -34908
##   [4]     417407          2        6660   uc001rfw.3         26936
##   [5]     165242          1        6875   uc002kvu.4         51251
##   -------
##   seqinfo: 22 sequences from hg19 genome
```

C

```
## Annotated peaks generated by ChIPseeker
## 1000/1000  peaks were annotated
## Genomic Annotation Summary:
##      Feature Frequency
## 6   Promoter       7.1
## 2     5' UTR       0.1
## 1     3' UTR       0.9
## 3       Exon       1.7
## 5     Intron      38.9
## 4 Intergenic      51.3
```

D

```
## Annotated peaks generated by ChIPseeker
## 1000/1000  peaks were annotated
## Genomic Annotation Summary:
##      Feature Frequency
## 6   Promoter       2.5
## 2     5' UTR       0.4
## 1     3' UTR       1.0
## 3       Exon       2.0
## 5     Intron      41.9
## 4 Intergenic      52.2
```

Figure 2 (**A**) Detail information of *Anolis carolinensis* in pre-prepared NCBI database in ChIPseeker. (**B**) GRange object of ChIPseq_CTCF_demo_anno_default. Screenshot of annotated results in ChIPseq_CTCF_demo_anno_with_options (**C**) and ChIPseq_CTCF_demo_anno_user_defined (**D**).

```
smRNA_demo     =     readPeakFile("smRNA_H1.test.bed",
                header=TRUE)
Methylseq_demo
```

As is shown (Fig. 1B), ChIPseeker can handle single-base resolution as well as regions of functional epigenomic datasets.

3. Downloading the files from the pre-prepared NCBI database.

ChIPseeker also collects around 17,000 epigenomic results that are deposited in GEO/GSM. By running the `getGEOgenomeVersion()` function, users can get a quick summary of all the collected species in alphabetical order. Here we show five of the species as an example (Fig. 1C):

```
head(getGEOgenomeVersion(), 5)
```

Then, we might obtain the specific information based on the value in the column of genomeVersion; here we use *Anolis carolinensis* as an example (Fig. 2A).

```
getGEOInfo(genome = "anoCar2")
```

The `downloadGEObedFiles()` and `downloadGSMbedFiles()` functions can download the files that have been previously collected within the ChIPseeker. Here we use *Anolis carolinensis* as an example:

```
downloadGEObedFiles(genome = "anoCar2",
                    destDir = "/path/to/download/location")
```

Besides downloading all the files of *Anolis carolinensis*, we could also specify the files that we need by the GSM numbers. Here, we use GSM1064688 and GSM1064689 as examples.

```
gsm = ("GSM1064688","GSM1064689")
downloadGSMbedFiles(gsm,
                    destDir = "/path/to/download/location")
```

After downloading the BED/bedGraph files, we could load them using the method described in step 2 (Loading epigenomic datasets in BED/bedGraph format).

BASIC PROTOCOL 2

ANNOTATION OF EPIGENOMIC DATASETS

As epigenetics refers to the regulation of gene expression other than direct changes in sequence, it is critical to annotate epigenomic datasets to genes. In addition, there are an increasing number of studies that report the coordination of multiple epigenetic marks. It is also imperative to analyze the annotation of different epigenetic marks.

In this protocol, we will talk about how to annotate epigenomic datasets using the TxDb database and user-defined regions.

Necessary Resources

Hardware

See Basic Protocol 1

Software

See Basic Protocol 1

Files

The files that were loaded in Basic Protocol 1. If users wish to do the annotation based on the known genes, they will need the TxDb object of the same reference that was used to generate the BED/bedGraph results previously. Please check the existing TxDb object at *http://bioconductor.org/packages/release/BiocViews.html#_TxDb*. If there is no pre-made TxDb database for the genome users want, users can learn how to create their own TxDb database at: *https://bioconductor.org/packages/release/bioc/vignettes/GenomicFeatures/inst/doc/GenomicFeatures.html*

1. Annotation of epigenomic datasets using the TxDb database.

Using the files from Basic Protocol 1 that were analyzed using the hg19 reference genome as an example, install the `TxDb.Hsapiens.UCSC.hg19.knownGene` package using BiocManager.

```
BiocManager::install("TxDb.Hsapiens.UCSC.hg19.knownGene")
```

2. Load the TxDb object from `TxDb.Hsapiens.UCSC.hg19.knownGene` package.

```
library("TxDb.Hsapiens.UCSC.hg19.knownGene")
TxDb_hg19 = TxDb.Hsapiens.UCSC.hg19.knownGene
```

For the genomic annotation, the genomic sites were defined as follows: Promoter; 5' UTR (five prime untranslated regions); 3' UTR (three prime untranslated regions); Exon; Intron; Downstream; and Distal Intergenic. Each functional region was assigned to only one of the categories indicated above, in the order listed. Here, we give examples by doing the annotation of the functional epigenomic datasets that are loaded in Basic Protocol 1.

```
ChIPseq_CTCF_demo_anno_default = annotatePeak(ChIPseq_CTCF_demo,
                                  TxDb=TxDb_hg19)
```

After annotation, we could check the brief summary of the annotated results (Fig. 1D).

```
ChIPseq_CTCF_demo_anno_default
```

3. Check the detailed information of the annotated results. Besides the genomic annotation, ChIPseeker calculated the distance to the nearest transcript. The results of the distance were listed in the columns of transcriptId and distanceToTSS (Fig. 2B).

```
head(as.GRanges(ChIPseq_CTCF_demo_anno_default), 5)
```

4. Save the annotated results by using the `as.data.frame()` and `write.table()` functions.

```
write.table(as.data.frame(ChIPseq_CTCF_demo_anno_default),
            file = "/path/to/download/location")
```

5. If users have their own preference for the annotation priority, the parameter of `genomicAnnotationPriority` can be used to modify it.

```
ChIPseq_CTCF_demo_anno_change_priority = annotatePeak(ChIPseq_CTCF_demo,
        TxDb=TxDb_hg19,
        genomicAnnotationPriority = c("Exon", "Intron", "5UTR", "3UTR",
  "Promoter", "Downstream", "Intergenic"))
```

6. To configure the genomics regions that users require annotated, ChIPseeker includes the function `options()`.

```
options(ChIPseeker.ignore_1st_exon = TRUE)
options(ChIPseeker.ignore_1st_intron = TRUE)
options(ChIPseeker.ignore_downstream = TRUE)
options(ChIPseeker.ignore_promoter_subcategory = TRUE)
ChIPseq_CTCF_demo_anno_with_options = annotatePeak(ChIPseq_CTCF_demo,
      TxDb=TxDb_hg19)
```

We could find our annotated results changed after setting the `options()` functions (Fig. 2C).

```
ChIPseq_CTCF_demo_anno_with_options
```

7. By default, the promoter is defined as 3000 bases upstream and downstream of the transcription start site (TSS). Depending on the research needs, users may modify this region by the parameter of `tssRegion` (Fig. 2D).

```
ChIPseq_CTCF_demo_anno_user_defined  =  annotatePeak(ChIPseq_CTCF_demo,
                                            tssRegion = c(-2000,0),
                                            TxDb=TxDb_hg19)
ChIPseq_CTCF_demo_anno_user_defined
```

8. ChIPseeker provides an optional parameter for the organism-level package OrgDb. This will output the relevant gene ID into the columns of ENSEMBL/ENTREZID, SYMBOL, and GENENAME. As we are analyzing the data from human, the org.Hs.eg.db annotation package was used for demonstration.

```
ChIPseq_CTCF_demo_anno_gene_name  =  annotatePeak(ChIPseq_CTCF_demo,
                                        tssRegion=c(-2000,0),
                                        TxDb=TxDb_hg19,
                                        annoDb="org.Hs.eg.db")
```

   By checking the detailed information, we can find the new columns of ENSEMBL/ENTREZID, SYMBOL, and GENENAME as we mentioned above (Fig. 3A).

```
head(as.GRanges(ChIPseq_CTCF_demo_anno_gene_name), 5)
```

9. Identify the genes surrounding the functional epigenetic regions. In the example, genes from the 5000 bases upstream and downstream of CTCF ChIP-seq peaks are annotated.

```
ChIPseq_CTCF_demo_anno_flank_5_kb  =  annotatePeak(ChIPseq_CTCF_demo,
                                        tssRegion=c(-2000,0),
                                        TxDb=TxDb_hg19,
                                        addFlankGeneInfo=TRUE,
                                        flankDistance=5000)
```

   By checking the detailed information, we can find the new columns of flank_txIds, flank_geneIds, and flank_gene_distances (Fig. 3B).

```
head(as.GRanges(ChIPseq_CTCF_demo_anno_flank_5_kb), 5)
```

10. Construct the GRangeList object based on previously loaded files using the `annotatePeak()` function for a GRangeList object.

```
Epi_data_list        =          GRangesList(CTCF=ChIPseq_CTCF_demo,
                          DHSs=DNaseseq_demo,
                          H3K4me1=ChIPseq_H3K4me1_demo1,
                          m5C=Methylseq_demo,
                          smRNA=smRNA_demo)
peakAnnoList_user_defined    =    lapply(Epi_data_list,
                          annotatePeak,
                          tssRegion=c(-2000,0),
                          TxDb=TxDb_hg19)
```

11. Annotate epigenomic datasets using user-defined regions.

   Any user-defined region in the GRanges object could be used for annotation. The GenomicRanges package has a detailed introduction to the GRanges object (Lawrence et al., 2013). For example:

```
user_defined_GRange = GRanges(seqnames = Rle(c("chr1", "chr10",
                                                "chr1", "chr20"),
                                                  c(1, 3, 1, 5)),
```

A

```
## GRanges object with 5 ranges and 14 metadata columns:
##       seqnames              ranges strand |       CTCF_peaks       level
##          <Rle>           <IRanges>  <Rle> |         <factor>   <numeric>
##   [1]     chr5     2156742-2156776      * | CTCF_merge_71009          70
##   [2]    chr12 128241396-128241477      * | CTCF_merge_22514 69.08333333
##   [3]     chr7   29811228-29811262      * | CTCF_merge_84581          70
##   [4]    chr12   24075670-24075701      * | CTCF_merge_18909          68
##   [5]    chr18   23857660-23857693      * | CTCF_merge_40461          70
##                                     annotation   geneChr geneStart   geneEnd
##                                    <character> <integer> <integer> <integer>
##   [1]                       Distal Intergenic         5   1877541   1887293
##   [2]                       Distal Intergenic        12 128366162 128383184
##   [3]                       Distal Intergenic         7  29846170  29956682
##   [4] Intron (uc001rfw.3/6660, intron 1 of 14)        12  23685231  24102637
##   [5] Intron (uc002kvs.4/6875, intron 5 of 15)        18  23806409  23971650
##       geneLength geneStrand      geneId transcriptId distanceToTSS
##        <integer>  <integer> <character>  <character>     <numeric>
##   [1]       9753          2       50805   uc031sir.1       -269449
##   [2]      17023          1      400087   uc001uhp.2       -124685
##   [3]     110513          1      644150   uc022aaz.1        -34908
##   [4]     417407          2        6660   uc001rfw.3         26936
##   [5]     165242          1        6875   uc002kvu.4         51251
##                ENSEMBL      SYMBOL                                      GENENAME
##            <character> <character>                                   <character>
##   [1] ENSG00000113430        IRX4                                iroquois homeobox 4
##   [2] ENSG00000256597   LINC02393   long intergenic non-protein coding RNA 2393
##   [3] ENSG00000122574       WIPF3  WAS/WASL interacting protein family member 3
##   [4] ENSG00000134532        SOX5               SRY-box transcription factor 5
##   [5] ENSG00000141384       TAF4B TATA-box binding protein associated factor 4b
##   -------
##   seqinfo: 22 sequences from hg19 genome
```

B

```
## GRanges object with 5 ranges and 14 metadata columns:
##       seqnames              ranges strand |       CTCF_peaks       level
##          <Rle>           <IRanges>  <Rle> |         <factor>   <numeric>
##   [1]     chr5     2156742-2156776      * | CTCF_merge_71009          70
##   [2]    chr12 128241396-128241477      * | CTCF_merge_22514 69.08333333
##   [3]     chr7   29811228-29811262      * | CTCF_merge_84581          70
##   [4]    chr12   24075670-24075701      * | CTCF_merge_18909          68
##   [5]    chr18   23857660-23857693      * | CTCF_merge_40461          70
##                                     annotation   geneChr geneStart   geneEnd
##                                    <character> <integer> <integer> <integer>
##   [1]                       Distal Intergenic         5   1877541   1887293
##   [2]                       Distal Intergenic        12 128366162 128383184
##   [3]                       Distal Intergenic         7  29846170  29956682
##   [4] Intron (uc001rfw.3/6660, intron 1 of 14)        12  23685231  24102637
##   [5] Intron (uc002kvs.4/6875, intron 5 of 15)        18  23806409  23971650
##       geneLength geneStrand      geneId transcriptId distanceToTSS
##        <integer>  <integer> <character>  <character>     <numeric>
##   [1]       9753          2       50805   uc031sir.1       -269449
##   [2]      17023          1      400087   uc001uhp.2       -124685
##   [3]     110513          1      644150   uc022aaz.1        -34908
##   [4]     417407          2        6660   uc001rfw.3         26936
##   [5]     165242          1        6875   uc002kvu.4         51251
##                                                                  flank_txIds
##                                                                  <character>
##   [1]                                                                   <NA>
##   [2]                                                                   <NA>
##   [3]                                                                   <NA>
##   [4] uc001rfw.3;uc010siv.3;uc001rfx.4;uc001rfy.4;uc010siw.1;uc001rfz.1;uc001rga.3
##   [5]                                                 uc002kvs.4;uc002kvt.4;uc002kvu.4
##                             flank_geneIds flank_gene_distances
##                               <character>          <character>
##   [1]                                 <NA>                 <NA>
##   [2]                                 <NA>                 <NA>
##   [3]                                 <NA>                 <NA>
##   [4] 6660;6660;6660;6660;6660;6660;6660        0;0;0;0;0;0;0
##   [5]                      6875;6875;6875                0;0;0
##   -------
##   seqinfo: 22 sequences from hg19 genome
```

Figure 3 GRange object of ChIPseq_CTCF_demo_anno_gene_name (**A**) and ChIPseq_CTCF_demo_anno_flank_5_kb (**B**).

```
ranges = IRanges(start = 55267513:55267522,
                         end = 55714466:55714475),
strand = Rle(strand(c("-","+","*","+","-")),
                         c(1, 2, 1, 4, 2)))
```

Now, one can annotate epigenetic regions by the GRange object above. Please note that if there are multiple functional epigenetic regions that overlap with a defined region, only the nearest functional epigenetic region will be annotated (Fig. 4A).

A

```
## Annotated peaks generated by ChIPseeker
## 153/1000  peaks were annotated
```

B

```
## GRanges object with 1 range and 8 metadata columns:
##       seqnames            ranges strand |      CTCF_peaks     level   geneChr
##          <Rle>         <IRanges>  <Rle> |        <factor> <numeric> <integer>
##   [1]     chr1 55267514-55267548      * | CTCF_merge_2368        70         1
##       geneStart   geneEnd geneLength geneStrand distanceToTSS
##       <integer> <integer>  <integer>  <integer>     <numeric>
##   [1]  55267517  55714470     446954          1             0
##   -------
##   seqinfo: 22 sequences from an unspecified genome; no seqlengths
```

C

```
##                                                    qSample
## ARmo_0M    GSM1295077_CBX7_BF_ChipSeq_mergedReps_peaks.bed.gz
## ARmo_1nM   GSM1295077_CBX7_BF_ChipSeq_mergedReps_peaks.bed.gz
## ARmo_100nM GSM1295077_CBX7_BF_ChipSeq_mergedReps_peaks.bed.gz
## CBX6_BF    GSM1295077_CBX7_BF_ChipSeq_mergedReps_peaks.bed.gz
##                                                    tSample qLen tLen N_OL
## ARmo_0M                      GSM1174480_ARmo_0M_peaks.bed.gz 1663  812    0
## ARmo_1nM                    GSM1174481_ARmo_1nM_peaks.bed.gz 1663 2296    8
## ARmo_100nM                GSM1174482_ARmo_100nM_peaks.bed.gz 1663 1359    3
## CBX6_BF    GSM1295076_CBX6_BF_ChipSeq_mergedReps_peaks.bed.gz 1663 1331  968
##                pvalue  p.adjust
## ARmo_0M    0.81818182 0.8181818
## ARmo_1nM   0.09090909 0.1818182
## ARmo_100nM 0.27272727 0.3636364
## CBX6_BF    0.09090909 0.1818182
```

Figure 4 (**A**) The annotation output of ChIPseq_CTCF_demo_anno_GR. (**B**) GRange object of ChIPseq_CTCF_demo_anno_GR. (**C**) Detail output of enrichPeakOverlap function.

```
ChIPseq_CTCF_demo_anno_GR = annotatePeak(ChIPseq_CTCF_demo,
                                         TxDb = user_defined_GRange)
ChIPseq_CTCF_demo_anno_GR
```

And by the value in the distanceToTSS column, the overlapped regions between them can be identified (Fig. 4B).

```
as.GRanges(ChIPseq_CTCF_demo_anno_GR)[as.GRanges(ChIPseq_CTCF_demo_anno_GR)
   $distanceToTSS == 0]
```

12. Users also have the option to annotate with the preloaded functional epigenetic regions. To check whether the CTCF motif is co-located with DNA methylation, one could annotate CTCF by DNA methylation loci using the `annotatePeak()` function.

```
CTCF_demo_anno_with_m5C_demo = annotatePeak(Epi_data_list$CTCF,
                                            TxDb=Epi_data_list$m5C)
```

BASIC PROTOCOL 3 COMPARISON OF EPIGENOMIC DATASETS

Although the complicated relationships of different epigenetic marks have been gradually uncovered, there is still limited knowledge of cooperativity between marks. Therefore, it is critical to compare diverse epigenomic datasets. By using ChIPseeker's features, biologists can easily get a preliminary hit of the relationship between multiple marks.

In this protocol, the visualization of the overlapped results among different functional regions is demonstrated. Also, ChIPseeker can be used to figure out if there is a significant enrichment between epigenetic regions by using the permutation test.

Necessary Resources

Hardware

The same requirement as Basic Protocol 1

Software

The same requirement as Basic Protocol 1

Files

The files that have been loaded in Basic Protocol 1, and annotated results from Basic Protocol 2

1. Installing and loading Bioconductor package (Gao, Yu, & Cai, 2021).
   We need to install the ggVennDiagram package (Gao et al., 2021) by BiocManager.

```
BiocManager::install("ggVennDiagram")
```

   After that, the ggVennDiagram package could be loaded.

```
library(ggVennDiagram)
```

2. Venn diagram of overlapping genes among epigenomic datasets.
   Based on the annotated results from Basic Protocol 2, the Venn diagram of overlapping genes among different types of epigenomic datasets is supported by ChIPseeker. It is known that there is a close relationship between CTCF and DHSs. Here we use them as an example (Fig. 5A).

```
vennplot(
    list(DHSs  =  as.data.frame(peakAnnoList_user_defined$DHSs)$geneId,
      CTCF = as.data.frame(peakAnnoList_user_defined$CTCF)$geneId),
      by = "ggVennDiagram")
```

   A Venn plot based on the annotated results from a GRangeList object might easily be created with the `vennplot()` function (Fig. 5B). Currently, `vennplot()` only supports the data within five dimensions.

```
peakAnnoList_user_defined_gene  =  lapply(peakAnnoList_user_defined,
                                   function(i) as.data.frame(i)$geneId)
vennplot(peakAnnoList_user_defined_gene,
     by = "ggVennDiagram",
     label_percent_digit = 2,
     edge_size = 1.5)
```

3. Enrichment analysis of overlap between epigenetic marks.
   Besides direct overlap of annotated results of multiple epigenetic marks, we also provide a permutation test between different profiles. We used the shuffle function to randomly shift each genomic site of a specified GRange object or BED/bedGraph format on the same chromosome, and the size of each feature was kept. The random data was created to estimate the overlap's background null distribution for the permutation test. The probability of detecting extreme overlap between the query and target marks is used to calculate the *p*-value, and several false discovery rate (FDR) adjustments are incorporated.

   In the output results (Fig. 4C), columns of qSample represent the query epigenomic dataset; tSample represents the target dataset for comparison; nShuffle represents the number of times that the target dataset is shuffled; qLen represents the number of regions in qSample; N_OL represents the number of overlaps between qSample and tSample; chainFile represents the liftover file between the tSample and qSample when their reference genomes are different. To check the detail of the lifeover file, please visit UCSC website (*https://genome.ucsc.edu/goldenPath/help/hgTracksHelp.html#Liftover*).

```
files                  <-          getSampleFiles()
enrichPeakOverlap(queryPeak = files[[5]],
                  targetPeak = unlist(files[1:4]),
```

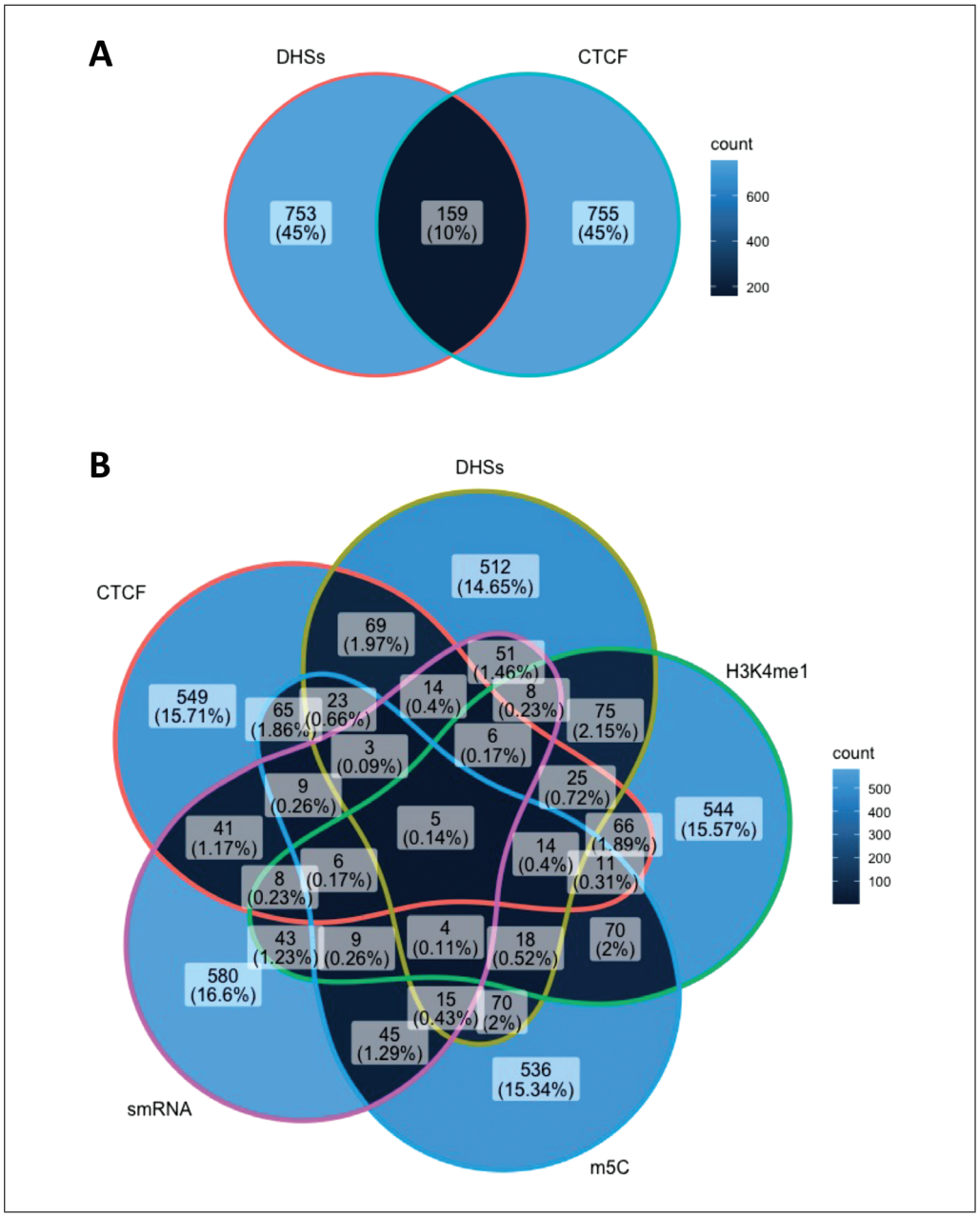

Figure 5 Venn plot of the genes that annotated in various epigenomics datasets. (**A**) Venn plot of the genes that annotated to CTCF peaks and DHSs peaks. (**B**) Venn plot of the genes that annotated to CTCF peaks, DHSs peaks, H3K4me1 peaks, m5C loci, and smRNA loci.

```
TxDb = TxDb_hg19,
pAdjustMethod = "BH",
nShuffle = 10,
chainFile = NULL,
verbose = FALSE)
```

Please note that the shuffle times should be set to 1000 or above for a more robust result. Here we only shuffle ten times for demonstration.

VISUALIZATION OF ANNOTATED RESULTS

BASIC PROTOCOL 4

It is well known that presenting results in graphic form conveys more understandable information than text. ChIPseeker incorporates a wide variety of functions to create plots of annotated results from Basic Protocol 2, all of which are geared toward making these results easier to understand.

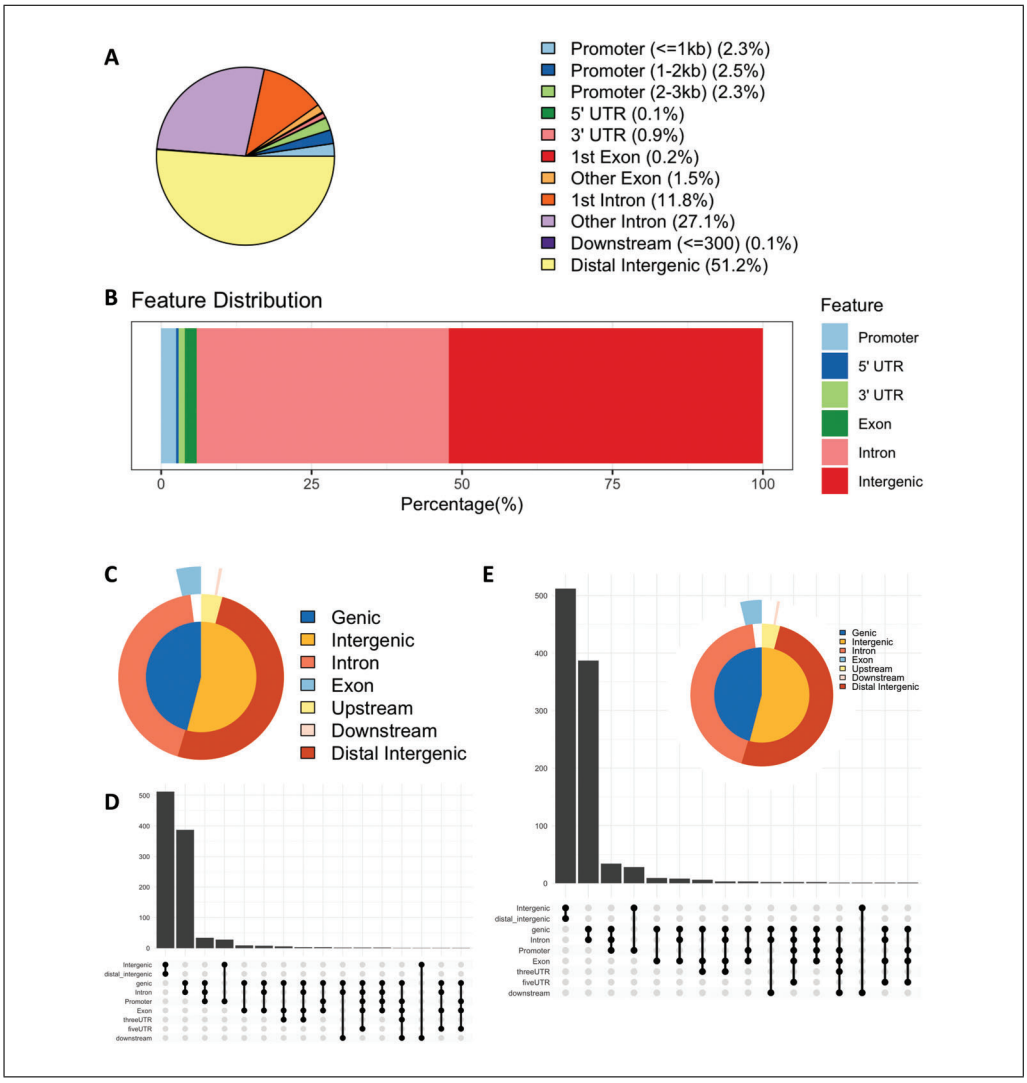

Figure 6 Various plots of annotated results. (**A**) Pie plot of the annotated results of CTCF peaks. (**B**) Bar plot of the annotated results of CTCF peaks. (**C**) Vennpie plot of the annotated results of CTCF peaks. (**D**) Upset plot of the annotated results of CTCF peaks. (**E**) Upset and vennpie plots of the annotated results of CTCF peaks.

In this protocol, we will demonstrate the functions that are deployed in ChIPseeker for creating plots of annotated results. The publication-level plots of pie, bar, vennpie, and Upset can be obtained easily and quickly.

Necessary Resources

Hardware

The same requirement as Basic Protocol 1

Software

The same requirement as Basic Protocol 1

Files

Annotated results from Basic Protocol 2

1. Visualization of genomic annotated results.

   ChIPseeker plots based on the annotation column from Basic Protocol 2 will be made in a variety of methods.

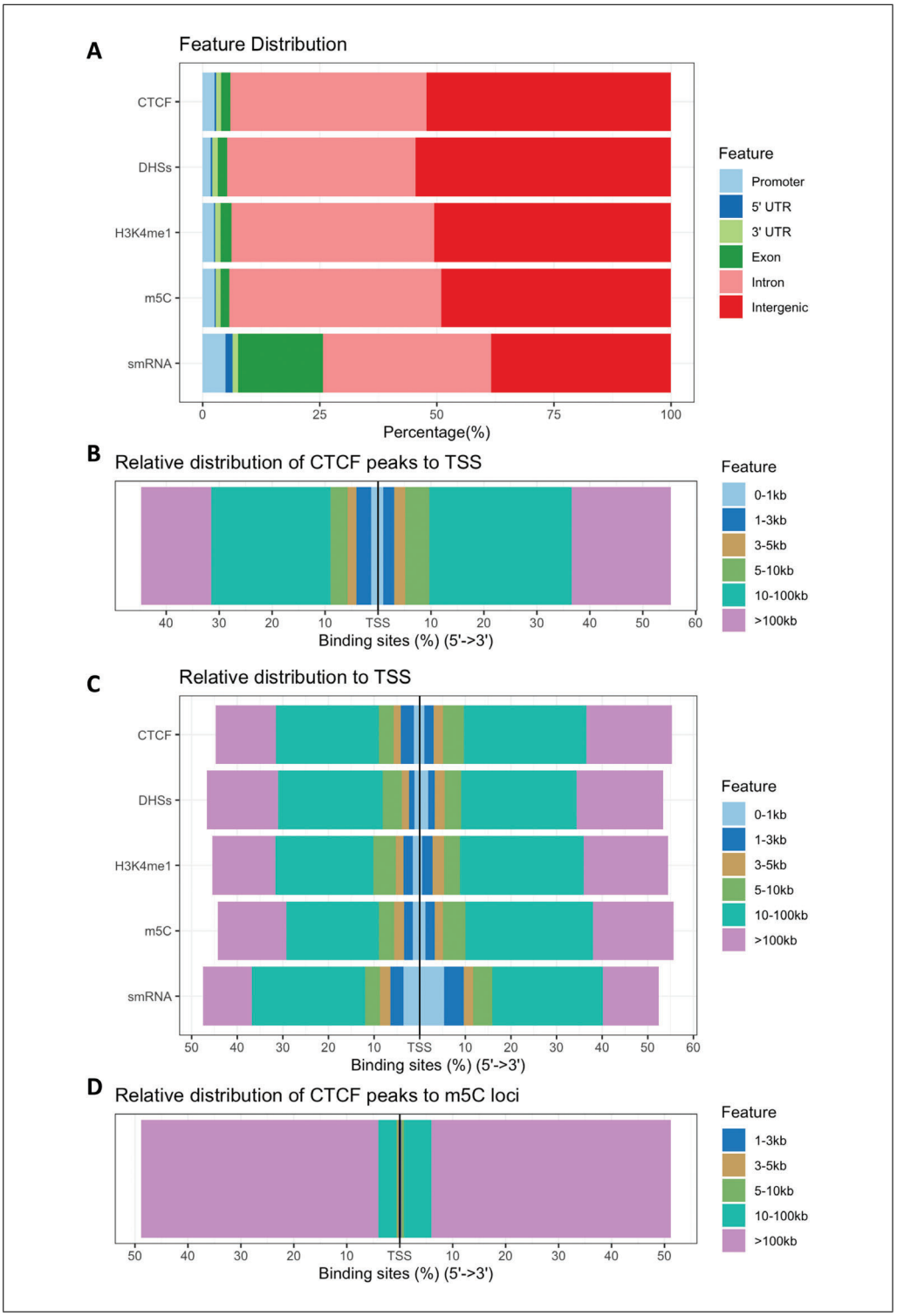

Figure 7 Bar plots of annotated results. (**A**) Bar plot of the annotated results of CTCF peaks, DHSs peaks, H3K4me1 peaks, m5C loci, and smRNA loci. (**B**) Bar plot of the relative distribution of CTCF peaks to TSS. (**C**) Bar plot of the relative distribution of CTCF peaks, DHSs peaks, H3K4me1 peaks, m5C loci, and smRNA loci to TSS. (**D**) Bar plot of the relative distribution of CTCF peaks to m5C loci.

Pie plot (Fig. 6A):

```
plotAnnoPie(ChIPseq_CTCF_demo_anno_default)
```

Bar plot (Fig. 6B) :

```
plotAnnoBar(ChIPseq_CTCF_demo_anno_user_defined)
```

The Venn diagram, which is presented as a pie chart, can be used to visualize both the full annotation and their partially overlapping results in a vennpie plot (Fig. 6C):

```
vennpie(ChIPseq_CTCF_demo_anno_default)
```

Upset plot is used to visualize the full overlapping results (Fig. 6D) :

```
upsetplot(ChIPseq_CTCF_demo_anno_default)
```

We could also add the vennpie plot to the Upset plot by setting the parameter of `vennpie=TRUE`. (Fig. 6E):

```
upsetplot(ChIPseq_CTCF_demo_anno_default, vennpie=TRUE)
```

In addition, the `plotAnnoBar()` function supports a list of GRange objects (Fig. 7A):

```
plotAnnoBar(peakAnnoList_user_defined)
```

2. Visualization of the relative distribution of epigenomic datasets in relation to a specific set of loci.

   Besides the plots that reflect the annotation of genomic regions. ChIPseeker also supports the bar plot for the relative distribution of the epigenomic dataset to the nearest TSS (Fig. 7B):

   ```
   plotDistToTSS(ChIPseq_CTCF_demo_anno_default,
         title="Relative distribution of CTCF peaks to TSS")
   ```

   We can make the bar plot by using the GRangeList object (Fig. 7C):

   ```
   plotDistToTSS(peakAnnoList_user_defined,
         title="Relative distribution to TSS")
   ```

   We can also visualize the annotated results of user-defined regions. Here we used the relative distribution of CTCF peaks to DNA methylation loci as an example (Fig. 7D):

   ```
   plotDistToTSS(CTCF_demo_anno_with_m5C_demo,
         title="Relative distribution of CTCF peaks to m5C loci")
   ```

FUNCTIONAL ENRICHMENT ANALYSIS OF EPIGENOMIC DATASETS

BASIC PROTOCOL 5

Functional enrichment analysis is always used in downstream analysis to discover the most significant biological function of a set of genes. One could simply perform functional enrichment analysis by extracting the annotated results from Basic Protocol 2. Besides attaching the epigenomic region to its nearest gene, ChIPseeker also offers the `seq2gene()` function to extract all the combinations between the gene and its promoter and flanking regions and the epigenomic regions. After a gene of interest is obtained, functional enrichment analysis is performed. There are several well-known methods of functional enrichment analysis. Our team has developed several R packages to perform these analyses, including DOSE (Yu, Wang, Yan, & He, 2015) for Disease Ontology analysis (Schriml et al., 2012); ReactomePA (Yu & He, 2016) for the Reactome pathway analysis (Croft et al., 2014); clusterProfiler (Wu, Xu, Guo, & Yu, 2021) for Gene Ontology (GO; Ashburner et al., 2000); and Kyoto Encyclopedia of Genes and Genomes (KEGG; Kanehisa, Goto, Kawashima, Okuno, & Hattori, 2004) for enrichment analysis.

And the clusterProfiler package also includes multiple methods to visualize the enrichment results.

In this protocol, we will demonstrate how to extract the genes that are related to epigenetic regions. Then, the users could pass any genes of interest to the enrichment analyses. There is also a brief demonstration of GO enrichment analysis and its visualization.

Necessary Resources

Hardware

The same requirement as Basic Protocol 1

Software

The software that is required in Basic Protocol 1

The Bioconductor package `clusterProfiler`

Files

The files that have been loaded in Basic Protocol 1 and annotated results from Basic Protocol 2.

1. Installing and loading the clusterProfiler library.

```
BiocManager::install("clusterProfiler")
library("clusterProfiler")
```

2. Extraction of the genes related to epigenetic marks can occur by several methods.

   a. Gene extraction from annotated results

   To do the functional analysis of the annotated results from Basic Protocol 2, we could directly extract their IDs by the column of geneId.

```
ChIPseq_CTCF_demo_anno_default_genes =
  as.data.frame(peakAnnoList_user_defined$CTCF)$geneId
```

   We could get their IDs from the annotated results of a GRangeList object in order to do the functional analysis by comparing different groups of genes.

```
peakAnnoList_user_defined_genes = lapply(peakAnnoList_user_defined, function(i)
  as.data.frame(i)$geneId)
```

   b. Gene extraction from the genes that are related to epigenetic marks in many-to-many mode.

   In addition to the one-to-many annotated results, ChIPseeker also offers extracting genes in a many-to-many mode. The function of `seq2gene()` could extract these kinds of genes with the files that were loaded from Basic Protocol 1 and it can extract the interaction of epigenetic marks with genes, promoters, and flanking regions. We are retrieving the interaction between the peaks of H3K4me3 as an example here.

```
ChIPseq_H3K4me3_demo2_genes = seq2gene(ChIPseq_H3K4me3_demo2,
                                       tssRegion = c(-1000, 1000),
                                       flankDistance = 500,
                                       TxDb = TxDb_hg19)
```

   As mentioned in substep a, we could also extract the genes in many-to-many mode with a list of epigenomic datasets.

```
Epi_data_list_genes = lapply(Epi_data_list, function(i) seq2gene(i, tssRegion
  = c(-1000, 1000), flankDistance = 500, TxDb = TxDb_hg19))
```

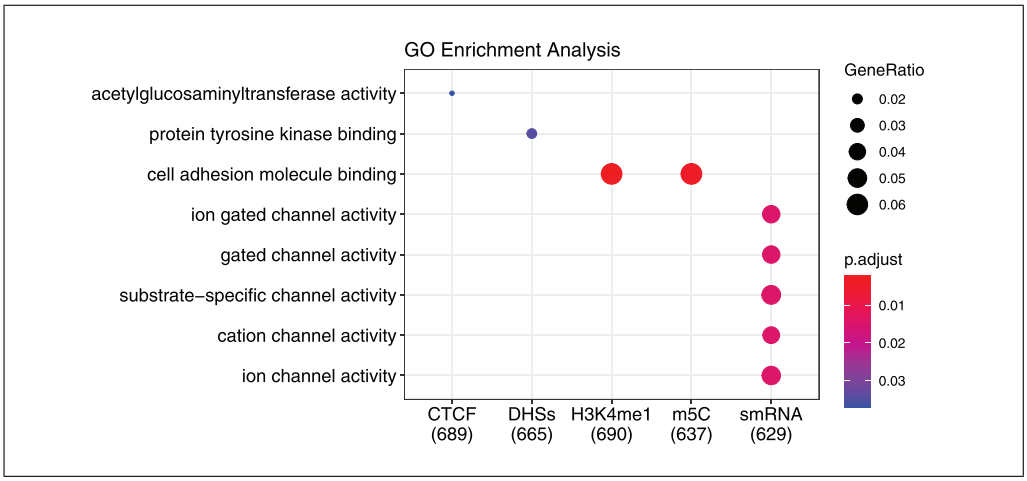

Figure 8 GO enrichment analysis of CTCF peaks, DHSs peaks, H3K4me1 peaks, m5C loci, and smRNA loci-related genes.

3. Enrichment analysis of the genes of interest.

   After getting the genes, pass them for functional enrichment analysis. As mentioned above, our team has developed several packages for different kinds of enrichment analysis. Please see *https://yulab-smu.top/biomedical-knowledge-mining-book/* for more information on these tools. Here we will use the GO analysis of CTCF peaks as a demonstration (Wu et al., 2021).

```
ChIPseq_CTCF_demo_anno_default_enrichGO = enrichGO(gene =
 ChIPseq_CTCF_demo_anno_default_genes,
                                                    OrgDb = "org.Hs.eg.db")
```

   In addition to the enrichment analysis of a set of genes, the function of `compareCluster()` offered by clusterProfiler could be used to perform the enrichment analysis on multiple sets of epigenetic regions (Wu et al., 2021).

```
Epi_data_list_genes_enrichGO = compareCluster(geneCluster =
   peakAnnoList_user_defined_genes,
                                              fun = "enrichGO",
                                              pvalueCutoff = 0.05,
                                              OrgDb = "org.Hs.eg.db")
```

4. Enrichment results visualization.
   In order to have a better view of the functional enrichment results, we also developed several plots for different kinds of enrichment results (Wu et al., 2021). Here we only use `dotplot()` to do the demonstration (Fig. 8).

```
dotplot(Epi_data_list_genes_enrichGO,
        title = "GO Enrichment Analysis")
```

GENOME-WIDE AND LOCUS-SPECIFIC DISTRIBUTION OF EPIGENOMIC DATASETS

BASIC PROTOCOL 6

Epigenetic markers may have different distributions across the genome, such as in the chromatin structure of heterochromatin and euchromatin, respectively. Besides, examining the overall distribution of epigenetic marks with similar published datasets will provide you with an indication of the data quality. Additionally, the locus-specific plot may highlight the specific distribution in a defined locus, such as differential regions. Overall, visualization of the genome-wide and locus-specific distributions of epigenomic datasets is very important.

This protocol will show you how to use the data from Basic Protocol 1 to generate graphs of genome-wide and locus-specific distribution.

Necessary Resources

Hardware

The same requirement as Basic Protocol 1

Software

The software that is required in Basic Protocol 1. The Bioconductor packages ggpplot2 and RColorBrewer

Files

The files that have been loaded in Basic Protocol 1

1. Installing and loading Bioconductor packages

   We need to install the ggplot2 and RColorBrewer packages by BiocManager.

```
BiocManager::install("ggplot2")
BiocManager::install("RColorBrewer")
```

   After that, the ggplot2 and RColorBrewer packages could be loaded.

```
library(ggplot2)
library(RColorBrewer)
```

2. Visualization of genome-wide distribution of the epigenomic datasets.
   Genome-wide distribution of a single file that is loaded in Basic Protocol 1 (Fig. 9A):

```
covplot(ChIPseq_CTCF_demo,
     weightCol="level",
      title="Genome-wide distribution of CTCF peaks")
```

   To get the genome-wide distribution of multiple files, we could plot the overall distribution of the `GRangeList` object (Fig. 10A).

```
covplot(Epi_data_list,
    weightCol="level",
    title="Genome-wide distribution") + labs(color="Library
type",fill="Library type") + scale_fill_brewer(palette = "Set2") +
scale_color_brewer(palette = "Set2")
```

   Also, we could separate the GRangeList in multi-panel by `facet_grid()` (Fig. 10B).

```
covplot(Epi_data_list,
    weightCol="level",
    title="Genome-wide distribution") + facet_grid(chr ~
.id,scales="free") + labs(color="Library type",fill="Library type") +
scale_fill_brewer(palette = "Set2") + scale_color_brewer(palette = "Set2")
```

3. Visualization of the locus-specific distribution of the epigenomic datasets

   We could make the locus-specific plot by `covplot()` (Fig. 9B).

```
covplot(ChIPseq_CTCF_demo,
     weightCol = "level",
     title = "locus-specific distribution of CTCF peaks",
     chrs = c("chr2","chr4"),
     xlim = c(1e6, 8e6))
```

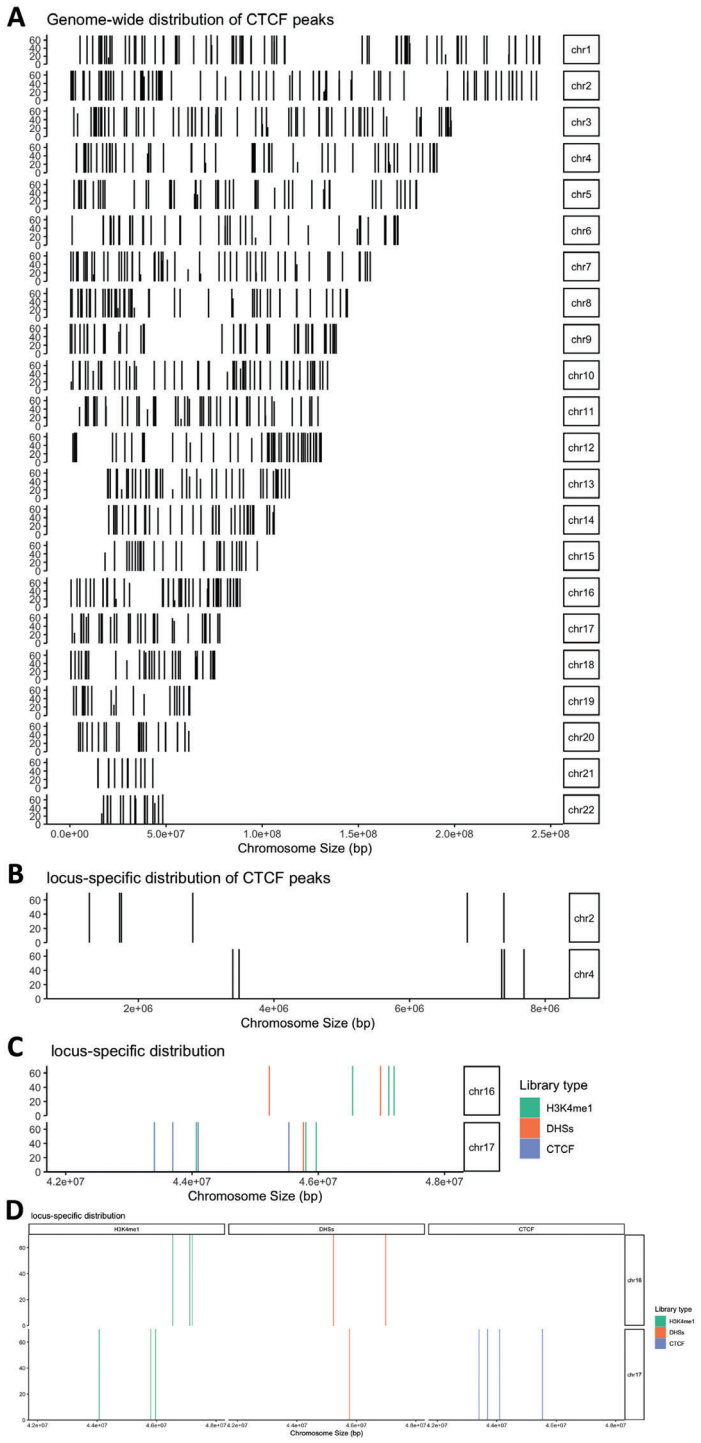

Figure 9 Genome-wide (**A**) and locus-specific (**B**) distribution of CTCF peaks. (**C**) Locus-specific distribution of CTCF, DHSs, and H3K4me1 peaks in single- (**C**) and multi- (**D**) panels.

Aside from the plot of the GRange object, we can also construct a locus-specific plot of GRangeList (Fig. 9C).

```
covplot(Epi_data_list,
        weightCol="level",
        title=" locus-specific distribution",
        chrs=c("chr16","chr17"),
```

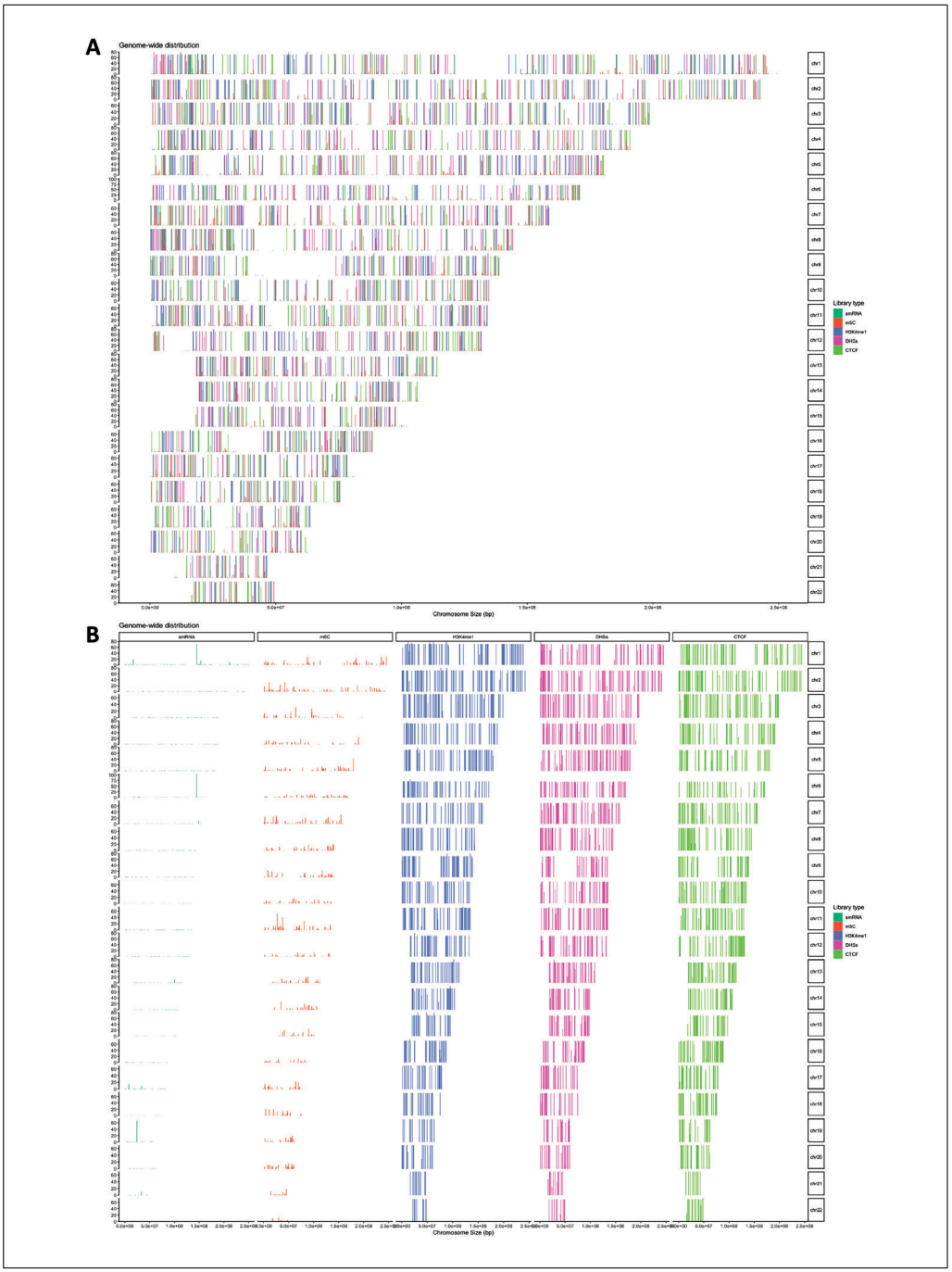

Figure 10 Genome-wide distribution of CTCF peaks, DHSs peaks, H3K4me1 peaks, m5C loci, and smRNA loci in single- (**A**) and multi- (**B**) panels.

```
        xlim=c(4.2e7, 4.8e7)) + labs(color="Library type",fill="Library type")
+ scale_fill_brewer(palette = "Set2") + scale_color_brewer(palette = "Set2")
```

We might also use the facet_grid() to plot the locus-specific in a multi-panel way (Fig. 9D).

```
covplot(Epi_data_list, weightCol="level",
        title=" locus-specific distribution",
        chrs=c("chr16","chr17"),
        xlim=c(4.2e7, 4.8e7)) + facet_grid(chr ~ .id,scales="free") +
labs(color="Library type",fill="Library type") + scale_fill_brewer(palette =
"Set2") + scale_color_brewer(palette = "Set2")
```

HEATMAPS AND METAPLOTS OF EPIGENOMIC DATASETS

There is always a distinct distribution of epigenetic marks across the biological regions. Therefore, it is necessary to profile the epigenomic datasets in a range of biological functional regions in order to have a better understanding of the regulation relationship between them.

In this protocol, we will introduce the profiling of epigenomic datasets in the biological regions by two types of plots: heatmap and metaplot. Besides the plots, here are two typical biological regions that require our attention. One is the distribution surrounding a set of biological loci, such as the transcription start sites (TSS). Another one is the distribution throughout a list of biological regions, such as gene bodies. In ChIPseeker, we need to first organize the data surrounding the biological regions that we are interested in. After that, we provide the functions to construct the plots. Additionally, we provide one-line commands to construct plots around the TSS and gene bodies, as these are the most commonly studied regions.

Necessary Resources

Hardware

The same requirement as Basic Protocol 1

Software

The same requirement as Basic Protocol 2

Files

The files that loaded in Basic Protocol 1

1. Preparing the tagMatrix.

   a. For a set of biological loci

   To obtain the plots around the functional loci, we need to prepare the desired data by `getTagMatrix()`. We could choose the type of `start_site` and `end_site`, and by gene, transcript, exon, intron, 3UTR , or 5UTR .

```
H3K4me3_TES_tagMatrix = getTagMatrix(ChIPseq_H3K4me3_demo2,
                        TxDb=TxDb_hg19, type = "end_site",
                        upstream = 3000,
                        downstream = 3000,
                        by = "gene")
```

   To obtain the plots around the functional loci, we must first prepare the windows we want by using `getBioRegion()`.

```
TSS           =           getBioRegion(TxDb=TxDb_hg19,
    upstream=3000,
    downstream=3000,
    by          =                              "gene",
    type = "start_site")
```

   Then we could prepare the tagMatrix by `getTagMatrix()`. In addition, we can also prepare the epigenomic datasets by using `lapply()`.

```
Epi_data_list_TSS_tagMatrix = lapply(Epi_data_list,
                              getTagMatrix,
                              windows = TSS)
```

   b. For a set of biological regions.

In addition to the functional loci, we need to prepare the tagMatrix in the functional regions and their surrounding regions. The `getBioRegion()` can also be used to prepare our interesting regions by setting the parameter of `type="body"`.

```
geneBody = getBioRegion(TxDb = TxDb_hg19,
                        by = "gene",
                        type = "body")
```

Since functional regions may vary in size, users can specify the normalizing length with the parameter `nbin`.

```
ChIPseq_H3K4me3_demo2_geneBody_tagMatrix =
getTagMatrix(ChIPseq_H3K4me3_demo2,
                    windows=geneBody,
                    nbin = 500,
                    upstream=2000,
                    downstream=2000)
```

Then we could prepare the tagMatrix by `getTagMatrix()`. In addition, we can also prepare the epigenomic datasets by using `lapply()`.

```
Epi_data_list_geneBody_tagMatrix = lapply(Epi_data_list,
                getTagMatrix,
                windows=geneBody,
                nbin = 800,
                upstream=1000,
                downstream=1000)
```

2. Heatmaps.

   a. For a set of biological loci

   We can make a heatmap plot of our selected regions (Fig. 11A).

```
tagHeatmap(H3K4me3_TES_tagMatrix,
        xlim=c(-3000, 3000),
        title = "H3K4me3 peaks around TES")
```

   The `tagHeatmap()` function also supports a list of tagMatrix (Fig. 11B).

```
tagHeatmap(Epi_data_list_TSS_tagMatrix,
        xlim=c(-3000, 3000),
        color=brewer.pal(length(Epi_data_list_TSS_tagMatrix),
          "Dark2"))
```

   TSS are the most commonly used point positions in this mode. ChIPseeker offers a one-line function `peakHeatmap()` for generating the heatmap graphic around the TSS regions from the data loaded in Basic Protocol 1 (Fig. 11C).

```
peakHeatmap(ChIPseq_H3K4me3_demo2,
               weightCol = "level",
               TxDb=TxDb_hg19,
               upstream=2000,
               downstream=2000,
               title = "H3K4me3 peaks around TSS")
```

   The `peakHeatmap()` function supports generating the heatmap of a list of BED files (Fig. 11D; Pemberton et al., 2014; Urbanucci et al., 2012).

```
peakHeatmap(files,
      TxDb = TxDb_hg19,
      upstream = 2000,
      downstream = 2000)
```

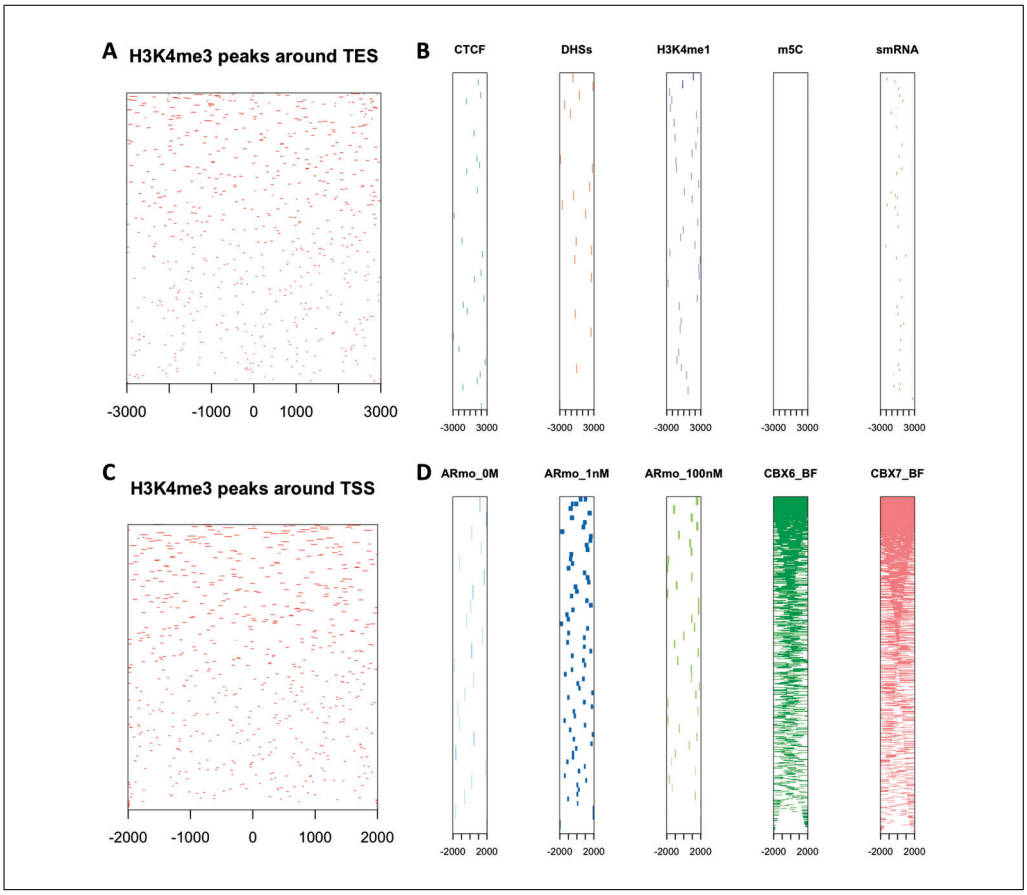

Figure 11 Heatmaps of epigenomics datasets. (**A**) Heatmap of H3K4me3 peaks around transcription end sites (TES). (**B**) Heatmap of CTCF peaks, DHSs peaks, H3K4me1 peaks, m5C loci, and smRNA loci around transcription start sites (TSS). (**C**) Heatmap of H3K4me3 peaks around transcription start sites (TSS). (**D**) Heatmap of CBX6, CBX7, ARmo 0 nM, ARmo 1 nM, and ARmo 100 nM peaks around transcription start sites (TSS).

Currently, ChIPseeker only supports the heatmap plot for a set of biological loci.

3. Metaplots.

   a. For a set of biological loci.

   We can make a metaplot around our selected biological loci (Fig. 12A). Estimate the variation of statistics obtained from the same data by performing computations on the data itself. The optional parameter conf features the method for bootstrap confidence intervals.

```
plotAvgProf(H3K4me3_TES_tagMatrix,
            xlim=c(-3000, 3000),
            origin_label="TES",
            conf = 0.95,
            resample = 500)
```

   The `plotAvgProf()` function also supports a list of tagMatrix (Fig. 12B).

```
plotAvgProf(Epi_data_list_TSS_tagMatrix,
            xlim=c(-3000, 3000),
            ylab = "Count Frequency")
```

   Like the heatmaps, ChIPseeker also provides a one-line function called `plotAvgProf2()` to generate the metaplots of the epigenomic datasets that are loaded in Basic Protocol 1 (Fig. 12C).

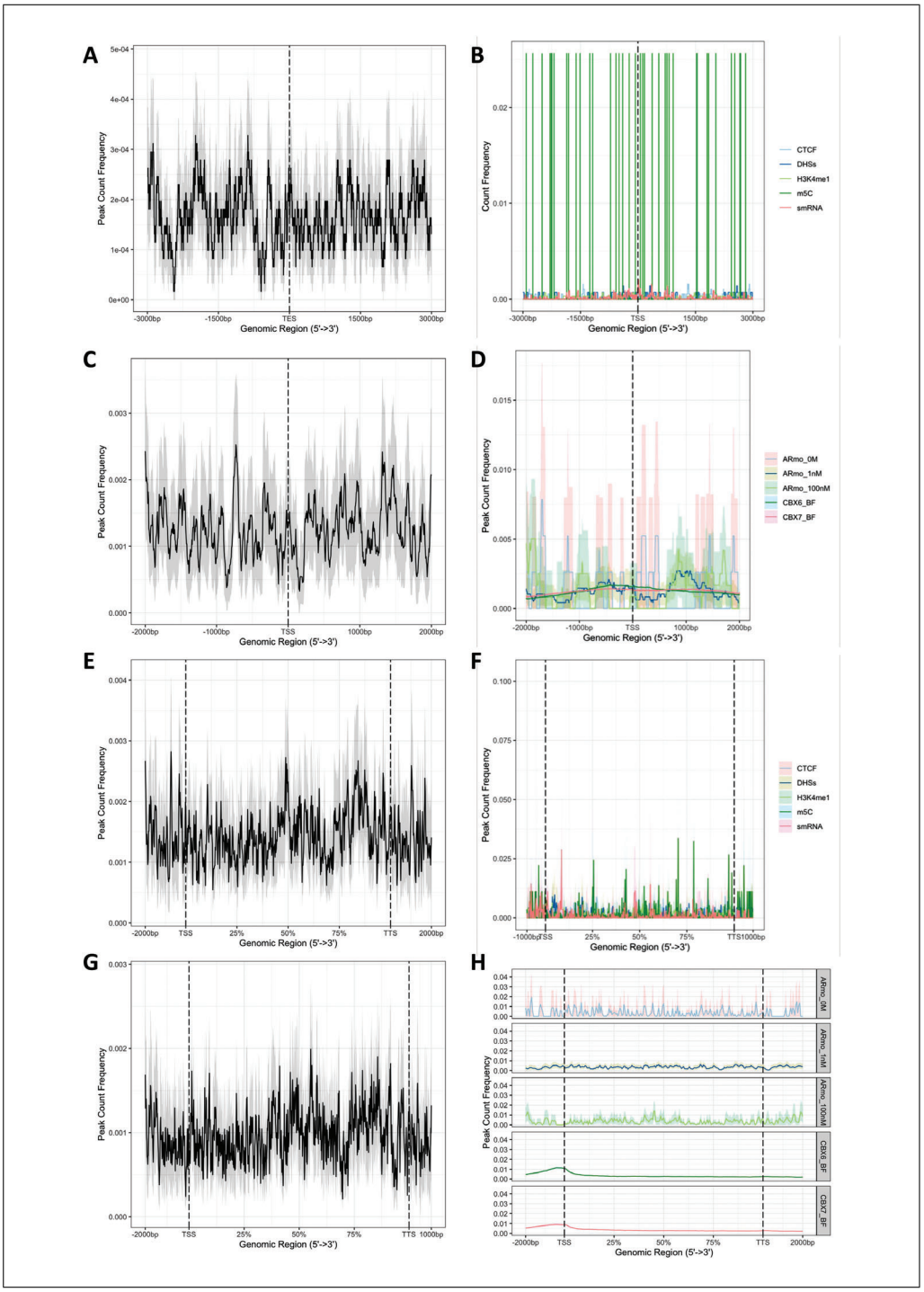

Figure 12 Metaplots of epigenomics datasets. (**A**) Metaplot of H3K4me3 peaks around transcription end sites (TES). (**B**) Metaplot of CTCF peaks, DHSs peaks, H3K4me1 peaks, m5C loci, and smRNA loci around transcription start sites (TSS). (**C**) Metaplot of H3K4me3 peaks around transcription start sites (TSS). (**D**) Metaplot of CBX6, CBX7, ARmo 0 nM, ARmo 1 nM, and ARmo 100 nM peaks around transcription start sites (TSS). (**E**) Metaplot of H3K4me3 peaks in genes and their flanking regions. (**F**) Metaplot of CTCF peaks, DHSs peaks, H3K4me1 peaks, m5C loci, and smRNA loci in genes and their flanking regions.(**G**) Metaplot of H3K4me3 peaks in genes and their flanking regions. (**H**) Metaplot of CBX6, CBX7, ARmo 0 nM, ARmo 1 nM, and ARmo 100 nM peaks in genes and their flanking regions.

```
plotAvgProf2(ChIPseq_H3K4me3_demo2,
          weightCol = "level",
          TxDb=TxDb_hg19,
          upstream=2000,
          downstream=2000,
          conf = 0.95,
          resample=500,
          facet="row")
```

The `plotAvgProf2()` function supports generating the heatmap of a list of BED files (Fig. 12D).

```
plotAvgProf2(files,
          TxDb=TxDb_hg19,
          upstream=2000,
          downstream=2000,
          conf = 0.95)
```

b. For a set of biological regions.

We can make a metaplot of our selected biological regions by (Fig. 12E).

```
plotPeakProf(ChIPseq_H3K4me3_demo2_geneBody_tagMatrix,
          conf = 0.95)
```

The `plotPeakProf()` function also supports a list of tagMatrix (Fig. 12F).

```
plotPeakProf(Epi_data_list_geneBody_tagMatrix,
          facet = "none",
          conf = 0.95)
```

And `plotPeakProf2()` function is a one-line function to make metaplots within the scale regions (Fig. 12G).

```
plotPeakProf2(peak = ChIPseq_H3K4me3_demo2,
          upstream = 2000,
          downstream = 1000,
          conf = 0.95,
          by = "gene",
          type = "body",
          nbin = 800,
          TxDb = TxDb_hg19,
          weightCol = "level",
          ignore_strand = F)
```

And `plotPeakProf2()` function supports generating the metaplots of a list of BED files (Fig. 12H).

```
plotPeakProf2(peak = files,
          upstream = 2000,
          downstream = 2000,
          conf = 0.9,
          by = "gene",
          type = "body",
          nbin = 200,
          TxDb = TxDb_hg19,
          ignore_strand = F,
          facet = "row",
          free_y = F)
```

COMMENTARY

Background Information

Epigenomics has become a powerful tool to study epigenetics. It aims to understand the regulatory implications of experimentally discovered regions. Nowadays, nearly all sequencing methods in epigenomics create regions of interest at a genome-wide level. It can be hard to figure out what these regions mean because most of them go beyond the well-defined protein-coding genes.

To incorporate epigenomic datasets, a range of computational tools have been created. While most of them focused on one field of interpretation with limitations, such as: ChIPpeakAnno (Zhu et al., 2010) ,which offers annotation without considering stranded information; PeakAnalyzer (Salmon-Divon, Dvinge, Tammoja, & Bertone, 2010) and PAVIS (Huang, Loganantharaj, Schroeder, Fargo, & Li, 2013), which only provide annotation with a single dataset; ngs.plot (Shen, Shao, Liu, & Nestler, 2014) and deeptools2 (Ramírez et al., 2016), which only present the visualization of metaplots and heatmaps; and regioneR (Gel et al., 2016), which supplies the comparison of genomic regions in the absence of a database.

We developed ChIPseeker with numerous essential functionalities for annotation (Basic Protocol 2), comparison (Basic Protocol 3), and visualization (Basic Protocols 4, 6, and 7) of epigenomic datasets. Furthermore, we have collected a database that includes around 17,000 epigenomic datasets (Basic Protocol 1). In addition to annotation, its results can be easily received by other functional enrichment analysis software (Basic Protocol 5). Also, we also give genome-wide and locus-specific visualization of epigenomic datasets (Basic Protocol 7). The ChIPseeker, on the other hand, has been limited by its name. As demonstrated, it could be applied to any type of epigenomic dataset.

Critical Parameters

When working with ChIPseeker, Basic Protocols 1 and 2 are the essential functions because they are prerequisites for downstream analysis in the other protocols.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (32270677) and the Startup Fund from Southern Medical University.

Author Contributions

Qianwen Wang: conceptualization, writing original draft, writing review and editing; **Ming Li**: software, writing review and editing; **Tianzhi Wu**: investigation; **Li Zhan**: data curation; **Lin Li**: data curation; **Meijun Chen**: visualization; **Wenqin Xie**: validation; **Zijing Xie**: validation; **Erqiang Hu**: writing review and editing; **Shuangbin Xu**: writing review and editing; **Guangchuang Yu**: conceptualization, software, supervision.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

All the data used in this protocol may be downloaded from *https://github.com/YuLab-SMU/ChIPseeker_current_protocols*.

Literature Cited

Allis, C. D., & Jenuwein, T. (2016). The molecular hallmarks of epigenetic control. *Nature Reviews Genetics*, *17*, 487–500. doi: 10.1038/nrg.2016.59

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., … Sherlock, G. (2000). Gene Ontology: Tool for the unification of biology. *Nature Genetics*, *25*, 25–29. doi: 10.1038/75556

Croft, D., Mundo, A. F., Haw, R., Milacic, M., Weiser, J., Wu, G., … D'eustachio, P. (2014). The Reactome pathway knowledgebase. *Nucleic Acids Research*, *42*, D472–D477. doi: 10.1093/nar/gkt1102

Dixon, J. R., Jung, I., Selvaraj, S., Shen, Y., Antosiewicz-Bourget, J. E., Lee, A. Y., … Ren, B. (2015). Chromatin architecture reorganization during stem cell differentiation. *Nature*, *518*, 331–336. doi: 10.1038/nature14222

Gao, C. H., Yu, G., & Cai, P. (2021). ggVennDiagram: An intuitive, easy-to-use, and highly customizable R package to generate venn diagram. *Frontiers in Genetics*, *12*, 706907. doi: 10.3389/fgene.2021.706907

Gel, B., Díez-Villanueva, A., Serra, E., Buschbeck, M., Peinado, M. A., & Malinverni, R. (2016). RegioneR: An R/Bioconductor package for the association analysis of genomic regions based on permutation tests. *Bioinformatics*, *32*, 289–291.

Huang, W., Loganantharaj, R., Schroeder, B., Fargo, D., & Li, L. (2013). PAVIS: A tool for peak annotation and visualization. *Bioinformatics*, *29*, 3097–3099. doi: 10.1093/bioinformatics/btt520

Kanehisa, M., Goto, S., Kawashima, S., Okuno, Y., & Hattori, M. (2004). The KEGG resource for deciphering the genome. *Nucleic Acids Research*, *32*, D277–80. doi: 10.1093/nar/gkh063

Kent, W. J., Sugnet, C. W., Furey, T. S., Roskin, K. M., Pringle, T. H., Zahler, A. M., & Haussler, A. D. (2002). The Human Genome Browser at UCSC. *Genome Research*, *12*, 996–1006. doi: 10.1101/gr.229102

Klemm, S. L., Shipony, Z., & Greenleaf, W. J. (2019). Chromatin accessibility and the regulatory epigenome. *Nature Reviews Genetics*, *20*, 207–220. doi: 10.1038/s41576-018-0089-8

Lawrence, M., Huber, W., Pagès, H., Aboyoun, P., Carlson, M., Gentleman, R., … Carey, V. J. (2013). Software for computing and annotating genomic ranges. *PloS Computational Biology*, *9*, 1–10. doi: 10.1371/journal.pcbi.1003118

Lister, R., Pelizzola, M., Dowen, R. H., Hawkins, R. D., Hon, G., Tonti-Filippini, J., … Ecker, J. R. (2009). Human DNA methylomes at base resolution show widespread epigenomic differences. *Nature*, *462*, 315–322. doi: 10.1038/nature08514

Pemberton, H., Anderton, E., Patel, H., Brookes, S., Chandler, H., Palermo, R., … Peters, G. (2014). Genome-wide co-localization of Polycomb orthologs and their effects on gene expression in human fibroblasts. *Genome Biology*, *15*, 1–16. doi: 10.1186/gb-2014-15-2-r23

Ramírez, F., Ryan, D. P., Grüning, B., Bhardwaj, V., Kilpert, F., Richter, A. S., … Manke, T. (2016). deepTools2: A next generation web server for deep-sequencing data analysis. *Nucleic Acids Research*, *44*, W160–W165. doi: 10.1093/nar/gkw257

Salmon-Divon, M., Dvinge, H., Tammoja, K., & Bertone, P. (2010). PeakAnalyzer: Genome-wide annotation of chromatin binding and modification loci. *BMC Bioinformatics*, *11*, 415. doi: 10.1186/1471-2105-11-415

Schriml, L. M., Arze, C., Nadendla, S., Chang, Y.-W. W., Mazaitis, M., Felix, V., … Kibbe, W. A. (2012). Disease ontology: A backbone for disease semantic integration. *Nucleic Acids Research*, *40*, 940–946. doi: 10.1093/nar/gkr972

Shen, L., Shao, N., Liu, X., & Nestler, E. (2014). Ngs.plot: Quick mining and visualization of next-generation sequencing data by integrating genomic databases. *BMC Genomics*, *15*(1), 284. doi: 10.1186/1471-2164-15-284

Urbanucci, A., Sahu, B., Seppälä, J., Larjo, A., Latonen, L. M., Waltering, K. K., … Visakorpi, T. (2012). Overexpression of androgen receptor enhances the binding of the receptor to the chromatin in prostate cancer. *Oncogene*, *31*, 2153–2163. doi: 10.1038/onc.2011.401

Wu, T., Hu, E., Xu, S., Chen, M., Guo, P., Dai, Z., … Yu, G. (2021). clusterProfiler 4.0: A universal enrichment tool for interpreting omics data. *Innovation*, *2*, 100141. doi: 10.1016/j.xinn.2021.100141

Yu, G., & He, Q.-Y. (2016). ReactomePA: An R/Bioconductor package for reactome pathway analysis and visualization. *Molecular Biosystems*, *12*, 477–479. doi: 10.1039/C5MB00663E

Yu, G., Wang, L. G., & He, Q. Y. (2015). ChIP seeker: An R/Bioconductor package for ChIP peak annotation, comparison and visualization. *Bioinformatics*, *31*, 2382–2383. doi: 10.1093/bioinformatics/btv145

Yu, G., Wang, L. G., Yan, G. R., & He, Q. Y. (2015). DOSE: An R/Bioconductor package for disease ontology semantic and enrichment analysis. *Bioinformatics*, *31*, 608–609. doi: 10.1093/bioinformatics/btu684

Zhao, L. Y., Song, J., Liu, Y., Song, C. X., & Yi, C. (2020). Mapping the epigenetic modifications of DNA and RNA. *Protein Cell*, *11*, 792–808. doi: 10.1007/s13238-020-00733-7

Zhu, L. J., Gazin, C., Lawson, N. D., Pagès, H., Lin, S. M., Lapointe, D. S., & Green, M. R. (2010). ChIPpeakAnno: A Bioconductor package to annotate ChIP-seq and ChIP-chip data. *BMC Bioinformatics*, *11*, 237–247. doi: 10.1186/1471-2105-11-237