ChIPseeker: an R package for ChIP peak Annotation, Comparision and Visualization

Guangchuang Yu
The University of Hong Kong

June 2, 2014

Contents

1 Introduction

Chromatin immunoprecipitation followed by high-throughput sequencing (ChIP-seq) has become standard technologies for genome wide identification of DNA-binding protein target sites. After read mappings and peak callings, the peak should be annotated to answer the biological questions. Annotation also create the possibility of integrate expression profile data to predict gene expression regulation. *ChIPseeker* was developed for annotating nearest genes and genomic features to peaks.

ChIP peak data set comparison is also very important. We can use it as an index to estimate how well biological replications are. Even more important is applying to infer cooperative regulation. If two ChIP seq data, obtained by two different binding proteins, overlap significantly, these two proteins may form a complex or have interaction in regulation chromosome remodelling or gene expression. *ChIPseeker* support statistical testing of significant overlap among ChIP seq data sets, and incorporate open access database GEO for users to compare their own dataset to those deposited in database. Protein interaction hypothesis can be generated by mining data deposited in database. Converting genome coordinations from one genome version to another is also supported, making this comparison available for different genome version and different species.

Several visualization functions are implemented to visualize the coverage of the ChIP seq data, peak annotation, average profile and heatmap of peaks binding to TSS region.

Functional enrichment analysis of the peaks can be performed by my Bioconductor packages *DOSE* , *ReactomePA*, *clusterProfiler* [1] .

```
## loading packages
require(ChIPseeker)
require(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
require(clusterProfiler)
```

2 ChIP profiling

The datasets CBX6 and CBX7 in this vignettes were downloaded from GEO (GSE40740) [2] while ARmo_0M, ARmo_1nM and ARmo_100nM were downloaded from GEO (GSE48308) [3] . *ChIPseeker* provides `readPeakFile` to load the peak and store in `GRanges` object. Most of the functions in *ChIPseeker* can accept input in peak file (bed format) or `GRanges` object.

```
files <- getSampleFiles()
print(files)

## $ARmo_0M
## [1] "/Volumes/DATA/opt/R-devel/R.framework/Versions/3.2/Resources/library/ChIPse
##
## $ARmo_1nM
## [1] "/Volumes/DATA/opt/R-devel/R.framework/Versions/3.2/Resources/library/ChIPse
##
## $ARmo_100nM
## [1] "/Volumes/DATA/opt/R-devel/R.framework/Versions/3.2/Resources/library/ChIPse
##
## $CBX6_BF
## [1] "/Volumes/DATA/opt/R-devel/R.framework/Versions/3.2/Resources/library/ChIPse
##
## $CBX7_BF
## [1] "/Volumes/DATA/opt/R-devel/R.framework/Versions/3.2/Resources/library/ChIPse

peak <- readPeakFile(files[[4]])
peak

## GRanges with 1331 ranges and 2 metadata columns:
##          seqnames                 ranges strand   |             V4        V5
##             <Rle>              <IRanges>  <Rle>   |       <factor> <numeric>
##      [1]     chr1     [ 815092,  817883]      *   |    MACS_peak_1     295.8
##      [2]     chr1     [1243287, 1244338]      *   |    MACS_peak_2      63.2
##      [3]     chr1     [2979976, 2981228]      *   |    MACS_peak_3     100.2
##      [4]     chr1     [3566181, 3567876]      *   |    MACS_peak_4     558.9
##      [5]     chr1     [3816545, 3818111]      *   |    MACS_peak_5      57.6
##      ...      ...                    ...    ... ...            ...       ...
##   [1327]     chrX [135244782, 135245821]      *   | MACS_peak_1327      55.5
##   [1328]     chrX [139171963, 139173506]      *   | MACS_peak_1328     270.2
##   [1329]     chrX [139583953, 139586126]      *   | MACS_peak_1329     918.7
##   [1330]     chrX [139592001, 139593238]      *   | MACS_peak_1330     210.9
##   [1331]     chrY [ 13845133,  13845777]      *   | MACS_peak_1331      58.4
##   ---
##   seqlengths:
##     chr1 chr10 chr11 chr12 chr13 chr14 ...  chr6  chr7  chr8  chr9  chrX  chrY
##       NA    NA    NA    NA    NA    NA ...    NA    NA    NA    NA    NA    NA
```

2.1 ChIP peaks over Chromosomes

After peak calling, we would like to know the peak locations over the whole genome, `plotChrCov` function calculates the coverage of peak regions over chromosomes and generate a figure to visualize.

```
plotChrCov(peak, weightCol = "V5")
```

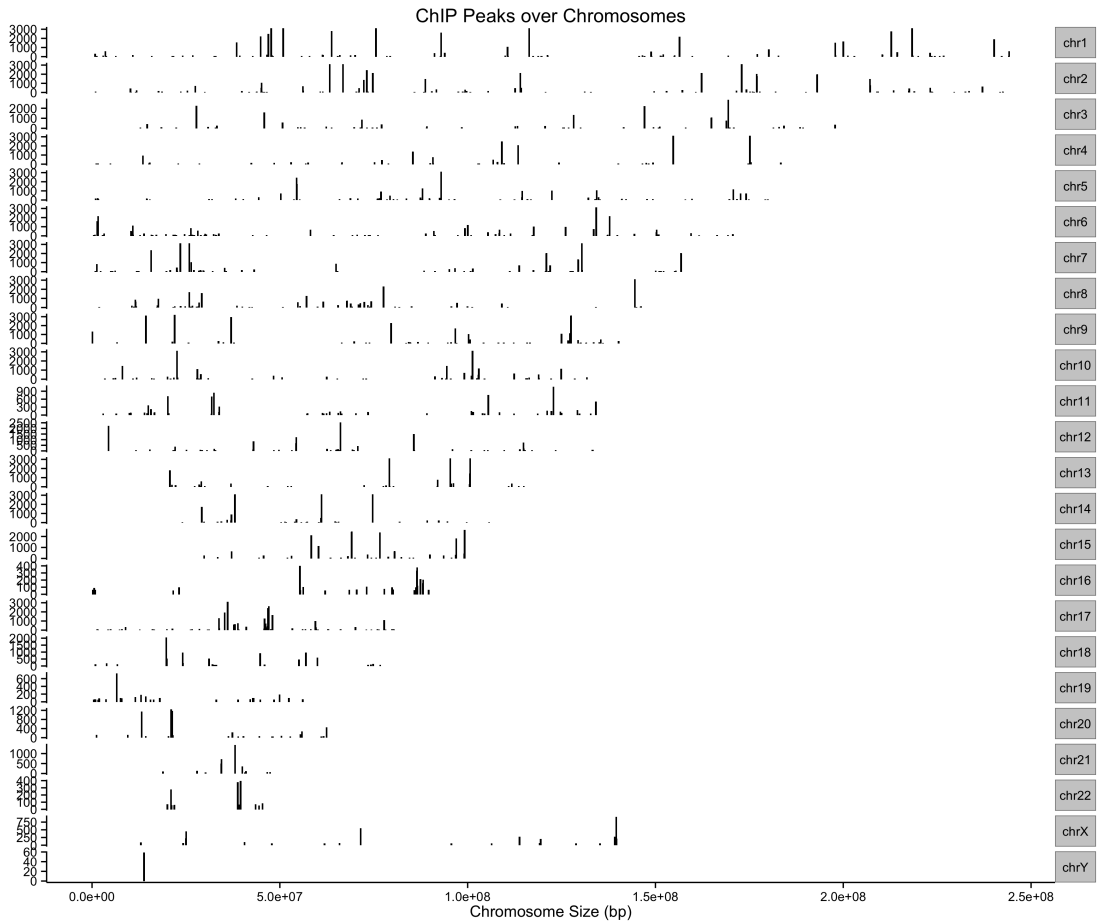

Figure 1: ChIP peaks over Chromosomes

2.2 Profile of ChIP peaks binding to TSS regions

First of all, for calculate the profile of ChIP peaks binding to TSS regions, we should prepare the TSS regions, which are defined as the flanking sequence of the TSS sites. Then align the peaks that are mapping to these regions, and generate the tagMatrix.

```
## promoter <- getPromoters(TranscriptDb=txdb,
## upstream=3000, downstream=3000) tagMatrix <-
## getTagMatrix(peak, windows=promoter) to speed up
## the compilation of this vignettes, we use a
## precalculated tagMatrix
data("tagMatrixList")
tagMatrix <- tagMatrixList[[4]]
```

In the above code, you should notice that tagMatrix is not restricted to TSS regions. The regions can be other types that defined by the user.

2.2.1 Heatmap of ChIP binding to TSS regions

```
tagHeatmap(tagMatrix, xlim = c(-3000, 3000), color = "red")
```

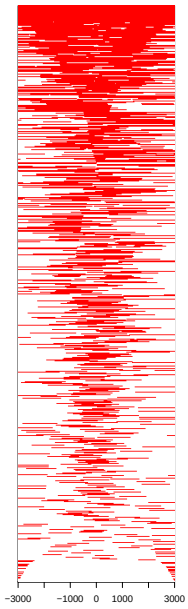

Figure 2: Heatmap of ChIP peaks binding to TSS regions

ChIPseeker provide a one step function to generate this figure from bed file. The following function will generate the same figure as above.

```
peakHeatmap(files[[4]], TranscriptDb = txdb, upstream = 3000,
    downstream = 3000, color = "red")
```

2.2.2 Average Profile of ChIP peaks binding to TSS region

```
plotAvgProf(tagMatrix, xlim = c(-3000, 3000), xlab = "Genomic Region (5'->3')",
    ylab = "Read Count Frequency")
```

The function `plotAvgProf2` provide a one step from bed file to average profile plot. The following command will generate the same figure as shown above.

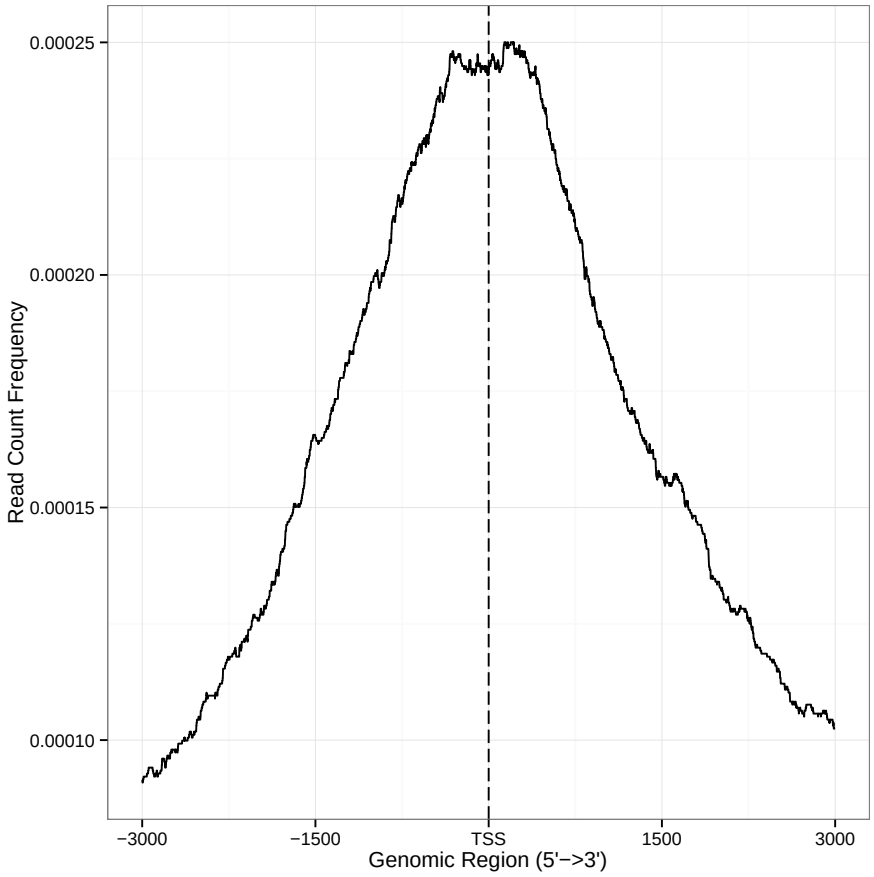

Figure 3: Average Profile of ChIP peaks binding to TSS region

```
plotAvgProf2(files[[4]], TranscriptDb = txdb, upstream = 3000,
    downstream = 3000, xlab = "Genomic Region (5'->3')",
    ylab = "Read Count Frequency")
```

3 Peak Annotation

```
peakAnno <- annotatePeak(files[[4]], tssRegion = c(-3000,
    3000), as = "GRanges", TranscriptDb = txdb, annoDb = "org.Hs.eg.db")
```

```
## >> loading peak file...  2014-06-02 23:51:08
## >> preparing features information...  2014-06-02 23:51:08
## >> identifying nearest features...  2014-06-02 23:51:22
## >> calculating distance from peak to TSS...  2014-06-02 23:51:22
## >> assigning genomic annotation...  2014-06-02 23:51:22
## >> adding gene annotation...  2014-06-02 23:52:21
## >> assigning chromosome lengths  2014-06-02 23:52:25
## >> done...  2014-06-02 23:52:25
```

```
head(peakAnno)
```

```
## GRanges with 6 ranges and 13 metadata columns:
##       seqnames             ranges strand |          V4        V5
##          <Rle>          <IRanges>  <Rle> |    <factor> <numeric>
##   [1]     chr1 [ 815092,  817883]      * | MACS_peak_1     295.8
##   [2]     chr1 [1243287, 1244338]      * | MACS_peak_2      63.2
##   [3]     chr1 [2979976, 2981228]      * | MACS_peak_3     100.2
##   [4]     chr1 [3566181, 3567876]      * | MACS_peak_4     558.9
##   [5]     chr1 [3816545, 3818111]      * | MACS_peak_5      57.6
##   [6]     chr1 [6304864, 6305704]      * | MACS_peak_6      54.6
##              annotation  geneChr geneStart   geneEnd geneLength geneStrand
##             <character> <factor> <integer> <integer>  <integer>   <factor>
##   [1] Distal Intergenic     chr1    803451    812182       8732          -
##   [2]  Promoter (<=1kb)     chr1   1227764   1244989      17226          -
##   [3]            3' UTR     chr1   2976181   2984289       8109          -
##   [4]  Promoter (2-3kb)     chr1   3569129   3652765      83637          +
##   [5]            5' UTR     chr1   3773845   3801993      28149          +
##   [6]  Promoter (<=1kb)     chr1   6304252   6305638       1387          +
##            geneId distanceToTSS         ENSEMBL      SYMBOL
##       <character>     <integer>     <character> <character>
##   [1]      284593          5701 ENSG00000230368      FAM41C
##   [2]      116983           651 ENSG00000131584       ACAP3
##   [3]      440556          3061 ENSG00000177133   LINC00982
##   [4]        7161         -2948 ENSG00000078900        TP73
##   [5]        1677        -42700 ENSG00000169598        DFFB
##   [6]      390992          -612 ENSG00000173673        HES3
##                                                                  GENENAM
##                                                                <character
##   [1]                                family with sequence similarity 41, member
##   [2]                   ArfGAP with coiled-coil, ankyrin repeat and PH domains
##   [3]                                  long intergenic non-protein coding RNA 98
##   [4]                                                       tumor protein p7
##   [5] DNA fragmentation factor, 40kDa, beta polypeptide (caspase-activated DNase
##   [6]                                      hes family bHLH transcription factor
##   ---
##   seqlengths:
##         chr1     chr10     chr11     chr12 ...      chr9      chrX      chrY
##    249250621 135534747 135006516 133851895 ... 141213431 155270560  59373566
```

Peak Annotation is performed by `annotatePeak`. User can define TSS (transcription start site) region, by default TSS is defined from -3kb to +3kb. The argument *as* can be one of "GRanges", "data.fram" and "txt" to specify the output format return by `annotatePeak`. If *as* is set to "txt", the output will save to a TXT file with name suffix by anno.txt.

`TranscriptDb` object contained transcript-related features of a particular genome. Bioconductor provides several package that containing `TranscriptDb` object of

model organisms with multiple commonly used genome version, for instance *TxDb.Hsapiens.UCSC.hg19.knownGene* and *TxDb.Hsapiens.UCSC.hg18.knownGene* for human genome hg19 and hg18, *TxDb.Mmusculus.UCSC.mm10.knownGene* and *TxDb.Mmusculus.UCSC.mm9.knownGene* for mouse genome mm10 and mm9, etc. User can also prepare their own `TranscriptDb` object by retrieving information from UCSC Genome Bioinformatics and BioMart data resources by R function `makeTranscriptDbFromBiomart` and `makeTranscriptDbFromUCSC`. TranscriptDb object should be passed for peak annotation.

All the peak information contained in peakfile will be retained in the output of `annotatePeak`. The position and strand information of nearest genes are reported. The distance from peak to the TSS of its nearest gene is also reported. The genomic region of the peak is reported in annotation column. Since some annotation may overlap, *ChIPseeker* adopted the following priority in genomic annotation.

- Promoter
- 5' UTR
- 3' UTR
- Exon
- Intron
- Downstream
- Intergenic

Downstream is defined as the downstream of gene end.

`annotatePeak` report detail information when the annotation is Exon or Intron, for instance "Exon (uc002sbe.3/9736, exon 69 of 80)", means that the peak is overlap with an Exon of transcript uc002sbe.3, and the corresponding Entrez gene ID is 9736 (Transcripts that belong to the same gene ID may differ in splice events), and this overlaped exon is the 69th exon of the 80 exons that this transcript uc002sbe.3 prossess.

Parameter annoDb is optional, if provided, extra columns including SYMBOL, GENENAME, ENSEMBL/ENTREZID will be added. The geneId column in annotation output will be consistent with the geneID in TranscriptDb. If it is ENTREZID, ENSEMBL will be added if annoDb is provided, while if it is ENSEMBL ID, ENTREZID will be added.

3.1 Visualize Genomic Annotation

To annotate the location of a given peak in terms of genomic features, `annotatePeak` assigns peaks to genomic annotation in "annotation" column of the output, which

includes whether a peak is in the TSS, Exon, 5' UTR, 3' UTR, Intronic or Intergenic. Many researchers are very interesting in these annotations. TSS region can be defined by user and `annotatePeak` output in details of which exon/intron of which genes as illustrated in previous section.

Pie and Bar plot are supported to visualize the genomic annotation.

```
plotAnnoPie(peakAnno)
```

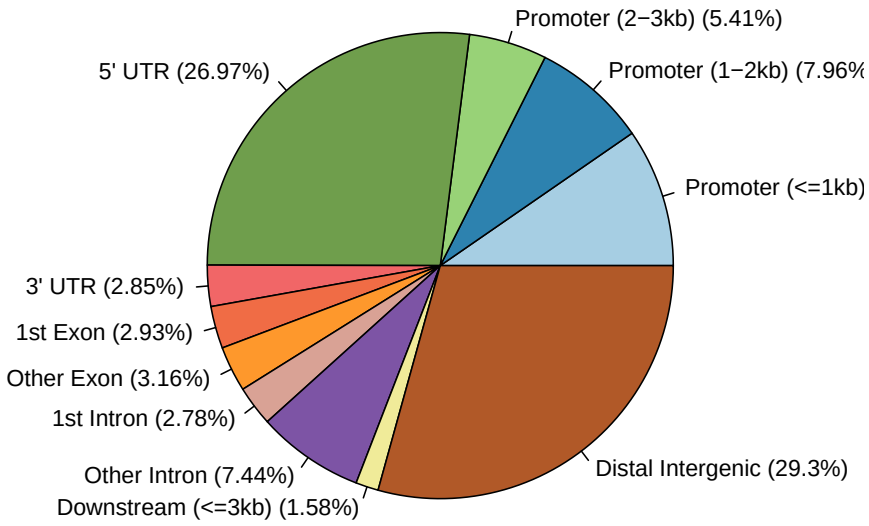

Figure 4: Genomic Annotation by pieplot

```
plotAnnoBar(peakAnno)
```

3.2 Visualize distribution of TF-binding loci relative to TSS

The distance from the peak (binding site) to the TSS of the nearest gene is calculated by `annotatePeak` and reported in the output. We provide `plotDistToTSS` to calculate the percentage of binding sites upstream and downstream from the TSS of the nearest genes, and visualize the distribution.

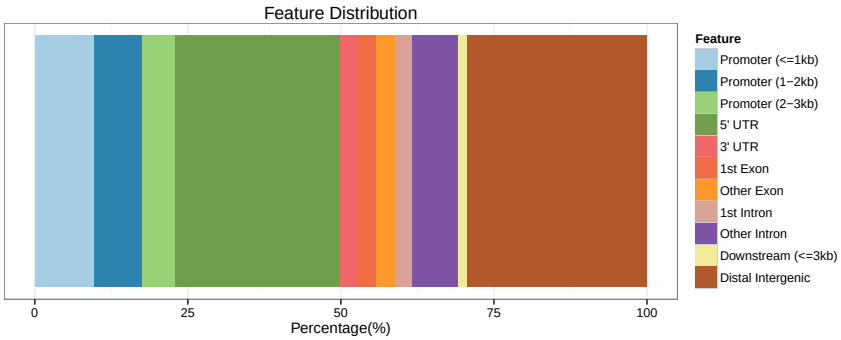

Figure 5: Genomic Annotation by barplot

```
plotDistToTSS(peakAnno, title = "Distribution of transcription factor-binding loci

## Warning:  Stacking not well defined when ymin != 0
```

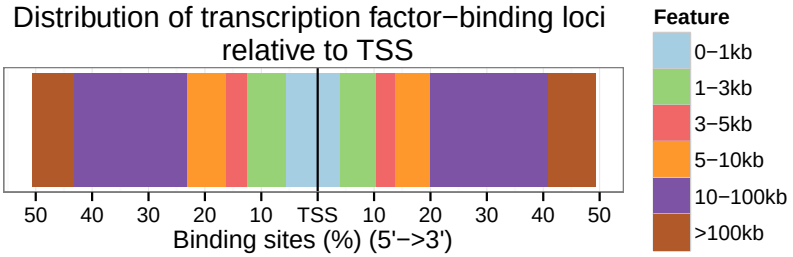

Figure 6: Distribution of Binding Sites

4 Functional enrichment analysis

Once we have obtained the annotated nearest genes, we can perform functional enrichment analysis to identify predominant biological themes among these genes by incorporating biological knowledge provided by biological ontologies. For instance, Gene Ontology (GO) [4] annotates genes to biological processes, molecular functions, and cellular components in a directed acyclic graph structure, Kyoto Encyclopedia of Genes and Genomes (KEGG) [5] annotates genes to pathways, Disease Ontology (DO) [6] annotates genes with human disease association, and Reactome [7] annotates gene to pathways and reactions.

Enrichment analysis is a widely used approach to identify biological themes. I have developed several Bioconductor packages for investigating whether the number of selected genes associated with a particular biological term is larger than expected, including *DOSE* for Disease Ontology, *ReactomePA* for reactome pathway, *clusterProfiler* [1] for Gene Ontology and KEGG enrichment analysis.

```
require(clusterProfiler)
bp <- enrichGO(unlist(peakAnno$geneId), ont = "BP",
    readable = TRUE)

## Loading required package:  GO.db

head(summary(bp))

##                    ID                          Description GeneRatio
## GO:0008150 GO:0008150                   biological_process   734/734
## GO:0007275 GO:0007275  multicellular organismal development   366/734
## GO:0044767 GO:0044767 single-organism developmental process   393/734
## GO:0032502 GO:0032502                 developmental process   395/734
## GO:0048731 GO:0048731                    system development   321/734
## GO:0048513 GO:0048513                     organ development   259/734
##                BgRatio   pvalue p.adjust   qvalue
## GO:0008150 15034/18207 3.65e-63 5.70e-60 2.71e-60
## GO:0007275  4274/18207 6.95e-57 5.43e-54 2.59e-54
## GO:0044767  4848/18207 3.86e-56 2.01e-53 9.58e-54
## GO:0032502  4899/18207 6.76e-56 2.64e-53 1.26e-53
## GO:0048731  3530/18207 1.42e-53 4.43e-51 2.11e-51
## GO:0048513  2489/18207 5.42e-52 1.41e-49 6.72e-50
##
## GO:0008150 TCF24/FRAT1/CDH6/LOC100506422/CASP12/CDH8/EDIL3/AASS/OLIG2/SLC17A2/SP
## GO:0007275
## GO:0044767
## GO:0032502
## GO:0048731
## GO:0048513
##            Count
## GO:0008150   734
## GO:0007275   366
## GO:0044767   393
## GO:0032502   395
## GO:0048731   321
## GO:0048513   259
```

More information can be found in the vignettes of Bioconductor packages *DOSE* , *ReactomePA*, *clusterProfiler* [1], which also provide several methods to visualize enrichment results. The *clusterProfiler* package is designed for comparing and visualizing functional profiles among gene clusters, and can directly applied to compare biological themes at GO, DO, KEGG, Reactome perspective.

5 ChIP peak data set comparison

5.1 Profile of several ChIP peak data binding to TSS region

Function `plotAvgProf` and `tagHeatmap` can accept a list of `tagMatrix` and visualize profile or heatmap among several ChIP experiments, while `plotAvgProf2` and `peakHeatmap` can accept a list of bed files and perform the same task in one step.

5.1.1 Average profiles

```
## promoter <- getPromoters(TranscriptDb=txdb,
## upstream=3000, downstream=3000) tagMatrixList <-
## lapply(files, getTagMatrix, windows=promoter) to
## speed up the compilation of this vigenettes, we
## load a precaculated tagMatrixList
data("tagMatrixList")
plotAvgProf(tagMatrixList, xlim = c(-3000, 3000))
```

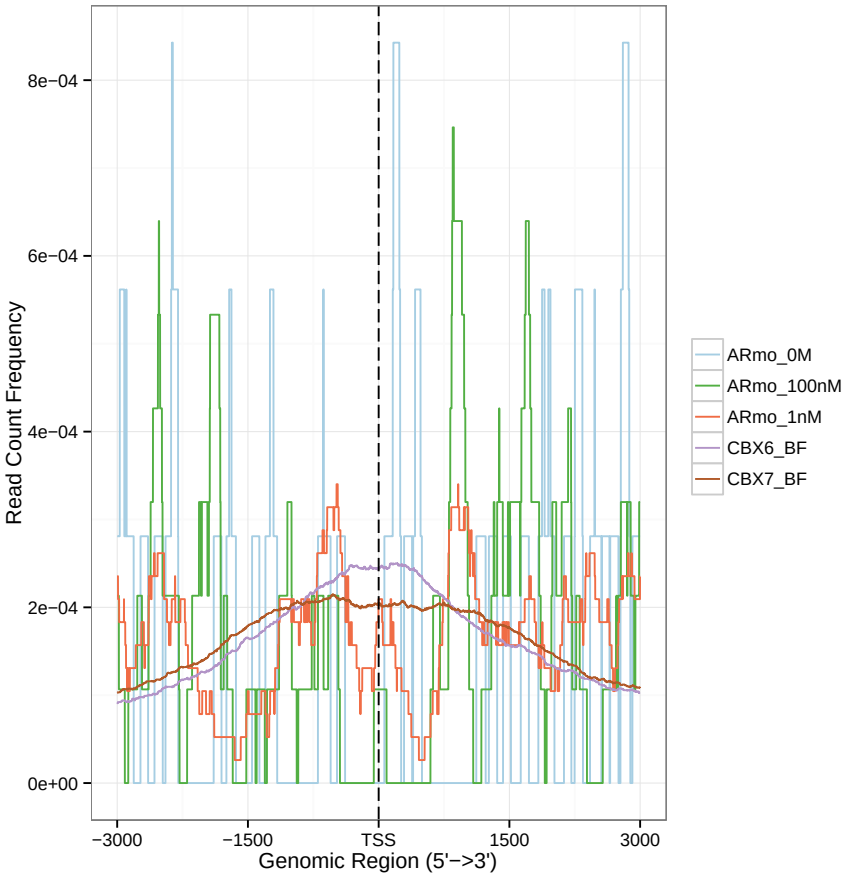

Figure 7: Average Profiles of ChIP peaks among different experiments

5.1.2 Peak heatmaps

```
tagHeatmap(tagMatrixList, xlim = c(-3000, 3000), color = NULL)
```

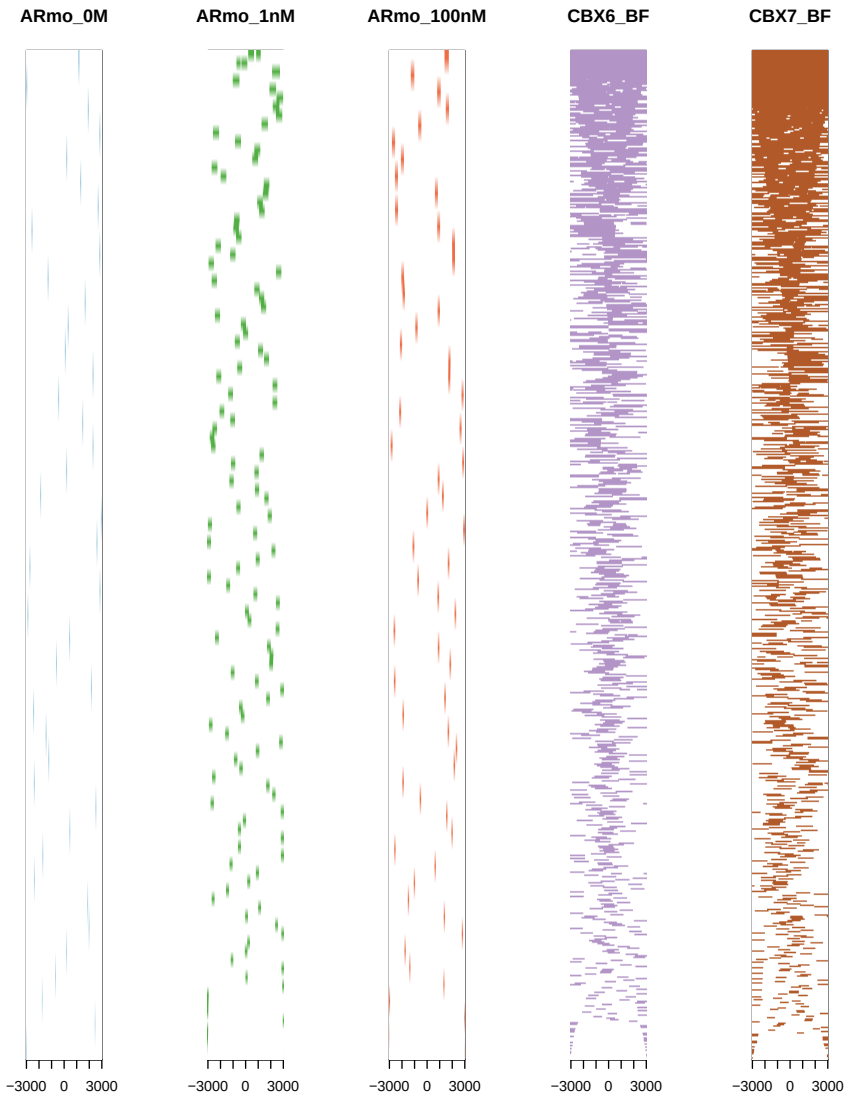

Figure 8: Heatmap of ChIP peaks among different experiments

5.2 ChIP peak annotation comparision

The `plotAnnoBar` and `plotDistToTSS` can also accept input of a named list of annotated peaks (output of `annotatePeak`).

```
peakAnnoList <- lapply(files, annotatePeak, TranscriptDb = txdb,
    tssRegion = c(-3000, 3000), verbose = FALSE)
```

We can use `plotAnnoBar` to comparing their genomic annotation.

```
plotAnnoBar(peakAnnoList)
```

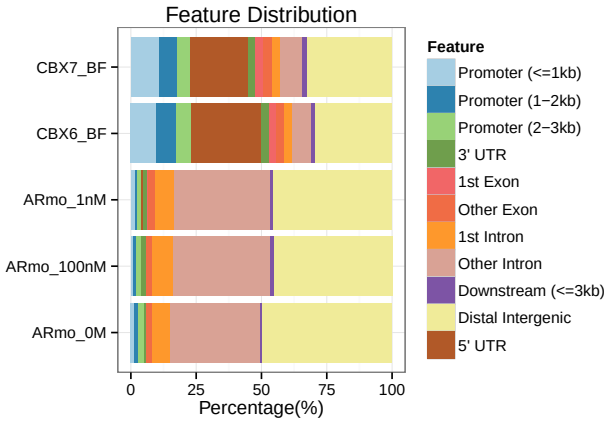

Figure 9: Genomic Annotation among different ChIPseq data

R function `plotDistToTSS` can use to comparing distance to TSS profiles among ChIPseq data.

```
plotDistToTSS(peakAnnoList)
```

```
## Warning:  Stacking not well defined when ymin != 0
```

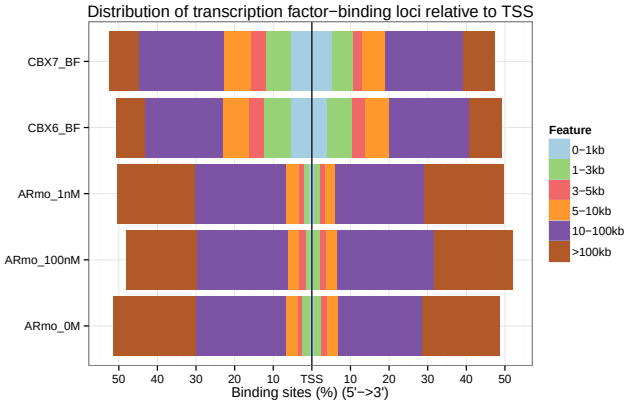

Figure 10: Distribution of Binding Sites among different ChIPseq data

5.3 Functional profiles comparison

As shown in section 4, the annotated genes can analyzed by *clusterProfiler*, *DOSE* and *ReactomePA* for Gene Ontology, KEGG, Disease Ontology and Reactome Pathway enrichment analysis.

The *clusterProfiler* package provide `compareCluster` function for comparing biological themes among gene clusters, and can be easily adopted to compare different ChIP peak experiments.

```
genes = lapply(peakAnnoList, function(i) i$geneId)
names(genes) = sub("_", "\n", names(genes))
compGO <- compareCluster(geneCluster = genes, fun = "enrichGO",
    ont = "MF", organism = "human", pvalueCutoff = 0.05,
    pAdjustMethod = "BH")
plot(compGO, showCategory = 20, title = "Molecular Function Enrichment")
```

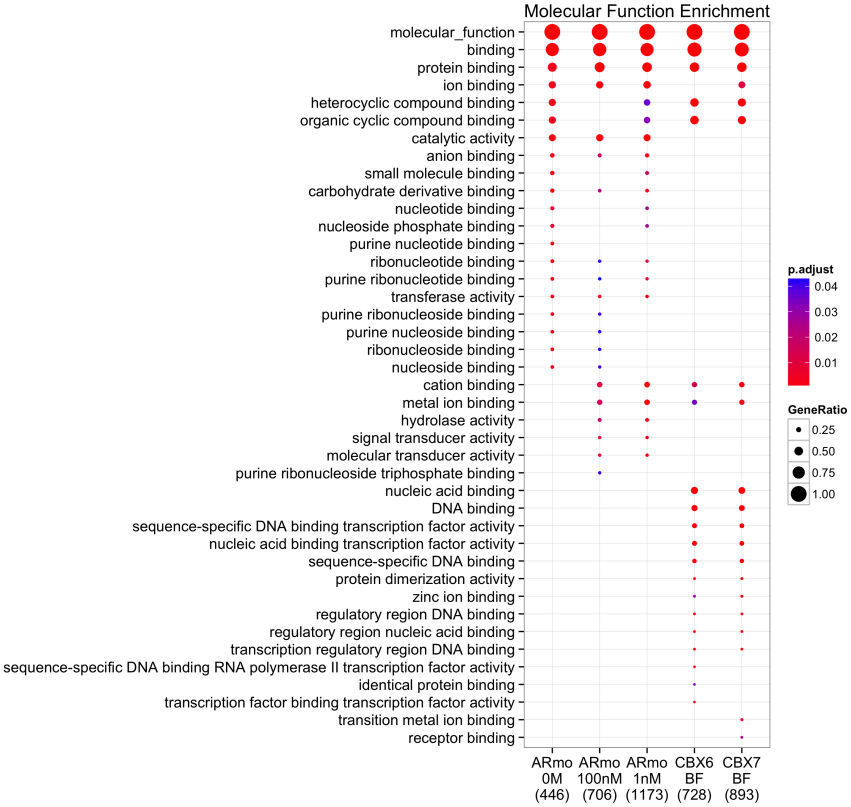

Figure 11: Compare Biological themes among different experiments

5.4 Overlap of peaks and annotated genes

User may want to compare the overlap peaks of replicate experiments or from different experiments. *ChIPseeker* provides `peak2GRanges` that can read peak file and stored in GRanges object. Several files can be read simultaneously using lapply, and then passed to `vennplot` to calculate their overlap and draw venn plot.

`vennplot` accept a list of object, can be a list of GRanges or a list of vector. Here,

I will demonstrate using `vennplot` to visualize the overlap of the nearest genes stored in peakAnnoList.

```
genes = lapply(peakAnnoList, function(i) unlist(i$geneId))
vennplot(genes)
```

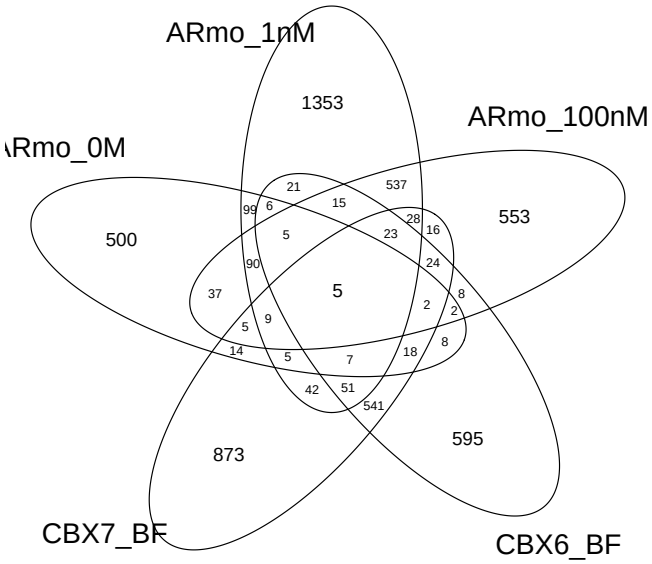

Figure 12: Overlap of annotated genes

6 Statistical testing of ChIP seq overlap

Overlap is very important, if two ChIP experiment by two different proteins overlap in a large fraction of their peaks, they may cooperative in regulation. Calculating the overlap is only touch the surface. *ChIPseeker* implemented statistical methods to measure the significance of the overlap.

6.1 Shuffle genome coordination

```
p <- GRanges(seqnames = c("chr1", "chr3"), ranges = IRanges(start = c(1,
    100), end = c(50, 130)))
shuffle(p, TranscriptDb = txdb)
```

```
## GRanges with 2 ranges and 0 metadata columns:
##       seqnames                ranges strand
##          <Rle>             <IRanges>  <Rle>
##   [1]     chr1 [ 79338456,  79338506]      *
##   [2]     chr3 [138178057, 138178088]      *
##   ---
##   seqlengths:
##    chr1 chr3
##      NA   NA
```

We implement the `shuffle` function to randomly permute the genomic locations of ChIP peaks defined in a genome which stored in `TranscriptDb` object.

6.2 Peak overlap enrichment analysis

With the ease of this `shuffle` method, we can generate thousands of random ChIP data and calculate the background null distribution of the overlap among ChIP data sets.

```
enrichPeakOverlap(queryPeak = files[[5]], targetPeak = unlist(files[1:4]),
    TranscriptDb = txdb, pAdjustMethod = "BH", nShuffle = 50,
    chainFile = NULL)
```

```
##                                                      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      0.94    0.940
## ARmo_1nM     0.20    0.400
## ARmo_100nM   0.50    0.667
## CBX6_BF      0.00    0.000
```

Parameter `queryPeak` is the query ChIP data, while `targetPeak` is bed file name or a vector of bed file names from comparison; `nShuffle` is the number to shuffle the peaks in `targetPeak`. To speed up the compilation of this vignettes, we only set `nShuffle` to 50 as an example for only demonstration. User should set the number to 1000 or above for more robust result. Parameter `chainFile` are chain file name for mapping the `targetPeak` to the genome version consistent with `queryPeak` when their genome version are different. This creat the possibility of comparison among different genome version and cross species.

In the output, `qSample` is the name of `queryPeak` and `qLen` is the the number of peaks in `queryPeak`. `N_OL` is the number of overlap between `queryPeak` and `targetPeak`.

7 Data Mining with ChIP seq data deposited in GEO

There are many ChIP seq data sets that have been published and deposited in GEO database. We can compare our own dataset to those deposited in GEO to search for significant overlap data. Significant overlap of ChIP seq data by different binding proteins may be used to infer cooperative regulation and thus can be used to generate hypotheses.

We collect about 15,000 bed files deposited in GEO, user can use `getGEOspecies` to get a summary based on speices.

7.1 GEO data collection

```
getGEOspecies()
```

```
##                                    species Freq
## 1                          Aedes aegypti   11
## 2                              Anabaena    6
## 3                    Anolis carolinensis    2
## 4                         Apis mellifera    5
## 5                Apis mellifera scutellata    1
## 6                       Arabidopsis lyrata    4
## 7                    Arabidopsis thaliana   65
## 8                    Atelerix albiventris    2
## 9                           Brassica rapa    8
## 10                Caenorhabditis elegans  164
## 11                     Candida albicans   25
## 12                  Candida dubliniensis   20
## 13                Canis lupus familiaris    7
## 14                  Chlorocebus aethiops    2
## 15                    Cleome hassleriana    1
## 16                         Columba livia    6
## 17                      Crassostrea gigas    1
## 18                Cryptococcus neoformans   39
## 19                           Danio rerio  122
## 20                Drosophila melanogaster  551
## 21               Drosophila pseudoobscura    7
## 22                    Drosophila simulans    9
## 23                     Drosophila virilis    2
## 24                      Drosophila yakuba    8
```

```
## 25                            Equus caballus    1
## 26                          Escherichia coli    1
## 27                 Escherichia coli BW25113    4
## 28                    Escherichia coli K-12    2
## 29 Escherichia coli str. K-12 substr. MG1655    8
## 30                             Gallus gallus   43
## 31             Geobacter sulfurreducens PCA    3
## 32                           Gorilla gorilla    2
## 33                         Histophilus somni    1
## 34                              Homo sapiens 7347
## 35                      Human herpesvirus 6B    2
## 36                       Human herpesvirus 8    6
## 37                    Legionella pneumophila    5
## 38                    Leishmania amazonensis    4
## 39                          Leishmania major    2
## 40                     Leishmania tarentolae   15
## 41                             Macaca mulatta   28
## 42                     Monodelphis domestica    4
## 43                Moraxella catarrhalis O35E    6
## 44                              Mus musculus 5558
## 45                Mus musculus x Mus spretus    1
## 46                Mycobacterium tuberculosis    2
## 47                           Myotis brandtii   15
## 48                      Nematostella vectensis   23
## 49                  Ornithorhynchus anatinus    5
## 50                              Oryza sativa   23
## 51                           Oryzias latipes    2
## 52                           Pan troglodytes    3
## 53                 Plasmodium falciparum 3D7   29
## 54                 Pseudomonas putida KT2440    2
## 55                       Pyrococcus furiosus    4
## 56                         Rattus norvegicus   38
## 57                Rhodopseudomonas palustris    6
## 58         Rhodopseudomonas palustris CGA009    3
## 59                  Saccharomyces cerevisiae  360
## 60                   Saccharomyces paradoxus    8
## 61             Schizosaccharomyces japonicus    2
## 62                 Schizosaccharomyces pombe   88
## 63                    Schmidtea mediterranea    7
## 64             Streptomyces coelicolor A3(2)    6
## 65                               Sus scrofa   17
## 66                          Tupaia chinensis    7
## 67             Xenopus (Silurana) tropicalis   62
## 68                                Zea mays   54
```

The summary can also based on genome version as illustrated below:

```
getGEOgenomeVersion()
```

```
##                          organism genomeVersion Freq
## 1             Anolis carolinensis       anoCar2    2
## 2          Caenorhabditis elegans          ce10    4
## 3          Caenorhabditis elegans           ce6   64
## 4                     Danio rerio       danRer6    6
## 5                     Danio rerio       danRer7   40
## 6         Drosophila melanogaster           dm3  340
## 7                   Gallus gallus       galGal3   20
## 8                   Gallus gallus       galGal4   15
## 9                    Homo sapiens          hg18 1936
## 10                   Homo sapiens          hg19 4948
## 11                   Mus musculus          mm10   21
## 12                   Mus musculus           mm8  465
## 13                   Mus musculus           mm9 4543
## 14          Monodelphis domestica       monDom5    4
## 15                Macaca mulatta       rheMac2   24
## 16      Saccharomyces cerevisiae       sacCer2  141
## 17      Saccharomyces cerevisiae       sacCer3  100
## 18                   Sus scrofa       susScr2   17
## 19 Xenopus (Silurana) tropicalis       xenTro3    3
```

User can access the detail information by `getGEOInfo`, for each genome version.

```
hg19 <- getGEOInfo(genome = "hg19", simplify = TRUE)
head(hg19)
```

```
##     series_id        gsm     organism
## 111  GSE16256  GSM521889 Homo sapiens
## 112  GSE16256  GSM521887 Homo sapiens
## 113  GSE16256  GSM521883 Homo sapiens
## 114  GSE16256 GSM1010966 Homo sapiens
## 115  GSE16256  GSM896166 Homo sapiens
## 116  GSE16256  GSM910577 Homo sapiens
##
## 111          Reference Epigenome: ChIP-Seq Analysis of H3K27me3 in IMR90 Cells;
## 112            Reference Epigenome: ChIP-Seq Analysis of H3K27ac in IMR90 Cells;
## 113            Reference Epigenome: ChIP-Seq Analysis of H3K14ac in IMR90 Cells;
## 114                      polyA RNA sequencing of STL003 Pancreas Cultured Cells;
## 115          Reference Epigenome: ChIP-Seq Analysis of H4K8ac in hESC H1 Cells;
## 116 Reference Epigenome: ChIP-Seq Analysis of H3K4me1 in Human Spleen Tissue; re
##
## 111         ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM521nnn/GSM521889/suppl/GSM
## 112          ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM521nnn/GSM521887/suppl/GS
## 113          ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM521nnn/GSM521883/suppl/GS
## 114 ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM1010nnn/GSM1010966/suppl/GSM101096
```

```
## 115              ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM896nnn/GSM896166/supp
## 116       ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM910nnn/GSM910577/suppl/GSM91
##     genomeVersion pubmed_id
## 111          hg19  19829295
## 112          hg19  19829295
## 113          hg19  19829295
## 114          hg19  19829295
## 115          hg19  19829295
## 116          hg19  19829295
```

If `simplify` is set to `FALSE`, extra information including `source_name`, `extract_protocol`, `description`, `data_processing`, and `submission_date` will be incorporated.

7.2 Download GEO ChIP data sets

ChIPseeker provide function `downloadGEObedFiles` to download all the bed files of a particular genome.

```
downloadGEObedFiles(genome = "hg19", destDir = "hg19")
```

Or a vector of GSM accession number by `downloadGSMbedFiles`.

```
gsm <- hg19$gsm[sample(nrow(hg19), 10)]
downloadGSMbedFiles(gsm, destDir = "hg19")
```

7.3 Overlap significant testing

After download the bed files from GEO, we can pass them to `enrichPeakOverlap` for testing the significant of overlap. Parameter `targetPeak` can be the folder, e.g. hg19, that containing bed files. `enrichPeakOverlap` will parse the folder and compare all the bed files. It is possible to test the overlap with bed files that are mapping to different genome or different genome versions, `enrichPeakOverlap` provide a parameter `chainFile` that can pass a chain file and liftOver the `targetPeak` to the genome version consistent with `queryPeak`. Signifcant overlap can be use to generate hypothesis of cooperative regulation.By mining the data deposited in GEO, we can identify some putative complex or interacted regulators in gene expression regulation or chromsome remodelling for further validation.

8 Session Information

The version number of R and packages loaded for generating the vignette were:

- R Under development (unstable) (2014-05-13 r65592), `x86_64-apple-darwin13.1.0`
- Locale: `C/UTF-8/C/C/C/C`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: AnnotationDbi 1.27.7, Biobase 2.25.0, BiocGenerics 0.11.2, ChIPseeker 1.1.6, DBI 0.2-7, GO.db 2.14.0, GenomeInfoDb 1.1.6, GenomicFeatures 1.17.9, GenomicRanges 1.17.17, IRanges 1.99.15, RSQLite 0.11.4, S4Vectors 0.0.7, TxDb.Hsapiens.UCSC.hg19.knownGene 2.14.0, XVector 0.5.6, clusterProfiler 1.13.1, ggplot2 1.0.0, knitr 1.6, org.Hs.eg.db 2.14.0
- Loaded via a namespace (and not attached): BBmisc 1.6, BatchJobs 1.2, BiocParallel 0.7.2, Biostrings 2.33.9, DO.db 2.8.0, DOSE 2.3.1, GOSemSim 1.23.0, GenomicAlignments 1.1.11, KEGG.db 2.14.0, KernSmooth 2.23-12, MASS 7.3-33, Matrix 1.1-3, RColorBrewer 1.0-5, RCurl 1.95-4.1, Rcpp 0.11.1, Rsamtools 1.17.18, XML 3.98-1.1, biomaRt 2.21.0, bitops 1.0-6, brew 1.0-6, caTools 1.17, codetools 0.2-8, colorspace 1.2-4, digest 0.6.4, evaluate 0.5.5, fail 1.2, foreach 1.4.2, formatR 0.10, gdata 2.13.3, gplots 2.13.0, grid 3.2.0, gtable 0.1.2, gtools 3.4.1, highr 0.3, igraph 0.7.1, iterators 1.0.7, labeling 0.2, lattice 0.20-29, munsell 0.4.2, plyr 1.8.1, proto 0.3-10, qvalue 1.39.1, reshape2 1.4, rtracklayer 1.25.9, scales 0.2.4, sendmailR 1.1-2, stats4 3.2.0, stringr 0.6.2, tools 3.2.0, zlibbioc 1.11.1

References

[1] Guangchuang Yu, Li-Gen Wang, Yanyan Han, and Qing-Yu He. clusterProfiler: an r package for comparing biological themes among gene clusters. *OMICS: A Journal of Integrative Biology*, 16(5):284–287, May 2012.

[2] Helen Pemberton, Emma Anderton, Harshil Patel, Sharon Brookes, Hollie Chandler, Richard Palermo, Julie Stock, Marc Rodriguez-Niedenfhr, Tomas Racek, Lucas de Breed, Aengus Stewart, Nik Matthews, and Gordon Peters. Genome-wide co-localization of polycomb orthologs and their effects on gene expression in human fibroblasts. 15(2):R23. PMID: 24485159.

[3] A Urbanucci, B Sahu, J Seppl, A Larjo, L M Latonen, K K Waltering, T L J Tammela, R L Vessella, H Lhdesmki, O A Jnne, and T Visakorpi. Overexpression of androgen receptor enhances the binding of the receptor to the chromatin in prostate cancer. 31(17):2153–2163. PMID: 21909140.

[4] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin

Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene ontology: tool for the unification of biology. 25:25–29.

[5] Minoru Kanehisa, Susumu Goto, Shuichi Kawashima, Yasushi Okuno, and Masahiro Hattori. The KEGG resource for deciphering the genome. 32:D277–D280. PMID: 14681412.

[6] L. M. Schriml, C. Arze, S. Nadendla, Y.-W. W. Chang, M. Mazaitis, V. Felix, G. Feng, and W. A. Kibbe. Disease ontology: a backbone for disease semantic integration. 40:D940–D946.

[7] D. Croft, A. F. Mundo, R. Haw, M. Milacic, J. Weiser, G. Wu, M. Caudy, P. Garapati, M. Gillespie, M. R. Kamdar, B. Jassal, S. Jupe, L. Matthews, B. May, S. Palatnik, K. Rothfels, V. Shamovsky, H. Song, M. Williams, E. Birney, H. Hermjakob, L. Stein, and P. D'Eustachio. The reactome pathway knowledgebase. 42:D472–D477.