Contents

1 Introduction 2

2 Quick start 2

3 Examples of using ChIPpeakAnno 3

3.1 Task 1: Find the nearest feature such as gene and the distance to the feature such as the transcription start site (TSS) of the nearest gene . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
1 Introduction

Chromatin immunoprecipitation (ChIP) followed by high-throughput tag sequencing (ChIP-seq) and ChIP followed by genome tiling array analysis (ChIP-chip) become more and more prevalent high throughput technologies for identifying the binding sites of DNA-binding proteins in a genome-wide bases. A number of algorithms have been published to facilitate the identification of the binding sites of the DNA-binding proteins of interest. The identified binding sites in the list of peaks are usually converted to BED or WIG file format to be loaded to UCSC genome browser as custom tracks for investigators to view the proximity to various genomic features such as genes, exons and conserved elements. However, clicking through the genome browser could be a daunting task for the biologist if the number of peaks gets large or the peaks spread widely across the genome.

Here we have developed a Bioconductor package called ChIPpeakAnno to facilitate the batch annotation of the peaks identified from either ChIP-seq or ChIP-chip experiments. We have implemented functionality to find the nearest gene, exon, miRNA, gene end or custom features supplied by users such as most conserved elements and other transcription factor binding sites leveraging IRanges. Since the genome annotation gets updated from time to time, we have leveraged the biomaRt package from Bioconductor to retrieve the annotation data on the fly if the annotation of interest is available via the biomaRt package. The users also have the flexibility to pass their own annotation data as GRanges (or RangedData) or pass in annotation data from GenomicFeatures. We have also leveraged BSgenome and biomaRt package on implementing functions to retrieve the sequences around the peak identified for peak validation. To understand whether the identified peaks are enriched around genes with certain GO terms, we have implemented GO enrichment test in ChIPpeakAnno package leveraging the hypergeometric test phyper in stats package and integrated with Gene Ontology (GO) annotation from GO.db package and multiplicity adjustment functions from multtest package.

2 Quick start

```r
> library(ChIPpeakAnno)
> ## import the MACS output
> macs <- system.file("extdata", "MACS_peaks.xls", package="ChIPpeakAnno")
> macsOutput <- toGRanges(macs, format="MACS")
> ## annotate the peaks with ensembl annotation
> data(TSS.human.GRCh38)
> macs.anno <- annotatePeakInBatch(macsOutput, AnnotationData=TSS.human.GRCh38,
> + output="overlapping", maxgap=5000L)
> ## add gene symbols
> library(org.Hs.eg.db)
> macs.anno <- addGeneIDs(annotatedPeak=macs.anno,
> + orgAnn="org.Hs.eg.db",
> + IDs2Add="symbol")
> head(macs.anno)
```

GRanges object with 6 ranges and 16 metadata columns:
<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>length</th>
<th>summit</th>
<th>tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>X01.ENSG00000117616</td>
<td>chr1 [ 25323511, 25324015]</td>
<td>*</td>
<td>505</td>
<td>252</td>
<td>45</td>
</tr>
<tr>
<td>X01.ENSG00000187010</td>
<td>chr1 [ 25323511, 25324015]</td>
<td>*</td>
<td>505</td>
<td>252</td>
<td>45</td>
</tr>
<tr>
<td>X02.ENSG00000183726</td>
<td>chr1 [ 25362585, 25362997]</td>
<td>*</td>
<td>313</td>
<td>211</td>
<td>33</td>
</tr>
<tr>
<td>X02.ENSG00000188672</td>
<td>chr1 [ 25362585, 25362997]</td>
<td>*</td>
<td>313</td>
<td>211</td>
<td>33</td>
</tr>
<tr>
<td>X03.NA</td>
<td>chr1 [145558192, 145558537]</td>
<td>*</td>
<td>386</td>
<td>59</td>
<td>39</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start_position</th>
<th>end_position</th>
<th>feature_strand</th>
<th>insideFeature</th>
</tr>
</thead>
<tbody>
<tr>
<td>X01.ENSG00000117616</td>
<td>25242237</td>
<td>25338213</td>
<td>inside</td>
</tr>
<tr>
<td>X01.ENSG00000187010</td>
<td>25272393</td>
<td>25330445</td>
<td>inside</td>
</tr>
<tr>
<td>X02.ENSG00000183726</td>
<td>25337917</td>
<td>25362361</td>
<td>downstream</td>
</tr>
<tr>
<td>X02.ENSG00000188672</td>
<td>25362249</td>
<td>25430192</td>
<td>inside</td>
</tr>
<tr>
<td>X03.NA</td>
<td>&lt;NA&gt;</td>
<td>&lt;NA&gt;</td>
<td>&lt;NA&gt;</td>
</tr>
<tr>
<td>X04.NA</td>
<td>&lt;NA&gt;</td>
<td>&lt;NA&gt;</td>
<td>&lt;NA&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC101928189;RSRP1</td>
</tr>
<tr>
<td>RHCE;RHD</td>
</tr>
<tr>
<td>TMEM50A</td>
</tr>
<tr>
<td>RHCE</td>
</tr>
</tbody>
</table>

---

seqinfo: 12 sequences from an unspecified genome; no seqlengths

```r
> if(interactive()) {## annotate the peaks with UCSC annotation
+ library(GenomicFeatures)
+ library(TxDb.Hsapiens.UCSC.hg38.knownGene)
+ ucsc.hg38.knownGene <- genes(TxDb.Hsapiens.UCSC.hg38.knownGene)
+ macs.anno <- annotatePeakInBatch(macsOutput, 
+ AnnotationData=ucsc.hg38.knownGene, 
+ output="overlapping", maxgap=5000L)
+ macs.anno <- addGeneIDs(annotatedPeak=macs.anno, 
+ orgAnn="org.Hs.eg.db", 
+ feature_id_type="entrez_id", 
+ IDs2Add="symbol")
+ head(macs.anno)
+ }
```

3 Examples of using ChIPpeakAnno
3.1 Task 1: Find the nearest feature such as gene and the distance to the feature such as the transcription start site (TSS) of the nearest gene

We have a list of peaks identified from ChIP-seq or ChIP-chip experiments and we would like to retrieve the nearest gene and distance to the corresponding gene transcription start site. We have retrieved all the genomic locations of the genes for human genome as TSS.human.NCBI36 data package for repeated use with function getAnnotation, now we just pass the annotation to the annotatePeakInBatch function.

```r
> library(ChIPpeakAnno)
> data(myPeakList)
> data(TSS.human.NCBI36)
> annotatedPeak <- annotatePeakInBatch(myPeakList[1:6,],
+ AnnotationData=TSS.human.NCBI36)
> annotatedPeak
```

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1_93_556427.ENSG00000212875</td>
<td>chr1 [556660, 556760]</td>
<td>*</td>
<td>X1_93_556427</td>
</tr>
<tr>
<td>X1_41_559455.ENSG00000212678</td>
<td>chr1 [559774, 559874]</td>
<td>*</td>
<td>X1_41_559455</td>
</tr>
<tr>
<td>X1_12_703729.ENSG00000197049</td>
<td>chr1 [703885, 703985]</td>
<td>*</td>
<td>X1_12_703729</td>
</tr>
<tr>
<td>X1_20_925025.ENSG00000182890</td>
<td>chr1 [926058, 926158]</td>
<td>*</td>
<td>X1_20_925025</td>
</tr>
<tr>
<td>X1_11_1041174.ENSG00000131591</td>
<td>chr1 [1041646, 1041746]</td>
<td>*</td>
<td>X1_11_1041174</td>
</tr>
<tr>
<td>X1_14_1269014.ENSG00000107404</td>
<td>chr1 [1270239, 1270339]</td>
<td>*</td>
<td>X1_14_1269014</td>
</tr>
</tbody>
</table>

**seqinfo:** 24 sequences from an unspecified genome; no seqlengths

To annotate the peaks with other genomic feature, you will need to call function getAnnotation with featureType, e.g., “Exon” for finding the nearest exon, and “miRNA” for finding the nearest miRNA, “5utr” or “3utr” for finding the overlapping 5 prime UTR or 3 prime UTR. Please refer to getAnnotation function for more details.
We have presented the examples using human genome as annotation source. To annotate your data with other species, you will need to pass to the function `getAnnotation` the appropriate dataset for example, `d rerio_gene_ensembl` for zebrafish genome, `mmusculus_gene_ensembl` for mouse genome and `rnorvegicus_gene_ensembl` for rat genome.

For a list of available biomart and dataset, please refer to the `biomaRt` package documentation (Durinck S. et al., 2005). For fast access, in addition to `TSS.human.NCBI36`, `TSS.human.GRCh37`, `TSS.human.GRCh38`, `TSS.mouse.NCBIM37`, `TSS.mouse.GRCm38`, `TSS.rat.RGSC3.4`, `TSS.rat.Rnor_5.0`, `TSS.zebrafish.Zv8`, and `TSS.zebrafish.Zv9` are included as annotation data packages.

You could also pass your own annotation data into the function `annotatePeakInBatch`. For example, if you have a list of transcription factor binding sites from literature and are interested in obtaining the nearest binding site of the transcription factor and distance to it for the list of peaks.

```r
> myPeak1 <- GRanges(seqnames=c("1", "2", "3", "4", "5", "6", "2", "6", "6", "6", "6", "6"),
+   ranges=IRanges(start=c(967654, 2010897, 2496704, 3075869,
+   3123260, 3857501, 201089, 1543200,
+   1557200, 1563000, 1569800, 167889600),
+   end= c(967754, 2010997, 2496804, 3075969,
+   3123360, 3857601, 201089, 1556199,
+   1560599, 1565199, 1573799, 167893599),
+   names=paste("Site", 1:12, sep=""))
> TFbindingSites <- GRanges(seqnames=c("1", "2", "3", "4", "5", "6", "1", "2", "3",
+   "4", "5", "6", "6", "6", "6", "5"),
+   ranges=IRanges(start=c(967659, 2010898, 2496700,
+   3075866, 3123260, 3857500,
+   96765, 201089, 249670, 307586,
+   312326, 385750, 1549800,
+   1564400, 1565000, 1569400,
+ 167888600),
+   end=c(967869, 2011108, 2496920,
+ 3076166, 3123470, 3857780,
+ 96985, 201299, 249980, 307796,
+ 312586, 385960, 1550599, 1567999,
+ 1563999, 1571199, 167888999),
+   names=paste("t", 1:17, sep="")),
> annotatedPeak2 <- annotatePeakInBatch(myPeak1, AnnotationData=TFbindingSites)
> annotatedPeak2
```

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>peak</th>
<th>feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site1.t1</td>
<td>chr1 [967654, 967754]</td>
<td>*</td>
<td>Site1</td>
<td>t1</td>
</tr>
<tr>
<td>Site2.t2</td>
<td>chr2 [2010897, 2010997]</td>
<td>*</td>
<td>Site2</td>
<td>t2</td>
</tr>
<tr>
<td>Site3.t3</td>
<td>chr3 [2496704, 2496804]</td>
<td>*</td>
<td>Site3</td>
<td>t3</td>
</tr>
<tr>
<td>Site4.t4</td>
<td>chr4 [3075869, 3075969]</td>
<td>*</td>
<td>Site4</td>
<td>t4</td>
</tr>
<tr>
<td>Site5.t5</td>
<td>chr5 [3123260, 3123360]</td>
<td>*</td>
<td>Site5</td>
<td>t5</td>
</tr>
<tr>
<td>Site8.t1</td>
<td>chr6 [1543200, 1555199]</td>
<td>*</td>
<td>Site8</td>
<td>t14</td>
</tr>
<tr>
<td>Site9.t1</td>
<td>chr6 [1557200, 1560599]</td>
<td>*</td>
<td>Site9</td>
<td>t14</td>
</tr>
<tr>
<td>Site10.t1</td>
<td>chr6 [1563000, 1565199]</td>
<td>*</td>
<td>Site10</td>
<td>t15</td>
</tr>
<tr>
<td>Site11.t1</td>
<td>chr6 [1569800, 1573799]</td>
<td>*</td>
<td>Site11</td>
<td>t16</td>
</tr>
<tr>
<td>Site12.t1</td>
<td>chr5 [167889600, 167893599]</td>
<td>*</td>
<td>Site12</td>
<td>t17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start_position</th>
<th>end_position</th>
<th>feature</th>
<th>strand</th>
<th>insideFeature</th>
<th>distanceToFeature</th>
</tr>
</thead>
<tbody>
<tr>
<td>967659</td>
<td>967869</td>
<td>overlapStart</td>
<td>-5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Both BED format and GFF format are common file format that provides a flexible way to define the peaks and annotations as the data lines. Therefore, conversion functions toGRanges were implemented for converting these data format to GRanges before calling annotatePeakInBatch.

Once you annotated the peak list, you can plot the distance to nearest feature such as TSS.
3.2 Task 2: Obtain overlapping peaks for potential transcription factor complex and determine the significance of the overlapping and generate Venn Diagram

Here is an example of obtaining overlapping peaks with maximum gap 1kb for two peak ranges.

```r
> peaks1 <- GRanges(seqnames=c("1", "2", "3", "4", "5", "6", "+
2", "6", "6", "6", "5"),
+ ranges=IRanges(start=c(967654, 2010897, 2496704, 3075869,
+ 3123260, 3857501, 201089, 1543200,
+ 1557200, 1563000, 1569800, 167889600),
+ end= c(967754, 2010997, 2496804, 3075969,
+ 3123360, 3857601, 201089, 1543200,
+ 1560599, 1565199, 1573799, 167893599),
+ names=paste("Site", 1:12, sep="")),
+ strand=+
>
> peaks2 <- GRanges(seqnames=c("1", "2", "3", "4", "5", "6", "1", "2", "3", "+
4", "5", "6", "6", "6", "6", "5"),
+ ranges=IRanges(start=c(967659, 2010898, 2496700,
+ 3075866, 3123260, 3857500,
+ 96765, 201089, 249670, 307586,
+ 312326, 385750, 1549800,
+ 1545400, 1565000, 1569400,
+ 167888600),
+ end= c(967869, 2011108, 2496920,
+ 3076166,3123470, 3857780,
+ 96985, 201299, 249890, 307796,
+ 312586, 385960, 1550599, 1560799,
+ 1565399, 1571199, 167889999),
+ names=paste("n", 1:17, sep="")),
"
>) ol <- findOverlapsOfPeaks(peaks1, peaks2, maxgap=1000)
> peaklist <- ol$peaklist

Here is a list of overlapping peaks with maximum gap 1kb and a pie graph describing the distribution of relative position of peaks1 to peaks2 for overlapping peaks.

```r
> overlappingPeaks <- ol$overlappingPeaks
> overlappingPeaks

```
Figure 2: Pie chart of common peaks among features.

Here is the merged overlapping peaks, which can be used to obtain overlapping peaks with another TF binding sites from a protein complex.

> pie(table(overlappingPeaks["peaks1///peaks2"]$overlapFeature))

> peaklist["peaks1///peaks2"]

GRanges object with 11 ranges and 1 metadata column:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;IRanges&gt;</td>
<td>&lt;Rle&gt;</td>
</tr>
<tr>
<td>[1]</td>
<td>[967654, 967869]</td>
<td>+</td>
</tr>
<tr>
<td>[2]</td>
<td>[201089, 201299]</td>
<td>*</td>
</tr>
</tbody>
</table>
Here is the peaks in peaks1 that not overlaps with peaks in peaks2

```r
> peaklist[["peaks1"]]

NULL
```

Here is the peaks in peaks2 that not overlap with peaks in peaks1

```r
> peaklist[["peaks2"]]

GRanges object with 5 ranges and 1 metadata column:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>peakNames</th>
</tr>
</thead>
</table>
```

Venn Diagram can be generated by the following function call using the results of `findOverlapsOfPeaks` as an input (Figure 3). P-values indicate whether the extent of overlapping is significant.

```r
> makeVennDiagram(ol, totalTest=1e+2)

$p.value

<table>
<thead>
<tr>
<th>peaks1</th>
<th>peaks2</th>
<th>pval</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>5.890971e-12</td>
</tr>
</tbody>
</table>

$vennCounts

<table>
<thead>
<tr>
<th>peaks1</th>
<th>peaks2</th>
<th>Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>83</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
</tbody>
</table>
```

attr("class")

[1] "VennCounts"
Figure 3: venn diagram of overlaps

Users can also try other tools to draw vennDiagrams such as Vennerable.

```r
> # install.packages("Vennerable", repos="http://R-Forge.R-project.org", type="source")
> # library(Vennerable)
> # venn_cnt2venn <- function(venn_cnt){
> # n <- which(colnames(venn_cnt)=='Counts') - 1
> # SetNames=colnames(venn_cnt)[1:n]
> # Weight=venn_cnt[,"Counts"]
> # names(Weight) <- apply(venn_cnt[,1:n], 1, paste, collapse="")
> # Venn(SetNames=SetNames, Weight=Weight)
> # }
> #
> # v <- venn_cnt2venn(ol$venn_cnt)
> # plot(v)
```

The `findOverlapsOfPeaks` function can be called to obtain overlaps upto 5 peak lists for example, the overlap peaks in peaks1, peaks2 and peaks3 (Figure 4).

```r
> peaks3 <- GRanges(seqnames=c("1", "2", "3", "4", "5",
+ "6", "1", "2", "3", "4"),
+ ranges=IRanges(start=c(967859, 2010868, 2496500, 3075966,
+ 3123460, 3851500, 9685, 201189,
+ 249600, 307386),
+ end= c(967969, 2011908, 2496720, 3076166,
+ 3123470, 3857680, 9695, 201299,
+ 249890, 307796),
+ names=paste("p", 1:10, sep="")),
+ strand=c("+", "+", "+", "+", "+",
+ "+", "+", "+", "+", "+")
> ol <- findOverlapsOfPeaks(peaks1, peaks2, peaks3, maxgap=1000, connectedPeaks="min")
> makeVennDiagram(ol, totalTest=1e+2)

$p.value
  peaks1 peaks2 peaks3 pvall
[1,]  0  1 1 1.123492e-09
Figure 4: venn diagram of overlaps for three input peak lists

```
> makeVennDiagram(list(peaks1, peaks2), NameOfPeaks=c("TF1", "TF2"),
+     maxgap=0, minoverlap =1, totalTest=100)

$p.value
  TF1 TF2     pval
[1,] 1 1 9.837922e-10

$vennCounts
   TF1 TF2 Counts
[1,] 0 0 82
[2,] 1 0 6
[3,] 0 1 1
[4,] 1 1 11
attr(,"class")
[1] "VennCounts"
```

Venn Diagram can also be generated by the following function call with p-value that indicates whether the extent of overlapping is significant (Figure 5,6). Note, the maxgap is changed to 0.

```
> makeVennDiagram(list(peaks1, peaks2), NameOfPeaks=c("TF1", "TF2"),
+     maxgap=0, minoverlap =1, totalTest=100)

$p.value
  TF1 TF2     pval
[1,] 1 1 9.837922e-10

$vennCounts
   TF1 TF2 Counts
[1,] 0 0 82
[2,] 1 0 6
[3,] 0 1 1
[4,] 1 1 11
attr(,"class")
[1] "VennCounts"
```
Figure 5: Venn diagram to depict the overlaps between two peak lists

```r
> makeVennDiagram(list(peaks1, peaks2, peaks3),
+   NameOfPeaks=c("TF1", "TF2", "TF3"),
+   maxgap=0, minoverlap =1, totalTest=100)

$p.value$

<table>
<thead>
<tr>
<th></th>
<th>TF1</th>
<th>TF2</th>
<th>TF3</th>
<th>pval</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1.123492e-09</td>
</tr>
<tr>
<td>[2,]</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5.131347e-06</td>
</tr>
<tr>
<td>[3,]</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>9.837922e-10</td>
</tr>
</tbody>
</table>

$vennCounts$

<table>
<thead>
<tr>
<th></th>
<th>TF1</th>
<th>TF2</th>
<th>TF3</th>
<th>Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
</tr>
<tr>
<td>[2,]</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>[3,]</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>[4,]</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>[5,]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>[6,]</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>[7,]</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>[8,]</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>
attr("class")
[1] "VennCounts"

3.3 Task 3: Obtain sequences surrounding the peaks for PCR validation or motif discovery

Here is an example of obtaining sequences surrounding the peak intervals including 20 bp upstream and downstream sequence.
Figure 6: venn diagram of overlaps for three input peaklists directly

```r
> peaks <- GRanges(seqnames=c("NC_008253", "NC_010468"),
+ ranges=IRanges(start=c(100, 500),
+ end=c(300, 600),
+ names=c("peak1", "peak2")))
> library(BSgenome.Ecoli.NCBI.20080805)
> peaksWithSequences <- getAllPeakSequence(peaks, upstream=20,
+ downstream=20, genome=Ecoli)

You can easily convert the obtained sequences into fasta format for motif discovery by calling the function write2FASTA.

```R
> write2FASTA(peaksWithSequences, "test.fa")
```  

3.4 Task 4: Obtain enriched gene ontology (GO) terms or KEGG terms near the peaks

Once you have obtained the annotated peak data from the example above, you can also use the function getEnriched to obtain a list of enriched gene ontology (GO) terms via GOstats. The ontology could also be set as KEGG or reactome.

Once you have obtained the annotated peak data from the example above, you can also use the function getEnrichedGO to obtain a list of enriched gene ontology (GO) terms using hypergeometric test.

library(org.Hs.eg.db)
enrichedGO = getEnrichedGO(annotatedPeak, orgAnn = "org.Hs.eg.db", maxP = 0.01, multiAdj = TRUE, minGOTerm = 10, multiAdjMethod = "BH")

> library(org.Hs.eg.db)
> over <- getEnrichedGO(annotatedPeak, orgAnn="org.Hs.eg.db", + maxP=0.01, multiAdj=FALSE, minGOTerm=10, multiAdjMethod="")
> head(over[["bp"]])

<table>
<thead>
<tr>
<th>go.id</th>
<th>go.term</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:0001736</td>
<td>establishment of planar polarity</td>
</tr>
<tr>
<td>GO:0001840</td>
<td>neural plate development</td>
</tr>
<tr>
<td>GO:0001941</td>
<td>postsynaptic membrane organization</td>
</tr>
<tr>
<td>GO:0001964</td>
<td>startle response</td>
</tr>
<tr>
<td>GO:0007164</td>
<td>establishment of tissue polarity</td>
</tr>
<tr>
<td>GO:0031122</td>
<td>cytoplasmic microtubule organization</td>
</tr>
</tbody>
</table>

Ontology count.InDataset count.InGenome pvalue totaltermInDataset
1 BP 1 28 0.008619994 405
2 BP 1 11 0.03395307 405
3 BP 1 22 0.006779114 405
4 BP 1 23 0.007086164 405
5 BP 1 28 0.008619994 405
6 BP 1 32 0.009845356 405

totaltermInGenome EntrezID
1 1310084 1855
2 1310084 1855
3 1310084 1855
4 1310084 1855
5 1310084 1855
6 1310084 1855

> head(over[["cc"]])

<table>
<thead>
<tr>
<th>go.id</th>
<th>go.term</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:0016328</td>
<td>lateral plasma membrane</td>
</tr>
</tbody>
</table>

Ontology count.InDataset count.InGenome pvalue totaltermInDataset
1 CC 1 48 0.008016845 61

totaltermInGenome EntrezID
1 363819 1855

> head(over[["mf"]])

<table>
<thead>
<tr>
<th>go.id</th>
<th>go.term</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:0005109</td>
<td>frizzled binding</td>
</tr>
<tr>
<td>GO:0017048</td>
<td>Rho GTPase binding</td>
</tr>
<tr>
<td>GO:0048365</td>
<td>Rac GTPase binding</td>
</tr>
</tbody>
</table>

Ontology count.InDataset count.InGenome pvalue totaltermInDataset
1 MF 1 37 0.003861301 24
2 MF 1 71 0.007396923 24
3 MF 1 32 0.003340340 24

totaltermInGenome EntrezID
1 229560 1855
2 229560 1855
Please note that org.Hs.eg.db is the GO gene mapping for Human, for other organisms, please refer to http://www.bioconductor.org/packages/release/data/annotation/ for additional org.xx.eg.db packages. Or you can try egOrgMap to get the annotation database.

```r
> egOrgMap("Mus musculus")
[1] "org.Mm.eg.db"
> egOrgMap("Homo sapiens")
[1] "org.Hs.eg.db"
```

### 3.5 Task 5: Find peaks with bi-directional promoters

Here is an example to find peaks with bi-directional promoters and output percent of peaks near bi-directional promoters.

```r
> data(myPeakList)
> data(TSS.human.NCBI36)
> annotatedBDP <- peaksNearBDP(myPeakList[1:10,]
+ AnnotationData=TSS.human.NCBI36,
+ MaxDistance=5000,
+ PeakLocForDistance="middle",
+ FeatureLocForDistance="TSS")
> annotatedBDP$peaksWithBDP
```

GRanges object with 6 ranges and 9 metadata columns:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr1</td>
<td>[1300503, 1300603]</td>
<td>*</td>
<td>X1_14_1300250</td>
</tr>
<tr>
<td>chr1</td>
<td>[1300503, 1300603]</td>
<td>*</td>
<td>X1_14_1300250</td>
</tr>
<tr>
<td>chr1</td>
<td>[559774, 559874]</td>
<td>*</td>
<td>X1_41_559455</td>
</tr>
<tr>
<td>chr1</td>
<td>[559774, 559874]</td>
<td>*</td>
<td>X1_41_559455</td>
</tr>
<tr>
<td>chr1</td>
<td>[556680, 556760]</td>
<td>*</td>
<td>X1_93_556427</td>
</tr>
<tr>
<td>chr1</td>
<td>[556680, 556760]</td>
<td>*</td>
<td>X1_93_556427</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>feature</th>
<th>start_position</th>
<th>end_position</th>
<th>feature_strand</th>
<th>insideFeature</th>
<th>distanceToFeature</th>
<th>shortestDistance</th>
<th>fromOverlappingOrNearest</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr1</td>
<td>1303908</td>
<td>1304275</td>
<td>+ upstream</td>
<td>-3355</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chr1</td>
<td>1298974</td>
<td>1300443</td>
<td>- upstream</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chr1</td>
<td>559620</td>
<td>560165</td>
<td>+ inside</td>
<td>204</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chr1</td>
<td>557860</td>
<td>557930</td>
<td>- upstream</td>
<td>-110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chr1</td>
<td>556318</td>
<td>557859</td>
<td>+ inside</td>
<td>1894</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chr1</td>
<td>556240</td>
<td>556304</td>
<td>- upstream</td>
<td>392</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

15
### 3.6 Task 6: Output a summary of motif occurrence in the peaks.

Here is an example to search the peaks for the motifs in examplepattern.fa file.

```r
> peaks <- GRanges(seqnames=c("NC_008253", "NC_010468"),
+ ranges=IRanges(start=c(100, 500),
+ end=c(300, 600),
+ names=c("peak1", "peak2")))
> filepath <- system.file("extdata", "examplePattern.fa", package="ChIPpeakAnno")
> library(BSgenome.Ecoli.NCBI.20080805)
> summarizePatternInPeaks(patternFilePath=filepath, format="fasta", skip=0L,
+ BSgenomeName=Ecoli, peaks=peaks)
```

<table>
<thead>
<tr>
<th>n.peaksWithPattern</th>
<th>n.totalPeaks</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>GGNCCK</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>AACCNM</td>
</tr>
</tbody>
</table>

### 3.7 Task 7: Add other IDs to annotated peaks or enrichedGO

Here is an example to add gene symbol to annotated peaks.

```r
> data(annotatedPeak)
> library(org.Hs.eg.db)
> addGeneIDs(annotatedPeak[1:6,], orgAnn=org.Hs.eg.db", ID2Add=c("symbol"))
```

GRanges object with 6 ranges and 9 metadata columns:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1_11_100272487.ENSG000000202254</td>
<td>1 [100272801, 100272900]</td>
<td>+</td>
<td>1_11_100272487</td>
</tr>
<tr>
<td>X1_11_108905539.ENSG000000186086</td>
<td>1 [108906026, 108906125]</td>
<td>+</td>
<td>1_11_108905539</td>
</tr>
<tr>
<td>X1_11_110106925.ENSG00000065135</td>
<td>1 [110107267, 110107366]</td>
<td>+</td>
<td>1_11_110106925</td>
</tr>
<tr>
<td>X1_11_110679983.ENSG000000197106</td>
<td>1 [110680489, 110680568]</td>
<td>+</td>
<td>1_11_110679983</td>
</tr>
<tr>
<td>X1_11_110681677.ENSG000000197106</td>
<td>1 [110682125, 110682224]</td>
<td>+</td>
<td>1_11_110681677</td>
</tr>
<tr>
<td>X1_11_110756560.ENSG000000116396</td>
<td>1 [110756823, 110756922]</td>
<td>+</td>
<td>1_11_110756560</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>feature</th>
<th>start_position</th>
<th>end_position</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1_11_100272487.ENSG000000202254</td>
<td>ENSG000000202254</td>
<td>100257218 100257309</td>
</tr>
<tr>
<td>X1_11_108905539.ENSG000000186086</td>
<td>ENSG000000186086</td>
<td>108918435 109013624</td>
</tr>
<tr>
<td>X1_11_110106925.ENSG00000065135</td>
<td>ENSG00000065135</td>
<td>110091233 110136975</td>
</tr>
<tr>
<td>X1_11_110679983.ENSG000000197106</td>
<td>ENSG000000197106</td>
<td>110693108 110744824</td>
</tr>
<tr>
<td>X1_11_110681677.ENSG000000197106</td>
<td>ENSG000000197106</td>
<td>110693108 110744824</td>
</tr>
<tr>
<td>X1_11_110756560.ENSG000000116396</td>
<td>ENSG000000116396</td>
<td>110753965 110776666</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>insideFeature</th>
<th>distanceToFeature</th>
<th>shortestDistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1_11_100272487.ENSG000000202254</td>
<td>downstream</td>
<td>15582 15491</td>
</tr>
<tr>
<td>X1_11_108905539.ENSG000000186086</td>
<td>upstream</td>
<td>-12410 12310</td>
</tr>
<tr>
<td>X1_11_110106925.ENSG00000065135</td>
<td>inside</td>
<td>16033 16033</td>
</tr>
<tr>
<td>X1_11_110679983.ENSG000000197106</td>
<td>upstream</td>
<td>-12640 12540</td>
</tr>
<tr>
<td>X1_11_110681677.ENSG000000197106</td>
<td>upstream</td>
<td>-10984 10884</td>
</tr>
<tr>
<td>X1_11_110756560.ENSG000000116396</td>
<td>inside</td>
<td>2857 2857</td>
</tr>
</tbody>
</table>
3.8 Task 8: annotate ChIP results from BED or GFF files or MACS output xls file

Here is an example to annotate peaks in BED file format and GFF file format.

```r
> bed <- system.file("extdata", "MACS_output.bed", package="ChIPpeakAnno")
> gr1 <- toGRanges(bed, format="BED", header=FALSE)
> # one can also try import from rtracklayer
> library(rtracklayer)
> gr1.import <- import(bed, format="BED")
> identical(start(gr1), start(gr1.import))

[1] TRUE
```

```
GRanges object with 2 ranges and 1 metadata column:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACS_peak_1</td>
<td>chr1 [28341, 29610] *</td>
<td></td>
<td>160.81</td>
</tr>
<tr>
<td>MACS_peak_2</td>
<td>chr1 [90821, 91234] *</td>
<td></td>
<td>133.12</td>
</tr>
</tbody>
</table>
```

```
> gr1.import[1:2] #note the name slot is different from gr1

GRanges object with 2 ranges and 2 metadata columns:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>score</th>
<th>name</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACS_peak_1</td>
<td>chr1 [28341, 29610] *</td>
<td></td>
<td>160.81</td>
<td>MACS_peak_1</td>
<td>160.81</td>
</tr>
<tr>
<td>MACS_peak_2</td>
<td>chr1 [90821, 91234] *</td>
<td></td>
<td>133.12</td>
<td>MACS_peak_2</td>
<td>133.12</td>
</tr>
</tbody>
</table>
```

```r
> gff <- system.file("extdata", "GFF_peaks.gff", package="ChIPpeakAnno")
> gr2 <- toGRanges(gff, format="GFF", header=FALSE, skip=3)
> ol <- findOverlapsOfPeaks(gr1, gr2)
> makeVennDiagram(ol)

$p.value

<table>
<thead>
<tr>
<th>gr1 gr2 pval</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 0</td>
</tr>
</tbody>
</table>
```
$\text{vennCounts}$

\begin{verbatim}
  gr1 gr2 Counts
  [1,] 0 0 0
  [2,] 0 1 61
  [3,] 1 0 62
  [4,] 1 1 164
attr(,"class")
[1] "VennCounts"
\end{verbatim}

\begin{verbatim}
> pie(table(ol$overlappingPeaks["gr1\///gr2"]$overlapFeature))
\end{verbatim}

Find all features within 5kb away from the overlapping peaks using `annotatePeakInBatch`.

\begin{verbatim}
> data(TSS.human.GRCh37)
> overlaps <- ol$peaklist["gr1\///gr2"]
> overlaps.anno <- annotatePeakInBatch(overlaps, AnnotationData=TSS.human.GRCh37,
+    output="overlapping", maxgap=5000L)
> overlaps.anno <- addGeneIDs(overlaps.anno, "org.Hs.eg.db", "symbol")
> head(overlaps.anno)
\end{verbatim}

\begin{verbatim}
GRanges object with 6 ranges and 11 metadata columns:

  seqnames ranges strand | <Rle> <IRanges> <Rle>
  X001.ENSG00000228327 chr1 [713791, 715578] * |
  X001.ENSG00000237491 chr1 [713791, 715578] * |
  X001.ENSG00000242937 chr1 [713791, 715578] * |
  X002.ENSG00000237491 chr1 [724851, 727191] * |
  X002.ENSG00000242937 chr1 [724851, 727191] * |
  X002.ENSG00000197049 chr1 [724851, 727191] * |

  peakNames peak
  <CharacterList> <character>
  X001.ENSG00000228327 gr1__MACS_peak_13,gr2__region_0,gr2__region_1 001
  X001.ENSG00000237491 gr1__MACS_peak_13,gr2__region_0,gr2__region_1 001
  X001.ENSG00000242937 gr1__MACS_peak_13,gr2__region_0,gr2__region_1 001
\end{verbatim}
Figure 8: Pie chart of common peaks among features

<table>
<thead>
<tr>
<th>feature</th>
<th>start_position</th>
<th>end_position</th>
<th>feature_strand</th>
<th>overlapStart</th>
<th>distancetoFeature</th>
<th>shortestDistance</th>
<th>fromOverlappingOrNearest</th>
</tr>
</thead>
<tbody>
<tr>
<td>X001.ENSG00000228327</td>
<td>700238</td>
<td>714006</td>
<td>-</td>
<td>-372</td>
<td>215</td>
<td>215</td>
<td>LOC100288069;LOC101929540</td>
</tr>
<tr>
<td>X001.ENSG00000237491</td>
<td>714163</td>
<td>740255</td>
<td>+</td>
<td>-3535</td>
<td>1748</td>
<td>10688</td>
<td>LOC100287934</td>
</tr>
<tr>
<td>X001.ENSG00000242937</td>
<td>717326</td>
<td>720070</td>
<td>+</td>
<td>7525</td>
<td>4781</td>
<td>4781</td>
<td>LOC100287934</td>
</tr>
<tr>
<td>X002.ENSG00000197049</td>
<td>721321</td>
<td>722513</td>
<td>+</td>
<td>3530</td>
<td>2338</td>
<td>2338</td>
<td>LOC100287934</td>
</tr>
</tbody>
</table>

Plot the distribution of aggregated peak scores or peak numbers around transcript start sites (Figure 9).

```r
> gr1.copy <- gr1
> gr1.copy$score <- 1
```
Figure 9: Distribution of aggregated peak scores or peak numbers around transcript start sites.

Summarize peak distribution over exon, intron, enhancer, proximal promoter, 5 prime UTR and 3 prime UTR in peak centric and nucleotide centric view using function assignChromosomeRegion(Figure 10). Setting nucleotideLevel = TRUE will give a nucleotide level distribution over different features.
Figure 10: Peak distribution over different genomic features.

4 References

4. S. Dudoit, J. P. Shaffer, and J. C. Boldrick (Submitted). Multiple hypothesis testing in microarray experiments.
5 Session Info

> toLatex(sessionInfo())

- R version 3.2.0 alpha (2015-03-20 r68043), x86_64-unknown-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8,
  LC_COLLATE=C, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_US.UTF-8,
  LC_PAPER=en_US.UTF-8, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C,
  LC_MEASUREMENT=en_US.UTF-8, LC_IDENTIFICATION=C
- Base packages: base, datasets, grDevices, graphics, grid, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.29.20, BSgenome 1.35.19,
  BSgenome.Ecoli.NCBI.20080805 1.3.1000, Biobase 2.27.2, BiocGenerics 0.13.8,
  Biostrings 2.35.11, ChIPpeakAnno 3.0.1, DBI 0.3.1, FDb.UCSC.tRNAs 1.0.1,
  GenomInfoDb 1.3.15, GenomicFeatures 1.19.33, GenomicRanges 1.19.47,
  IRanges 2.1.43, RSQLite 1.0.0, S4Vectors 0.5.22,
  TxDb.Hsapiens.UCSC.hg19.knownGene 3.1.2, VennDiagram 1.6.9, XVector 0.7.4,
  biomaRt 2.23.5, mirbase.db 1.2.0, org.Hs.eg.db 3.1.2, rtracklayer 1.27.9
- Loaded via a namespace (and not attached): BiocInstaller 1.17.6, BiocParallel 1.1.21,
  BiocStyle 1.5.3, GO.db 3.1.2, GenomicAlignments 1.3.32, MASS 7.3-40, RBGL 1.43.0,
  RCurl 1.95-4.5, Rsamtools 1.19.47, XML 3.98-1.1, bitops 1.0-6, futile.logger 1.4,
  futile.options 1.0.0, graph 1.45.2, lambda.r 1.1.7, limma 3.23.11, multtest 2.23.0,
  splines 3.2.0, survival 2.38-1, tools 3.2.0, zlibbioc 1.13.3