The ChIPpeakAnno user's guide

Lihua Julie Zhu*

October 18, 2010

Contents

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

*julie.zhu@umassmed.edu

Bioconducor 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 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 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 Examples of using ChIPpeakAnno

2.1 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.

```
> library(ChIPpeakAnno)
> data(myPeakList)
> data(TSS.human.NCBI36)
> annotatedPeak = annotatePeakInBatch(myPeakList[1:6, ], AnnotationData = TSS.human.NCBI36)
> as.data.frame(annotatedPeak)
```

```
  space   start     end width                        names         peak strand
1     1  703885  703985   101  1_12_703729 ENSG00000197049  1_12_703729      +
2     1  559774  559874   101  1_41_559455 ENSG00000212678  1_41_559455      +
3     1  556660  556760   101  1_93_556427 ENSG00000212875  1_93_556427      +
4     1 1041646 1041746   101 1_11_1041174 ENSG00000131591 1_11_1041174      -
5     1 1270239 1270339   101 1_14_1269014 ENSG00000107404 1_14_1269014      -
6     1  926058  926158   101  1_20_925025 ENSG00000188290  1_20_925025      -
          feature start_position end_position insideFeature distancetoFeature
1 ENSG00000197049         711183       712376      upstream             -7298
2 ENSG00000212678         559619       560165        inside               155
3 ENSG00000212875         556317       557859        inside               343
4 ENSG00000131591        1007061      1041341      upstream              -305
5 ENSG00000107404        1260522      1274623        inside              4384
6 ENSG00000188290         924208       925333      upstream              -725
  shortestDistance fromOverlappingOrNearest
1             7198             NearestStart
2              155             NearestStart
```

```
3              343             NearestStart
4              305             NearestStart
5             4284             NearestStart
6              725             NearestStart
```

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, drerio_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.mouse.NCBIM37, TSS.rat.RGSC3.4 and TSS.zebrafish.Zv8 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 biding 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.

```
> myPeak1 = RangedData(IRanges(start = c(967654, 2010897, 2496704,
+     3075869, 3123260, 3857501, 201089, 1543200, 1557200, 1563000,
+     1569800, 167889600), end = c(967754, 2010997, 2496804, 3075969,
+     3123360, 3857601, 201089, 1555199, 1560599, 1565199, 1573799,
+     167893599), names = c("Site1", "Site2", "Site3", "Site4",
+     "Site5", "Site6", "Site7", "Site8", "Site9", "Site10", "Site11",
+     "Site12")), space = c("1", "2", "3", "4", "5", "6", "2",
+     "6", "6", "6", "6", "5"))
> TFbindingSites = RangedData(IRanges(start = c(967659, 2010898,
+     2496700, 3075866, 3123260, 3857500, 96765, 201089, 249670,
+     307586, 312326, 385750, 1549800, 1554400, 1565000, 1569400,
+     167888600), end = c(967869, 2011108, 2496920, 3076166, 3123470,
+     3857780, 96985, 201299, 249890, 307796, 312586, 385960, 1550599,
+     1560799, 1565399, 1571199, 167888999), names = c("t1", "t2",
+     "t3", "t4", "t5", "t6", "t7", "t8", "t9", "t10", "t11", "t12",
+     "t13", "t14", "t15", "t16", "t17")), space = c("1", "2",
+     "3", "4", "5", "6", "1", "2", "3", "4", "5", "6", "6", "6",
+     "6", "6", "5"), strand = c(1, 1, 1, 1, 1, 1, -1, -1, -1,
+     -1, -1, -1, 1, 1, 1, 1, 1))
> annotatedPeak2 = annotatePeakInBatch(myPeak1, AnnotationData = TFbindingSites)
> pie(table(as.data.frame(annotatedPeak2)$insideFeature))
> as.data.frame(annotatedPeak2)

   space     start       end width      names   peak strand feature
1      1    967654    967754   101   Site1 t1  Site1      +      t1
2      2   2010897   2010997   101   Site2 t2  Site2      +      t2
3      2    201089    201089     1   Site7 t8  Site7      -      t8
4      3   2496704   2496804   101   Site3 t3  Site3      +      t3
5      4   3075869   3075969   101   Site4 t4  Site4      +      t4
6      5 167889600 167893599  4000 Site12 t17 Site12      +     t17
7      5   3123260   3123360   101   Site5 t5  Site5      +      t5
8      6   1563000   1565199  2200 Site10 t15 Site10      +     t15
9      6   1569800   1573799  4000 Site11 t16 Site11      +     t16
10     6   3857501   3857601   101   Site6 t6  Site6      +      t6
11     6   1543200   1555199 12000  Site8 t13  Site8      +     t13
```

```
12     6   1557200   1560599  3400  Site9 t14  Site9      +     t14
   start_position end_position  insideFeature distancetoFeature
1          967659       967869   overlapStart                -5
2         2010898      2011108   overlapStart                -1
3          201089       201299         inside               210
4         2496700      2496920         inside                 4
5         3075866      3076166         inside                 3
6       167888600    167888999     downstream              1000
7         3123260      3123470         inside                 0
8         1565000      1565399   overlapStart             -2000
9         1569400      1571199     overlapEnd               400
10        3857500      3857780         inside                 1
11        1549800      1550599 includeFeature             -6600
12        1554400      1560799         inside              2800
   shortestDistance fromOverlappingOrNearest
1                 5             NearestStart
2                 1             NearestStart
3                 0             NearestStart
4                 4             NearestStart
5                 3             NearestStart
6               601             NearestStart
7                 0             NearestStart
8               199             NearestStart
9               400             NearestStart
10                1             NearestStart
11             4600             NearestStart
12              200             NearestStart
```

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 RfunctionBED2RangedData and RfunctionGFF2RangedData were implemented for converting these data format to RangedData before calling `annotatePeakInBatch`

Once you annotated the peak list, you can plot the distance to nearest feature such as TSS.

2.2 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.

```
> peaks1 = RangedData(IRanges(start = c(967654, 2010897, 2496704,
+     3075869, 3123260, 3857501, 201089, 1543200, 1557200, 1563000,
+     1569800, 167889600), end = c(967754, 2010997, 2496804, 3075969,
+     3123360, 3857601, 201089, 1555199, 1560599, 1565199, 1573799,
+     167893599), names = c("Site1", "Site2", "Site3", "Site4",
+     "Site5", "Site6", "Site7", "Site8", "Site9", "Site10", "Site11",
+     "Site12")), space = c("1", "2", "3", "4", "5", "6", "2",
+     "6", "6", "6", "6", "5"), strand = as.integer(1))
> peaks2 = RangedData(IRanges(start = c(967659, 2010898, 2496700,
+     3075866, 3123260, 3857500, 96765, 201089, 249670, 307586,
+     312326, 385750, 1549800, 1554400, 1565000, 1569400, 167888600),
+     end = c(967869, 2011108, 2496920, 3076166, 3123470, 3857780,
+         96985, 201299, 249890, 307796, 312586, 385960, 1550599,
+         1560799, 1565399, 1571199, 167888999), names = c("t1",
+         "t2", "t3", "t4", "t5", "t6", "t7", "t8", "t9", "t10",
```

```
+         "t11", "t12", "t13", "t14", "t15", "t16", "t17")), space = c("1",
+     "2", "3", "4", "5", "6", "1", "2", "3", "4", "5", "6", "6",
+     "6", "6", "6", "5"), strand = c(1, 1, 1, 1, 1, 1, -1, -1,
+     -1, -1, -1, -1, 1, 1, 1, 1, 1))
> t1 = findOverlappingPeaks(peaks1, peaks2, maxgap = 1000, multiple = F,
+     NameOfPeaks1 = "TF1", NameOfPeaks2 = "TF2")
```

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.

```
> overlappingPeaks = t1$OverlappingPeaks
> overlappingPeaks
```

```
      TF1 chr TF2 TF2_start   TF2_end strand TF1_start   TF1_end strand1
1   Site1   1  t1    967659    967869      +    967654    967754       +
5   Site2   2  t2   2010898   2011108      +   2010897   2010997       +
10  Site7   2  t8    201089    201299      -    201089    201089       +
6   Site3   3  t3   2496700   2496920      +   2496704   2496804       +
7   Site4   4  t4   3075866   3076166      +   3075869   3075969       +
4  Site12   5 t17 167888600 167888999      + 167889600 167893599       +
8   Site5   5  t5   3123260   3123470      +   3123260   3123360       +
2  Site10   6 t15   1565000   1565399      +   1563000   1565199       +
3  Site11   6 t16   1569400   1571199      +   1569800   1573799       +
9   Site6   6  t6   3857500   3857780      +   3857501   3857601       +
11  Site8   6 t13   1549800   1550599      +   1543200   1555199       +
12  Site9   6 t14   1554400   1560799      +   1557200   1560599       +
   overlapFeature shortestDistance
1    overlapStart                5
5    overlapStart                1
10         inside                0
6          inside                4
7          inside                3
4      downstream              601
8          inside                0
2    overlapStart              199
3      overlapEnd              400
9          inside                1
11 includeFeature             4600
12         inside              200
```

```
> pie(table(overlappingPeaks$overlapFeature))
```

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

```
> as.data.frame(t1$MergedPeaks)
```

```
   space     start       end width              names
1      1    967654    967869   216   TF1-Site1-TF2-t1
2      2   2010897   2011108   212   TF1-Site2-TF2-t2
3      2    201089    201299   211   TF1-Site7-TF2-t8
4      3   2496700   2496920   221   TF1-Site3-TF2-t3
5      4   3075866   3076166   301   TF1-Site4-TF2-t4
6      5 167888600 167893599  5000 TF1-Site12-TF2-t17
7      5   3123260   3123470   211   TF1-Site5-TF2-t5
8      6   1563000   1565399  2400 TF1-Site10-TF2-t15
9      6   1569400   1573799  4400 TF1-Site11-TF2-t16
10     6   3857500   3857780   281   TF1-Site6-TF2-t6
11     6   1543200   1555199 12000  TF1-Site8-TF2-t13
12     6   1554400   1560799  6400  TF1-Site9-TF2-t14
```

Here is the peaks in peaks1 that overlaps with peaks in peaks2

```
> as.data.frame(t1$Peaks1withOverlaps)
```

```
   space     start       end width  names strand
1      1    967654    967754   101  Site1      +
2      2   2010897   2010997   101  Site2      +
3      2    201089    201089     1  Site7      +
4      3   2496704   2496804   101  Site3      +
5      4   3075869   3075969   101  Site4      +
6      5 167889600 167893599  4000 Site12      +
7      5   3123260   3123360   101  Site5      +
8      6   1563000   1565199  2200 Site10      +
9      6   1569800   1573799  4000 Site11      +
10     6   3857501   3857601   101  Site6      +
11     6   1543200   1555199 12000  Site8      +
12     6   1557200   1560599  3400  Site9      +
```

Here is the peaks in peaks2 that overlap with peaks in peaks1

```
> as.data.frame(t1$Peaks2withOverlaps)
```

```
   space     start       end width names strand
1      1    967659    967869   211    t1      +
2      2   2010898   2011108   211    t2      +
3      2    201089    201299   211    t8      -
4      3   2496700   2496920   221    t3      +
5      4   3075866   3076166   301    t4      +
6      5 167888600 167888999   400   t17      +
7      5   3123260   3123470   211    t5      +
8      6   1565000   1565399   400   t15      +
9      6   1569400   1571199  1800   t16      +
10     6   3857500   3857780   281    t6      +
11     6   1549800   1550599   800   t13      +
12     6   1554400   1560799  6400   t14      +
```

The `findOVerlappingPeaks` function can be repeatedly called to obtain for example, the peaks in peaks1 that overlap with peaks in both peaks2 and peaks3.

```
> peaks3 = RangedData(IRanges(start = c(967859, 2010868, 2496500,
+     3075966, 3123460, 3851500, 96865, 201189, 249600, 307386),
+     end = c(967969, 2011908, 2496720, 3076166, 3123470, 3857680,
+         96985, 201299, 249890, 307796), names = c("p1", "p2",
+         "p3", "p4", "p5", "p6", "p7", "p8", "p9", "p10")), space = c("1",
+     "2", "3", "4", "5", "6", "1", "2", "3", "4"), strand = c(1,
+     1, 1, 1, 1, -1, -1, -1, -1))
> findOverlappingPeaks(findOverlappingPeaks(peaks1, peaks2, maxgap = 1000,
+     multiple = F, NameOfPeaks1 = "TF1", NameOfPeaks2 = "TF2")$Peaks1withOverlap,
+     peaks3, maxgap = 1000, multiple = F, NameOfPeaks1 = "TF1TF2",
+     NameOfPeaks2 = "TF3")$Peaks1withOverlap
```

```
RangedData with 7 rows and 1 value column across 6 spaces
            space             ranges |      strand
      <character>          <IRanges> | <character>
Site1           1 [ 967654,  967754] |           +
Site2           2 [2010897, 2010997] |           +
Site7           2 [ 201089,  201089] |           +
Site3           3 [2496704, 2496804] |           +
Site4           4 [3075869, 3075969] |           +
Site5           5 [3123260, 3123360] |           +
Site6           6 [3857501, 3857601] |           +
```

Venn Diagram can be generated by the following function call with p-value that indicates whether the extent of overlapping is significant.

```
> makeVennDiagram(RangedDataList(peaks1, peaks2), NameOfPeaks = c("TF1",
+     "TF2"), maxgap = 0, totalTest = 100, cex = 1, counts.col = "red")

$p.value
[1] 9.837922e-10

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

2.3 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.

```
> peaks = RangedData(IRanges(start = c(100, 500), end = c(300,
+     600), names = c("peak1", "peak2")), space = c("NC_008253",
+     "NC_010468"))
> 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`.

```
> write2FASTA(peaksWithSequences)

>peak1
GGTTACCTGCCGTGAGTAAATTAAAATTTTATTGACTTAGGTCACTAAATACTTTAACCAATATAGGCATAGCGCACAGA
CAGATAAAAATTACAGAGTACACAACATCCATGAAACGCATTAGCACCACCATTACCACCACCATCACCATTACCACAGG
TAACGGTGCGGGCTGACGCGTACAGGAAACACAGAAAAAAGCCCGCACCTGACAGTGCGGGCTTTTTTTTCGACCAAAGG
T
>peak2
AAATCTAACCAACTGGCGCGCGCGGCGGCTCGCCAGGTGGCGGATAACCCTGGCGGTGCCTATAACCCGTTGTTCCTTTA
TGGCGGCACGGGTCTGGGTAAAACTCACCTGCTGCATGCGGTGGGTAACGGCATTATGGCG
```

2.4 Task 3: Obtain enriched gene ontology (GO) terms near the peaks

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")

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.

```
> data(enrichedGO)
```

Here is a list of enriched GO biological process for myPeakList dataset.

```
> enrichedGO$bp[1:6, ]
```

```
       go.id
1 GO:0000187
2 GO:0002573
3 GO:0002702
4 GO:0002761
5 GO:0002763
6 GO:0006213
                                                                    go.term
1                                                  activation of MAPK activity
2                                               myeloid leukocyte differentiation
3 positive regulation of production of molecular mediator of immune response
4                                 regulation of myeloid leukocyte differentiation
5                       positive regulation of myeloid leukocyte differentiation
6                                    pyrimidine nucleoside metabolic process

1
2
3
4
5
6 The chemical reactions and pathways involving any pyrimidine nucleoside, one of a fami
  Ontology count.InDataset count.InGenome      pvalue totaltermInDataset
1       BP              17             65 0.001673400              85892
2       BP              19             81 0.004192510              85892
3       BP               4             10 0.005921074              85892
4       BP              13             50 0.004712934              85892
5       BP               8             22 0.001277580              85892
6       BP               4             10 0.005921074              85892
  totaltermInGenome
1            644151
2            644151
3            644151
4            644151
5            644151
6            644151
```

Here is a list of enriched GO molecular functions for myPeakList dataset.

```
> enrichedGO$mf[1:6, ]
```

```
      go.id                                                        go.term
1 GO:0003702                   RNA polymerase II transcription factor activity
2 GO:0003705 RNA polymerase II transcription factor activity, enhancer binding
3 GO:0004112                      cyclic-nucleotide phosphodiesterase activity
4 GO:0004114               3',5'-cyclic-nucleotide phosphodiesterase activity
5 GO:0004659                                    prenyltransferase activity
6 GO:0004896                                    cytokine receptor activity

1                                  Functions to initiate or regulate RNA polymerase
2 Functions to initiate or regulate RNA polymerase II transcription by binding an enhanc
3               Catalysis of the reaction: a nucleoside cyclic phosphate + H2O = a nucl
4          Catalysis of the reaction: nucleoside 3',5'-cyclic phosphate + H2O = nucleos
5           Catalysis of the transfer of a prenyl group from one compound (donor) to an
6                                    Combining with a cytokine to initiate a change
  Ontology count.InDataset count.InGenome       pvalue totaltermInDataset
1       MF              39            214 0.0065818928              29657
2       MF              11             29 0.0001003699              29657
3       MF               9             26 0.0007622170              29657
4       MF               9             25 0.0005282939              29657
5       MF               9             23 0.0002346785              29657
6       MF              16             66 0.0027160003              29657
  totaltermInGenome
1            235991
2            235991
3            235991
4            235991
5            235991
6            235991
```

Heres is a list of enriched GO cellular components for myPeakList dataset.

```
> enrichedGO$cc
```

```
      go.id                           go.term
1 GO:0005811                     lipid particle
2 GO:0005942 phosphoinositide 3-kinase complex
3 GO:0016363                     nuclear matrix
4 GO:0034399                  nuclear periphery

1                                     Any particle of coalesced lipids in the cytopl
2 A complex containing a heterodimer of a catalytic subunit and a regulatory (adaptor) s
```

```
3                                                                The dense fibrillar network l
4                                                                    The portion of the nuclea
  Ontology count.InDataset count.InGenome      pvalue totaltermInDataset
1       CC               5              15 0.006685158              45317
2       CC               4              11 0.007074546              45317
3       CC              12              49 0.005607016              45317
4       CC              12              52 0.009516449              45317
  totaltermInGenome
1            365523
2            365523
3            365523
4            365523
```

3 References

1. Y. Benjamini and Y. Hochberg (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. J. R. Statist. Soc. B. Vol. 57: 289-300.
2. Y. Benjamini and D. Yekutieli (2001). The control of the false discovery rate in multiple hypothesis testing under dependency. Annals of Statistics. Accepted.
3. S. Durinck et al. (2005) BioMart and Bioconductor: a powerful link between biological biomarts and microarray data analysis. Bioinformatics, 21, 3439-3440.
4. S. Dudoit, J. P. Shaffer, and J. C. Boldrick (Submitted). Multiple hypothesis testing in microarray experiments.
5. Y. Ge, S. Dudoit, and T. P. Speed. Resampling-based multiple testing for microarray data hypothesis, Technical Report #633 of UCB Stat. http://www.stat.berkeley.edu/ gyc
6. R. Gentleman et al. (2004) Bioconductor: open software development for computational biology and bioinformatics. Genome Biol., 5:R80
7. Y. Hochberg (1988). A sharper Bonferroni procedure for multiple tests of significance, Biometrika. Vol. 75: 800-802.
8. S. Holm (1979). A simple sequentially rejective multiple test procedure. Scand. J. Statist.. Vol. 6: 65-70.
9. N. L. Johnson,S. Kotz and A. W. Kemp (1992) Univariate Discrete Distributions, Second Edition. New York: Wiley
10. G. Robertson et al. (2007) Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. Nat Methods, 4:651-7.
11. Zhu L.J. et al. (2010) ChIPpeakAnno: a Bioconductor package to annotate ChIP-seq and ChIP-chip data. BMC Bioinformatics 2010, 11:237doi:10.1186/1471-2105-11-237.

4 Session Info

```
> sessionInfo()
```

```
R version 2.12.0 (2010-10-15)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=C              LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] ChIPpeakAnno_1.6.0                 limma_3.6.0
 [3] org.Hs.eg.db_2.4.6                 GO.db_2.4.5
 [5] RSQLite_0.9-2                      DBI_0.2-5
 [7] AnnotationDbi_1.12.0               BSgenome.Ecoli.NCBI.20080805_1.3.16
 [9] BSgenome_1.18.0                    GenomicRanges_1.2.0
[11] Biostrings_2.18.0                  IRanges_1.8.0
[13] multtest_2.6.0                     Biobase_2.10.0
[15] biomaRt_2.6.0

loaded via a namespace (and not attached):
[1] MASS_7.3-8      RCurl_1.4-3     XML_3.2-0       splines_2.12.0
[5] survival_2.35-8 tools_2.12.0
```