Biconductor Exercises

Gopuraja Dharmalingam

04 February 2015

1. Print few gene names from org.Hs.eg.db

```
library("org.Hs.eg.db")
columns(org.Hs.eg.db)
##  [1] "ENTREZID"     "PFAM"         "IPI"          "PROSITE"
##  [5] "ACCNUM"       "ALIAS"        "CHR"          "CHRLOC"
##  [9] "CHRLOCEND"    "ENZYME"       "MAP"          "PATH"
## [13] "PMID"         "REFSEQ"       "SYMBOL"       "UNIGENE"
## [17] "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS" "GENENAME"
## [21] "UNIPROT"      "GO"           "EVIDENCE"     "ONTOLOGY"
## [25] "GOALL"        "EVIDENCEALL"  "ONTOLOGYALL"  "OMIM"
## [29] "UCSCKG"
head(keys(org.Hs.eg.db,keytype="GENENAME"))
## [1] "alpha-1-B glycoprotein"
## [2] "alpha-2-macroglobulin"
## [3] "alpha-2-macroglobulin pseudogene 1"
## [4] "N-acetyltransferase 1 (arylamine N-acetyltransferase)"
## [5] "N-acetyltransferase 2 (arylamine N-acetyltransferase)"
## [6] "N-acetyltransferase pseudogene"
```

2. Print non-redundant list of chromosomes from org.Mm.eg.db

```
library("org.Mm.eg.db")
unique(keys(org.Mm.eg.db,keytype="CHR"))
##  [1] "6"  "11" "4"  "3"  "2"  "X"  "17" "1"  "7"  "5"  "15" "12" "9"  "8"
## [15] "16" "13" "19" "14" "10" "18" "Y"  "MT" "Un"
```

3. Retrieve gene name, chromosome and Ensembl gene identifiers for "HEBP2" and "PRND" from org.Hs.eg.db

```
select(org.Hs.eg.db,keys=c("HEBP2","PRND"),keytype="SYMBOL",
       columns=c("GENENAME","CHR","ENSEMBL"))
##   SYMBOL                 GENENAME CHR         ENSEMBL
## 1  HEBP2   heme binding protein 2   6 ENSG00000051620
## 2   PRND prion protein 2 (dublet)  20 ENSG00000171864
```

4. Retrieve gene symbol, gene name and gene alias for genes in chromosome 2 for human using org.Hs.eg.db

```
chr2Genes <- select(org.Hs.eg.db,keys="2",keytype="CHR",
                    columns=c("SYMBOL","GENENAME","ALIAS"))
head(chr2Genes)
##   CHR SYMBOL                                  GENENAME ALIAS
## 1   2   AAMP angio-associated, migratory cell protein  AAMP
## 2   2  ACADL       acyl-CoA dehydrogenase, long chain ACAD4
## 3   2  ACADL       acyl-CoA dehydrogenase, long chain  LCAD
## 4   2  ACADL       acyl-CoA dehydrogenase, long chain ACADL
## 5   2   ACP1              acid phosphatase 1, soluble  HAAP
## 6   2   ACP1              acid phosphatase 1, soluble  ACP1
```

5. Retrieve genomic coordinates for human protein coding genes from Ensembl biomart and build GRanges object. Include genes in only main chromosomes (1-22,X,Y).

Tips:

- You can select main chromosomes and "protein coding" genes by using appropriate filter and value.
- Search for "biotype" in available filters using `grep()`
- Run `filterOptions("biotype",selectedmart)` to see the accepted values for "biotype" filter
- When multiple filters specified, "values" argument should be a list of vectors; each vector corresponds to each specified filter.
- Annotation fields to retrieve: "chromosome_name", "start_position", "end_position", "ensembl_gene_id", "strand", "external_gene_name"
- Before creating GRanges object, add "chr" prefix to chromosome using `paste` function, Ex: change 1 to chr1 (required for next task)

```
library("biomaRt")
ensembl <- useMart("ensembl") # select ensembl
ens_datasets <- listDatasets(ensembl) # list datasets
ens_human <- useDataset("hsapiens_gene_ensembl",mart=ensembl) # select human dataset
ens_human_Attr <- listAttributes(ens_human) # list available annotation
ens_human_filters <- listFilters(ens_human) # list available filters
availFilters <- filterOptions("biotype",ens_human) # Displays accepted values for "biotype"

hg19Gene <- getBM(
          attributes = c("chromosome_name","start_position","end_position",
                         "ensembl_gene_id","strand","external_gene_name"),
          filter=c("chromosome_name","biotype"),
          values=list(c(1:22,"X","Y"),"protein_coding"), mart=ens_human)
head(hg19Gene)
##   chromosome_name start_position end_position ensembl_gene_id strand
## 1              20       31465506     31476757 ENSG00000180383     -1
## 2               1        9997206     10016020 ENSG00000162444      1
## 3               X       48186220     48277578 ENSG00000165583     -1
## 4               8       30156297     30183640 ENSG00000104671      1
## 5              20       33407955     33443892 ENSG00000101400     -1
## 6               X       51490011     51496596 ENSG00000196368     -1
##   external_gene_name
## 1            DEFB124
## 2               RBP7
```

```
## 3               SSX5
## 4              DCTN6
## 5              SNTA1
## 6             NUDT11
```

Now create GRanges object using the above data frame.

```
library(GenomicRanges)

# add 'chr' prefix to chromosome name
hg19Gene$chromosome_name <- paste("chr",hg19Gene$chromosome_name,sep="")

hg19Gene.GR <- GRanges(seqnames=hg19Gene$chromosome_name,
                       ranges=IRanges(start=hg19Gene$start_position,end=hg19Gene$end_position),
                       strand=ifelse(hg19Gene$strand==1,"+","-"),
                       EnsemblID=hg19Gene$ensembl_gene_id,
                       Symbol=hg19Gene$external_gene_name)
hg19Gene.GR
## GRanges object with 19850 ranges and 2 metadata columns:
##           seqnames                 ranges strand   |       EnsemblID
##              <Rle>              <IRanges>  <Rle>   |     <character>
##       [1]    chr20   [31465506, 31476757]      -   | ENSG00000180383
##       [2]     chr1   [ 9997206, 10016020]      +   | ENSG00000162444
##       [3]     chrX   [48186220, 48277578]      -   | ENSG00000165583
##       [4]     chr8   [30156297, 30183640]      +   | ENSG00000104671
##       [5]    chr20   [33407955, 33443892]      -   | ENSG00000101400
##       ...      ...                    ...    ... ...             ...
##   [19846]     chr1 [171248471, 171285978]      +   | ENSG00000010932
##   [19847]    chr19 [ 43872363,  43901385]      -   | ENSG00000176222
##   [19848]     chr1 [ 99708703,  99766631]      -   | ENSG00000156869
##   [19849]    chr19 [ 57876765,  57916591]      -   | ENSG00000269476
##   [19850]    chr19 [ 57803841,  57814913]      -   | ENSG00000178935
##                  Symbol
##             <character>
##       [1]       DEFB124
##       [2]          RBP7
##       [3]          SSX5
##       [4]         DCTN6
##       [5]         SNTA1
##       ...           ...
##   [19846]          FMO1
##   [19847]        ZNF404
##   [19848]         FRRS1
##   [19849] CTD-2583A14.9
##   [19850]        ZNF552
##   -------
##   seqinfo: 24 sequences from an unspecified genome; no seqlengths
```

6. Filter the above GRanges object for genes in chr1:1544000-2371000

```
chr1genes <- hg19Gene.GR[seqnames(hg19Gene.GR)=="chr1" &
                         start(hg19Gene.GR) > 1544000 &
                           end(hg19Gene.GR) < 2371000]
head(chr1genes)
## GRanges object with 6 ranges and 2 metadata columns:
##       seqnames             ranges strand |       EnsemblID      Symbol
##          <Rle>          <IRanges>  <Rle> |     <character> <character>
##   [1]     chr1 [2184461, 2212720]      - | ENSG00000162585     C1orf86
##   [2]     chr1 [2019324, 2030751]      + | ENSG00000187730       GABRD
##   [3]     chr1 [2050470, 2185395]      + | ENSG00000067606       PRKCZ
##   [4]     chr1 [1702730, 1724324]      - | ENSG00000008128      CDK11A
##   [5]     chr1 [1615415, 1630610]      + | ENSG00000197530        MIB2
##   [6]     chr1 [1598011, 1600096]      - | ENSG00000228594    C1orf233
##   -------
##   seqinfo: 24 sequences from an unspecified genome; no seqlengths

# alternate
chr1genes <- subset(hg19Gene.GR,start>1544000 & end<2371000 & seqnames=="chr1")
```

7. Create a GRanges of Transcription start sites (1 bp range) for the GRanges object created in Q5.

Tip:

- How to identify TSS for genes in forward/reverse strand?

```
hg19Gene$TSS <- ifelse(hg19Gene$strand==1,hg19Gene$start_position,hg19Gene$end_position)

hg19TSS <- GRanges(seqnames=hg19Gene$chromosome_name,
                       ranges=IRanges(start=hg19Gene$TSS,end=hg19Gene$TSS),
                       strand=ifelse(hg19Gene$strand==1,"+","-"),
                       EnsemblID=hg19Gene$ensembl_gene_id,
                       Symbol=hg19Gene$external_gene_name)
hg19TSS
## GRanges object with 19850 ranges and 2 metadata columns:
##           seqnames                 ranges strand   |       EnsemblID
##              <Rle>              <IRanges>  <Rle>   |     <character>
##       [1]    chr20   [31476757, 31476757]      -   | ENSG00000180383
##       [2]     chr1   [ 9997206,  9997206]      +   | ENSG00000162444
##       [3]     chrX   [48277578, 48277578]      -   | ENSG00000165583
##       [4]     chr8   [30156297, 30156297]      +   | ENSG00000104671
##       [5]    chr20   [33443892, 33443892]      -   | ENSG00000101400
##       ...      ...                    ...    ... ...             ...
##   [19846]     chr1 [171248471, 171248471]      +   | ENSG00000010932
##   [19847]    chr19 [ 43901385,  43901385]      -   | ENSG00000176222
##   [19848]     chr1 [ 99766631,  99766631]      -   | ENSG00000156869
##   [19849]    chr19 [ 57916591,  57916591]      -   | ENSG00000269476
##   [19850]    chr19 [ 57814913,  57814913]      -   | ENSG00000178935
##                  Symbol
##             <character>
##       [1]       DEFB124
```

```
##       [2]          RBP7
##       [3]          SSX5
##       [4]         DCTN6
##       [5]         SNTA1
##       ...           ...
##   [19846]          FMO1
##   [19847]        ZNF404
##   [19848]         FRRS1
##   [19849] CTD-2583A14.9
##   [19850]        ZNF552
##   -------
##   seqinfo: 24 sequences from an unspecified genome; no seqlengths
```

8. Create a GRanges object of human promoters with TSS ± 2000bp (using the GRanges object created in Q5). Tip: Read the documentation for `promoters` function.

```
hg19Promoters <- promoters(hg19Gene.GR,upstream=2000,downstream=2000)
hg19Promoters
## GRanges object with 19850 ranges and 2 metadata columns:
##           seqnames                ranges strand   |       EnsemblID
##              <Rle>             <IRanges>  <Rle>   |     <character>
##       [1]    chr20   [31474758, 31478757]      -   | ENSG00000180383
##       [2]     chr1   [ 9995206,  9999205]      +   | ENSG00000162444
##       [3]     chrX   [48275579, 48279578]      -   | ENSG00000165583
##       [4]     chr8   [30154297, 30158296]      +   | ENSG00000104671
##       [5]    chr20   [33441893, 33445892]      -   | ENSG00000101400
##       ...      ...                   ...    ... ...             ...
##   [19846]     chr1 [171246471, 171250470]      +   | ENSG00000010932
##   [19847]    chr19 [ 43899386,  43903385]      -   | ENSG00000176222
##   [19848]     chr1 [ 99764632,  99768631]      -   | ENSG00000156869
##   [19849]    chr19 [ 57914592,  57918591]      -   | ENSG00000269476
##   [19850]    chr19 [ 57812914,  57816913]      -   | ENSG00000178935
##                Symbol
##             <character>
##       [1]       DEFB124
##       [2]          RBP7
##       [3]          SSX5
##       [4]         DCTN6
##       [5]         SNTA1
##       ...           ...
##   [19846]          FMO1
##   [19847]        ZNF404
##   [19848]         FRRS1
##   [19849] CTD-2583A14.9
##   [19850]        ZNF552
##   -------
##   seqinfo: 24 sequences from an unspecified genome; no seqlengths
```

9. Import ELF1 binding sites in K562 cell from Encode (ELF1_K562.bed) and create GRanges object.

Tips:

- Import the ELF1 binding sites using `import.bed()` function from `rtracklayer` package and compare it with the above GRanges object
- Find ELF1 binding sites overlap with promoters (TSS ± 1kb) using `findOverlaps` and `subsetByOverlaps`
- Remember BED format uses 0-based coordinates

```
library("rtracklayer")
ELF1 <- read.table("ELF1_K562.bed",sep="\t",header=F)
ELF1GR <- GRanges(seqnames=ELF1$V1, IRanges(start=ELF1$V2+1,end=ELF1$V3))
ELF1GR_A <- import.bed("ELF1_K562.bed")


# ELF1 binding sites overlap with promoters using `findOverlaps`
hg19Promoters <- promoters(hg19Gene.GR,upstream=1000,downstream=1000)
ELF1GR <- reduce(ELF1GR) # merging overlapping peaks
ELF1overlap <- findOverlaps(ELF1GR,hg19Promoters,ignore.strand=T)
ELF1overlap.m <- as.matrix(ELF1overlap)
ELF1_promoters <- ELF1GR[ELF1overlap.m[,"queryHits"],]
ELF1_promoters
## GRanges object with 1480 ranges and 0 metadata columns:
##          seqnames                ranges strand
##             <Rle>             <IRanges>  <Rle>
##      [1]     chr1     [ 999543,  999772]      *
##      [2]     chr1     [1115885, 1116069]      *
##      [3]     chr1     [1174741, 1175016]      *
##      [4]     chr1     [1307428, 1307703]      *
##      [5]     chr1     [1310482, 1311094]      *
##      ...      ...                    ...    ...
##   [1476]     chrX [130110006, 130110281]      *
##   [1477]     chrX [153763052, 153763327]      *
##   [1478]     chrX [153775333, 153775532]      *
##   [1479]     chrX [153775672, 153775982]      *
##   [1480]     chrX [153777173, 153777448]      *
##   -------
##   seqinfo: 23 sequences from an unspecified genome; no seqlengths

# ELF1 binding sites overlap with promoters using `subsetByOverlaps`
ELF1_promoters1 <- subsetByOverlaps(ELF1GR,hg19Promoters)
ELF1_promoters1
## GRanges object with 1303 ranges and 0 metadata columns:
##          seqnames                ranges strand
##             <Rle>             <IRanges>  <Rle>
##      [1]     chr1     [ 999543,  999772]      *
##      [2]     chr1     [1115885, 1116069]      *
##      [3]     chr1     [1174741, 1175016]      *
##      [4]     chr1     [1307428, 1307703]      *
##      [5]     chr1     [1310482, 1311094]      *
##      ...      ...                    ...    ...
```

```
##   [1299]     chrX [130110006, 130110281]      *
##   [1300]     chrX [153763052, 153763327]      *
##   [1301]     chrX [153775333, 153775532]      *
##   [1302]     chrX [153775672, 153775982]      *
##   [1303]     chrX [153777173, 153777448]      *
##   -------
##   seqinfo: 23 sequences from an unspecified genome; no seqlengths
```

Note the differences in the outputs!

10. Retrieve the transcript coordinates for genes as GRangesList from `TxDb.Hsapiens.UCSC.hg19.knownGene` (install it from Bioconductor if required)

```
# source("http://bioconductor.org/biocLite.R")
# biocLite("TxDb.Hsapiens.UCSC.hg19.knownGene")
library("TxDb.Hsapiens.UCSC.hg19.knownGene")
hg19txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
TranscrtipsByGene <- transcriptsBy(hg19txdb,by="gene") # inspect the output
```

11. Retrieve 200 bp upstream promoter sequences for the given gene symbols `AQP1, ASNSP2, KPNA2, FRMD4A, NSUN5, VAC14` from Ensembl human biomart

Tips:

- Read documentation for `getSequence`
- Use `type="hgnc_symbol"` and `seqType="coding_gene_flank"`

```
symbols <- c("AQP1", "ASNSP2", "KPNA2", "FRMD4A", "NSUN5", "VAC14")
seq = getSequence(id=symbols,
                  type="hgnc_symbol",
                  seqType="coding_gene_flank",upstream=200, mart = ens_human)
```

Additional Exercise

- Download the following BAM and index files (*.bai) (ENCODE data - ChIP-Seq of CTCF in Ag04449 human fibroblast cells)
  - http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeUwTfbs/wgEncodeUwTfbsAg04449CtcfS bam
  - http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeUwTfbs/wgEncodeUwTfbsAg04449CtcfS bam.bai
- Use `readGAlignments` to read the bam. Construct an `ScanBamParam` object that accepts only aligned reads, passing quality control and not duplicates.
- Compute genome wide coverage using `coverage` function
- Compute number of reads overlapping with hg19 promoters (TSS $\pm$ 1kb) and export the results as text file.

```
library("GenomicAlignments")
bamFile <- "wgEncodeUwTfbsAg04449CtcfStdAlnRep1.bam"
flag <- scanBamFlag()
param <- ScanBamParam(
    flag=scanBamFlag(isUnmappedQuery=FALSE, isDuplicate=FALSE, isNotPassingQualityControls=FALSE)
    )

# Read the BAM
CTCF <- readGAlignments(bamFile,param=param)

# Generate the genomic coverage and inspect the output
CTCFCov <- coverage(CTCF)

# Reads overlapping with promoters
CTCFCounts <- countOverlaps(hg19Promoters,CTCF)

# Add CTCF counts as elementMetadata to hg19Promoters object
mcols(hg19Promoters)$CTCF <- CTCFCounts

# Export the results as text file
hg19Promoters.df <- as.data.frame(hg19Promoters)
write.table(hg19Promoters.df,"hg19Promoters_CTCF.txt",sep="\t",row.names=F)
```