# Bioconductor Basics: Granges and Biostrings

- Core Bioconductor structures for representing genes and genetic sequences

## Motivation and Introduction

- Case study: given genomic DNA extracted from human cells, where on the genome does the nuclear protein ESRRA (estrogen related receptor alpha) bind?
- Role of estrogen receptors in breast cancer
- Data comes from analysis of ChIP-seq experiments: performed in ENCODE project - import info for files in "narrowPeak" format and analyze in Bioconductor GRanges object
- Identifying nearest transcriptional start site for each binding peak - assess whether regulatory activity of ESRRA occurs in transcriptional promoter regions

```
library(ERBS)
data(HepG2)
class(HepG2)
```

```
## [1] "GRanges"
## attr(,"package")
## [1] "GenomicRanges"
```

## GenomicRanges

- ERBS library from github repo
- Load two datasets - GM12878, HepG2. Estrogen receptor binding site datasets from two cell lines (cell-type dependent outcome).
- Contains: Chromosome start + end (1 row / region), strand information, score from peaks
- Access the GRanges objects as a matrix, i.e. subsetting is okay.
- **seqnames** function to access chromosome for each row. Returns object of type *Rle* - more efficient to save ordered by chromosome with counts. Can turn into character using **as.character**
- Most of analysis is focused on first 23 chromosomes
- Function to order by genomic region
- Iranges function not specific to genomics - Granges builds on Iranges in relation to genomics

```
# install ERBS
library(devtools)
```

```
## Loading required package: usethis
```

```
install_github("genomicsclass/ERBS")
```

```
## Skipping install of 'ERBS' from a github remote, the SHA1 (9f16eb6a) has not changed since last insta
##   Use `force = TRUE` to force installation
```

```
library(GenomicRanges)
```

```
## Loading required package: stats4
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:base':
##
##     expand.grid

## Loading required package: IRanges

## Loading required package: GenomeInfoDb
```

```
# load GM12878 and HepG2 objects from ERBS package
library(ERBS)
data(GM12878)
data(HepG2)

# inspect HepG2 GRanges object
class(HepG2)
```

```
## [1] "GRanges"
## attr(,"package")
## [1] "GenomicRanges"
```

```
HepG2
```

```
## GRanges object with 303 ranges and 7 metadata columns:
##         seqnames              ranges strand |      name     score       col
##            <Rle>           <IRanges>  <Rle> | <numeric> <integer> <logical>
##     [1]     chr2   20335378-20335787      * |        NA         0      <NA>
##     [2]    chr20       328285-329145      * |        NA         0      <NA>
##     [3]     chr6 168135432-168136587      * |        NA         0      <NA>
##     [4]    chr19     1244419-1245304      * |        NA         0      <NA>
##     [5]    chr11   64071828-64073069      * |        NA         0      <NA>
##     ...      ...                 ...    ... .       ...       ...       ...
##   [299]     chr4     1797182-1797852      * |        NA         0      <NA>
##   [300]     chr1 110198573-110199126      * |        NA         0      <NA>
##   [301]    chr17   17734052-17734469      * |        NA         0      <NA>
##   [302]     chr1   48306453-48306908      * |        NA         0      <NA>
```

```
##   [303]    chr12 123867207-123867554      * |        NA         0      <NA>
##         signalValue    pValue      qValue      peak
##           <numeric> <numeric>   <numeric> <integer>
##     [1]      58.251    75.899 6.14371e-72       195
##     [2]      10.808    69.685 5.02806e-66       321
##     [3]      17.103    54.311 7.93067e-51       930
##     [4]      12.427    43.855 1.35976e-40       604
##     [5]      10.850    40.977 7.33386e-38       492
##     ...         ...       ...         ...       ...
##   [299]       9.681    10.057 1.42334e-08       402
##   [300]       7.929    10.047 1.44208e-08       197
##   [301]       5.864     9.990 1.63892e-08       227
##   [302]       5.660     9.948 1.79941e-08       211
##   [303]      13.211     9.918 1.92180e-08       163
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

```
values(HepG2)
```

```
## DataFrame with 303 rows and 7 columns
##          name     score       col signalValue    pValue      qValue      peak
##     <numeric> <integer> <logical>   <numeric> <numeric>   <numeric> <integer>
## 1          NA         0        NA      58.251    75.899 6.14371e-72       195
## 2          NA         0        NA      10.808    69.685 5.02806e-66       321
## 3          NA         0        NA      17.103    54.311 7.93067e-51       930
## 4          NA         0        NA      12.427    43.855 1.35976e-40       604
## 5          NA         0        NA      10.850    40.977 7.33386e-38       492
## ...       ...       ...       ...         ...       ...         ...       ...
## 299        NA         0        NA       9.681    10.057 1.42334e-08       402
## 300        NA         0        NA       7.929    10.047 1.44208e-08       197
## 301        NA         0        NA       5.864     9.990 1.63892e-08       227
## 302        NA         0        NA       5.660     9.948 1.79941e-08       211
## 303        NA         0        NA      13.211     9.918 1.92180e-08       163
```

```
# seqnames extracts chromosome names
seqnames(HepG2)     # stored as type Rle
```

```
## factor-Rle of length 303 with 292 runs
##   Lengths:     1     1     1     1     1 ...     1     1     1     1     1
##   Values : chr2  chr20 chr6  chr19 chr11 ... chr4  chr1  chr17 chr1  chr12
## Levels(93): chr1 chr2 chr3 ... chrUn_gl000247 chrUn_gl000248 chrUn_gl000249
```

```
chr = seqnames(HepG2)
as.character(chr)    # view as character type
```

```
##   [1] "chr2"  "chr20" "chr6"  "chr19" "chr11" "chr20" "chr19" "chr2"  "chr16"
##  [10] "chr3"  "chr6"  "chr20" "chr7"  "chr16" "chr9"  "chr11" "chr22" "chrX"
##  [19] "chr8"  "chr16" "chr16" "chr19" "chr17" "chr17" "chr16" "chr1"  "chr16"
##  [28] "chr9"  "chr17" "chr16" "chr12" "chr6"  "chr2"  "chr3"  "chr11" "chr16"
##  [37] "chr6"  "chr2"  "chr8"  "chr1"  "chr17" "chr20" "chr4"  "chr14" "chr19"
##  [46] "chr20" "chr9"  "chr2"  "chr2"  "chr19" "chr8"  "chr14" "chr22" "chr2"
##  [55] "chr14" "chr6"  "chr20" "chr2"  "chr19" "chr8"  "chr2"  "chr19" "chr12"
##  [64] "chr2"  "chr2"  "chr11" "chr12" "chr7"  "chr19" "chr22" "chr17" "chr3"
##  [73] "chr8"  "chr3"  "chr15" "chr6"  "chr9"  "chr10" "chr6"  "chr2"  "chr19"
##  [82] "chr11" "chr8"  "chr17" "chr15" "chr21" "chr7"  "chr2"  "chr2"  "chr3"
##  [91] "chr2"  "chr16" "chr10" "chr20" "chr17" "chr13" "chr2"  "chr5"  "chr14"
```

```
## [100] "chr11" "chr8"  "chr20" "chr3"  "chr7"  "chr1"  "chr1"  "chr3"  "chr17"
## [109] "chrX"  "chr19" "chr20" "chr6"  "chr7"  "chr16" "chr7"  "chr17" "chr20"
## [118] "chr2"  "chr5"  "chrX"  "chr7"  "chr6"  "chr19" "chr17" "chr16" "chr5"
## [127] "chr12" "chr9"  "chr20" "chr2"  "chr12" "chr3"  "chr7"  "chr2"  "chr20"
## [136] "chr20" "chr17" "chr12" "chr19" "chr1"  "chr7"  "chr20" "chr14" "chr12"
## [145] "chr10" "chr6"  "chr9"  "chr6"  "chr1"  "chr18" "chr8"  "chr15" "chr6"
## [154] "chr2"  "chr1"  "chr18" "chr16" "chr9"  "chr20" "chr19" "chr17" "chr10"
## [163] "chr6"  "chr2"  "chrX"  "chr16" "chr20" "chr16" "chr20" "chr16" "chr20"
## [172] "chr5"  "chr16" "chr17" "chr17" "chr3"  "chr8"  "chr18" "chr18" "chr7"
## [181] "chr20" "chr16" "chr19" "chr11" "chr12" "chr2"  "chr17" "chr1"  "chr20"
## [190] "chr4"  "chr17" "chr1"  "chr6"  "chr5"  "chr13" "chr7"  "chr20" "chr2"
## [199] "chr16" "chr6"  "chr11" "chr5"  "chr20" "chr1"  "chr9"  "chr2"  "chr16"
## [208] "chr10" "chr9"  "chr2"  "chr2"  "chr21" "chr1"  "chr16" "chr18" "chr10"
## [217] "chr16" "chr3"  "chr6"  "chr16" "chr2"  "chr6"  "chr10" "chr16" "chr22"
## [226] "chr2"  "chr16" "chr8"  "chr20" "chr19" "chr16" "chr20" "chr2"  "chr3"
## [235] "chr10" "chr14" "chr6"  "chr18" "chr15" "chr9"  "chr14" "chr7"  "chr20"
## [244] "chr3"  "chr6"  "chr10" "chr4"  "chr1"  "chr9"  "chr15" "chr6"  "chr16"
## [253] "chr2"  "chr3"  "chr14" "chr19" "chr2"  "chr5"  "chr22" "chr16" "chr6"
## [262] "chr16" "chr17" "chr11" "chr8"  "chr3"  "chr1"  "chr16" "chr21" "chr12"
## [271] "chr16" "chr1"  "chr2"  "chr2"  "chr9"  "chr2"  "chr16" "chr17" "chr12"
## [280] "chr17" "chr7"  "chr20" "chr7"  "chr6"  "chr12" "chr2"  "chr1"  "chr5"
## [289] "chr6"  "chr2"  "chr1"  "chr12" "chr2"  "chr6"  "chr20" "chr2"  "chr17"
## [298] "chr3"  "chr4"  "chr1"  "chr17" "chr1"  "chr12"
```

```
# make a table of numbers of sequences on each chromosome
table(chr)
```

```
## chr
##                  chr1                  chr2                  chr3
##                    18                    38                    15
##                  chr4                  chr5                  chr6
##                     4                     8                    24
##                  chr7                  chr8                  chr9
##                    14                    11                    12
##                 chr10                 chr11                 chr12
##                     9                     9                    13
##                 chr13                 chr14                 chr15
##                     2                     8                     5
##                 chr16                 chr17                 chr18
##                    31                    21                     6
##                 chr19                 chr20                 chr21
##                    16                    27                     3
##                 chr22                  chrX                  chrY
##                     5                     4                     0
##                  chrM  chr1_gl000191_random  chr1_gl000192_random
##                     0                     0                     0
##        chr4_ctg9_hap1  chr4_gl000193_random  chr4_gl000194_random
##                     0                     0                     0
##         chr6_apd_hap1         chr6_cox_hap2         chr6_dbb_hap3
##                     0                     0                     0
##        chr6_mann_hap4         chr6_mcf_hap5         chr6_qbl_hap6
##                     0                     0                     0
##        chr6_ssto_hap7  chr7_gl000195_random  chr8_gl000196_random
##                     0                     0                     0
##  chr8_gl000197_random  chr9_gl000198_random  chr9_gl000199_random
```

```
##                     0                     0                     0
##  chr9_gl000200_random  chr9_gl000201_random chr11_gl000202_random
##                     0                     0                     0
##       chr17_ctg5_hap1 chr17_gl000203_random chr17_gl000204_random
##                     0                     0                     0
## chr17_gl000205_random chr17_gl000206_random chr18_gl000207_random
##                     0                     0                     0
## chr19_gl000208_random chr19_gl000209_random chr21_gl000210_random
##                     0                     0                     0
##        chrUn_gl000211        chrUn_gl000212        chrUn_gl000213
##                     0                     0                     0
##        chrUn_gl000214        chrUn_gl000215        chrUn_gl000216
##                     0                     0                     0
##        chrUn_gl000217        chrUn_gl000218        chrUn_gl000219
##                     0                     0                     0
##        chrUn_gl000220        chrUn_gl000221        chrUn_gl000222
##                     0                     0                     0
##        chrUn_gl000223        chrUn_gl000224        chrUn_gl000225
##                     0                     0                     0
##        chrUn_gl000226        chrUn_gl000227        chrUn_gl000228
##                     0                     0                     0
##        chrUn_gl000229        chrUn_gl000230        chrUn_gl000231
##                     0                     0                     0
##        chrUn_gl000232        chrUn_gl000233        chrUn_gl000234
##                     0                     0                     0
##        chrUn_gl000235        chrUn_gl000236        chrUn_gl000237
##                     0                     0                     0
##        chrUn_gl000238        chrUn_gl000239        chrUn_gl000240
##                     0                     0                     0
##        chrUn_gl000241        chrUn_gl000242        chrUn_gl000243
##                     0                     0                     0
##        chrUn_gl000244        chrUn_gl000245        chrUn_gl000246
##                     0                     0                     0
##        chrUn_gl000247        chrUn_gl000248        chrUn_gl000249
##                     0                     0                     0
```

```
table(chr)[1:24]    # restrict to autosomes, X and Y
```

```
## chr
##  chr1  chr2  chr3  chr4  chr5  chr6  chr7  chr8  chr9 chr10 chr11 chr12 chr13
##    18    38    15     4     8    24    14    11    12     9     9    13     2
## chr14 chr15 chr16 chr17 chr18 chr19 chr20 chr21 chr22  chrX  chrY
##     8     5    31    21     6    16    27     3     5     4     0
```

```
# GRanges can be subsetted and ordered
HepG2[chr=="chr20",]
```

```
## GRanges object with 27 ranges and 7 metadata columns:
##        seqnames            ranges strand |      name     score       col
##           <Rle>         <IRanges>  <Rle> | <numeric> <integer> <logical>
##    [1]    chr20     328285-329145      * |        NA         0      <NA>
##    [2]    chr20 22410891-22411863      * |        NA         0      <NA>
##    [3]    chr20 56039583-56040249      * |        NA         0      <NA>
##    [4]    chr20 16455811-16456232      * |        NA         0      <NA>
##    [5]    chr20   3140243-3140774      * |        NA         0      <NA>
```

```
##    ...      ...                 ...    ... .        ...        ...        ...
##   [23]    chr20   5591571-5592037      * |         NA          0       <NA>
##   [24]    chr20 25519664-25520238      * |         NA          0       <NA>
##   [25]    chr20 19900951-19901275      * |         NA          0       <NA>
##   [26]    chr20 35156796-35157140      * |         NA          0       <NA>
##   [27]    chr20 25036720-25037716      * |         NA          0       <NA>
##        signalValue    pValue      qValue      peak
##          <numeric> <numeric>   <numeric> <integer>
##    [1]      10.808    69.685 5.02806e-66       321
##    [2]       6.419    41.020 7.74961e-38       660
##    [3]       7.796    36.977 3.66693e-34       315
##    [4]       7.351    21.831 1.59668e-19       199
##    [5]       7.296    21.587 2.62536e-19       315
##    ...         ...       ...         ...       ...
##   [23]       8.766    11.433 7.67742e-10       249
##   [24]       3.300    11.419 7.89520e-10       206
##   [25]       4.809    11.155 1.37954e-09       140
##   [26]      10.154    10.313 8.30971e-09       163
##   [27]       4.381    10.087 1.33278e-08       170
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

```
x = HepG2[order(HepG2),]
seqnames(x)     # demonstrate usefulness of Rle type
```

```
## factor-Rle of length 303 with 23 runs
##   Lengths:    18    38    15     4     8 ...    16    27     3     5     4
##   Values : chr1  chr2  chr3  chr4  chr5  ... chr19 chr20 chr21 chr22 chrX
## Levels(93): chr1 chr2 chr3 ... chrUn_gl000247 chrUn_gl000248 chrUn_gl000249
```

```
as.character(seqnames(x))
```

```
##   [1] "chr1"  "chr1"  "chr1"  "chr1"  "chr1"  "chr1"  "chr1"  "chr1"  "chr1"
##  [10] "chr1"  "chr1"  "chr1"  "chr1"  "chr1"  "chr1"  "chr1"  "chr1"  "chr1"
##  [19] "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"
##  [28] "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"
##  [37] "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"
##  [46] "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"  "chr2"
##  [55] "chr2"  "chr2"  "chr3"  "chr3"  "chr3"  "chr3"  "chr3"  "chr3"  "chr3"
##  [64] "chr3"  "chr3"  "chr3"  "chr3"  "chr3"  "chr3"  "chr3"  "chr3"  "chr4"
##  [73] "chr4"  "chr4"  "chr4"  "chr5"  "chr5"  "chr5"  "chr5"  "chr5"  "chr5"
##  [82] "chr5"  "chr5"  "chr6"  "chr6"  "chr6"  "chr6"  "chr6"  "chr6"  "chr6"
##  [91] "chr6"  "chr6"  "chr6"  "chr6"  "chr6"  "chr6"  "chr6"  "chr6"  "chr6"
## [100] "chr6"  "chr6"  "chr6"  "chr6"  "chr6"  "chr6"  "chr6"  "chr6"  "chr7"
## [109] "chr7"  "chr7"  "chr7"  "chr7"  "chr7"  "chr7"  "chr7"  "chr7"  "chr7"
## [118] "chr7"  "chr7"  "chr7"  "chr7"  "chr8"  "chr8"  "chr8"  "chr8"  "chr8"
## [127] "chr8"  "chr8"  "chr8"  "chr8"  "chr8"  "chr8"  "chr9"  "chr9"  "chr9"
## [136] "chr9"  "chr9"  "chr9"  "chr9"  "chr9"  "chr9"  "chr9"  "chr9"  "chr9"
## [145] "chr10" "chr10" "chr10" "chr10" "chr10" "chr10" "chr10" "chr10" "chr10"
## [154] "chr11" "chr11" "chr11" "chr11" "chr11" "chr11" "chr11" "chr11" "chr11"
## [163] "chr12" "chr12" "chr12" "chr12" "chr12" "chr12" "chr12" "chr12" "chr12"
## [172] "chr12" "chr12" "chr12" "chr12" "chr13" "chr13" "chr14" "chr14" "chr14"
## [181] "chr14" "chr14" "chr14" "chr14" "chr14" "chr15" "chr15" "chr15" "chr15"
## [190] "chr15" "chr16" "chr16" "chr16" "chr16" "chr16" "chr16" "chr16" "chr16"
## [199] "chr16" "chr16" "chr16" "chr16" "chr16" "chr16" "chr16" "chr16" "chr16"
```

```
## [208] "chr16" "chr16" "chr16" "chr16" "chr16" "chr16" "chr16" "chr16" "chr16"
## [217] "chr16" "chr16" "chr16" "chr16" "chr16" "chr17" "chr17" "chr17" "chr17"
## [226] "chr17" "chr17" "chr17" "chr17" "chr17" "chr17" "chr17" "chr17" "chr17"
## [235] "chr17" "chr17" "chr17" "chr17" "chr17" "chr17" "chr17" "chr17" "chr18"
## [244] "chr18" "chr18" "chr18" "chr18" "chr18" "chr19" "chr19" "chr19" "chr19"
## [253] "chr19" "chr19" "chr19" "chr19" "chr19" "chr19" "chr19" "chr19" "chr19"
## [262] "chr19" "chr19" "chr19" "chr20" "chr20" "chr20" "chr20" "chr20" "chr20"
## [271] "chr20" "chr20" "chr20" "chr20" "chr20" "chr20" "chr20" "chr20" "chr20"
## [280] "chr20" "chr20" "chr20" "chr20" "chr20" "chr20" "chr20" "chr20" "chr20"
## [289] "chr20" "chr20" "chr20" "chr21" "chr21" "chr21" "chr22" "chr22" "chr22"
## [298] "chr22" "chr22" "chrX"  "chrX"  "chrX"  "chrX"
```

**Assessment: Genomic Ranges**

```
library(GenomicRanges)
paste("median of signal value column for HepG2 data: ")
```

```
## [1] "median of signal value column for HepG2 data: "
```

```
median(mcols(HepG2)$signalValue)
```

```
## [1] 7.024
```

```
paste("chromosome in region with highest signal value: ")
```

```
## [1] "chromosome in region with highest signal value: "
```

```
max_index <- which.max(mcols(HepG2)$signalValue)
chr = seqnames(HepG2)
as.character(chr)[max_index]
```

```
## [1] "chrX"
```

```
paste("Number of regions from chromosome 16: ")
```

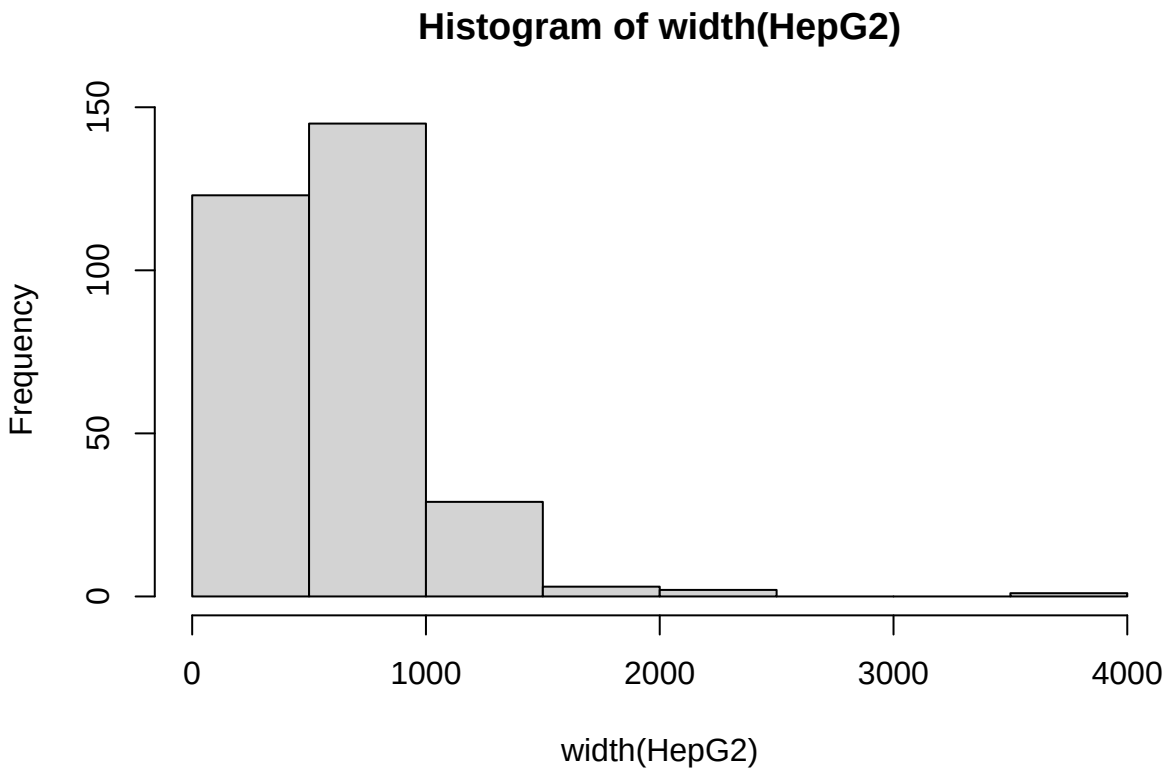
```
## [1] "Number of regions from chromosome 16: "
```

```
HepG2[chr == "chr16",]
```

```
## GRanges object with 31 ranges and 7 metadata columns:
##        seqnames            ranges strand |      name     score       col
##           <Rle>         <IRanges>  <Rle> | <numeric> <integer> <logical>
##    [1]    chr16 70191209-70192150      * |        NA         0      <NA>
##    [2]    chr16   1701039-1702137      * |        NA         0      <NA>
##    [3]    chr16 25189109-25190026      * |        NA         0      <NA>
##    [4]    chr16 85325101-85325686      * |        NA         0      <NA>
##    [5]    chr16 29986461-29986872      * |        NA         0      <NA>
##    ...      ...               ...    ... .       ...       ...       ...
##   [27]    chr16 57481218-57481854      * |        NA         0      <NA>
##   [28]    chr16 85322504-85322950      * |        NA         0      <NA>
##   [29]    chr16 19134897-19135280      * |        NA         0      <NA>
##   [30]    chr16   2586101-2586737      * |        NA         0      <NA>
##   [31]    chr16 29975932-29976255      * |        NA         0      <NA>
##        signalValue    pValue      qValue      peak
##          <numeric> <numeric>   <numeric> <integer>
##    [1]       8.371    37.774 8.19277e-35       688
##    [2]      16.157    36.264 1.65696e-33       783
```

```
##     [3]       5.979    31.808 3.44356e-29       606
##     [4]       7.664    31.429 7.88321e-29       223
##     [5]      14.795    29.018 1.73008e-26       198
##     ...         ...       ...         ...       ...
##    [27]       5.126    10.761 3.20978e-09       196
##    [28]       4.331    10.725 3.43494e-09       223
##    [29]       5.380    10.562 4.92563e-09       203
##    [30]       6.521    10.514 5.42123e-09       472
##    [31]       6.897    10.436 6.37196e-09       145
##    -------
##    seqinfo: 93 sequences (1 circular) from hg19 genome
```

```
hist(width(HepG2))
```




```
median_width <- median(width(HepG2))
paste("Median width of all chromosomes: ", median_width)
```

```
## [1] "Median width of all chromosomes:  560"
```

### Bioconductor Infrastructure for genomics, microarray and NGS

- IRanges package - representing ranges of integers. Base pair arrangements we want to manipulate in genomics
- Vignette about classes and functions in IRanges package
- Simple functions have good performance
- Summary of most important functions
- IRanges - start, end, width (i.e., 5, 10, 6bP long)

- Start, end, and width functions
- Can specify > 1 range at a time to make IRanges objects of length n
- Intra-range methods:
- **Shift** - Intra range methods for IRanges - doesn't depend on other ranges contained in IRanges object. I.e., shift IRange to the left by 2.
- **Narrow** - relative to start, start at nth base pair
- **Flank** - get flanking sequence 3 base pairs from start or end (start = False). Also bi-directional (both=True)
- Inter-range methods:
- **range** - will give beginning of the IRanges to the end, including gaps in between
- **reduce** - gives us base pairs covered by the original ranges (do not get gaps). Can ask for gaps.
- **disjoint** - set of ranges which has the same coverage as original IRanges object but non-overlapping. Contain union of all endpoints of the original range.

**Assessment: IRanges**

```
library(IRanges)
ir <- IRanges(101, 200)
paste("*2 zooms in, giving range with half the width. New starting point: ", start(ir*2))
```

```
## [1] "*2 zooms in, giving range with half the width. New starting point:  126"
```

```
n_ir <- narrow(ir, start=20)
paste("narrow function with start of 20. New starting point: ", start(n_ir))
```

```
## [1] "narrow function with start of 20. New starting point:  120"
```

```
paste("+25 operation gives width of resulting range: ", width(ir+25))
```

```
## [1] "+25 operation gives width of resulting range:  150"
```

```
m_ir <- IRanges(start=c(1, 11, 21),end=c(3, 15, 27))
paste("sum of widths of multiple IRanges objects:", sum(width(m_ir)))
```

```
## [1] "sum of widths of multiple IRanges objects: 15"
```

```
x <- IRanges(start=c(101,106,201,211,221,301,306,311,351,361,401,411,501), end=c(150,160,210,270,225,31
library(ph525x)
```

```
## Loading required package: png
```

```
## Loading required package: grid
```

```
## Loading required package: Biobase
```

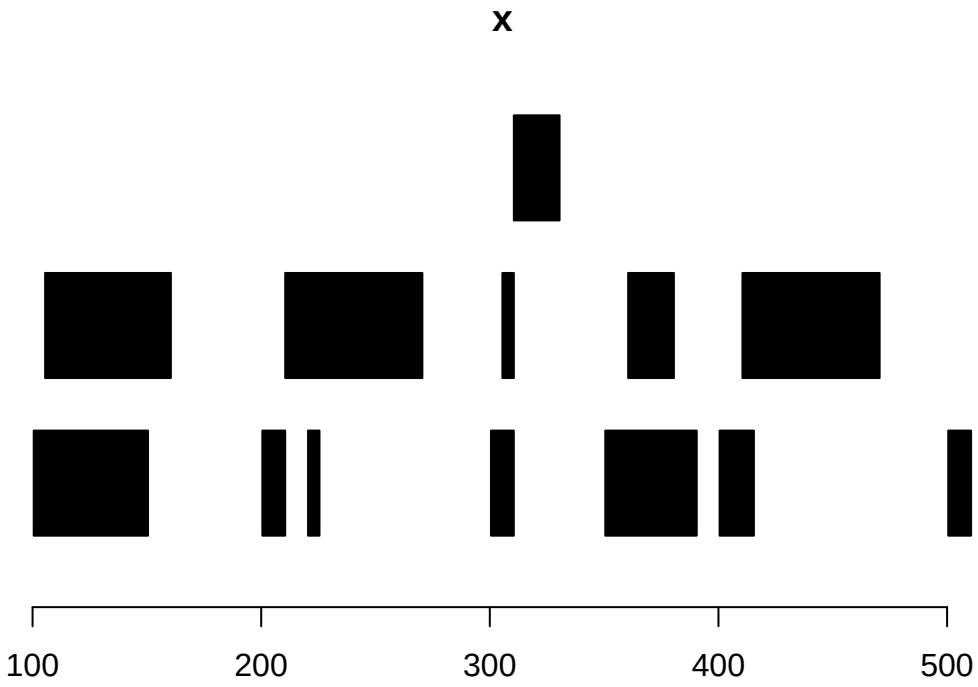
```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
## Loading required package: Homo.sapiens
```

```
## Loading required package: AnnotationDbi
```

```
## Loading required package: OrganismDbi
## Loading required package: GenomicFeatures
## Loading required package: GO.db
##
## Loading required package: org.Hs.eg.db
##
## Loading required package: TxDb.Hsapiens.UCSC.hg19.knownGene
```

```
plotRanges(x)
```



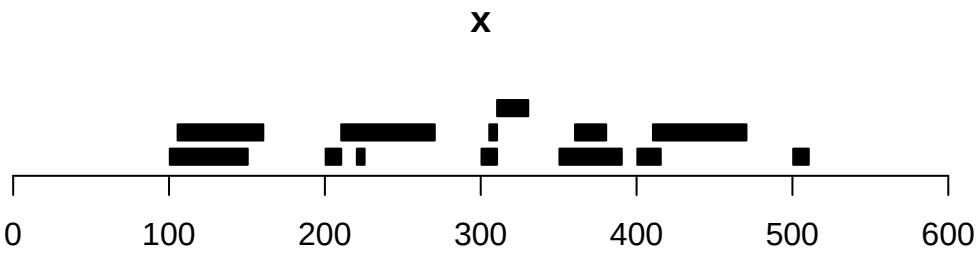

```
paste("Total width not covered by ranges in x:", sum(width(gaps(x))))
```

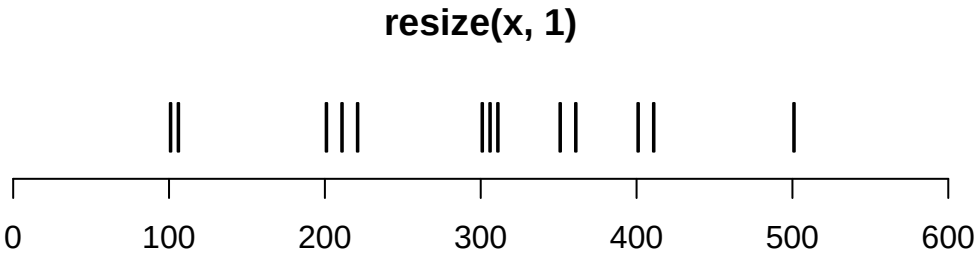
```
## [1] "Total width not covered by ranges in x: 130"
```

```
paste("Number of disjoint ranges within ranges in x:", length(disjoin(x)))
```

```
## [1] "Number of disjoint ranges within ranges in x: 17"
```

```
par(mfrow=c(2, 1))
plotRanges(x, xlim=c(0, 600))
plotRanges(resize(x, 1), xlim=c(0, 600))
```




### Genomic ranges: GRanges

- Extension of IRanges
- Contain a sequence name - IRange of chromosome Z.
- Can contain chromosome information and sequence length
- Sequence names as Rle
- IRanges and strange as Rle also
- Can shift similar to IRanges - will go off end of chromosome if exceeds length
- Wrap in trim function to make sure that the end at chromosome end does not exceed
- Metadata accessed with *mcols*
- Can add cols by *mcols$*
- Additional package called *GRangesList* - groups GRanges together by wrapping in function call
- Example of *GRangesList* - grouping exons by gene or by transcript
- Application of package - find overlaps between GRanges objects
- *findOverlaps* function - query and subject (see in help() function)
- output of *findOverlaps* is a hits object with length representing # overlaps
- Same way to get the overlaps is *%over%* function - which returns logical vector
- *Rle* object defined by IRanges but similar object in base R = Run length encoding
- If vector repeats certain values, can save memory by number and number of times repeated
- *str* function gives us the compact representation
- Peering into *Rle* object - can use *Views* object to see *IRanges* from start to end. Only a virtual class - saves *Rle* and number of views / windows into it
- Can also use for *Fasta* files or other objects

#### Assessment: GRanges

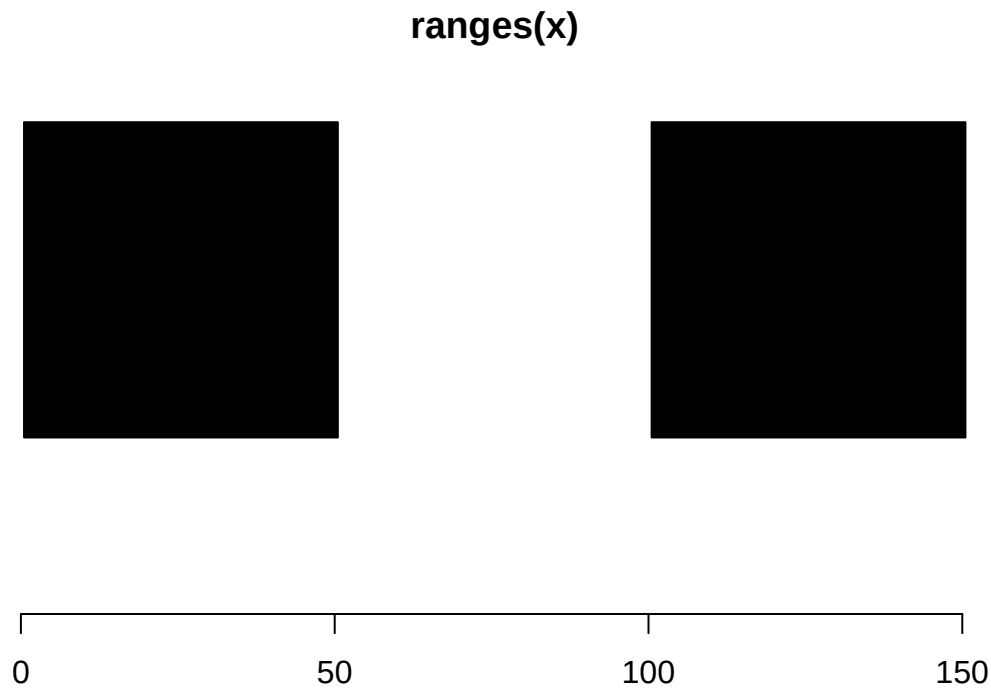
- GRanges object extends concept of interval ranges
- Ranges can be defined by:
  - chromosome we are referring to (seqnames in Bioconductor)
  - strand of DNA we are referring to (+ or -)

- These two pieces of information are necessary for specification of a range of DNA

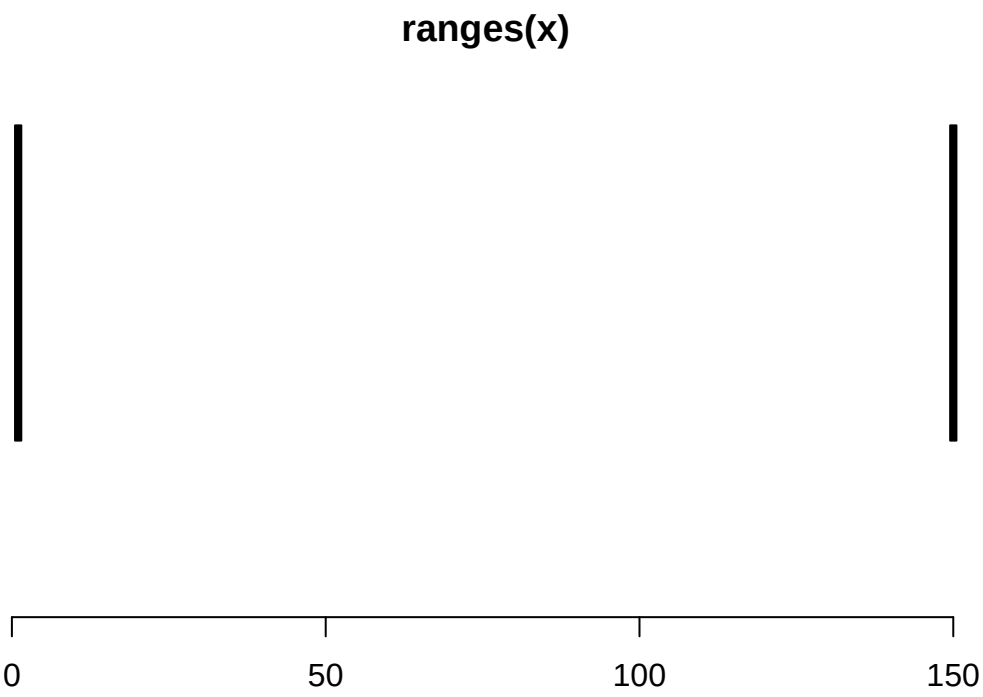
```
library(GenomicRanges)
library(IRanges)
library(ph525x)
x = GRanges("chr1", IRanges(c(1,101),c(50,150)), strand=c("+","-"))
paste("Get the internal IRanges from a GRanges object: ")
```

```
## [1] "Get the internal IRanges from a GRanges object: "
```

```
plotGRanges = function(x) plotRanges(ranges(x))
plotGRanges(x)
```




```
plotGRanges(resize(x, 1))
```



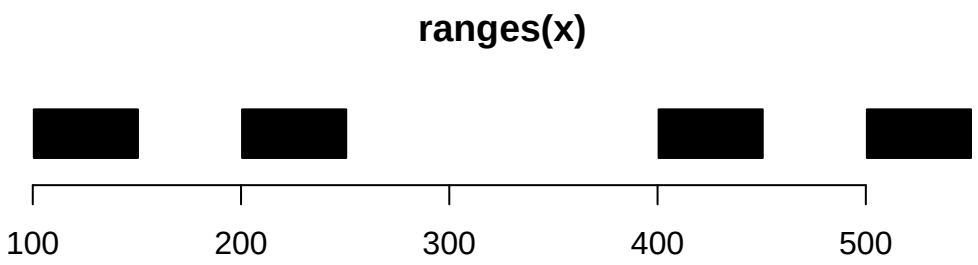

```
paste("Result of running resize(x, 1) is two ranges of width 1 which start at left-most point of the + s
      and the right-most point of the - strand ranges in x (+ -> -)")
```

```
## [1] "Result of running resize(x, 1) is two ranges of width 1 which start at left-most point of the +
```

```
paste("Q2: Intersecting transcripts with basic operations")
```

```
## [1] "Q2: Intersecting transcripts with basic operations"
```

```
x = GRanges("chr1", IRanges(c(101, 201, 401, 501), c(150, 250, 450, 550)), strand="+")
y = GRanges("chr1", IRanges(c(101, 221, 301, 401, 541), c(150, 250, 350, 470, 550)), strand="+")
par(mfrow=c(2,1))
plotGRanges(x)
plotGRanges(y)
```





```
multiple_granges_list <- GRangesList(x,y)
single_granges_list <- GRangesList(c(x, y))

findOverlaps(x, y)
```

```
## Hits object with 4 hits and 0 metadata columns:
##       queryHits subjectHits
##       <integer>   <integer>
##   [1]         1           1
##   [2]         2           2
##   [3]         3           4
##   [4]         4           5
##   -------
##   queryLength: 4 / subjectLength: 5
```

```
paste("width of overlap between x and y: ", sum(width(union(x, y))) - sum(width(intersect(x, y))))
```

```
## [1] "width of overlap between x and y:  130"
```

```
z = GRanges("chr1", range(ranges(x)), strand="-")
```

### Operating on GRanges

- Small set of ranges = intervals on chromosome
- Operations:
  - *reduce* - project all of the occupied bases into contiguous intervals and leaves empty parts with no coverage
  - *disjoin* - set of intervals / ranges generated by disjoin of set of ranges. Same occupancy as original GRanges object
    - Maximal complexity set of intervals where wherever there was an endpoint, we will not cross in a set of ranges.

  - *gap* - set xlim to show the regions that are never expressed. could be regarded as introns, spliced out. (gaps of exons are introns)
- Elaborate the set of intervals by turning it into a GRanges object by specifying seqnames and range information.
- Metadata that should be specified - strand information, genome, seqlengths, seqinfo
- How to pick out transcription start sites - plot overlapping genes. Resize with argument 1 to get down to one base from start. Gives us the addresses of start sites
- Finding promoters - interval of three bases upstream of bases upstream is regarded as a promoter. Use *flank* operation with argument 3 - gives us the locations of the upstream promoters. Use start=FALSE to indicate flank at the end of the interval rather than start.

## Finding Overlaps

- Example: finding genes that are close to reported binding sites and add some annotation to those genes
- HepG2 + GM12878 - reported binding sites for 2 cell lines
- Want to find the genes that are nearest to them. Instead of seperately, create a consensus GRanges which includes only sites that are common to both GRanges
- Function: *findOverlaps* uses query and subject - for each range, see if it appears in another range and return pair. Returns object of class *hits*. Only want the ones where there is a hit - use *queryHits* function and subset based on queryHits
- Extract just region information using *granges* function
- Show extraction of genes in next video, and matching of the regions in ERBS dataset to genes

```
# load packages
library(GenomicFeatures)
library(GenomicRanges)
library(IRanges)
library(ERBS)

# load ESRRA ChIP data
data(HepG2)
data(GM12878)

# browseVignettes("GenomicRanges")

# find binding sites common to both HepG2 and GM12878
?findOverlaps
```

```
## Help on topic 'findOverlaps' was found in the following packages:
##
##   Package               Library
##   SummarizedExperiment
##                       /Library/Frameworks/R.framework/Versions/4.0/Resources/library
##   GenomicRanges         /Library/Frameworks/R.framework/Versions/4.0/Resources/library
##   GenomicAlignments     /Library/Frameworks/R.framework/Versions/4.0/Resources/library
##   IRanges               /Library/Frameworks/R.framework/Versions/4.0/Resources/library
##
##
## Using the first match ...
```

```
# for each row in query, return overlapping row in subject
res = findOverlaps(HepG2, GM12878)
class(res)
```

```
## [1] "SortedByQueryHits"
```

```
## attr(,"package")
## [1] "S4Vectors"
```

```
res
```

```
## Hits object with 75 hits and 0 metadata columns:
##        queryHits subjectHits
##        <integer>   <integer>
##    [1]         1          12
##    [2]         2          78
##    [3]         4         777
##    [4]         5           8
##    [5]         8          13
##    ...       ...         ...
##   [71]       285         621
##   [72]       287         174
##   [73]       291        1855
##   [74]       294         512
##   [75]       300         144
##   -------
##   queryLength: 303 / subjectLength: 1873
```

```
# ranges from the query for which we found a hit in the subject
index = queryHits(res)
erbs = HepG2[index,]
erbs
```

```
## GRanges object with 75 ranges and 7 metadata columns:
##        seqnames              ranges strand |      name     score       col
##           <Rle>           <IRanges>  <Rle> | <numeric> <integer> <logical>
##    [1]     chr2   20335378-20335787      * |        NA         0      <NA>
##    [2]    chr20       328285-329145      * |        NA         0      <NA>
##    [3]    chr19     1244419-1245304      * |        NA         0      <NA>
##    [4]    chr11   64071828-64073069      * |        NA         0      <NA>
##    [5]     chr2   16938364-16938840      * |        NA         0      <NA>
##    ...      ...                 ...    ... .       ...       ...       ...
##   [71]    chr12 118558730-118559158      * |        NA         0      <NA>
##   [72]     chr1   35331750-35332300      * |        NA         0      <NA>
##   [73]     chr1   26146200-26147004      * |        NA         0      <NA>
##   [74]     chr6   44224657-44225693      * |        NA         0      <NA>
##   [75]     chr1 110198573-110199126      * |        NA         0      <NA>
##        signalValue    pValue      qValue      peak
##          <numeric> <numeric>   <numeric> <integer>
##    [1]      58.251    75.899 6.14371e-72       195
##    [2]      10.808    69.685 5.02806e-66       321
##    [3]      12.427    43.855 1.35976e-40       604
##    [4]      10.850    40.977 7.33386e-38       492
##    [5]      12.783    38.004 5.36029e-35       255
##    ...         ...       ...         ...       ...
##   [71]       8.292    10.294 8.59089e-09       195
##   [72]      10.458    10.233 9.81822e-09       341
##   [73]       5.742    10.176 1.10429e-08       337
##   [74]       3.525    10.102 1.29621e-08       838
##   [75]       7.929    10.047 1.44208e-08       197
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

```
# extract only the ranges
granges(erbs)
```

```
## GRanges object with 75 ranges and 0 metadata columns:
##        seqnames              ranges strand
##           <Rle>           <IRanges>  <Rle>
##    [1]     chr2   20335378-20335787      *
##    [2]    chr20       328285-329145      *
##    [3]    chr19     1244419-1245304      *
##    [4]    chr11   64071828-64073069      *
##    [5]     chr2   16938364-16938840      *
##    ...      ...                 ...    ...
##   [71]    chr12 118558730-118559158      *
##   [72]     chr1   35331750-35332300      *
##   [73]     chr1   26146200-26147004      *
##   [74]     chr6   44224657-44225693      *
##   [75]     chr1 110198573-110199126      *
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

```
erbs
```

```
## GRanges object with 75 ranges and 7 metadata columns:
##        seqnames              ranges strand |      name     score       col
##           <Rle>           <IRanges>  <Rle> | <numeric> <integer> <logical>
##    [1]     chr2   20335378-20335787      * |        NA         0      <NA>
##    [2]    chr20       328285-329145      * |        NA         0      <NA>
##    [3]    chr19     1244419-1245304      * |        NA         0      <NA>
##    [4]    chr11   64071828-64073069      * |        NA         0      <NA>
##    [5]     chr2   16938364-16938840      * |        NA         0      <NA>
##    ...      ...                 ...    ... .       ...       ...       ...
##   [71]    chr12 118558730-118559158      * |        NA         0      <NA>
##   [72]     chr1   35331750-35332300      * |        NA         0      <NA>
##   [73]     chr1   26146200-26147004      * |        NA         0      <NA>
##   [74]     chr6   44224657-44225693      * |        NA         0      <NA>
##   [75]     chr1 110198573-110199126      * |        NA         0      <NA>
##        signalValue    pValue      qValue      peak
##          <numeric> <numeric>   <numeric> <integer>
##    [1]      58.251    75.899 6.14371e-72       195
##    [2]      10.808    69.685 5.02806e-66       321
##    [3]      12.427    43.855 1.35976e-40       604
##    [4]      10.850    40.977 7.33386e-38       492
##    [5]      12.783    38.004 5.36029e-35       255
##    ...         ...       ...         ...       ...
##   [71]       8.292    10.294 8.59089e-09       195
##   [72]      10.458    10.233 9.81822e-09       341
##   [73]       5.742    10.176 1.10429e-08       337
##   [74]       3.525    10.102 1.29621e-08       838
##   [75]       7.929    10.047 1.44208e-08       197
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

### Assessment: Finding Overlaps

```
library(ERBS)
data(HepG2)
data(GM12878)
paste("17th region of HepG2 starts at:", start(granges(HepG2[17])))
```

```
## [1] "17th region of HepG2 starts at: 46528596"
```

```
dtn <- distanceToNearest(HepG2[17], GM12878)
gm_idx <- subjectHits(dtn)
start_site <- start(GM12878[gm_idx])
distance_to_closest = mcols(dtn)$distance
paste("Start site of closest region to 17th region of HepG2: ", start_site)
```

```
## [1] "Start site of closest region to 17th region of HepG2:  46524762"
```

```
paste("Distance between closest region to 17th region of HepG2: ", distance_to_closest)
```

```
## [1] "Distance between closest region to 17th region of HepG2:  2284"
```

```
X <- vector(mode="integer", length=length(HepG2))
for(i in seq_along(HepG2)) {
     closest_region = distanceToNearest(HepG2[i], GM12878)
     distance = mcols(closest_region)$distance
     X[i] = distance
}
proportion_lt_2k_bp <- length(X[X < 2000]) / length(X)
paste("proportion of distances < 2000 bp: ", proportion_lt_2k_bp)
```

```
## [1] "proportion of distances < 2000 bp:  0.267326732673267"
```