Section 1: What we measure and why

Mammaprint Gene Signature

- Exploring genes used in the Mammaprint gene signature - assess risk of breast cancer
- Diagnostic signature using gene expression levels of 70 genes
- Information about the 70 gene signature used in the Mammaprint algorithm

```
library(genefu)
```

```
## Loading required package: survcomp
## Loading required package: survival
## Loading required package: prodlim
## Loading required package: mclust
## Package 'mclust' version 5.4.7
## Type 'citation("mclust")' for citing this R package in publications.
## Loading required package: limma
## Loading required package: biomaRt
## Loading required package: iC10
## Loading required package: pamr
## Loading required package: cluster
## Loading required package: impute
## Loading required package: iC10TrainingData
## Loading required package: AIMS
## Loading required package: e1071
## Loading required package: Biobase
## 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 object is masked from 'package:limma':
##
##     plotMA
## 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
```

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

```
data(sig.gene70)
dim(sig.gene70)
```

```
## [1] 70  9
```

```
head(sig.gene70)[,1:6]
```

```
##                         probe correlation average.good.prognosis.profile
## NM_003748           NM_003748   -0.420671                      0.12350000
## NM_003862           NM_003862   -0.410964                      0.05159091
## Contig32125_RC Contig32125_RC   -0.409054                      0.05409091
## U82987                 U82987   -0.407002                      0.06150000
## AB037863            AB037863    -0.402335                      0.06334091
## NM_020974           NM_020974   -0.399987                     -0.06231818
##                EntrezGene.ID NCBI.gene.symbol HUGO.gene.symbol
## NM_003748               8659          ALDH4A1          ALDH4A1
## NM_003862               8817            FGF18            FGF18
## Contig32125_RC            NA             <NA>             <NA>
## U82987                 27113             BBC3             BBC3
## AB037863                  NA             <NA>             <NA>
## NM_020974              57758           SCUBE2           SCUBE2
```

```
count_nan_gene_symbol <- sum(is.na(sig.gene70$NCBI.gene.symbol))
paste("Count of NaN NCBI gene symbols: ", count_nan_gene_symbol)
```

```
## [1] "Count of NaN NCBI gene symbols:  14"
```

```
subset_matching_desc <- sig.gene70[which(sig.gene70$Description == "cyclin E2"), ]
paste("NCBI gene matching the description cyclin E2: ", subset_matching_desc$NCBI.gene.symbol)
```

```
## [1] "NCBI gene matching the description cyclin E2:  CCNE2"
```

```
number_kinase_coding_genes <- length(grep("kinase", sig.gene70$Description))
paste("Number of kinase coding genes responsible for cell to cell communication: ", number_kinase_coding
```

```
## [1] "Number of kinase coding genes responsible for cell to cell communication:  4"
```

Assessment: Phenotypes

- COPDSexualDimorphism.data package - phenotypes (cols) individuals (rows)
- Data to assess incidence of COPD and emphysema by gender and smoking status
- The pkyrs variable in the expr.meta data.frame represents pack years smoked. Other variables include gender and diagmaj (disease status). These variables correspond to phenotypes.

```
library(COPDSexualDimorphism.data)
data(lgrc.expr.meta)
head(expr.meta)
```

```
##     tissueid     sample_name  newid   GENDER age       cigever pkyrs
## 1 LT001098RU LT001098RU_COPD 161745 2-Female  46 2-Ever (>100)    35
## 2 LT001796RU LT001796RU_CTRL 212671   1-Male  48 2-Ever (>100)    19
## 3 LT005419RU LT005419RU_COPD 291396   1-Male  70 2-Ever (>100)    43
## 4 LT007392RU LT007392RU_COPD 169067   1-Male  46 2-Ever (>100)    45
## 5 LT009615LU LT009615LU_CTRL  49801 2-Female  49 2-Ever (>100)    45
## 6 LT010491LL LT010491LL_COPD 180409   1-Male  78 2-Ever (>100)    51
##            diagmaj   gender
## 1 2-COPD/Emphysema 2-Female
## 2        3-Control   1-Male
## 3 2-COPD/Emphysema   1-Male
## 4 2-COPD/Emphysema   1-Male
## 5        3-Control 2-Female
## 6 2-COPD/Emphysema   1-Male
```

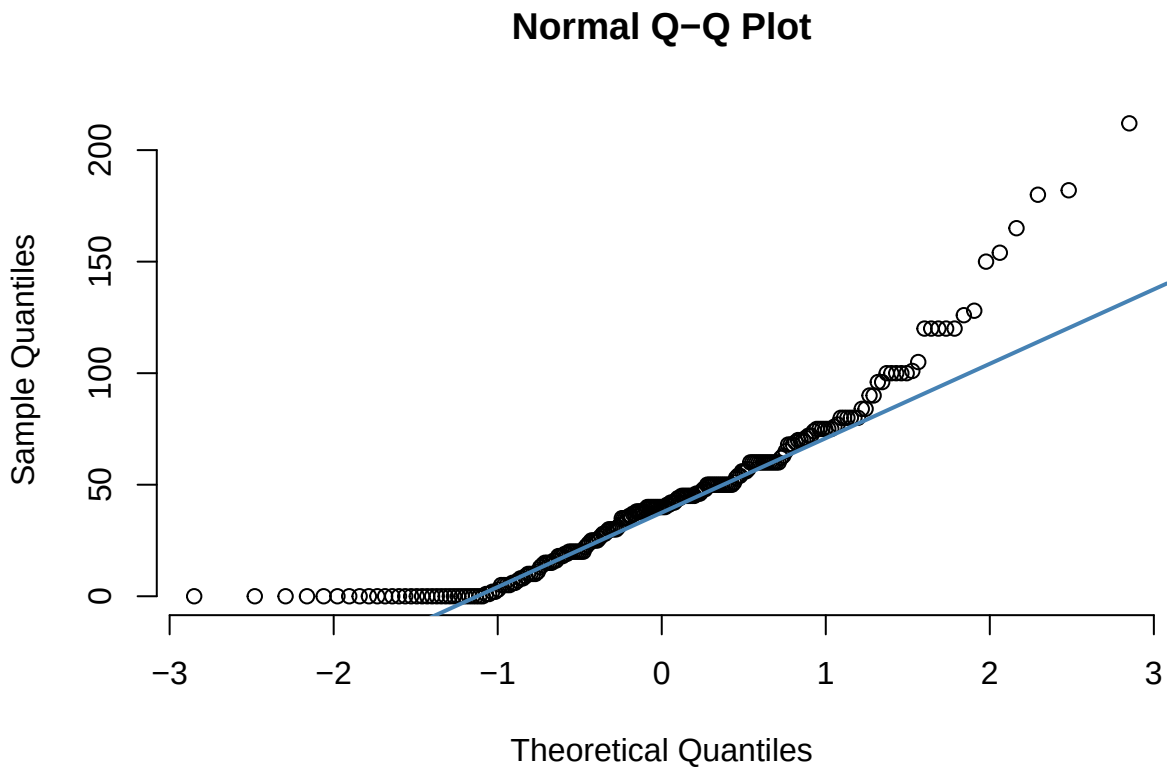
```
table(expr.meta$GENDER)
```

```
##
##   1-Male 2-Female
##      119      110
```

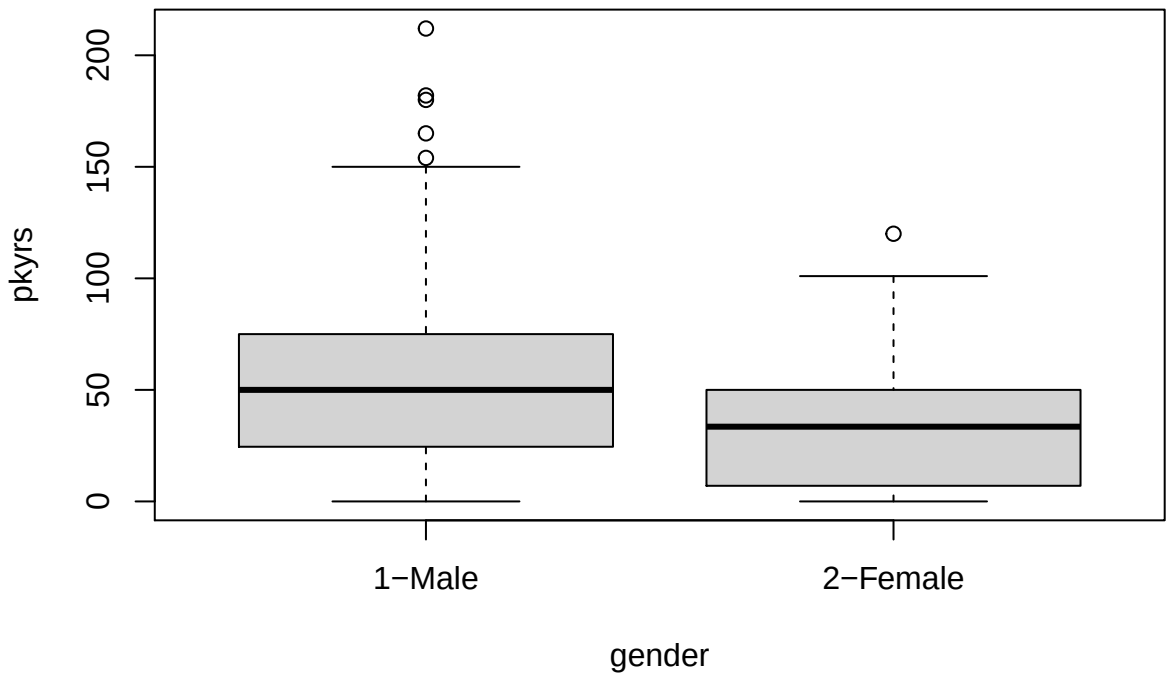
```
summary(expr.meta$pkyrs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   15.00   40.00   44.17   60.00  212.00
```

```
qqnorm(expr.meta$pkyrs, pch=1, frame=FALSE)
qqline(expr.meta$pkyrs, col = "steelblue", lwd = 2)
```




```
boxplot(pkyrs~gender, data=expr.meta)
```



Assessment: Chromosomes and SNPs

- GWAS (Genome-wide association studies)
- Comparing individuals with disease vs. controls using SNP chips or DNA sequencing.
- SNPs with association are investigated for disruption of gene regulation or function

- Bioconductor *gwascat* package

```
library(gwascat)
```

```
## gwascat loaded.  Use makeCurrentGwascat() to extract current image.

##  from EBI.  The data folder of this package has some legacy extracts.
```

```
data(ebicat_2020_04_30)
ebicat_2020_04_30
```

```
## gwasloc instance with 50000 records and 38 attributes per record.
## Extracted:  2020-04-30 23:24:51
## metadata()$badpos includes records for which no unique locus was given.
## Genome:  GRCh38
## Excerpt:
## GRanges object with 5 ranges and 3 metadata columns:
##       seqnames    ranges strand |   DISEASE/TRAIT        SNPS   P-VALUE
##          <Rle> <IRanges>  <Rle> |     <character> <character> <numeric>
##   [1]       10  58153390      * | Crohn's disease   rs1819658     9e-17
##   [2]        1 206766559      * | Crohn's disease   rs3024505     2e-14
##   [3]       13  42478744      * | Crohn's disease   rs2062305     5e-10
##   [4]       19   1124836      * | Crohn's disease    rs740495     8e-12
##   [5]       12  40398498      * | Crohn's disease  rs11564258     6e-21
##   -------
##   seqinfo: 24 sequences from GRCh38 genome
```

```
sort(table(ebicat_2020_04_30$CHR_ID), decreasing=TRUE)
```

```
##
##    1    2    6    3   11    5    4    7   12    8   17   10    9   16   19   15
## 4294 4290 4085 3202 2995 2908 2587 2530 2447 2307 2281 2138 2010 1972 1965 1746
##   14   20   18   13   22   21    X
## 1341 1270 1154 1090  790  401  197
```

Microarray Technology 1: How Hybridization Works

- Two technologies: microarray and NGS
- Both counting DNA or RNA molecules
- Both use a trick which allows us to take double-stranded DNA and convert to single-stranded
- Both require thousands - millions of molecules for us to be able to measure anything
- If a few cells only, they must be amplified

Microarray Technology

1. Denaturation (single-stranded)
2. Hybridization - when you have a single strand in solution and it finds complimentary DNA, it will hybridize to form 2 stranded DNA. This can be exploited to count molecules
3. Can create probes / troughs for different sequences. Put on location on piece of solid for the molecules we want to be able to count. Probes have compliments to the DNA that we want to count.

How microarray technology works

- Piece of solid where we put probes - 1x1 cm piece of silicone that gets divided into thousands to millions of cells (difference squares)

- squares correspond to probes which represent molecules we are trying to count
- 25bP long probes in example
- second step: label a sample with fluorescent tags and put on array. hope that right molecules hybridize to right probes

Two-color microarrays

- Hybridize two samples onto one array - two different labels that scanner can recognize
- Advantages: cost savings
- Sample 1: color 1, Sample 2: color 2. Let hybridize and get both hybridized to same probes, but scanner can distinguish two types of labels.
- Two numbers per probe – converted into RGB color combining red and green

Applications of microarray technology

- 3 different applications

1. **Measuring gene expression** - gene chip array.

- For every gene, we know the sequence and take 11 sequences for individual transcripts and hybridize.
- On this array, probes are towards 3' end of transcripts b/c RNA tends to degrade more on one side (5' end).
- 11 probes scattered around array to avoid confounding location with gene for each transcript.
- Label the RNA, put it on the array. Will see lots of hybridization if there are many copies of that transcript.
- High intensity = highly expressed gene. For each gene, select n probes and put them on the array and analyze the data.

2. **Genotyping SNP** - different alleles 2 of same or 1 of each.

- I.e., AA, AG, GG.
- If we want to know which of the three possibilities, we can do this for SNPs.
- Use probes to hybridize to piece of sequence which has A, G for example.
- Genotype millions of SNPs at a time. Arrays popular for GWAS studies to understand which alleles are associated with genes of interest.

3. **Detection of transcription factor binding sites** - genome is more than just sequence, measuring the chemical processes taking place around the genome, i.e. where specific protein is bound.

- Transcription factor = proteins that start gene expression. * Have DNA, want to know where specific protein is bound. Start by fragmenting DNA, some pieces have protein and others do not.
- Divide by presence of protein vs. not - hybridize the part with protein with tiling array and if lights up, the location is where the protein was bound.
- Intensities are not that reliable, must be controlled by hybridizing the total DNA for comparison.

Labeling

Need indirect ways to count molecules. Labeling adds a chemical to each molecule, use optical scanner to identify the different intensities based on # labels and quantify.

Design attribute of different technologies: synthetically sequenced, or cloned. Densities of probes put on the solid is also variable across different technologies. Also # samples on each array differs. Major manufacturers:

1. Affymetrix (high density, one color)
2. Agilent (circles on grid, one or two color)
3. Illumina (high density, one or two color)
    - Uses beads instead of in-situ sequencing

Brief introduction of NGS

- Early 21st century Human Genome sequenced
- Took DNA from several humans, pooled together and sequenced base x base the entire thing (1st generation)
- Back then, millions of clones 1k bP long. Different labs would sequence each clone, and then put together using computational methods. Cost billions of dollars
- NGS is high throughput - billions of bP in a week for thousands of dollars.
- Many copies of DNA to obtain measurement.

1. Fragment DNA using mol bio –> feed into NGS sequencer
2. Read sequences for all of the fragments to get the reads (bases)

Illumina flow cells - 8 lanes (1 per sample)

- For each lane, we get 160mill short reads (50-70bP long)
- Starts with DNA sample with fragments. Add adapters to each one to allow fragments to attach to pieces of solid. Once attached, amplify each one so that we have millions of copies (clusters) - adapter + fragment + copies.
- Add labeled nucleotides. Different from microArray - not having two molecules join.
- First base of the sequence attached to compliment, and starts forming double stranded. Once the first nucleotide attaches, take a picture and read intensity of labels. Keep doing this until we get through almost the entire fragment.
- Images from sequencing machine represent clusters - first base of sequence represented by cluster.
- Next step we get another image corresponding to bases on the second location and assemble the whole molecular sequence.

Applications of NGS

- Similarly to microArray technology, we can measure gene expression, genotype, location of transcription factor binding sites
- First application: sequencing the genome
- Resequencing: do not sequence the whole genome with a new subject of same species, only areas of interest. SNP discovery, genotyping, variant discovery and quantification
- Measuring methalation
- Used be 1000s genomes project, human epigenome project

Going from series of reads to measurements * First step in analyzing the NGS sequences: finding where the reads came from. All reads get mapped to the genome : matches to the reference genome * First application: Variant detection * Finding new SNPs. Take sample, sequence, align to genome, go to any specific location and ask if it is a SNP by analyzing whether there are G's and A's (heterozygous) * There are sequencing errors * Deletion: alleles are missing a base

- RNA seq - NGS to quantify gene expression
- We have RNA - two samples to see if they are different.
- RNA –> DNA, sequence DNA and map / align to reference genome
- Compare gene expression in the two samples
- ChipSeq - finding transcription factor binding sites
- DNA –> separate fragments bound to specific proteins and sequence those sections
- Location of genome bound to many reads –> means that it was bound to that protein
- Peak Detectors to identify locations of protein binding sites

Analyzing gene expression microarray dataset

```
library(tissuesGeneExpression)
data(tissuesGeneExpression)
paste("log scale intensities for microarray probes: ")
```

```
## [1] "log scale intensities for microarray probes: "
```

```
head(e[,1:5])
```

```
##           GSM11805.CEL.gz GSM11814.CEL.gz GSM11823.CEL.gz GSM11830.CEL.gz
## 1007_s_at       10.191267       10.509167       10.272027       10.252952
## 1053_at          6.040463        6.696075        6.144663        6.575153
## 117_at           7.447409        7.775354        7.696235        8.478135
## 121_at          12.025042       12.007817       11.633279       11.075286
## 1255_g_at        5.269269        5.180389        5.301714        5.372235
## 1294_at          8.535176        8.587241        8.277414        8.603650
##           GSM12067.CEL.gz
## 1007_s_at       10.157605
## 1053_at          6.606701
## 117_at           8.116336
## 121_at          10.832528
## 1255_g_at        5.334905
## 1294_at          8.303227
```

```
paste("tissue types of each sample: ")
```

```
## [1] "tissue types of each sample: "
```

```
table(tissue)
```

```
## tissue
##  cerebellum       colon endometrium hippocampus      kidney       liver
##          38          34          15          31          39          26
##    placenta
##           6
```

```
paste("overall mean expression of 209169_at", mean(e["209169_at",]))
```

```
## [1] "overall mean expression of 209169_at 7.26365039170786"
```

```
paste("mean expression by tissue: ")
```

```
## [1] "mean expression by tissue: "
```

```
sort(by(e["209169_at",], tissue, mean))
```

```
## tissue
##       colon    placenta       liver      kidney endometrium  cerebellum
##    5.076710    5.186711    5.249247    5.325370    5.585281   10.149866
## hippocampus
##   11.466372
```

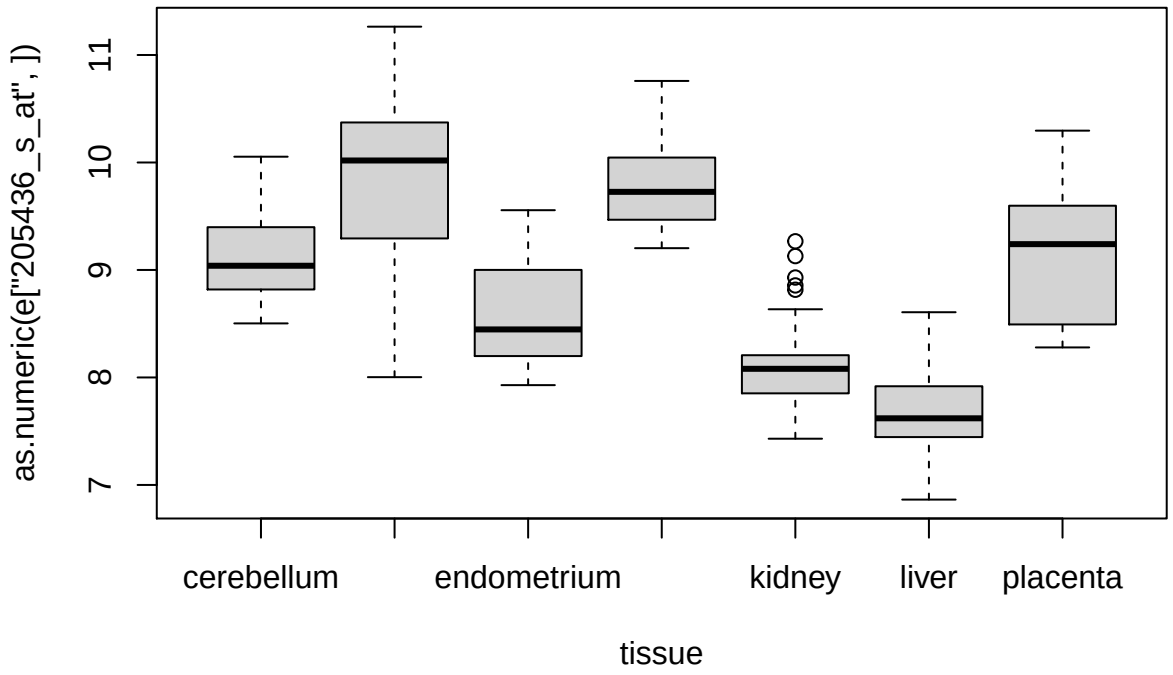
Gene Associated with probe ID

```
library(hgu133a.db)
```

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

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

```
## Loading required package: IRanges
## Loading required package: S4Vectors
##
## Attaching package: 'S4Vectors'
## The following object is masked from 'package:base':
##
##     expand.grid
## Loading required package: org.Hs.eg.db
##
##
symbol = mapIds(hgu133a.db, keys=rownames(e), column="SYMBOL", keytype="PROBEID")
## 'select()' returned 1:many mapping between keys and columns
paste("gene associated with probe ID 209169_at", symbol["209169_at"])
## [1] "gene associated with probe ID 209169_at GPM6B"
num_features <- sum(symbol == "H2AX", na.rm=TRUE)
paste("number of features measuring expression of H2AX: ", num_features)
## [1] "number of features measuring expression of H2AX:  4"
paste("associated probes: ")
## [1] "associated probes: "
symbol[grep("H2AX", symbol)]
## 205436_s_at 212524_x_at 212525_s_at 213344_s_at
##      "H2AX"      "H2AX"      "H2AX"      "H2AX"
paste("comparing distributions across tissues: ")
## [1] "comparing distributions across tissues: "
boxplot(as.numeric(e["205436_s_at",])~tissue)
```




```
paste("finding gene specific to placenta: ")
```

```
## [1] "finding gene specific to placenta: "
```

```
IDs = c("201884_at", "209169_at", "206269_at", "207437_at", "219832_s_at", "212827_at")
sort(rowMeans(e[IDs, which(tissue == "placenta")]))
```

```
##   207437_at   209169_at   212827_at   201884_at 219832_s_at   206269_at
##    4.410814    5.186711    6.481558    6.637451    7.319096   11.372918
```

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: 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: ")
```

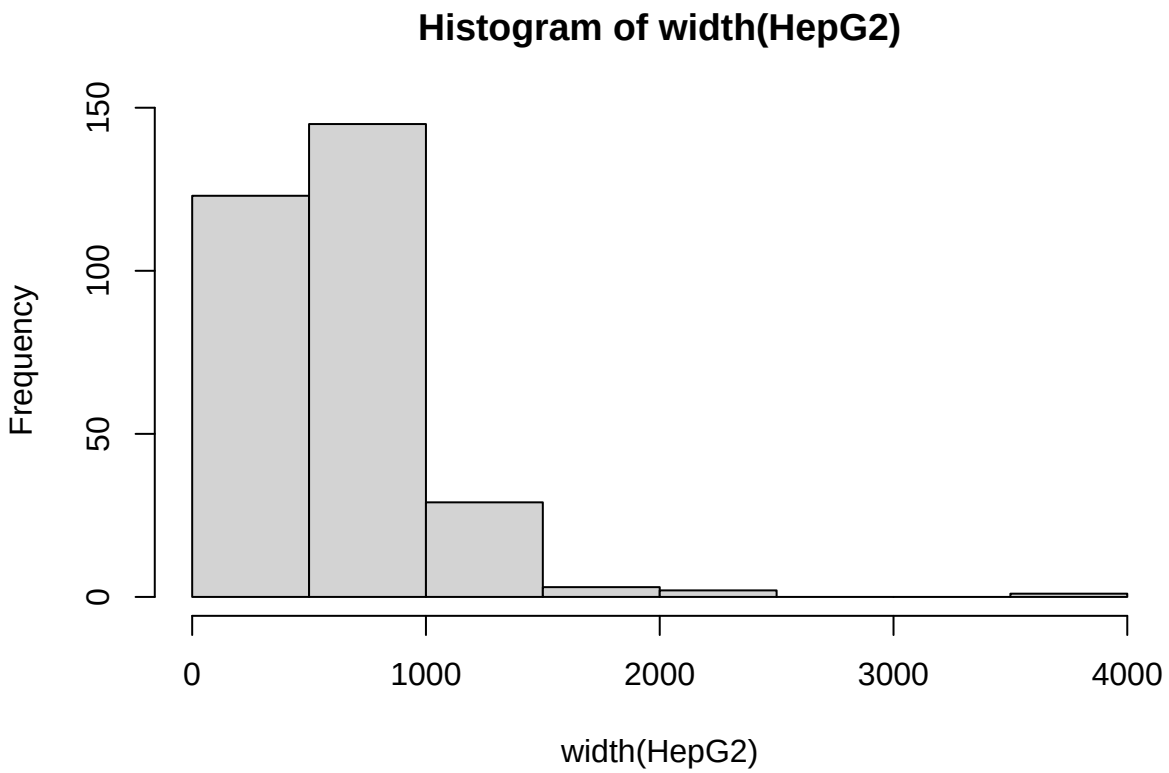
```
## [1] "Number of regions from chromosome 16: "
```

```
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"
```

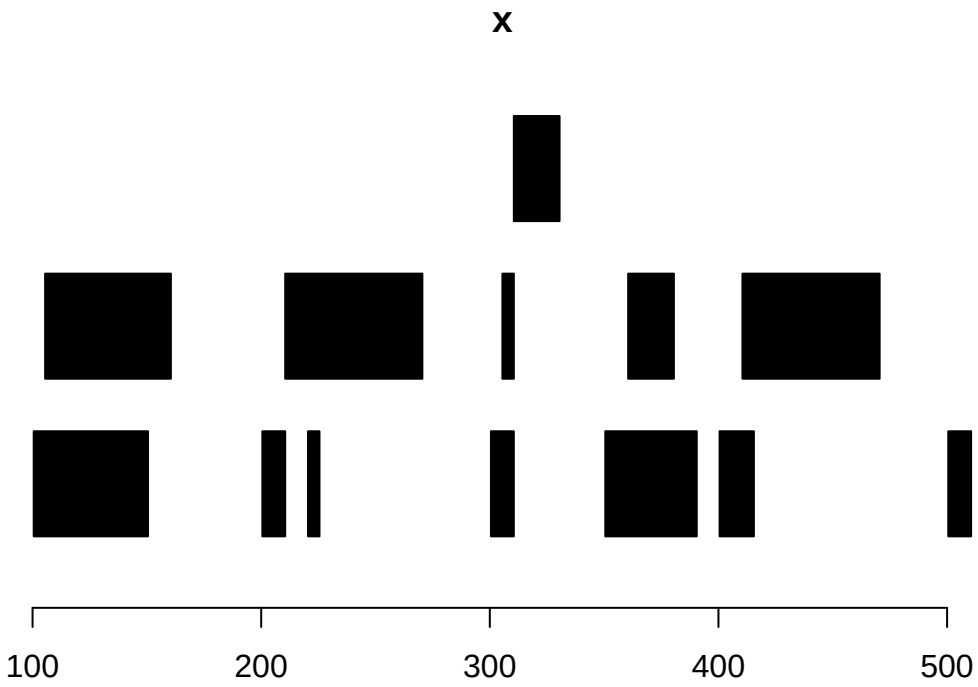
```
m_ir <- IRanges(start=c(1, 11, 21),end=c(3, 15, 27))
paste("sum of widths of multiple IRanges objects:", sum(width(m_ir)))
```

```
## [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: Homo.sapiens
## Loading required package: OrganismDbi
## Loading required package: GenomicFeatures
## Loading required package: GO.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))
```