Genome Annotation and Visualisation using R and Bioconductor

Mark Dunning

29/07/2015

Previously

- Introduced Bioconductor facilites for manipulating strings and ranges
- Executed workflow to find to identify genes and regions of interest in an RNA-seq experiment

Aims for this session

- Obtaining annotation information from different sources
  - Biomart
  - Pre-built Bioconductor packages
  - Browser tracks
- Visualise
  - Aligned sequencing reads
  - Coverage
  - Gene models

biomaRt

biomaRt

- A wealth of annotation resources are available online through the biomart (http://www.biomart.org) web software suite.
- One-off queries are possible. But are they reproducible? What if you need to do further analysis on the results in R?
- Results generated using Bioconductor can be easily annotated against the vast wealth of online data available in biomart
- User does not need to construct complex SQL queries

Connecting to biomaRt

```
library(biomaRt)
head(listMarts(), 5)
```

```
##         biomart                          version
## 1       ensembl      ENSEMBL GENES 80 (SANGER UK)
## 2           snp  ENSEMBL VARIATION 80 (SANGER UK)
## 3    regulation ENSEMBL REGULATION 80 (SANGER UK)
## 4          vega              VEGA 60  (SANGER UK)
## 5 fungi_mart_26         ENSEMBL FUNGI 26 (EBI UK)
```

```
ensembl <- useMart("ensembl")
```

Connecting to biomaRt

```
ensembl <- useMart("ensembl",
                   dataset = "hsapiens_gene_ensembl")
head(listDatasets(ensembl),10)
```

```
##                          dataset
## 1          oanatinus_gene_ensembl
## 2         cporcellus_gene_ensembl
## 3         gaculeatus_gene_ensembl
## 4          lafricana_gene_ensembl
## 5  itridecemlineatus_gene_ensembl
## 6         choffmanni_gene_ensembl
## 7          csavignyi_gene_ensembl
## 8             fcatus_gene_ensembl
## 9        rnorvegicus_gene_ensembl
## 10         psinensis_gene_ensembl
##                                 description         version
## 1      Ornithorhynchus anatinus genes (OANA5)           OANA5
## 2             Cavia porcellus genes (cavPor3)         cavPor3
## 3      Gasterosteus aculeatus genes (BROADS1)         BROADS1
## 4          Loxodonta africana genes (loxAfr3)         loxAfr3
## 5  Ictidomys tridecemlineatus genes (spetri2)         spetri2
## 6         Choloepus hoffmanni genes (choHof1)         choHof1
## 7              Ciona savignyi genes (CSAV2.0)         CSAV2.0
## 8         Felis catus genes (Felis_catus_6.2) Felis_catus_6.2
## 9          Rattus norvegicus genes (Rnor_6.0)        Rnor_6.0
## 10     Pelodiscus sinensis genes (PelSin_1.0)      PelSin_1.0
```

An example query

Say we want to find out more information about a given ***Entrez*** gene(s).

- Essentially we want to subset the database according to a particular *filter*.
- Available filters can be listed.

```
head(listFilters(ensembl), 5)
```

```
##              name     description
## 1 chromosome_name Chromosome name
## 2           start Gene Start (bp)
## 3             end   Gene End (bp)
## 4      band_start      Band Start
## 5        band_end        Band End
```

```
flt <- listFilters(ensembl)
flt[grep("entrez", flt[,1]),]
```

```
##                              name
## 28                 with_entrezgene
## 29 with_entrezgene_transcript_name
## 87                      entrezgene
## 88      entrezgene_transcript_name
##                                                    description
## 28                                     with EntrezGene ID(s)
## 29                        with EntrezGene Transcript Name(s)
## 87                            EntrezGene ID(s) [e.g. 115286]
## 88 EntrezGene transcript name ID(s) [e.g. CTD-2350J17.1-002]
```

Attributes

- *Attributes* are the information that can be retreived

```
head(listAttributes(ensembl), 25)
```

```
##                                name                     description
## 1                   ensembl_gene_id                Ensembl Gene ID
## 2             ensembl_transcript_id          Ensembl Transcript ID
## 3                ensembl_peptide_id             Ensembl Protein ID
## 4                   ensembl_exon_id                Ensembl Exon ID
## 5                       description                    Description
## 6                  chromosome_name                Chromosome Name
## 7                    start_position                Gene Start (bp)
## 8                      end_position                  Gene End (bp)
## 9                            strand                         Strand
## 10                            band                           Band
## 11                transcript_start          Transcript Start (bp)
## 12                  transcript_end            Transcript End (bp)
## 13        transcription_start_site Transcription Start Site (TSS)
## 14               transcript_length              Transcript length
## 15                  transcript_tsl Transcript Support Level (TSL)
## 16        transcript_gencode_basic       GENCODE basic annotation
## 17               transcript_appris              APPRIS annotation
## 18              external_gene_name           Associated Gene Name
## 19            external_gene_source         Associated Gene Source
## 20        external_transcript_name     Associated Transcript Name
## 21 external_transcript_source_name   Associated Transcript Source
## 22                transcript_count               Transcript count
## 23           percentage_gc_content                   % GC content
## 24                     gene_biotype                      Gene type
## 25              transcript_biotype                Transcript type
```

Forming the query

- We are going to use `entrezgene`
- First specify the filter type, and values
  - these must be valid identifiers for the filter type
  - in our case, valid Entrez IDs

```
entrez <- c("673", "837")
myfilter <- "entrezgene"
```

- Specify the attributes you want to retrieve
  - this must be in the first column of the output of `listAttributes`

```
attr = c("entrezgene", "hgnc_symbol", "ensembl_gene_id","description")
allAttr <- listAttributes(ensembl)
attr %in% allAttr[,1]
```

```
## [1] TRUE TRUE TRUE TRUE
```

- Plug all the values into the `getBM` function

```
myInfo <- getBM(filters="entrezgene",
    values=entrez,
    attributes=attr,
    mart=ensembl)
```

View the results

```
myInfo
```

```
##   entrezgene hgnc_symbol ensembl_gene_id
## 1        673        BRAF ENSG00000157764
## 2        673        BRAF         LRG_299
## 3        837       CASP4 ENSG00000196954
##                                                                    des
cription
## 1   B-Raf proto-oncogene, serine/threonine kinase [Source:HGNC Symbol;Acc:HG
NC:1097]
## 2   B-Raf proto-oncogene, serine/threonine kinase [Source:HGNC Symbol;Acc:HG
NC:1097]
## 3 caspase 4, apoptosis-related cysteine peptidase [Source:HGNC Symbol;Acc:HG
NC:1505]
```

- Note that we don't necesarily get a data frame with one row per ID we specified
  - in this case, one gene had more than one ensembl ID
  - technically, we would say the mapping is *one-to-many*

Using multiple filters

- A common query is to list genes within a certain genomic interval
  - e.g. regions of interest from a CHiP-seq analysis
- This time, our filters would be chromosome name, start and end
  - we can specify these in a vector
  - check the correct names by looking at the output of `listFilters`

```
myfilters <- c("chromosome_name", "start", "end")
```

- The values need to be specified in a list

```
myvalues <- list(16, 1100000, 1250000)
```

- Define attributes as before
  - be careful that `start` and `end` are not valid *attribute* names

```
head(allAttr[grep("start", allAttr[,1]),])
```

```
##                         name                    description
## 7             start_position                Gene Start (bp)
## 11          transcript_start          Transcript Start (bp)
## 13  transcription_start_site Transcription Start Site (TSS)
## 135              pirsf_start                    PIRSF start
## 138        superfamily_start              SUPERFAMILY start
## 141              smart_start                    SMART start
```

```
attr <- c("ensembl_gene_id", "hgnc_symbol","entrezgene","chromosome_name", "sta
rt_position", "end_position")
```

Make the query

```
myInfo <- getBM(attributes = attr,
  filters = myfilters,
  values=myvalues,mart=ensembl)
myInfo
```

```
##    ensembl_gene_id hgnc_symbol entrezgene chromosome_name start_position
## 1  ENSG00000260702                     NA              16        1103280
## 2  ENSG00000260532                     NA              16        1111627
## 3  ENSG00000273551                     NA              16        1148224
## 4  ENSG00000172236      TPSAB1       7177              16        1240696
## 5  ENSG00000197253       TPSB2      64499              16        1227272
## 6  ENSG00000261294                     NA              16        1206560
## 7  ENSG00000259910                     NA              16        1159548
## 8  ENSG00000116176       TPSG1      25823              16        1221651
## 9  ENSG00000260403                     NA              16        1156976
## 10 ENSG00000277010                     NA              16        1223639
## 11 ENSG00000196557     CACNA1H       8912              16        1153241
##    end_position
## 1       1105461
## 2       1113399
## 3       1148754
## 4       1242554
## 5       1230184
## 6       1207124
## 7       1160176
## 8       1225257
## 9       1157974
## 10      1224143
## 11      1221771
```

Reversing the query

- i.e supply gene names and get their positions

```
myfilters <- "ensembl_gene_id"
values = c("ENSG00000261713","ENSG00000261720","ENSG00000181791")
attr <- c("ensembl_gene_id","chromosome_name","start_position", "end_position",
"entrezgene")
getBM(attributes = attr, filters = myfilters, values = values,
ensembl
)
```

```
##   ensembl_gene_id chromosome_name start_position end_position entrezgene
## 1 ENSG00000261713              16        1064093      1078731     146336
## 2 ENSG00000261720              16        1065240      1066502         NA
```

Bioconductor Annotation Resources

Organism-level Packages

- Bioconductor maintain a number of organism-level packages which are re-built every 6 months. A central identifier (Entrez gene id) is used.
- These are listed on the annotation section of Bioconductor
  - here (http://bioconductor.org/packages/release/BiocViews.html#___AnnotationData)
  - named *org.X.ID.db*
  - where X is a two-letter organism acronym; i.e. Hs for human)
  - ID represents which identifier scheme is used i.e. eg for Entrez
- Installed in the same way as regular Bioconductor packages
  - `source("http://www.bioconductor.org/biocLite.R")`
  - `biocLite(.....)`

```
library(org.Hs.eg.db)
```

- Larger download size, but you only need to download once per-Bioconductor release
- Enable *offline* queries

Filtering an organism package

- `keytypes` are the names of the filters we can use

```
keytypes(org.Hs.eg.db)
```

```
##  [1] "ENTREZID"     "PFAM"         "IPI"          "PROSITE"
##  [5] "ACCNUM"       "ALIAS"        "ENZYME"       "MAP"
##  [9] "PATH"         "PMID"         "REFSEQ"       "SYMBOL"
## [13] "UNIGENE"      "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
## [17] "GENENAME"     "UNIPROT"      "GO"           "EVIDENCE"
## [21] "ONTOLOGY"     "GOALL"        "EVIDENCEALL"  "ONTOLOGYALL"
## [25] "OMIM"         "UCSCKG"
```

- We can see the names of valid keys

```
length(keys(org.Hs.eg.db,keytype="ENTREZID"))
```

```
## [1] 56340
```

```
head(keys(org.Hs.eg.db,keytype="ENTREZID"))
```

```
## [1] "1"  "2"  "3"  "9"  "10" "11"
```

Selecting attributes

- the attributes are `columns`
  - think the columns of a table that we want to look up

```
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"
```

Example query

```
entrez <- c("673", "837")
select(org.Hs.eg.db, keys=entrez,
  keytype="ENTREZID",
  columns=c("SYMBOL","CHRLOC","CHRLOCEND"))
```

```
##   ENTREZID SYMBOL     CHRLOC CHRLOCCHR  CHRLOCEND
## 1      673   BRAF -140433813         7 -140624564
## 2      837  CASP4 -104813594        11 -104827422
## 3      837  CASP4 -104813594        11 -104839325
```

Another query

> Give me the *Symbols* of every gene with *GO* ontology *GO:0003674 (GO:0003674)*

```
head(select(org.Hs.eg.db, keys = "GO:0003674",
keytype = "GO", columns = "SYMBOL"))
```

```
##           GO EVIDENCE ONTOLOGY  SYMBOL
## 1 GO:0003674       ND       MF    A1BG
## 2 GO:0003674       ND       MF   AP2A2
## 3 GO:0003674       ND       MF    AIF1
## 4 GO:0003674       ND       MF    AIM1
## 5 GO:0003674       ND       MF   BCL7A
## 6 GO:0003674       ND       MF CEACAM1
```

Managing gene models: GenomicFeatures

- The GenomicFeatures package retrieves and manages transcript-related features from the UCSC Genome site and BioMart data resources
- Transcript metadata is stored in an *TranscriptDb* object
- The object maps 5 and 3 UTRS, protein coding sequences (CDS) and exons for a set of mRNA transcripts to their associated genome
- *SQLite* database used to manage relationships between transcripts, exons, CDS and gene identifiers
- Again, *offline* queries can be made

Pre-built packages

- Again a full list of packages is available on the BioC website
  - here (http://bioconductor.org/packages/release/BiocViews.html#___AnnotationData)
- For humans, latest version is `TxDb.Hsapiens.UCSC.hg19.knownGene`
  - a convention is to assign the object to a shorter name to save some typing

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
```

```
## Loading required package: GenomicFeatures
## Loading required package: GenomicRanges
```

```
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
```

The transcriptDB object

```
txdb
```

```
## TxDb object:
## # Db type: TxDb
## # Supporting package: GenomicFeatures
## # Data source: UCSC
## # Genome: hg19
## # Organism: Homo sapiens
## # UCSC Table: knownGene
## # Resource URL: http://genome.ucsc.edu/
## # Type of Gene ID: Entrez Gene ID
## # Full dataset: yes
## # miRBase build ID: GRCh37
## # transcript_nrow: 82960
## # exon_nrow: 289969
## # cds_nrow: 237533
## # Db created by: GenomicFeatures package from Bioconductor
## # Creation time: 2015-03-19 13:55:51 -0700 (Thu, 19 Mar 2015)
## # GenomicFeatures version at creation time: 1.19.32
## # RSQLite version at creation time: 1.0.0
## # DBSCHEMAVERSION: 1.1
```

keys for the object

- As for the organism packages, we can see what keys are available

```
keytypes(txdb)
```

```
## [1] "GENEID"   "TXID"     "TXNAME"   "EXONID"   "EXONNAME" "CDSID"
## [7] "CDSNAME"
```

```
columns(txdb)
```

```
##  [1] "CDSID"      "CDSNAME"    "CDSCHROM"   "CDSSTRAND"  "CDSSTART"
##  [6] "CDSEND"     "EXONID"     "EXONNAME"   "EXONCHROM"  "EXONSTRAND"
## [11] "EXONSTART"  "EXONEND"    "GENEID"     "TXID"       "EXONRANK"
## [16] "TXNAME"     "TXTYPE"     "TXCHROM"    "TXSTRAND"   "TXSTART"
## [21] "TXEND"
```

Making a query

```
select(txdb, keys=entrez,
keytype="GENEID",
columns=c("TXID",
"TXCHROM", "TXSTART",
"TXEND"))
```

```
##   GENEID  TXID TXCHROM   TXSTART     TXEND
## 1    673 31502    chr7 140433813 140624564
## 2    837 44976   chr11 104813594 104827422
## 3    837 44977   chr11 104813594 104839325
## 4    837 44978   chr11 104815475 104839325
## 5    837 44979   chr11 104819547 104839325
## 6    837 44980   chr11 104822124 104839325
```

Querying the exons

```
mygene <- select(txdb, keys = "673", keytype = "GENEID",
columns = c("EXONID", "EXONCHROM", "EXONSTART","EXONEND","EXONSTRAND"))
mygene
```

```
##    GENEID EXONID EXONCHROM EXONSTRAND EXONSTART   EXONEND
## 1     673 112179      chr7          - 140624366 140624564
## 2     673 112178      chr7          - 140549911 140550012
## 3     673 112177      chr7          - 140534409 140534672
## 4     673 112176      chr7          - 140508692 140508795
## 5     673 112175      chr7          - 140507760 140507862
## 6     673 112174      chr7          - 140501212 140501360
## 7     673 112173      chr7          - 140500162 140500281
## 8     673 112172      chr7          - 140494108 140494267
## 9     673 112171      chr7          - 140487348 140487384
## 10    673 112170      chr7          - 140482821 140482957
## 11    673 112169      chr7          - 140481376 140481493
## 12    673 112168      chr7          - 140477791 140477875
## 13    673 112167      chr7          - 140476712 140476888
## 14    673 112166      chr7          - 140453987 140454033
## 15    673 112165      chr7          - 140453075 140453193
## 16    673 112164      chr7          - 140449087 140449218
## 17    673 112163      chr7          - 140439612 140439746
## 18    673 112162      chr7          - 140433813 140434570
```

Exon Strucutre

- We could of course create a `GRanges` object from this

```
GRanges(mygene$EXONCHROM, IRanges(mygene$EXONSTART,
mygene$EXONEND),strand=mygene$EXONSTRAND,exon_id=mygene$EXONID)
```

```
## GRanges object with 18 ranges and 1 metadata column:
##        seqnames                 ranges strand   |   exon_id
##           <Rle>              <IRanges>  <Rle>   | <integer>
##    [1]     chr7 [140624366, 140624564]      -   |    112179
##    [2]     chr7 [140549911, 140550012]      -   |    112178
##    [3]     chr7 [140534409, 140534672]      -   |    112177
##    [4]     chr7 [140508692, 140508795]      -   |    112176
##    [5]     chr7 [140507760, 140507862]      -   |    112175
##    ...      ...                    ...    ... ...       ...
##   [14]     chr7 [140453987, 140454033]      -   |    112166
##   [15]     chr7 [140453075, 140453193]      -   |    112165
##   [16]     chr7 [140449087, 140449218]      -   |    112164
##   [17]     chr7 [140439612, 140439746]      -   |    112163
##   [18]     chr7 [140433813, 140434570]      -   |    112162
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

Convenience Functions

```
trs <- transcripts(txdb)
trs
```

```
## GRanges object with 82960 ranges and 2 metadata columns:
##           seqnames               ranges strand   |     tx_id     tx_name
##              <Rle>            <IRanges>  <Rle>   | <integer> <character>
##       [1]     chr1     [ 11874,  14409]      +   |         1  uc001aaa.3
##       [2]     chr1     [ 11874,  14409]      +   |         2  uc010nxq.1
##       [3]     chr1     [ 11874,  14409]      +   |         3  uc010nxr.1
##       [4]     chr1     [ 69091,  70008]      +   |         4  uc001aal.1
##       [5]     chr1     [321084, 321115]      +   |         5  uc001aaq.2
##       ...      ...                  ...    ... ...       ...         ...
##   [82956]     chrY [27605645, 27605678]      -   |     78803  uc004fwx.1
##   [82957]     chrY [27606394, 27606421]      -   |     78804  uc022cpc.1
##   [82958]     chrY [27607404, 27607432]      -   |     78805  uc004fwz.3
##   [82959]     chrY [27635919, 27635954]      -   |     78806  uc022cpd.1
##   [82960]     chrY [59358329, 59360854]      -   |     78807  uc011ncc.1
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

Retrieve all exons at once

```
exs <- exons(txdb)
exs
```

```
## GRanges object with 289969 ranges and 1 metadata column:
##            seqnames               ranges strand   |   exon_id
##               <Rle>            <IRanges>  <Rle>   | <integer>
##        [1]     chr1       [11874, 12227]      +   |         1
##        [2]     chr1       [12595, 12721]      +   |         2
##        [3]     chr1       [12613, 12721]      +   |         3
##        [4]     chr1       [12646, 12697]      +   |         4
##        [5]     chr1       [13221, 14409]      +   |         5
##        ...      ...                   ...    ... ...       ...
##   [289965]     chrY [27607404, 27607432]      -   |    277746
##   [289966]     chrY [27635919, 27635954]      -   |    277747
##   [289967]     chrY [59358329, 59359508]      -   |    277748
##   [289968]     chrY [59360007, 59360115]      -   |    277749
##   [289969]     chrY [59360501, 59360854]      -   |    277750
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

Group by genes

```
exons <- exonsBy(txdb, "gene")
is(exons)
```

```
## [1] "GRangesList"                       "CompressedList"
## [3] "GenomicRangesList"                 "GenomicRangesORGRangesList"
## [5] "List"                              "GenomicRangesORGenomicRangesList"
## [7] "Vector"                            "Annotated"
```

```
length(exons)
```

```
## [1] 23459
```

see also `transcriptsBy` , `intronsByTranscript` , `fiveUTRsByTranscript` , `threeUTRsByTranscript`

Subset this object

```
exons[["673"]]
```

```
## GRanges object with 18 ranges and 2 metadata columns:
##        seqnames                 ranges strand   |   exon_id   exon_name
##           <Rle>              <IRanges>  <Rle>   | <integer> <character>
##    [1]     chr7 [140433813, 140434570]      -   |    112162        <NA>
##    [2]     chr7 [140439612, 140439746]      -   |    112163        <NA>
##    [3]     chr7 [140449087, 140449218]      -   |    112164        <NA>
##    [4]     chr7 [140453075, 140453193]      -   |    112165        <NA>
##    [5]     chr7 [140453987, 140454033]      -   |    112166        <NA>
##    ...      ...                    ...    ... ...       ...         ...
##   [14]     chr7 [140507760, 140507862]      -   |    112175        <NA>
##   [15]     chr7 [140508692, 140508795]      -   |    112176        <NA>
##   [16]     chr7 [140534409, 140534672]      -   |    112177        <NA>
##   [17]     chr7 [140549911, 140550012]      -   |    112178        <NA>
##   [18]     chr7 [140624366, 140624564]      -   |    112179        <NA>
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

Implications

- We now have a way of retrieving transcript and exon locations as `GRanges` .
- Any function that uses a `GRanges` object can easily interact with gene locations
    - Reading subset of a bam file
    - Counting overlaps
    - Retrieving genome sequence

Examples

Retreive the subset of reads that overlap a particular gene.

- First, return the positional information about the gene as a `GRanges` object

```
gr <- exons[["49"]]
```

- Then, pass the `GRanges` object into the `readGAlignments` function
    - here, the `system.time` function is used to report how long the function takes

```
library(GenomicAlignments)
```

```
## Loading required package: Biostrings
## Loading required package: XVector
## Loading required package: Rsamtools
```

```
system.time(bam.sub <- readGAlignments(file = mybam,
    use.names = TRUE, param = ScanBamParam(which = gr)))
```

```
##    user  system elapsed
##   0.102   0.000   0.151
```

Examine the output

```
bam.sub
```

```
## GAlignments object with 1917 alignments and 0 metadata columns:
##                      seqnames strand       cigar    qwidth     start
##                         <Rle>  <Rle> <character> <integer> <integer>
##    SRR076681.239386       22      -       1S67M        68  51176595
##    SRR078452.251117       22      -         68M        68  51176597
##    SRR076696.585674       22      -         68M        68  51176597
##    SRR078501.824091       22      +         68M        68  51176605
##    SRR078568.818440       22      +         68M        68  51176606
##                 ...      ...    ...         ...       ...       ...
##     SRR076132.39409       22      -         68M        68  51183674
##    SRR076898.252854       22      -         68M        68  51183679
##    SRR076176.943759       22      -         68M        68  51183687
##     SRR076340.66381       22      -         68M        68  51183699
##   SRR076936.1030386       22      -         68M        68  51183724
##                           end     width     njunc
##                      <integer> <integer> <integer>
##    SRR076681.239386  51176661        67         0
##    SRR078452.251117  51176664        68         0
##    SRR076696.585674  51176664        68         0
##    SRR078501.824091  51176672        68         0
##    SRR078568.818440  51176673        68         0
##                 ...       ...       ...       ...
##     SRR076132.39409  51183741        68         0
##    SRR076898.252854  51183746        68         0
##    SRR076176.943759  51183754        68         0
##     SRR076340.66381  51183766        68         0
##   SRR076936.1030386  51183791        68         0
##   -------
##   seqinfo: 86 sequences from an unspecified genome
```

Retrieving gene sequences

```
library(BSgenome.Hsapiens.UCSC.hg19)
hg19 <- BSgenome.Hsapiens.UCSC.hg19
```

```
system.time(seqs <- getSeq(hg19, exons[["49"]]))
```

```
##    user  system elapsed
##   0.230   0.008   0.256
```

```
seqs
```

```
##   A DNAStringSet instance of length 6
##     width seq
## [1]    89 AGTGCCAGGAGTATGGTTGAGATGCTACCAA...CCGTGGTTGCTAAAGATAACGCCACGTGTGA
## [2]   204 TGGCCCCTGTGGGTTACGGTTCAGGCAAAAC...TCACTGCTGCTCACTGCTTCGTCGGCAAAAA
## [3]   284 TAATGTGCATGACTGGAGACTGGTTTTCGGA...GTGGCCGGCTGGGGATATATAGAAGAGAAAG
## [4]   666 TAATGTGCATGACTGGAGACTGGTTTTCGGA...TGTGGCCGTATGACAGTGCCTTCCACTCTCT
## [5]   146 CCCCCAGGCCATCATCTATACTGATGGAGGC...GTATCCTGTAGGCAAGATCGACACCTGCCAG
## [6]   647 GGAGACAGCGGCGGGCCTCTCATGTGCAAAG...ATAAATAAATAAACATATATATATAGATATA
```

```
width(exons[["49"]])
```

```
## [1]  89 204 284 666 146 647
```

Alternative counting

```
bam <- readGAlignments(file = mybam)
countOverlaps(gr, bam)
```

```
## [1]  37  46 175 182 212 297
```

Other sources of annotation

- The `rtracklayer` package allows a number of standard genome *tracks* to be imported
  - *bed*
  - *gff*
  - *wig*
- The result is a `GRanges` object - of course!

```
library(rtracklayer)
download.file("http://www.nimblegen.com/downloads/annotation/ez_exome_v3/SeqCap
EZ_Exome_v3.0_Design_Annotation_files.zip",destfile="Nimblgen-regions.zip")
unzip("Nimblgen-regions.zip")
nimb <- import("SeqCap_EZ_Exome_v3_primary.bed")
nimb
```

```
## UCSC track 'target_region'
## UCSCData object with 242232 ranges and 1 metadata column:
##            seqnames               ranges strand   |
##               <Rle>            <IRanges>  <Rle>   |
##        [1]     chr1       [14426, 14627]      *   |
##        [2]     chr1       [14638, 14883]      *   |
##        [3]     chr1       [14903, 15103]      *   |
##        [4]     chr1       [15670, 15990]      *   |
##        [5]     chr1       [16590, 17074]      *   |
##        ...      ...                  ...    ... ...
##   [242228]     chrY [59355662, 59356146]      *   |
##   [242229]     chrY [59356745, 59357067]      *   |
##   [242230]     chrY [59357675, 59357797]      *   |
##   [242231]     chrY [59357856, 59358098]      *   |
##   [242232]     chrY [59358152, 59358273]      *   |
##                                                                  name
##                                                            <character>
##        [1] gn|RP11-34P13.2;ens|ENSG00000227232;vega|OTTHUMG00000000958
##        [2] gn|RP11-34P13.2;ens|ENSG00000227232;vega|OTTHUMG00000000958
##        [3] gn|RP11-34P13.2;ens|ENSG00000227232;vega|OTTHUMG00000000958
##        [4] gn|RP11-34P13.2;ens|ENSG00000227232;vega|OTTHUMG00000000958
##        [5] gn|RP11-34P13.2;ens|ENSG00000227232;vega|OTTHUMG00000000958
##        ...                                                          ...
##   [242228]       gn|WASH6P;ens|ENSG00000182484;vega|OTTHUMG00000022677
##   [242229]       gn|WASH6P;ens|ENSG00000182484;vega|OTTHUMG00000022677
##   [242230]       gn|WASH6P;ens|ENSG00000182484;vega|OTTHUMG00000022677
##   [242231]       gn|WASH6P;ens|ENSG00000182484;vega|OTTHUMG00000022677
##   [242232]       gn|WASH6P;ens|ENSG00000182484;vega|OTTHUMG00000022677
##   -------
##   seqinfo: 24 sequences from an unspecified genome; no seqlengths
```

Practical time

Exploring RNA-seq results

- Using biomaRt
- organism packages
- transcript databases

Visualisation

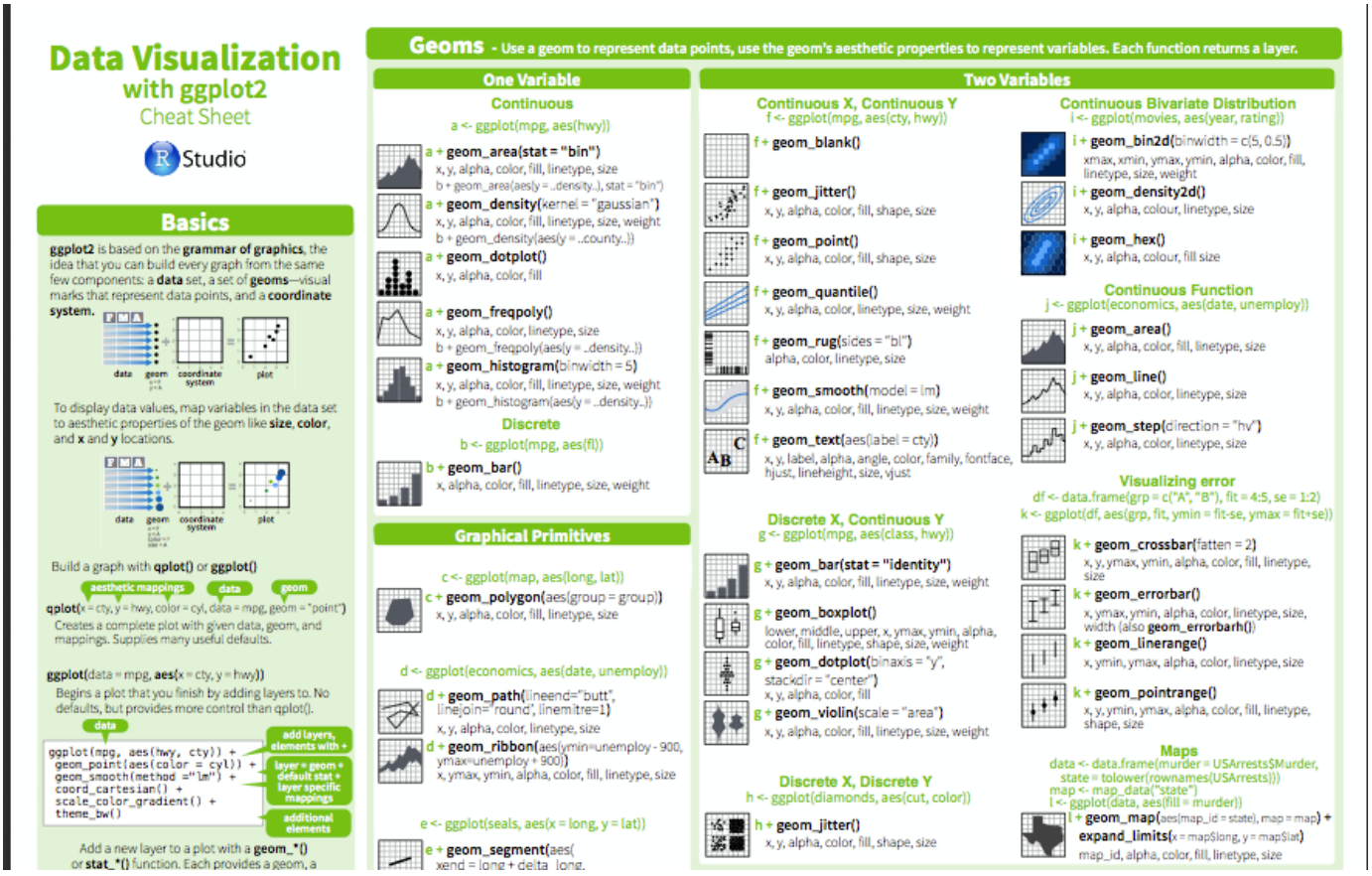
More-advanced graphics in R

- Base graphics in R use a canvas model
  - series of instructions that sequentially fill the plotting canvas
- ggplot2 employs a `grammar of graphics approach`
- The components are
  - a datset
  - geometric object that is visual representation of the data
    - e.g. points, lines, etc

- mapping of variables to visual properties of plot
  - ***aesthetics***
- (statistical summarisation rule)
- (coordinate system)
- (facet specification)

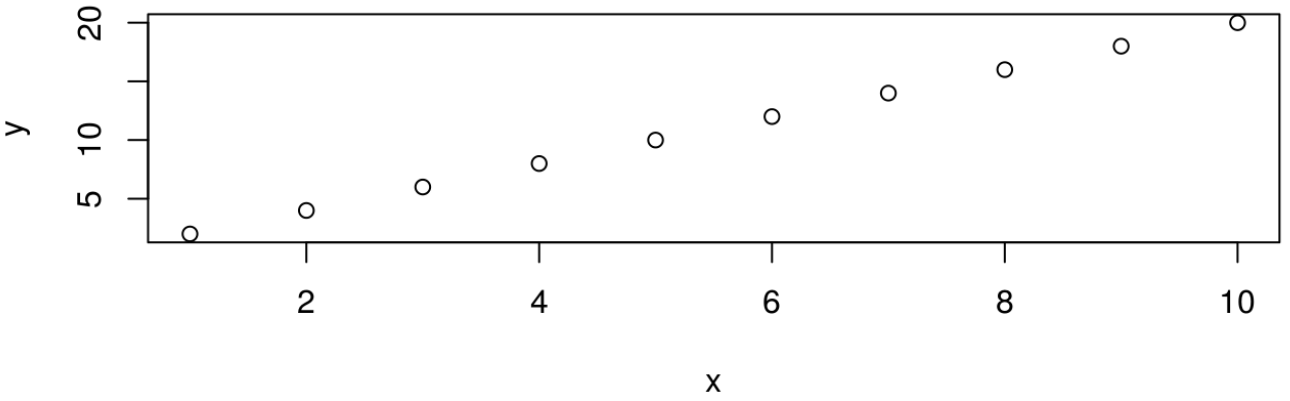
ggplot2 overview

ggplot2 cheat-sheet (https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf)

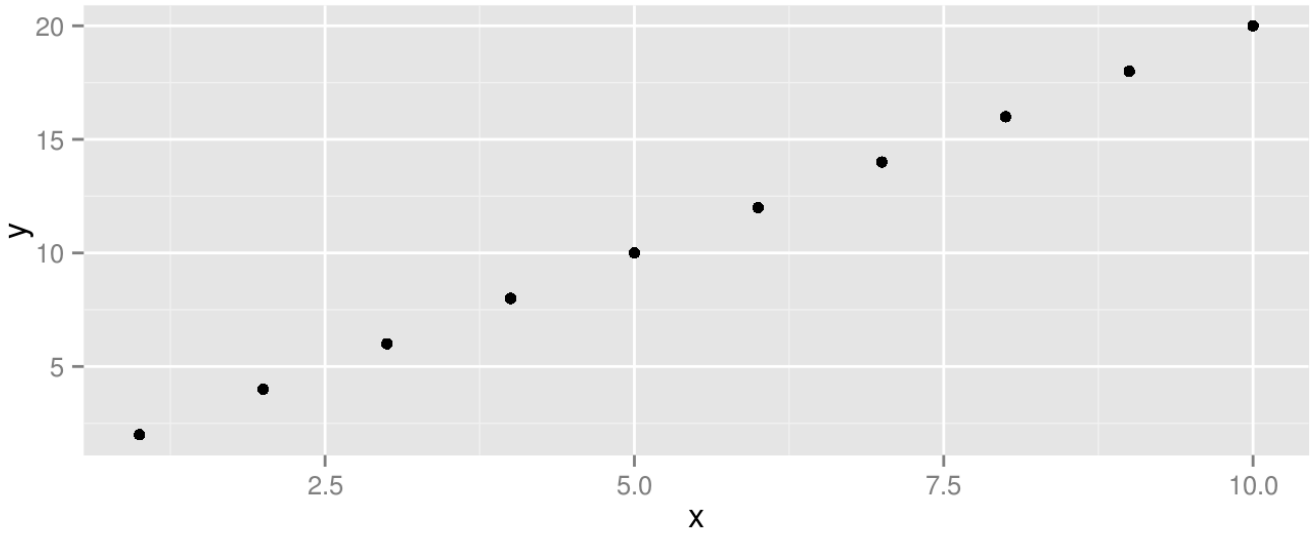


Plot Comparison

```
x <- 1:10
y <- 2*x
plot(x,y)
```




```
library(ggplot2)
df <-data.frame(x,y)
ggplot(df, aes(x=x,y=y)) + geom_point()
```



Plot construction

- ggplot2 needs data as a data frame
- It needs to be `long` format

```
library(reshape2)
df <- data.frame(A = rnorm(5,3), B=rnorm(5,1))
df[1:3,]
```

```
##          A         B
## 1 3.387681 1.4769919
## 2 4.207090 0.7612296
## 3 2.268161 2.1824987
```

```
df2 <- melt(df)
```

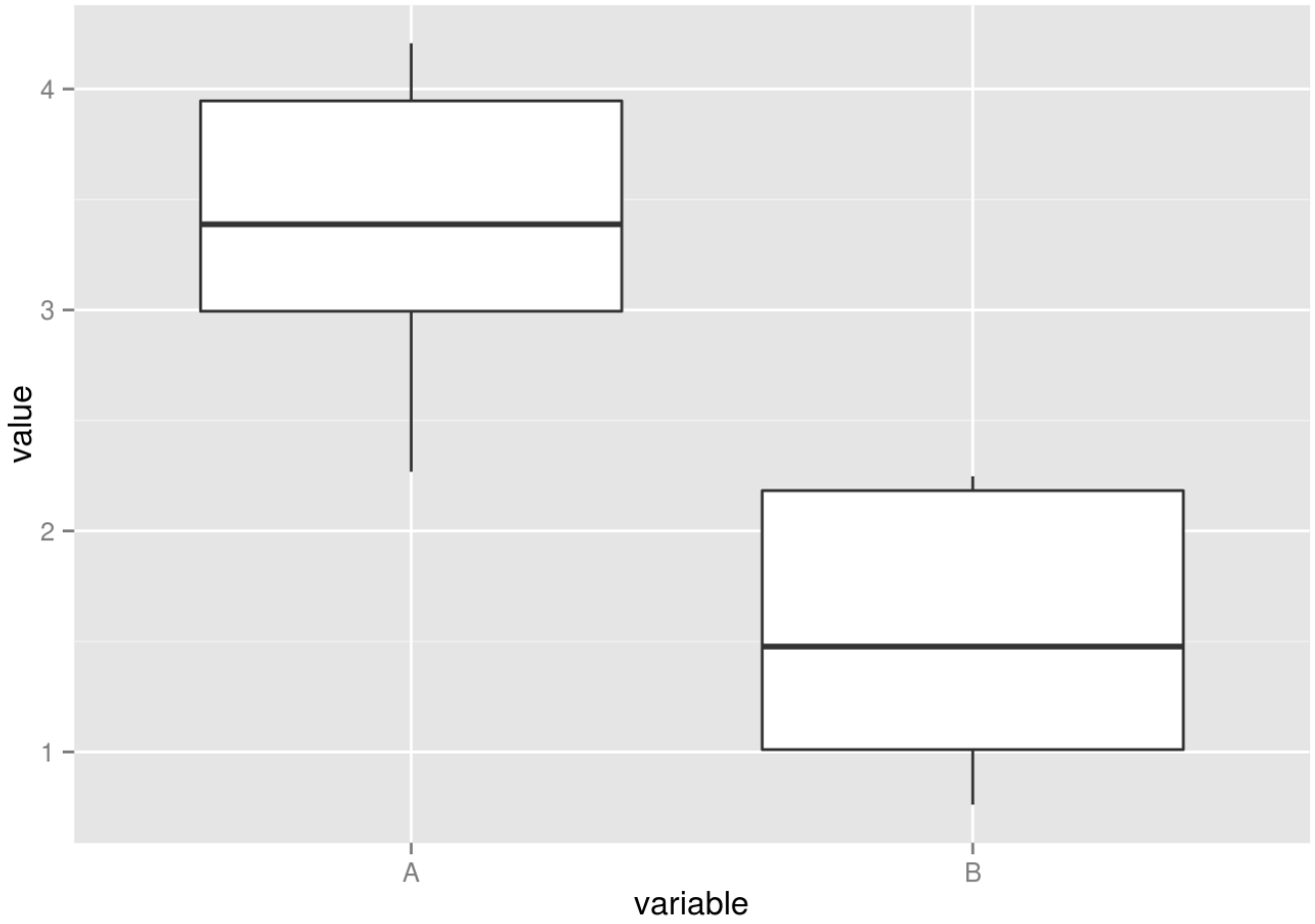
```
## No id variables; using all as measure variables
```

```
df2
```

```
##    variable     value
## 1         A 3.3876809
## 2         A 4.2070899
## 3         A 2.2681607
## 4         A 3.9463350
## 5         A 2.9943387
## 6         B 1.4769919
## 7         B 0.7612296
## 8         B 2.1824987
## 9         B 1.0106740
## 10        B 2.2474967
```

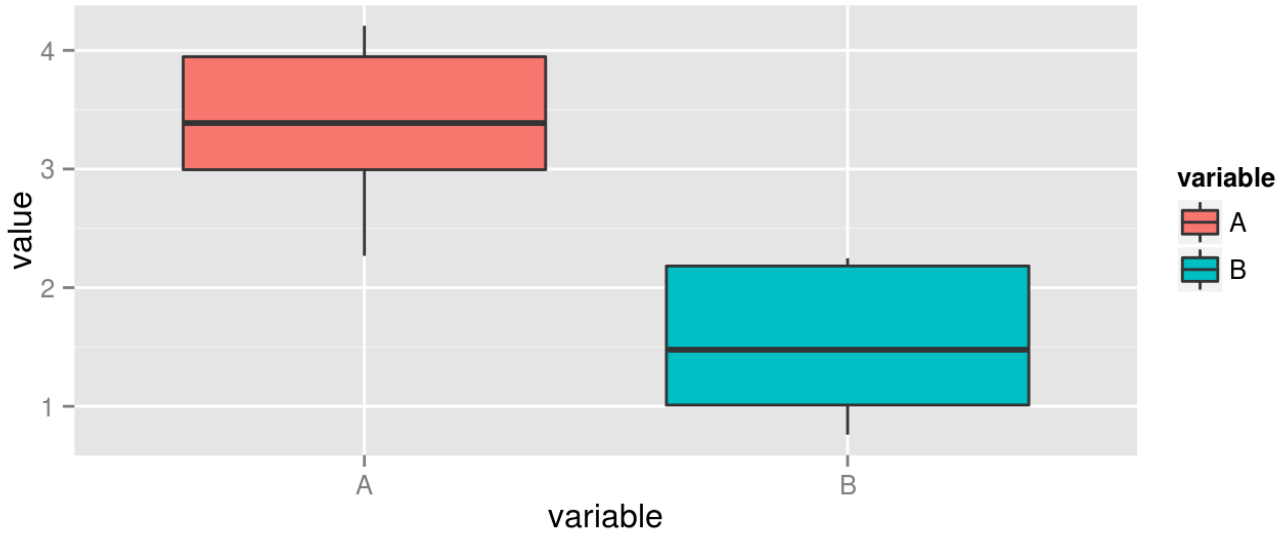
Plot construction

```
ggplot(df2, aes(x = variable,y=value)) + geom_boxplot()
```



Plot construction

```
ggplot(df2, aes(x = variable,y=value,fill=variable)) + geom_boxplot()
```



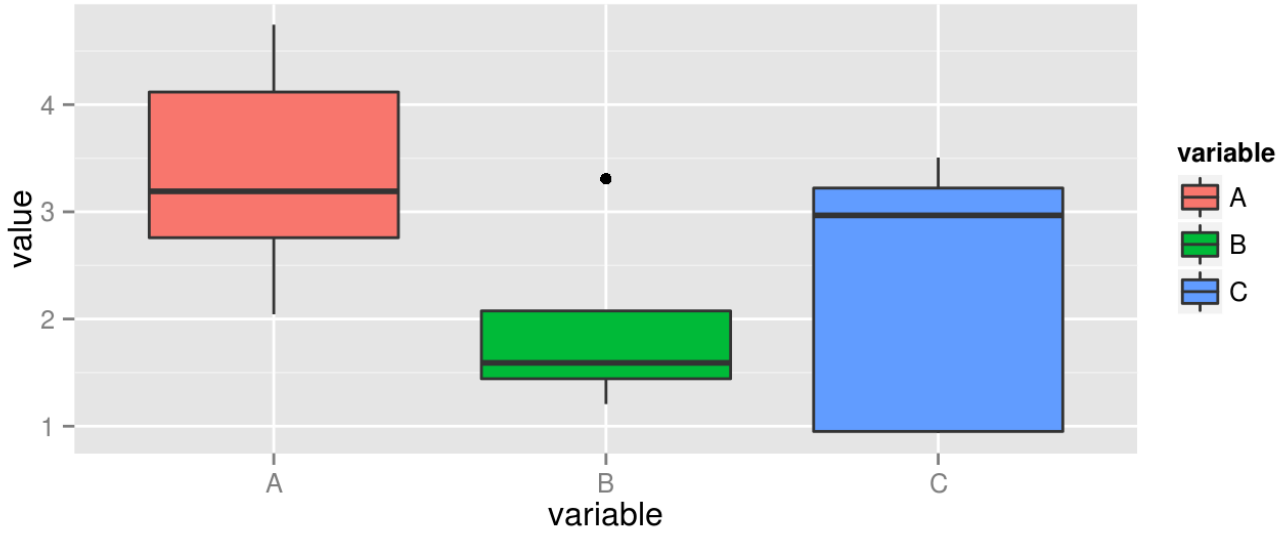
Updating a plot

- ggplot2 will easily re-drawn a plot as new variables are added
  - a real advantage!

```
df <- data.frame(A = rnorm(5,3), B=rnorm(5,1),C=rnorm(5,2))
df2 <- melt(df)
```

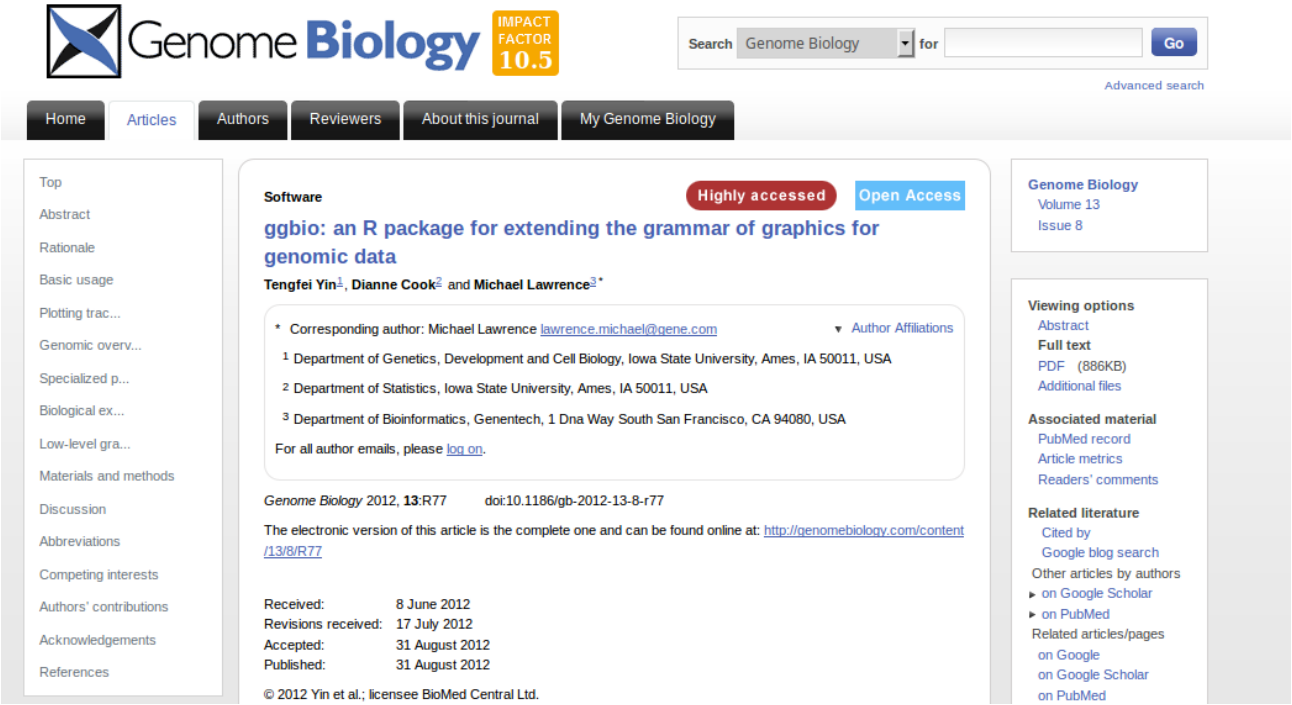
```
## No id variables; using all as measure variables
```

```
ggplot(df2, aes(x = variable,y=value,fill=variable)) + geom_boxplot()
```



Introducing ggbio

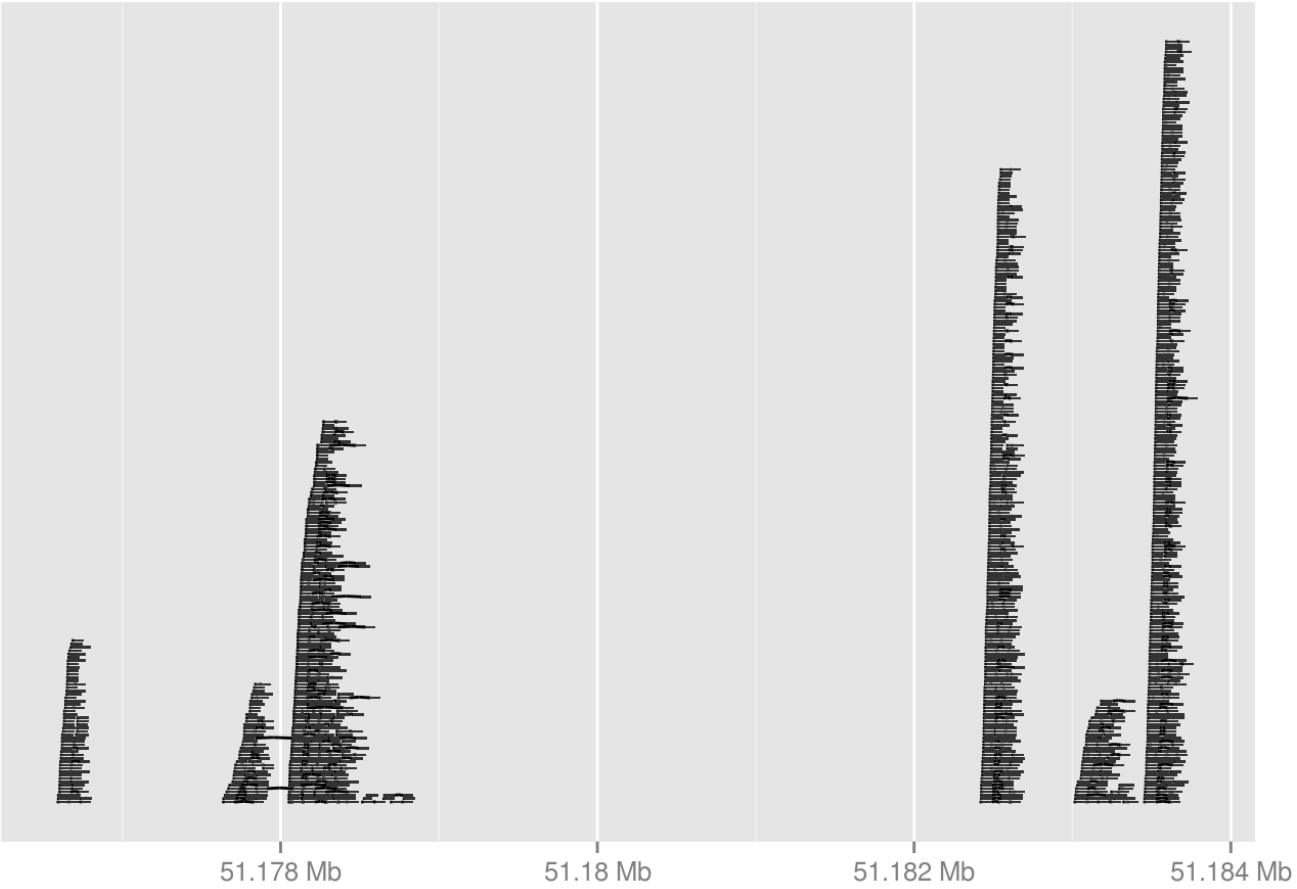
- A consistent representation of ranges and genomic data helps with visualisation
- The `ggbio` package is a toolkit for producing publication-quality images from genomic data
- It extends the Grammar of Graphics approach taken by `ggplot2`
- It knows about the standard Bioconductor classes we have already introduced
- Published in Genome Biology (http://www.genomebiology.com/2012/13/8/R77)



The autoplot function

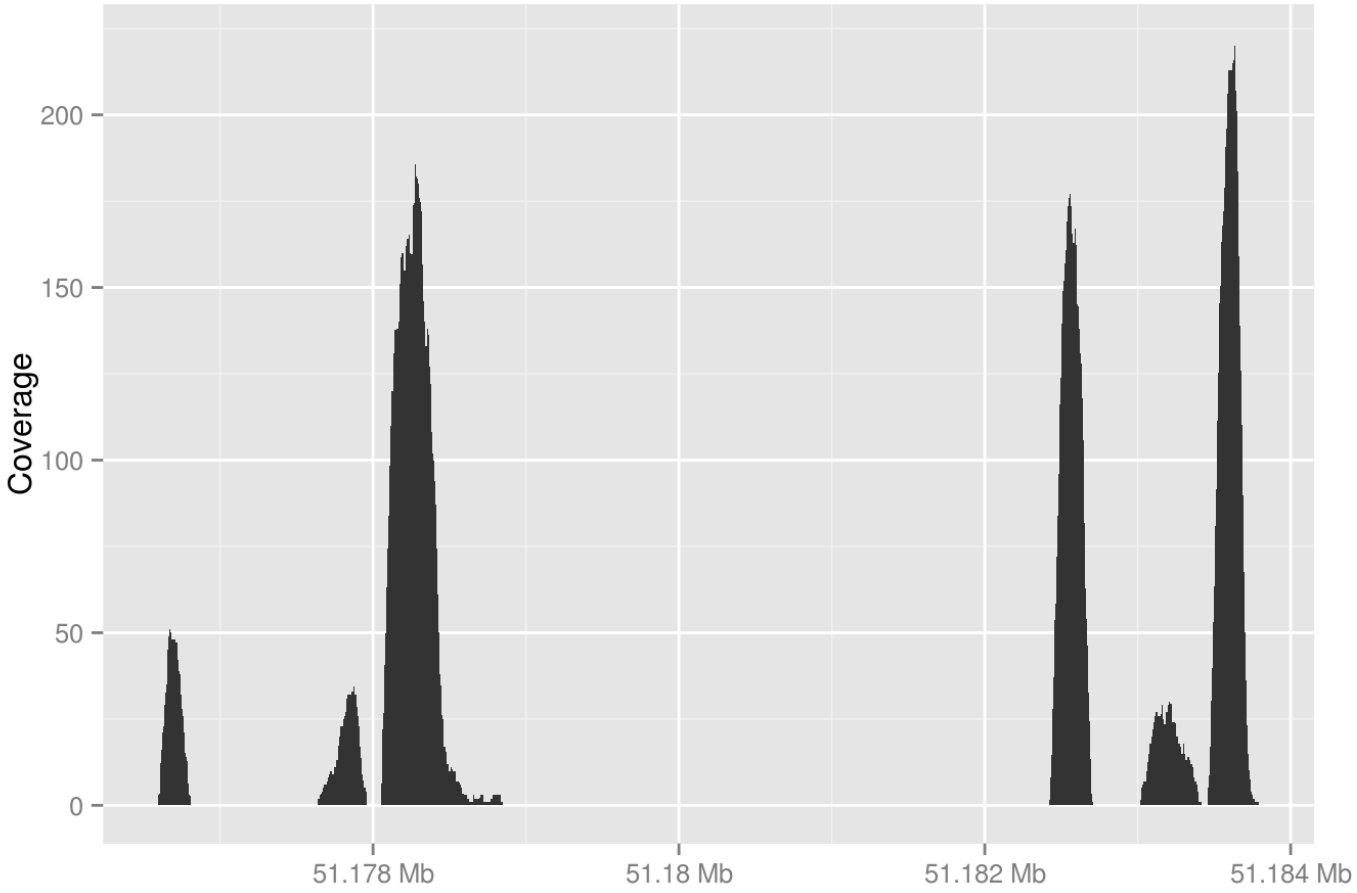
- Guesses what type of plot you want from the data
- Figures out the x and y coordinates

```
library(ggbio)
autoplot(bam.sub)
```



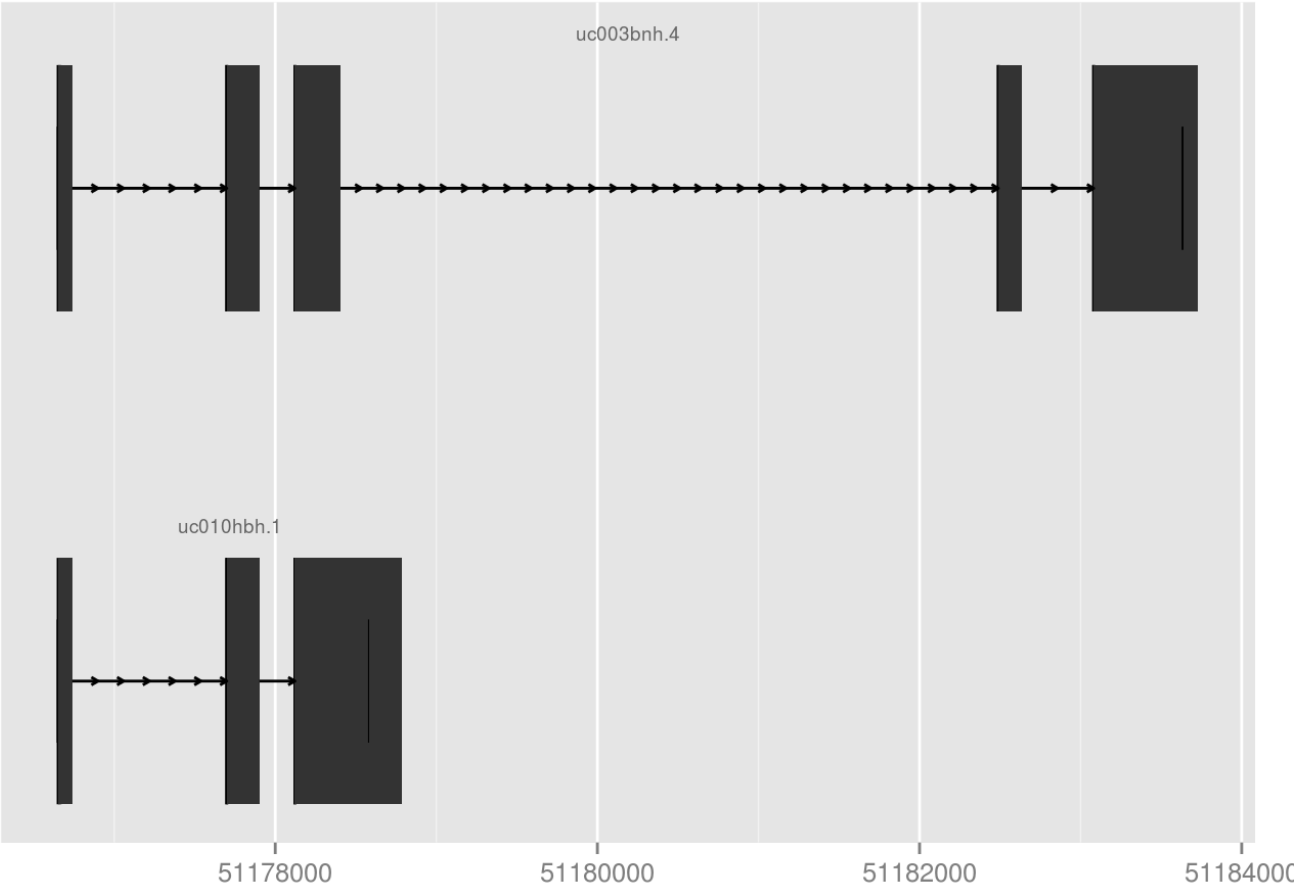
Can choose a summary statistic

```
autoplot(bam.sub,stat="coverage")
```



Plotting gene structure

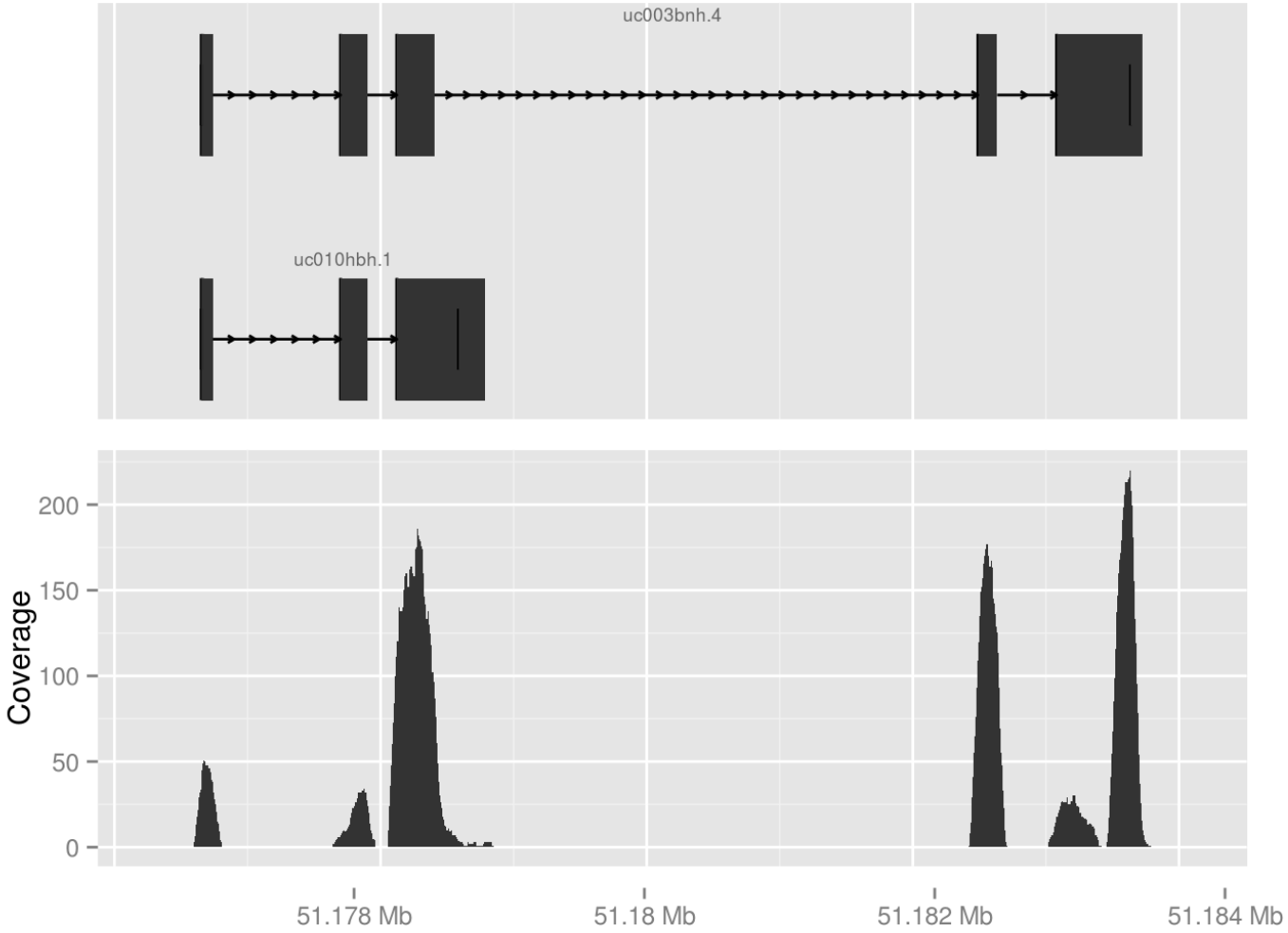
```
autoplot(txdb,which=exons[["49"]])
```



Combining plots

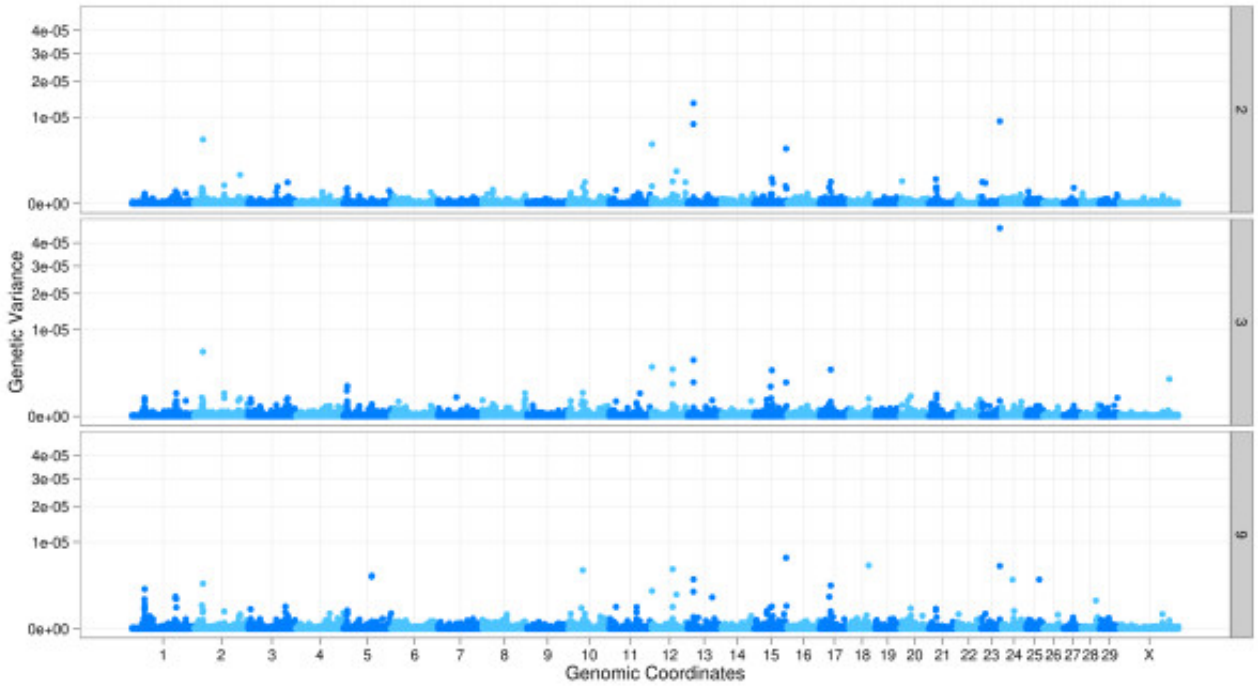
- plots made by `ggplot2` cannot be customised in the usual way with `par`
  - e.g. `par(mfrow=c(1,2))`
- `tracks` can do this job in `ggbio`
- x-axis structure is consistent between plots

```
tracks(autoplot(txdb,which=exons[["49"]]),
autoplot(bam.sub,stat="coverage"))
```



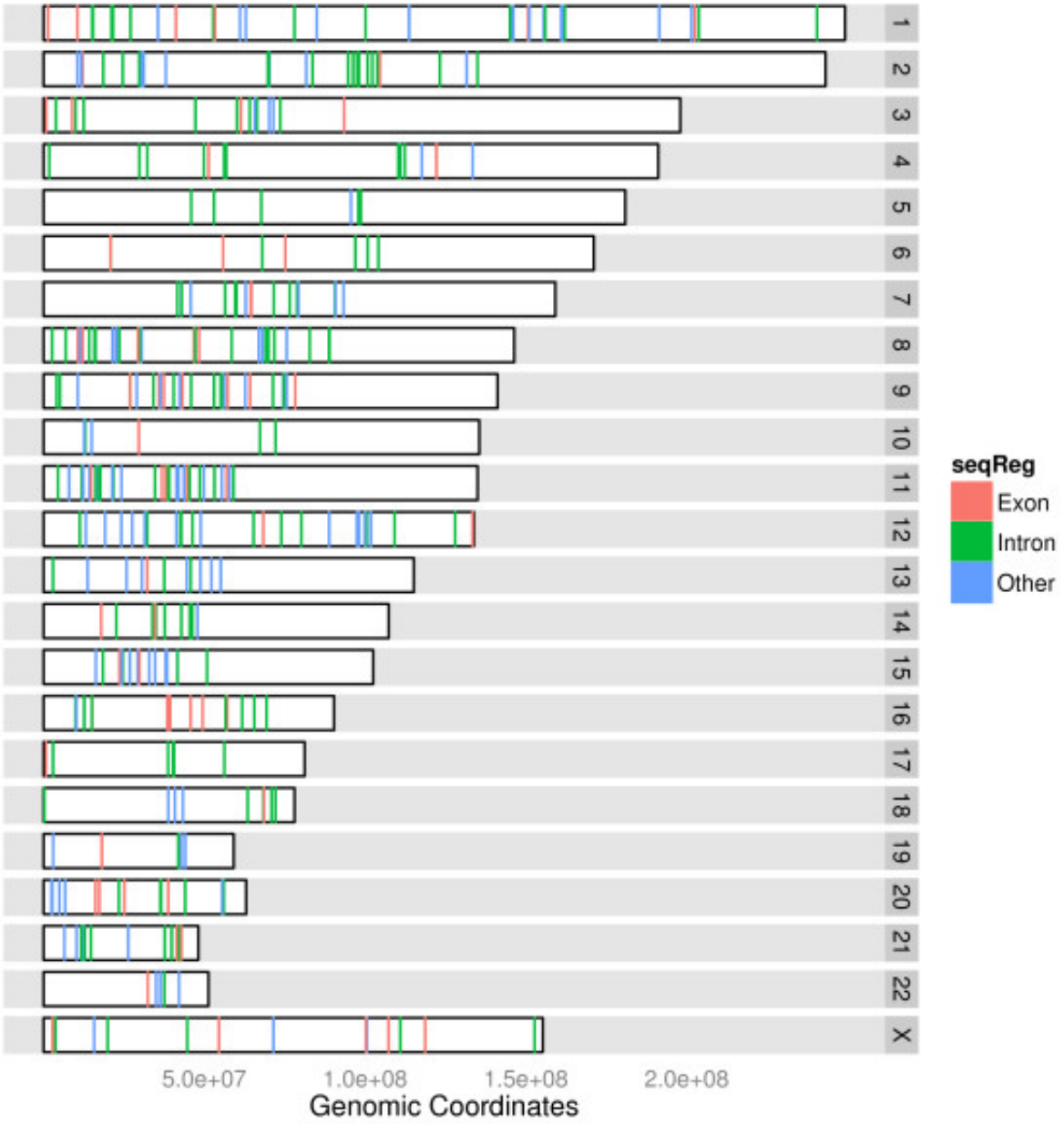
Different layouts available

- Can easily switch between different ploy layouts
  - geoms in ggplot2 terms
- Also set aesthetics from properties in the data
  - using `aes` ; like in `ggplot2`
- e.g. Manhattan plot

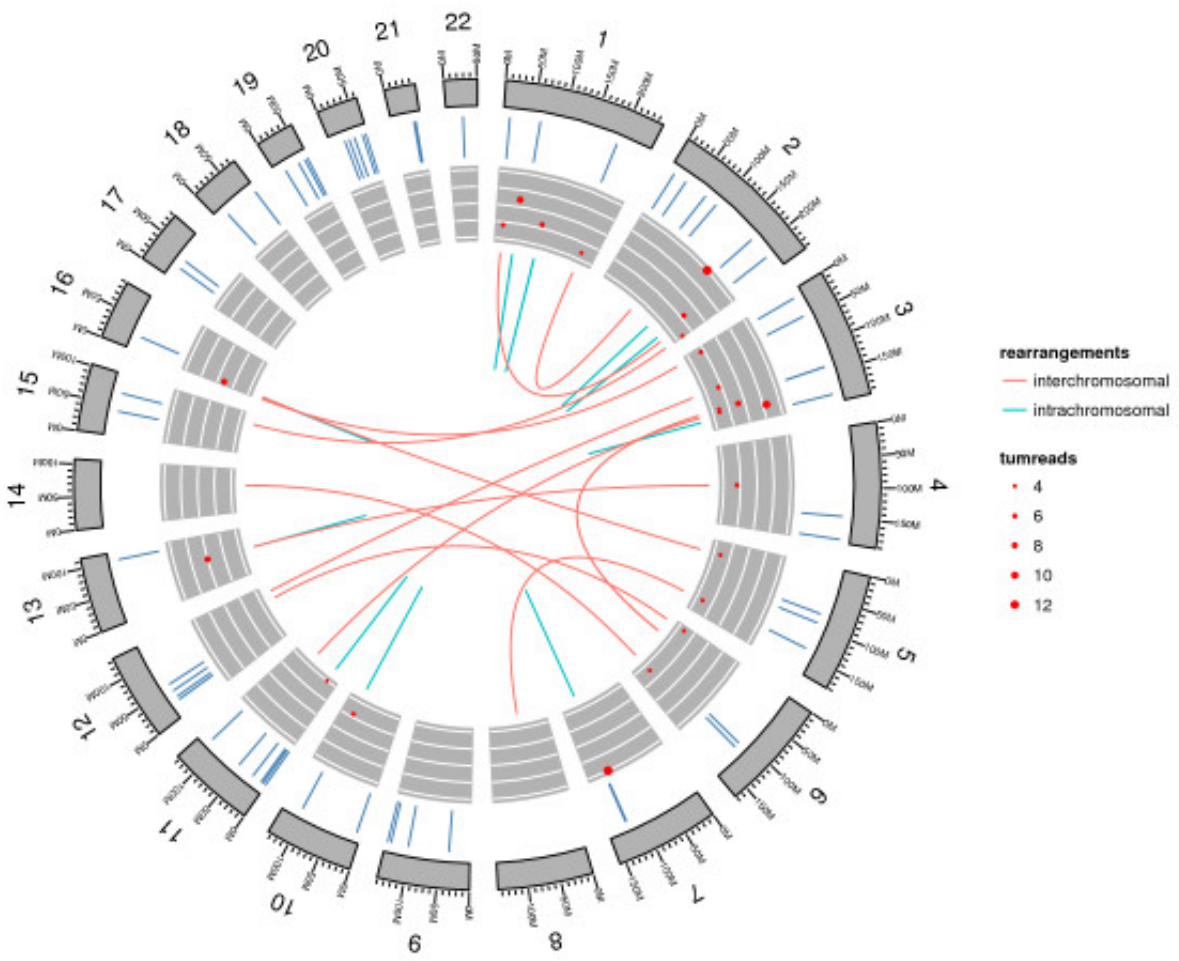


Karyograph

Karyogram



Circular



Practical time

- Use `ggplot2` and `ggbio` to explore the RNA-seq results
- Feel free to experiment
- Or check-out the vignettes for either package