Genomic Annotation and visualisation using R and Bioconductor

Mark Dunning

Cancer Research Uk
Cambridge Institute
Robinson Way
Cambridge

November 2, 2013

Outline

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

- Obtaining annotation information from different sources
  - Biomart
  - Pre-built Bioconductor packages
  - Browser tracks
- Visualise sequencing results and overlay with genomic annotations

- A wealth of annotation resources are available online through the biomart web software suite - www.biomart.org
- 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

Selecting a 'mart'

Need an internet connection for this to work!

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

##                biomart
## 1              ensembl
## 2                  snp
## 3 functional_genomics
## 4                 vega
## 5       fungi_mart_20
##                              version
## 1      ENSEMBL GENES 73 (SANGER UK)
## 2  ENSEMBL VARIATION 73 (SANGER UK)
## 3 ENSEMBL REGULATION 73 (SANGER UK)
## 4              VEGA 53  (SANGER UK)
## 5         ENSEMBL FUNGI 20 (EBI UK)

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

Select a dataset

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

##                          dataset
## 1          oanatinus_gene_ensembl
## 2           tguttata_gene_ensembl
## 3         cporcellus_gene_ensembl
## 4         gaculeatus_gene_ensembl
## 5          lafricana_gene_ensembl
## 6  itridecemlineatus_gene_ensembl
## 7         mlucifugus_gene_ensembl
## 8           hsapiens_gene_ensembl
## 9         choffmanni_gene_ensembl
## 10         csavignyi_gene_ensembl
##                                  description
## 1      Ornithorhynchus anatinus genes (OANA5)
## 2     Taeniopygia guttata genes (taeGut3.2.4)
## 3             Cavia porcellus genes (cavPor3)
## 4      Gasterosteus aculeatus genes (BROADS1)
## 5          Loxodonta africana genes (loxAfr3)
## 6  Ictidomys tridecemlineatus genes (spetri2)
## 7             Myotis lucifugus genes (myoLuc2)
```

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

listFilters(ensembl)[122, ]

##           name
## 122 entrezgene
##                            description
## 122 EntrezGene ID(s) [e.g. 100287163]
```

The information we can retrieve are known as attributes

```
head(listAttributes(ensembl), 5)

##                    name
## 1       ensembl_gene_id
## 2 ensembl_transcript_id
## 3    ensembl_peptide_id
## 4       ensembl_exon_id
## 5           description
##             description
## 1       Ensembl Gene ID
## 2 Ensembl Transcript ID
## 3    Ensembl Protein ID
## 4       Ensembl Exon ID
## 5           Description
```

Annotate a set of EntrezGene identifiers. *e.g. The results of a differential-expression analysis, or similar*.

```
entrez <- c("673", "837")
attr = c("entrezgene", "hgnc_symbol", "ensembl_gene_id",
    "description")
myInfo <- getBM(filters = "entrezgene", values = entrez,
    attributes = attr, mart = ensembl)
```

Give me the Symbol and Ensembl ID for genes with Entrez ID 673 and 837

```
head(myInfo)

##   entrezgene hgnc_symbol
## 1        673        BRAF
## 2        837       CASP4
##   ensembl_gene_id
## 1 ENSG00000157764
## 2 ENSG00000196954
##
## 1   v-raf murine sarcoma viral oncogene homolog B [Source:HGNC Symbo
## 2 caspase 4, apoptosis-related cysteine peptidase [Source:HGNC Symbo
```

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

```
getBM(c("ensembl_gene_id", "hgnc_symbol",
    "entrezgene"), filters = c("chromosome_name",
    "start", "end"), values = list(16, 1100000,
    1250000), mart = ensembl)[1:3, ]

##   ensembl_gene_id hgnc_symbol
## 1 ENSG00000261713   SSTR5-AS1
## 2 ENSG00000261720
## 3 ENSG00000181791
##   entrezgene
## 1     146336
## 2         NA
## 3         NA
```

Give me the ensembl, entrez and symbols of all genes between 1110000 and 1120000 on chromosome 16

Can also do the query the other way around

```r
getBM(c("ensembl_gene_id", "chromosome_name",
    "start_position", "end_position", "entrezgene"),
    filters = "ensembl_gene_id", values = c("ENSG00000261713",
        "ENSG00000261720", "ENSG00000181791"),
    ensembl)
```

```
##   ensembl_gene_id chromosome_name
## 1 ENSG00000181791              16
## 2 ENSG00000261713              16
## 3 ENSG00000261720              16
##   start_position end_position
## 1        1115299      1116349
## 2        1114093      1128707
## 3        1115240      1116502
##   entrezgene
## 1         NA
## 2     146336
## 3         NA
```

Many more examples in biomaRt vignette

But....

We had to define chromosome location in previous example

```
values = list(8, 148350, 148612)
```

- I'm doing my analysis using GRanges. Can't I use the object directly!
- Bioconductor provides a number of pre-built annotation resources for each organism
- What if I'm not on the internet?
- Bioconductor provides a number of pre-built annotation resources for each organism

Genome Representation

We have already seen that Genome sequences have an efficient representation in Bioconductor

```
library(BSgenome.Hsapiens.UCSC.hg19)
hg19 <- BSgenome.Hsapiens.UCSC.hg19
gr <- GRanges("chr16", IRanges(1100000, 1250000))
getSeq(hg19, gr)

##   A DNAStringSet instance of length 1
##      width seq
## [1] 150001 GAGACTCTGCTCT...TGGACTTGGGCTG
```

Give me the genome sequence between 1100000 and 1250000 on chromosome 16

Organism Packages

Bioconductor maintain a number of organism-level packages which are re-built every 6 months. A central identifier (Entrez gene id) is used.

```
library(org.Hs.eg.db)
cols(org.Hs.eg.db)[1:20]

##  [1] "ENTREZID"     "PFAM"
##  [3] "IPI"          "PROSITE"
##  [5] "ACCNUM"       "ALIAS"
##  [7] "CHR"          "CHRLOC"
##  [9] "CHRLOCEND"    "ENZYME"
## [11] "MAP"          "PATH"
## [13] "PMID"         "REFSEQ"
## [15] "SYMBOL"       "UNIGENE"
## [17] "ENSEMBL"      "ENSEMBLPROT"
## [19] "ENSEMBLTRANS" "GENENAME"
```

keytypes perform the same function as filters

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

##  [1] "ENTREZID"     "PFAM"
##  [3] "IPI"          "PROSITE"
##  [5] "ACCNUM"       "ALIAS"
##  [7] "CHR"          "CHRLOC"
##  [9] "CHRLOCEND"    "ENZYME"
## [11] "MAP"          "PATH"
## [13] "PMID"         "REFSEQ"
## [15] "SYMBOL"       "UNIGENE"
## [17] "ENSEMBL"      "ENSEMBLPROT"
## [19] "ENSEMBLTRANS" "GENENAME"
## [21] "UNIPROT"      "GO"
## [23] "EVIDENCE"     "ONTOLOGY"
## [25] "GOALL"        "EVIDENCEALL"
## [27] "ONTOLOGYALL"  "OMIM"
## [29] "UCSCKG"
```

Get the location of particular genes

```
entrez

## [1] "673" "837"

select(org.Hs.eg.db, keys = entrez, keytype = "ENTREZID",
    cols = c("SYMBOL", "CHR", "CHRLOC", "CHRLOCEND"))

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

Give me the genomic location of genes with Entrez ID 673 and 837

Genes for a particular GO term

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

Give with the Symbols of every gene with GO ontology GO:0003674

GenomicFeatures

- The GenomicFeatures package retrieves and manages transcript-related features from the UCSC Genome Bioinformatics 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

Pre-built packages

A full list of packages is available on the BioC website

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
```

Name of package indicates the organism, transcript source and genome build

```
txdb

## TranscriptDb object:
## | Db type: TranscriptDb
## | 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.p5
## | transcript_nrow: 80922
## | exon_nrow: 286852
## | cds_nrow: 235842
## | Db created by: GenomicFeatures package from Bioconductor
## | Creation time: 2013-03-08 09:43:09 -0800 (Fri, 08 Mar 2013)
## | GenomicFeatures version at creation time: 1.11.14
## | RSQLite version at creation time: 0.11.2
## | DBSCHEMAVERSION: 1.0
```

```
cols(txdb)

##  [1] "CDSID"       "CDSNAME"
##  [3] "CDSCHROM"    "CDSSTRAND"
##  [5] "CDSSTART"    "CDSEND"
##  [7] "EXONID"      "EXONNAME"
##  [9] "EXONCHROM"   "EXONSTRAND"
## [11] "EXONSTART"   "EXONEND"
## [13] "GENEID"      "TXID"
## [15] "EXONRANK"    "TXNAME"
## [17] "TXCHROM"     "TXSTRAND"
## [19] "TXSTART"     "TXEND"
```

```
keytypes(txdb)

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

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

```
##   GENEID  TXID   TXSTART     TXEND
## 1    673 30759 140433813 140624564
## 2    837 43899 104813594 104827422
## 3    837 43900 104813594 104839325
## 4    837 43901 104815475 104839325
## 5    837 43902 104819547 104839325
## 6    837 43903 104822124 104839325
```

Give my the transcrips for genes with Entrez ID 673 and 837

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

## Warning:  'select' resulted in 1:many mapping
## between keys and return rows

mygene

##    GENEID EXONID EXONCHROM EXONSTART
## 1     673 111268      chr7 140624366
## 2     673 111267      chr7 140549911
## 3     673 111266      chr7 140534409
## 4     673 111265      chr7 140508692
## 5     673 111264      chr7 140507760
## 6     673 111263      chr7 140501212
## 7     673 111262      chr7 140500162
## 8     673 111261      chr7 140494108
## 9     673 111260      chr7 140487348
## 10    673 111259      chr7 140482821
## 11    673 111258      chr7 140481376
## 12    673 111257      chr7 140477791
## 13    673 111256      chr7 140476712
## 14    673 111255      chr7 140453987
```

could then create a GRanges object from this

```
GRanges(mygene$EXONCHROM, IRanges(mygene$EXONSTART,
    mygene$EXONEND))

## GRanges with 18 ranges and 0 metadata columns:
##        seqnames                 ranges
##           <Rle>              <IRanges>
##    [1]     chr7 [140624366, 140624564]
##    [2]     chr7 [140549911, 140550012]
##    [3]     chr7 [140534409, 140534672]
##    [4]     chr7 [140508692, 140508795]
##    [5]     chr7 [140507760, 140507862]
##    ...      ...                    ...
##   [14]     chr7 [140453987, 140454033]
##   [15]     chr7 [140453075, 140453193]
##   [16]     chr7 [140449087, 140449218]
##   [17]     chr7 [140439612, 140439746]
##   [18]     chr7 [140433813, 140434570]
##        strand
##         <Rle>
##    [1]      *
##    [2]      *
##    [3]      *
##    [4]      *
```

Convenience Functions

An alternative is to retrieve all transcripts at once

```
trs <- transcripts(txdb)
trs[1:2]

## GRanges with 2 ranges and 2 metadata columns:
##       seqnames         ranges strand |
##          <Rle>      <IRanges>  <Rle> |
##   [1]     chr1 [11874, 14409]      + |
##   [2]     chr1 [11874, 14409]      + |
##            tx_id     tx_name
##        <integer> <character>
##   [1]          1  uc001aaa.3
##   [2]          2  uc010nxq.1
##   ---
##   seqlengths:
##             chr1 ... chrUn_gl000249
##        249250621 ...          38502
```

```
exons <- exonsBy(txdb, "gene")
exons[["146336"]]

## GRanges with 4 ranges and 2 metadata columns:
##       seqnames               ranges
##          <Rle>            <IRanges>
##   [1]    chr16 [1114082, 1116526]
##   [2]    chr16 [1116919, 1117043]
##   [3]    chr16 [1127624, 1127712]
##   [4]    chr16 [1128458, 1128731]
##       strand |   exon_id   exon_name
##        <Rle> | <integer> <character>
##   [1]      - |    207842        <NA>
##   [2]      - |    207843        <NA>
##   [3]      - |    207844        <NA>
##   [4]      - |    207846        <NA>
##   ---
##   seqlengths:
##              chr1 ... chrUn_gl000249
##         249250621 ...          38502
```

Or all exons

```
exs <- exons(txdb)
exs[1:2]

## GRanges with 2 ranges and 1 metadata column:
##       seqnames         ranges strand |
##          <Rle>      <IRanges>  <Rle> |
##   [1]     chr1 [11874, 12227]      + |
##   [2]     chr1 [12595, 12721]      + |
##         exon_id
##       <integer>
##   [1]         1
##   [2]         2
##   ---
##   seqlengths:
##               chr1 ... chrUn_gl000249
##          249250621 ...          38502
```

Grouping Genes

A functions exists to do this efficiently

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

## [1] "GRangesList"
## [2] "CompressedList"
## [3] "GenomicRangesList"
## [4] "GenomicRangesORGRangesList"
## [5] "List"
## [6] "Vector"
## [7] "Annotated"

length(exons)

## [1] 22932
```

see also transcriptsBy, intronsByTranscript, fiveUTRsByTranscript, threeUTRsByTranscript

The result can be subset by Gene ID (entrez)

```
exons[["673"]]

## GRanges with 18 ranges and 2 metadata columns:
##          seqnames                 ranges
##             <Rle>              <IRanges>
##      [1]     chr7 [140433813, 140434570]
##      [2]     chr7 [140439612, 140439746]
##      [3]     chr7 [140449087, 140449218]
##      [4]     chr7 [140453075, 140453193]
##      [5]     chr7 [140453987, 140454033]
##      ...      ...                    ...
##     [14]     chr7 [140507760, 140507862]
##     [15]     chr7 [140508692, 140508795]
##     [16]     chr7 [140534409, 140534672]
##     [17]     chr7 [140549911, 140550012]
##     [18]     chr7 [140624366, 140624564]
##          strand   |   exon_id   exon_name
##           <Rle>   | <integer> <character>
##      [1]      -   |    111251        <NA>
##      [2]      -   |    111252        <NA>
##      [3]      -   |    111253        <NA>
##      [4]      -   |    111254        <NA>
##      [5]      -   |    111255        <NA>
```

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"]]
```

Pass the GRanges object into the readGappedAlignments function

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

##    user  system elapsed
##   0.155   0.014   0.173
```

```
bam.sub

## GappedAlignments with 1917 alignments and 0 metadata columns:
##                      seqnames strand
##                         <Rle>  <Rle>
##    SRR076681.239386        22      -
##    SRR078452.251117        22      -
##    SRR076696.585674        22      -
##    SRR078501.824091        22      +
##    SRR078568.818440        22      +
##                 ...       ...    ...
##     SRR076132.39409        22      -
##    SRR076898.252854        22      -
##    SRR076176.943759        22      -
##     SRR076340.66381        22      -
##   SRR076936.1030386        22      -
##                            cigar
##                      <character>
##    SRR076681.239386        1S67M
##    SRR078452.251117          68M
##    SRR076696.585674          68M
##    SRR078501.824091          68M
##    SRR078568.818440          68M
##                 ...          ...
```

Extension

What if we want per-exon counts?

```
exonList <- split(gr, values(gr)$exon_id)
names(exonList)

## [1] "261128" "261129" "261130" "261131"
## [5] "261132" "261133"

exonList[[1]]

## GRanges with 1 range and 2 metadata columns:
##       seqnames               ranges
##          <Rle>            <IRanges>
##   [1]       22 [51176652, 51176740]
##       strand |   exon_id   exon_name
##        <Rle> | <integer> <character>
##   [1]      + |    261128        <NA>
##   ---
##   seqlengths:
##              chr1 ... chrUn_gl000249
##         249250621 ...          38502
```

```r
system.time(bam.sub2 <- lapply(exonList,
    function(x) readGappedAlignments(file = mybam,
        use.names = TRUE, param = ScanBamParam(which = x))))

##    user  system elapsed
##   0.409   0.056   0.475
```

```
names(bam.sub2)

## [1] "261128" "261129" "261130" "261131"
## [5] "261132" "261133"

bam.sub2[[1]]

## GappedAlignments with 91 alignments and 0 metadata columns:
##                     seqnames strand
##                        <Rle>  <Rle>
##   SRR076681.239386       22      -
##   SRR078452.251117       22      -
##   SRR076696.585674       22      -
##   SRR078501.824091       22      +
##   SRR078568.818440       22      +
##                  ...      ...    ...
##   SRR076578.648409       22      -
##   SRR076578.596591       22      -
##   SRR077073.807083       22      +
##   SRR076786.188214       22      -
##   SRR076099.491556       22      -
##                           cigar
##                     <character>
##   SRR076681.239386       1S67M
```

Retrieving gene sequences

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

##    user  system elapsed
##   0.843   0.055   0.943
```

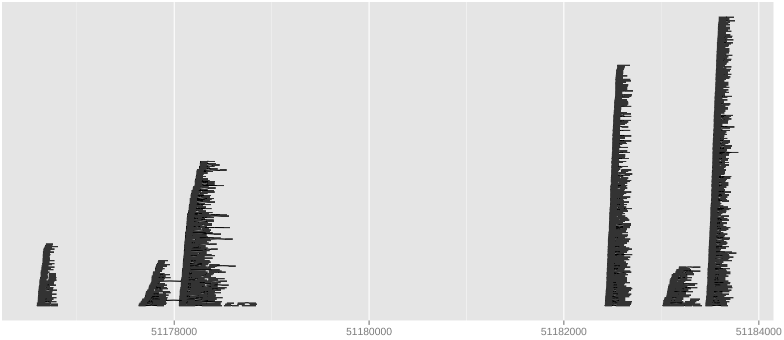
```
bam <- readGappedAlignments(file = mybam)
countOverlaps(gr, bam)

## Note:  method with signature 'Vector#GappedAlignments' chosen
for function 'countOverlaps',
##  target signature 'GRanges#GappedAlignments'.
##  "GenomicRanges#Vector" would also be valid

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

Visualisation - ggbip

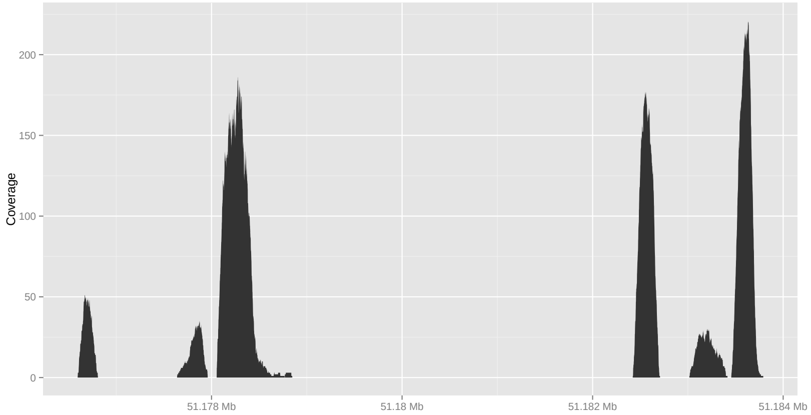
A consistent representation of ranges and genomic data helps with visualisation

- 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

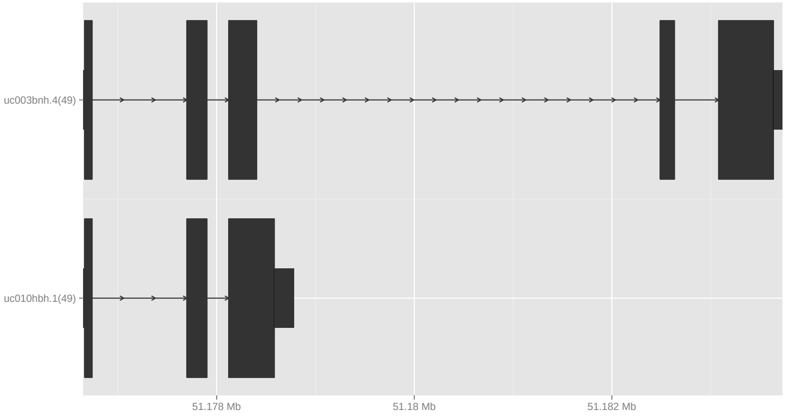
```r
library(ggbio)
autoplot(bam.sub)

## Object of class "ggbio"
## NULL
```



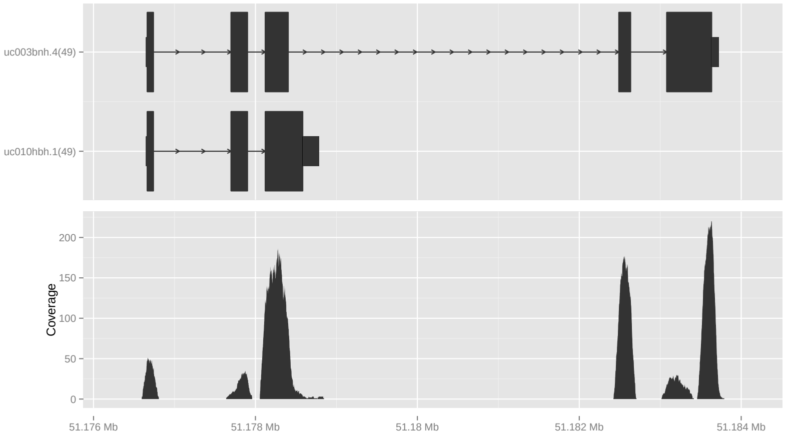
```
autoplot(bam.sub, stat = "coverage")

## Object of class "ggbio"
## NULL
```



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



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



This talk was brought to you by...

```
sessionInfo()

## R version 3.0.1 (2013-05-16)
## Platform: x86_64-apple-darwin10.8.0 (64-bit)
##
## locale:
## [1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics
## [4] grDevices utils     datasets
## [7] methods   base
##
## other attached packages:
##  [1] GeneticsHTSCourse2013_1.0
##  [2] ggbio_1.8.5
##  [3] ggplot2_0.9.3.1
##  [4] rtracklayer_1.20.4
##  [5] pasillaBamSubset_0.0.7
##  [6] TxDb.Dmelanogaster.UCSC.dm3.ensGene_2.9.0
##  [7] org.Dm.eg.db_2.9.0
```