Genomic Annotation and Visualisation using R and Bioconductor

Mark Dunning, Bioinformatics Training Coordinator

Cancer Research Uk
Cambridge Institute
Robinson Way
Cambridge

2nd June 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

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

- 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    regulation
## 4          vega
## 5 fungi_mart_26
##                             version
## 1      ENSEMBL GENES 79 (SANGER UK)
## 2  ENSEMBL VARIATION 79 (SANGER UK)
## 3 ENSEMBL REGULATION 79 (SANGER UK)
## 4               VEGA 59  (SANGER UK)
## 5         ENSEMBL FUNGI 26 (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         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
## 1      Ornithorhynchus anatinus genes (OANA5)
## 2            Cavia porcellus genes (cavPor3)
## 3      Gasterosteus aculeatus genes (BROADS1)
## 4          Loxodonta africana genes (loxAfr3)
## 5  Ictidomys tridecemlineatus genes (spetri2)
## 6        Choloepus hoffmanni genes (choHof1)
```

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 with_affy_hc_g110
##                                                description
## 122 with Affymetrix Microarray hc g110 probeset ID(s)
```

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        673        BRAF
## 3        837       CASP4
##   ensembl_gene_id
## 1 ENSG00000157764
## 2         LRG_299
## 3 ENSG00000196954
##
## 1   B-Raf proto-oncogene, serine/threonine kinase [Source:HGNC Symbo
## 2   B-Raf proto-oncogene, serine/threonine kinase [Source:HGNC Symbo
## 3 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 ENSG00000260702
## 2 ENSG00000260532
## 3 ENSG00000273551
##   entrezgene
## 1         NA
## 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

```
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 ENSG00000261713              16
## 2 ENSG00000261720              16
##   start_position end_position
## 1        1064093      1078731
## 2        1065240      1066502
##   entrezgene
## 1     146336
## 2         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)
columns(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] "ENZYME"       "MAP"
##  [9] "PATH"         "PMID"
## [11] "REFSEQ"       "SYMBOL"
## [13] "UNIGENE"      "ENSEMBL"
## [15] "ENSEMBLPROT"  "ENSEMBLTRANS"
## [17] "GENENAME"     "UNIPROT"
## [19] "GO"           "EVIDENCE"
## [21] "ONTOLOGY"     "GOALL"
## [23] "EVIDENCEALL"  "ONTOLOGYALL"
## [25] "OMIM"         "UCSCKG"
```

Get the location of particular genes

```
entrez

## [1] "673" "837"

select(org.Hs.eg.db, keys=entrez,
       keytype="ENTREZID",
       columns=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 -104827422
## 3        11 -104839325
```

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", 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
```

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

## 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
```

```
columns(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] "TXTYPE"     "TXCHROM"
## [19] "TXSTRAND"   "TXSTART"
## [21] "TXEND"
```

```
keytypes(txdb)
```

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

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

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

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

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

```
##    GENEID EXONID EXONCHROM EXONSTART
## 1     673 112179      chr7 140624366
## 2     673 112178      chr7 140549911
## 3     673 112177      chr7 140534409
## 4     673 112176      chr7 140508692
## 5     673 112175      chr7 140507760
## 6     673 112174      chr7 140501212
## 7     673 112173      chr7 140500162
## 8     673 112172      chr7 140494108
## 9     673 112171      chr7 140487348
## 10    673 112170      chr7 140482821
## 11    673 112169      chr7 140481376
## 12    673 112168      chr7 140477791
## 13    673 112167      chr7 140476712
## 14    673 112166      chr7 140453987
## 15    673 112165      chr7 140453075
## 16    673 112164      chr7 140449087
## 17    673 112163      chr7 140439612
## 18    673 112162      chr7 140433813
##      EXONEND
```

could then create a GRanges object from this

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

## GRanges object 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 object 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
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

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

## GRanges object 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]      - |    210263        <NA>
##   [2]      - |    210264        <NA>
##   [3]      - |    210265        <NA>
##   [4]      - |    210267        <NA>
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

Or all exons

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

## GRanges object 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
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

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] "GenomicRangesORGenomicRangesList"
## [7] "Vector"
## [8] "Annotated"

length(exons)

## [1] 23459
```

see also transcriptsBy, intronsByTranscript, fiveUTRsByTranscript, threeUTRsByTranscript

The result can be subset by Gene ID (entrez)

```
exons[["673"]]

## GRanges object 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]      -   |    112162        <NA>
##    [2]      -   |    112163        <NA>
##    [3]      -   |    112164        <NA>
##    [4]      -   |    112165        <NA>
##    [5]      -   |    112166        <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 <- readGAlignments(file = mybam,
    use.names = TRUE, param = ScanBamParam(which = gr)))
```

```
bam.sub

## GAlignments object 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] "263988" "263989" "263990" "263991"
## [5] "263992" "263993"

exonList[[1]]

## GRanges object with 1 range and 2 metadata columns:
##       seqnames              ranges
##          <Rle>           <IRanges>
##   [1]       22 [51176652, 51176740]
##       strand |   exon_id   exon_name
##        <Rle> | <integer> <character>
##   [1]      + |    263988        <NA>
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome

gr[1]
```

```
system.time(bam.sub2 <- lapply(exonList, function(x) readGAlignments(fi
                      use.names=TRUE,
                      param=ScanBamParam(which=x))))

##    user  system elapsed
##   0.375   0.017   0.389
```

```
names(bam.sub2)

## [1] "263988" "263989" "263990" "263991"
## [5] "263992" "263993"

bam.sub2[[1]]

## GAlignments object 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
##   0.178   0.005   0.181
```

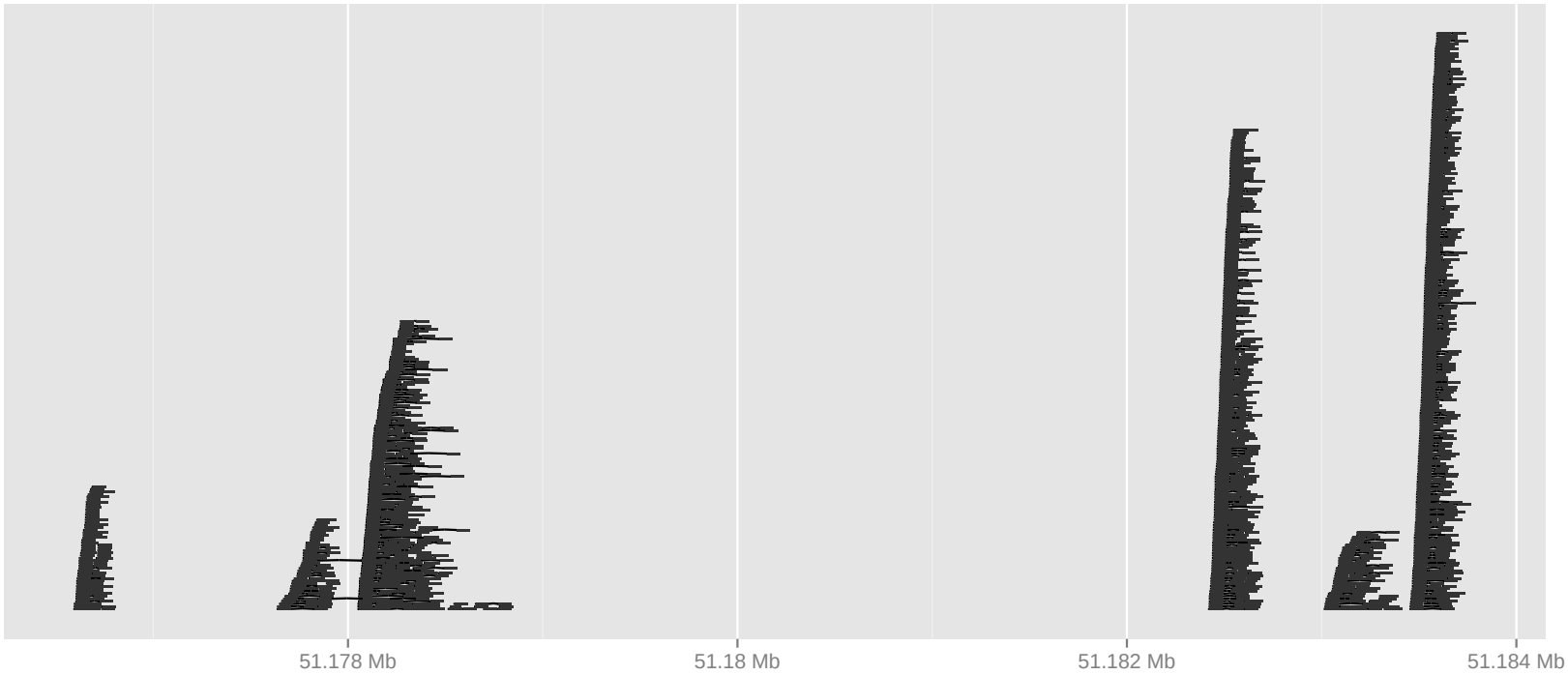
```
bam <- readGAlignments(file = mybam)
countOverlaps(gr, bam)

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

Visualisation - ggbio

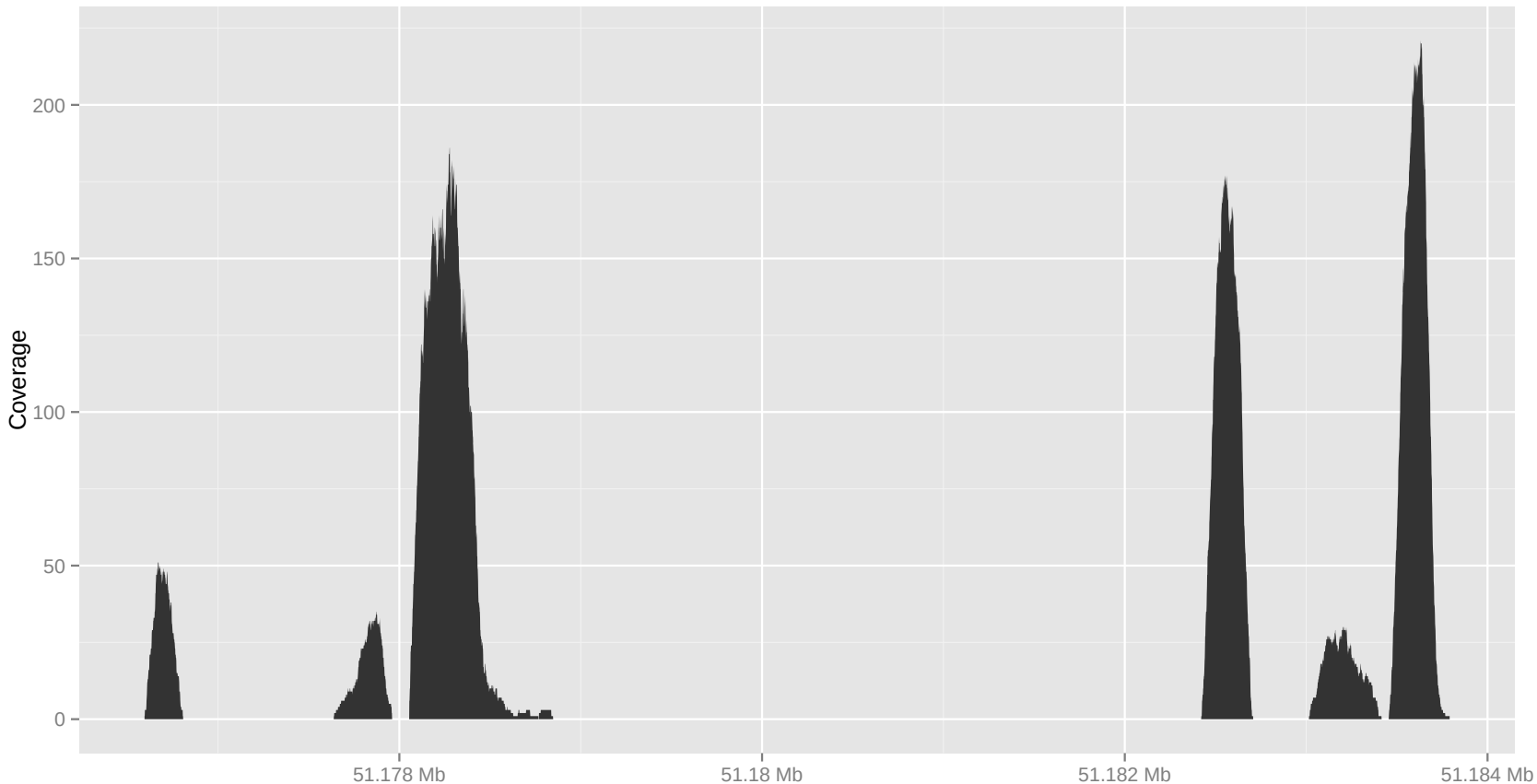
A consistent representation of ranges and genomic data helps with visualisation

- 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

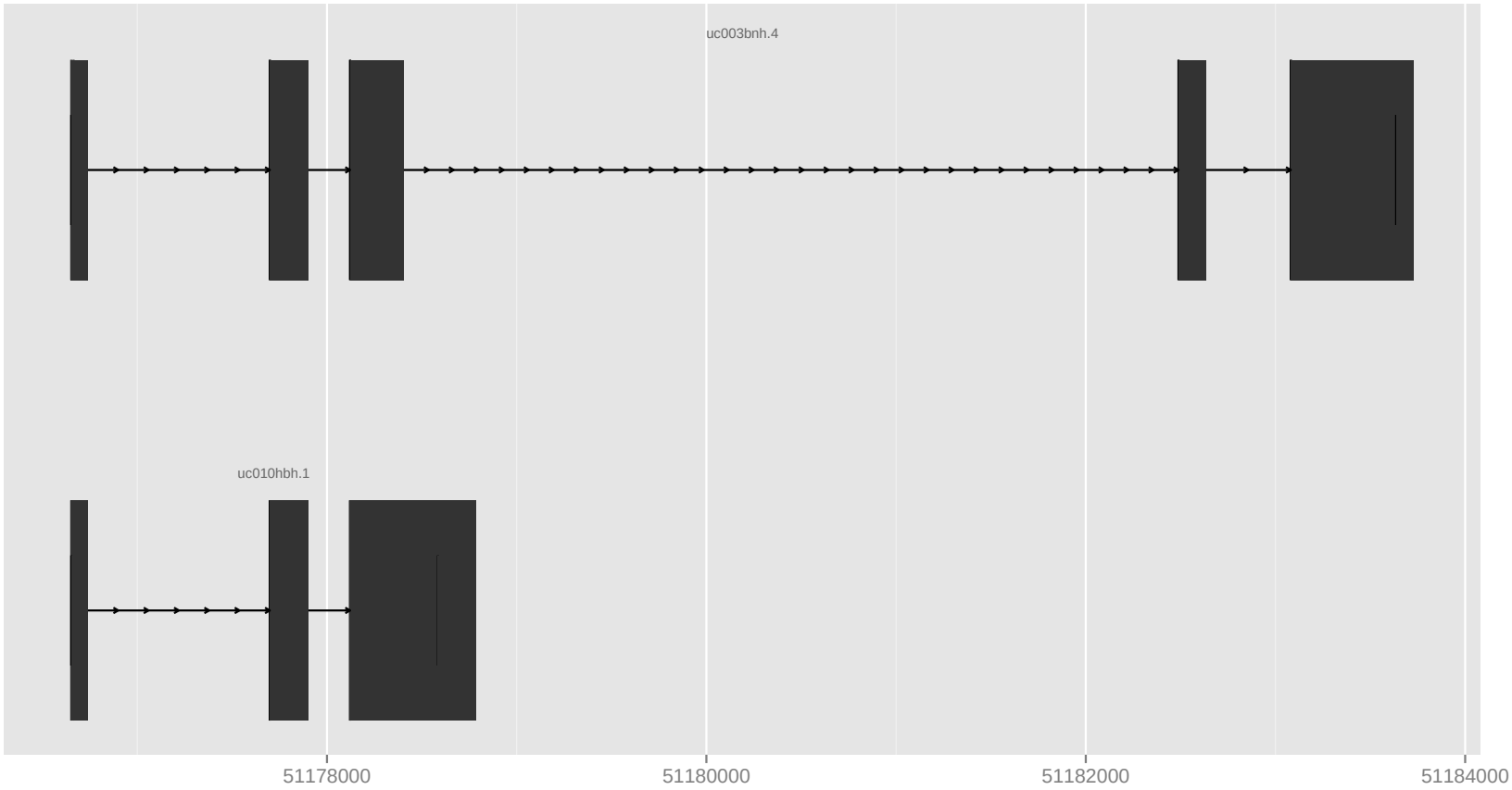
```
library(ggbio)
autoplot(bam.sub)
```



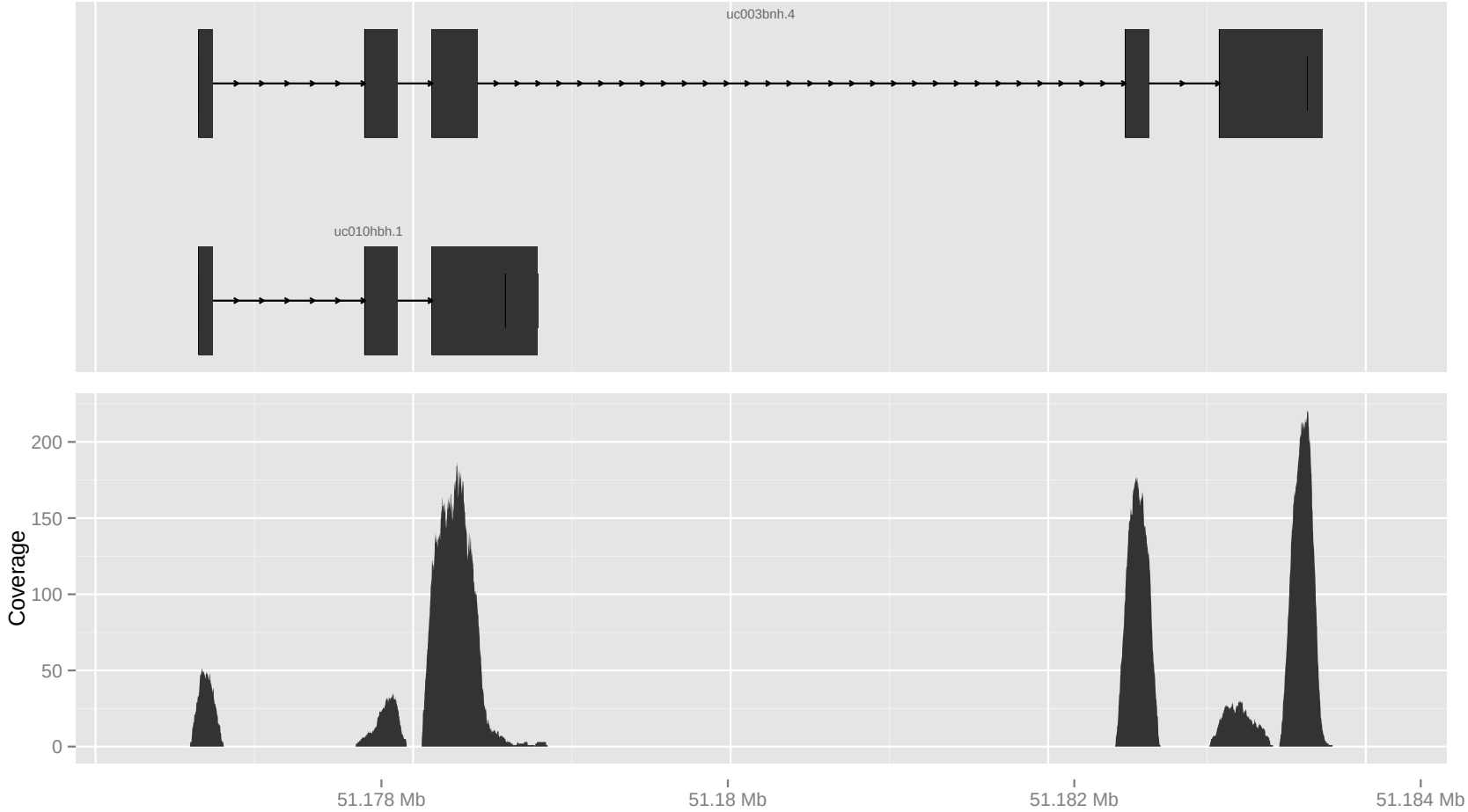
```
autoplot(bam.sub,stat="coverage")
```



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



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



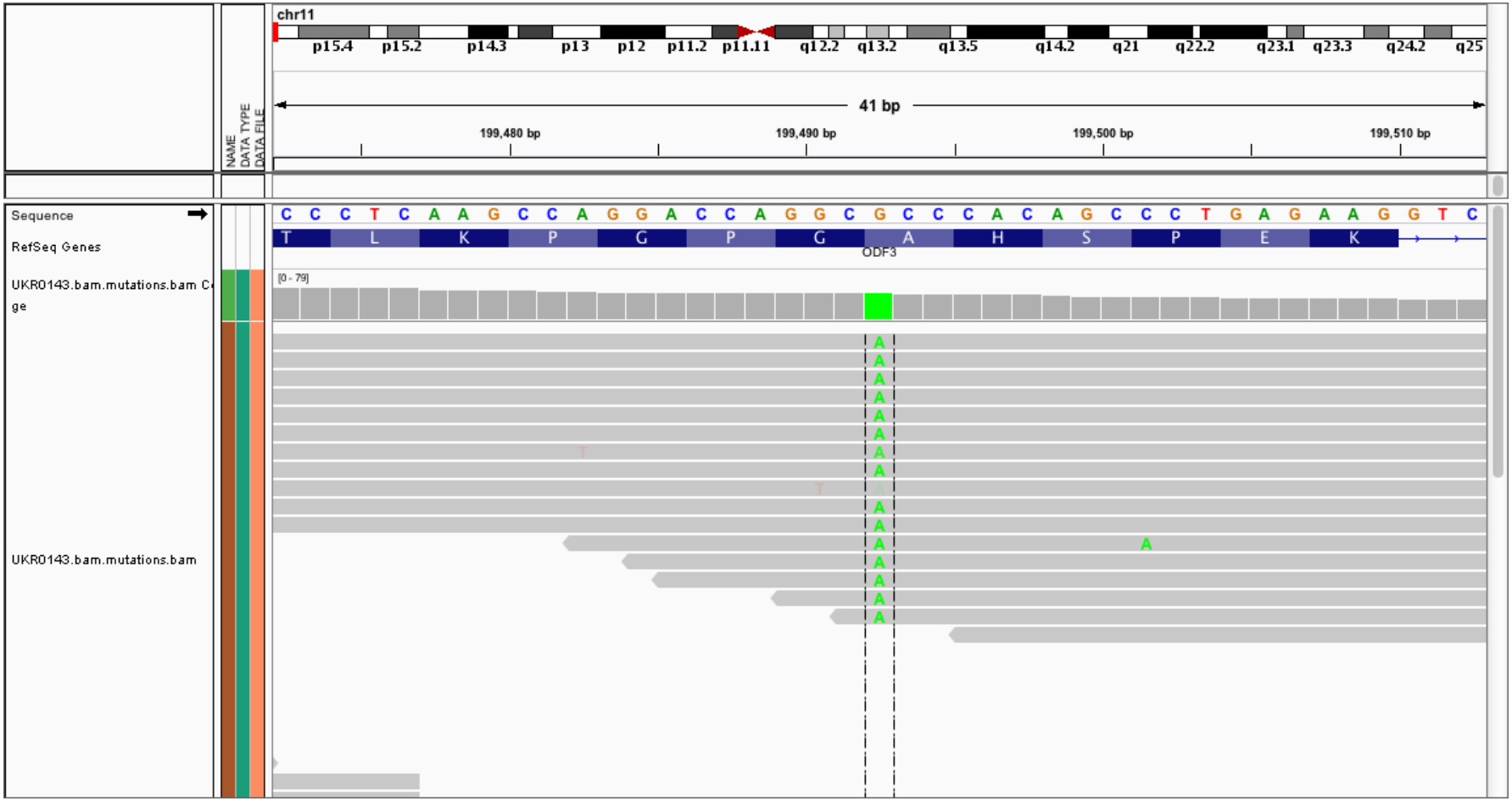
Types of Genomic Variation

- Germline variation
  - SNP: Single Nucelotide Polymorphism
  - Known variants are reported in dbSNP
  - indel: short insertion or deletion
  - copy number variation
- Somatic mutations
  - variation in cancer
  - SNV: single nucelotide variation
- Structural variation
  - Re-arrangements
  - Fusions
  - Large deletions / insertions

Problem definition

- Variant Calling
  - Discover differences between the genome and the reference
- Genotyping
  - Discover the genotype of the sequenced genome
- Differences
  - Between matched genomes (e.g. tumour and normal)
  - Differences between aligned genomes

How sequencing helps - SNPs



CANCER RESEARCH UK | CAMBRIDGE INSTITUTE

Bioconductor tools

Variant-calling is a developing area in Bioconductor

- VariantTools - Work in Progress
- VariantAnnotation - Import and manipulate variant calls
- VariantFiltering
- deepSNV - Determine SNVs from targeted sequencing
- ensemblVEP - Interface to ensembl

Variant Call Format

Emerging standard to capture results from DNA sequencing analysis.
Full details

- Format is tab-delimited with numerous header lines
- Each row is a different variant
  - FIXED - variant position, base change quality and filter
  - INFO - semicolon-separated series of short keys with optional values. e.g. Depth (DP), Allele Frequency (AF) and caller-specific output
  - Genotypes - One column per sample
- File can be indexed for easy access

The VariantAnnotation package is used to manipulate and process data in VCF format

```
library(VariantAnnotation)

vcf <- readVcf(myvcf, genome="hg19")
```

Can specify ranges to read in

`?readVcf`

Fixed

Retrieve the fixed information for each variant

```
head(fixed(vcf))
```

```
## DataFrame with 6 rows and 4 columns
##                REF                     ALT      QUAL      FILTER
##   <DNAStringSet> <DNAStringSetList> <numeric> <character>
## 1              A                   G       100        PASS
## 2              C                   T       100        PASS
## 3              G                   A       100        PASS
## 4              C                   T       100        PASS
## 5              C                   T       100        PASS
## 6              G                   A       100        PASS
```

info

```
head(info(vcf))
```

```
## DataFrame with 6 rows and 22 columns
##                    LDAF   AVGPOST       RSQ     ERATE
##               <numeric> <numeric> <numeric> <numeric>
## rs7410291        0.3431    0.9890    0.9856     2e-03
## rs147922003      0.0091    0.9963    0.8398     5e-04
## rs114143073      0.0098    0.9891    0.5919     7e-04
## rs141778433      0.0062    0.9950    0.6756     9e-04
## rs182170314      0.0041    0.9981    0.7909     7e-04
## rs115145310      0.0117    0.9975    0.9169     5e-04
##                   THETA          CIEND          CIPOS       END
##               <numeric> <IntegerList> <IntegerList> <integer>
## rs7410291        0.0005          NA,NA          NA,NA        NA
## rs147922003      0.0011          NA,NA          NA,NA        NA
## rs114143073      0.0008          NA,NA          NA,NA        NA
## rs141778433      0.0003          NA,NA          NA,NA        NA
## rs182170314      0.0004          NA,NA          NA,NA        NA
## rs115145310      0.0004          NA,NA          NA,NA        NA
##                     HOMLEN          HOMSEQ     SVLEN
##               <IntegerList> <CharacterList> <integer>
## rs7410291                                         NA
```

Descriptions of the info fields are given in the file header

```
hdr <- exptData(vcf)[["header"]]
info(hdr)[1:3,]
```

```
## DataFrame with 3 rows and 3 columns
##                Number        Type
##           <character> <character>
## LDAF                1       Float
## AVGPOST             1       Float
## RSQ                 1       Float
##                                                 Description
##                                                 <character>
## LDAF                MLE Allele Frequency Accounting for LD
## AVGPOST Average posterior probability from MaCH/Thunder
## RSQ       Genotype imputation quality from MaCH/Thunder
```

Genotypes

0 = Reference, 1 = Alternate

```
head(geno(vcf)$GT)

##             HG00096 HG00097 HG00099 HG00100 HG00101
## rs7410291   "0|0"   "0|0"   "1|0"   "0|0"   "0|0"
## rs147922003 "0|0"   "0|0"   "0|0"   "0|0"   "0|0"
## rs114143073 "0|0"   "0|0"   "0|0"   "0|0"   "0|0"
## rs141778433 "0|0"   "0|0"   "0|0"   "0|0"   "0|0"
## rs182170314 "0|0"   "0|0"   "0|0"   "0|0"   "0|0"
## rs115145310 "0|0"   "0|0"   "0|0"   "0|0"   "0|0"

table(geno(vcf)$GT[,1])

##
##  0|0  0|1  1|0  1|1
## 9407  394  200  375
```

Genomic postions are given by `rowData`

```
head(rowData(vcf),3)
```

```
## GRanges object with 3 ranges and 5 metadata columns:
##                seqnames              ranges strand |
##                   <Rle>           <IRanges>  <Rle> |
##     rs7410291       22 [50300078, 50300078]      * |
##   rs147922003       22 [50300086, 50300086]      * |
##   rs114143073       22 [50300101, 50300101]      * |
##                paramRangeID            REF
##                    <factor> <DNAStringSet>
##     rs7410291          <NA>              A
##   rs147922003          <NA>              C
##   rs114143073          <NA>              G
##                                ALT      QUAL      FILTER
##                <DNAStringSetList> <numeric> <character>
##     rs7410291                   G       100        PASS
##   rs147922003                   T       100        PASS
##   rs114143073                   A       100        PASS
##   -------
##   seqinfo: 1 sequence from hg19 genome; no seqlengths
```

- We can now retrieve variants in GRanges in format
- ....we can also get transcripts as GRanges
- ....and genome sequence
- ....so we can annotate our variants easily

`locateVariants` does the hard work of locating variants with respect to gene function. N.B. Take care over chromosome naming.

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
seqlevels(vcf) <- c("22"="chr22")
rd <- rowData(vcf)

## Warning:  'rowData' is deprecated.
## Use 'rowRanges' instead.
## See help("Deprecated")

loc <- locateVariants(rd, txdb, CodingVariants())
```

Can also choose *IntronVariants*, *FiveUTRVariants*, *ThreeUTRVariants*, *IntergenicVariants*, *SpliceSiteVariants* or *PromoterVariants*

```
head(loc,3)
```

```
## GRanges object with 3 ranges and 9 metadata columns:
##     seqnames              ranges strand | LOCATION
##        <Rle>           <IRanges>  <Rle> | <factor>
##   1    chr22 [50301422, 50301422]      * |   coding
##   2    chr22 [50301476, 50301476]      * |   coding
##   3    chr22 [50301488, 50301488]      * |   coding
##      LOCSTART    LOCEND   QUERYID         TXID          CDSID
##     <integer> <integer> <integer> <character> <IntegerList>
##   1       939       939        24       75253        218562
##   2       885       885        25       75253        218562
##   3       873       873        26       75253        218562
##           GENEID       PRECEDEID        FOLLOWID
##     <character> <CharacterList> <CharacterList>
##   1       79087
##   2       79087
##   3       79087
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

Predicting Consequences

```
library(BSgenome.Hsapiens.UCSC.hg19)
coding <- predictCoding(vcf, txdb, seqSource=Hsapiens)
coding[1]
```

```
## GRanges object with 1 range and 17 metadata columns:
##                seqnames               ranges strand |
##                   <Rle>            <IRanges>  <Rle> |
##   rs114335781    chr22 [50301422, 50301422]      - |
##                paramRangeID              REF
##                    <factor> <DNAStringSet>
##   rs114335781         <NA>               G
##                                ALT      QUAL      FILTER
##                <DNAStringSetList> <numeric> <character>
##   rs114335781                   A       100        PASS
##                     varAllele     CDSLOC    PROTEINLOC
##                <DNAStringSet>  <IRanges> <IntegerList>
##   rs114335781               T [939, 939]           313
##                  QUERYID        TXID         CDSID
##                <integer> <character> <IntegerList>
##   rs114335781        24       75253        218562
##                    GENEID CONSEQUENCE       REFCODON
##                <character>    <factor> <DNAStringSet>
```

```
table(coding$CONSEQUENCE)

##
##    frameshift      nonsense nonsynonymous    synonymous
##             2            17          1535          1268
```

- synonymous - no change in amino acid
- nonsynonymous - change in amino acid
- nonsense - premature stop codon
- frameshift- number of nucleotides in a DNA sequence that is not divisible by three

Checking for novelty

```
library(SNPlocs.Hsapiens.dbSNP.20101109)
chr22Snps <- getSNPlocs("ch22",as.GRanges = TRUE)
chr22Snps

## GRanges object with 331060 ranges and 2 metadata columns:
##             seqnames                ranges strand   |
##                <Rle>             <IRanges>  <Rle>   |
##         [1]     ch22 [16050353, 16050353]      *    |
##         [2]     ch22 [16050994, 16050994]      *    |
##         [3]     ch22 [16051107, 16051107]      *    |
##         [4]     ch22 [16051209, 16051209]      *    |
##         [5]     ch22 [16051241, 16051241]      *    |
##         ...      ...                   ...    ... ...
##    [331056]     ch22 [51239222, 51239222]      *    |
##    [331057]     ch22 [51239281, 51239281]      *    |
##    [331058]     ch22 [51239296, 51239296]      *    |
##    [331059]     ch22 [51239304, 51239304]      *    |
##    [331060]     ch22 [51239324, 51239324]      *    |
##               RefSNP_id alleles_as_ambig
##              <character>      <character>
##         [1]    56342815                K
##         [2]     7288968                S
```