Using the *GenomicFeatures* package

Marc Carlson

Fred Hutchinson Cancer Research Center

July 30th 2010
Outline

Introduction

GenomicFeatures

A simple RNA-seq example
Bioconductor Annotation Packages

- PLATFORM PKGS
- GENE ID
- HOMOLOGY PKGS
- GENE ID
- SYSTEM BIOLOGY (GO, KEGG)
- ONTO ID’S
- ORG PKGS
- GENE ID
- ONTO ID
- TRANSCRIPT DBS
- GENE ID
Bioconductor Sequence Packages

- Rsamtools
- ShortRead
- GenomicRanges
- IRanges
- Biostrings
- BSgenome
- rtracklayer
- GenomicFeatures
Bioconductor Sequence Annotations

The GenomicFeatures package

- Transcript information stored in SQLite databases
- These databases attempt to represent the relational nature of splicing data correctly
- Transcript information exposed to user via the GenomicRanges infrastructure
- Parsers are available to construct these databases from biomaRt or the UCSC genome browser. Also there is a generic parser for custom jobs.
Outline

Introduction

GenomicFeatures

A simple RNA-seq example
GenomicFeatures transcript sources

Constructors

makeTranscriptDbFromBiomart, makeTranscriptDbFromUCSC

> library(GenomicFeatures)
> nrow(supportedUCSCtables())

[1] 24

> head(supportedUCSCtables(), 10)

<table>
<thead>
<tr>
<th>track</th>
<th>subtrack</th>
</tr>
</thead>
<tbody>
<tr>
<td>knownGene</td>
<td>UCSC Genes</td>
</tr>
<tr>
<td>knownGeneOld3</td>
<td>Old UCSC Genes</td>
</tr>
<tr>
<td>wgEncodeGencodeAutoRel2</td>
<td>Gencode Genes</td>
</tr>
<tr>
<td>wgEncodeGencodePolyaRel2</td>
<td>Gencode Genes</td>
</tr>
<tr>
<td>ccdsGene</td>
<td>Consensus CDS</td>
</tr>
<tr>
<td>refGene</td>
<td>RefSeq Genes</td>
</tr>
<tr>
<td>xenoRefGene</td>
<td>Other RefSeq</td>
</tr>
<tr>
<td>vegaGene</td>
<td>Vega Genes</td>
</tr>
<tr>
<td>vegaPseudoGene</td>
<td>Vega Genes</td>
</tr>
</tbody>
</table>
TranscriptDb basics

Making a TranscriptDb object

```r
> mm9KG <-
+   makeTranscriptDbFromUCSC(genome = "mm9",
+                              tablename = "knownGene")
```

Saving and Loading

```r
> saveFeatures(mm9KG, file="mm9KG.sqlite")

> mm9KGChr9 <-
+   loadFeatures(system.file("extdata", "mm9KG.sqlite",
+                               package = "HTSandGeneCentricLabs"))
```
TranscriptDb class

> mm9KGChr9

TranscriptDb object:
| Db type: TranscriptDb
| Data source: UCSC
| Genome: mm9
| UCSC Table: knownGene
| Type of Gene ID: Entrez Gene ID
| Full dataset: no
| transcript_nrow: 2910
| exon_nrow: 14110
| cds_nrow: 12270
| Db created by: GenomicFeatures package from Bioconductor
| Creation time: 2010-05-13 17:02:30 -0700 (Thu, 13 May 2010)
| GenomicFeatures version at creation time: 1.1.1
| RSQLite version at creation time: 0.8-3
TranscriptDb schema

transcript
  _tx_id INTEGER PRIMARY KEY,
  tx_name TEXT NULL,
  tx_chrom TEXT NOT NULL,
  tx_strand TEXT NOT NULL,
  tx_start INTEGER NOT NULL,
  tx_end INTEGER NOT NULL,
  FOREIGN KEY (tx_chrom) REFERENCES chrominfo (chrom)

cds
  _cds_id INTEGER PRIMARY KEY,
  cds_name TEXT NULL,
  cds_chrom TEXT NOT NULL,
  cds_strand TEXT NOT NULL,
  cds_start INTEGER NOT NULL,
  cds_end INTEGER NOT NULL,
  FOREIGN KEY (cds_chrom) REFERENCES chrominfo (chrom)

gene
  gene_id TEXT NOT NULL,
  _tx_id INTEGER NOT NULL,
  UNIQUE (gene_id, _tx_id),
  FOREIGN KEY (_tx_id) REFERENCES transcript

splicing
  _tx_id INTEGER NOT NULL,
  exon_rank INTEGER NOT NULL,
  _exon_id INTEGER NOT NULL,
  _cds_id INTEGER NULL,
  UNIQUE (_tx_id, exon_rank),
  FOREIGN KEY (_tx_id) REFERENCES transcript

exon
  _exon_id INTEGER PRIMARY KEY,
  exon_name TEXT NULL,
  exon_chrom TEXT NOT NULL,
  exon_strand TEXT NOT NULL,
  exon_start INTEGER NOT NULL,
  exon_end INTEGER NOT NULL,
  FOREIGN KEY (exon_chrom) REFERENCES chrominfo (chrom)
Ungrouped transcript-related information

Extractors

transcripts, exons, cds

```r
> tx <- transcripts(mm9KGChr9)
> length(tx)

[1] 2910

> head(tx, 5)

GRanges with 5 ranges and 2 elementMetadata values

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>tx_id</th>
<th>tx_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr9</td>
<td>[3215314, 3215339]</td>
<td>+</td>
<td>24312</td>
<td>uc009oas.1</td>
</tr>
<tr>
<td>chr9</td>
<td>[3335231, 3385846]</td>
<td>+</td>
<td>24315</td>
<td>uc009oat.1</td>
</tr>
<tr>
<td>chr9</td>
<td>[3335473, 3343608]</td>
<td>+</td>
<td>24313</td>
<td>uc009oau.1</td>
</tr>
<tr>
<td>chr9</td>
<td>[3335473, 3380423]</td>
<td>+</td>
<td>24314</td>
<td>uc009oav.1</td>
</tr>
<tr>
<td>chr9</td>
<td>[3335478, 3385846]</td>
<td>+</td>
<td>24316</td>
<td>uc009oaw.1</td>
</tr>
</tbody>
</table>

seqlengths

<table>
<thead>
<tr>
<th>chr1</th>
<th>chr2</th>
<th>chrX_random</th>
<th>chrY_random</th>
</tr>
</thead>
<tbody>
<tr>
<td>197195432</td>
<td>181748087</td>
<td>...</td>
<td>1785075</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>58682461</td>
</tr>
</tbody>
</table>
```
Grouped transcript-related information

Extractors

transcriptsBy, exonsBy, cdsBy, intronsByTranscript,
fiveUTRsByTranscript, threeUTRsByTranscript

> txExons <- exonsBy(mm9KGChr9)
> txIntrons <- intronsByTranscript(mm9KGChr9)
> txExons[6]

GRangesList of length 1
$24313
GRanges with 3 ranges and 3 elementMetadata values

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>exon_id</th>
<th>exon_name</th>
<th>exon_rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr9</td>
<td>[3335473, 3335594]</td>
<td>+</td>
<td>117005</td>
<td>NA</td>
<td>1</td>
</tr>
<tr>
<td>chr9</td>
<td>[3338456, 3338591]</td>
<td>+</td>
<td>117006</td>
<td>NA</td>
<td>2</td>
</tr>
<tr>
<td>chr9</td>
<td>[3343015, 3343608]</td>
<td>+</td>
<td>117007</td>
<td>NA</td>
<td>3</td>
</tr>
</tbody>
</table>

seqlengths

chr1  197195432
chr2  181748087
...   ...
chrX_random  1785075
chrY_random  58682461
Standard GRanges accessors can be used

Accessors
names, length, elementMetaData, width, ranges, start, end

Examples:

>`head(start(tx))`

[1] 3215314 3335231 3335473 3335473 3335478 3379207

>`head(ranges(txExons), n=1)`

CompressedIRangesList of length 1

$`/\text{24308}`$

IRanges of length 1

<table>
<thead>
<tr>
<th>start</th>
<th>end</th>
<th>width</th>
</tr>
</thead>
<tbody>
<tr>
<td>3186316</td>
<td>3186344</td>
<td>29</td>
</tr>
</tbody>
</table>

>`head(elementMetadata(tx), n=2)`

Dataframe with 2 rows and 2 columns

<table>
<thead>
<tr>
<th>tx_id</th>
<th>tx_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;integer&gt;</td>
<td>&lt;character&gt;</td>
</tr>
<tr>
<td>1</td>
<td>uc009oas.1</td>
</tr>
<tr>
<td>2</td>
<td>uc009oat.1</td>
</tr>
</tbody>
</table>
Overlapping with transcripts

Methods
findOverlaps, countOverlaps, match, %in%, subsetByOverlaps

Usage and help:

> findOverlaps(query, subject, maxgap = 0L, minoverlap = 1L,
+    type = c("any", "start", "end"),
+    select = c("all", "first"))
> help("findOverlaps,GRanges,GRangesList-method")

Example

> grngs <- GRanges("chr9", gaps(ranges(txExons[[6]])), "+")
> countOverlaps(grngs, tx)
[1] 4 4

> rbind(countOverlaps(grngs, txExons), countOverlaps(grngs, txIntrons))
     [,1] [,2]
[1,]    1    0
GenomicFeatures exercise 1

Load the transcriptDB object above and then gather the annotations for transcripts as grouped by their genes.
GenomicFeatures exercise 1 solution

```r
> library(GenomicFeatures)
> mm9KGChr9 <- loadFeatures(system.file("extdata", "mm9KG.sqlite",
+                          package = "HTSandGeneCentricLabs"))
> myTranscripts <- transcriptsBy(mm9KGChr9, by="gene")
```
Outline

Introduction

GenomicFeatures

A simple RNA-seq example
A simple RNA-seq example

1st lets just load some data

readBamGappedAlignments

```r
> library(Rsamtools)
> testFile <- system.file("bam", "isowt5_13e.bam", + package="leeBamViews")
> aligns <- readBamGappedAlignments(testFile)
> head(aligns)
```

GappedAlignments of length 6

<table>
<thead>
<tr>
<th>rname</th>
<th>strand</th>
<th>cigar</th>
<th>qwidth</th>
<th>start</th>
<th>end</th>
<th>width</th>
<th>ngap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scchr13</td>
<td>-</td>
<td>36M</td>
<td>36</td>
<td>799975</td>
<td>800010</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>Scchr13</td>
<td>-</td>
<td>36M</td>
<td>36</td>
<td>799977</td>
<td>800012</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>Scchr13</td>
<td>-</td>
<td>36M</td>
<td>36</td>
<td>799979</td>
<td>800014</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>Scchr13</td>
<td>-</td>
<td>36M</td>
<td>36</td>
<td>799980</td>
<td>800015</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>Scchr13</td>
<td>+</td>
<td>36M</td>
<td>36</td>
<td>799986</td>
<td>800021</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>Scchr13</td>
<td>+</td>
<td>36M</td>
<td>36</td>
<td>799986</td>
<td>800021</td>
<td>36</td>
<td>0</td>
</tr>
</tbody>
</table>
RNA-seq

Retrieve corresponding Annotations

makeTranscriptDbFromUCSC

```r
> library(GenomicFeatures)
> txdb <- makeTranscriptDbFromUCSC(genome="sacCer2",
+   tablename="sgdGene")
```

extract exons by transcripts

exonsBy

```r
> exonRanges <- exonsBy(txdb, "tx")
> length(exonRanges)
[1] 6717
> exonRanges[1]

GRangesList of length 1

$1

GRanges with 1 range and 3 elementMetadata values

seqnames       ranges strand | exon_id   exon_name

<Rle>  <IRanges>  <Rle>  | <integer> <character>

[1] chrI [130802, 131986] + | 1      NA

exon_rank
```


Correct for chromosome naming

```r
levels

> levels(rname(aligns))

[1] "Scchr01" "Scchr02" "Scchr03" "Scchr04" "Scchr05" "Scchr06"
[7] "Scchr07" "Scchr08" "Scchr09" "Scchr10" "Scchr11" "Scchr12"
[13] "Scchr13" "Scchr14" "Scchr15" "Scchr16" "Scmito"

> levels(rname(aligns)) <-
+   c(paste("chr", as.roman(1:16), sep=""), "chrM")
```
Now that we have the data and annotations stored in the appropriate objects, use the appropriate method to count the overlaps between them.
RNA-seq: GenomicFeatures exercise 2 solution

Use countOverlaps to count Reads

countOverlaps

> txdb <- makeTranscriptDbFromUCSC(genome="sacCer2", + tablename="sgdGene")
> exonRanges <- exonsBy(txdb, "tx")
> library("Rsamtools")
> testFile <- system.file("bam", "isowt5_13e.bam", + package="leeBamViews")
> aligns <- readBamGappedAlignments(testFile)
> levels(rname(aligns))
> levels(rname(aligns)) <-
+ c(paste("chr", as.roman(1:16), sep=""), "chrM")
> counts <- countOverlaps(exonRanges, aligns)
Can then calculate RPKM

\[
\begin{align*}
\text{numBases} & \leftarrow \text{sum(width(exonRanges))} \\
\text{geneLengthsInKB} & \leftarrow \text{numBases} / 1000 \\
\text{millionsMapped} & \leftarrow \text{sum(counts)} / 1000000 \\
\text{rpm} & \leftarrow \text{counts} / \text{millionsMapped} \\
\text{rpkm} & \leftarrow \text{rpm} / \text{geneLengthsInKB}
\end{align*}
\]
Results can be sorted

> sortedRPKM <- sort(rpkms)
> highScoreGenes <- tail(sortedRPKM)
Now that we have some of the things that were measured the most, we want to know what they are. We don’t want to use the transcript ID’s though because they are not guaranteed to be meaningful outside of this database. What we want instead are the gene IDs. So how can we use GenomicFeatures to extract the gene IDs matched to the transcript IDs?
RNA-seq: GenomicFeatures exercise 3 solution

```r
> ##All the RPKM stuff since count and then:
> txs <- transcripts(txdb,
+       vals=list(tx_id=names(highScoreGenes)),
+       columns=c("tx_id","gene_id"))
> systNames <- as.vector(
+       unlist(elementMetadata(txs)["gene_id"]))
```
RNA-seq

use other annot’s to learn more

> library(org.Sc.sgd.db)
> toTable(org.Sc.sgdGENENAME[systNames])

<table>
<thead>
<tr>
<th>systematic_name</th>
<th>gene_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 YMR295C</td>
<td>IBI2</td>
</tr>
<tr>
<td>2 YMR297W</td>
<td>PRC1</td>
</tr>
<tr>
<td>3 YMR305C</td>
<td>SCW10</td>
</tr>
<tr>
<td>4 YMR307W</td>
<td>GAS1</td>
</tr>
</tbody>
</table>

(combine into sets as needed)

> deSeqframe <- data.frame(counts, counts2)