The *Ranges Suite

Use cases and examples from high-throughput sequencing

Michael Lawrence

Genentech

July 28, 2011
Outline

1. Introduction
2. ChIP-seq
3. RNA-seq
4. Conclusion
Sequencing Approaches

Source  Genome, transcriptome, synthetic

Enrichment  WGS, ChIP, PCR, poly-A RNA, exome capture, etc
1. QA on raw instrument output, see *ShortRead*
2. Usually, external alignment of data, i.e., gSNAP
3. Import of alignments and/or sequences into R
4. Analysis of sequences, alignments and enrichment patterns
The *Ranges Packages

**IRanges**
Base of the sequence analysis infrastructure in Bioconductor
- Data structures for interval datasets and genome-scale vectors
- Routines for finding regions of enrichment and overlap between features

**GenomicRanges**
Extension of *IRanges* for genomic (biological) datasets, including sequence annotations and experimental measurements
Outline

1. Introduction
2. ChIP-seq
3. RNA-seq
4. Conclusion
ChIP-seq Protocol

1. Cell Nucleus
2. Crosslink Protein and Shear DNA
3. Add Protein-Specific Antibody
4. Immunoprecipitate and purify complexes
5. Reverse Crosslinks, Purify DNA and prepare for sequencing
6. Sequence DNA fragment and map to genome
ChIP-seq Questions

• Where are the peaks?
• Do the peaks tend to fall in a certain genomic context, e.g., promoters or conserved regions?
• How do the peaks correspond to and inform TF motif analyses?
• Is there a relationship between binding and expression?
Workflow Overview

- ShortRead QA report on reads
- Read alignment (i.e., gSNAP)
- Import of read alignments
- Resize alignments to estimated mean fragment length
- Calculate coverage
- Estimate peak cutoff and call peaks
- Various peak-level analyses
Workflow Overview

- ShortRead QA report on reads
- Read alignment (i.e., gSNAP)
- Import of read alignments
- Resize alignments to estimated mean fragment length
- Calculate coverage
- Estimate peak cutoff and call peaks
- Various peak-level analyses
Example Data: `data(cstest)`

- Solexa sequencing of CTCF and GFP (control) ChIP in mouse
- Aligned with MAQ
- One lane each, subset to three chromosomes
Workflow Overview

1. ShortRead QA report on reads
2. Read alignment (i.e., gSNAP)
3. Import of read alignments
4. Resize alignments to estimated mean fragment length
5. Calculate coverage
6. Estimate peak cutoff and call peaks
7. Various peak-level analyses
Representing Read Alignments

Read alignments are intervals on stranded sequences

<table>
<thead>
<tr>
<th>Sequence Name</th>
<th>Start</th>
<th>End</th>
<th>Strand</th>
<th>metadata...</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr10</td>
<td>3012936</td>
<td>3012959</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>chr10</td>
<td>3012941</td>
<td>3012964</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>chr10</td>
<td>3012944</td>
<td>3012967</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>
Read alignments are intervals on stranded sequences

<table>
<thead>
<tr>
<th>Sequence Name</th>
<th>Start</th>
<th>End</th>
<th>Strand</th>
<th>metadata...</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr10</td>
<td>3012936</td>
<td>3012959</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>chr10</td>
<td>3012941</td>
<td>3012964</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>chr10</td>
<td>3012944</td>
<td>3012967</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

All genomic data fits this basic format
Genomic Datasets in R

The **GRanges** Class

A *Vector* of genomic intervals, with metadata

- Constructor: `GRanges(seqnames, ranges, strand, ...)`
- `seqnames(x)`: sequence name
- `start(x), end(x), width(x)`: interval information
- `strand(x)`: strand (+/-/*)
- `values(x)`: a *Data*Frame of metadata columns, like score or gene
- `seqinfo(x)`: a *Seqinfo* with information about the sequences

Group multiple *GRanges* in a *GRangesList*
Loading the CTCF Data

> library(chipseq)
> data(cstest)
> names(cstest)

[1] "ctcf" "gfp"

> head(cstest$ctcf, 1)

GRanges with 1 range and 0 element Metadata values

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Rle&gt;</td>
<td>&lt;IRanges&gt;</td>
<td>&lt;Rle&gt;</td>
</tr>
<tr>
<td>[1]</td>
<td>chr10 [3012936, 3012959] +</td>
<td></td>
</tr>
</tbody>
</table>

seqlengths

<table>
<thead>
<tr>
<th>chr1</th>
<th>chr1_random</th>
<th>...</th>
<th>chrM</th>
</tr>
</thead>
<tbody>
<tr>
<td>197195432</td>
<td>1231697</td>
<td>...</td>
<td>16299</td>
</tr>
</tbody>
</table>
1. Count the number of reads on each chromosome
2. Count the number of reads on each strand
3. Sort the reads by start position
Peak-level Analyses

- Summarize peaks
- Intersect with genomic annotations
- Visualization
Workflow Overview

1. ShortRead QA report on reads
2. Read alignment (i.e., gSNAP)
3. Import of read alignments
4. Resize alignments to estimated mean fragment length
5. Calculate coverage
6. Call regions of enrichment (peaks)
7. Various peak-level analyses
Estimating Fragment Length

```r
> fraglen <-
+   estimate.mean.fraglen(cstest$ctcf,
+                          method = "correlation",
+                          seqLen = 35)

> fraglen
chr10  chr11  chr12
   265    265    255

> median(fraglen)
[1] 265
```
> ctcf.ext <- resize(cstest$ctcf, 
+       width = median(fraglen))
> head(ctcf.ext, 3)

GRanges with 3 ranges and 0 elementMetadata values

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Rle&gt;</td>
<td>&lt;IRanges&gt;</td>
<td>&lt;Rle&gt;</td>
</tr>
</tbody>
</table>
[1] chr10  [3012936, 3013200] + |
[2] chr10  [3012941, 3013205] + |
[3] chr10  [3012944, 3013208] + |

seqlengths

<table>
<thead>
<tr>
<th>chr1</th>
<th>chr1_random ...</th>
<th>chrM</th>
</tr>
</thead>
<tbody>
<tr>
<td>197195432</td>
<td>1231697 ...</td>
<td>16299</td>
</tr>
</tbody>
</table>
Workflow Overview

1. ShortRead QA report on reads
2. Read alignment (i.e., gSNAP)
3. Import of read alignments
4. Resize alignments to estimated mean fragment length
5. Calculate coverage
6. Call regions of enrichment (peaks)
7. Various peak-level analyses
Representing Coverage

- As often the case with genomic data, coverage contains long runs of identical values
- Could use a *GRanges*, with a range for each run
- More efficient is a Run-Length Encoding (RLE)
- Already an *rle* class in R, but lacks functionality
The Rle Class

A Vector that run-length encodes any atomic vector type, e.g., logical, integer, character, etc.

- Constructor: `Rle(x)`
- Usually treated like any other R vector
- `runLength`, `runValue`: get the lengths and values

Multiple chromosomes fit into an `RleList`
Calculating the Coverage

```r
> cov.ctcf <- coverage(ctcf.ext)
> cov.ctcf$chr10

'integer' Rle of length 129993255 with 289551 runs
  Lengths: 3012734  97  ...  265  6212
  Values :   0  1  ...  1  0
```
Calculate how many elements were saved through the run-length encoding vs. an ordinary vector.
Workflow Overview

1. ShortRead QA report on reads
2. Read alignment (i.e., gSNAP)
3. Import of read alignments
4. Resize alignments to estimated mean fragment length
5. Calculate coverage
6. Call regions of enrichment (peaks)
7. Various peak-level analyses
Finding a peak cutoff is a complex problem

Many peak detection methods rely on "island" summaries

An island is a contiguous region with depth $\geq 1$

Analyzing the islands, or any set of enriched regions, requires combining the coverage with the ranges of interest
Combining a Vector with Ranges

The Views Class

A Vector of views, by overlaying a set of Ranges on a subject Vector

The RleViews Class

A Views subclass with an Rle subject; useful for coverage

The RleViewsList Class

A (Views)List of RleViews; useful for coverage over multiple chromosomes
Slicing the Coverage into Islands

The slice Function

```r
> islands <- slice(cov.ctcf, lower = 1)
> head(islands$chr10, 3)
```

Views on a 129993255-length Rle subject

views:

<table>
<thead>
<tr>
<th>start</th>
<th>end</th>
<th>width</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3012735</td>
<td>3013335</td>
<td>601</td>
<td>[1 1 1 1 1 1 1 1 1 ...]</td>
</tr>
<tr>
<td>3018464</td>
<td>3018728</td>
<td>265</td>
<td>[1 1 1 1 1 1 1 1 1 ...]</td>
</tr>
<tr>
<td>3020766</td>
<td>3021030</td>
<td>265</td>
<td>[1 1 1 1 1 1 1 1 1 ...]</td>
</tr>
</tbody>
</table>
Calling the Peaks

Assume we ended up choosing a cutoff of 8.

```r
> peak_viewsList <- slice(cov.ctcf, lower = 8)
> peak_rangesList <- ranges(peak_viewsList)
> peaks <- as(peak_rangesList, "GRanges")
> head(peaks, 3)

GRanges with 3 ranges and 0 elementMetadata values

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Rle&gt;</td>
<td>&lt;IRanges&gt;</td>
<td>&lt;Rle&gt;</td>
</tr>
</tbody>
</table>
[1] chr10  [3012955, 3013200]  *  |
[2] chr10  [3234798, 3234896]  *  |
[3] chr10  [3269945, 3270362]  *  |

seqlengths

<table>
<thead>
<tr>
<th>chr1</th>
<th>chr1_random</th>
<th>...</th>
<th>chrM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>NA</td>
<td>...</td>
<td>NA</td>
</tr>
</tbody>
</table>
Island Calling Exercise

Generate a similar `GRanges` for the GFP lane
Calling Peaks in Four Lines

```r
> findPeaks <- function(reads) {
+   fraglen <- estimate.mean.fraglen(reads, 
+       method = "correlation", 
+       seqLen = 35)
+   reads_ext <- resize(reads, fraglen)
+   cov <- coverage(reads_ext)
+   slice(cov, lower = 8)
+ }
```
Workflow Overview

1. ShortRead QA report on reads
2. Read alignment (i.e., gSNAP)
3. Import of read alignments
4. Resize alignments to estimated mean fragment length
5. Calculate coverage
6. Call regions of enrichment (peaks)
7. Various peak-level analyses
Summarizing Coverage by Peaks

Some statistics of interest:

- Maximal coverage under peak
- Total coverage under peak
- Summit interval
Summarizing RleViews(List) of Peaks

Maximal coverage under peak

```r
> values(peaks)$max <-
+ unlist(viewMaxs(peak_viewsList))
```

Sum of coverage under peak

```r
> values(peaks)$sum <-
+ unlist(viewSums(peak_viewsList))
```

Find summits

```r
> values(peaks)$summits <-
+ unlist(viewRangeMaxs(peak_viewsList))
```

Calling a view* summary function on a RleViewsList returns a List, which we unlist.
Peak Annotation

- Genomic context (promoters, exons, etc)
- Motif hits
- Conservation
- ...
Many ways to represent transcript models as ranges:
- Transcripts
- Exons, introns
- CDS, UTRs

Reference annotations, so prefer persistent storage
The *TxDb* Class

Reference to SQLite DB with transcript information

- `transcripts(x)`: whole transcript ranges
- `exons(x)`: exon ranges
- `cds(x)`: coding exon ranges
- `*By(x)` variants: `GRangesList` object, grouping by transcript, gene
- `*ByOverlaps(x)` variants: annotations overlapping query ranges

For common organisms/models, *TxDb* packages available
Obtaining the Mouse Transcripts

Find regions 500 bp upstream, 200 downstream of TSS

```r
> library(TxDB.Mmusculus.UCSC.mm9.knownGene)
> mm9_tx <- transcripts(Mmusculus_UCSC_mm9_knownGene_TxDB,
+ columns = "gene_id")
```
Fixing up the gene_id Column

- Sometimes possible for transcript to belong to multiple genes
- Not the case for our mouse genes
- Need to coerce the *CharacterList* to *character*

```r
> gene_id <- values(mm9_tx)$gene_id
> all(elementLengths(gene_id) <= 1)
[1] TRUE

> flat_gene_id <- character(length(mm9_tx))
> values(mm9_tx)$gene_id <- flat_gene_id
```
Obtaining the Mouse Promoters

Find regions 500 bp upstream, 200 downstream of TSS

> promoters <- resize(flank(mm9_tx, 500), 700)
Finding Peaks that Overlap the Promoters

%in%: any overlap?

```r
> values(peaks)$in_promoter <- peaks %in% promoters
> table(values(peaks)$in_promoter)

FALSE  TRUE
5622   391
```

match: find index of first overlap

```r
> values(peaks)$in_promoter_of <-
+   values(promoters)$gene_id[match(peaks, promoters)]
> head(subset(values(peaks), in_promoter)$in_promoter_of, 3)

[1] "270685" "67844" ""
```
Peak Annotation Exercises

1. Find the peaks in the region 10kb upstream of the TSS
2. Find the peaks in the introns (i.e., in transcript, but not exons)
Comparing Peaks Across Samples

```r
> ctcf_peaks <- findPeaks(cstest$ctcf)
> gfp_peaks <- findPeaks(cstest$gfp)
> peak_summary <- diffPeakSummary(ctcf_peaks, gfp_peaks)
> colnames(peak_summary)
[1] "comb.max" "sums1"   "sums2"   "maxs1"
[5] "maxs2"
```
Creating a *SummarizedExperiment*

The range-aware version of *ExpressionSet*

```r
> peak_summary_gr <- as(ranges(peak_summary), "GRanges")
> max_matrix <- with(peak_summary, cbind(maxs1, maxs2))
> SummarizedExperiment(max_matrix, rowData = peak_summary_gr)
```

```
class: SummarizedExperiment
dim: 6021 2
assays(1): ''
rownames: NULL
rowData values names(0):
colnames(2): maxs1 maxs2
colData names(0):
```
Outline

1. Introduction
2. ChIP-seq
3. RNA-seq
4. Conclusion
RNA-seq Questions

- Which genes, exons, alleles, isoforms, etc are expressed, and which are differentially expressed?
- Are there any expressed variants/editing?
- Are there any novel splicing events?
Workflow Overview

- QA, alignment
- Import of alignments
- Counting of reads/coverage over various intervals
Workflow Overview

- QA, alignment
- Import of alignments
- Counting of reads/coverage over various intervals
Data from *leeBamViews*

- Four samples from a Yeast RNA-seq experiment
- Two wildtype, two RLP mutants
- Alignments in *leeBamViews* for positions 800000 to 900000 on chromosome XIII
- Stored as BAM files
Workflow Overview

- QA, alignment
- Import of alignments
- Counting of reads/coverage over various intervals
Workflow Overview

• QA, alignment
• Import of alignments
• Counting of reads/coverage over various intervals
Could store SAM alignments in a *GRanges*, but common enough for a formal representation

The *GappedAlignments* Class

A *Vector* of alignments with SAM-specific fields

- Load with `readGappedAlignments(file)`
- Access `cigar`, `qpos`, etc.
- Often acts like *GRanges*, with `start(x)`, `coverage(x)`, etc.
Loading a BAM File

`library(leeBamViews)`

`bams <- dir(system.file("bam", package="leeBamViews"),` +
`  full = TRUE, pattern = "bam$")`

`reads_ga <- readGappedAlignments(bams[1])`

`head(reads_ga, 1)`

GappedAlignments of length 1

```
  rname strand cigar qwidth start end
[1] Scchr13    -   36M     36 799975 800010
```

```
  width ngap
[1]     36   0
```

seqlengths

```
  Scchr01 Scchr02 Scchr03  ...  Scchr16 Scmito
  230208  813178  316617  ...  948062  85779
```
Workflow Overview

- QA, alignment
- Import of alignments
- Counting of reads/coverage over various intervals
Workflow Overview

- QA, alignment
- Import of alignments
- Counting of reads/coverage over various intervals
Representing Ranges with Gaps

- Read alignments may contain gaps, e.g., cross a splice junction
- Multiple ranges per read
- Need to group ranges by read
- Use GRangesList, with one GRanges per read
Creating a \textit{GRangesList} from \textit{GappedAlignments}

\begin{verbatim}
> reads_grl <- grglist(reads_ga)
\end{verbatim}
Creating a `GRangesList` from `GappedAlignments`

```r
> reads_grl <- grglist(reads_ga)
```

**Note**

This groups by read, not by read pair. Pair grouping currently takes a little more work.
Prep 1ing Some Transcri  nt Models

Reads were not mapped to UCSC sacCer2 assembly, so we use some annotations of expressed regions from *leeBamViews*

```
> data(leeUnn)
> leeUnn <-
+  subset(leeUnn, lengthWithoutMask > 0 & !is.na(chr))
> leeUnn$strand <- c("-", "*", "+")[leeUnn$strand + 2]
> sc2_tx <- with(leeUnn,
+  GRanges(sprintf("Scchr%02d", chr),
+    IRanges(start, end), strand))
> seqlevels(sc2_tx)[length(seqlevels(sc2_tx))] <- "Scmito"
```
Count Reads in Exons, by Transcript

Using `countOverlaps`

```r
> values(sc2_tx)$counts <- 
+   countOverlaps(sc2_tx, reads_grl, ignore.strand = TRUE)
```

These counts could then be passed to *DEseq* or *edgeR*
Find Transcript Hits for Each Read

Using `findOverlaps`

```r
> ol <- findOverlaps(reads_grl, sc2_tx,
+                     ignore.strand = TRUE)
> reads_factor <- factor(queryHits(ol),
+                        seq_len(length(reads_grl)))
> values(reads_grl)$tx_hits <-
+    split(subjectHits(ol), reads_factor)
> head(table(elementLengths(values(reads_grl)$exon_hits)))
integer(0)
```
Outline

1. Introduction
2. ChIP-seq
3. RNA-seq
4. Conclusion
Further Analysis

ChIP-seq Annotation  *ChIPpeakAnno, chipseq*

RNA-seq DE  *edgeR, DEseq, DEXseq*