Annotations, ChIP-seq and Bioconductor The rtracklayer package

Michael Lawrence

November 12, 2008

Introduction

2 Manipulating Genomic Data (Tracks)

3 Interacting with a Genome Browser

4 Conclusion

・ロト・西ト・ヨト・ヨー うへで

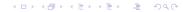
Outline

Introduction

2 Manipulating Genomic Data (Tracks)

Interacting with a Genome Browser





Annotations and ChIP-seq

- Often want to leverage existing genomic knowledge in ChIP-seq analysis
- Examples:
 - Genes and exons (previous talk)
 - Conservation scores (which peaks are conserved?)
 - Transcription factor binding sites, TransFac (any nearby binding partners?)
- Need to integrate and visualize ChIP-seq data (coverage, islands, ...) with annotations

▲□▶ ▲□▶ ▲□▶ ▲□▶ □ ○ ○ ○

The rtracklayer package

The *rtracklayer* package is an interface (or *layer*) between \mathbf{R} , genome browsers and genomic annotations.

Feature overview

- Annotation track representation and import/export (files and online databases)
- The control and querying of external genome browser sessions and views.
- Currently supports UCSC browser and database.

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ ▲ □ ▶ ④ ▲ ◎

Demonstration: Investigating an aberrant island

Goals:

- 1 Convert coverage to a genomic annotation track
- Visualize the track in its genomic context, focusing on the aberration
- **3** Retrieve annotations in the region for further analysis

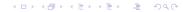
Outline

Introduction

2 Manipulating Genomic Data (Tracks)

3 Interacting with a Genome Browser





Finding the aberrant island

1 Import a lane from Solexa with ShortRead

- 2 Calculate the coverage with IRanges
- **3** Extract islands with height >= 8
- ④ Find an interesting island

Finding the aberrant island

1 Import a lane from Solexa with ShortRead

Code

- > library(ShortRead)
- > setwd("../") # or your path to course data
- > p <- "extdata/ELAND/080828_HWI-EAS88_0003"
- > sp <- SolexaPath(p)</pre>
- > filter <- chromosomeFilter("chr2.fa")</pre>
- > aln <- readAligned(sp, "s_1_1_export_head.txt\$", + filter = filter)
 - 2 Calculate the coverage with *IRanges*
 - **3** Extract islands with height >= 8
 - 4 Find an interesting island

Finding the aberrant island

- Import a lane from Solexa with ShortRead
- 2 Calculate the coverage with IRanges
- **3** Extract islands with height >= 8
- ④ Find an interesting island

Finding the aberrant island



2 Calculate the coverage with *IRanges*

Code

> cov <- coverage(aln, start=1)[[1]]</pre>

- **3** Extract islands with height >= 8
- 4 Find an interesting island

Finding the aberrant island

- Import a lane from Solexa with ShortRead
- 2 Calculate the coverage with IRanges
- **3** Extract islands with height >= 8
- ④ Find an interesting island

Finding the aberrant island

- 1 Import a lane from Solexa with ShortRead
- 2 Calculate the coverage with *IRanges*
- **3** Extract islands with height >= 8

Code

- > islands <- slice(cov, 8)</pre>
 - 4 Find an interesting island

Finding the aberrant island

- 1 Import a lane from Solexa with ShortRead
- 2 Calculate the coverage with IRanges
- **3** Extract islands with height >= 8
- 4 Find an interesting island

Finding the aberrant island

- 1 Import a lane from Solexa with ShortRead
- 2 Calculate the coverage with IRanges
- 3 Extract islands with height >= 8
- 4 Find an interesting island

Code

- > islandSums <- viewSums(islands)</pre>
- > max(islandSums)

```
[1] 129189
```

> island <- islands[which.max(islandSums)]</pre>

・ロト・日本・日本・日本・日本・日本

Storing data on genomic intervals The GenomicData object

- *GenomicData* objects, defined by the *IRanges* package, describe intervals along the genome.
- Two components
 - 1 The interval starts and widths, segregated by chromosome
 - 2 The variables describing the intervals

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● のへぐ

Storing the coverage in a GenomicData object

Construct GenomicData from the coverage

2 Subset for high coverage

Conclusion

Storing the coverage in a GenomicData object

Construct GenomicData from the coverage

| Code |
|--|
| > gd <- as(cov, "GenomicData") > head(start(gd), 3) |
| [1] 1 3018136 3018172 |
| > names(gd) <- "chr2" > head(as.data.frame(gd), 3) ## e.g. for plotting |
| space start end width score |
| 1 chr2 1 3018135 3018135 0 |
| 2 chr2 3018136 3018171 36 2 |
| 3 chr2 3018172 3018622 451 0 |

Subset for high coverage

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● のへぐ

Storing the coverage in a GenomicData object

Construct GenomicData from the coverage

2 Subset for high coverage

Storing the coverage in a GenomicData object

① Construct GenomicData from the coverage

2 Subset for high coverage

Code

> gd <- gd["chr2"] ## by chromosome (no effect) > gd <- gd[gd[["score"]] > 10,] ## subsetting

Outline

Introduction

2 Manipulating Genomic Data (Tracks)

3 Interacting with a Genome Browser

④ Conclusion

The trackSet class in rtracklayer

- The *trackSet* class represents a genome browser track.
- Allows interchange of data between R and genome browsers
- Input/output in GFF, BED or WIG formats

Viewing the coverage in a genome browser

1 Construct a *trackSet* holding the coverage

- 2 Export the coverage as a WIG file
- **3** Upload track to UCSC manually

▲□▶ ▲□▶ ▲□▶ ▲□▶ □ ○ ○ ○

Viewing the coverage in a genome browser

1 Construct a *trackSet* holding the coverage

2 Export the coverage as a WIG file

3 Upload track to UCSC manually

Viewing the coverage in a genome browser

- Construct a trackSet holding the coverage
- 2 Export the coverage as a WIG file
- Opload track to UCSC manually

Viewing the coverage in a genome browser

① Construct a trackSet holding the coverage

2 Export the coverage as a WIG file

Code

- > ## export as WIG
- > export(peaksTrack, "peaks.wig", name = "peaks")

3 Upload track to UCSC manually

Viewing the coverage in a genome browser

- Construct a trackSet holding the coverage
- 2 Export the coverage as a WIG file
- **3** Upload track to UCSC manually

Direct interaction with genome browser from R

1 Upload peak track to UCSC and open a view

- 2 Zoom out for context
- 3 Replot in green with hypothetical cutoff line at 1000
- ④ Download conservation scores around anomaly

Direct interaction with genome browser from R

1 Upload peak track to UCSC and open a view

Code > r <- ranges(gd)[[1]] > island <- islands[which.max(islandSums)]

- > anomaly <- peaksTrack[r %in% island]</pre>
- > session <- browseGenome(anomaly)</pre>

2 Zoom out for context

- 3 Replot in green with hypothetical cutoff line at 1000
- ④ Download conservation scores around anomaly

Direct interaction with genome browser from R

- Upload peak track to UCSC and open a view
- 2 Zoom out for context
- 3 Replot in green with hypothetical cutoff line at 1000
- ④ Download conservation scores around anomaly

Direct interaction with genome browser from R

Upload peak track to UCSC and open a view

2 Zoom out for context

Code > segment <- genomeSegment(session) > segment <- segment / 4 > view <- browserView(session, segment)

3 Replot in green with hypothetical cutoff line at 1000

Download conservation scores around anomaly

Direct interaction with genome browser from R

- Upload peak track to UCSC and open a view
- 2 Zoom out for context
- **3** Replot in green with hypothetical cutoff line at 1000
- ④ Download conservation scores around anomaly

Direct interaction with genome browser from R

- Upload peak track to UCSC and open a view
- 2 Zoom out for context
- **3** Replot in green with hypothetical cutoff line at 1000

Code

```
> session <- layTrack(session, anomaly,
+ "anomalyCutoff",
+ yLineMark = 1000,
+ yLineOnOff = TRUE,
+ color = c(OL, 255L, OL))
> view <- browserView(session, full = "anomalyCutoff",
+ hide = "anomaly")
```

Download conservation scores around anomaly

Direct interaction with genome browser from R

- 1 Upload peak track to UCSC and open a view
- 2 Zoom out for context
- 3 Replot in green with hypothetical cutoff line at 1000
- 4 Download conservation scores around anomaly

Direct interaction with genome browser from R

- ① Upload peak track to UCSC and open a view
- 2 Zoom out for context
- 3 Replot in green with hypothetical cutoff line at 1000
- 4 Download conservation scores around anomaly

Code

```
> segment <- genomeSegment(view) / 100
> track <- trackSet(session, "Conservation",
+ segment,
+ "phastCons30way")
> gd <- GenomicData(IRanges(start(track), end(track)),
+ phastCons = dataVals(track),
+ chrom = chrom(segment))
```

Outline

Introduction

2 Manipulating Genomic Data (Tracks)

3 Interacting with a Genome Browser

4 Conclusion

▲□▶ ▲□▶ ▲目▶ ▲目▶ ▲□ ◇�?

Beyond rtracklayer

- rtracklayer operates in the context of genome browsers
- Bioconductor has other sources of annotations:
 - The annotation packages
 - biomaRt

Session info

```
> sessionInfo()
R version 2.9.0 Under development (unstable) (--)
i686-pc-linux-gnu
locale:
С
attached base packages:
[1] tools
             stats
                       graphics grDevices utils datasets methods
[8] base
other attached packages:
[1] rtracklayer_1.2.2 RCurl_0.91-0
                                   ShortRead 1.1.9
                                                        lattice 0.17-15
[5] Biobase_2.3.0 Biostrings_2.11.0 IRanges_1.0.5
loaded via a namespace (and not attached):
[1] Matrix 0.999375-16 XML 1.98-1
                                         grid 2.9.0
                                                           rJava 0.6-0
```