

Four exercises using *Bioconductor* Sequence Infrastructure

Patrick Aboyoun & Martin Morgan

Fred Hutchinson Cancer Research Center

29-30 July, 2010

Objective

Genomic ranges

Short reads

SAM-based alignments

Coverage vectors

Genomic annotations

Exercises

Outline

Objective

Genomic ranges

Short reads

SAM-based alignments

Coverage vectors

Genomic annotations

Exercises

Objective

The objective of this lab is to learn about the *Bioconductor* sequence infrastructure by walking through exercises on:

Exercises

- ▶ Reading and manipulating short reads
- ▶ An introduction to *Rsamtools*
- ▶ A simple ChIP-seq workflow
- ▶ A simple RNA-seq use case

Outline

Objective

Genomic ranges

Short reads

SAM-based alignments

Coverage vectors

Genomic annotations

Exercises

Important classes

Classes

- ▶ *IRanges's IRanges* - generic interval/range
- ▶ *GenomicRanges's GRanges* - genomic interval/range

Example

```
> library(GenomicRanges)
> grngs <-
+   GRanges(seqnames=c("seq1", "seq2", "seq2"),
+           ranges=IRanges(c(1000, 100, 1000),
+                           c(2000, 1000, 2000)),
+           strand=c("+", "+", "-"))
> grngs
```

GRanges with 3 ranges and 0 elementMetadata values

| | seqnames | ranges | strand | |
|-----|----------|--------------|--------|--|
| | <Rle> | <IRanges> | <Rle> | |
| [1] | seq1 | [1000, 2000] | + | |
| [2] | seq2 | [100, 1000] | + | |
| [3] | seq2 | [1000, 2000] | - | |

Accessors

Methods

- ▶ `seqnames`
- ▶ `ranges`
- ▶ `strand`
- ▶ `start, end, width`
- ▶ `seqlengths`

Example

```
> seqnames(grngs)
```

```
'factor' Rle of length 3 with 2 runs  
  Lengths:    1    2  
  Values  : seq1 seq2  
Levels(2): seq1 seq2
```

Basic interval manipulations

Methods

- ▶ `shift` - moves the intervals to the left or right
- ▶ `resize` - anchors the 5' end of the interval and moves the 3' end

Example

```
> resize(grngs, 1)
```

```
GRanges with 3 ranges and 0 elementMetadata values
```

| | seqnames | ranges | strand | |
|-----|----------|--------------|--------|--|
| | <Rle> | <IRanges> | <Rle> | |
| [1] | seq1 | [1000, 1000] | + | |
| [2] | seq2 | [100, 100] | + | |
| [3] | seq2 | [2000, 2000] | - | |

```
seqlengths
```

| seq1 | seq2 |
|------|------|
| NA | NA |

Interval overlap operations

Methods

- ▶ `findOverlaps` - maps query set to target set
- ▶ `countOverlaps` - counts query set overlaps
- ▶ `subsetByOverlaps` - subsets based on overlaps

Example

```
> findOverlaps(grngs, grngs)
```

```
An object of class "RangesMatching"
```

```
Slot "matchMatrix":
```

| | query | subject |
|------|-------|---------|
| [1,] | 1 | 1 |
| [2,] | 2 | 2 |
| [3,] | 3 | 3 |

```
Slot "DIM":
```

```
[1] 3 3
```

Outline

Objective

Genomic ranges

Short reads

SAM-based alignments

Coverage vectors

Genomic annotations

Exercises

Importing short read data via *ShortRead*

Methods

- ▶ `readAligned` - alignment files
- ▶ `readFastq` - fastq files
- ▶ `readXStringColumns` - collection of *XStringSet* objects

Example

```
> library(ShortRead)
> extdataDir <- system.file("extdata", "slx",
+                           package="HTSandGeneCentricLabs")
> pattern <- "s_1_1_export_head.txt"
> aln <- readAligned(extdataDir, pattern, "SolexaExport")
> aln
```

```
class: AlignedRead
length: 500000 reads; width: 35 cycles
chromosome: 255:255:255 255:255:255 ... QC QC
position: NA NA ... NA NA
strand: NA NA ... NA NA
alignQuality: NumericQuality
alignData varLabels: run lane ... filtering contig
```

Short read related classes

Classes

- ▶ *ShortRead's AlignedRead* - general container for alignments
- ▶ *ShortRead's SolexaExportQA* - a quality assurance processing of Solexa export files
- ▶ *Biostrings's DNASTringSet* - a collection of short reads
- ▶ *Biostrings's BStringSet* - a collection of base call qualities

Accessors

Methods

- ▶ `length` - number of alignments
- ▶ `id` - read identifier
- ▶ `sread` - read base calls
- ▶ `quality` - quality base calls
- ▶ `chromosome` - aligned chromosome
- ▶ `position` - aligned leftmost position
- ▶ `strand` - aligned strand
- ▶ `alignQuality` - overall alignment quality
- ▶ `alignData` - data associated with alignment

Counting letters in strings

Methods

- ▶ `alphabet` - available letters for string
- ▶ `alphabetFrequency` - letter frequency per string
- ▶ `alphabetByCycle/consensusMatrix` - letter frequency per position

Example

```
> alphabetByCycle(sread(aln))[1:4, 1:6]
```

```
      cycle
alphabet  [,1]  [,2]  [,3]  [,4]  [,5]  [,6]
A 145955 147044 128640 140609 135751 133546
C 117538 106357  98356 109948 101961 106672
G 118619 109879 100303 112267 102233 103751
T 115666 118470 106223 112917 107716 105441
```

Filtering alignments

Methods

- ▶ `chromosomeFilter` - filter on chromosome
- ▶ `alignDataFilter` - filter on alignment data (e.g. Illumina's chastity determination)
- ▶ `alignQualityFilter` - filter on overall quality
- ▶ `compose` - functional composition of filters
- ▶ `[` - square-bracket operator

Example

```
> filt1 <- alignDataFilter(expression(filtering=="Y"))
> filt2 <- chromosomeFilter("chr[0-9XYM]+.fa")
> filt <- compose(filt1, filt2)
> caln <- aln[filt(aln)]
> length(caln) / length(aln)

[1] 0.391438
```

Quality assurance of sequencing

Methods

- ▶ `qa` - generate QA measure
- ▶ `report` - create QA report

Example (Not Run)

```
> rpt <- report(qa(extdataDir, "_export.txt"),  
+               dest="reports/my_report.pdf")
```


Outline

Objective

Genomic ranges

Short reads

SAM-based alignments

Coverage vectors

Genomic annotations

Exercises

Importing from BAM files

Read parameter class

`ScanBamParam` - specifies read filter

Minimalist container class

`GappedAlignments` - “shape” (CIGAR) and location of alignments

Methods

- ▶ `scanBam` - creates list of specified columns
- ▶ `readBamGappedAlignments` - create a *GappedAlignments* instance

GappedAlignments class

Example

```
> library(Rsamtools)
> bamFile <-
+   system.file("extdata", "ex1.bam", package="Rsamtools")
> readBamGappedAlignments(bamFile)[1:10]
```

GappedAlignments of length 10

| | rname | strand | cigar | qwidth | start | end | width | ngap |
|------|-------|--------|-------|--------|-------|-----|-------|------|
| [1] | seq1 | + | 36M | 36 | 1 | 36 | 36 | 0 |
| [2] | seq1 | + | 35M | 35 | 3 | 37 | 35 | 0 |
| [3] | seq1 | + | 35M | 35 | 5 | 39 | 35 | 0 |
| [4] | seq1 | + | 36M | 36 | 6 | 41 | 36 | 0 |
| [5] | seq1 | + | 35M | 35 | 9 | 43 | 35 | 0 |
| [6] | seq1 | + | 35M | 35 | 13 | 47 | 35 | 0 |
| [7] | seq1 | + | 36M | 36 | 13 | 48 | 36 | 0 |
| [8] | seq1 | + | 35M | 35 | 15 | 49 | 35 | 0 |
| [9] | seq1 | - | 35M | 35 | 18 | 52 | 35 | 0 |
| [10] | seq1 | + | 35M | 35 | 22 | 56 | 35 | 0 |

scanBam method

Example

```
> which <- GRanges(seqnames=c("seq1", "seq2", "seq2"),  
+                 ranges=IRanges(c(1000, 100, 1000),  
+                               c(2000, 1000, 2000)))  
> what <- c("rname", "strand", "pos", "qwidth", "seq")  
> param <- ScanBamParam(which=which, what=what)  
> bam <- scanBam(bamFile, param=param)  
> class(bam)
```

```
[1] "list"
```

```
> names(bam)
```

```
[1] "seq1:1000-2000" "seq2:100-1000" "seq2:1000-2000"
```

References to BAM files

BAM files reference class

BamViews - defines views of interest

Accessors

- ▶ *bamPaths* - BAM file paths
- ▶ *bamSamples* - sample information
- ▶ *bamRanges* - *GRanges* specifying genomic intervals of interest
- ▶ *bamExperiment* - high-level information for the experiment

BamViews class

Example

```
> fls <- list.files(system.file("extdata", package="Rsamtools"),
+                  "\\\\.bam$", full=TRUE)
> rngs <-
+   GRanges(seqnames = Rle(c("chr1", "chr2"), c(9, 9)),
+           ranges =
+             c(IRanges(seq(10000, 90000, 10000), width=500),
+               IRanges(seq(100000, 900000, 100000), width=5000)),
+           Count = seq_len(18L))
> bv <- BamViews(fl, bamRanges=rngs)
> bv
```

BamViews dim: 18 ranges x 1 samples

names: ex1.bam

detail: use bamPaths(), bamSamples(), bamRanges(), ...

```
> basename(bamPaths(bv))
```

```
[1] "ex1.bam"
```

Outline

Objective

Genomic ranges

Short reads

SAM-based alignments

Coverage vectors

Genomic annotations

Exercises

Coverage vectors class

Class

IRanges's RleList - list of run-length encoded vectors

Constructor

coverage - generates coverage vectors

Example

```
> cover <- coverage(caln)
> cover[1]
```

```
SimpleRleList of length 1
```

```
$chr1.fa
```

```
'integer' Rle of length 197167317 with 27254 runs
```

```
Lengths: 3018534      35    16955 ...      35    59319      35
Values :      0      1      0 ...      1      0      1
```


Coverage vector operations

Methods

- ▶ `runwtsum` - running window weighted sum smoother
- ▶ `pmin` - parallel (elementwise) minimum of two or more vectors
- ▶ `quantile` - quantiles for vectors

Example

```
> quantile(cover, c(0.99, 0.9999, 0.999999))[,1:6]
```

| | chr1.fa | chr10.fa | chr11.fa | chr12.fa | chr13.fa | chr14.fa |
|----------|---------|----------|----------|----------|----------|----------|
| 99% | 0 | 0 | 0 | 0 | 0 | 0 |
| 99.99% | 1 | 1 | 1 | 1 | 1 | 1 |
| 99.9999% | 3 | 5 | 4 | 6 | 3 | 5 |

Views on coverage vectors

Class

IRanges's RleViewsList - contiguous regions within the coverage vectors

Constructor

`slice` - views based on fixed cutoffs (peak thresholds)

Method

`viewMaxs` - maximum for each view

Example

```
> islands <- slice(cover, 1)
> viewMaxs(islands)[1:3]
```

SimpleIntegerList of length 3

```
[["chr1.fa"]] 1 1 1 1 1 1 1 1 1 1 1 1 ... 1 1 1 1 1 1 1 1 1 1
[["chr10.fa"]] 1 1 1 1 1 1 1 1 1 1 1 ... 1 1 1 1 1 1 1 1 1 1
[["chr11.fa"]] 5 1 1 3 1 1 1 4 1 1 2 ... 1 1 1 1 1 1 1 1 1 1
```

Outline

Objective

Genomic ranges

Short reads

SAM-based alignments

Coverage vectors

Genomic annotations

Exercises

Importing/exporting genomic features via *rtracklayer*

Methods

- ▶ `import` - gff, bed, wig, etc.
- ▶ `export` - gff, bed, wig, etc.

Example

```
> library(rtracklayer)
> import(system.file("tests", "bed.wig",
+                   package="rtracklayer"))[1:3,]
```

UCSC track 'Bed Format'

UCSCData with 3 rows and 1 value column across 3 spaces

| | space | ranges | | score |
|---|-------------|----------------------|--|-----------|
| | <character> | <IRanges> | | <numeric> |
| 1 | chr1 | [59302001, 59302300] | | -1.00 |
| 2 | chr1 | [59302301, 59302600] | | -0.75 |
| 3 | chr19 | [59302901, 59303200] | | -0.25 |

Transcript databases via *GenomicFeatures*

Functions

- ▶ *makeTranscriptDbFromBiomart* - create transcript database using BioMart
- ▶ *loadFeatures* - reference serialized database
- ▶ *transcripts* - extract transcript bounds

Example (Not Run)

```
> dmTxDb <-  
+   makeTranscriptDbFromBiomart(biomart = "ensembl",  
+                               dataset =  
+                               "dmelanogaster_gene_ensembl")
```

Outline

Objective

Genomic ranges

Short reads

SAM-based alignments

Coverage vectors

Genomic annotations

Exercises

Exercises

Steps

1. Install HTSandGeneCentricLabs and its suggested packages
2. Locate labs withing *R* using
`browseVignettes("HTSandGeneCentricLabs")`
3. Look over vignettes with a prefix of Examples:
Exercises: A Simple ChIP-Seq Workflow
Exercises: A Simple RNA-seq Use Case
Exercises: An introduction to Rsamtools
Exercises: Reading and Manipulating Short Reads
4. Choose vignettes that are of most interest since they would involve more than two hours of work to complete.