Reads, Sequences, and Alignments

Martin Morgan

2012-07-03 Tue

Contents

1 Introduction

Running example: quality assessment
Pasilla data set

- splicing events regulated by Pasilla
- RNAi / RNASeq

Single- and paired-end reads
Relativley 'early' generation technology

2 Reads: ShortRead

2.1 FASTQ files

```
@SRR031724.1 HWI-EAS299_4_30M2BAAXX:5:1:1513:1024 length=37
GTTTTGTCCAAGTTCTGGTAGCTGAATCCTGGGGCGC
+SRR031724.1 HWI-EAS299_4_30M2BAAXX:5:1:1513:1024 length=37
IIIIIIIIIIIIIIIIIIIIIIIIIIII+HIIII<IE
```

Quality scores

```
POOR... !\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
        PQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{|}~ ....GREAT!
```

2.2 Input

```
## fastq manipulation
library(ShortRead)

## sample data
library(ReadsAlignmentsVariantsLab)
fl <- dir(file.path(bigdata(), "fastq"), full=TRUE)
```

readFastq FastqStreamer, FastqSampler

```
fq <- readFastq(fl)
```

2.3 Accessing & manipulating

sread, quality, id [] subsetting alphabetByCycle

```
## Monday's example
abc <- alphabetByCycle(sread(fq))
abc[1:4, 1:5]
matplot(t(abc[1:4,]), type="l", lwd=2, lty=1)
```

as(sread(fq), "matrix") as(quality(fq), "matrix")

```
m <- as(quality(fq), "matrix")
plot(colMeans(m), type="b")
```

tables

```
tbl <- tables(sread(fq))
head(tbl$top)
tail(tbl$distribution)
head(tbl$distribution)
```

2.4 Further exploration

Low complexity?

```
dust <- dustyScore(sread(fq))
plot(ecdf(dust / width(fq)^2 ))
```

3 Sequences: Biostrings

3.1 Representation

DNAString, DNAStringSet, AAStringSet, BStringSet

3.2 Manipulation

[], [[]] subseting alphabetFrequency dinucleotideFrequency consensusMatrix (like alphabetByCycle) letterFrequency letterFrequencyInSlidingView

```
library(Biostrings)
sread(fq)[1:4]
gc <- letterFrequency(sread(fq), c("G", "C"))
hist(rowSums(gc) / width(sread(fq)))  # GC bias
```

complement, reverseComplement translate

3.3 Pattern matching

matchPDict, countPDict trimLRPatterns
e.g., Adapter contamination
matchPWM pairwiseAlignment

4 Alignments: Rsamtools, GenomicRanges

4.1 SAM / BAM files

Common aligners: Bowtie / Bowtie2; BWA; TopHat; GSNAP Also: Rsubread; matchPDict

```
B7_591:4:96:693:509 73 seq1 1 99 36M * 0 0 CACTAGTGGCTCATTGTAAATGTGTGGTTTAACTCG \
<<<<<<<<<<<<<<;<<<<<<<<<5<<<<<;:<;7 MF:i:18 Aq:i:73 NM:i:0 UQ:i:0 H0:i:1 H1:i:0
```

Name	Value
QNAME	Query (read) NAME
FLAG	Bitwise FLAG, e.g., strand of alignment
RNAME	Reference sequence NAME
POS	1-based leftmost POSition of sequence
MAPQ	MAPping Quality (Phred-scaled)
CIGAR	Extended CIGAR string
RNEXT	Reference name of NEXT segment
PNEXT	1-based POSition of NEXT segment
TLEN	Observed Template LENgth
SEQ	Query SEQuence on the reference strand
QUAL	Query QUALity
OPT	OPTional fields, format TAG:VTYPE:VALUE

'leftmost':

```
                          P
                          +++++----------
'+' strand: 5' ....|....|....|....|....|....|.. 3'
'-' strand: 3' ....|....|....|....|....|....|.. 5'
                  ----------+++++
                          P
```

4.2 BAM Input

BamFile BamFileList

- 'devel': yieldSize

scanBamHeader seqinfo

```
library(Rsamtools)
library(GenomicRanges)
fls <- dir(file.path(bigdata(), "bam"), "bam$", full=TRUE)
fl <- BamFile(fls[[1]])
seqinfo(fl)
```

readGappedAlignments readGappedAlignmentPairs

- encodeOverlaps (devel)

scanBam

```
aln <- readBamGappedAlignments(fl)
```

4.3 Selecting

ScanBamParam

- flag, e.g., positive strand, proper pair, ...
- which – GRanges selection
- what – for scanBam, what fields to read
- tag – what tags to extract

```
flag <- scanBamFlag(isMinusStrand=TRUE)
which <- GRanges("chr3L",
             IRanges(c(1871574, 14769596),
                     c(1876336, 14779523)))
param <- ScanBamParam(flag=flag, which=which)
aln1 <- readBamGappedAlignments(fl, param=param)
```

4.4 Transforming

asBam filterBam sortBam indexBam mergeBam

4.5 Overlaps

findOverlaps summarizeOverlaps # counting!!

```
?summarizeOverlaps
```

4.6 Coverage

```
aln <- readBamGappedAlignments(fl)
cvg <- coverage(aln)
class(cvg)
length(cvg)
names(cvg)
rle <- cvg[["chr3L"]]
depthOfCvg <- sapply(split(runLength(rle), runValue(rle)), sum)
```

5 Summary: quality assessment

5.1 Generate a QA report

```
library(ShortRead)

fls <- dir("/path/to/bamdir", pattern="bam$")
qas <- qa(fls)
rpt <- report(qas)
browseURL(rpt)
```

5.2 Highlights

Typical quality score profile (for runs at the time) Non-random 'random hexamers' Puzzling sawtooth nucleotide frequencies
Single vs. paired-end differences

6 Challenge

Given an R object 'roi', a GRangesList of exons grouped by gene

1. Determine per-gene GC content of reads, using ScanBamParam, readBamGappedAlignments, findOverlaps, letterFrequency.
2. Determine per-gene GC content of reference, using BSgenome.Dmelanogaster.UCSC.dm3, getSeq, letterFrequency.
3. Relationship between read and reference GC content, and read count per gene

7 Resources

Vignettes:

```
vignette("Overview", "ShortRead")
vignette("Rsamtools-Overview", "Rsamtools")
vignette(package="Biostrings")
vignette(package="GenomicRanges")
```