ShortRead and *Rsamtools* for Input and Quality Assessment

Martin Morgan

Fred Hutchinson Cancer Research Center

7-9 June, 2010

Outline

Work flow

Prior to analysis

- Biological preparation, e.g., ChIP.
- 'Sequencing': library preparation, cluster generation, imaging, . . .

Analysis

1. Pre-processing, quality assessment, exploratory analysis
2. Domain-specific analysis
   - ChIP-seq
   - Digital gene expression
   - RNA-seq
   - Microbial / community structure
   - . . .
3. Annotation & integration

Outline

ShortRead data input

```
> library(EatonEtAlChIPseq)

> fl <- system.file("extdata",
+   "GSM424494_wt_G2_orc_chip_rep1_S288C_14.mapview.txt.gz
+   package="EatonEtAlChIPseq")
> aln <- readAligned(fl, type = "MAQMapview")
```

The *AlignedRead* class

```
> aln

class: AlignedRead
length: 478774 reads; width: 39 cycles
chromosome: S288C_14 S288C_14 ... S288C_14 S288C_14
position: 2 4 ... 784295 784295
strand: + - ... + +
alignQuality: IntegerQuality
alignData varLabels: nMismatchBestHit mismatchQuality nExac

> table(strand(aln), useNA="always")

     +      -      *   <NA>
 64170 414604      0      0
```

Accessing reads, base quality, and other data

```
> head(sread(aln), 3)

  A DNAStringSet instance of length 3
    width seq
[1]    39 CGGCTTTCTGACCG...AAAAATGAAAATG
[2]    39 GATTTATGAAAGAA...AAATGAAAATGAA
[3]    39 CTTTCTGACCGAAA...AATGAAAATGAAA

> head(quality(aln), 3)

class: FastqQuality
quality:
  A BStringSet instance of length 3
    width seq
[1]    39 >>>>>>>>>>><>>>...<<<<44444///,
[2]    39 ,%//4&/14&&:<<...>>>>>>>>>>>>>
[3]    39 >>>>>>>>>>>>>>>...<<<<44444////
```

Alphabet by cycle

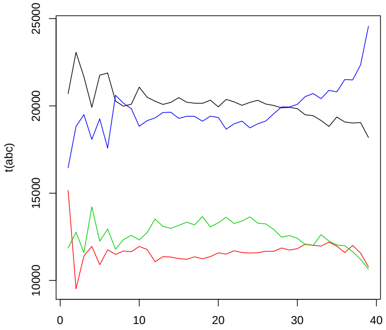
Expectation: nucleotide independent of cycle

```
> alnp <- aln[strand(aln) == "+"]
> abc <- alphabetByCycle(sread(alnp))
> class(abc)

[1] "matrix"

> abc[1:6,1:4]

        cycle
alphabet  [,1]  [,2]  [,3]  [,4]
       A 20701 23067 21668 19920
       C 15159  9523 11402 11952
       G 11856 12762 11599 14220
       T 16454 18818 19501 18078
       M     0     0     0     0
       R     0     0     0     0

> abc <- abc[1:4,]
```

Alphabet by cycle

`matplot` takes a matrix and plots each column as a set of points

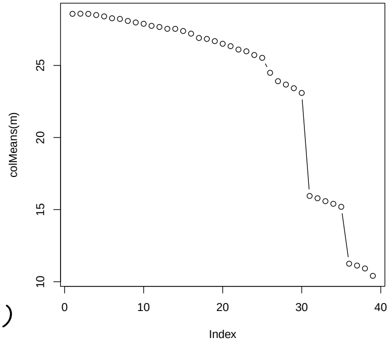
```
> matplot(t(abc), type="l",
+            lty=rep(1, 4))
```



Quality by cycle

Encoded quality scores can be decoded to their numerical values and represented as a matrix. Calculating the average of the column means creates a vector of average quality scores across cycle.

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



Recoding and updating

1. Access the chromosome
2. Extract the chromosome number from the factor level
3. Recode the chromosome number to roman (!), create new levels, and update the chromosome
4. Update the *AlignedRead*

```
> chrom <- chromosome(alnp)
> i <- sub("S288C_([[:digit:]]+)", "\\1", levels(chrom))
> levels(chrom) <- paste("chr", as.roman(i), sep="")
> alnp <- renew(alnp, chromosome=chrom)
```

Quality assessment

Two-step process

1. qa: visit each input file and collate statistics. Long and computationally intensive; can be done in parallel.
2. report: summarize collected statisitcs into an HTML-based report

```
> bowtieDir <- "/path/to/alignments"
> qa <- qa(bowtieDir, ".*map$", type="Bowtie")
> rpt <- report(qa)
> browseURL(rpt)
```

Outline

samtools and *Rsamtools*

samtools

- Data format – text (SAM) and binary (BAM)
- Tools to manipulate (e.g., merge, pileup) and view
- Bindings for other languages, e.g., Picard,

Rsamtools

- Input and represent BAM files.
- High-level: `readAligned`; with `type="BAM"`; `readPileup`
- Flexible: `scanBam`
- Experiment-wide: `BamViews`

Input

ScanBamParam

which	*GRanges* selecting reference, genome coordinates, strand.
flag	select paired / mapped / mate mapped reads
what	fields to retrieve, e.g., query name, reference name, strand, position, width, cigar

Remote access

- E.g., 1000 genomes individual NA19240, chromosome 6, 'Solexa' reads, aligned with MAQ available via ftp

Gapped alignments

Limitations to the *AlignedRead* in *ShortRead*

- Hard to input a subset of reads
- Sequence, quality, identifier information include
- Reads assumed to be *ungapped*

The *GappedAlignments* class in *GenomicRanges*

- `readGappedAlignments` uses `scanBam`
- Genomic coordinates, ‘cigar’, covered intervals
- Cigar: run length encoding; M (match), I, D (insertion, deletion), N (skipped), S, H (soft, hard clip), P (padding). E.g., `35M`, `18M2I15M`
- Accessors, subsetting, narrowing, `pintersect`, `coverage`, ...

BamViews

- Overall experiment represented by ‘regions of interest’ (rows) in several samples (columns).
- Represent this as a ‘view’ on which coordinated operations can be performed.
- Extended examples: *Rsamtools* vignette, *leeBamViews*

Example 1: local access

```
> fl <- system.file("extdata", "ex1.bam",
+                   package="Rsamtools")
> aln <- readAligned(fl, type="BAM")  ## All reads
> ## reads overlapping seq2 nucleotides 500, 1000
> grange <- GRanges("seq2",
+                   IRanges(c(500, 1000), width=1))
> param <- ScanBamParam(which=grange,
+     reverseComplement=TRUE, simpleCigar=TRUE)
> aln <- readAligned(fl, type="BAM", param=param)
```

Example 2: remote access, scanBam

```
> ## na19240url <- ftp://ftp-trace.ncbi.nih.gov/1000ge...
> which <- GRange("6", IRanges(100000L, 110000L))
> param <- ScanBamParam(which=which)
> na19240bam <- scanBam(na19240url, param=param)
```

- Index file downloaded, or locally referenced
- `scanBam` returns a nested list
  - One element for each row of `GRanges`
  - Nested elements correspond to `what`

Examples 3: BamViews

```
> browseVignettes('Rsamtools')
```

Outline

AnnotationDbi packages

- *R* packages with versioned data.
- Pre-built *org.*.db*, *GO.db*, *KEGG.db* and custom-built.

Example: starts / ends of yeast genes, from SGD

```
> library(org.Sc.sgd.db)
> ls('package:org.Sc.sgd.db') # Discovery
> start <- toTable(org.Sc.sgdCHRLOC)
> end <- toTable(org.Sc.sgdCHRLOCEND)
> head(merge(start, end))
```

biomaRt

- Web accessible annotations; from ENSEMBL
- Discovery: `listMarts`, `listDatasets`.
- Use: `useMart`.

```
> library(biomaRt)
> listMarts()
> mart <- useMart("ensembl")
> listDatasets(mart)
> ens <- useMart("ensembl",
+                    dataset="scerevisiae_gene_ensembl")
```

Extracting data with *biomaRt*

- Apply *filters* (`listFilters`) and *attributes* (`listAttributes`)

```
> head(listFilters(ens))
> head(listAttributes(ens))
> ## example query
> getBM(attributes=c("ensembl_gene_id","chromosome_name",
+          "strand","start_position","end_position"),
+        filters="entrezgene",
+        values=c(1466398,1466399,1466400), mart=ens)
```

rtracklayer

Import UCSC Genome Browser data into *R*

- Creates a session: `browserSession`.
- List available genomes from UCSC: `ucscGenomes`.
- Set up a genome object: `genome`.
- List available tracks: `trackNames`.

```
> library(rtracklayer)
> session <- browserSession()
> head(ucscGenomes())
> genome(session) <- "hg18"
> head(trackNames(session))
```

Managing tracks with *rtracklayer*

- Generate a query for UCSC: `ucscTableQuery`.
- Retrieve a UCSC track: `getTable`.

```
> ## generate a query
> query <- ucscTableQuery(session, "refGene")
> ## get the data
> track <- getTable(query)
```

- Also possible to push tracks to UCSC