

# Sequence Analysis : An Introduction

Valerie Obenchain (vobencha@fhcrc.org) and  
Martin Morgan (mtmorgan@fhcrc.org)

Fred Hutchinson Cancer Research Center

9-10 December, 2010

## Work flow

Experiment

Technology

Pre-processing

Analysis

Annotation and Integration

## Data I/O : Short Reads

ShortRead

Input and exploration

Manipulation

Rsamtools and GenomicRanges

## Other Data Examples

## Resources

# Experiments

- ▶ ChIP-seq
- ▶ Differential expression
- ▶ RNA-seq (alternate splicing)
- ▶ Metagenomic
- ▶ ...

# Technology

## Platforms

- ▶ Illumina / Genome Analyzer
- ▶ Roche / 454
- ▶ Applied Biosystems / SOLiD
- ▶ Complete Genomics

# Pre-processing

## Vendor and third-party

- ▶ Image processing, base calling
- ▶ Machine quality assessment
- ▶ Alignment

## Bioconductor

- ▶ Quality assessment and representation: *ShortRead*, *GenomicRanges*
- ▶ Read remediation, trimming, primer removal, specialized manipulation: *IRanges*, *ShortRead*, *Biostrings*
- ▶ Specialized alignment tasks: *Biostrings*, *BSgenome*

# Bioconductor Sequence Analysis Packages

## ChIP-seq

- ▶ *BayesPeak*, *chipseq*, *ChIPseqR*, *ChIPsim*, *PICS*

## RNA-seq and snRNA discovery

- ▶ *Genominator*, *rnaSeqMap*, *segmentSeq*

## Metagenomics

- ▶ *OTUbase*

## Methyl-seq

- ▶ *MEDIPS*, *methVisual*

## Related: genotyping

- ▶ *GGtools*, *VanillaICE* (variant calls from SNP arrays)

# Annotation and Integration

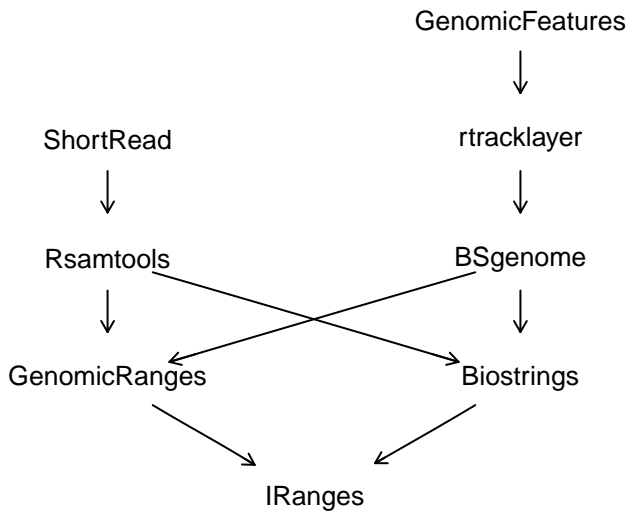
## Annotation

- ▶ Genome coordinate / gene (and other) relationships, *GenomicFeatures*, *ChIPpeakAnno*
- ▶ Resources originally developed for microarray analysis – *AnnotationDbi*, *org.\*.db*, *KEGG.db*, *GO.db*, *Category*, *GOstats*

## Integration

- ▶ Digital and microarray differential expression
- ▶ RNAseq and gene ontology / pathway, *goseq*
- ▶ HapMap, 1000 genomes, UCSC, Sequence Read Archive, GEO, ArrayExpress, *rtracklayer*, *biomaRt*, *Rsamtools*, *GEOquery*, *SRadb*

# Bioconductor Sequence Packages





## ShortRead data input

```
> library(SeattleIntro2010)
> library(ShortRead)
> fl <- system.file("extdata", "SRR002051.chrI-V.bam",
+   package="SeattleIntro2010")
> aln <- readAligned(fl, type = "BAM")
```

## The *AlignedRead* class

```
> aln  
  
class: AlignedRead  
length: 446075 reads; width: 33 cycles  
chromosome: chrI chrI ... chrV chrV  
position: 11 1062 ... 576545 576836  
strand: - + ... - -  
alignQuality: NumericQuality  
alignData varLabels: flag  
  
> table(strand(aln), useNA="always")  
  
      +      -      *    <NA>  
215256 230819      0      0
```

## Accessing reads, base quality, and other data

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

```
A DNAStringSet instance of length 3  
width seq
```

```
[1]    33 TGTGGTGTGTGGTG...GTGGGTGTGTGGG  
[2]    33 TGCATCTTTAATCT...TTACACTACTCAT  
[3]    33 TTAAATAACGTACC...AGTATCGTCTTGA
```

# Alphabet by cycle

Expectation: nucleotide use independent of cycle

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

```
[1] "matrix"
```

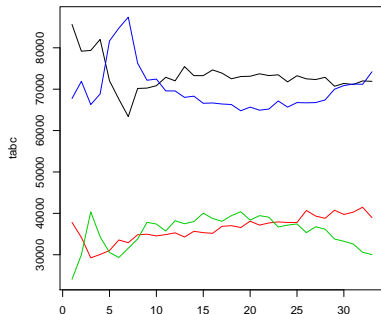
```
> abc[1:6,1:4]
```

	cycle			
alphabet	[,1]	[,2]	[,3]	[,4]
A	85632	79174	79363	82020
C	37815	34182	29250	30064
G	24054	30002	40356	34334
T	67755	71896	66287	68838
M	0	0	0	0
R	0	0	0	0

# Alphabet by cycle

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

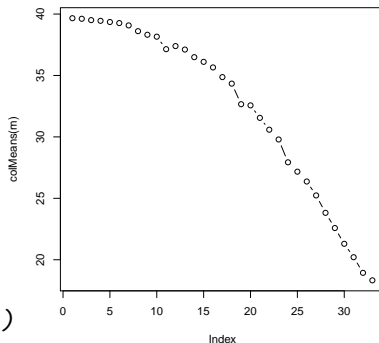
```
> tabc <- t(abc[1:4,])  
> matplot(tabc, type="l",  
+         lty=rep(1, 4))
```



## 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 information
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)
```

## Enhancements to *AlignedRead* and *ShortRead*

- ▶ Easy to input arbitrary subset of reads
- ▶ Not necessary to read sequence, quality, identifier and other information when not necessary
- ▶ Reads can be aligned with indels and gaps



# *samtools* and *Rsamtools*

## *samtools*

- ▶ Data format – text (SAM) and binary (BAM)
- ▶ Tools to manipulate (e.g., merge), analyze (e.g., pileup) and view
- ▶ For developers – bindings to other languages, e.g., Picard

## *Rsamtools*

- ▶ Input and represent BAM files.
- ▶ High-level: `readAligned(..., type="BAM");`  
`readGappedAlignments; readPileup`
- ▶ Flexible: `scanBam`

# 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

# GenomicRanges : Gapped alignments

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, ...

## Example

```
> ## reads on chr III overlapping 100000-110000  
> which <- GRange("chrIII", IRanges(100000, 110000))  
> param <- ScanBamParam(which=which)  
> bf <- scanBam("path/to/bamfile", param=param)
```

scanBam returns a nested list

- ▶ One element for each row of GRanges
- ▶ Nested elements correspond to what

## 454 Microbiome Pre-Processing

```
> library(ShortRead)
> dir <- "/not/public"
> bar <- read454(dir)           # Input
> code <- narrow(sread(bar), 1, 8) # Extract bar code
> aBar <- bar[code == "AAGCGCTT"] # Subset one bar code
> noBar <-                      # Remove bar code
+   narrow(aBar, 11, width(aBar))
> pcrPrimer <- "GGACTACCVGGGTATCTAAT"
> trimmed <-                   # Remove primer
+   trimLRPatterns(pcrPrimer, noBar, Lfixed=FALSE)
> writeFastq(trimmed,          # Output
+   file.path(dir, "trimmed.fastq"))
```

# Digital Gene Expression

```
> library(GenomicFeatures)
> bamFile <- "/path/to/file.bam"
> aligns <- readGappedAlignments(bamFile)
> ## ... txdb: transcripts from UCSC 'knownGenes'
> exonRanges <- exonsBy(txdb, "tx")
> ## ... housekeeping
> counts <- countOverlaps(exonRanges, aligns)
> ## ... normalization --> 'highScores' variable
> txs <- transcripts(txdb,
+                   vals=list(tx_id=names(highScores)),
+                   columns=c("tx_id", "gene_id"))
> systematicNames <- elementMetadata(txs)[["gene_id"]]
```

# Resources

## *Bioconductor* Web site

- ▶ `http://bioconductor.org`
- ▶ 'Installation', 'Software', and 'Mailing lists' links.

## Help in *R*

- ▶ `help.start()` to view a help browser.
- ▶ `help(package = "Biostrings")`
- ▶ `?readAligned`
- ▶ `browseVignettes("GenomicRanges")`