High Throughput Sequence Manipulation and Annotation

Martin Morgan

Fred Hutchinson Cancer Research Center

19-21 January, 2011, edited 31 December, 2010

Contents

1 Introduction **1**

2 Data input and exploration **2**

2.1 Sequences 2

2.2 Gapped alignments 3

3 Annotation: transcripts **4**

4 Counting and measuring **6**

5 Differential representation **7**

6 Annotation: biological function **10**

A Appendix: counting overlaps **12**

1 Introduction

This lab uses data from Nagalakshmi et al. [1]. This is the publication that defined the term 'RNA-seq'; it is primarily methodological. The authors investigated two different approaches to generating DNA from poly(A) RNA, via 'random hexamer' (RH), and 'oligo(dT)' (dT). There were both biological and technical replicates. Sequencing used the Illumina GAI platform, so there are relative few reads (< 5 million per lane) of short (33bp) reads.

2 Data input and exploration

2.1 Sequences

Attach the *ShortRead* and *IWB2011* packages, and locate the file path to the sample data set of the *IWB2011* package.

```
> library(ShortRead)
> library(IWB2011)
> fastqFile <- system.file("extdata", "SRR002051.reads1-50k.fastq",
+                           package="IWB2011")
```

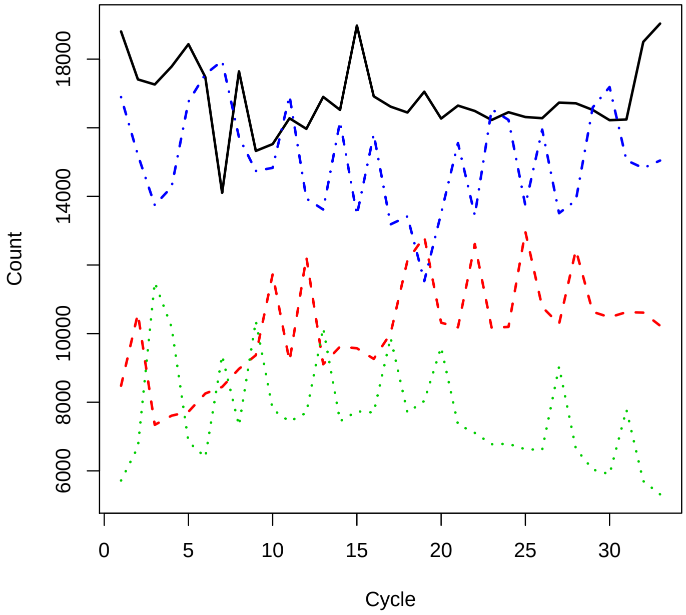
Hint: if `fastqFile` is `""`, then the `system.file` argument is incorrect.

Input the fastq sequences, and explore the resulting object.

```
> fq <- readFastq(fastqFile)
> fq

class: ShortReadQ
length: 50000 reads; width: 33 cycles

> head(sread(fq))

  A DNAStringSet instance of length 6
    width seq
[1]    33 AAAGAACATTAAAGCTATATTATAAGCAAAGAT
[2]    33 AAGTTATGAAATTGTAATTCCAATATCGTAAGC
[3]    33 AATTTCTTACCATATTAGACAAGGCACTATCTT
[4]    33 AGATTTCTAATATGGTTAAGAAGCGAACTTTTT
[5]    33 AAAGCAGCAGCACGTAGTTCTTCATCCTTCTTC
[6]    33 AAGAATTATTAGTCTTTTCTTCTTATTTTTTCA

> head(quality(fq))

class: FastqQuality
quality:
  A BStringSet instance of length 6
    width seq
[1]    33 IIIIIIIIIIIIIIIIIIIIIIIII'II@I$)-
[2]    33 IIIIIIIIIIIIIIIIIIIIIIIIIIIIIII07
[3]    33 IIIIIIIIIIIIIIIIIIIIIIIIIIIIIII&I
[4]    33 IIIIIIIIIIIIIIIIIII?IIIIII<IIIIII
[5]    33 IIIIIIIIIIIIIIIIIIIIIIIFIIIIII%.I
[6]    33 IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII4
```

As a simple illustration of tasks that can be performed, use the `alphabetByCycle`, `t` (matrix transposition), and `matplot` ('matrix plot') functions to visualize how nucleotide use changes with cycle.

```
> abc <- alphabetByCycle(sread(fq))
> matplot(t(abc[1:4,]), type="l", xlab="Cycle", ylab="Count", lwd=2)
```



As a second illustration, coerce quality scores to a matrix using `as`. Then use `apply` and `mean` to summarize the mean base quality per read. Create a subset of 'high quality' reads with mean quality greater than 20, using the `[` operator and a (vectorized) logical test.

```
> m <- as(quality(fq), "matrix")
> baseQualPerRead <- apply(m, 1, mean)
> fq[baseQualPerRead > 20]

class: ShortReadQ
length: 35304 reads; width: 33 cycles
```

2.2 Gapped alignments

Attach the *GenomicRanges* package, determine the path to the sample BAM file, and read in the alignments using the `readGappedAlignments` function. Use `head` to view the first half dozen records.

```
> library(GenomicRanges)
> bamFile <- system.file("extdata", "SRR002051.chrI-V.bam",
+                        package="IWB2011")
```

```
> galn <- readGappedAlignments(bamFile)
> head(galn)

GappedAlignments of length 6
    rname strand cigar qwidth start  end width ngap
[1]  chrI      -   33M     33    11   43    33    0
[2]  chrI      +   33M     33  1062 1094    33    0
[3]  chrI      -   33M     33  1114 1146    33    0
[4]  chrI      +   33M     33  1366 1398    33    0
[5]  chrI      +   33M     33  4336 4368    33    0
[6]  chrI      +   33M     33  4581 4613    33    0
```

Be sure to verify that the file path has been specified correctly!

Use the `cigar` accessor to obtain the CIGAR code of each read. Use the `table`, `sort` and `tail` functions to summarize the most commonly occurring CIGARs.

```
> tail(sort(table(cigar(galn))))

5M1I27M 26M1I6M 3M1I29M 4M1I28M 27M1I5M     33M
    179     197     210     222     322  446075
```

3 Annotation: transcripts

Sequence-based annotations are generally more complicated than the simple gene-level annotations from microarray data. The following script downloads and saves a UCSC genome browser track describing annotated genes in yeast

```
> readScript("create-txdb.R")

[1] pkgroot <- "/home/mtmorgan/IWB2011"
[2] txdbFile <- file.path(pkgroot, "inst", "extdata", "sacCer2_sgdGene.sqlite")
[3]
[4] library(GenomicFeatures)
[5] txdb <- makeTranscriptDbFromUCSC(genome="sacCer2", tablename="sgdGene")
[6] saveFeatures(txdb, txdbFile)
```

Load this file into your current session, and find out more information about it with the following commands:

```
> library(GenomicFeatures)
> txdbFile <- system.file("extdata", "sacCer2_sgdGene.sqlite",
+                         package="IWB2011")
> txdb <- loadFeatures(txdbFile)
> txdb
```

```
TranscriptDb object:
| Db type: TranscriptDb
| Data source: UCSC
| Genome: sacCer2
| UCSC Table: sgdGene
| Type of Gene ID: ID of canonical transcript in cluster
| Full dataset: yes
| transcript_nrow: 6717
| exon_nrow: 7083
| cds_nrow: 7061
| Db created by: GenomicFeatures package from Bioconductor
| Creation time: 2010-12-06 13:16:42 -0800 (Mon, 06 Dec 2010)
| GenomicFeatures version at creation time: 1.2.2
| RSQLite version at creation time: 0.9-2
| DBSCHEMAVERSION: 1.0
```

Query the `txdb` object for transcripts using the `transcripts` function and `columns="gene_id"` argument. Summarize the number of transcripts associated with each gene symbol by: (a) extracting the `gene_id` column using `values` and column selectin; and (b) using `table` twice, first to count the number of times each gene identifier occurs, and then the number of times any gene identifier occurs once, twice, ...

```
> tscript <- transcripts(txdb, columns="gene_id")
> head(tscript)

GRanges with 6 ranges and 1 elementMetadata value
    seqnames       ranges strand |                 gene_id
       <Rle>    <IRanges>  <Rle> | <CompressedCharacterList>
[1]  2micron [ 252, 1523]      + |                  R0010W
[2]  2micron [3271, 3816]      + |                  R0030W
[3]  2micron [1887, 3008]      - |                  R0020C
[4]  2micron [5308, 6198]      - |                  R0040C
[5]     chrI [ 335,  649]      + |                 YAL069W
[6]     chrI [ 538,  792]      + |                 YAL069W

seqlengths
   chrIV   chrXV  chrVII  chrXII ...   chrVI    chrI    chrM 2micron
 1531919 1091289 1090947 1078175 ...  270148  230208   85779    6318

> geneIds <- as(values(tscript)$gene_id, "character")
> table(table(geneIds))

   1    2    3    4    6
6394  149    5    1    1

> ## following line changes how 'gene_id' is stored, internally
> values(tscript)$gene_id <- geneIds
```

For simplicity in subsequent analyses, select the subset of genes with exactly one transcript. Do this using `table`, (names), and logical subsetting to identify candidate genes, and `%in%` to select the corresponding rows of `tscript`. Confirm that the resulting subset is consistent with the tabulation performed in the previous exercise.

```
> x <- table(geneIds)
> ugeneIds <- names(x)[x==1]
> idx <- values(tscript)$gene_id %in% ugeneIds
> tscript1 <- tscript[idx]
> length(tscript1)
```

```
[1] 6394
```

4 Counting and measuring

The goal is to count the number of times a read ‘hits’ a transcript. We need to make decisions about what constitutes a ‘hit’, and what to do with reads that hit more than one transcript. We define a hit as any overlap between the aligned read and a transcript, and we discard reads that align to more than one transcript. The RNAseq protocol used in this experiment does not distinguish between strands, so in preparation for further analysis we indicate that the strand of the transcript is unimportant

```
> strand(tscript1) <- "*"
```

Use the *IRanges* package and `countOverlaps` function to count the number of times each read aligns to a transcript. Subset the reads to include only those that align to exactly one transcript. Use `countOverlaps` again, but with arguments reversed, to count the number of times each transcript overlaps a read.

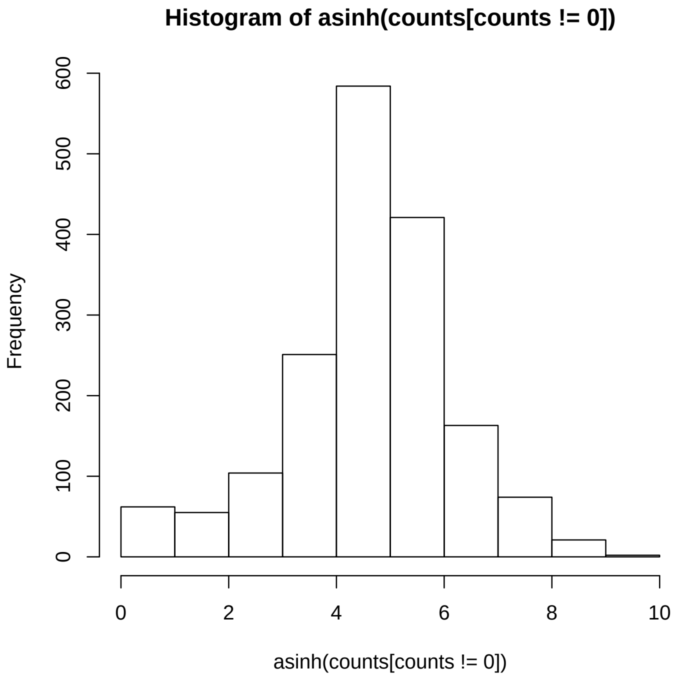
```
> library(IRanges)
> hits <- countOverlaps(galn, tscript1)
> table(hits)
```

```
hits
     0      1      2      3
164635 265107  19780      2
```

```
> galn1 <- galn[hits==1]
> counts <- countOverlaps(tscript1, galn1)
```

Explore the `counts` object to ensure that the results are reasonable. For instance, non-zero counts should only involve chromosomes I-V (because the subset of reads we are using are from these chromosomes). Visualize the non-zero counts as a histogram; use the `asinh` (inverse hyperbolic sine) function for a log-like transformation of the data.

```
> table(seqnames(tscript1[counts!=0]))

 chrIV  chrII   chrV chrIII   chrI
   775    411    292    157    102

> hist(asinh(counts[counts!=0]))
```



5 Differential representation

This section is optional; greater detail on differential expression will be provided tomorrow.

Code presented in the appendix was used to create an object summarizing the entire data set. Load the data set into *R*. The object is a kind of `data.frame`, designed for large data. A feature is that the columns can be annotated with additional information. Access this information with the `elementMetadata` function. Coerce `tscriptCounts` to a standard `data.frame` and summarize the `asinh`-transformed counts across samples using `splom` function from the *lattice* package. Use the `pch` argument to set the plot character to ".". Can you identify, 'by eye', the technical and biological replicates within each protocol?

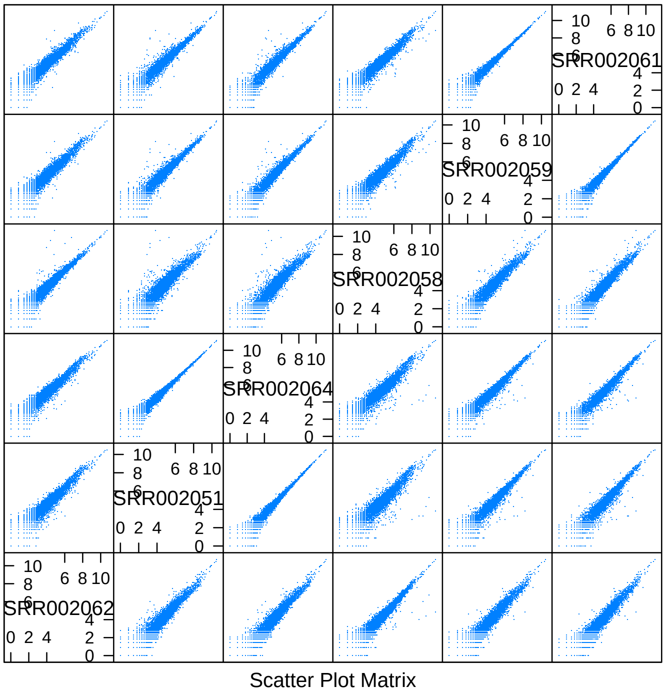
```
> data("tscriptCounts")
> head(tscriptCounts)
```

```
DataFrame with 6 rows and 6 columns
           SRR002062 SRR002051 SRR002064 SRR002058 SRR002059 SRR002061
           <integer> <integer> <integer> <integer> <integer> <integer>
R0010W           397       655      1190       518       785      1219
R0030W           146       291       470       167       273       429
R0020C           264       458       658       415       476       742
R0040C           209       374       677       354       443       692
YAL066W            1         3         1         0         0         0
YAL064W-B          1         1         6         2         2         6
```

```
> elementMetadata(tscriptCounts)
```

```
DataFrame with 6 rows and 3 columns
              Protocol   Replicate         SRR
           <character> <character> <character>
SRR002062           dT  Biological   SRR002062
SRR002051           dT    Original   SRR002051
SRR002064           dT   Technical   SRR002064
SRR002058           RH  Biological   SRR002058
SRR002059           RH    Original   SRR002059
SRR002061           RH   Technical   SRR002061
```

```
> df <- as(tscriptCounts, "data.frame")
> print(splom(asinh(df), pch="."))
```



Create a subset of the data that includes only biological replicates. Do this by creating an index variable based on the 'Replicate' column of the elementMetadata of tscriptCounts, and using logical subsetting. Remove any gene for which no transcripts were found by using rowSums to sum over the number of reads per row, and creating a logical index variable that is TRUE only when the row contains some reads. Use this to subset tscriptCounts1.

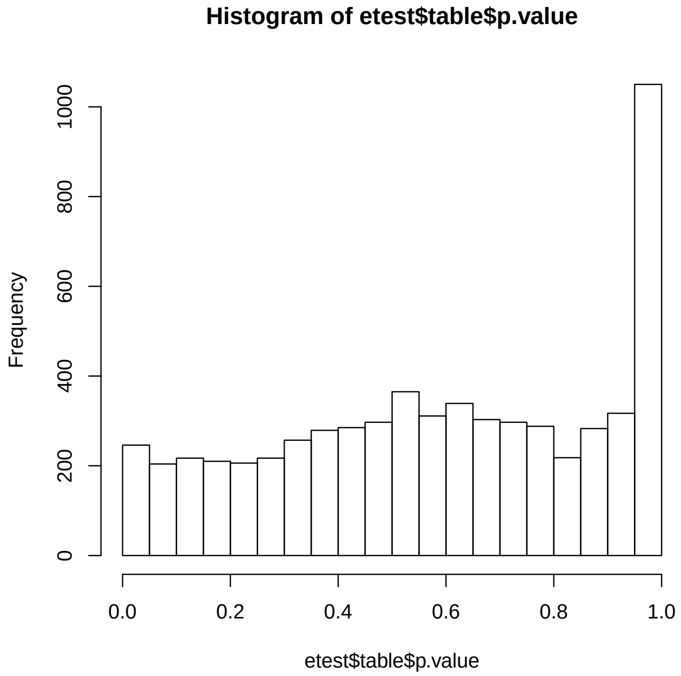
```
> idx <- elementMetadata(tscriptCounts)$Replicate
> tscriptCounts1 <- tscriptCounts[,idx != "Technical"]
> ridx <- rowSums(as(tscriptCounts1, "data.frame")) != 0
> tscriptCounts1 <- tscriptCounts1[ridx,]
```

We use the *edgeR* package to assess differential representation. This package requires counts (created by coercing tscriptCounts1 to a data.frame) and a variable to indicate groups to compare (from the 'Protocol' column of the elementMetadata), presented to the DGEList function. *edgeR* implements methods for normalizing library representations (with the calcNormFactors function), estimating dispersion parameters of the negative binomial model (estimateCommonDisp), performing exact tests of statistical significance (exactTest), and presenting a 'top table' of differentially represented tags (topTags). There are important statistical decisions at each step of the analysis; evaluate the following command as a 'fast track' through this work flow.

```
> library(edgeR)
> df <- as(tscriptCounts1, "data.frame")
> grps <- elementMetadata(tscriptCounts1)$Protocol
> dge <- DGEList(df, group=grps)
> dge <- calcNormFactors(dge)
> dge <- estimateCommonDisp(dge)
> etest <- exactTest(dge)

Comparison of groups:  dT - RH

> ttags <- topTags(etest, n=10, adjust.method="BH")
```

Explore these results. For instance, use hist to visualize the distribution of (unadjusted) P values available in the etest object.

```
> hist(etest$table$p.value, 21)
```



6 Annotation: biological function

We can find out about the biological features identified as differentially represented. For instance, use `rownames` to extract the most differentially represented transcripts in `ttags`. Use these to query `tscript` for the genomic locations of these transcripts.

```
> geneIds <- rownames(ttags$table)
> idx <- values(tscript)$gene_id %in% geneIds
> (topScripts <- tscript[idx])
```

```
GRanges with 10 ranges and 1 elementMetadata value
     seqnames           ranges strand |     gene_id
        <Rle>        <IRanges>  <Rle> | <character>
 [1]    chrXI [258717, 258866]      - |   YKL096C-B
 [2]   chrXII [452439, 452624]      + |   YLR154W-A
 [3]   chrXII [454393, 454557]      + |   YLR154W-B
 [4]   chrXII [454697, 455071]      + |   YLR154W-C
 [5]   chrXII [455434, 455637]      + |   YLR154W-E
 [6]   chrXII [455884, 456024]      + |   YLR154W-F
 [7]   chrXII [490407, 490595]      + |   YLR162W-A
 [8]   chrXII [462523, 462672]      - |   YLR154C-G
 [9]   chrXII [468828, 468959]      - |   YLR154C-H
[10]   chrXII [485712, 485843]      - |   YLR159C-A

seqlengths
   chrIV   chrXV  chrVII  chrXII ...   chrVI    chrI    chrM 2micron
 1531919 1091289 1090947 1078175 ...  270148  230208   85779    6318
```

The *BSgenome.** packages provide DNA sequences of model organisms. Attach the *BSgenome.Scerevisiae.UCSC.sacCer2* package and display a summary of its content using the organism part of the package name. Use `getSeq` to retrieve the sequences corresponding to these regions, using the `as.character=FALSE` argument to return the results in a *DNAStringSet* object. Use the `alphabetFrequency` function with the `baseOnly=TRUE` argument from *Biostrings* to summary nucleotide use. What is the GC content of the top transcripts?

```
> library(BSgenome.Scerevisiae.UCSC.sacCer2)
> Scerevisiae

Yeast genome
|
| organism: Saccharomyces cerevisiae (Yeast)
| provider: UCSC
| provider version: sacCer2
| release date: June 2008
| release name: SGD June 2008 sequence
|
| sequences (see '?seqnames'):
|   chrI     chrII    chrIII   chrIV    chrV     chrVI    chrVII
|   chrVIII  chrIX    chrX     chrXI    chrXII   chrXIII  chrXIV
|   chrXV    chrXVI   chrM     2micron
|
| (use the '$' or '[[' operator to access a given sequence)

> (topSeqs <- getSeq(Scerevisiae, topScripts, as.character=FALSE))

  A DNAStringSet instance of length 10
     width seq
 [1]   150 ATGAAGTTATTTATTCTTGATTATGAGA...TTCAATTTTTTCGTTGTTTTTTTTTTAG
 [2]   186 ATGGTTTGTATTCACACTGAAAATCAAA...CCCGGATCAGCCCCGAATGGGACCTTGA
 [3]   165 ATGCTCTTACTCAAATCCATCCGAAGAC...TCCGTGTTTCAAGACGGGCGGCATATAA
 [4]   375 ATGCGAGATTCCCCTACCCACAAGGAGC...CACCGAAGGTACCAGATTTCAAATTTGA
 [5]   204 ATGCCCCCTGGAATACCAAGGGGCGCAA...TTTTGACAAAAAATTTAATGAATAGATAA
 [6]   141 ATGCTCTTGCCAAAACAAAAAAATCCAT...CCAAGTTTGTCCAAATTCTCCGCTCTGA
 [7]   189 ATGGTTTGTATTCACACTGAAAATCAAA...CATAAACTGTATTATGAACAGAAGGTAG
 [8]   150 ATGTGGAGACGTCGGCGCGAGCCCTGGG...GCTCCGGTGCGCTTGTGACGGCCCGTGA
 [9]   132 ATGTATTCGTGCGCGAAAAAAAAAAACAA...TCGGACGGGAAACGGTGCTTTCTGGTAG
[10]   132 ATGTATTCGTGCGCGAAAAAAAAAAACAA...TCGGACGGGAAACGGTGCTTTCTGGTAG
```

```
> (abc <- alphabetFrequency(topSeqs, baseOnly=TRUE))

        A   C  G   T other
 [1,]  55  10 29  56     0
 [2,]  44  54 36  52     0
 [3,]  40  47 34  44     0
 [4,] 102 114 58 101     0
 [5,]  63  37 38  66     0
 [6,]  43  34 16  48     0
 [7,]  50  41 35  63     0
 [8,]  25  36 49  40     0
 [9,]  36  33 31  32     0
[10,]  36  33 31  32     0
```

```
> rowSums(abc[,c("G", "C")]) / rowSums(abc)

 [1] 0.2600000 0.4838710 0.4909091 0.4586667 0.3676471 0.3546099
 [7] 0.4021164 0.5666667 0.4848485 0.4848485
```

The *org.** packages represent a snapshot of organism-centric annotation information collated before each *Bioconductor* release. Attach the *Saccharomyces cerivisiea* annotation package created from SGD annotations. Use `mget` to look up the GENENAME corresponding to gene ids of differentially represented transcripts; only a couple of these transcripts have names, use `Filter` and `Negate` to select those with names

```
> library(org.Sc.sgd.db)
> (namedIds <- Filter(Negate(is.na), mget(geneIds, org.Sc.sgdGENENAME)))

$`YLR154W-C`
[1] "TAR1"

$`YLR162W-A`
[1] "RRT15"
```

There are many other annotation resources available in *Bioconductor*. Explore, on your own, the *biomaRt* and *rtracklayer* packages.

References

[1] U. Nagalakshmi, Z. Wang, K. Waern, C. Shou, D. Raha, M. Gerstein, and M. Snyder. The transcriptional landscape of the yeast genome defined by RNA sequencing. *Science*, 320:1344–1349, Jun 2008.

A Appendix: counting overlaps

This script counts reads overlapping transcripts across six samples.

```
> readScript("create-tscriptCounts.R")

 [1] library(GenomicFeatures)
 [2] library(multicore)
 [3]
 [4] pkgroot <- "/home/mtmorgan/IWB2011"
 [5] datasrc <- file.path(pkgroot, "NagalakshmiEtAl", "aln")
 [6] txdbFile <- file.path(pkgroot, "inst", "extdata", "sacCer2_sgdGene.sqlite")
 [7] countsFile <- file.path(pkgroot, "data", "tscriptCounts.rda")
 [8]
 [9] ## transcript coordinates of gene symbols with exactly one transcript
[10] txdb <- loadFeatures(txdbFile)
[11] tscript <- transcripts(txdb, column="gene_id")
[12] geneIds <- as(values(tscript)[["gene_id"]], "character")
[13] values(tscript)[["gene_id"]] <- geneIds
[14] x <- table(geneIds)
[15] ugeneIds <- names(x)[x==1]
[16] tscript1 <- tscript[geneIds %in% ugeneIds]
[17] strand(tscript1) <- "*"              # protocol doesn't distinguish strand
[18]
[19] ## reads and counts
[20] fls <- list.files(datasrc, pattern="fastq.sorted.bam$", full=TRUE)
[21] counts <- mclapply(fls, function(fl, ts) {
[22]     print(fl)
[23]     ga <- readGappedAlignments(fl)
[24]     hits <- countOverlaps(ga, ts)
[25]     countOverlaps(ts, ga[hits==1])
[26] }, tscript1)
[27] tscriptCounts <- as(counts, "DataFrame")
[28] dimnames(tscriptCounts) <-
[29]     list(as(values(tscript1)[["gene_id"]], "character"),
[30]          sub(".fastq.*", "", basename(fls)))
[31]
[32] ## sample annotations
[33] df <- DataFrame(Protocol=rep(c("RH", "dT"), each=3),
[34]                 Replicate=rep(c("Biological", "Original", "Technical"), 2),
[35]                 SRR=c("SRR002058", "SRR002059", "SRR002061",
[36]                   "SRR002062", "SRR002051", "SRR002064"))
[37] elementMetadata(tscriptCounts) <-
[38]     df[match(colnames(tscriptCounts), df$SRR),]
[39] o <- with(elementMetadata(tscriptCounts),
[40]           order(Protocol, Replicate))
[41] tscriptCounts <- tscriptCounts[ridx,o]
[42]
[43] save(tscriptCounts, file=countsFile)
```