# Working with aligned nucleotides (WORK-IN-PROGRESS!)

#### Hervé Pagès

Last modified: January 2014; Compiled: March 27, 2018

## Contents

1		1
2	Load the aligned reads and their sequences from a BAM file.	2
3	Compute the original query sequences	4
4	Mismatches, indels, and gaps.	4
5	Put the read sequences and reference sequences "side by side"	5
6	OLD STUFF (needs to be recycled/updated)6.1Load paired-end reads from a BAM file	5 5
7	<pre>sessionInfo()</pre>	8

# 1 Introduction

This vignette belongs to the *GenomicAlignments* package. It illustrates how to use the package for working with the nucleotide content of aligned reads.

After the reads generated by a high-throughput sequencing experiment have been aligned to a reference genome, the questions that are being asked about these alignments typically fall in two broad categories: **positional only** and **nucleotide-related**.

**Positional only** questions are about the position of the alignments with respect to the reference genome. Note that the position of an alignment is actually better described in terms of genomic ranges (1 range for an alignment with no gaps, 2 or more ranges for an alignment with gaps). Knowing the ranges of the alignments is sufficient to perform common tasks like *read counting* or for *computing the coverage. Read counting* is the process of counting the number of aligned reads per gene or exon and is typically performed in the context of a differential analysis. This task can be accomplished with the summarizeOverlaps function provided in the *GenomicAlignments* package and is explained in details in the "Counting reads with summarizeOverlaps" vignette (also located in this package). *Computing the coverage* 

is often the preliminary step to peak detection (ChIP-seq analysis) or to a copy number analysis. It can be accomplished with the coverage function. See ?`coverage-methods` for more information.

**Nucleotide-related** questions are about the nucleotide content of the alignments. In particular how this content compares to the corresponding nucleotides in the reference genome. These questions typically arise in the context of small genetic variation detection between one or more samples and a reference genome.

The *GenomicAlignments* package provides a suite of low- to mid-level tools for dealing with **nucleotide-related** questions about the alignments. In this vignette we illustrate their use on the single-end and paired-end reads of an RNA-seq experiment.

Note that these tools do NOT constitute a complete variant toolbox. If this is what you're looking for, other *Bioconductor* packages might be more appropriate. See the GeneticVariability and SNP views at this URL http://bioconductor.org/packages/release/BiocViews.html# \_\_\_\_\_AssayDomains for a complete list of packages that deal with small genetic variations. Most of them provide tools of higher level than the tools described in this vignette. See for example the *VariantTools* and *VariantAnnotation* packages for a complete variant toolbox (including variant calling capabilities).

# 2 Load the aligned reads and their sequences from a BAM file

In this section, we illustrate how aligned reads and their sequences can be loaded from a BAM file. The reads we're going to use for this are paired-end reads from a published study by Zarnack et al., 2012. A subset of these reads are stored in the BAM files located in the *RNAseqData.HNRNPC.bam.chr14* data package. The package contains 8 BAM files, 1 per sequencing run:

```
> library(RNAseqData.HNRNPC.bam.chr14)
> bamfiles <- RNAseqData.HNRNPC.bam.chr14_BAMFILES
> names(bamfiles) # the names of the runs
[1] "ERR127306" "ERR127307" "ERR127308" "ERR127309" "ERR127302" "ERR127303"
```

[7] "ERR127304" "ERR127305"

Each BAM file was obtained by (1) aligning the reads (paired-end) to the full hg19 genome with TopHat2, and then (2) subsetting to keep only alignments on chr14. See ?RNAseq Data.HNRNPC.bam.chr14 for more information about this data set.

As a preliminary step, we check whether the BAM files contain single- or paired-end alignments. This can be done with the quickBamFlagSummary utility from the *Rsamtools* package:

> library(Rsamtools)
> quickBamFlagSummary(bamfiles[1], main.groups.only=TRUE)

group | nb of | nb of | mean / max of | records | unique | records per records | in group | QNAMEs | unique QNAME All records..... A | 800484 | 393300 | 2.04 / 10 o template has single segment.... S | 0 | 0 | NA / NA

o template has multiple segments.	Μ	800484	393300   2.04 / 10	
- first segment	F	400242	393300   1.02 / 5	
- last segment	L	400242	393300   1.02 / 5	
- other segment	0	0	0   NA / NA	

Note that (S, M) is a partitioning of A, and (F, L, O) is a partitioning of M. Indentation reflects this.

This confirms that all the alignments in the 1st BAM file (run ERR127306) are paired-end. This means that we should preferably load them with the readGAlignmentPairs function from the *GenomicAlignments* package. However for the purpose of keeping things simple, we will ignore the pairing for now and load only the alignments corresponding to the first segment of the pairs. We will use the readGAlignments function from the *GenomicAlignments* package for this, together with a *ScanBamParam* object for the filtering. See ?ScanBamParam in the *Rsamtools* package for the details.

Furthermore, while preparing the *ScanBamParam* object to perform the filtering, we'll also get rid of the PCR or optical duplicates (flag bit 0x400 in the SAM format, see the SAM Spec <sup>1</sup> for the details), as well as reads not passing quality controls (flag bit 0x200 in the SAM format).

<sup>1</sup>http://samtools. sourceforge.net/

Finally we also request the read sequences (a.k.a. the *segment sequences* in the SAM Spec, stored in the SEQ field) via the *ScanBamParam* object:

```
> flag1 <- scanBamFlag(isFirstMateRead=TRUE, isSecondMateRead=FALSE,
+ isDuplicate=FALSE, isNotPassingQualityControls=FALSE)
> param1 <- ScanBamParam(flag=flag1, what="seq")</pre>
```

We're now ready to load the alignments and their sequences. Note that we use <u>use.names=TRUE</u> in order to also load the *query names* (a.k.a. the *query template names* in the SAM Spec, stored in the QNAME field) from the BAM file. <u>readGAlignments</u> will use them to set the names of the returned object:

```
> library(GenomicAlignments)
> gal1 <- readGAlignments(bamfiles[1], use.names=TRUE, param=param1)</pre>
```

This returns a *GAlignments* object. The read sequences are stored in the seq metadata column of the object:

```
> mcols(gal1)$seq
```

A DNAStringSet instance of length 400242 width seq 72 TGAGAATGATGATTTCCAATTTCATCCATGT...GACATGAACTCATCATTTTTTATGGCTGCAT [1] [2] 72 CCCCAGGTATACACTGGACTCCAGGTGGACA...GTTGGATACACACACTCAAGGTGGACACCAG [3] 72 CATAGATGCAAGAATCCTCAATCAAATACTA...ΤΤCAACAGCACATTAAAAAGATAACTTACCA 72 TGCTGGTGCAGGATTTATTCTACTAAGCAAT...ATCAAATCCACTTTCTTATCTCAGGAATCAG [4] 72 CAGGAGGTAGGCTGTGCGTTCAGCAGTTGGT...GGTACTGGTTGATCACCTTGACTGTCTGGTC [5] . . . . . . . . . . [400238] 72 GGGAGGCCCTTTATATAACCATCAGGTCTTG...CTCACTAATAGGATAAAAGCATGGAGAGAAC 72 CCTGAGAGCCCCTTGCTGTCCTGAGCACCTC...GAGCGCCCTCTGGTGTTCTGATCACTCTCTG [400239] 72 CAACTTTTATTTCTTAAACACAAGACATTCC...CTGTTCTCAGGTGAGCTGTCGAGCAGGGAGG [400240] [400241] 72 CAAAGCTGGATGTGTCTAGTGTTTTTATCAG...CCGTAATAAGAGCATGTGTGGTTTTGCTGCC [400242] 72 CATGACTTGATGGCTGGAACAAATACATTTA...CTCCAATACTAGCCTTTGCCATACAGTATTT

### 3 Compute the *original query sequences*

Because the BAM format imposes that the read sequence is "reverse complemented" when a read aligns to the minus strand, we need to "reverse complement" it again to restore the *original query sequences* (i.e. the sequences before alignment, that is, as they can be seen in the FASTQ file assuming that the aligner didn't perform any hard-clipping on them):

```
> oqseq1 <- mcols(gal1)$seq
> is_on_minus <- as.logical(strand(gal1) == "-")
> oqseq1[is_on_minus] <- reverseComplement(oqseq1[is_on_minus])</pre>
```

Because the aligner used to align the reads can report more than 1 alignment per read (i.e. per sequence stored in the FASTQ file), we shouldn't expect the names of gal1 to be unique:

```
> is_dup <- duplicated(names(gal1))
> table(is_dup)
is_dup
FALSE TRUE
393300 6942
```

However, sequences with the same *query name* should correspond to the same *original query* and therefore should be the same. Let's do a quick sanity check:

```
> dup2unq <- match(names(gal1), names(gal1))
> stopifnot(all(oqseq1 == oqseq1[dup2unq]))
```

Finally, let's reduce oqseq1 to one *original query sequence* per unique *query name* (like in the FASTQ file containing the 1st end of the unaligned reads):

```
> oqseq1 <- oqseq1[!is_dup]</pre>
```

#### 4 Mismatches, indels, and gaps

Because the aligner possibly tolerated a small number of mismatches, indels, and/or gaps during the alignment process, the sequences in mcols(gal1)\$seq gnerally don't match exactly the reference genome.

The information of where indels and/or gaps occur in the alignments is represented in the CIGAR strings. Let's have a look at these string. First the most frequent cigars:

> head(sort(table(cigar(gal1)), decreasing=TRUE))

 72M
 35M123N37M
 64M316N8M
 38M670N34M
 18M123N54M
 2M131N70M

 301920
 134
 133
 119
 96

Then a summary of the total number of insertions (I), deletions (D), and gaps (N):

<pre>&gt; colSums(cigar0pTable(cigar(gal1)))</pre>								
М	I	D	Ν	S	Н	Р	=	
28815928	1496	818 3309	945002	Θ	Θ	Θ	Θ	
Х								
Θ								

This tells us that the aligner that was used supports indels (I/D) and junction reads (N). Finally we count and summarize the number of gaps per alignment:

Some reads contain up to 3 gaps (i.e. span 3 splice junctions).

# 5 Put the read sequences and reference sequences "side by side"

TODO (with sequenceLayer)

# 6 OLD STUFF (needs to be recycled/updated)

#### 6.1 Load paired-end reads from a BAM file

BAM file untreated3\_chr4.bam (located in the *pasillaBamSubset* data package) contains paired-end reads from the "Pasilla" experiment and aligned against the dm3 genome (see ?untreated3\_chr4 in the *pasillaBamSubset* package for more information about those reads). We use the readGAlignmentPairs function to load them into a *GAlignmentPairs* object:

```
> library(pasillaBamSubset)
```

```
> flag0 <- scanBamFlag(isDuplicate=FALSE, isNotPassingQualityControls=FALSE)</pre>
```

```
> param0 <- ScanBamParam(flag=flag0)</pre>
```

> U3.galp <- readGAlignmentPairs(untreated3\_chr4(), use.names=TRUE, param=param0)
> head(U3.galp)

GAlignmentPairs object with 6 pairs, strandMode=1, and 0 metadata columns: seqnames strand : ranges -ranges <Rle> <Rle> : <IRanges> --<IRanges> SRR031715.1138209 chr4 + : [169, 205] -- [ 326, 362] SRR031714.756385 chr4 + : [943, 979] -- [1086, 1122] + : [944, 980] -- [1119, 1155] SRR031714.2355189 chr4

seqinfo: 8 sequences from an unspecified genome

The show method for *GAlignmentPairs* objects displays two ranges columns, one for the *first* alignment in the pair (the left column), and one for the *last* alignment in the pair (the right column). The strand column corresponds to the strand of the *first* alignment.

> head(first(U3.galp))

- - - - - - -

GAlignments object w	with 6 alig	gnments and	0 metad	lata columr	ıs:	
	seqnames s	strand	cigar	qwidth	start	end
	<rle></rle>	<rle> <char< td=""><td>~acter&gt;</td><td><integer></integer></td><td><integer></integer></td><td><integer></integer></td></char<></rle>	~acter>	<integer></integer>	<integer></integer>	<integer></integer>
SRR031715.1138209	chr4	+	37M	37	169	205
SRR031714.756385	chr4	+	37M	37	943	979
SRR031714.2355189	chr4	+	37M	37	944	980
SRR031714.5054563	chr4	+	37M	37	946	982
SRR031715.1722593	chr4	+	37M	37	966	1002
SRR031715.2202469	chr4	+	37M	37	966	1002
	width	njunc				
	<integer></integer>	<integer></integer>				
SRR031715.1138209	37	Θ				
SRR031714.756385	37	Θ				
SRR031714.2355189	37	Θ				
SRR031714.5054563	37	Θ				
SRR031715.1722593	37	Θ				
SRR031715.2202469	37	Θ				
seqinfo: 8 sequen	ces from an	n unspecifie	ed genom	ne		
<pre>&gt; head(last(U3.galp)</pre>	))					
	· /					
GAlignments object v		gnments and	0 metac	lata columr	ıs:	
			0 metac cigar	lata columr qwidth	ns: start	end
	with 6 alig seqnames s		cigar	qwidth	start	
	with 6 alig seqnames s	strand	cigar	qwidth	start	
GAlignments object w	with 6 alig seqnames s <rle></rle>	strand <rle> <char< td=""><td>cigar racter&gt;</td><td>qwidth <integer></integer></td><td>start <integer></integer></td><td><integer></integer></td></char<></rle>	cigar racter>	qwidth <integer></integer>	start <integer></integer>	<integer></integer>
GAlignments object of SRR031715.1138209	with 6 alig seqnames s <rle> chr4</rle>	strand <rle> <char< td=""><td>cigar racter&gt; 37M</td><td>qwidth <integer> 37</integer></td><td>start <integer> 326</integer></td><td><integer> 362</integer></td></char<></rle>	cigar racter> 37M	qwidth <integer> 37</integer>	start <integer> 326</integer>	<integer> 362</integer>
GAlignments object of SRR031715.1138209 SRR031714.756385	with 6 alig seqnames <rle> chr4 chr4</rle>	strand <rle> <char - -</char </rle>	cigar racter> 37M 37M	qwidth <integer> 37 37</integer>	start <integer> 326 1086</integer>	<integer> 362 1122</integer>
GAlignments object of SRR031715.1138209 SRR031714.756385 SRR031714.2355189	with 6 alig seqnames 9 <rle> chr4 chr4 chr4</rle>	strand <rle> <char - - -</char </rle>	cigar racter> 37M 37M 37M	qwidth <integer> 37 37 37</integer>	start <integer> 326 1086 1119</integer>	<integer> 362 1122 1155</integer>
GAlignments object of SRR031715.1138209 SRR031714.756385 SRR031714.2355189 SRR031714.5054563	with 6 alig seqnames 9 <rle> chr4 chr4 chr4 chr4 chr4</rle>	strand <rle> <char - - - - -</char </rle>	cigar racter> 37M 37M 37M 37M	qwidth <integer> 37 37 37 37</integer>	start <integer> 326 1086 1119 986</integer>	<integer> 362 1122 1155 1022</integer>
GAlignments object of SRR031715.1138209 SRR031714.756385 SRR031714.2355189 SRR031714.5054563 SRR031715.1722593	vith 6 alig seqnames s <rle> chr4 chr4 chr4 chr4 chr4 chr4 chr4</rle>	strand <rle> <char - - - - -</char </rle>	cigar Facter> 37M 37M 37M 37M 37M	qwidth <integer> 37 37 37 37 37</integer>	start <integer> 326 1086 1119 986 1108</integer>	<integer> 362 1122 1155 1022 1144</integer>
GAlignments object of SRR031715.1138209 SRR031714.756385 SRR031714.2355189 SRR031714.5054563 SRR031715.1722593	vith 6 alig seqnames s <rle> chr4 chr4 chr4 chr4 chr4 chr4 chr4 chr4</rle>	strand <rle> <char - - - - - - -</char </rle>	cigar Facter> 37M 37M 37M 37M 37M	qwidth <integer> 37 37 37 37 37</integer>	start <integer> 326 1086 1119 986 1108</integer>	<integer> 362 1122 1155 1022 1144</integer>
GAlignments object of SRR031715.1138209 SRR031714.756385 SRR031714.2355189 SRR031714.5054563 SRR031715.1722593	vith 6 alig seqnames s <rle> chr4 chr4 chr4 chr4 chr4 chr4 chr4 chr4</rle>	strand <rle> <char - - - - - - njunc</char </rle>	cigar Facter> 37M 37M 37M 37M 37M	qwidth <integer> 37 37 37 37 37</integer>	start <integer> 326 1086 1119 986 1108</integer>	<integer> 362 1122 1155 1022 1144</integer>
GAlignments object of SRR031715.1138209 SRR031714.756385 SRR031714.2355189 SRR031714.5054563 SRR031715.1722593 SRR031715.2202469	with 6 alig seqnames 9 <rle> chr4 chr4 chr4 chr4 chr4 chr4 chr4 width <integer></integer></rle>	strand <rle> <char - - - - - njunc <integer></integer></char </rle>	cigar Facter> 37M 37M 37M 37M 37M	qwidth <integer> 37 37 37 37 37</integer>	start <integer> 326 1086 1119 986 1108</integer>	<integer> 362 1122 1155 1022 1144</integer>
GAlignments object of SRR031715.1138209 SRR031714.756385 SRR031714.2355189 SRR031714.5054563 SRR031715.1722593 SRR031715.2202469 SRR031715.1138209	with 6 alig seqnames 9 <rle> chr4 chr4 chr4 chr4 chr4 chr4 chr4 width <integer> 37</integer></rle>	strand <rle> <char - - - - njunc <integer> 0</integer></char </rle>	cigar Facter> 37M 37M 37M 37M 37M	qwidth <integer> 37 37 37 37 37</integer>	start <integer> 326 1086 1119 986 1108</integer>	<integer> 362 1122 1155 1022 1144</integer>
GAlignments object of SRR031715.1138209 SRR031714.756385 SRR031714.2355189 SRR031714.5054563 SRR031715.1722593 SRR031715.2202469 SRR031715.1138209 SRR031714.756385	with 6 alig seqnames 9 <rle> chr4 chr4 chr4 chr4 chr4 chr4 chr4 width <integer> 37 37</integer></rle>	strand <rle> <char - - - - njunc <integer> 0 0</integer></char </rle>	cigar Facter> 37M 37M 37M 37M 37M	qwidth <integer> 37 37 37 37 37</integer>	start <integer> 326 1086 1119 986 1108</integer>	<integer> 362 1122 1155 1022 1144</integer>
GAlignments object of SRR031715.1138209 SRR031714.756385 SRR031714.2355189 SRR031714.5054563 SRR031715.1722593 SRR031715.2202469 SRR031715.1138209 SRR031714.756385 SRR031714.2355189	with 6 alig seqnames 9 <rle> chr4 chr4 chr4 chr4 chr4 chr4 width <integer> 37 37 37</integer></rle>	strand <rle> <char - - - - njunc <integer> 0 0 0</integer></char </rle>	cigar Facter> 37M 37M 37M 37M 37M	qwidth <integer> 37 37 37 37 37</integer>	start <integer> 326 1086 1119 986 1108</integer>	<integer> 362 1122 1155 1022 1144</integer>
GAlignments object of SRR031715.1138209 SRR031714.756385 SRR031714.2355189 SRR031714.5054563 SRR031715.1722593 SRR031715.2202469 SRR031715.1138209 SRR031714.756385 SRR031714.2355189 SRR031714.5054563	with 6 alig seqnames 9 <rle> chr4 chr4 chr4 chr4 chr4 chr4 width <integer> 37 37 37 37</integer></rle>	strand <rle> <char - - - - njunc <integer> 0 0 0 0</integer></char </rle>	cigar Facter> 37M 37M 37M 37M 37M 37M	qwidth <integer> 37 37 37 37 37</integer>	start <integer> 326 1086 1119 986 1108</integer>	<integer> 362 1122 1155 1022 1144</integer>
GAlignments object of SRR031715.1138209 SRR031714.756385 SRR031714.2355189 SRR031714.5054563 SRR031715.1722593 SRR031715.2202469 SRR031714.756385 SRR031714.2355189 SRR031714.5054563 SRR031715.1722593 SRR031715.2202469	with 6 alig seqnames 9 <rle> chr4 chr4 chr4 chr4 chr4 chr4 width <integer> 37 37 37 37 37 37</integer></rle>	strand <rle> <char - - - - njunc <integer> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</integer></char </rle>	cigar racter> 37M 37M 37M 37M 37M	qwidth <integer> 37 37 37 37 37 37</integer>	start <integer> 326 1086 1119 986 1108</integer>	<integer> 362 1122 1155 1022 1144</integer>
GAlignments object of SRR031715.1138209 SRR031714.756385 SRR031714.2355189 SRR031714.5054563 SRR031715.1722593 SRR031715.2202469 SRR031715.1138209 SRR031714.756385 SRR031714.2355189 SRR031714.5054563 SRR031715.1722593	with 6 alig seqnames 9 <rle> chr4 chr4 chr4 chr4 chr4 chr4 width <integer> 37 37 37 37 37 37</integer></rle>	strand <rle> <char - - - - njunc <integer> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</integer></char </rle>	cigar racter> 37M 37M 37M 37M 37M	qwidth <integer> 37 37 37 37 37 37</integer>	start <integer> 326 1086 1119 986 1108</integer>	<integer> 362 1122 1155 1022 1144</integer>

According to the SAM format specifications, the aligner is expected to mark each alignment pair as *proper* or not (flag bit 0x2 in the SAM format). The SAM Spec only says that a pair is *proper* if the *first* and *last* alignments in the pair are "properly aligned according to the aligner". So the exact criteria used for setting this flag is left to the aligner.

We use isProperPair to extract this flag from the GAlignmentPairs object:

```
> table(isProperPair(U3.galp))
FALSE TRUE
```

29581 45828

Even though we could do *overlap encodings* with the full object, we keep only the *proper* pairs for our downstream analysis:

```
> U3.GALP <- U3.galp[isProperPair(U3.galp)]</pre>
```

Because the aligner used to align those reads can report more than 1 alignment per *original query template* (i.e. per pair of sequences stored in the input files, typically 1 FASTQ file for the *first* ends and 1 FASTQ file for the *last* ends), we shouldn't expect the names of U3.GALP to be unique:

```
> U3.GALP_names_is_dup <- duplicated(names(U3.GALP))
> table(U3.GALP_names_is_dup)
```

U3.GALP\_names\_is\_dup FALSE TRUE 43659 2169

Storing the *query template names* in a factor will be useful:

```
> U3.uqnames <- unique(names(U3.GALP))
> U3.GALP_qnames <- factor(names(U3.GALP), levels=U3.uqnames)</pre>
```

as well as having the mapping between each *query template name* and its first occurence in U3.GALP\_qnames:

> U3.GALP\_dup2unq <- match(U3.GALP\_qnames, U3.GALP\_qnames)</pre>

Our reads can have up to 1 gap per end:

```
> head(unique(cigar(first(U3.GALP))))
```

[1] "37M" "6M58N31M" "25M56N12M" "19M62N18M" "29M222N8M" "9M222N28M"

> head(unique(cigar(last(U3.GALP))))

[1] "37M" "19M58N18M" "12M58N25M" "27M2339N10M" "29M2339N8M" [6] "9M222N28M" > table(njunc(first(U3.GALP)), njunc(last(U3.GALP))) 0 1

0 44510 596 1 637 85

Like for our single-end reads, the following tables indicate that indels were not allowed/supported during the alignment process:

#### Working with aligned nucleotides (WORK-IN-PROGRESS!)

> colSums(c	igar0pTa	able(c	igar(first	(U3.GALP)	)))				
М	I	D	Ν	S	н	Р	=	х	
1695636	Θ	Θ	673919	Θ	Θ	Θ	Θ	Θ	
<pre>&gt; colSums(cigar0pTable(cigar(last(U3.GALP))))</pre>									
М	I	D	Ν	S	н	Р	=	Х	
1695636	Θ	0	630395	Θ	Θ	Θ	Θ	Θ	

#### 7 sessionInfo()

> sessionInfo()

```
R version 3.4.4 (2018-03-15)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.4 LTS
Matrix products: default
BLAS: /home/biocbuild/bbs-3.6-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.6-bioc/R/lib/libRlapack.so
locale:
 [1] LC_CTYPE=en_US.UTF-8
                                LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8
                                LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8
                                LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8
                                LC_NAME=C
 [9] LC_ADDRESS=C
                                LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
attached base packages:
                                  graphics grDevices utils
                                                                 datasets
[1] stats4
              parallel stats
[8] methods
              base
other attached packages:
 [1] RNAseqData.HNRNPC.bam.chr14_0.16.0
 [2] BSgenome.Dmelanogaster.UCSC.dm3_1.4.0
 [3] BSgenome_1.46.0
 [4] rtracklayer_1.38.3
 [5] TxDb.Dmelanogaster.UCSC.dm3.ensGene_3.2.2
 [6] GenomicFeatures_1.30.3
 [7] AnnotationDbi_1.40.0
 [8] pasillaBamSubset_0.16.0
 [9] GenomicAlignments_1.14.2
[10] Rsamtools_1.30.0
[11] Biostrings_2.46.0
[12] XVector_0.18.0
[13] SummarizedExperiment_1.8.1
[14] DelayedArray_0.4.1
```

```
[15] matrixStats_0.53.1
```

[16] Biobase_2.38.0		
<pre>[17] GenomicRanges_1.30.3</pre>		
<pre>[18] GenomeInfoDb_1.14.0</pre>		
[19] IRanges_2.12.0		
[20] S4Vectors_0.16.0		
<pre>[21] BiocGenerics_0.24.0</pre>		
loaded via a namespace (and	not attached):	
[1] Rcpp_0.12.16	compiler_3.4.4	prettyunits_1.0.2
<pre>[4] progress_1.1.2</pre>	bitops_1.0-6	tools_3.4.4
[7] zlibbioc_1.24.0	biomaRt_2.34.2	bit_1.1-12
[10] digest_0.6.15	memoise_1.1.0	evaluate_0.10.1
[13] RSQLite_2.0	lattice_0.20-35	pkgconfig_2.0.1
[16] Matrix_1.2-12	DBI_0.8	yaml_2.1.18
[19] GenomeInfoDbData_1.0.0	httr_1.3.1	stringr_1.3.0
[22] knitr_1.20	rprojroot_1.3-2	bit64_0.9-7
[25] grid_3.4.4	R6_2.2.2	XML_3.98-1.10
[28] RMySQL_0.10.14	BiocParallel_1.12.0	rmarkdown_1.9
[31] blob_1.1.1	magrittr_1.5	backports_1.1.2
<pre>[34] htmltools_0.3.6</pre>	assertthat_0.2.0	BiocStyle_2.6.1
[37] stringi_1.1.7	RCurl_1.95-4.10	