A basic guide to mapping RNA-seq reads to genes and quantifying their abundances with Bioconductor

In this document, we will give a basic overview of how to use Bioconductor packages to map aligned RNA-seq reads to genes and find the count for each gene (the number of reads mapping to the gene). We assume that the reads have been aligned to a reference genome and that the data are stored in a file `AlignedReads.bam` in the BAM format (that is, the binary version of the SAM format). The SAM format is the standard output format from many alignment packages. We start by installing and loading some Bioconductor packages into `R`:

```
source("http://bioconductor.org/biocLite.R")
biocLite("GenomicRanges")
biocLite("Rsamtools")
biocLite("GenomicFeatures")

library(GenomicRanges)
library(Rsamtools)
library(GenomicFeatures)
```

Load reference genome

The loaded Bioconductor packages will give access to a number of different reference genomes that can be used as annotation sources. Here, we will show how to use the UCSC genome browser. With the commands

```
ucscGenomes()
supportedUCSCtables()
```

we can list the available genomes and the supported UCSC tables (for example, RefSeq genes and Ensembl genes).

We can now load a reference genome and a table and make a transcript database by the following command:

```
txdb <- makeTranscriptDbFromUCSC(genome = "hg19", tablename =
"refGene")
```

The resulting `txdb` is a `TranscriptDb` object. By writing `txdb` in the `R` console, we can retrieve information about the number of available transcripts, exons and coding sequences (cds). For this particular example, there are 40,784 transcripts, 234,319 exons and 205,946 coding sequences.

Group transcripts by gene

We can now group the transcripts in `txdb` by gene, using the `transcriptsBy` function, as follows:

```
transcriptRanges <- transcriptsBy(txdb,by="gene")
```

We can also choose `by="exon"` or `by="cds"` to group the transcripts in `txdb` by other genomic features. We can further group the exons or coding sequences in `txdb` by using the functions `exonsBy` and `cdsBy`. The `transcriptRanges` object is a `GRanges` object, and it is structured as seen in Figure 1. This object has 23,241 elements, one for each gene. As a comparison, we can group the exons in `txdb` by gene, which gives the object `exonRanges`, partly shown in Figure 2, where we can see for example that there are eight exons corresponding to the first gene. Grouping transcripts (or exons, or coding sequences) by exons instead, we obtain an object with 234,319 elements, one for each exon in the loaded table.

In the `transcriptRanges` object, we can find the name of each gene (in the `tx_name` column) and the length of each corresponding transcript (in the `<ranges>` column).

<u>Find lengths of genomic features</u>
When the transcripts (exons/cds) in `txdb` have been grouped by some genomic feature (e.g. gene), we can find the length (the number of nucleotides) of each gene with the command

```
numBases <- sum(width(transcriptRanges))
```

Now, `numBases` is a vector with 23,241 elements, giving the lengths (in nucleotides) of the respective genes. The length of a gene is equal to the sum of the lengths of the intervals in the `<ranges>` column and the row corresponding to the gene in the `transcriptRanges` object (see Figure 1).

<u>Map reads to transcripts</u>
Now assume that we are given aligned reads in the file `AlignedReads.bam`, and that we want to map these reads to genes and count the number of reads mapping to each gene. We load the BAM file by

```
aligns <- readBamGappedAlignments(AlignedReads.bam)
```

The counts can be obtained by

```
counts <- countOverlaps(transcriptRanges,aligns)
```

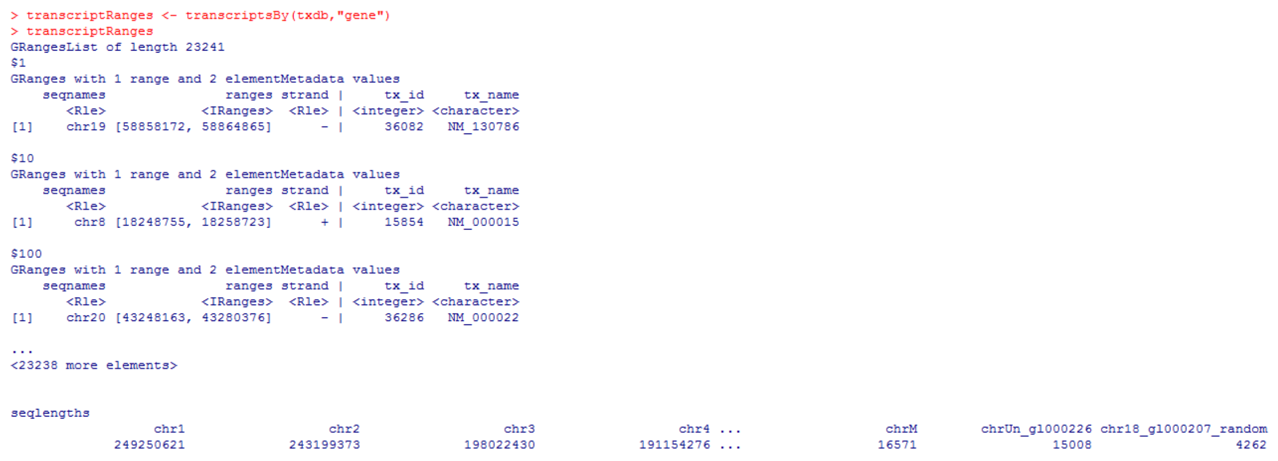
which returns a vector of integers representing the counts for the genes in `transcriptRanges`.

<u>Other operations</u>
We can use the function `transcripts` (or, correspondingly, `exons` or `cds`) to extract genomic features from the `txdb` object, as follows:

```
txs <- transcripts(txdb,columns=c("tx_id","tx_name","gene_id"))
```

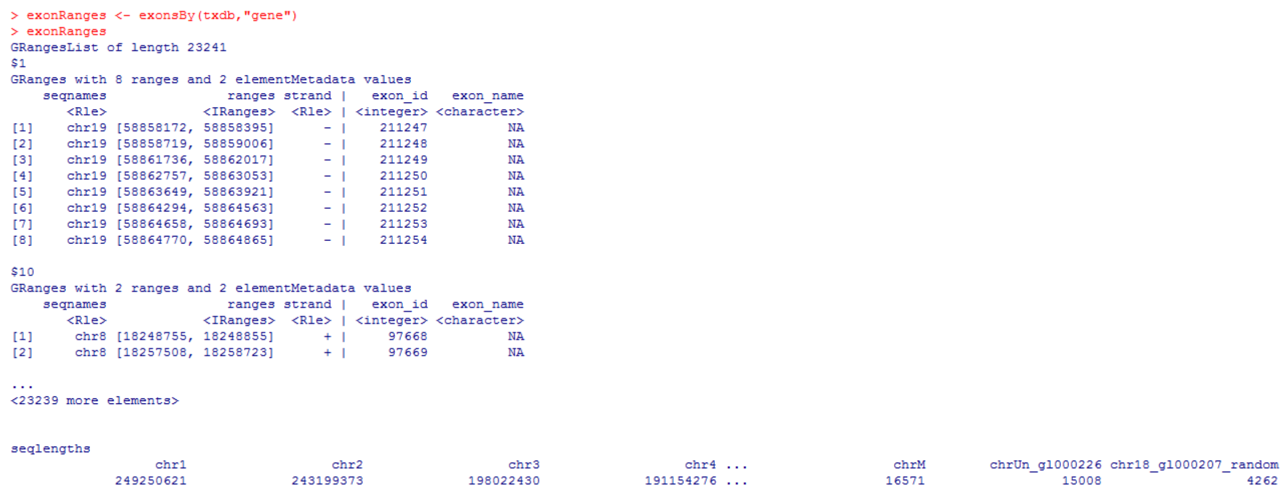
This extracts the indicated columns from `txdb`. An optional `vals` argument, consisting of a list of vectors to restrict the output, can be added. For example, `vals = list(tx_id=...)`. The extracted arguments can be accessed by

```
transcriptNames <-
as.vector(unlist(elementMetadata(txs)["tx_name"]))
```



```
> transcriptRanges <- transcriptsBy(txdb,"gene")
> transcriptRanges
GRangesList of length 23241
$1
GRanges with 1 range and 2 elementMetadata values
    seqnames               ranges strand |     tx_id     tx_name
       <Rle>            <IRanges>  <Rle> | <integer> <character>
[1]    chr19 [58858172, 58864865]      - |     36082   NM_130786

$10
GRanges with 1 range and 2 elementMetadata values
    seqnames               ranges strand |     tx_id     tx_name
       <Rle>            <IRanges>  <Rle> | <integer> <character>
[1]     chr8 [18248755, 18258723]      + |     15854   NM_000015

$100
GRanges with 1 range and 2 elementMetadata values
    seqnames               ranges strand |     tx_id     tx_name
       <Rle>            <IRanges>  <Rle> | <integer> <character>
[1]    chr20 [43248163, 43280376]      - |     36286   NM_000022

...
<23238 more elements>


seqlengths
                  chr1                  chr2                  chr3                  chr4 ...                  chrM        chrUn_gl000226 chr18_gl000207_random
             249250621             243199373             198022430             191154276 ...                 16571                 15008                  4262
```

Figure 1. The `transcriptRanges` object.



```
> exonRanges <- exonsBy(txdb,"gene")
> exonRanges
GRangesList of length 23241
$1
GRanges with 8 ranges and 2 elementMetadata values
    seqnames               ranges strand |   exon_id   exon_name
       <Rle>            <IRanges>  <Rle> | <integer> <character>
[1]    chr19 [58858172, 58858395]      - |    211247          NA
[2]    chr19 [58858719, 58859006]      - |    211248          NA
[3]    chr19 [58861736, 58862017]      - |    211249          NA
[4]    chr19 [58862757, 58863053]      - |    211250          NA
[5]    chr19 [58863649, 58863921]      - |    211251          NA
[6]    chr19 [58864294, 58864563]      - |    211252          NA
[7]    chr19 [58864658, 58864693]      - |    211253          NA
[8]    chr19 [58864770, 58864865]      - |    211254          NA

$10
GRanges with 2 ranges and 2 elementMetadata values
    seqnames               ranges strand |   exon_id   exon_name
       <Rle>            <IRanges>  <Rle> | <integer> <character>
[1]     chr8 [18248755, 18248855]      + |     97668          NA
[2]     chr8 [18257508, 18258723]      + |     97669          NA

...
<23239 more elements>


seqlengths
                  chr1                  chr2                  chr3                  chr4 ...                  chrM        chrUn_gl000226 chr18_gl000207_random
             249250621             243199373             198022430             191154276 ...                 16571                 15008                  4262
```

Figure 2. The `exonRanges` object.