An introduction to qRNASeq

Qiang Hu, Li Yan, Dan Wang, Song Liu

May 9, 2012

Contents

1 Introduction

qRNASeq is an R package to quantify RNA-seq data in different level, such as gene, exon, intron, UTR and exon junction. Given genomic annotation file and aligned sequence result, the package can produce the expression values with user-specified quantification method and genomic regions. The package read aligned sequence data of BAM format directly. The genomic annotation can be GTF, bed and GenePred format, which is available at most of the genomic databases, such as UCSC genome and Ensembl database.

Two quantification methods are developed in the package. The reads counts define the expression levels with the total number of reads mapped in the given region. The RPKM value normalizes the counts by the region width, which can also be calculated in the package. The maximum depth (maxdepth) method estimate the expression level with maximum depth of the given region.

R packages, such as, edgeR, DESeq and beySeq, are well developed to analyze differential expressed genes for the RNA-seq platform. All of these packages require read counts as input data. Our package can be used to prepare the expression values for the subsequent analysis. In the future version, differential test functions will be developed and integrated in the package.

2 Work flow

The package is easy to use. Only three steps are needed to quantify RNA-seq data. First, we need to prepare the gene regions from the genomic annotation data. Then an aligned bam file is used to quantify expression levels in interested genomic regions. At last, the results can be summerized in gene level with different methods.

2.1 Load genomic annotation data

The package use `GRanges` class from the R package*GenomicRanges* to store the genome annotation data. The GTF, bed and GenePred formats are supported in our package. In the following example, a small GTF file built in the package will be used to define the gene regions.

```
> library("qRNASeq")
> gtffile <- system.file("extdata", "gene2.GTF", package="qRNASeq")
> gtfGRL <- read.GTF(file=gtffile, feature=c("exon", "CDS", "intron", "utr"))

load GTF file ...
parse attributes ...
extract exon ...
extract CDS ...
extract intron/utr ...

> names(gtfGRL)

[1] "exon"   "CDS"    "intron" "utr3"   "utr5"

> head(gtfGRL$CDS)

GRanges with 6 ranges and 1 elementMetadata col:
      seqnames               ranges strand |   transcript_id
         <Rle>            <IRanges>  <Rle> |     <character>
  [1]        1 [36748165, 36748301]      + | ENST00000354618
  [2]        1 [36751969, 36752871]      + | ENST00000354618
  [3]        1 [36754661, 36755365]      + | ENST00000354618
  [4]        1 [36756975, 36757147]      + | ENST00000354618
  [5]        1 [36758199, 36758310]      + | ENST00000354618
  [6]        1 [36759452, 36759536]      + | ENST00000354618
  ---
  seqlengths:
    1
   NA
```

The "gene2.GTF" contains two human gene annotation from Ensembl database. The function `read.GTF` can read GTF format file into a list of *GRanges* objects as required in the feature option.

2.2 Quantification

To quantify gene expression levels, the mapped bam file and defined genomic regions are used in the function `QuanBam`. The genomic regions should be a *GRanges* object from the last step. Gene_id is necessary in the annotation columns of `elementMetadata` for each range, because Reads mapped in an exon can come from different transcripts. If the input ranges are annotated in transcript level, some of the regions from different transcripts but from the same gene can be overlapped. The read mapped in the overlapped regions can't be decided which transcript it belongs to. Therefore, we usually quantify abundance in gene level.

For example, we want to quantify the CDS regions of `gtfGRL`. First the gene_id annotation should be added to these regions.

```
> tidx <- match(gtfGRL$CDS@elementMetadata$transcript_id,
+               gtfGRL$exon@elementMetadata$transcript_id)
> gtfGRL$CDS@elementMetadata$gene_id <- gtfGRL$exon@elementMetadata$gene_id[tidx]
> head(gtfGRL$CDS)

GRanges with 6 ranges and 2 elementMetadata cols:
      seqnames               ranges strand |   transcript_id         gene_id
         <Rle>            <IRanges>  <Rle> |     <character>        <factor>
  [1]        1 [36748165, 36748301]      + | ENST00000354618 ENSG00000054118
  [2]        1 [36751969, 36752871]      + | ENST00000354618 ENSG00000054118
  [3]        1 [36754661, 36755365]      + | ENST00000354618 ENSG00000054118
  [4]        1 [36756975, 36757147]      + | ENST00000354618 ENSG00000054118
  [5]        1 [36758199, 36758310]      + | ENST00000354618 ENSG00000054118
  [6]        1 [36759452, 36759536]      + | ENST00000354618 ENSG00000054118
  ---
  seqlengths:
    1
   NA
```

Then the function QuanBam can be used to quantify the expression values based on aligned bam file. There are two ways to handle the overlapped regions in the function. The first straightforward method is to unite all overlapped regions that are from different transcripts in the same genes. In this way, the regions of different transcripts will be reduced into new ranges in their gene levels. Reads from different transcripts but in the same ranges will be counted together.

```
> bamfile <- system.file("extdata", "gene2.bam", package="qRNASeq")
> cdsGR1 <- QuanBam(bam=bamfile, GR=gtfGRL$CDS, method="both",
+                   Reduce="union", ovtype="within")

466 reads are loaded
22 ranges left after reduced
Filtering:
95 reads are not 'within' mapped
371 reads are left
Summarize counts ...
Summarize maximum depths ...

> cdsGR1

GRanges with 22 ranges and 3 elementMetadata cols:
       seqnames               ranges strand   |         gene_id    counts
          <Rle>            <IRanges>  <Rle>   |     <character> <integer>
   [1]        1 [36748165, 36748301]      +   | ENSG00000054118         6
   [2]        1 [36751969, 36752871]      +   | ENSG00000054118        75
   [3]        1 [36754661, 36755365]      +   | ENSG00000054118        53
   [4]        1 [36756975, 36757147]      +   | ENSG00000054118        18
   [5]        1 [36758199, 36758310]      +   | ENSG00000054118         7
   [6]        1 [36759452, 36759536]      +   | ENSG00000054118         8
   [7]        1 [36762184, 36762371]      +   | ENSG00000054118        19
   [8]        1 [36766487, 36766685]      +   | ENSG00000054118        30
   [9]        1 [36767154, 36767297]      +   | ENSG00000054118        20
   ...       ...                  ...     ... ...             ...       ...
  [14]        1 [17739569, 17739674]      -   | ENSG00000179051        10
  [15]        1 [17740033, 17740213]      -   | ENSG00000179051        12
  [16]        1 [17742976, 17743142]      -   | ENSG00000179051        11
  [17]        1 [17747210, 17747324]      -   | ENSG00000179051         3
  [18]        1 [17748699, 17748787]      -   | ENSG00000179051         7
  [19]        1 [17749201, 17749332]      -   | ENSG00000179051        10
  [20]        1 [17752037, 17752180]      -   | ENSG00000179051         9
  [21]        1 [17755602, 17755695]      -   | ENSG00000179051         5
  [22]        1 [17764726, 17765010]      -   | ENSG00000179051         0
         mdepths
       <integer>
   [1]         3
   [2]        11
   [3]         8
   [4]         9
   [5]         4
   [6]         5
   [7]         8
   [8]        14
   [9]        11
   ...       ...
  [14]         9
  [15]         6
  [16]         5
  [17]         3
  [18]         6
  [19]         4
  [20]         6
  [21]         4
  [22]         3
  ---
  seqlengths:
    1
   NA
```

Two methods can be used to quantify the data with the option "method", counts and maxdepth. The counts method calculates the total numbers of the reads that start in the region. The maxdepth method calculate the maximum of the depths in the interested regions. The two methods can be used a if "both" is selected in the option. The option "ovtype" is used to define the reads that correctly align to certain regions. The option with "within" means that the reads only mapped within the certain region are used in the quantification.

Another way to reduce the overlapped regions is the intersection of the regions, which is defined as the most shared regions among the transcripts of a gene. For example, a gene has two transcripts, then the intersection ranges are the common regions that shared by the two transcripts.

```
> cdsGR2 <- QuanBam(bam=bamfile, GR=gtfGRL$CDS, method="both",
+                   Reduce="intersection", ovtype="any")

466 reads are loaded
13 ranges left after reduced
Filtering:
371 reads are not 'any' mapped
95 reads are left
Summarize counts ...
Summarize maximum depths ...

> cdsGR2

GRanges with 13 ranges and 3 elementMetadata cols:
       seqnames               ranges strand |         gene_id    counts
          <Rle>            <IRanges>  <Rle> |     <character> <integer>
   [1]        1 [36748165, 36748285]      + | ENSG00000054118         7
   [2]        1 [17735586, 17735690]      - | ENSG00000179051         6
   [3]        1 [17736470, 17736547]      - | ENSG00000179051         6
   [4]        1 [17738618, 17738690]      - | ENSG00000179051         7
   [5]        1 [17739569, 17739674]      - | ENSG00000179051        10
   [6]        1 [17740033, 17740213]      - | ENSG00000179051        12
   [7]        1 [17742976, 17743142]      - | ENSG00000179051        11
   [8]        1 [17747210, 17747324]      - | ENSG00000179051         3
   [9]        1 [17748699, 17748787]      - | ENSG00000179051         7
  [10]        1 [17749201, 17749332]      - | ENSG00000179051        12
  [11]        1 [17752037, 17752180]      - | ENSG00000179051         9
  [12]        1 [17755602, 17755695]      - | ENSG00000179051         5
  [13]        1 [17764726, 17765010]      - | ENSG00000179051         0
         mdepths
       <integer>
   [1]         2
   [2]         4
   [3]         4
   [4]         7
   [5]         9
   [6]         6
   [7]         5
   [8]         3
   [9]         6
  [10]         6
  [11]         6
  [12]         4
  [13]         3
  ---
  seqlengths:
    1
   NA
```

2.3 Summary

The counts and mdepths results in exon or CDS level can be summarized to gene level by different methods, such as sum, mean, median, max, min and so on. RPKM is reads per kilobase of exon model per million mapped reads, which is also supported in the function `QuanGR`.

```
> cdsRes1 <- QuanGR(cdsGR1, method="mean", Quan="mdepths", by="gene_id")
> cdsRes1

                mean_mdepths
ENSG00000054118     9.200000
ENSG00000179051     4.916667

> cdsRes2 <- QuanGR(cdsGR1, method="sum", Quan="counts", by="gene_id")
> cdsRes2

                sum_counts
ENSG00000054118        290
ENSG00000179051         81

> cdsRes3 <- QuanGR(cdsGR1, method="RPKM", Quan="counts", by="gene_id")
> cdsRes3

                RPKM_counts
ENSG00000054118    272549.2
ENSG00000179051    139151.6
```

The results can be used as input data in the downstream differential expression analysis. Also the results with the abundance in different level can be used in the isoform analysis.

3 Differential expression

4 Session information

```
> sessionInfo()

R version 2.15.0 (2012-03-30)
Platform: x86_64-pc-linux-gnu (64-bit)

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=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base
```

```
other attached packages:
 [1] qRNASeq_0.0.1       ShortRead_1.14.2    latticeExtra_0.6-19
 [4] RColorBrewer_1.0-5  Rsamtools_1.8.3     lattice_0.20-6
 [7] Biostrings_2.24.1   GenomicRanges_1.8.3 IRanges_1.14.2
[10] BiocGenerics_0.2.0

loaded via a namespace (and not attached):
[1] Biobase_2.16.0 bitops_1.0-4.1 grid_2.15.0    hwriter_1.3    stats4_2.15.0
[6] tools_2.15.0   zlibbioc_1.2.0
```