An Introduction to Sequgio

Chen Suo, Stefano Calza, Agus Salim and Yudi Pawitan

November 4, 2013

Contents

1 Introduction

This document provides a brief guide to the *Sequgio* package, which is a package for gene isoform expression and isoform-specific read distribution estimation based on RNA-seq data.

There are two components to this package. These are: i) Construct design matrices used in expression estimation. ii) Output the expression levels and optionally the read distribution, standard error of the expression estimates and etc.

The Sequgio method is motivated upon the fact that read intensity in RNA sequencing data is often not uniform, in which case standard methods would produce biased estimates. The problem is that the read intensity pattern is not identifiable from data observed in a single sample. The method accounts for non-uniform isoform-specific read distribution and gene isoform expression estimation. A statistical regularization with L_1 smoothing penalty is imposed to control the estimation. Also, for estimability reasons, the method uses information across samples from the same gene [**?**].

The *Sequgio* package is available at bioconductor.org and can be downloaded via `biocLite`:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("Sequgio")
```

```
> library(Sequgio)
```

For better performances the package support parallel computing via the *BiocParallel* package which is loaded automatically. For parallel processing set the parameters to the ones suiting your platform. We will use sequencial computation here.

```
> param <- SerialParam()
```

2 Example data

We demonstrate the functionality of this R package using RNA sequencing samples provided byt the *RNAseqData.HNRNPC.bam.chr14*.

```
> library(RNAseqData.HNRNPC.bam.chr14)
```

2.1 Step 1: create the annotation template

The first step is to provide a `TranscriptDb` object. We will use the one provided by the package *TxDb.Hsapiens.UCSC.hg19.knov*

```
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
```

The `TranscriptDb` object must be preprocessed to generate *disjoint* regions using the `reshapeTxDb` function. We need to set the read length paramters to the one matching your experiment.

Given that the data we use are limited to chromosome 14 we subset the `TranscriptDb` to reduce computation burden.

```
> seqs <- seqnames(seqinfo(TxDb.Hsapiens.UCSC.hg19.knownGene))
> sel <- rep(FALSE,length(seqs))
> names(sel) <- seqs
> sel["chr14"] <- TRUE
> isActiveSeq(TxDb.Hsapiens.UCSC.hg19.knownGene) <- sel
```

```
> txdb <- reshapeTxDb(TxDb.Hsapiens.UCSC.hg19.knownGene,probelen = 50L,mcpar=param)
```

This step has to be repeated only if the user changes annotation database, or some paramters (with/without junctions, etc...).

2.2 Step 2: create the design matrix object

The second step is to create design matrices for each "transcriptional unit" (see references). These matrices will be used in the fitting procedure.

Several parameters can be tuned. The main one regards which kind of library is the experiment using: *paired-end* ("PE") or *single-end* ("SE", the default). This can be set with the `method` argument. The required `mulen` argument provides an estimate of the average *fragment length*.

Here we will use paired-end data.

```
> Design <- makeXmatrix(txdb,method="PE",probelen=50L,mulen=200,mcpar=param)
```

As for *Step 1* also this step has to be performed only once.

For demostration *Sequgio* provides the previous objects.

```
> data("TxDb")
> data("Design")
```

2.3 Step 3: import BAM files and create a *counts* matrix

Models are fit based on a matrix with read counts for every *region* in every sample. We will now import the aligned read counts in BAM files into Ras an object called 'allCounts'. To do so you need to create a `target` object storing the filenames (with full path) and sample names to be used for count matrix headings. If BAI file are available, they can be provided in the `target` object.

The resulting object (`allCounts`) will count for every exons the overlapping reads.

Let's first create the `target` object storing BAM files locations and sample names

```
> target <- data.frame(filenames=RNAseqData.HNRNPC.bam.chr14_BAMFILES,
+                      index=RNAseqData.HNRNPC.bam.chr14_BAMFILES,
+                      samplenames=RNAseqData.HNRNPC.bam.chr14_RUNNAMES, stringsAsFactors=FALSE)
```

We then counts reads.

```
> bigM <- getCounts(target,txdb,mcpar=param)
> allCounts <- as.matrix(bigM)
> rm(bigM)
```

Again for ease of computation counts object is already provided.

```
> data("Counts")
```

We can see how many read we counted

```
> colSums(allCounts)
```

```
ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303 ERR127304
   192785    209910    209477    187342    178531    188635    185209
ERR127305
   182696
```

2.4 Step 4: fit models

Using the region(exons)-by-sample counts matrix (`allCounts`) and the design matrices object (`Design`) we can now fit models.

```
> data(Counts)
> data(Design)
> iGenes <- names(Design)
> ## Fit a single 'transcriptional unit' (one element in Design)
> fit1 <- fitModels(iGenes[22],design=Design,counts=allCounts)
> # More than one using list/for loops/mclapply/etc...
> fit2 <- lapply(iGenes[21:22],fitModels,design=Design,counts=allCounts)
```