Rsubread: a tool kit for processing next-gen sequencing data

Wei Shi

30 May 2013

1 Introduction

This vignette provides a brief description to the Rsubread package. For more detailed description and case studies, you may refer to the Users Guide. The Users Guide can be accessed via typing the following command in your R session:

```
> library(Rsubread)
> RsubreadUsersGuide()
```

The Rsubread package provides facilities for processing the read data generated by the next-gen sequencing technologies. These facilities include quality assessment, read alignment, read summarization, exon-exon junction detection, absolute expression calling and SNP discovery. This package can be used to process both short and long reads. It supports major sequencing platforms such as Illumina GA/HiSeq, Roche 454, ABI SOLiD and Ion Torrent.

The Subread aligner (`align` function) is very efficient and accurate in mapping reads generated by the next-gen sequencing technologies (including both gDNA sequencing and RNA sequencing). It adopts a mapping paradigm called "seed-and-vote", which extracts a number of 16mers (called seeds or subreads) for each read and uses these subreads to vote for the mapping location of the read. This paradigm greatly reduces the computational burden of the read mapping and achieves a superior mapping accuracy, compared to the traditional "seed-and-extend" paradigm (Liao et al. 2013).

This package also includes an exon-exon junction detection function, `subjunc`, which makes use of the powerful "seed-and-vote" paradigm as well. In addition to the accurate and efficient discovery of exon-exon junctions from using RNA-seq data, `subjunc` has a much higher accuracy and sensivity in mapping exon-spanning reads than other junction detector (Liao et al. 2013). This makes it particulary useful for the genomic variation detection performed on RNA-seq data.

An important step for processing the next-gen sequencing data is to assign mapped reads to genomic features such as genes, exons, and promoters. This package includes

a general-purpose read summarization function `featureCounts`, which takes as input the SAM/BAM files and assigns them to genomic features. In-built annotations are provided for users' convenience. `featureCounts` is an extremely fast and flexible read summarization function.

Different from microarray technologies, the next-gen sequencing technologies do not provide Present/Absent calls for genomic features such as genes. We have developed an algorithm to use the background noise measured from the RNA-seq data to call absolutely expressed genes. The function `detectionCall` returns a detection p value for each gene from the read mapping results.

We are now developing a new SNP calling algorithm which is being implemented in function `callSNPs`. Our prelimiarly results showed that it compared favorably to competing methods, but it is an order of magnitude faster. It does not require the indel realignment and quality recalibration steps.

This package also includes some other useful functions such as quality assessment (`qualityScores`, `atgcContent`), duplicate read removal (`removeDupReads`) and mapping percentage calculation (`propmapped`).

Most of the functions in this package are written in C programming language and thus they are very efficient. Many of these functions are also implemented in the `Subread` package that is entirely written in C and can be downloaded from `http://subread.sourceforge.net`.

2 Read alignment

Using Rsubread to perform read alignments is straightforward. It includes two steps: the first step is to create an index (an one-off operation) and the second is just to perform the alignment.

Step 1: Build an index for the reference genome

The Rsubread package includes a dummy reference sequence which was made up from 900 read sequences. One thousand 100bp reads were taken from a human brain reference RNA-seq dataset created by the SEQC project. Sequences from 900 of them were concatenated to build an artifical reference sequence, which is used as the reference for the mapping of these 1000 reads in this example. These reads were generated using Illumina Genome Analyzer in 2010.

Below are the commands for building an index for this dummy reference sequence:

```
> library(Rsubread)
> ref <- system.file("extdata","reference.fa",package="Rsubread")
> path <- system.file("extdata",package="Rsubread")
> buildindex(basename=file.path(path,"reference_index"),reference=ref)
```

```
Building a base-space index.
Size of memory specified=3700 MB
Base name of the built index = /private/tmp/Rtmp5lMQdO/Rinst8008495ca940/Rsubread/extd
Checking the integrity of provided reference sequences ...
Scanning uninformative subreads in reference sequences ...
completed=8.3%; time used=1.2s; rate=6.3k bps/s; total=0m bps
completed=16.7%; time used=1.2s; rate=12.7k bps/s; total=0m bps
completed=25.0%; time used=1.2s; rate=19.0k bps/s; total=0m bps
completed=33.3%; time used=1.2s; rate=25.2k bps/s; total=0m bps
completed=41.7%; time used=1.2s; rate=31.5k bps/s; total=0m bps
completed=50.0%; time used=1.2s; rate=37.7k bps/s; total=0m bps
completed=58.3%; time used=1.2s; rate=43.9k bps/s; total=0m bps
completed=66.7%; time used=1.2s; rate=50.1k bps/s; total=0m bps
completed=75.0%; time used=1.2s; rate=56.3k bps/s; total=0m bps
completed=83.3%; time used=1.2s; rate=62.4k bps/s; total=0m bps
completed=91.7%; time used=1.2s; rate=68.6k bps/s; total=0m bps
completed=100.0%; time used=1.2s; rate=74.7k bps/s; total=0m bps
1 uninformative subreads were found and they were excluded from the index.
Building the index...
completed=8.3%; time used=1.5s; rate=126.4k bps/s; total=0m bps
completed=16.7%; time used=1.6s; rate=128.6k bps/s; total=0m bps
completed=25.0%; time used=1.6s; rate=127.8k bps/s; total=0m bps
completed=33.3%; time used=1.7s; rate=129.7k bps/s; total=0m bps
completed=41.7%; time used=1.7s; rate=130.3k bps/s; total=0m bps
completed=50.0%; time used=1.8s; rate=133.1k bps/s; total=0m bps
completed=58.3%; time used=1.8s; rate=136.3k bps/s; total=0m bps
completed=66.7%; time used=1.9s; rate=138.1k bps/s; total=0m bps
completed=75.0%; time used=1.9s; rate=139.5k bps/s; total=0m bps
completed=83.3%; time used=2.0s; rate=142.8k bps/s; total=0m bps
completed=91.7%; time used=2.0s; rate=145.7k bps/s; total=0m bps
completed=100.0%; time used=2.0s; rate=146.2k bps/s; total=0m bps
Processing chromosome files ...
Saving the current block of index ...
  [ 0.0% finished ]
  [ 6.7% finished ]
  [ 13.3% finished ]
  [ 20.0% finished ]
  [ 26.7% finished ]
  [ 33.3% finished ]
  [ 40.0% finished ]
  [ 46.7% finished ]
  [ 53.3% finished ]
```

```
  [ 60.0% finished ]
  [ 66.7% finished ]
  [ 73.3% finished ]
  [ 80.0% finished ]
  [ 86.7% finished ]
  [ 93.3% finished ]
  [ 100.0% finished ]

Index /private/tmp/Rtmp5lMQdO/Rinst8008495ca940/Rsubread/extdata/reference_index was s
```

The created index files are saved to the "extdata" folder in the directory where Rsubread package was installed. Rsubread creates a hash table for indexing the reference genome. Keys in the hash table are the 16bp sequences and hash values are their corresponding chromosomal locations. Color space index can be built by setting the `colorsapce` argument to `TRUE`.

A unique feature of Rsubread is that it allows users to control the computer memory usage in the read mapping process. Users can do this by specifying the amount of memory (in MB) to be used for mapping. By default, 3700MB of memory will be used. This will for example partition the index into two chunks for the human genome. Only one chunk of index will be existent in the memory at any time. To load the entire index into the memory, users can specify the requested amount of memory to be 8000MB for the human genome (the actual memory usage is up to 7.6GB). With this setting, Subread has the fastest mapping speed.

Step 2: Map reads to the reference sequence

Map the 1000 reads to the reference using the index built in Step 1:

```
> reads <- system.file("extdata","reads.txt",package="Rsubread")
> align(index=file.path(path,"reference_index"),readfile1=reads,output_file=file.pat

Number of selected subreads = 10
Consensus threshold = 3
Number of threads=1
Number of indels allowed=5
Loading the 01-th index file ...

Processing reads ...

Detecting indels ...
  [ 0.0% finished ]
  [ 100.0% finished ]
```

```
1000 reads were processed.
Saving mapping results for these reads ...
  [ 0.0% finished ]
  [ 10.0% finished ]
  [ 20.0% finished ]
  [ 30.0% finished ]
  [ 40.0% finished ]
  [ 50.0% finished ]
  [ 60.0% finished ]
  [ 70.0% finished ]
  [ 80.0% finished ]
  [ 90.0% finished ]

 1000 reads were processed in 0.6 seconds.
 Percentage of mapped reads is 90.30%.

 Done.
```

Up to 16 indels are allowed in the mapping. The mapping can be performed in multithread mode (`nthreads` option).

3 Counting mapped reads for genomic features

The `featureCounts` function is a general-purpose read summarization function, which assigns to the genomic features the mapped reads that were generated from genomic DNA and RNA sequencing.

This function takes as input a set of files containing read mapping results and then assigns the mapped reads to the genomic features. The input files can be in SAM or BAM format. It can be used to assign reads to genes, exons or any features users defined. This function includes the NCBI gene annotations for mm9, mm10 and hg19.

4 Finding exon junctions

The RNA-seq technology provides a unique opportunity to identify the alternative splicing events that occur during the gene transcription process. This function makes use of a powerful read mapping paradigm we developed ("seed-and-vote") to accurately detect the exon-exon junctions (Liao et al. 2013).

This function takes as input the mapping results (in SAM format) from a read aligner, preferably the Subread aligner included in this package (`align`). It extracts a number of subreads (16mers) from each read, maps them to the reference genome and then

identifies the two best mapping locations for each read. A verification step is then applied to determine whether each read should be mapped as an exon-spanning read or not and to report the discovered exon-exon junctions. The donor and receptor sites, which are splicing signals, are required to be present when calling exon-exon junctions. The indels identified in the mapped reads included in the provided SAM file will be kept in the output of this function. This function also reports the indels discovered in the exon-spanning reads.

The output of this function includes the discovered exon-exon junctions and also read mapping results.

5 Base quality scores

Quality scores give the probability of base calling being wrong for each base in the reads, which is useful for examining quality of the sequencing data. The `qualityScores` function returns quality score information from a specified number of reads included in a `FASTQ` format file.

```
> library(Rsubread)
> reads <- system.file("extdata","reads.txt",package="Rsubread")
> x <- qualityScores(filename=reads,nreads=1000)
> boxplot(x)
```

6 GC content

The `atgcContent` function returns the fraction of each nucleotide type in the reads at each base location.

```
> library(Rsubread)
> reads <- system.file("extdata","reads.txt",package="Rsubread")
> ## Fraction of A,T,G and C in the entire dataset
> x <- atgcContent(filename=reads,basewise=FALSE)
> ## Fraction of A,T,G and C at each base location across all the reads
> xb <- atgcContent(filename=reads,basewise=TRUE)
```

7 Mapping percentage

Function `propmapped` returns the proportion of mapped reads include in a SAM/BAM file.

```
> library(Rsubread)
> results <- system.file("extdata","alignResults.SAM",package="Rsubread")
> propmapped(results)
```

```
                                                                      Samples
1 /private/tmp/Rtmp5lMQdO/Rinst8008495ca940/Rsubread/extdata/alignResults.SAM
  NumTotal NumMapped PropMapped
1     1000       903      0.903
```

8 Citation

Yang Liao, Gordon K Smyth and Wei Shi (2013). The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. Nucleic Acids Research, Volume 41, accepted on 8 March (In Press).

9 Authors

Drs Wei Shi and Yang Liao
Bioinformatics Division
The Walter and Eliza Hall Institute of Medical Research
1G Royal Parade, Parkville, Victoria 3052
Australia

10 Contact

Please contact Wei Shi (shi at wehi dot edu dot au) if you have any inquires.