Rsubread: An R package for aligning next-generation sequencing reads

Wei Shi

1 September 2011

1 Introduction

This package can be used to align next-generation sequencing reads to the reference genome and find exon junctions using RNA-seq read data. The read alignment uses a novel mapping paradigm called "seed-and-vote", which has been demonstrated to be much more efficient and sensitive than the "seed-and-extend" algorithm. The exon junction algorithm implemented in this package employs the similar idea. This package also implements an algorithm which is designed for calling absolutely expressed genes (each gene is assigned a detection p value).

Read mapping and exon junction detection functions in this package are R wrapper functions which call the underlying `C` functions to perform the alignment. The C programs, which implements these novel algorithms, can be freely downloaded from `http://subread.sourceforge.net`.

In addition to the read mapping and exon junction detection functions, this package provides other useful functions such as summarization of read counts to genomic features, quality assessment and so on.

2 Read alignment

There are two steps for mapping reads using Rsubread:

Step 1: Build an index for reference genome
Index building in Rsubread is simple and quick. Building an index for human genome takes about 1 hour.

The Rsubread package includes a sample reference sequence which was made up from 900 read sequences. 1000 reads were extracted from one of the Human Brain Reference RNA-seq datasets generated by the SEQC project (`http://www.fda.gov/ScienceResearch/BioinformaticsTools/MicroarrayQualityControlProject/default.htm`). Read sequences from 900 of them were concatenated to form a synthetic reference

sequence, which is used as the reference for the mapping of these 1000 reads. This small set of reads were also included in this package including their sequences and quality scores. These reads are 100bp long and generated by Illumina Genome Analyzer.

Here is an example of building an index for a reference genome using the sample reference included in this package:

```
> 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)
```

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 human genome. Only one chunk of index will be existent in the memory at any time. If larger memory is used and the entire index can be loaded into the memory in one go (e.g. 7400GB of memory is used for human genome), then the running time will be reduced by half. In a comparison with six "seed-and-extend" aligner, Subread was found to to be twice as fast as the second fastest aligner and 4-50 time faster than other aligners (the default setting of Subread aligner was used for comparison). When less memory is used, running time will increase accordingly.

Step 2: Map reads to the reference genome

Mapping reads using the read data included in this package and also the index built in the last step:

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

Two key parameters used by this function are the number of subreads selected (`nsubreads` option) and the consensus threshold (`TH1` option) for determining mapping locations (`TH2` for the second read in a pair). We recommend using the default setting of these parameters because they were found to perform best in the evaluations using both simulation data and real data. Up to 16 indels are allowed in the mapping (`indels` option). Paired-end read mapping is also supported (`readfile2,TH2,min_distance,max_distance`

options). The mapping can be carried out in multithread mode (`nthreads` option). Mapping results are saved in a SAM format file. Please refer to the help page for this function for more details.

This function supports the read mapping for all major platforms including Illumina GA/HiSeq, ABI SOLiD, Roche 454 and Heliscope. It can map reads of both fixed length and variable length. It is capable for mapping long reads (>200bp) as well. When mapping long reads, the default setting for `nsubreads` and `TH1` are still recommended.

3 Counting mapped reads for genomic features

The `featureCounts` function summarizes read counts to genomic features. It uses the in-built annotation files or annotation provided by users to count the number of mapped reads for each feature. The in-built annotation files include annotation information for each exon in the mouse and human genomes (using NCBI build 37.2 annotation). Read counts can be summarized to gene level or exon level. This function can be used for general-purpose read summarization as well (for both DNA-seq reads and RNA-seq reads).

4 Finding exon junctions

The `subjunc` function takes as input FASTQ/FASTA files and outputs the exon junction locations found in the reference genome and also the numbers of reads supporting the discovered junctions.

5 Calling SNPs and INDELs

The `callSNPs` function implements a very simple method for calling SNPs or INDELs for a sequencing dataset. To call a chromosomal location containing a SNP or INDEL, that location must be covered by a certain number of reads (5 by default) and at least a certain fraction (0.5 by default) of these reads must provide bases (or insertions/deletions) which are different from the reference base. The input to this function is a SAM file, which is the output of a read alignment.

6 Removing duplicated reads

The functionremoveDupReads takes as input a SAM file and removes those reads, which are not only found to be mapped to exactly the same location (as determined by the mapping location of the first base of the read) and but also are found to be duplicated more than a certain number of times (50 by default).

7 Quality assessment

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 randomly extracts quality score information for a specified number of reads from a `FASTQ` format file.

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

GC content
Bias of GC content has been reported for the sequencing data. The `atgcContent` function can be used to get the fraction of each nucleotide (A,T,G,C) in the entire data file or at each base location across all the reads.

```
> 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)
```

8 Others

Percentage of mapped reads

Function `propmapped` counts the number of mapped reads in a SAM format file and gives their proportion in all the reads:

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

```
                                                                 Samples NumTotal
1 /tmp/Rtmp2Qxe88/Rinst41a041052d87/Rsubread/extdata/alignResults.SAM     1000
  NumMapped PropMapped
1       903      0.903
```

9 Citation

Yang Liao and Wei Shi. "Seed-and-vote: the next-generation read mapping paradigm", submitted.

10 Authors

Wei Shi (maintainer of Rsubread), Bioinformatics Division, The Walter and Eliza Hall Institute of Medical Research, Australia
Yang Liao (maintainer of Subread), Department of Computer Science and Software Engineering, The University of Melbourne, Australia
Jenny Zhiying Dai, Department of Computer Science and Software Engineering, The University of Melbourne, Australia

11 Contact

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