Mapping, read counts and visualization

Mapping

Mapping refers to the process of aligning short reads to a reference sequence, whether the reference is a complete genome, transcriptome, or a *de novo* genome/transcriptome assembly. There are numerous programs that have been developed to map reads to a reference sequence that vary in their algorithms and therefore speed (see Flicek and Birney 2009 and references therein). The program that we will utilize is called BWA (Li and Durbin 2009). It uses a Burrow's Wheeler Transform method that results in much faster processing than the first wave of programs that used a hash-based algorithm such as MAQ (Li et al. 2008). The goal of mapping is to create an alignment file also known as a Sequence/Alignment Map (SAM) file for each of your samples. This SAM file will contain one line for each of the reads in your sample denoting the reference sequence (genes, contigs, or gene regions) to which it maps, the position in the reference sequence, and a Phred-scaled quality score of the mapping, among other details (Li et al. 2009). We will then use the SAM files to visualize the alignment in a browser called IGV, and also count the number of reads that map to each reference contig, data that could form the basis of a differential gene expression analysis. Tomorrow, we will also use the SAM files to identify polymorphisms across the dataset and to genotype the 12 individuals.

There are several parameters that can be defined for the alignment process, including: the number of differences allowed between reference and query (-n), the length (-l) and number of differences allowed in the seed (-k), the number allowed and penalty for gap openings (-o, -O), and the number and penalty for gap extensions (-e, -E). Changing these parameters will change the number and quality of reads that map to reference and the time it takes to complete mapping a sample. For a complete list of the parameters and their default values go to http://bio-bwa.sourceforge.net/bwa.shtml.

We will map the reads from each of your trimmed and clipped FASTQ files to the *de novo* reference assembly that you have just created. Specifically, we will 1) create an index for the reference assembly (just once), which will help the aligner (and other downstream software) to quickly scan the reference, 2) for each sample, map reads to the reference assembly, and 3) convert the resulting file into the SAM file format and append "read group" names to the SAM file for each sample. Steps 2 and 3 are "piped," or put together feeding the output of one program in as the input for the next program. The read groups, which can have the same names as your sample names, will be appended to each file and will become critical for the downstream SNP detection step. The read group name in each SAM file will connect the reads back to individual samples after files have been merged for SNP detection. All of the above steps for all samples can be "batch" processed at once by editing the bash script `BWAaln.sh`. We then want to remove all duplicate reads, for which we need to use the `MarkDuplicates` program from the software package "Picard". Picard uses the binary equivalent of SAM files, BAM, as input, so first we need to convert the files using SAMtools.

These steps are performed by the `convert_to_BAM_and_dedup.sh` bash script.

Resources

Burrows-Wheeler Aligner (BWA) – available at: http://bio-bwa.sourcefourge.net
SAM file description - http://samtools.sourceforge.net/SAM1.pdf
SAMtools - http://samtools.sourceforge.net/samtools.shtml

Flicek, P, Birney, E. 2009. Sense from sequence reads: methods for alignment and assembly. *Nature methods* 6:S6-S12.

Li, H, Durbin, R. 2009. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* 25:1754-1760.

Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, and Durbin R. 2009. The sequence alignment/map format and SAMtools. *Bioinformatics* 25: 2078-2079.

Li, H, Ruan, J, Durbin, R. 2008. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome research* 18:1851-1858.

Picard tools: http://picard.sourceforge.net/ : Java-based programs for manipulation of .sam and .bam files.

Exercise

1. Open the `BWAaln.sh` script in nano, by typing:

```
nano ../scripts/BWAaln.sh

The default parameters are currently set as:

-n .01 -k 5 -l 30 -t 2

You can change them to something else if you like.
```

It is in many cases very useful to test the effects of the different parameters on the alignment. Parameters that you can specify in bwa are:

-n *NUM*	Maximum edit distance if the value is INT, or the fraction of missing alignments given 2% uniform base error rate if FLOAT. In the latter case, the maximum edit distance is automatically chosen for different read lengths. [0.04]
-o *INT*	Maximum number of gap opens [1]
-e *INT*	Maximum number of gap extensions, -1 for k-difference mode (disallowing long gaps) [-1]
-d *INT*	Disallow a long deletion within INT bp towards the 3'-end [16]
-i *INT*	Disallow an indel within INT bp towards the ends [5]
-l *INT*	Take the first INT subsequence as seed. If INT is larger than the query sequence, seeding will be disabled. For long reads, this option is typically ranged from 25 to 35 for '-k 2'. [inf]
-k *INT*	Maximum edit distance in the seed [2]
-t *INT*	Number of threads (multi-threading mode) [1]
-M *INT*	Mismatch penalty. BWA will not search for suboptimal hits with a score lower than (bestScore-misMsc). [3]
-O *INT*	Gap open penalty [11]
-E *INT*	Gap extension penalty [4]

The parameters that potentially can have the largest effects on the output (and the speed) are –n (maximum allowed mismatches), -l (size of the seed) and –k (maximum allowed mismatches in the seed).

2. Execute the script by typing:

```
sh ../scripts/BWAaln.sh
```

Then note how long it takes for the process to finish. These files are small, but for large files aligning can be a lengthy process!!

3. When the computer has finished mapping, we want to see what the .sam file format looks like. We can easily view the bottom 50 lines of a file using the very useful bash command "tail". Type:

```
tail -50 FR32_ATCACG_before.sam > FR32_ATCACG_before_tail_50.txt
```

Then open the newly created file with a text editor to see the results.
To learn what the different column represent, see http://samtools.sourceforge.net/SAM1.pdf for details on the SAM file format.

4. Convert your .sam files to .bam, sort and remove duplicate reads.

From now on, we will work with the binary equivalent of the SAM file format: BAM. BAM files take up less place on a hard drive, and can be processed faster. Most SNP detection software are made to process BAM files. The drawback is that they cannot be examined directly in a text editor. Our first task is to remove

any duplicate reads from the alignments, for which we also need to sort our aligned reads by alignment position.

Identical, duplicate reads can be a result of biology and represent highly expressed transcripts. However, they are also quite likely to be an artifact of the PCR step in the sample preparation procedure. As we cannot distinguish between these two factors, it is prudent to remove duplicates from the dataset if we want to study gene expression differences. Artifactual duplicates can also skew genotype estimates so they must be identified also for SNP estimation.

Open the `convert_to_bam_and_dedup.sh` script in nano and make sure that in- and output files are correctly specified. The `convert_to_bam_and_dedup.sh` script has 2 elements: 1) It converts the .sam file to a binary bam file and sorts the reads within it. 2) It marks and removes duplicate reads using the `MarkDuplicates` program from the Picard package. Run the script with

```
sh ../scripts/convert_to_bam_and_dedup.sh
```

5. Count reads.

An interesting statistic of a mapping protocol is the number of reads that map to each contig. This data can also be used as a baseline for gene expression studies when dealing with RNA sequencing. For counting, we'll use a python script called countpression.py, which uses deduplicated SAM files as input, so the first step is a file conversion.

a. convert to SAM.

   We can use the `convert_bam_to_sam.sh` script to process all files at once. The script uses SAMtools to convert the file. First check that the input file names are correct in the script, then run it by typing:

```
sh ../scripts/convert_bam_to_sam.sh
```

b. Extract count data from the deduplicated .sam file from each individual using the countxpression.py script. Execute the script from the Terminal shell.

```
python ../scripts/countxpression.py 20 20
summarystats.txt *dedup.sam
```

The first argument is the mapping quality threshold, the second argument is a length threshold, summarystats.txt is an output file combining count statistics from all files. The last argument tells the program to count reads from all .sam files in the folder.

c. Take a look at the summary stats output. What fraction of your reads mapped to only one place in the assembly (column called PropQualAligned)? This number can be an indication of the assembly quality, and also if there are contaminant sequences in your fastq files.

d. Study the “counts.txt” file from an individual. The second column of data is the number of unique reads mapped to each contig, and could be used for gene expression analysis or just to assess what proportion of the contigs have been sequenced to a certain depth.

6. As a last step for today, we can view the results of the alignment using a genome browser. IGV (Integrative Genomics Viewer) is an excellent java-based such, which is freely available from the Broad Institute. Unfortunately, it is a bit tricky to get it to work remotely, so you will have to copy the assembly file (./assembly/Trinity.fasta) and the deduplicated SAM alignment file of interest (just choose one) to your local computer using scp or with graphical ssh software such as mobiXterm or Cyberduck.

   Once you have gotten IGV started, load in the assembly (Genomes → Load genome from file), then load in the deduplicated SAM file (File → Load from file…). When it asks you to create an index, click on OK. Now you can browse the contigs in the assembly, see where the reads have aligned, and where there are sequencing errors or potential SNPs.