RNASeq quickstart - from sequences to alignments to visualization in IGB

Introduction

High-throughput sequencing of cDNA, also called RNA-Seq, can provide many levels of information about gene expression, such as information about previously unannotated genes, expression of pseudogenes, differential expression both within and across samples, and alternative splicing.

Making the libraries and sending them out for sequencing is only the first step in performing an RNA-Seq experiment. What most people find is that processing, analyzing, and interpreting the data can often be just as time-consuming.

Take heart. These data are sets are so rich that you may never fully exhaust their potential. However, there are a few first steps you'll want to perform right away, as well as some quality control steps that will help you assess how well your experiment worked.

What follows is a description of RNA-Seq processing steps we do fairly routinely in the Loraine lab, along with links to software programs we've developed for in-house use. Please be aware that these programs are very much works in progress and so may not always work as advertised. If you find bugs or inconsistencies, please let us know - contact Ann (aloraine@uncc.edu) with feedback, suggestions, and bug reports.

RNA-Seq tutorial

The following protocol describes processing data from Illumina HiSeq pipeline. Whether you should perform exactly these steps with your data sets will depend on your data - its age, whether you've done single-end or paired-end sequencing, the read lengths, your reference genome, and so on. For example, when you run TopHat, you may need to adjust parameters to accommodate smaller read lengths if you are running data from pre-HiSeq instruments.

Check sequence quality.

Your first step upon downloading your reads will be to check the quality of your sequence. One of the easiest to use tools we (in the Loraine lab) have found for quality checking is a terrific program called FastQC, from the Babraham Institute. You can run it interactively or as a command line tool, and it's written in Java, which means you can run it on a Mac, a Linux machine, or a Windows computer. Like IGB, any computer that supports Java can run FastQC.

What you should be looking for in your data is evidence of poor sequencing quality as well as spots in your sequence where you have lots and lots of N characters - these usually correspond to place with poor quality. If your reads are generally low quality, you should ask your sequencing facility to try again.

If your data have many, many over-represented sequences (something FastQC can tell you), then you may want to make another library for that sample.

In our experience, low yields and low quality generally happen because something went wrong with sequence. Over-represented sequences and so-called PCR duplicates (many copies of the same sequence) are usually due to problems in library construction.

Align your sequence.

Let's assume that (happily) you have good-quality sequence. Your next step should be to align your sequences onto a reference genome or refernece transcriptome. Indeed, even if you haven't got high-quality sequence, you should still try to align it, because the alignments can tell you a lot about what went wrong.

For this tutorial, we'll use a spliced alignment tool to align the RNA-Seq reads onto a reference genome.

For RNA-Seq data sets, we mainly have used TopHat, from the University of Maryland. Others are available, but since we have the most experience with TopHat and seems to be one of the more widely used programs, the following examples will demonstrate how to run it.

TopHat is a spliced alignment tool that first runs BowTie (a non-spliced alignment tool from the same group) and then attempts to align any reads BowTie couldn't align by splitting them across putative introns. For this reason, to run TopHat you'll have to install BowTie. You'll also have to install samtools, a program that TopHat and BowTie use to generate alignment files called "BAM" (binary alignment) files. IGB can display data from BAM files, once you've created an index for them. More on indexing BAM files will come later.

Many different versions of TopHat have been released over the past couple of years and each behaves slightly differently. However, a few things seem to remain stable. First, TopHat will typically report multiple alignments for some number of reads. This is to be expected. However, depending on your experimental goals, you may want to focus on the reads that map exactly once onto the genome. You can figure out which reads mapped to multiple locations by looking at the "NH" flag in each alignment. Also, you should determine the minimum and maximum intron sizes for your genome and provide these as parameters to TopHat. For details on running TopHat, see the TopHat Manual.

Here is an example invocation of TopHat, fine-tuned for *Arabidopsis thaliana*. TopHat is a command line program, which means you run it by typing commands into a Unix terminal. (On Mac, this is the Terminal program from the Applications Utilities.)

For the rest of this tutorial, wherever you see a line that starts with a "$" sign, this means you type the command into a terminal. (The "$" means: the Unix prompt.)

```
$ tophat2 --max-intron-length 2000 /data/bowtieindex/A_thaliana_Jun_2009 Sample.fastq -o Sample
```

However, you're probably better off running TopHat on a fairly powerful machine. If you can find a multi-processor server with 16 or more of RAM, I would strongly recommend using it to run the alignment step. If you do get access to a multi-processor server, you should tell TopHat to take advantage of the extra processing power using the -p parameter that allows you to specify the number of processors TopHat can use.

To run bowtie2, you have to first make an index file for your genome. And to make an index for your genome, you'll need a fasta file for your genome. This is easy to get, however. You can either download the fasta files directly from the genome data provider (e.g., UCSC or NBCI) or you can get a 2bit file from the IGBQuickLoad Web site and convert it to fasta.

To get a sequence data file from IGBQuickLoad.org, go to the genome directory and download the "2bit" file you find there. Then use Jim Kent's 2bit2
In this example, I'll run bowtie2-build using a fasta file A_thaliana_Jun_2009.fa to create index files bowtie2 uses to speed up the alignment process. We TopHat will created a directory called Sample (the -o parameter).

In IGB, we use files in "2bit" format to represent genomic sequence data. It's easy to convert a 2bit file into a fasta file and back again.

To create a fasta file, do something like:

```
twoBitToFa A_thaliana_Jun_2009.2bit A_thaliana_Jun_2009.fa
```

Please note that you need to make sure that the names of chromosomes in the fasta file (and hence the BowTie index files) match with what IGB uses. For Arabidopsis, chromosome names are chr1, chr2, chr3, chr4, chr5, chrC, and chrM.

IGB has a synonyms system that allows it to "understand" that chr1 and Chr1 are really the same thing, but other programs you will run into might not be able to match names in this way. For this reason, it's a good idea to use the same names for chromosome throughout all the different steps of processing data.

Process output files

Index (and rename) your alignment file

When TopHat finishes, it will have created a file called "accepted_hits.bam" that contains your alignments. Your next step in the process will be to index this file, creating what's called a "bai" index file. For this, you'll use samtools, a freely available command-line tool with many useful functions. You can learn about samtools at this Web site and download the code from this page hosted at sourceforge.net.

Why make an index? This is a crucial step because it allows programs like IGB to do what is called a "region-based query." That is, once you've started IGB, you can load the BAM file, zoom in on a region, and then ask IGB to load in just the reads that overlap your region of interest. This works because IGB reads the index file, which tells IGB exactly where to look in the larger BAM file (which may be 1 gig or more!) for just the subset of reads that are relevant to the region in view.

To make an index (.bai) file for a BAM file, you typically would first have to sort the BAM file first.

However, the "accepted_hits.bam" file from TopHat should already be sorted. So all you should need to do next is rename it and make an index for it, like so:

```
$ mv accepted_hits.bam Sample.bam
$ samtools index Sample.bam
```

which will create an index file named Sample.bam.bai.

Make a bedgraph (wiggle) genome coverage file

A genome coverage (or depth) file reports the number of reads overlapping regions or individual bases of the reference genome. Older versions of TopHat used to create a coverage file, but more recent versions of Tophat no longer do this. However, you can create coverage graphs using `bedtools` genomecov command. For information on where to get `bedtools` and how to install it, visit the bedtools documentation.

To create a genome coverage bedgraph file from your BAM alignments file on a Linux machine, do this:

```
$ genomecov -ibam Sample.bam -split -bg > Sample.bedgraph
```

What should happen next is that a new file should appear named `Sample.bedgraph`, which will have a structure that looks a bit like:

```
chr1    3667    3697    2
chr1    3697    3707    3
chr1    3707    3726    5
chr1    3726    3742    6
chr1    3742    3744    4
chr1    3744    3749    5
chr1    3749    3755    6
```

At this point, you should be able to open the "bedgraph" file directly in IGB. However, IGB will work better if you sort the file, compress it using bgzip, and then make an index for the file using tabix.

```
$ sort -k1,1 -k2,2n Sample.bedgraph | bgzip > Sample.bedgraph.gz
$ tabix -s 1 -b 2 -e 3 Sample.bedgraph.gz
```

The sort command will read the entire `Sample.bedgraph` file into memory, sort the data, and then output the data in sorted order. The sorted data will first be ordered by chromosome name (field 1) and then by interval start position (field 2). In the command above, the sorted data are piped into the bgzip compression tool and then the compressed, sorted data are saved to a file called `Sample.bedgraph.gz`. The `tabix` command then creates an index file (extension `.tbi`) for the newly compressed, sorted `Sample.bedgraph.gz` file.

Note: To view `Sample.bedgraph.gz` file in IGB, you should always keep the index file (named `Sample.bedgraph.gz.tbi`) in the same folder.

To get a copy of `bgzip` and `tabix`, go to tabix download at SourceForge. Note that development of tabix has moved to this repository at github.

View data in IGB

Once you've created the files, you should then be able to open them in IGB. However, be sure to keep the index files (.bai from samtools and .tbi from tabix) in the same folder with their corresponding alignments or bedgraph files.

Once you open them in IGB, scroll and zoom to a smallist region of your genome -- a region where you can see maybe five or six different genes. (Use the "search" tab to zoom in on a particular gene of interest.)

Then, click "Load Data" to load data from the files you selected. (All selected files should appear in the **Data Management Table** on the right side of the **Data Access** tab.)

If all goes well, the bedgraph file data should appear in a graph track. To change its appearance, e.g., add a y-axis, change the color, etc. select the track label and click the Graph Adjuster tab, which contains various controls for working with graphs.

The reads (from the BAM) file appear in tracks above and below the axis. To change how reads look, choose File->Preferences->Tracks and use the options you see there to merge all the reads into a single +/- track, use color to indicate strand, change the number of reads that are shown in stacks, and so on.

For more information on working with tracks and graphs, please see the IGB User's Guide.

And remember, if you have questions, get in touch with us or use Google. All IGB tutorials and user guide materials are publicly available on our wiki and have (mostly) been indexed in Google.