Standard operating procedure for performing gene expression analysis with RNA-Seq data

Introduction

This document briefly outlines the essential steps in the process of analyzing gene expression data using RNA sequencing (mRNA, specifically). and recommends commonly used tools and techniques for this purpose. It is assumed in this document that the differential expression is being assessed between 2 experimental conditions, i.e. a simple 1:1 comparison, with some information about analyzing data from complex experimental designs. The focus of the SOP is on single-end strand-specific reads, however special measures to be taken for analysis of paired-end data are also briefly discussed. The recommended coverage for RNA-Seq on human samples is 30-50 million reads (single-end).

The procedures outlined below are recommendations to the H3ABioNet groups planning to do differential gene expression analysis on human RNA-Seq data, and are not meant to be prescriptive. Our goal is to help the groups set up their procedures and workflows, and to provide an overview of the main steps involved and the tools that can be used to implement them.

Glossary of associated terms and jargon

- FASTQ format & quality scores
    - FASTQ format is the standard format of raw sequence data
    - Quality scores assigned in the FASTQ files represent the probability that a certain base was called incorrectly. These scores are encoded in various ways and it is important to know the type of encoding for a given fastq file.
- Single-end vs paired-end, read vs fragment
    - DNA fragments can be sequenced from one end or both ends, single-end (SE) or paired-end (PE) respectively. During data processing PE data will often be represented as **fragments** consisting of 2 reads, instead of 2 separate reads.
- Strandedness
    - During library prep RNA can be processed into cDNA such the strand information is maintained. This is important in regions where there are overlapping genes on the two DNA strands
- RPKM, FPKM, TPM
    - RPKM - Reads Per Kilobase per Million mapped reads (for SE data)
    - FPKM - Fragments Per Kilobase per Million mapped reads (for PE data)
    - TPM - Transcripts Per Million
- TMM
    - This is a type of normalization and is an acronym for “Trimmed Mean of Ms” (Robinson and Oshlack, Genome Biology, 2010).

Procedural steps

This protocol paper (Trapnell *et al.* 2012) is a very good resource for understanding the procedural steps involved in any RNA-Seq analysis. The datasets they use in that paper are freely available, but the source of RNA was the fruitfly *Drosophila melanogaster*, and not Human tissue. In addition, they exclusively use the "tuxedo" suite developed in their group.

Alternatives to their tools are suggested below.

Figure 1. Steps in the Workflow.

Phase 1: Preprocessing of the raw reads

The following steps prepare reads for analysis and should be always performed prior to alignment.

Step 1.1*: Quality check*

The overall quality of the sequence information received from the sequencing center will determine how the quality trimming should be set up in Step 1.2. Tools like FastQC will enable the collection of this information. Sequencing facilities usually produce read files in fastq format, which contain a base sequence and a quality score for each base in a read. FastQC measures several metrics associated with the raw sequence data in the fastq file, including read length, average quality score at each sequenced base, GC content, presence of any overrepresented sequences (k-mers), and so on. The key metric to watch for is the graph representing the average quality scores (see figure 2), and the range of scores at each base along the length of the reads (reads are usually the same length at this time, and this length is the X-axis, the Y-axis has the quality scores). Note that for large projects, you may collate all of the FastQC reports by using MultiQC. MultiQC will generate an html file that visually summarizes these metrics across all samples, as well as provide tab-delimited files containing all the FastQC stats.
Note: FastQC has very stringent criteria to assess whether the data "Pass" or "Fail" for a given metric it measures, so even if it looks like your data has "failed" with respect to a given metric, please read carefully about the criteria employed. In most situations a "failed" reading for multiple metrics is not a death sentence for the dataset.

Figure 2. Graphs generated by FastQC detailing the average quality scores across all reads at each base.

Step 1.2*: Adaptor and Quality trimming + Removal of very short reads*

In this step we deal with **3 major preprocessing steps** that clean up the data and reduce noise in the overall analysis.

1. Adaptors (glossary term) are pieces of DNA introduced prior to sequencing to ensure that the DNA fragment being sequenced attaches to the sequencing flow cell. Usually these adaptors get sequenced, and have already been removed from the reads. But sometimes bits of adaptors are left

behind, anywhere from 90% to 20% of the adaptor length. These need to be removed from the reads. The adaptor sequence for this step will have to be obtained from the same source as the sequence data.

2. Frequently, the quality of bases sequenced tends to drop off toward one end of the read. A low quality base means that the the nucleotide assigned has a higher probability of being incorrect. It is best to trim off any low quality bases at the ends of reads to ensure the best alignment to the reference. Usually a quality score of <25 is considered as a "poor" quality score.
3. Once the adaptor remnants and low quality ends have been trimmed, some reads may end up being very short (i.e. <20 bases). These short reads are likely to align to multiple (wrong) locations on the reference, introducing noise. Hence any reads that are shorter than a predetermined cutoff (e.g. 20) need to be removed

One tool that deals with all of these issues at once is Trimmomatic. There are various other tools that can perform these 3 clean up steps one after the other, and they are listed below. For data that are paired ended, it is very important to perform the trimming for both read1 and read2 simultaneously. This is because all downstream applications expect paired information, and if one of the 2 reads is lost because it is too short, then the other read becomes unpaired (orphaned) and cannot be used properly for most applications. Trimmomatic has 2 modes, one for single end data (SE) and another one for paired end data (PE). If using paired end reads, please be sure to use the PE mode with both read1 and read2 fastq files for the same run.

Alternative Tools:

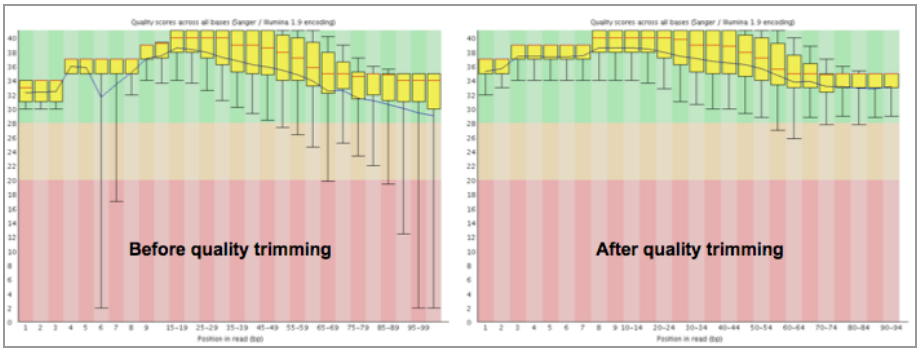
For adaptor trimming - Flexbar (Dodt *et al.* 2012) and one of the many tools listed here.
For trimming low quality bases from the ends of reads - FASTX-Toolkit (fastq_quality_filter), PrinSeq, SolexaQA (Cox *et al.* 2010).
For removing very short reads - PrinSeq

***Step 1.3**: Quality recheck*

Once the trimming step is complete, it is always good practice to make sure that your dataset looks better by rerunning FastQC on the trimmed data. The metrics to compare between trimmed and raw fastq data, in the context of the tool FastQC are listed below:

- Sequence length and Sequence length distribution (the minimum will be lower in trimmed data)
- The quality score graph (majority of the bases will have a minimum quality value at or above 30, see Figure 3 below)
- Per base sequence content (should remain about the same)

Figure 3. Graphs generated by FastQC detailing the change in average base quality across all reads after trimming in an example dataset.



Phase 2: Determining how many read counts are associated with known genes

Step 2.1*: Alignment*

Once the data are cleaned up, the next step is alignment to the reference genome. There are various tools available for this step, but it is important that the alignment tool chosen here is a "splice-aware" tool. That means that the tool should have the capability to align reads that contain exonic sequences from 2 exons on either side of one intron (also called intron-spanning reads). STAR, HISAT2, GSNAP, SOAPSplice are some of the many splice-aware aligners available. Note that TopHat used to be a very commonly-used tool from the Tuxedo suite, however this software is no longer supported and has been superseded by HISAT2 (recommended for human data).

When performing alignments it is imperative to set up the parameters properly to ensure the best alignment. Irrespective of the aligner used, it needs the following information:

- Are the data made up of single-end reads or paired-end reads?
- Are the data stranded, if so, was the standard dUTP method employed (STAR can detect this automatically)?

The other information that should be provided when setting up the alignment is the gene annotation information, a **gtf** or **gff3** file that contains the information about the location of all the genes in the context of the reference genome (a **fasta** file). It is very important to pick the gene annotation file that corresponds to the reference genome, i.e. the same version number and from the same source (Ensembl, UCSC or NCBI).

Note, that most all aligners require an index file to be created from the reference genome (a **fasta** file). This helps speed up alignment drastically. For STAR, this index can be created using the "genomeGenerate" mode with or without an annotation file.

Once the alignment is complete, the final result will be a file in **sam** or **bam** format. For a STAR run, the main alignment output will end in "Aligned.out.sam" by default, but it may also be returned in a sorted or unsorted **bam** file. In addition, there are two log files that are returned from STAR that report the progress of the run and save a summary of the final results. There is also a splice junctions (SJ) file that details high confidence splice junctions.

Once the alignment is completed, the first step is to check how many reads aligned to the genome. For STAR, all of these details can be found in the "Log.final.out" file. For RNA-Seq on Human samples, for good quality data, about 70 - 90% of the reads should match somewhere on the genome. If the data in question are of good quality but < 60 % of the reads are mapping to the genome, it is worth evaluating the parameters, and testing unmapped reads for presence of potential contaminants.

Step 2.2*: Count generation*

Once it is determined that the alignment step was successful, the next step is to enumerate the number of reads that are associated with the genes. There are multiple tools to perform this step (e.g. HTSeq's htseq-count, Subread's featureCounts), in addition, there are statistical analysis tools that do not require this step (e.g. Cufflinks' Cuffdiff and Ballgown). Both scenarios will be tackled in Phase III.
We recommend using featureCounts to collect the raw gene count information; this tool will require the alignment file (**bam**), and the associated gene annotation file (**gtf**). Irrespective of the counting tool used, it needs the following information:

- Different sources have slightly different formats, so it is essential to specify how the counting needs to be performed, irrespective of the counting tool employed. Gene counts should be collected for each gene ("-g" set as gene_id, for featureCounts), and at the level of the exon ("-t" set as exon, for featureCounts).
- Are the data stranded, if so, was the standard dUTP method employed ("-s" set as 2 for reverse, using featureCounts)?

The final output of any of these programs is a tab-delimited file with gene names in column one and counts in the second column. featureCounts will return an additional file that ends in ".summary" that specifies the number of reads that did not map only to one gene, split into various categories. It is normal for the total sum of all the rows in this file to be higher than the number of aligned reads for a sample, because if one read maps to two locations, featureCounts will count it twice in the "Unassigned_MultiMapping" category.

Step 2.3*: Collecting and tabulating alignment stats*

For a given RNA-Seq run it is valuable to collect several stats related to the alignment and counting steps; this is an important step in the evaluation process. There are many tools that gather this information from the **sam** or **bam** output file, e.g samtools' flagstat, Picard's CollectAlignmentSummaryMetrics. However there are quirks with each reporting tool, hence it is recommended to collect this information as described in the table below.

Gather numbers for the following categories	Calculating or gathering the information	Tool generating the information
Total reads	"Total Sequences" in the "Basic Statistics" section	FastQC
Total reads after trimming	"Total Sequences" in the "Basic Statistics" section (FastQC) **OR** "Number of input reads" in "Log.final.out" file (STAR)	FastQC **OR** STAR (*prefix*_Log.final.out)
Unmapped reads	Take the "Number of input reads" and subtract "Uniquely mapped reads number", "Number of reads mapped to multiple loci", and "Number of reads mapped to too many loci"	STAR (*prefix*_Log.final.out)
Reads mapped to genome	Add "Uniquely mapped reads number", "Number of reads mapped to multiple loci", and "Number of reads mapped to too many loci"	STAR (*prefix*_Log.final.out)

Multiply mapped reads	Add "Number of reads mapped to multiple loci", and "Number of reads mapped to too many loci"	STAR (*prefix*_Log.final.out)
Reads mapped to genes	This number is listed as "Assigned" in ".summary" file	featureCounts (*prefix*.txt.summary)
Uniquely mapped reads without an associated gene	This number is listed as "Unassigned_NoFeatures" in ".summary" file	featureCounts (*prefix*.txt.summary)
Uniquely mapped reads with an ambiguous gene assignment	This number is listed as "Unassigned_Ambiguity" in ".summary" file	featureCounts (*prefix*.txt.summary)

You may also find it easier to collate all of the summary data from FastQC, STAR, and featureCounts by running MultiQC on all of the reports generated by these three programs. MultiQC will generate an html file that visually summarizes this data as well as tab-delimited files containing all the stats produced by these programs.

Phase 3: Statistical analysis

Step 3.1*: QC and outlier/batch detection*

Once the counts have been generated, it is good practice to do some QC checks in addition to the ones listed above and to cluster the samples to see if there are any outliers or batch effects. Batch effects are usually caused by obtaining or processing the samples in batches and can obscure detection of expression differences if not adjust for statistically.

Below are 2 examples of variation between samples plotted using the plotDensity function from the "affy" package in R (see note about R below). Whatever the shape of the distribution, ideally it will be about the same for all samples. One or two samples that are very different could be outliers, but if there are two or more distinct groups, see if they correspond to treatment groups or known or unknown batch effects.

There are other QC steps that are recommended, like simple hierarchical clustering (hclust + plot functions in base R) or Principle Component Analysis clustering (plotPCA function in the affy package). This will help validate the presence of outliers.

Note that while Cufflinks has facilities for count generation, normalization and statistical analysis, it does not have any internal methods for sample QC or clustering. Instead, the R statistical software and add-on packages from Bioconductor are an excellent way to handle all aspects of statistical analysis of RNA-Seq data. They are free and available for any computer platform, although the command line interface can have a steep learning curve. Learning how to use R is well worth the time investment as it is a general tool for any sort of data manipulation, statistical analyses and graphing needs.

Step 3.2*: Remove low count genes and normalize*

The QC investigations in Step 3.1 should be done on the output from htseq-count, which typically is the entire transcriptome (all genes). However, for most eukaryotic species, only ~40-60% of genes are expressed in any given cell, tissue or age and so a large proportion of the genes may contain 0 or only a few reads. It is recommended to remove these genes because they do not contain enough information to be statistically valid and it reduces the amount of multiple hypothesis testing (Step 3.3). A common practice is to require a minimum number of reads in a minimum number of samples. Often 1 CPM (count per million, relative to the total number of reads in the sample) is used as the minimum threshold but this may need to be adjusted up or down depending on the total number of reads per sample to get a threshold between 5 and 20 reads per gene.

The CPM adjustment is a minimal normalization that is necessary because of the normal large variation in total reads per sample. However, this assumes that the total number of reads *should* have been the same

for all samples. This assumption can often be violated by extremely high expression in a few genes in one treatment or if the DE genes are predominantly in one direction. The traditional FPKM normalization in Cufflinks' Cuffdiff cannot adjust for this and was one of the reasons FPKM was found to be inferior to other normalization methods (Dillies et al. 2012). The newest version of Cufflinks has the default normalization in Cuffdiff set to the geometric mean method of DESeq, which along with TMM were the top two normalization methods found by Dillies et al. 2012.

Which extra normalization, DESeq or TMM, to use in R depends on which package, DESeq2 or edgeR, you prefer to use in R for statistical analysis. Both use extra normalization methods that are comparable and adjust for moderate biases in the number and direction of gene expression changes. Both are based on negative binomial statistical modeling and were found to compare quite evenly. Both have functions for outputting normalized expression values for use in QC and downstream visualizations. These normalized expression values should be put through the same QC metrics in Step 3.1 to see if the extra normalization rectified any problems indicated. Severe outliers should be removed and the data re-normalized. Remaining batch effects should be adjusted for in the statistical model.

Step 3.3*: Statistics for differential expression*

The main goal of most RNA-Seq is detection of differential expression between two or more groups. This is done for thousands of genes with often only a few replicates per group, so a statistical method must be used. This field is under rapid development and various statistical methods have been proposed. The modeling approach used by Cuffdiff is quite sophisticated but it is limited to a simple 2-group comparison. It cannot handle paired-sample data, batch effects or more complex experimental designs. However, both DESeq2 and edgeR can handle paired samples, other batch effects and complex designs. The choice between the two can simply be made based on which one is easier to understand and implement for the user.

In any statistical test of differential expression between groups, the amount of change from group to group must be evaluated by how much variation there is among the replicates of a group and how many replicates were used. These three factors, the amount of change, the amount of variation and the number of replicates are combined by the statistical test into one "p-value" which assesses the amount of evidence for differential expression. The traditional p-value cutoff of 0.05 for significance means that if you were to randomly sample two sets of replicates from the sample population 100 times, only 5 times would the value of the test statistic be larger than what you did see. This is a reasonable threshold when you only test a single gene, but in RNA-Seq you are measuring and testing thousands of genes at the same time. Therefore, some sort of adjustment of the p-values needs to be done. The most popular method is the False Discovery Rate method (Benjamimi & Hochberg, 1991), which adjusts the p-values so that the entire set of genes with values less than 0.05 is expected to contain 5% false-positives. Depending on the goals of your experiment, the FDR p-value threshold can be raised to gain more true-positives at the expense of a higher false-positive rate. However, keep in mind that the FDR correction depends on how many genes have low p-values compared with the number expected to to have low p-values and the lack of any genes with reasonably low p-values does not mean that no genes were truly changing.

Alternative to Step 3.2 and Step 3.3: *Using Cufflinks for normalization and differential gene expression*

The Cufflinks package that is part of the Tuxedo suite is a very popular and a relatively simple alternative to an R-based analysis. Cufflinks is made up of several smaller modules that are implemented in a stepwise-manner. The confusingly named Cufflinks module within the package is a reference-based transcriptome assembler and is unnecessary for human datasets, since the genes on the human genome are very well annotated. Hence for datasets with a simple experimental design (e.g. experimental vs control), it is advisable to directly use the CuffDiff module within the Cufflinks package directly after the alignment step. This module performs statistical analysis to identify differentially expressed genes using the bam files generated by the alignment tool, and in the context of the annotation file (GFF or GTF).

A note about Galaxy

If it is desirable to perform all processing in Galaxy, it should not be a problem for smaller experiments with a 1:1 comparisons between samples. For experiments with a large number of samples, and also for complex comparisons (e.g. 2x2 factorial design), Galaxy is not recommended for the final analysis; instead we recommend using the command line interface. However, Galaxy can be used to test parameter settings on a subset of the data, prior to switching over to command line for the whole analysis. Another issue of note is that the latest versions of tools might not be available in Galaxy, so please make a note of the version number (this is excellent practice in any case).

Trimmomatic is not available within Galaxy, but several other trimming tools are available which work well for SE reads but not for PE reads. When trimming PE reads within Galaxy, seek out programs like Trim Galore or Sickle.

The following tools are readily available within Galaxy:

- FastQC,
- Tophat2 (Bowtie2),
- Cufflinks (Cuffdiff),
- Htseq-count,
- Trim Galore
- Sickle
- FastX toolkit
- FASTQ Trimmer

References

https://genome.ucsc.edu/ENCODE/protocols/dataStandards/RNA_standards_v1_2011_May.pdf

Appendix 1: Alphabetized list of recommended tools (Needs change)

- **Bioconductor**----------http://bioconductor.org/

Provides tools for the analysis and comprehension of high-throughput genomic data. Bioconductor uses the R statistical programming language, and is open source and open development.

Sequence trimming

Use trimLRPatterns for adaptor trimming. From http://manuals.bioinformatics.ucr.edu/home/ht-seq:

```
# Create sample sequences.
myseq <-DNAStringSet(c("CCCATGCAGACATAGTG", "CCCATGAACATAGATCC", "CCCGTACAGATCACGTG"));
names(myseq) <- letters[1:3];

# Remove the specified R/L-patterns. The number of maximum mismatches can be specified
# for each pattern individually. Indel matches can be specified with the arguments:
# with.Lindels and with.Rindels.
trimLRPatterns(Lpattern ="CCC", Rpattern="AGTG", subject=myseq, max.Lmismatch = 0.33,
               max.Rmismatch = 1)

# To remove partial adaptors, the number of mismatches for all possible partial matches
# can be specified by providing a numeric vector of length nchar(mypattern).
# The numbers specifiy the number of mismatches for each partial match.
# Negative numbers are used to prevent trimming of a minimum fragment length,
# e.g. most terminal nucleotides.
trimLRPatterns(Lpattern = "CCC", Rpattern="AGTG", subject=myseq,
               max.Lmismatch=c(0,0,0), max.Rmismatch=c(1,0,0))

# With the setting 'ranges=TRUE' one can retrieve the corresponding
# trimming coordinates.
trimLRPatterns(Lpattern = "CCC", Rpattern="AGTG", subject=myseq,
               max.Lmismatch=c(0,0,0), max.Rmismatch=c(1,0,0), ranges=T)
```

Bioconductor-Galaxy integration
http://www.bioconductor.org/packages/2.12/bioc/html/RGalaxy.html
Use Rgalaxy package to make bioconductor – or any R package – available on Galaxy

- **Bowtie2**----------http://bowtie-bio.sourceforge.net/bowtie2/index.shtml

An ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100s or 1,000s of characters to relatively long (e.g. mammalian) genomes. Utilized by TopHat for alignments.

Bowtie-Galaxy integration
http://wiki.galaxyproject.org/Admin/NGS Local Setup

- **FASTX-Toolkit**----------http://hannonlab.cshl.edu/fastx_toolkit/

A collection of command line tools for Short-Reads FASTA/FASTQ files preprocessing.

fastq_quality_filter

Filters sequences based on quality.
http://hannonlab.cshl.edu/fastx_toolkit/commandline.html#fastq_quality_filter_usage

```
fastq_quality_filter [-q N] [-p N] [-z] [-i INFILE] [-o OUTFILE]

[-q N]   =   Minimum quality score to keep.
[-p N]   =   Minimum percent of bases that must have [-q] quality.
[-z]     =   Compress output with GZIP.
[-i INFILE]   =   FASTA/Q input file. default is STDIN.
[-o OUTFILE]   =   FASTA/Q output file. default is STDOUT.
```

fastx-collapser

Collapses identical sequences in a FASTQ/A file into a single sequence).
http://hannonlab.cshl.edu/fastx_toolkit/commandline.html#fastx_collapser_usage

```
fastx_collapser [-i INFILE] [-o OUTFILE]
```

FASTX-Toolkit - Galaxy integration
Part of standard installation. http://hannonlab.cshl.edu/fastx_toolkit/galaxy.html

- **SAMtools**----------http://samtools.sourceforge.net/

SAM Tools provide various utilities for manipulating alignments in the SAM format, including sorting, merging, indexing and generating alignments in a per-position format.

SAMtools-Galaxy integration
General setup: http://wiki.galaxyproject.org/Admin/NGS Local Setup
Tutorial for general usage: https://docs.uabgrid.uab.edu/wiki/Galaxy_DNA-Seq_Tutorial