BioInfoSummer 2012
Introduction to Next Generation Sequencing
Hands-on Workshop

Sean McWilliam
Nathan S. Watson-Haigh
Stephen Bent
Ute Baumann

Adelaide, Australia 4th Dec. 2012

Contents

Workshop Information

Document Structure

We have provided you with an electronic copy of the workshop's hands-on tutorial documents. We have done this for two reasons: 1) you will have something to take away with you at the end of the workshop, and 2) you can save time (mis)typing commands on the command line by using copy-and-paste.

We advise you to use Acrobat Reader to view the PDF. This is because it properly supports some features we have implemented to ensure that copy-and-paste of commands works as expected. This includes the appropriate copy-and-paste of special characters like tilde and hyphens as well as skipping line numbers for easy copy-and-past of whole code blocks.

While you could fly through the hands-on sessions doing copy-and-paste you will learn more if you take the time, saved from not having to type all those commands, to understand what each command is doing!

The commands to enter at a terminal look something like this:

```
1 tophat --solexa-quals -g 2 --library-type fr-unstranded -j \
    annotation/Danio_rerio.Zv9.66.spliceSites -o tophat/ZV9_2cells \
    genome/ZV9 data/2cells_1.fastq data/2cells_2.fastq
```

The following icons are used in the margin, throughout the documentation to help you navigate around the document more easily:

 Important

 For reference

 Follow these steps

 Questions to answer

 Warning - STOP and read

 Bonus exercise for fast learners

 Advanced exercise for super-fast learners

Resources Used

We have provided you with an environment which contains all the tools and data you need for the duration of this workshop. However, we also provide details about the tools and data used by each module at the start of the respective module documentation.

Module: Data Quality

Primary Author(s):
Sonika Tyagi sonika.tyagi@agrf.org.au

Contributor(s):
Nathan S. Watson-Haigh nathan.watson-haigh@awri.com.au

Key Learning Outcomes

After completing this practical the trainee should be able to:

- Assess the overall quality of NGS sequence reads
- Visualise the quality, and other associated matrices, of reads to decide on filters and cutoffs for cleaning up data ready for downstream analysis
- Clean up and pre-process the sequences data for further analysis

Resources You'll be Using

Tools Used

FastQC
http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

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

Picard
http://picard.sourceforge.net/

Useful Links

FASTQ Encoding
http://en.wikipedia.org/wiki/FASTQ_format#Encoding

Introduction

Going on a blind date with your read set? For a better understanding of the consequences please check the data quality!

For the purpose of this tutorial we are focusing only on Illumina sequencing which uses 'sequence by synthesis' technology in a highly parallel fashion. Although Illumina high throughput sequencing provides highly accurate sequence data, several sequence artifacts, including base calling errors and small insertions/deletions, poor quality reads and primer/adapter contamination are quite common in the high throughput sequencing data. The primary errors are substitution errors. The error rates can vary from 0.5-2.0% with errors mainly rising in frequency at the 3' ends of reads.

One way to investigate sequence data quality is to visualize the quality scores and other metrics in a compact manner to get an idea about the quality of a read data set. Read data sets can be improved by post processing in different ways like trimming off low quality bases, cleaning up any sequencing adapters and removing PCR duplicates. We can also look at other statistics such as, sequence length distribution, base composition, sequence complexity, presence of ambiguous bases etc. to assess the overall quality of the data set.

Highly redundant coverage (>15X) of the genome can be used to correct sequencing errors in the reads before assembly and errors. Various k-mer based error correction methods exist but are beyond the scope of this tutorial.

Prepare the Environment

To investigate sequence data quality we will demonstrate tools called FastQC and FASTX-Toolkit. FastQC will process and present the reports in a visual manner. Based on the results, the sequence data can be processed using the FASTX-Toolkit. We will use one data set in this practical, which can be found in the QC directory on your desktop.

Open the Terminal and go to the directory where the data are stored:

```
cd ~/QC/
pwd
```

At any time, help can be displayed for FastQC using the following command:

```
fastqc -h
```

Quality Visualisation

We have a file for a good quality and bad quality statistics. FastQC generates results in the form of a zipped and unzipped directory for each input file.

Execute the following command on the two files:

```
1 fastqc -f fastq bad_example.fastq
2 fastqc -f fastq good_example.fastq
```

View the FastQC report file of the dab data using a web browser such as firefox.

```
1 firefox bad_example_fastqc/fastqc_report.html &
```

The report file will have a Basic Statistics table and various graphs and tables for different quality statistics. E.g.:

Table 1: FastQC Basic Statistics table

Filename	bad_example.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	40000
Filtered Sequences	0
Sequence length	100
%GC	48

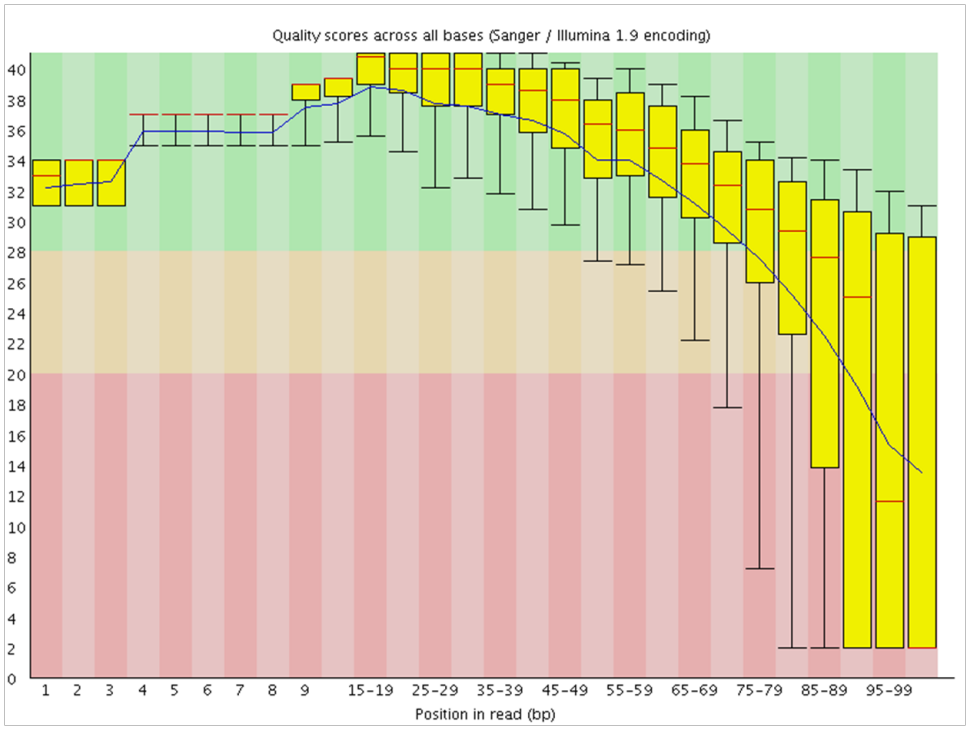

Figure 1: Per base sequence quality plot for `bad_example.fastq`.

A Phred quality score (or Q-score) expresses an error probability. In particular, it serves as a convenient and compact way to communicate very small error probabilities. The probability that base A is wrong ($P(\sim A)$) is expressed by a quality score, $Q(A)$, according to the relationship:

$$Q(A) = -10log10(P(\sim A))$$

The relationship between the quality score and error probability is demonstrated with the following table:

Table 2: Error probabilities associated with various quality (Q) values

Quality score, Q(A)	**Error probability, P(∼A)**	**Accuracy of the base call**
10	0.1	90%
20	0.01	99%
30	0.001	99.9%
40	0.0001	99.99%
50	0.00001	99.999%

How many sequences were there in your file? What is the read length?

Does the quality score values vary throughout the read length? (hint: look at the 'per base sequence quality plot')

What is the quality score range you see?

At around which position do the scores start falling below Q20?

How can we trim the reads to filter out the low quality data?

Good Quality Data

View the FastQC report files `fastqc_report.html` to see examples of a good quality data and compare the quality plot with that of the `bad_example_fastqc`.

```
1 firefox good_example_fastqc/fastqc_report.html &
```

Sequencing errors can complicate the downstream analysis, which normally requires that reads be aligned to each other (for genome assembly) or to a reference genome (for

detection of mutations). Sequence reads containing errors may lead to ambiguous paths in the assembly or improper gaps. In variant analysis projects sequence reads are aligned against the reference genome. The errors in the reads may lead to more mismatches than expected from mutations alone. But if these errors can be removed or corrected, the read alignments and hence the variant detection will improve. The assemblies will also improve after pre-processing the reads with errors.

Read Trimming

Read trimming can be done in a variety of different ways. Choose a method which best suits your data. Here we will use quality-based trimming but a fixed-length trimming example is provided in the appendix.

Quality Based Trimming

Base call quality scores can also be used to dynamically determine the trim points for each read. A quality score threshold and minimum read length following trimming can be used to remove low quality data.

Run the following commands to quality trim your bad data:

```
cd ~/QC
fastq_quality_trimmer -h
fastq_quality_trimmer -Q 33 -t 20 -l 50 -i bad_example.fastq -o \
    bad_example_quality_trimmed.fastq
```

Run FastQC on the quality trimmed file and visualise the quality scores.

```
fastqc -f fastq bad_example_quality_trimmed.fastq
firefox bad_example_quality_trimmed_fastqc/fastqc_report.html &
```

The output should look like:

Table 3: FastQC Basic Statistics table

Filename	bad_example_quality_trimmed.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	38976
Filtered Sequences	0
Sequence length	50-100
%GC	48

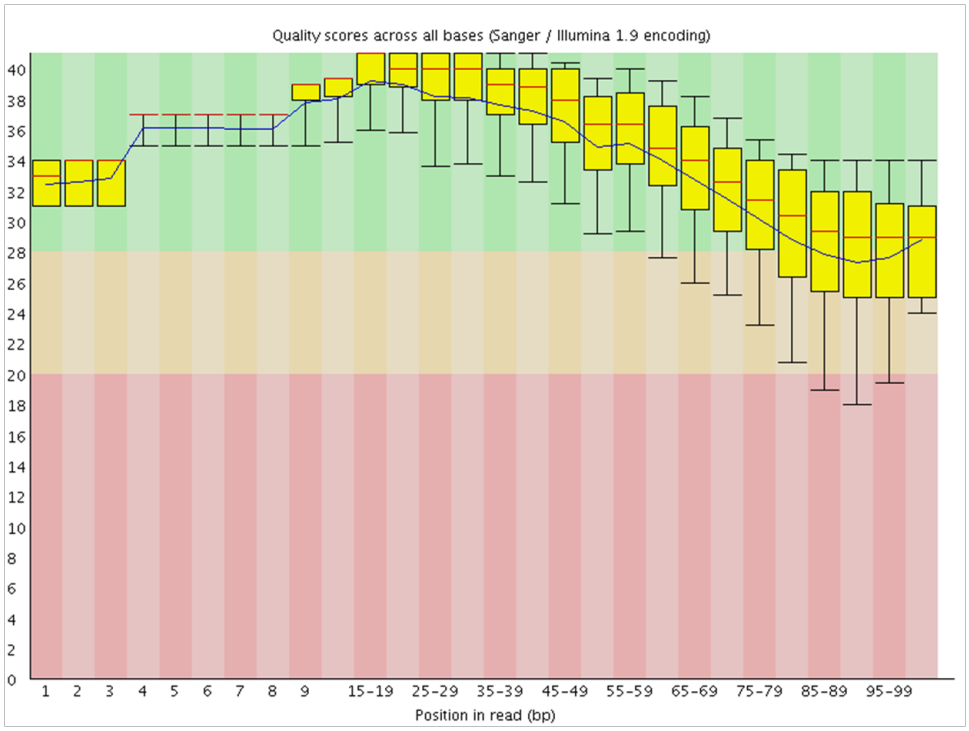

Figure 2: Per base sequence quality plot for the quality-trimmed `bad_example.fastq` reads.

How many sequences did you have following the trimming?

What was the quality score range following trimming?

How did the read lengths change following trimming?

Appendix

Fixed-Length Read Trimming

Low quality read ends can be trimmed using a fixed-length trimming. We will use the `fastx_trimmer` from the FASTX-Toolkit. Usage message to find out various options you can use with this tool. Type `fastx_trimmer -h` at anytime to display help.

We will no do fixed-length trimming of the `bad_example.fastq` file using the following command.

```
cd ~/QC
fastx_trimmer -h
fastx_trimmer -Q 33 -f 1 -l 80 -i bad_example.fastq -o \
    bad_example_trimmed01.fastq
```

We used the following options in the command above:

`-Q 33` Indicates the input quality scores are Phred+33 encoded

`-f` First base to be retained in the output

`-l` Last base to be retained in the output

`-i` Input FASTQ file name

`-o` Output file name

Run FastQC on the trimmed file and visualise the quality scores of the trimmed file.

```
fastqc -f fastq bad_example_trimmed01.fastq
firefox bad_example_trimmed01_fastqc/fastqc_report.html &
```

The output should look like:

Table 4: FastQC Basic Statistics table

Filename	bad_example_trimmed01.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	40000
Filtered Sequences	0
Sequence length	80
%GC	48

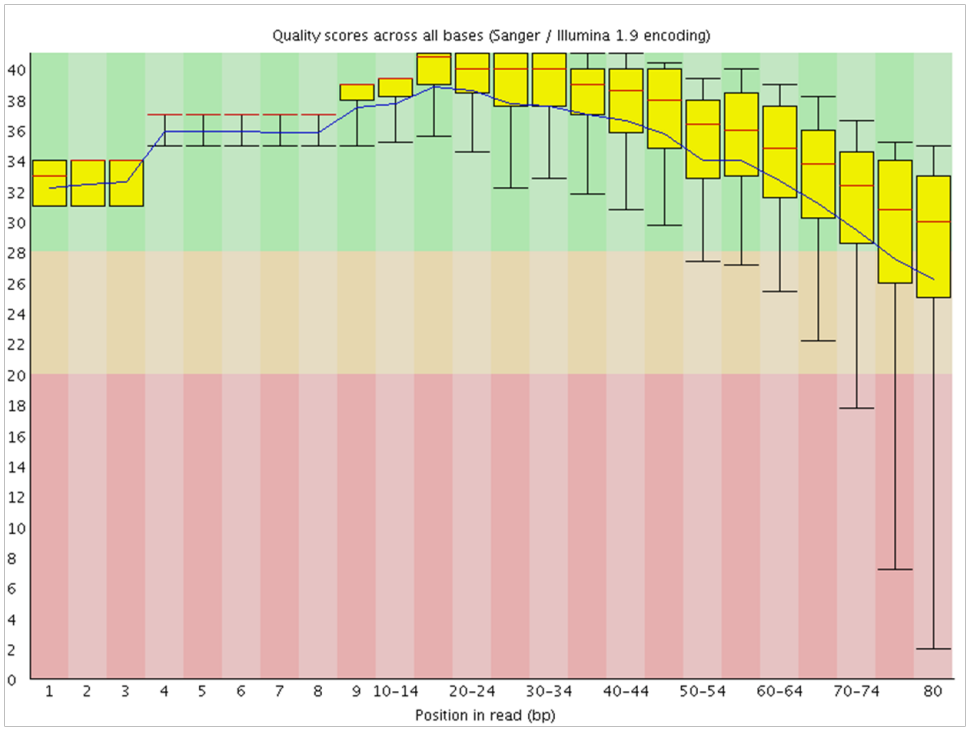

Figure 3: Per base sequence quality plot for the fixed-length trimmed `bad_example.fastq` reads.

What values would you use for `-f` if you wanted to trim off 10 bases at the 5' end of the reads?

Adapter Clipping

Sometimes sequence reads may end up getting the leftover of adapters and primers used in the sequencing process. It's good practice to screen your data for these possible contamination for more sensitive alignment and assembly based analysis.

This is particularly important when read lengths can be longer than the molecules being sequenced. For example when sequencing miRNAs.

Various QC tools are available to screen and/or clip these adapter/primer sequences from your data. (e.g. FastQC, FASTX-Toolkit, cutadapt).

Here we are demonstrating `fastx_clipper` to trim a given adapter sequence.

```
1 cd ~/QC
2 fastx_clipper -h
3 fastx_clipper -v -Q 33 -l 20 -M 15 -a GATCGGAAGAGCGGTTCAGCAGGAATGCCGAG \
    -i bad_example.fastq -o bad_example_clipped.fastq
```

An alternative tool, not installed on this system, for adapter clipping is `fastq-mcf`. A list of adapters is provided in a text file. For more information, see FastqMcf at http://code.google.com/p/ea-utils/wiki/FastqMcf.

Removing Duplicates

Duplicate reads are the ones having the same start and end coordinates. This may be the result of technical duplication (too many PCR cycles), or over-sequencing (very high fold coverage). It is very important to put the duplication level in context of your experiment. For example, duplication level in targeted or re-sequencing projects may mean something different in RNA-seq experiments. In RNA-seq experiments oversequencing is usually necessary when detecting low abundance transcripts.

The duplication level computed by FastQC is based on sequence identity at the end of reads. Another tool, Picard, determines duplicates based on identical start and end positions in SAM/BAM alignment files.

We will not cover Picard but provide the following for your information.

Picard is a suite of tools for performing many common tasks with SAM/BAM format files. For more information see the Picard website and information about the various

command-line tools available:

http://picard.sourceforge.net/command-line-overview.shtml

Picard 1.69 is installed on this system in `/usr/share/java/picard-1.69/`

One of the Picard tools (MarkDuplicates) can be used to analyse and remove duplicates from the raw sequence data. The input for Picard is a sorted alignment file in BAM format. Short read aligners such as, bowtie, BWA and tophat can be used to align FASTQ files against a reference genome to generate SAM/BAM alignment format.

Interested users can use the following general command to run the MarkDuplicates tool at their leisure. You only need to provide a BAM file for the INPUT argument (not provided):

```
1 cd ~/QC
2 java -jar /usr/share/java/picard/MarkDuplicates.jar \
    INPUT=<alignment_file.bam> VALIDATION_STRINGENCY=LENIENT \
    OUTPUT=alignment_file.dup METRICS_FILE=alignment_file.matric \
    ASSUME_SORTED=true REMOVE_DUPLICATES=true
```

Module: RNA-Seq

Primary Author(s):
Myrto Kostadima, EMBL-EBI kostadmi@ebi.ac.uk
Remco Loos, EMBL-EBI remco@ebi.ac.uk

Contributor(s):
Nathan S. Watson-Haigh nathan.watson-haigh@awri.com.au

Key Learning Outcomes

After completing this practical the trainee should be able to:

- Understand and perform a simple RNA-Seq analysis workflow.
- Perform gapped alignments to an indexed reference genome using TopHat.
- Perform transcript assembly using Cufflinks.
- Visualize transcript alignments and annotation in a genome browser such as IGV.
- Be able to identify differential gene expression between two experimental conditions.

Resources You'll be Using

Tools Used

Tophat
http://tophat.cbcb.umd.edu/

Cufflinks
http://cufflinks.cbcb.umd.edu/

Samtools
http://samtools.sourceforge.net/

BEDTools
http://code.google.com/p/bedtools/

UCSC tools
http://hgdownload.cse.ucsc.edu/admin/exe/

IGV
http://www.broadinstitute.org/igv/

DAVID Functional Analysis
http://david.abcc.ncifcrf.gov/

Sources of Data

http://www.ebi.ac.uk/ena/data/view/ERR022484
http://www.ebi.ac.uk/ena/data/view/ERR022485

Introduction

The goal of this hands-on session is to perform some basic tasks in the downstream analysis of RNA-seq data. We will start from RNA-seq data aligned to the zebrafish genome using Tophat.

We will perform transcriptome reconstruction using Cufflinks and we will compare the gene expression between two different conditions in order to identify differentially expressed genes.

Prepare the Environment

We will use a dataset derived from sequencing of mRNA from *Danio rerio* embryos in two different developmental stages. Sequencing was performed on the Illumina platform and generated 76bp paired-end sequence data using polyA selected RNA. Due to the time constraints of the practical we will only use a subset of the reads.

The data files are contained in the subdirectory called `data` and are the following:

`2cells_1.fastq` and `2cells_2.fastq`
: These files are based on RNA-seq data of a 2-cell zebrafish embryo

`6h_1.fastq` and `6h_2.fastq`
: These files are based on RNA-seq data of zebrafish embryos 6h post fertilization

Open the Terminal and go to the `RNA-seq` working directory:

```
1 cd ~/RNA-seq/
```

All commands entered into the terminal for this tutorial should be from within the **~/RNA-seq** directory.

Check that the `data` directory contains the above-mentioned files by typing:

```
1 ls data
```

Alignment

There are numerous tools for performing short read alignment and the choice of aligner should be carefully made according to the analysis goals/requirements. Here we will use Tophat, a widely used ultrafast aligner that performs spliced alignments.

Tophat is based on the Bowtie aligner and uses an indexed genome for the alignment to speed up the alignment and keep its memory footprint small. Created the index for the *Danio rerio* genome:

```
1 cd ~/RNA-seq
2 bowtie-build genome/Danio_rerio.Zv9.66.dna.fa genome/ZV9
```

Tophat has a number of parameters in order to perform the alignment. To view them all type:

```
1 tophat --help
```

The general format of the tophat command is:

```
1 tophat [options]* <index_base> <reads_1> <reads_2>
```

Where the last two arguments are the `.fastq` files of the paired end reads, and the argument before is the basename of the indexed genome.

The quality values in the FASTQ files used in this hands-on session are Phred+33 encoded. We explicitly tell tophat of this fact by using the command line argument `--solexa-quals`. See the appendix for further information about FASTQ encoding.

You can look at the first few reads in the file `data/2cells_1.fastq` with:

```
1 head -n 20 data/2cells_1.fastq
```

Some other parameters that we are going to use to run Tophat are listed below:

`-g` Maximum number of multihits allowed. Short reads are likely to map to more than one location in the genome even though these reads can have originated from only one of these regions. In RNA-seq we allow for a limited number of multihits, and in this case we ask Tophat to report only reads

that map at most onto 2 different loci.

`--library-type` Before performing any type of RNA-seq analysis you need to know a few things about the library preparation. Was it done using a strand-specific protocol or not? If yes, which strand? In our data the protocol was NOT strand specific.

`-j` Improve spliced alignment by providing Tophat with annotated splice junctions. Pre-existing genome annotation is an advantage when analysing RNA-seq data. This file contains the coordinates of annotated splice junctions from Ensembl. These are stored under the sub-directory `annotation` in a file called `ZV9.spliceSites`.

`-o` This specifies in which subdirectory Tophat should save the output files. Given that for every run the name of the output files is the same, we specify different directories for each run.

It takes some time (approx. 20 min) to perform tophat spliced alignments, even for this subset of reads. Therefore, we have pre-aligned the `2cells` data for you using the following command:

You DO NOT need to run this command yourself - we have done this for you.

```
1 tophat --solexa-quals -g 2 --library-type fr-unstranded -j \
    annotation/Danio_rerio.Zv9.66.spliceSites -o tophat/ZV9_2cells \
    genome/ZV9 data/2cells_1.fastq data/2cells_2.fastq
```

Align the `6h` data yourself using the following command:

```
1 # Takes approx. 20mins
2 tophat --solexa-quals -g 2 --library-type fr-unstranded -j \
    annotation/Danio_rerio.Zv9.66.spliceSites -o tophat/ZV9_6h \
    genome/ZV9 data/6h_1.fastq data/6h_2.fastq
```

The `6h` read alignment will take approx. 20 min to complete. Therefore, we'll take a look at some of the files, generated by tophat, for the pre-computed `2cells` data.

Alignment Visualisation in IGV

The Integrative Genomics Viewer (IGV) is able to provide a visualisation of read alignments given a reference sequence and a BAM file. We'll visualise the information contained in the `accepted_hits.bam` and `junctions.bed` files for the pre-computed `2cells` data. The former, contains the tophat sliced alignments of the reads to the reference while the latter stores the coordinates of the splice junctions present in the data set.

Open the `RNA-seq` directory on your Desktop and double-click the `tophat` subdirectory and then the `ZV9_2cells` directory.

1. Launch IGV by double-clicking the "IGV 2.1" icon on the Desktop (ignore any warnings that you may get as it opens). *NOTE: IGV may take several minutes to load for the first time, please be patient.*
2. Choose "Zebrafish (Zv9)" from the drop-down box in the top left of the IGV window.
3. Load the `accepted_hits.sorted.bam` file by clicking the "File" menu, selecting "Load from File" and navigating to the `Desktop/RNA-seq/tophat/ZV9_2cells` directory.
4. Rename the track by right-clicking on its name and choosing "Rename Track". Give it a meaningful name like "2cells BAM".
5. Load the `junctions.bed` from the same directory and rename the track "2cells Junctions BED".
6. Load the Ensembl annotations file `Danio_rerio.Zv9.66.gtf` stored in the `RNA-seq/annotation` directory.
7. Navigate to a region on chromosome 12 by typing `chr12:20,270,921-20,300,943` into the search box at the top of the IGV window.

Can you identify the splice junctions from the BAM file?

Are the junctions annotated for `CBY1` consistent with the annotation?

Are all annotated genes, from both RefSeq and Ensembl, expressed?

Once tophat finishes aligning the 6h data you will need to sort the alignments found in the BAM file and then index the sorted BAM file.

```
1 samtools sort tophat/ZV9_6h/accepted_hits.bam \
     tophat/ZV9_6h/accepted_hits.sorted
2 samtools index tophat/ZV9_6h/accepted_hits.sorted.bam
```

Load the sorted BAM file into IGV, as described previously, and rename the track appropriately.

Isoform Expression and Transcriptome Assembly

There are a number of tools that perform reconstruction of the transcriptome and for this workshop we are going to use Cufflinks. Cufflinks can do transcriptome assembly either *ab initio* or using a reference annotation. It also quantifies the isoform expression in Fragments Per Kilobase of exon per Million fragments mapped (FPKM).

Cufflinks has a number of parameters in order to perform transcriptome assembly and quantification. To view them all type:

```
1 cufflinks --help
```

We aim to reconstruct the transcriptome for both samples by using the Ensembl annotation both strictly and as a guide. In the first case Cufflinks will only report isoforms that are included in the annotation, while in the latter case it will report novel isoforms as well.

The Ensembl annotation for *Danio rerio* is available in `annotation/Danio_rerio.Zv9.66.gtf`.

The general format of the `cufflinks` command is:

```
1 cufflinks [options]* <aligned_reads.(sam|bam)>
```

Where the input is the aligned reads (either in SAM or BAM format).

Some of the available parameters for Cufflinks that we are going to use to run Cufflinks are listed below:

`-o` Output directory.

`-G` Tells Cufflinks to use the supplied GTF annotations strictly in order to estimate isoform annotation.

`-b` Instructs Cufflinks to run a bias detection and correction algorithm which can significantly improve accuracy of transcript abundance estimates. To do this Cufflinks requires a multi-fasta file with the genomic sequences against which we have aligned the reads.

`-u` Tells Cufflinks to do an initial estimation procedure to more accurately weight reads mapping to multiple locations in the genome (multi-hits).

`--library-type` Before performing any type of RNA-seq analysis you need to know a few things about the library preparation. Was it done using a strand-specific protocol or not? If yes, which strand? In our data the protocol was NOT strand specific.

Perform transcriptome assembly, strictly using the supplied GTF annotations, for the `2cells` and `6h` data using cufflinks:

```
# 2cells data (takes approx. 5mins):
cufflinks -o cufflinks/ZV9_2cells_gtf -G \
    annotation/Danio_rerio.Zv9.66.gtf -b \
    genome/Danio_rerio.Zv9.66.dna.fa -u --library-type fr-unstranded \
    tophat/ZV9_2cells/accepted_hits.bam
# 6h data (takes approx. 5mins):
cufflinks -o cufflinks/ZV9_6h_gtf -G annotation/Danio_rerio.Zv9.66.gtf \
    -b genome/Danio_rerio.Zv9.66.dna.fa -u --library-type fr-unstranded \
    tophat/ZV9_6h/accepted_hits.bam
```

Cufflinks generates several files in the specified output directory. Here's a short description of these files:

`genes.fpkm_tracking` Contains the estimated gene-level expression values.

`isoforms.fpkm_tracking` Contains the estimated isoform-level expression values.

`skipped.gtf` Contains loci skipped as a result of exceeding the maximum number of fragments.

`transcripts.gtf` This GTF file contains Cufflinks' assembled isoforms.

The complete documentation can be found at: http://cufflinks.cbcb.umd.edu/manual.html#cufflinks_output

So far we have forced cufflinks, by using the `-G` option, to strictly use the GTF annotations provided and thus novel transcripts will not be reported. We can get cufflinks to perform a GTF-guided transcriptome assembly by using the `-g` option instead. Thus, novel transcripts will be reported.

GTF-guided transcriptome assembly is more computationally intensive than strictly using the GTF annotations. Therefore, we have pre-computed these GTF-guided assemblies for you and have placed the results under subdirectories: `cufflinks/ZV9_2cells_gtf_guided` and `cufflinks/ZV9_6h_gft_guided`.

You DO NOT need to run these commands. We provide them so you know how we generated the the GTF-guided transcriptome assemblies:

```
# 2cells guided transcriptome assembly (takes approx. 30mins):
cufflinks -o cufflinks/ZV9_2cells_gtf_guided -g \
    annotation/Danio_rerio.Zv9.66.gtf -b \
    genome/Danio_rerio.Zv9.66.dna.fa -u --library-type fr-unstranded \
    tophat/ZV9_2cells/accepted_hits.bam
# 6h guided transcriptome assembly (takes approx. 30mins):
cufflinks -o cufflinks/ZV9_6h_gtf_guided -g \
    annotation/Danio_rerio.Zv9.66.gtf -b \
    genome/Danio_rerio.Zv9.66.dna.fa -u --library-type fr-unstranded \
    tophat/ZV9_6h/accepted_hits.bam
```

1. Go back to IGV and load the pre-computed, GTF-guided transcriptome assembly for the `2cells` data (`cufflinks/ZV9_2cells_gtf_guided/transcripts.gtf`).
2. Rename the track as "2cells GTF-Guided Transcripts".
3. In the search box type `ENSDART00000082297` in order for the browser to zoom in to the gene of interest.

Do you observe any difference between the Ensembl GTF annotations and the GTF-guided transcripts assembled by cufflinks (the "2cells GTF-Guided Transcripts" track)?

Differential Expression

One of the stand-alone tools that perform differential expression analysis is Cuffdiff. We use this tool to compare between two conditions; for example different conditions could be control and disease, or wild-type and mutant, or various developmental stages.

In our case we want to identify genes that are differentially expressed between two developmental stages; a `2cells` embryo and `6h` post fertilization.

The general format of the cuffdiff command is:

```
cuffdiff [options]* <transcripts.gtf> \
   <sample1_replicate1.sam[,...,sample1_replicateM]> \
   <sample2_replicate1.sam[,...,sample2_replicateM.sam]>
```

Where the input includes a `transcripts.gtf` file, which is an annotation file of the genome of interest, and the aligned reads (either in SAM or BAM format) for the conditions. Some of the Cufflinks options that we will use to run the program are:

`-o` Output directory.

`-L` Labels for the different conditions

`-T` Tells Cuffdiff that the reads are from a time series experiment.

`-b` Instructs Cufflinks to run a bias detection and correction algorithm which can significantly improve accuracy of transcript abundance estimates. To do this Cufflinks requires a multi-fasta file with the genomic sequences against which we have aligned the reads.

`-u` Tells Cufflinks to do an initial estimation procedure to more accurately weight reads mapping to multiple locations in the genome (multi-hits).

`--library-type` Before performing any type of RNA-seq analysis you need to know a few things about the library preparation. Was it done using a strand-specific protocol or not? If yes, which strand? In our data the protocol was NOT strand specific.

Run cuffdiff on the cufflinks generated BAM files for the 2cells vs. 6h data sets:

```
cuffdiff -o cuffdiff/ -L ZV9_2cells,ZV9_6h -T -b \
    genome/Danio_rerio.Zv9.66.dna.fa -u --library-type fr-unstranded \
    annotation/Danio_rerio.Zv9.66.gtf \
    tophat/ZV9_2cells/accepted_hits.bam tophat/ZV9_6h/accepted_hits.bam
```

We are interested in the differential expression at the gene level. The results are reported by Cuffdiff in the file `cuffdiff/gene_exp.diff`. Look at the first few lines of the file using the following command:

```
head -n 20 cuffdiff/gene_exp.diff
```

We would like to see which are the most significantly differentially expressed genes. Therefore we will sort the above file according to the q value (corrected p value for multiple testing). The result will be stored in a different file called `gene_exp_qval.sorted.diff`.

```
sort -t$'\t' -g -k 13 cuffdiff/gene_exp.diff > \
    cuffdiff/gene_exp_qval.sorted.diff
```

Look again at the first few lines of the sorted file by typing:

```
head -n 20 cuffdiff/gene_exp_qval.sorted.diff
```

Copy an Ensembl transcript identifier from the first two columns for one of these genes (e.g. `ENSDARG00000077178`). Now go back to the IGV browser and paste it in the search box.

Do you see any difference in the read coverage between the `2cells` and `6h` conditions that might have given rise to this transcript being called as differentially expressed?

Note that the coverage on the Ensembl browser is based on raw reads and no normalisation has taken place contrary to the FPKM values.

Appendix

Quality Value Encoding Schema

In order to use a single character to encode Phred qualities, ASCII characters are used (http://www.asciitable.com/). All ASCII characters have a decimal number associated with them but the first 32 characters and non-printable (e.g. backspace, shift, return, escape). Therefore, the first printable ASCII character is number 33, the exclamation mark (!). In Phred+33 encoded quality values the exclamation mark takes the Phred quality score of zero.

Early Solexa (now Illumina) sequencing needed to encode negative quality values. Because ASCII characters < 33 and non-printable, using the Phred+33 encoding was not possible. Therefore, they simply moved the offset from 33 to 64 thus inventing the Phred+64 encoded quality values. In this encoding a Phred quality of zero is denoted by the ASCII number 64 (the @ character). Since Illumina 1.8, quality values are now encoded using Phred+33.

FASTQ does not provide a way to describe what quality encoding is used for the quality values. Therefore, you should find this out from your sequencing provider. Alternatively, you may be able to figure this out by determining what ASCII characters are present in the FASTQ file. E.g the presence of numbers in the quality strings, can only mean the quality values are Phred+33 encoded. However, due to the overlapping nature of the Phred+33 and Phred+64 encoding schema it is not always possible to identify what encoding is in use. For example, if the only characters seen in the quality string are (@ABCDEFGHI), then it is impossible to know if you have really good Phred+33 encoded qualities or really bad Phred+64 encoded qualities.

For a grapical representation of the different ASCII characters used in the two encoding schema see: http://en.wikipedia.org/wiki/FASTQ_format#Encoding.

Functional Annotation of Differentially Expressed Genes

After you have performed the differential expression analysis you are interested in identifying if there is any functionality enrichment for your differentially expressed genes. On your Desktop click:

```
1 Applications >> Internet >> Firefox Web Browser
```

And go to the following URL: http://david.abcc.ncifcrf.gov/ On the left side click on Functional Annotation. Then click on the Upload tab. Under the section Choose from File, click Choose File and navigate to the `cuffdiff` directory. Select

the file called `globalDiffExprs_Genes_qval.01_top100.tab`. Under Step 2 select ENSEMBL_GENE_ID from the drop-down menu. Finally select Gene list and then press Submit List. Click on Gene Ontology and then click on the CHART button of the GOTERM_BP_ALL item.

Do these categories make sense given the samples we're studying?

Browse around DAVID website and check what other information are available.

References

1. Trapnell, C., Pachter, L. & Salzberg, S. L. TopHat: discovering splice junctions with RNA-Seq. Bioinformatics 25, 1105-1111 (2009).

2. Trapnell, C. et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. Nat. Biotechnol. 28, 511-515 (2010).

3. Langmead, B., Trapnell, C., Pop, M. & Salzberg, S. L. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol. 10, R25 (2009).

4. Roberts, A., Pimentel, H., Trapnell, C. & Pachter, L. Identification of novel transcripts in annotated genomes using RNA-Seq. Bioinformatics 27, 2325-2329 (2011).

5. Roberts, A., Trapnell, C., Donaghey, J., Rinn, J. L. & Pachter, L. Improving RNA-Seq expression estimates by correcting for fragment bias. Genome Biol. 12, R22 (2011).