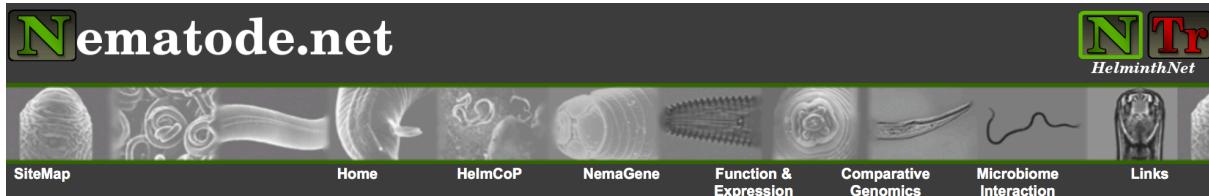


**N**ematode.net



SiteMap      Home      HelmCoP      NemaGene      Function & Expression      Comparative Genomics      Microbiome Interaction      Links

**N** **T**  
*HelminthNet*

**T**rematode.net



SiteMap      Home      TremaGene      Function & Expression      Comparative Genomics      Microbiome Interaction      Links

**N** **T**  
*HelminthNet*

# Bioinformatics Workshop for Helminth Genomics

September 10-11, 2015

## Sponsors:



## Table of contents – Curriculum

*September 10, 2015*

### Section 0: Cloud Computing

<b>Module 0 – Introduction to cloud computing using Amazon's AWS.....</b>	5
---	---

### Section 1: Genome

<b>Module 1 – Sequencing platforms.....</b>	8
---	---

- Common sequencing platforms
- Choosing appropriate sequencing platform
- Sequencer output(s)
- QC sequence output

<b>Module 2 – Sequence data files.....</b>	11
--	----

- Common sequencing file formats
- Convert between formats

<b>Module 3 – Analytical processing of sequences.....</b>	14
---	----

- Learn how to process genomic data to a cleaned state, ready for analysis

<b>Module 4 – Genome assembly.....</b>	20
--	----

- De novo genome assembly
- Assembly improvement
- QC de novo assemblies

<b>Module 5 – Genome annotation.....</b>	23
--	----

- Identify & mask repeats in an assembly
- De novo gene calling
- Assess gene annotations
- Improve gene annotations
- Common genome annotation formats

<b>Module 6 – Functional annotation.....</b>	29
--	----

- Assign basic functional annotation to predicted genes
- Similarity search on custom databases
- Common resources used for functional annotation

## Section 2: Transcriptome

<b>Module 0 – RNA isolation to sequence production.....</b>	33
▪ RNAseq data production, RNA isolation to sequencing	
▪ Analytical processing of RNAseq data to a cleaned state, ready for analysis	
<b>Module 1 – Genome based RNA-seq analyses.....</b>	44
▪ Align RNAseq data to a genome assembly	
▪ Visualizing alignments	
<b>Module 2 – <i>De novo</i> transcript assembly.....</b>	48
▪ Read normalization	
▪ <i>De novo</i> transcript assembly	
▪ Quality filtering of assembled transcripts	

September 11, 2015

<b>Module 3 – Expression and differential expression.....</b>	54
▪ Experimental design (biological replicates, time courses, stages, tissues, etc.)	
▪ PCA and hierarchical clustering	
▪ Analyze differential expression	
▪ Measure, interpret and visualize expression in MS Excel	
▪ Organize and mine a database of gene annotation	
▪ Functional enrichment of differentially expressed genes	

## Section 3: Variome

<b>Module 0 – Re-sequencing genomes.....</b>	91
▪ Re-sequencing genomes	
<b>Module 1 – Processing and alignment.....</b>	95
▪ Analytical processing and alignment of reads	
▪ Refining alignments for variant calling	
<b>Module 2 – Variant calling.....</b>	99
▪ Basics of variant calling & how to filter for high quality loci	
▪ Visualization of variants	
<b>Module 3 – Variant annotation.....</b>	108
▪ Variant annotation	
▪ Annotation interpretation	

## Section 4: Final topics

<b>Module 0 – Finding sequence resources.....</b>	111
▪ Locate and download annotated references	
▪ Finding Helminth resources	
<b>Module 1 – Bioinformatics packages.....</b>	111
▪ Popular Bioinformatics toolsets	
▪ Galaxy platform for running bioinformatics workflows	
<b>Module 2 – Sources of help for bioinformaticians.....</b>	112
▪ Get help with bioinformatics problems	
<b>Module 3 – Open discussion.....</b>	112
▪ Questions & Answers	

---

**Workshop organizer and facilitator:**

**Makedonka Mitreva, PhD**  
**Associate Professor of Medicine @ Washington University School of Medicine**

<http://genome.wustl.edu/people/individual/makedonka-mitreva/>

---

---

**Workshop instructors (Mitreva lab):**

**Section 1:**

John Martin

Philip Ozersky

**Section 2:**

Samantha McNulty, PhD

Bruce A. Rosa, PhD

**Section 3:**

Young-Jun Choi, PhD

Rahul Tyagi, PhD

---

## Section 0: Cloud Computing

### Module 0: Introduction to cloud computing using Amazon's AWS

#### Introduction

This document contains command lines & information that will be helpful to you during the workshop. Think of it as CliffsNotes for the material being presented. It will be helpful to have this document open electronically while following along with the demonstrations for copy-pasting the command lines.

Command lines will be printed in a distinctive font to make them stand out from other text.

In this module you will learn the basics of cloud computing, and a little bit about how we'll be taking advantage of Amazon's AWS. You'll connect to your EC2 instance and we'll show you how to transfer files to and from the EC2 instance. Finally we'll download some files that will be needed during later parts of the class.

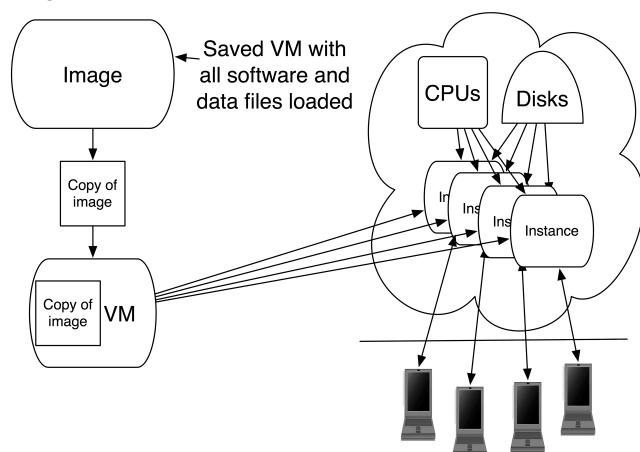
#### Cloud computing

This phrase refers to the use of remote hardware resources (CPU, memory, disk space, etc...) from a local platform (your laptop, a lab workstation, a desktop in your home, etc...). Cloud computing enables users with limited local resources to have access to powerful computer hardware, and only pay for the resources they actually consume. Some examples of commercially available cloud computing services are Amazon's AWS, Google's Compute Engine, Microsoft's Azure & IBM's Bluemix.

#### Virtual Machines

A Virtual Machine (VM) is a software emulation of a computer running an operating system. Because the processes occurring within a VM are entirely software, it's possible to 'save' a virtual machine's current state by taking a 'snapshot' of the running software and saving it to disk.

That snapshot is called an 'image'. This image can then be restarted at a later time to restore the exact state of the original VM. It's possible to launch multiple copies of an image, creating many 'instances' of that parent image, all running in the same state as the original VM when the snapshot was taken.



#### Workshop image and student instances

To prepare for this workshop we launched a VM on the Amazon EC2 service. We started with a pre-built image from Amazon that had Red Hat Enterprise Linux 7.1 already installed. We launched one instance of that basic starting image on the cloud and then loaded all the software and data files we wanted to provide for the course. Once everything was loaded and tested we took a snapshot and saved it as our workshop image. Before the workshop

starts on day 1, we'll be launching one instance for each attending student. Each student will then log into their own working instance in which they will follow along with the demonstrations we'll be showing.

#### Connecting to a running EC2 instance

Here is how to connect to your personal EC2 instance. You should have received an email from which you can find your specific instance address. You should also have received the public key file as an attachment to that email. You'll need to save the key to your laptop and remember its location. We suggest creating a WORKSHOP folder on your laptop's desktop, and then saving the key in that location. There will be several parts of the workshop that will involve downloading data and running tools locally, on your laptop. For convenience we suggest keeping all workshop related files inside this WORKSHOP folder on your desktop.

Launch your terminal program (MAC users can use 'Terminal', and WIN users can use MobaXTerm). Then from inside that terminal run these commands

**0.0.1 (MAC)** : cd /Users/<your\_username>/Desktop  
**0.0.1 (WIN)** : cd /home/mobaxterm/Desktop

**0.0.2**: mkdir WORKSHOP  
**0.0.3**: cd WORKSHOP

Save the PublicKey150713.pem file into this location now, then back in the terminal, change the permission on the public key file

**0.0.4**: chmod 400 PublicKey150713.pem

**0.0.5**: ssh -YC -c blowfish-cbc,arcfour -i PublicKey150713.pem ec2-user@<instance address>  
\*note: Replace "<instance address>" with your personal instance address that was emailed to you

A special note for laptop users without access to mouse buttons, which will be needed to copy-paste commands for many of the demonstrations. Here is how to enable 'three button mouse' clicking on a Mac

**0.0.6 (MAC)** : pull XQuartz to the front  
**0.0.7 (MAC)** : Open the X11 menu and select 'Preferences'  
**0.0.8 (MAC)** : Check the 'Emulate three button mouse' box

For Windows users, here is how to enable copy-paste via the right-click menu

**0.0.6 (WIN)** : Select 'configuration' from the settings menu  
**0.0.7 (WIN)** : Select the X11 tab  
**0.0.8 (WIN)** : Set the Clipboard setting to 'enabled'  
\*note: This will either be a drop menu or a checkbox for 'Shared clipboard' that you need to check

## Uploading and Downloading from an EC2 instance

Here is how to copy files from your laptop to the EC2 instance

```
scp -i <path to public key> <file on laptop> ec2-user@<instance address>:<path to destination on EC2>
```

And here is how to copy files from the EC2 instance to your laptop

```
scp -i <path to public key> ec2-user@<instance address>:<path to file on EC2> <path to destination on laptop>
```

You can also use scp to copy entire directories and their contents using the **-r** (recursive) argument:

```
scp -r -i <path to public key> ec2-user@<instance address>:<path to directory on EC2> <path to destination on laptop>
```

We have a couple of compressed tar files containing data files & resources you'll need for later sections of this class that need to be downloaded to your laptop. First open a 2<sup>nd</sup> terminal window on your laptop, because we conveniently have the full address of the EC2 instance, its easiest to do these copies while sitting on the laptop side of the cloud

**0.0.9 (MAC)** : cd /Users/<your\_username>/Desktop/WORKSHOP  
**0.0.9 (WIN)** : cd /home/mobaxterm/Desktop/WORKSHOP

Then use scp to download the wanted data files

**0.0.10:** scp -I PublicKey150713.pem ec2-

user@<instance\_address>:~/WORKSHOP\_RESOURCES/SECTION\_2.tgz .

**0.0.11:** scp -I PublicKey150713.pem ec2-

user@<instance\_address>:~/WORKSHOP\_RESOURCES/SECTION\_3.tgz .

*Useful information:*

(Amazon's documentation on EC2) <https://aws.amazon.com/ec2/>

---

## Section 1: Genome

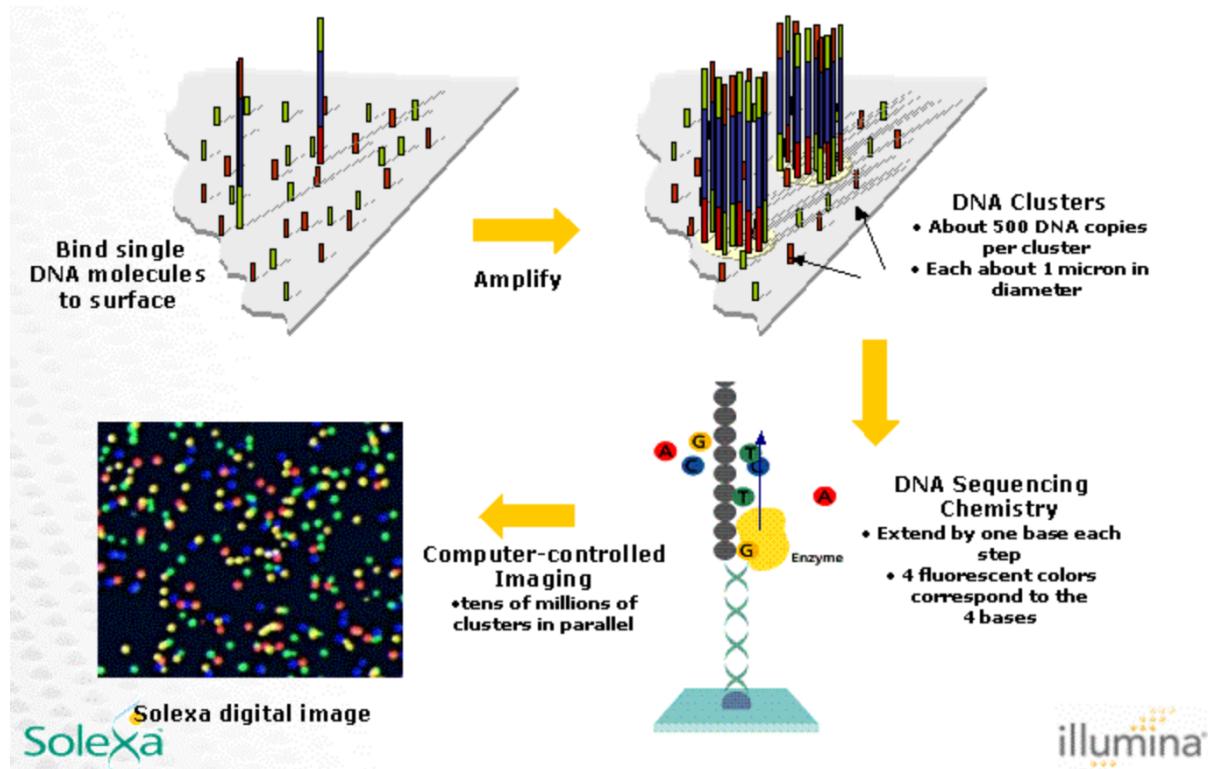
### Module 1: Sequencing platforms

---

In this module, we'll introduce you to several of the sequencing platforms in use at our center and we'll look into what makes these systems unique, and what traits may be important to you when you are deciding which platform(s) to use for your project. We'll also talk about the specific case of the data we'll be using during the genomic section of this workshop. We'll describe the format in which it comes off the sequencing machine, and we'll look at one method we use for assessing the quality of raw data.

#### Illumina sequence-by-synthesis

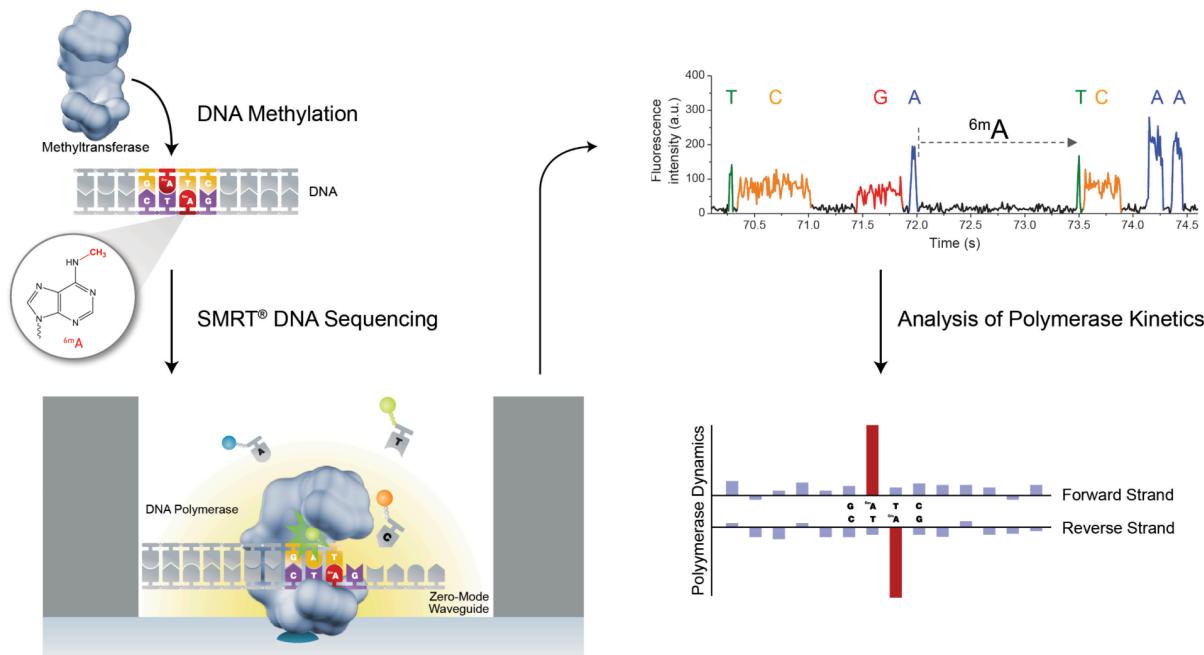
The Illumina sequencers primarily use a sequence-by-synthesis approach, using fluorescently labeled reversible-terminator nucleotides on clonally amplified DNA templates that are immobilized on an acrylamide coating on the surface of a glass flowcell. As nucleotides are incorporated onto the growing molecule attached to the flowcell, they release pulses of light that are captured by the sequencer and processed to derive base sequence.



#### Pacific Biosciences (PacBio) sequencing

PacBio's sequencing method is dubbed Single Molecule Real Time (SMRT) sequencing. DNA polymerase molecules, bound to a dna template, are attached to the bottom of 50nm wells termed Zero-Mode Waveguides (ZMWs). Each ZMW is small enough to see a single nucleotide being incorporated by the bound polymerase. Each of the four bases is attached to a unique

fluorescent dye, and when a nucleotide is incorporated the fluorescent tag is released and diffuses away from the observable area in the ZMW. A detector watches these fluorescent signals and records the fluorescence to determine the base incorporated. These fluorescences and their intensity are recorded over time, and these kinetics are used to calculate the base sequence.



### Comparing capabilities

Each of these systems brings unique strengths to the table, and careful thought should go into your choice of sequencing platform for any given project.

For example, the Illumina platform (HiSeq2500 1T) is good for *de novo* genome sequencing if large insert size libraries used to facilitate scaffolding. However, in case of highly repetitive genomes, polymorphic genomes, or sequencing a population of individuals, the short Illumina reads would not provide optimal results. In such cases, one would need to use long read sequencing platforms such as the PacBio sequencers, and generate *de novo* PacBio assembly or hybrid Illumina/PacBio assembly. Illumina platforms are suitable for cost-effective re-sequencing of isolates if a reference genome is already available and the rapid run of HiSeq2500 (27hrs vs 6 days) or MiSeq (21 days) could be used (depending on the amount of sequence data needed to be generated) as a time-efficient platform.

### Data used in 'Section 1: Genome'

The data we'll be using for the genomic section of the workshop is from the pig whipworm *Trichuris suis* which was chosen for its relatively small size compared to other worm genomes (~80Mb). For expediency's sake, some of the demonstrations will only use a subset of the full dataset that would normally be involved in the genomic analysis of a standard helminth. We'll also fast-forward through some of the lengthier steps and simply move to finished data after showing you how to start the programs involved in each step.

### Getting data off the sequencing machine

Our *T. suis* data was sequenced on a HiSeq 2000 machine. That machine (as with all Illumina platforms) first generates sequence data in a format called ‘Bcl’. Bcl is a binary format that contains base calls and quality scores, but is only machine readable and not anything a typical user will interact with directly. Illumina’s Real-Time Analysis (RTA) software calls and records the series of cycle-specific cluster images per spot on the flowcell and converts that image data into bases and quality values in the Bcl file. It then converts that Bcl file into paired end fastq format using another Illumina program called ‘bcl2fastq’. These paired-end fastq files are the starting point for our analysis.

### An introduction to the Fastq format

While we plan to cover the common sequencing data formats in module 2 of this section, it’s useful at this point to introduce the fastq data format.

Fastq is a plain-text-based file format that contains exactly four lines per sequence record. It starts with a header line, followed by the nucleotide sequence. Then is typically a line containing a plus sign (+), and finally a line containing encoded quality values:

```
@K5HV3:00029:00029  
AAAAAGGGTAAAACGATCGTCACAGG  
+  
AB>>(44*44;:::447444C765?@-
```

The sequence and quality lines must be of the same length (i.e. one quality value per base), and the third line (beginning with a '+') is allowed to contain test (sometimes you may see this third line repeat the sequence header line after the starting plus sign). The quality values in line 4 are encoded such that each numeric value can be represented by a single character. This coding involves converting these quality scores to ascii characters.

Fastq files only support nucleotide sequence data, the format is not meant to house amino acid sequence. ‘Paired end’ fastq usually refers to a set of two fastq files with each file containing the sequencing data for each end of each read fragment. Very importantly, these ends must be ordered identically. There is an alternate form of paired end fastq in which the sequence of each end of the read fragment are kept one after the other within a single fastq file. This format is called ‘interleaved’ fastq.

### Assessing the quality of newly generated fastq

We’ll use the program **FastQC** to check out the quality of the paired end fastq we’ll be using for the next few modules of this section. The FastQC program works on fastq files (as well as sam and bam files, which we’ll discuss in the next module) and runs a number of quality control metrics we can use to assess sequencing data. It’s important to remember that while FastQC uses hard, fast rules to determine when to flag a quality metric with a warning or fail notice, those warning and fail notices do NOT always mean your data is bad. A simple example of this is if you have sequence data from a polyA primed sequencing library, FastQC will likely throw up a fail flag for Kmer Content and possibly for Overrepresented Sequences because many reads will have strings of the base ‘A’. You need to review FastQC results thoughtfully and with awareness of the data you are checking.

Here is how to run **FastQC**:

```
1.1.1: cd ~/WORKSHOP_RESOURCES/Section_1/module_1/QC_sequence_output
```

```
1.1.2: mkdir FASTQC_OUTPUT  
1.1.3: fastqc -o FASTQC_OUTPUT -extract -f fastq raw_data/6p_7kb_TSAC-  
Adult1-g846_g847.1.raw.fastq.gz raw_data/6p_7kb_TSAC-Adult1-  
g846_g847.2.raw.fastq.gz
```

Here is how to view the results

```
1.1.4: chrome FASTQC_OUTPUT/6p_7kb_TSAC-Adult1-  
g846_g847.1.raw.fastq.gz/fastqc_report.html
```

```
1.1.5: chrome FASTQC_OUTPUT/6p_7kb_TSAC-Adult1-  
g846_g847.2.raw.fastq.gz/fastqc_report.html
```

*Useful information:*

(IUPAC code) <http://www.bioinformatics.org/sms/iupac.html>

(Illumina HiSeq 2000 information)

[http://www.illumina.com/documents/products/datasheets/datasheet\\_hiseq2000.pdf](http://www.illumina.com/documents/products/datasheets/datasheet_hiseq2000.pdf)

(Illumina MiSeq information)

[http://www.illumina.com/documents/products/datasheets/datasheet\\_miseq.pdf](http://www.illumina.com/documents/products/datasheets/datasheet_miseq.pdf)

(PacBio RS II information) <http://www.pacificbiosciences.com/products/>

(FastQC) <http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>

---

## Section 1: Genome

### Module 2: Sequence data files

---

This module will be a review of the common formats used to store sequencing data. We'll look at: fasta, fastq, sam & bam. You will also be introduced to the Picard & Fastx toolkits and shown how to convert between these formats.

#### **Fastq**

We've already discussed the fastq data format above. Just as a reminder, this is a four-line-per-sequence-record, nucleotide-only data format that provides both base sequence as well as quality in a single file. Commonly this format will house paired-end data, with the read from each end of the DNA fragments housed in separate, paired fastq files.

#### **Fasta**

One of the most common sequence formats out there, fasta files, are simple text files with each sequence record represented by a header line, and then a variable number of lines containing the sequence data itself. The header lines must begin with the greater-than symbol (>), and after that the line is relatively free-form. Be aware that many programs will only recognize the first white-space delimited word on the header and use that as the identifier for that sequence. For this reason, you will often find the sequence IDs as the first string on these header lines. The sequence section of the fasta format is free-form. Sequence data is often listed using a fixed number of bases per line, but it's completely valid to put an entire genome's worth of sequence on a single line. Some older fasta files used to split the sequence lines with a blank every 10 characters to help make longer sequences more human-readable. You can't make many assumptions about the specific format you will see inside a fasta file. The only safe assumption is that every sequence record will be separated by a header line:

```
>gi|5524378|gb|AAD44166.1| Cytochrome b [Elephas maximus maximus]
ATGATGATGATGATGAAGACAAGGTGAGCCTAAGTAAACTATCAA
CGACGTCAATCAATACTTCTGTGAGGTGCCTACGTAATCAAGCAA
TAATATGATAGAGGGTGGATCAAAACGATTCAAATTGCGCTAACAAAGAG
TTAATGCTTCTTCTTATCCT
```

The fasta format is valid for both nucleotide and protein data.

### **Sam (and Bam)**

The sam format (Sequence Alignment/Map) is an information rich data format for hosting nucleotide (-only) base and quality values. This format also supports the storage of alignment information, but can be used as a simple sequence data format as well. The sam format consists of 11 tab-delimited columns per sequence record (or several lines worth of 11 columns for sequence alignment data in which the same sequence maps to multiple things), as well as a number of header lines. For sequence only sam files (no alignments), there will usually be very few header lines, but for sam files hosting alignment information, there will be at least one header line per reference used in the mapping. These alignment sam files tend to have many of these header lines, so it may be useful view the sam file without the headers (I'll show you how to do this in a bit).

The columns in a sam file are setup to contain a lot of information:

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	[!-?A-~]{1,254}	Query template NAME
2	FLAG	Int	[0,2 <sup>16</sup> -1]	bitwise FLAG
3	RNAME	String	\*  [!-()+-<>-~] [!-~]*	Reference sequence NAME
4	POS	Int	[0,2 <sup>31</sup> -1]	1-based leftmost mapping POSition
5	MAPQ	Int	[0,2 <sup>8</sup> -1]	MAPping Quality
6	CIGAR	String	\*  ([0-9]+[MIDNSHPX=])+	CIGAR string
7	RNEXT	String	\* =  [!-()+-<>-~] [!-~]*	Ref. name of the mate/next read
8	PNEXT	Int	[0,2 <sup>31</sup> -1]	Position of the mate/next read
9	TLEN	Int	[-2 <sup>31</sup> +1,2 <sup>31</sup> -1]	observed Template LENgth
10	SEQ	String	\*  [A-Za-z=.]+	segment SEQuence
11	QUAL	String	[!-~]+	ASCII of Phred-scaled base QUALity+33

Much of this complexity is in place to support the storage of alignment information. If you are dealing with sam as a format solely for hosting sequence data, the important columns are the QNAME, SEQ and QUAL columns (columns 1, 10 and 11 respectively).

The bam file format is often mentioned interchangeably with the sam file format. A bam file is simply the binary (compressed) version of a sam file. It is convenient to keep sam files in their compressed bam format to save space. In fact, many programs prefer bam as input over sam files. The samtools package provides a number of convenient tools for manipulating sam and bam files.

### **Converting between formats**

Here is how to view the contents of a bam file (with headers):

```
samtools view -h <bam>
```

Here is how to view it without headers:

```
samtools view <bam>
```

Here is how to convert a sam file into a bam file:

```
samtools view -bSh -o <bam output> <sam input>
```

Here is how to convert a bam file back into sam:

```
samtools view -h <input bam> > <output sam>
```

Next, we'll practice some ways to convert between fastq, fasta and sam/bam using the Picard and Fastx bioinformatics toolsets. First, we'll create a sorted bam file from a set of paired end fastq files using the Picard toolset

```
1.2.1: cd ~/WORKSHOP_RESOURCES/Section_1/module_2/Convert_between_formats  
1.2.2: java -jar ~/bin/picard.jar FastqToSam F1=raw_data/6p_7kb_TSAC-  
Adult1-g846_g847.1.raw.fastq.gz F2=raw_data/6p_7kb_TSAC-Adult1-  
g846_g847.2.raw.fastq.gz SAMPLE_NAME=6p_7kb_TSAC-Adult1-g846_g847  
SORT_ORDER=queryname OUTPUT=6p_7kb_TSAC-Adult1-  
g846_g847.PE.name_sorted.bam
```

Here we introduce the idea of the 'sorted' bam file. A sorted bam file is simply a bam file that has been sorted either by 'name order' or by 'coordinate order'. Coordinate order sorting is meant for alignment bam files. It re-orders the sequence records within the bam file based on their alignment positions to each reference piece, with the references themselves being ordered alphabetically. Note that if you coordinate sort a bam file that is not aligned, it will work but the ordering will not be correct. Name ordering is the only valid ordering for alignment-free bam files, and it simply orders the sequences based on their names.

Here is how to extract paired end fastq from a bam file:

```
1.2.3: java -jar ~/bin/picard.jar SamToFastq INPUT=6p_7kb_TSAC-Adult1-  
g846_g847.PE.name_sorted.bam F=6p_7kb_TSAC-Adult1-  
g846_g847.end1.new_fastq F2=6p_7kb_TSAC-Adult1-  
g846_g847.end2.new_fastq
```

Here is how to extract fasta from fastq using the Fastx toolset:

```
1.2.4: fastq_to_fasta -i 6p_7kb_TSAC-Adult1-g846_g847.end1.new_fastq -  
o 6p_7kb_TSAC-Adult1-g846_g847.end1.new.fasta
```

*Useful information:*

(Bam/Sam specification) <https://samtools.github.io/hts-specs/SAMv1.pdf>

(Picard Tools) <http://broadinstitute.github.io/picard/command-line-overview.html#Overview>

(Fastx toolkit) [http://hannonlab.cshl.edu/fastx\\_toolkit/](http://hannonlab.cshl.edu/fastx_toolkit/)

## Section 1: Genome

### Module 3: Analytical processing of sequences

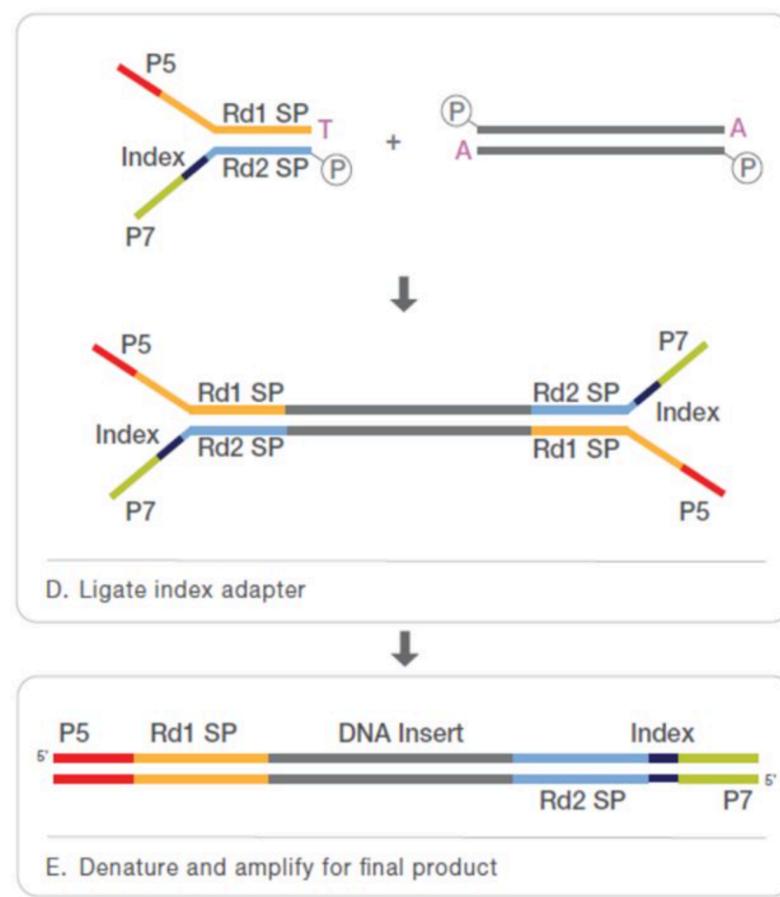
In this module, we'll demonstrate typical steps involved in processing raw sequence data from the HiSeq 2000 platform to an analysis-ready state for assembly. This tutorial will be done on a subset of the *T. suis* data that will be used in the full assembly. In a real world usage case you'd most likely be running this process on multiple pairs of fastq files.

#### Why data is not analysis-ready directly off the sequencing machine

Raw sequence-data from a sequencing machine is technically capable of being used directly in most downstream analyses, but there are a number if factors that make that very ill advised.

#### Sequencing adapters

To prepare DNA material for sequencing, a sequencing library must be made. In our example case, we used the TruSeq genomic library preparation kit for the HiSeq 2000. The DNA sample is first fragmented into pieces roughly 200bp in length. Then TruSeq universal adapters and a specific version of the TruSeq index adapter are ligated onto each end of the fragments via a single base (A) overhang.



The adapter-DNA fragment complex is then denatured and amplified to produce a final product containing the DNA insert, end-specific sequencing primers on either end, as well as a specific index for use in identifying this library out of a pool of libraries.

These TruSeq sequence adapters are normally not visible in the final, sequenced product, because the sequencing primers are immediately adjacent to the DNA insert. However, if some fraction of the DNA inserts are shorter than the expected length, it is possible that sequencing can go all the way across the insert and read into the adapter sequence on the far end. Many of the analyses we typically want can be negatively affected by having adapter sequence left within the reads. It decreases mapping efficiency, confuses

assemblies, etc. It is a good practice to identify and trim off any adapter sequence that may be present in your reads.

### Low sequence quality

During sequencing, each called base is typically assigned a quality score that refers to the likelihood that the base was correctly called by the sequencer. The common value used to represent these per-base confidences is the Phred score.

$$Q = -10 * \log_{10} P$$

$Q \rightarrow$  Phred score

$P \rightarrow$  probability of an error occurring

Eg. Phred 20 implies that you are likely to see 1 error per 100 bases, Phred 30 implies 1 error per 1000 bases, Phred 40 implies 1 error per 10000 bases

Poor quality sequence can interfere with downstream analysis as seriously as untrimmed adapters. It makes mapping less reliable, confuses assemblers, and is a major impediment to variant calling. As with adapter sequence, it is a good practice to trim off low quality sequence that may be present in your reads.

### Length filtering

After trimming reads for adapter and low quality, its possible that some of the reads have been cut down to a very short size. We typically apply a length filter requiring that after the above trimming there be at least 60 bases of read left, otherwise we discard the sequence as a 'short read'. Note that the 60bp threshold is the value we will use for the *T. suis* dataset. If your reads are shorter or longer, you may need to adjust that cutoff.

### Low sequence complexity

These are regions that have an unusual composition that can create problems in sequence similarity searching (as well as other kinds of analyses). These regions contain low information content and can be 'sticky' during alignments. It is a good practice to filter your sequence data for low complexity regions before running downstream analysis.

### Contaminant filtering

Finally, we filter our reads to remove contaminant. By contaminant, we mean any read whose source is not what we expect (in our case, our reads should originate from *T. suis*). Typical sources of contamination are:

Host, bacteria, other (environmental contaminants). Also, for RNA-Seq work, it is often common to filter for 18/16s ribosomal data. This is because the amount of ribosomal sequence present can sometimes dwarf the amount of actual expressed transcript amongst your reads. So it is helpful for downstream analysis to get rid of it.

For this demonstration, we'll be screening for host contaminant only, in this case from pig. In general, you will want to pick and choose the contaminant db's you'll use based on the situation of each project. To save compute resources, we only want to screen for contaminants we expect might be a problem.

Finally, be aware that contamination screening should be done after filtering out adapters, low quality and low complexity sequence. Those earlier issues, if left unfixed, can impede the identification of a read as contaminant.

### Discard both ends or only one?

One thing that must be considered when filtering and screening your data is whether to discard both ends of a paired end set, or only one. Because most sequence data generated is actually sequence from both ends of a single DNA insert, you need to think about whether a problem seen on one end should be considered to apply to both ends or not. In general, issues with adapter, sequencing error, and low complexity are not issues that necessarily affect both ends of a sequence insert. In those cases, we usually will just discard the problem end and retain the other. In the case of one end being identified as a contaminant, we normally will consider both ends contaminants and discard them both.

### Processing raw reads into an analysis-ready state

Now we're going to walk through typical steps we'd use to prepare our *T. suis* reads for assembly. To do this we need to accomplish these things:

- remove any adapters that may have been introduced during sequencing library preparation
- remove low quality, terminal regions
- apply a length filter to remove short reads after trimming
- remove reads of low sequence complexity
- remove reads that originate from host organism

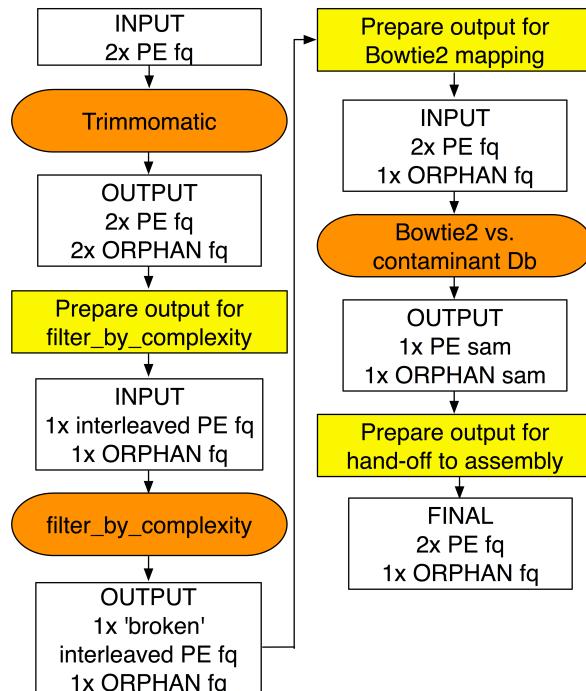
We'll use the program Trimmomatic for adapter removal, quality trimming and length filtering. The filter\_by\_complexity script from the seq\_crumb package will remove reads of low sequence complexity, and we'll use the Bowtie2 aligner to map the trimmed reads against a host database. Between these steps some file manipulation is required to get the sequences into the format needed for the next step. This data shuffling will be done using parts of the samtools & KHMER packages, as well as some old fashioned command line unix.

### The analysis-ready output

This processing will result in a set of paired end fastq, and an extra fastq of orphaned reads whose mates were discarded (due to filtering steps that removed only a single end from a pair). This process can be messy on a technical level due to the need to convert data between the bam & fastq formats, and the need to keep the paired-end data synchronized and free of orphans. In practice, we would normally assemble all these steps into a single pipeline script. For the purposes of this demonstration, we'll walk through each step manually.

Be aware that there are alternatives to the software we're showing for most of these steps. The programs we're using are generally robust, but you may want to experiment with other options for your data. No tool does a perfect job, and you may be able to find tools that perform better or more efficiently for your specific dataset.

Finally, it's often reasonable to simply work only with paired end data, and discard the small fraction of orphaned reads generated at each step. This simplifies the process at the expense of



a small fraction of your reads. This is actually a fairly common practice, especially if you find yourself doing an extra hour of coding work to preserve a few thousand reads out of 200 million reads.

### Processing the data

Here are the steps involved in running Trimmomatic and preparing the output for the next step. This step trims off adapter, quality trims and filters the trimmed reads based on length.

```
1.3.1: cd ~/WORKSHOP_RESOURCES/Section_1/module_3/Processing_genomic_data_to_cleaned_state
1.3.2: java -jar ~/bin/trimmomatic-0.33.jar PE -threads 8 -phred33 -trimlog TRIMLOG.txt raw_data/6p_7kb_TSAC-Adult1-g846_g847.1.raw.fastq.gz raw_data/6p_7kb_TSAC-Adult1-g846_g847.2.raw.fastq.gz 6p_7kb_TSAC-Adult1-g846_g847.PE_end1.fastq 6p_7kb_TSAC-Adult1-g846_g847.ORPHANS_end1.fastq 6p_7kb_TSAC-Adult1-g846_g847.PE_end2.fastq 6p_7kb_TSAC-Adult1-g846_g847.ORPHANS_end2.fastq
ILLUMINACLIP:databases/TruSeq_adapters.fna:2:30:10 SLIDINGWINDOW:5:20
LEADING:20 TRAILING:20 MINLEN:60
1.3.3: cat 6p_7kb_TSAC-Adult1-g846_g847.ORPHANS_end1.fastq 6p_7kb_TSAC-Adult1-g846_g847.ORPHANS_end2.fastq > Tsuis_genomic_7kb_insert.trimmomatic_ALL_ORPHANS.fastq
1.3.4: java -jar ~/bin/picard.jar FastqToSam F1=6p_7kb_TSAC-Adult1-g846_g847.PE_end1.fastq F2=6p_7kb_TSAC-Adult1-g846_g847.PE_end2.fastq SAMPLE_NAME=Tsuis_genomic_7kb_insert SORT_ORDER=coordinate OUTPUT=Tsuis_genomic_7kb_insert.trimmomatic_PE_coord_sorted.bam
1.3.5: java -jar ~/bin/picard.jar SamToFastq INPUT=Tsuis_genomic_7kb_insert.trimmomatic_PE_coord_sorted.bam INTERLEAVE=true FASTQ=Tsuis_genomic_7kb_insert.trimmomatic_PE_interleaved.fastq
```

Next we filter out low complexity data using `filter_by_complexity` from the `seq_crumb`s package, and then prepare the output for the final Bowtie2 mapping step

```
1.3.6: filter_by_complexity -o Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.brokenPE_interleaved.fastq --paired_reads --fail_drags_pair False Tsuis_genomic_7kb_insert.trimmomatic_PE_interleaved.fastq
1.3.7: filter_by_complexity -o Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.ORPHANS.fastq Tsuis_genomic_7kb_insert.trimmomatic_ALL_ORPHANS.fastq
1.3.8: source /home/ec2-user/bin/KHMER/khmerEnv/bin/activate
1.3.9: extract-paired-reads.py -f Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.brokenPE_interleaved.fastq
1.3.10: deactivate
```

```

1.3.11: cat
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.ORPHANS.fastq
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.brokenPE_interleaved.fastq.se >
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.ALL_ORPHANS.fastq
1.3.12: paste - - - - - <
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.brokenPE_interleaved.fastq.pe | tee >(cut -f 1-4 | tr "\t" "\n" >
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.PE_end1.fastq) |
cut -f 5-8 | tr "\t" "\n" >
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.PE_end2.fastq

```

Finally, we will map the cleaned reads against a host database, pig in this case, and remove all reads and read pairs that have either end detected as a contaminant. We'll use Bowtie2 for this mapping, and we'll prepare the final, cleaned & contaminant free data for assembly

```

1.3.13: bowtie2-build Sus_scrofa.Sscrofa10.2.dna_rm.toplevel.fa
Sus_scrofa.Sscrofa10.2.dna_rm.toplevel
1.3.14: bowtie2 -q -x databases/Sus_scrofa.Sscrofa10.2.dna_rm.toplevel
-1 Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.PE_end1.fastq -
2 Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.PE_end2.fastq -S
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.mapped_to_host.PE.sam
1.3.15: bowtie2 -q -x databases/Sus_scrofa.Sscrofa10.2.dna_rm.toplevel
-U
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.ALL_ORPHANS.fastq
-S
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.mapped_to_host.ORPHANS.sam

```

```

1.3.16: samtools view -bSh -o
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.mapped_to_host.ORPHANS.bam
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.mapped_to_host.ORPHANS.sam
1.3.17: samtools sort
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.mapped_to_host.ORPHANS.bam
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.mapped_to_host.ORPHANS.sorted
1.3.18: bamtools filter -in
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.mapped_to_host.ORPHANS.sorted.bam -out
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.host_free.ORPHANS.bam -isMapped false
1.3.19: java -jar ~/bin/picard.jar SamToFastq
INPUT=Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.host_free.ORPHANS.bam
FASTQ=Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.host_free.ORPHANS.fastq

```

```

1.3.20: samtools view -bSh -o
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.mapped_to_host.PE.
bam
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.mapped_to_host.PE.
sam
1.3.21: samtools sort
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.mapped_to_host.PE.
bam
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.mapped_to_host.PE.
sorted
1.3.22: bamtools filter -in
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.mapped_to_host.PE.
sorted.bam -out
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.host_free.PE.bam -
isMapped false -isMateMapped false
1.3.23: java -jar ~/bin/picard.jar SamToFastq
INPUT=Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.host_free.PE
.bam
FASTQ=Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.host_free.PE
_end1.fastq
SECOND_END_FASTQ=Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.h
ost_free.PE_end2.fastq

```

#### Evaluating our analysis-ready data

Now that we've processed our data to an analysis-ready state, lets run FastQC again on the final output and compare it back to the FastQC results from the original, raw data

```

1.3.24: cd /home/ec2-
user/WORKSHOP_RESOURCES/Section_1/module_3/Processing_genomic_data_to_
cleaned_state
1.3.25: mkdir FASTQC_OUTPUT
1.3.26: fastqc -o FASTQC_OUTPUT -extract -f
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.host_free.PE_end1.
fastq
Tsuis_genomic_7kb_insert.trimmomatic_and_complexity.host_free.PE_end2.
fastq

```

We'll then use the chrome browser (as before) to compare the final paired fastq files to the original, raw paired fastq files.

#### *Useful information:*

(Trimmomatic) <http://www.usadellab.org/cms/?page=trimmomatic>  
 (Bowtie2) <http://bowtie-bio.sourceforge.net/bowtie2/index.shtml>  
 (seq\_crumps) [https://bioinf.comav.upv.es/seq\\_crumps/available\\_crumps.html](https://bioinf.comav.upv.es/seq_crumps/available_crumps.html)  
 (bamtools) <https://github.com/pezmaster31/bamtools>

---

## **Section 1: Genome**

### **Module 4: Genome assembly**

---

There are a lot of choices when deciding on a genome assembler. Considerations include the predicted genome size, the technology type, and the cost (computational, and paying for the assembler). Today's demonstration will be using ALLPATHS-LG, which is a de Bruijn Graph assembler for large genomes. ALLPATHS-LG requires paired end reads from at least one fragment and one jumping library sequenced on the Illumina platform. The use of multiple libraries enables ALLPATHS-LG to build a higher quality assembly. When using ALLPATHS\_LG, our recommended sequence coverage requirements are: 45x fragments, 45x 3-8kb and 10x-20x lrg insert ie. 5kb+. In our assembly, we will be using 11,898,407 fragment read pairs, 4,960,173 3kb read pairs and 2,975,142 7kb read pairs

ALLPATHS-LG requires a specific format for input sequence data files in order to run the assembler. PrepareAllPathsInputs.pl, an ALLPaths script, will be run after we begin by setting up two dependency files:

#### **Dependency File #1: in\_groups.csv**

```
100,Illumina_011,/home/ec2-
user/WORKSHOP_RESOURCES/Section_1/module_4/Tsuis/all_path_data/13p_fra-
gment.*.trimPaired.fastq.gz
200,Illumina_012,/home/ec2-
user/WORKSHOP_RESOURCES/Section_1/module_4/Tsuis/all_path_data/33p_3-
5kb_.*.trimPaired.fastq.gz
300,Illumina_013,/home/ec2-
user/WORKSHOP_RESOURCES/Section_1/module_4/Tsuis/all_path_data/6p_7kb_-
*.trimPaired.fastq.gz
```

Notes: This file does not require a header with each field type.

Group name: unique name for data set (free form)

Library name: library name for data set (free form but good practice to use some identifying nomenclature)

File name: absolute path to data file. Wildcard characters \* and ? are accepted in the name of the file but NOT the file extension.

Supported extensions are .bam, .fasta, .fa, .fq, .fastq.gz, and fq.gz (all case specific). Also, if you use .fasta or .fa, the script expects a corresponding .quala or .qa file to exist for each respective file.

#### **Dependency File #2: in\_libs.csv**

```
library_name,project_name,organism_name,type,paired,frag_size,frag_std-
dev,insert_size,insert_stddev,read_orientation,genomic_start,genomic_e-
nd
Illumina_011,Awesome,T.suis,fragment,1,205,10,,,inward,,
Illumina_012,Awesome,T.suis,jumping,1,,,7475,500,outward,,,
Illumina_013,Awesome,T.suis,jumping,1,,,2833,500,outward,,
```

Notes: Every field is required in this file. A header is used, and each field must be represented in the data entered (a comma separated field can be left blank; see example above).

Library\_name: must match same field in in\_group.csv

Project\_name: free form name

Organism\_name: organism

Type: informative field only ie fragment, jumping EcoP15, etc.

Paired: 0 = unpaired reads; 1: paired reads

Frag\_size: average # of bases in the fragment library

Frag\_stddev: estimated standard deviation of the fragment sizes

Insert\_size: average # of bases in the jumping library inserts (if larger than 20kb the library is considered Long Jumping)

Insert\_stddev: estimated standard deviation of the insert sizes

Read\_orientation: inward or outward

Genomic\_start: index of the FIRST genomic base in the reads. If not zero, then all bases before the genomic start will be trimmed off

Genomic\_end: index of the LAST genomic base in the reads. If not zero, then all bases after the genomic end will be trimmed off

With these two files prepared, you can now run:

```
1.4.1: PrepareAllPathsInputs.pl DATA_DIR=/home/ec2-
user/WORKSHOP_RESOURCES/Section_1/module_4/Tsuis/all_path_data
PLOIDY=2
```

Other optional settings include:

PICARD\_TOOLS\_DIR (use version 1.101) if you are using .bam files in the .csv files made above.

INCLUDE\_NON\_PF\_READS=1 allows you to use the orphan reads kept in the previous module. GENOME\_SIZE, FRAG\_COVERAGE, JUMP\_COVERAGE, and LONG\_JUMP\_COVERAGE used together you can set the desired coverage percentage based on the estimated size set for GENOME\_SIZE

Now we can start the assembly:

```
1.4.2: RunAllPathsLG PRE=/home/ec2-
user/WORKSHOP_RESOURCES/Section_1/module_4 REFERENCE_NAME=Tsuis
DATA_SUBDIR=all_path_data RUN=myrun SUBDIR=attempt1
```

Notes: All of the **ALLPATHS** arguments are to set the pipeline directory names. If your ALLPATHS run fails at any point, you can troubleshoot the issue and then restart ALLPATHS and it will restart on the stage that failed (as long as you don't delete any of the directories/data that was produced up to that point).

Use the following command which adds "OVERWRITE=True":

```
1.4.3: RunAllPathsLG PRE=/home/ec2-
user/WORKSHOP_RESOURCES/Section_1/module_4 REFERENCE_NAME=Tsuis
DATA_SUBDIR=all_path_data RUN=myrun SUBDIR=attempt1 OVERWRITE=True
```

This assembly took 5.3 hours. When the assembly finishes it will be found at the following location:

/home/ec2-user/WORKSHOP\_RESOURCES/Section\_1/module\_4/Tsuis/all\_path\_data/myrun/ASSEMBLY/attempts1/final.assembly.fasta

*Useful information:*

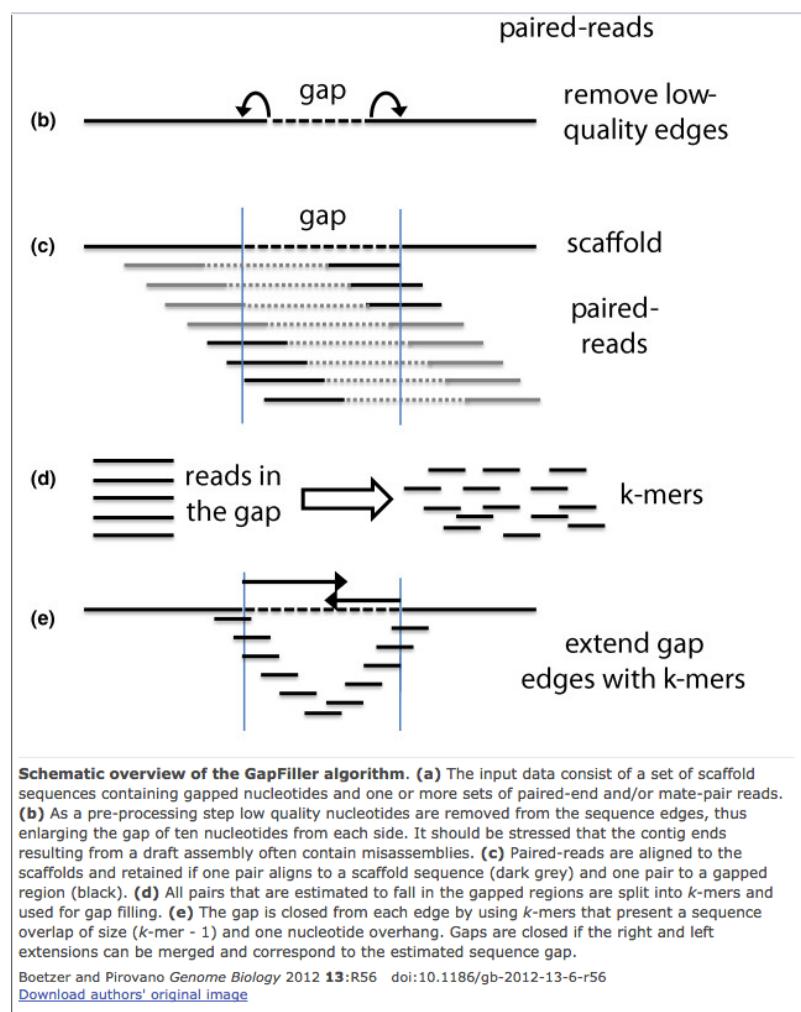
[ftp://ftp.broadinstitute.org/pub/crd/ALLPATHS/Release-LG/AllPaths-LG\\_Manual.pdf](ftp://ftp.broadinstitute.org/pub/crd/ALLPATHS/Release-LG/AllPaths-LG_Manual.pdf)

Quality Assessment

Assembly improvement and QC of de novo assemblies go hand in hand since high-quality draft genomes lead to more successful and accurate annotation. We use a combination of CEGMA, N50, and RNA mapping to assess the quality of an assembly.

**CEGMA** (Core Eukaryotic Genes Mapping Approach) uses a defined set of 458 single-copy, conserved eukaryotic genes, and searches for orthologs of these proteins in the de novo genome assembly. Since these proteins are conserved across eukaryotes ranging from yeast to plants to humans, the completeness of this protein set in a draft genome is a useful indicator of the genome quality. CEGMA produces a completeness report, but we prefer to parse the cegma.gff file against the core proteins to get a count of the CEGs (Core Eukaryotic Genes) and lCEGs found in the assembly.

**1.4.4:** cegma --genome final.assembly.fasta -threads 8 &



**N50** is a basic statistic for describing how contiguous an assembly is. The longer the N50 is, the better the assembly.

**RNA mapping** looks at the percent of gene contained within the assembly

Assembly Improvement  
After assessing an assembly, we can take advantage of numerous assembly improvement tools. Two open source options that we use are **GapFiller** and **PBJelly**. PBJelly (part of PBSuites) is able to fill gaps and merge scaffolds utilizing long reads (which is particularly useful for PacBio data). For this project, we do not have any available PacBio data, so we will be utilizing GapFiller instead. The image below illustrates how GapFiller fills the contig gaps:

We will start by creating the libraries.txt file:

```
Libraries File: libraries.txt
lib100 bowtie
/gscmnt/gc2546/mitrevalab/research/t_suis_class/25p_fragment_lib_TSAC-
Adult1-g846_g847.1.raw.fastq
/gscmnt/gc2546/mitrevalab/research/t_suis_class/25p_fragment_lib_TSAC-
Adult1-g846_g847.2.raw.fastq 205 0.3 FR
lib200 bowtie /gscmnt/gc2546/mitrevalab/research/t_suis_class/65p_3-
5kb_TSAC-Adult1-g846_g847.1.raw.fastq
/gscmnt/gc2546/mitrevalab/research/t_suis_class/65p_3-5kb_TSAC-Adult1-
g846_g847.2.raw.fastq 7475 0.5 RF
lib300 bowtie
/gscmnt/gc2546/mitrevalab/research/t_suis_class/12p_7kb_TSAC-Adult1-
g846_g847.1.raw.fastq
/gscmnt/gc2546/mitrevalab/research/t_suis_class/12p_7kb_TSAC-Adult1-
g846_g847.2.raw.fastq 2833 0.5 RF
```

Notes:

Library Name: free form  
Mapper: bowtie or bwa  
Path to both mate pairs files  
Insert size  
Error  
Read orientation

We then run **GapFiller** using:

```
1.4.5: GapFiller.pl -l libraries.txt -s final.assembly.fasta -T 8 -b
Tsuis -i 5
```

Notes: -l is the file made above; -s assembly file; -T threads; -b directory and root file name; -i iterations. Runtime varies based on number of gaps and amount of data used

*Useful links:*

<http://korflab.ucdavis.edu/datasets/cegma/README>

---

## Section 1: Genome

### Module 5: Genome annotation

---

“A beginner’s guide to eukaryotic genome annotation”

<http://www.nature.com/nrg/journal/v13/n5/full/nrg3174.html> is a great resource. The first step when annotating a genome is to identify repeat sequences, because they can interfere with gene predictors and evidence alignment.

Masking repeats

**Tandem Repeat Finder (TRF)**: Start by using TRF to mask short interspersed tandem repeats:

```
1.5.1: trf Tsuis.gapfilled.final.fa2 7 7 80 10 50 500 -d -m -h >>  
TRF.stdout
```

Now we need to create a blast database for **RepeatModeler**:

```
1.5.2: makeblastdb -in final.assembly.fasta.2.7.7.80.10.50.500.mask -  
dbtype nucl
```

Running **RepeatModeler** (run time is 24-36 hours):

```
RepeatModeler -database Tsuis.gapfilled.final.fa.masked.fasta >>  
RM_stdout
```

Repeatmodeler will create an RM\_[PID].[DATE]/ directory,  
(e.g. RM\_10825.ThuAug271528572015/)

Once RepeatModeler has completed, you will need to QC the output to check for repeats that are really genes (gene families) or RNA features.

The following are the screening steps for QC:

**Blastx** vs nr for protein coding genes:

```
1.5.3: blastx nr consensi.fa.classified E=10e-5 -o  
consensi.fa.classified.nrcheck.blast.out
```

**Blastn** vs RNA database for ribosomal or other RNA genes (Rfam.fasta comes with the Rfam download):

```
1.5.4: blastn Rfam.fasta consensi.fa.classified 10e-5 -o  
$1.rnacheck.blast.out
```

Retrotransposon check:

```
1.5.5: blastx transposonDb consensi.fa.classified E=10e-5 -o  
$1.retrocheck.blast.out
```

The final file output from RM is consensi.fa.classified file in the RM directory (e.g. .M\_10825.ThuAug271528572015/consensi.fa.classified). We then screen the blast.out files with tools that look at P >=0.01 identity/coverage (50% PID/20% Identity) and naming that is known to be acceptable and database types that lead us to believe the protein has been checked:

"unknown", "hypothetical", "oxidase", "histone", "kinase", "protease", "reductase", "RNA", "synthase", "ATPase", "phosphatase", "cytochrome", "ribosomal", "titin", "extensin", "abductin", "tRNA", "drosophila", "nucleosome", "transferase", "unnamed", "polyprotein", "putative", "peptide", "resolvase", "alpha", "beta", "fusion", "lactamase", "galact", "integrase", "ref", "emb", "dbj", "gb", "pir", "prf", "sp", "pdb", "intron", "synthetase"

```
1.5.6: mkdir RepeatMasker
```

```
1.5.7: cd RepeatMasker
```

```
1.5.8: RepeatMasker -lib repeats.lib  
trf.masked.fasta >>RepeatMasker.stdout
```

Note: The input sequence can be split into chunks to expedite.

**RepeatMasker** outputs the following files:

D918.newname.fsa.masked - Masked fasta

D918.newname.fsa.tbl - Summary table gives total %masked and breakdown of types

D918.newname.fsa.log - run output

Other output files with details of repeats/positions etc.

D918.newname.fsa.cat D918.newname.fsa.out D918.newname.fsa.ref

*Useful links:*

<https://tandem.bu.edu/trf/trf.html>

<http://www.repeatmasker.org/RepeatModeler.html>

<http://www.repeatmasker.org/webrepeatmaskerhelp.html>

We also annotate non-coding RNAs using the Rfam and tRNA scan. We mask these predictions before running the predictor programs, in order to further simplify the regions the predictors have to look at.

**Rfam** - <http://nar.oxfordjournals.org/content/43/D1/D130>

```
1.5.9: rfam_scan -f tab -o Rfam.out File.fasta
```

-f specifies format

-o specifies output location

The last argument is just the sequence file to use

Notes: For rfam scan, we modified the script so that it skips the rare group II introns, because if greatly reduces the run time.

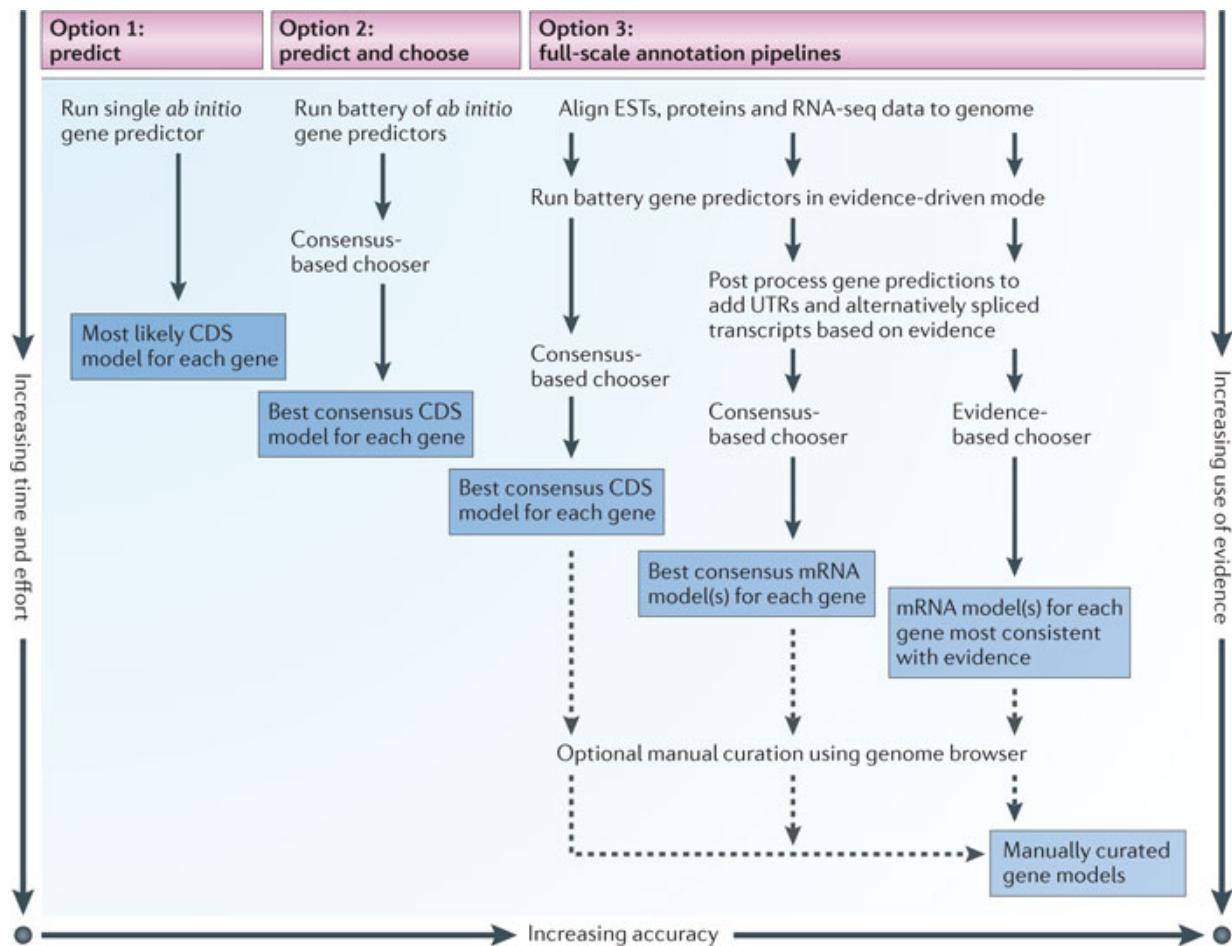
We can scan a sequence file for tRNAs using tRNAscan, EufindtRNA & tRNA covariance.

**tRNAscan** - <http://lowelab.ucsc.edu/tRNAscan-SE/Manual>

```
1.5.10: tRNAscan-SE -o tRNAscan.output File.fasta
```

Annotation

Producing gene predictions to produce a high quality final set of gene annotations.



Nature Reviews | Genetics

A beginner's guide to eukaryotic genome annotation. M Yandell & D Ence *Nature Reviews Genetics* 13, 329-342 (May 2012).

We will perform annotation using **Maker** (M. Yandell et. al., 2007)  
<http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2134774/pdf/188.pdf>

First, we generate config files for **Maker**:

1.5.11: `maker -CTL`

This creates 3 default config files:

`maker_bopts.ctl` : blast type and cut-off  
`maker_exe.ctl` : program paths  
`maker_opts.ctl` : all other parameters

The `-CTL` option will give you default parameters. You will need to set up paths in each file to match the system you are on (paths to blast databases, etc). For our maker runs, we only need to do the `-CTL` once, and then we copy the `ctl` files to the new directories so we don't have to update paths for blast exe's etc. We only need to change the `maker_opts.ctl` file for blast db's.

Change any parameters and/or paths which are different from the working copy, including:

path to sequence file  
protein database path  
EST database path  
alt-est database path if needed  
ab initio predictors being run  
ab initio corresponding model files  
model\_gff and/or pred\_gff or other \_gff files  
evidence predictors

Open maker\_opts.ctl and add path to these lines

Find this line:

#----EST Evidence (for best results provide a file for at least one):

**1.5.12:** est= #set of ESTs or assembled mRNA-seq in fasta format

Beneath this EST Evidence section, also change:

#----Protein Homology Evidence (for best results provide a file for at least one)

**1.5.13:** protein= #protein sequence file in fasta format (i.e. from multiple organisms)

To run any of the predictors: **Snap**, **Fgenesh**, **Augustus** you need to train them. **Fgenesh** is a commercial predictor that you would need to purchase, but snap is free and maker also works with **GeneMark** and others, but those are the most common. We are not going to go perform overpredictor training today.

To run maker:

**1.5.14:** maker --RM\_off -g File.fasta maker\_bopts.ctl maker\_exe.ctl  
maker\_opts.ctl

Here is some information on the directory structure and the files that maker outputs:

Path/Maker

maker\_bopt.ctl  
maker\_ext.ctl  
maker\_opts.ctl

GENOME.maker.output/ - contains all output for a given run of MAKER

maker\_bopts.log : These are logs of the control files used for this run of MAKER  
maker\_opts.log  
maker\_exe.log

seen.dbm

sequnce\_maker\_length\_99.db  
sequnce\_maker\_length\_99\_master\_datastore\_index.log - log of MAKER run progress as well as an index for traversing through the output

mpi\_blastdb/ - Contains fasta indexes and error corrected fasta files built from the EST and protein database provided by the user.

\*.mpi.10/ - contains indexed database files

nematode\_protein\_new.mpi.10/ - contains indexed database files

<Sequence\_name>\_datastore/ - contains subdirectories that hold the output for each individual contig of the input fasta file. See DATASTORE DIRECTORY STRUCTURE section in README for more information

08/ 25/Contig#/ - first two directories; numbers/letters vary

run.log

<Sequence\_name>.gff - a gff file that can be loaded into GMOD, GBROWSE, or Apollo

<Sequence\_name>.maker.snap.proteins.fasta - a fasta file of ab-initio predicted protein sequences from program

<Sequence\_name>.maker.snap.transcripts.fasta - a fasta file of ab-initio predicted transcript sequences from program

<Sequence\_name>.maker.transcripts.fasta - a fasta file of the MAKER annotated transcript sequences

<Sequence\_name>.maker.proteins.fasta - a fasta file of the MAKER annotated protein sequences

<Sequence\_name>.maker.non\_overlapping\_ab\_initio.proteins.fasta - a fasta file of filtered ab-initio protein sequences that don't overlap maker annotations

<Sequence\_name>.maker.non\_overlapping\_ab\_initio.transcripts.fasta - a fasta file of filtered ab-initio transcript sequences that don't overlap maker annotations

theVoid.Contig#/ - a directory containing all of the raw output files produced by MAKER, including BLAST reports, SNAP output, exonnerate output and the masked genomic sequence.

## Explaining GFF3 files

<http://www.broadinstitute.org/annotation/argo/help/gff3.html>

Field Descriptions (Note: Except for the last field [9], all gff flavors are the same):

1. **seqname** - The name of the sequence. Typically a chromosome or a contig. Argo does not care what you put here. It will superimpose gff features on any sequence you like.
2. **source** - The program that generated this feature. Argo displays the value of this field in the inspector but does not do anything special with it.
3. **feature** - The name of this type of feature. The official GFF3 spec states that this should be a term from the SOFA ontology, but Argo does not do anything with this value except display it.
4. **start** - The starting position of the feature in the sequence. The first base is numbered 1.
5. **end** - The ending position of the feature (inclusive).
6. **score** - A score between 0 and 1000. If there is no score value, enter ".".
7. **strand** - Valid entries include '+', '-', or '.' (for don't know/don't care).
8. **frame** - If the feature is a coding exon, *frame* should be a number between 0-2 that represents the reading frame of the first base. If the feature is not a coding exon, the value should be '..'. Argo does not do anything with this field except display its value.
9. **GFF3: grouping attributes** *Attribute keys and values are separated by '=' signs. Values must be URI encoded. quoted. Attribute pairs are separated by semicolons. Certain, special attributes are used for grouping and identification (See below). This field is the one important difference between GFF flavors.*

Special Field 9 Attributes:

The first special thing about field 9 attributes is that they can be associated with transcripts. Previous flavors of GFF restricted attributes to the lowest level subfeature (exons).

Any key=value attribute pair will be displayed by argo, but the following have special meaning:

1. ID - unique identifier for this feature.
2. Parent - identifier of parent feature.
3. Name - used as the feature label in the feature map.

---

## **Section 1: Genome**

### **Module 6: Functional annotation**

---

This module will review two standard methods for assigning functional annotations to a de novo geneset. We'll run a protein vs. protein alignment using the NCBI's BLASTP+, and we'll discuss the interproscan program and take a look at typical interproscan output and how to make use of it.

## Using NCBI's BLASTP+ to assign functional annotation

One common method of assigning a function to a set of de novo gene calls is simply by mapping them to an already annotated set of genes from closely related organisms. We'll go over the details of actually running a blastp in a bit, but first we'll review how to locate and prepare a database for this mapping.

If you happen to have a highly conserved organism's gene set handy that happens to already be well annotated, you may not need to do anymore digging. For example, if you are working with a non-parasitic nematode, you can't do much better than to simply use the highly curated and well annotated *C.elegans* gene set for this mapping. But if you are working with an organism not in that happy circumstance (as most of us are, all the time), the next best thing is to walk the lineage of your species using GenBank's Entrez Records which are available when using a taxonomy search. Lets do this now:

1. Open a browser on your laptop
2. Go to the NCBI website at <http://www.ncbi.nlm.nih.gov>
3. Enter "*Trichuris suis*" in the search box at the top of the screen, and set the search menu to "Taxonomy", then click "Search"
4. Click through the "*Trichuris suis*" link
5. Notice the "Entrez records" table on the right side of the screen. What we want is to find a level of taxonomy above our species for which GenBank has a good number of "Protein" available
6. Click on the genus level link in the lineage (Trichuris)
7. Click the "Trichuris" link at the top of the list of species to get back to the "Entrez records" table at that level in the taxonomy
8. Notice that GenBank has 48,510 proteins available for this taxa, click on the "48.510" number which is a link that will prepare an output set of those proteins
9. Now we will download this protein set. Open the "Send to:" menu in the upper right corner of the page
10. Choose Destination "File"
11. Set the Format to "Fasta"
12. Click on "Create File" to download the file

For the purposes of this workshop I've already prepared a somewhat smaller db for use in our demonstration, which is already available in the EC2 instance (i.e. you don't really need to download the above). But the above process is very useful for when you don't have a specific protein db in mind, yet you want to assign blastp annotations to your gene set basic on homology to related organisms.

## Running NCBI's BLASTP+

Now we're going to actually map our gene set to our protein database and filter based on alignment strength. First we need to prepare the blast database for use.

```
1.6.1: cd /home/ec2-
user/WORKSHOP_RESOURCES/Section_1/module_6/NCBI_Blast+/database
1.6.2: makeblastdb -in Ttrichuira_geneset.fna -dbtype prot
```

Next we can start the blastp alignment

```
1.6.3: cd /home/ec2-
user/WORKSHOP_RESOURCES/Section_1/module_6/NCBI_Blast+
1.6.4: blastp -db database/Ttrichiura_geneset.fna -query
Tsuis.protein.faa -num_threads 8 -outfmt 6 -max_target_seqs 1 -out
Tsuis_vs_Ttrichiura.raw_blastp.tsv
```

Then we would typically parse the results using some alignment scoring threshold to filter out only the solid hits

```
1.6.5: awk '{if ($11<1e-05) print $0;}' <
Tsuis_vs_Ttrichiura.raw_blastp.tsv >
Tsuis_vs_Ttrichiura.raw_blastp.tsv.hits_at_1e-05
```

The output format we selected using the `-outfmt 6` argument produces results in this tab-delimited format:

Query id  
Subject id  
Percent identity  
Alignment length  
Mismatch count  
Gap open count  
Start of alignment in query  
End of alignment in query  
Start of alignment in subject  
End of alignment in subject  
E-value  
Bitscore

The results at this point will provide associations between our de novo gene set and the genes from our database. We would then use a lookup script to go back and extract the full line annotations from our database and add them to our new genes. While we won't cover that in this workshop, we'd be happy to provide scripts for this on request after the class.

### Interproscan

Interproscan is a program that searches a collection of databases and reports associations to all these databases for each gene searched. For our purposes we are mainly interested in the IPR and GO annotations provided by this software. But here is a full listing of what is searched:

PANTHER, PFAM, PIRSF, PRINTS, PRODOM, PROSITE, PROFILE, SMART, TIGRFAMs, GENE3D, SSF, SWISSPROT, TREMBL, INTERPRO, GO, MEROPS, UniProt, HAMAP, PFAMB

Due to its resource intensive nature, and the size of the databases needed in its execution we are not able to demonstrate this software live in our workshop. So we've short-cut this section and deposited pre-built interproscan output for the *T.suis* gene set in the EC2 instance. First lets take a look at the raw output

```
1.6.6: cd /home/ec2-
user/WORKSHOP_RESOURCES/Section_1/module_6/Interproscan
1.6.7: more trichuris_suis_interpro_results
```

That command will scroll through that file one page at a time. The length of each line will cause the output to wrap on your screen, making it look messy. But the output of interproscan is actually organized into tab-delimited columns:

1. Protein Accession
2. Sequence MD5 digest
3. Sequence Length
4. Analysis (i.e. the db that was searched on this line)
5. Signature Accession
6. Signature Description
7. Start location
8. Stop location
9. Score (i.e. usually the e-value of the match reported by member database method, although sometimes a specific search engine will report a non- e-value based score)
10. Status
11. Date
12. InterPro annotations – accession (optional column)
13. InterPro annotations – description (optional column)
14. GO annotations (optional column)
15. Pathways annotation (optional column)

### Parsing Interproscan results for downstream use

In order to prepare these annotations for downstream analysis (primarily the building of the gene summary table, and the expression analysis that will be shown in Section 2) we need to parse our raw interproscan output into a pre-arranged format that we typically use for that later work. This requires the use of a locally generated perl script (that we're happy to share on request), and would normally build files for both GO and IPR annotations. As a demonstration we'll show how we use this script to generate the GO index

**1.6.8:** scripts/prepare\_files\_for\_FUNC.no\_parents.pl -iprscan\_file trichuris\_suis\_interpro\_results -GO\_description GO.terms\_and\_ids.obo.120531 -gene\_fof tsuis\_full\_gene\_list.txt -output Tsuis.GO\_annotation\_index

If you ‘more’ the output file, you’ll see that this is a much simpler format than trying to work with the native interproscan result file. This parsed annotation file will be used in Section 2 to help populate the gene summary table in Excel. This format (3 simple columns) should be easy to work with within the spreadsheet.

#### *Useful information:*

- (NCBI BLAST+ UNIX tutorial) [https://molevol.mbl.edu/wiki/index.php/BLAST\\_UNIX\\_Tutorial](https://molevol.mbl.edu/wiki/index.php/BLAST_UNIX_Tutorial)
- (NCBI BLAST+ command line arguments) <http://www.ncbi.nlm.nih.gov/books/NBK279675/>
- (Interproscan) <https://github.com/ebi-pf-team/interproscan/wiki>
- (Interpro Db) <https://www.ebi.ac.uk/interpro/about.html>
- (Gene Ontology) <http://geneontology.org>

---

## **Section 2: Transcriptome**

### **Module 0: RNA isolation to sequence production**

---

- 1) Experimental design
- 2) Library construction & sequencing



---

## **Experimental design**

---

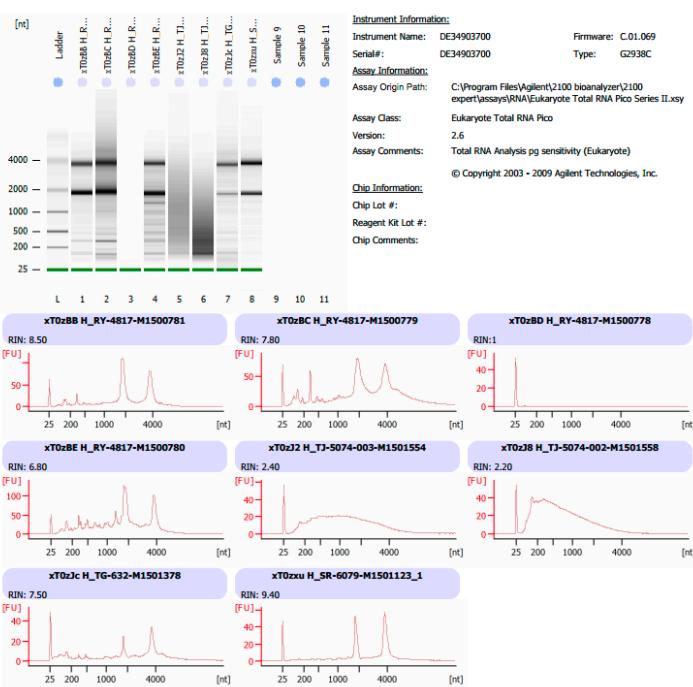
- What's the purpose?
  - Gene discovery
  - Differential expression
- More reads = more confidence
  - Depth
    - Depends on genome size, coding features, etc.
    - More for discovery of novel features, low expression genes
  - Replicates
    - Biological, not technical
    - More is better for differential expression, 3 per condition
- Collect appropriate meta-data when you collect your RNA
  - Strain/isolate/batch
  - Sex, age, patency
  - Treatments

Resource: [https://genome.ucsc.edu/ENCODE/protocols/dataStandards/ENCODE\\_RNAseq\\_Standards\\_V1.0.pdf](https://genome.ucsc.edu/ENCODE/protocols/dataStandards/ENCODE_RNAseq_Standards_V1.0.pdf),  
<http://gkno2.tumblr.com/post/24629975632/thinking-about-rna-seq-experimental-design-for>



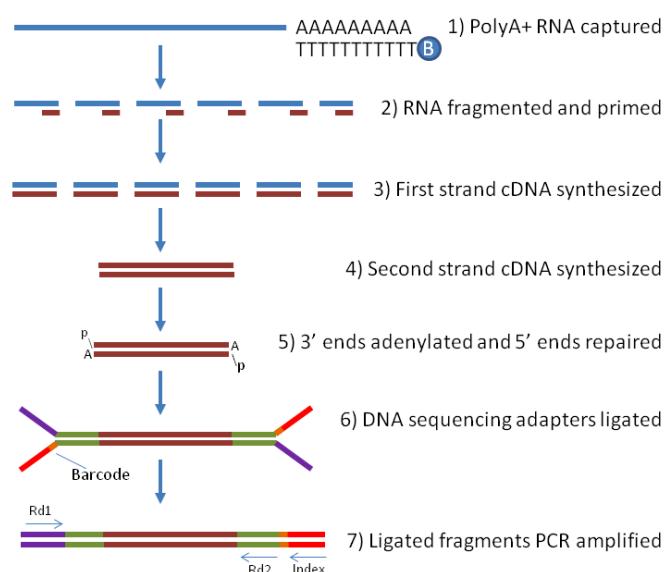
# Quality control of RNA sample

- Nanodrop quantitation
  - Standard equipment
  - Peaks at particular absorbance range can signal contamination
  - Can't distinguish between DNA, RNA, free nucleotides
- Qubit fluorometric quantitation
  - Use separate kits to measure RNA, DNA and protein individually
- Agilent bioanalyzer to assess integrity
  - RNA integrity number (RIN)

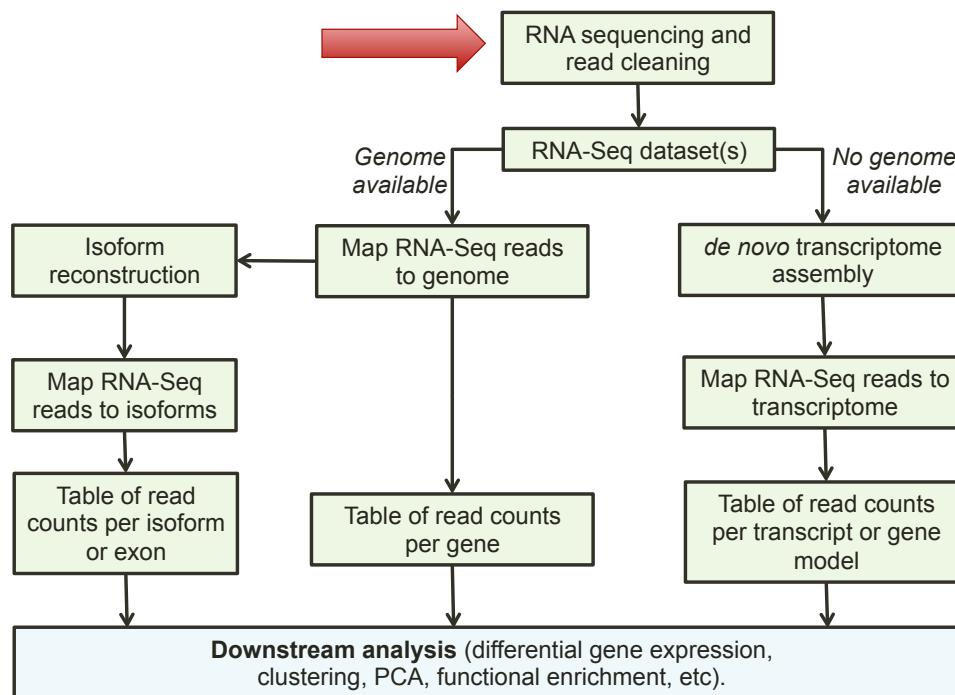


# Production of Illumina RNAseq data

- Assess quality & concentration
- DNase treatment
- Poly(A) selection
- Fragmentation
- cDNA synthesis
  - oligo(dT) & random hexamers
- Library preparation
- Sequencing



# RNA-seq analysis overview



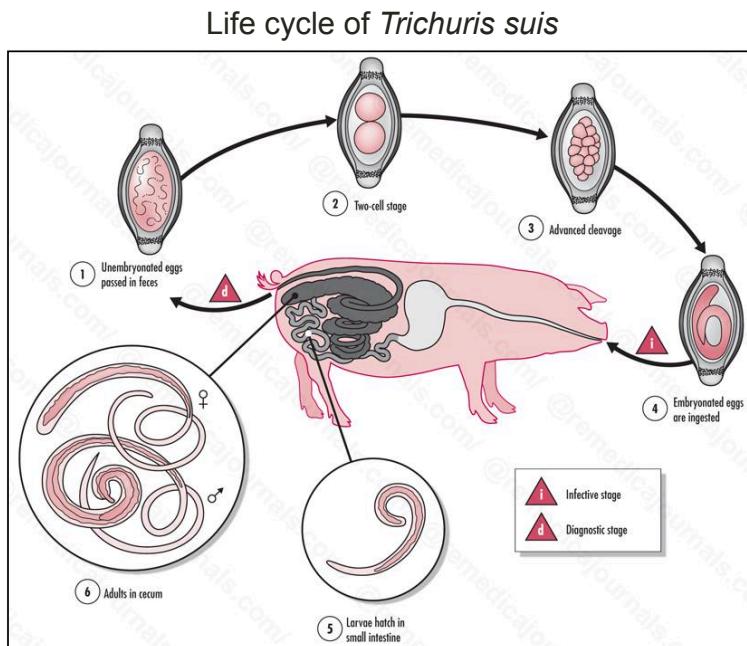
## Read pre-processing and filtering: a very stringent protocol

- 1) Adapter removal
- 2) Quality trimming & filtering
- 3) Contaminant filtering

Resource: <http://www.nature.com/nprot/journal/v8/n8/pdf/nprot.2013.084.pdf>, specifically Box 1



## Our “test” dataset



- Larval**
  - 10 days post inoculation (dpi), L2
  - 16 dpi, L3
  - 17 dpi, L3
  - 21 dpi, L4
- Adult**
  - 42 dpi, L5
  - Adult rep1
  - Adult rep2



## Our “test” dataset

300-500bp fragment

Read 1 → ----- ← Read 2

	L2_10d	L3_16d	L3_17d	L4_21d	L5_42d	L5_r163	L5_r179	Total
Total raw pairs	43,592,929	54,459,409	47,371,505	58,231,629	55,800,467	32,809,672	41,902,924	334,168,535
Downsampled raw pairs	4,435,622	5,511,063	4,817,349	5,891,002	5,644,329	3,337,590	4,258,806	33,895,761

- Counting reads in a bam file
 

```
samtools view -b -c input.bam
```

  - Divide by 2 to get pairs!
- Downsampling:
 

```
samtools view -b -s XX.XX -o output.bam input.bam
```

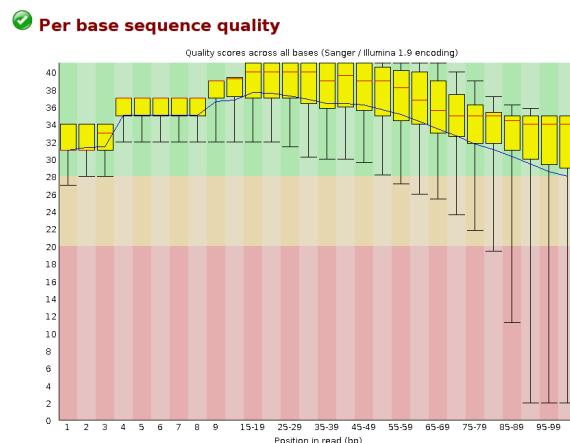
  - b: input is bam format
  - s: random down-sampling, integer before the decimal is seed for random number generator, after the decimal is the % reads to maintain
  - o: output file name
- Convert bam → fastq as before

Resource: <http://www.htslib.org/doc/samtools.html>



# Adapter detection

- Use fastqc to identify any adapter sequences that may need to be clipped



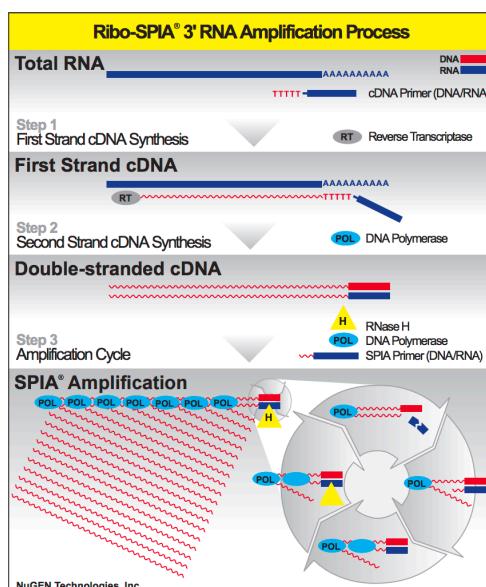
## [WARN] Overrepresented sequences

Sequence	Count	Percentage	Possible Source
CTTTGTGTTGATTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT	116516	0.13364093979553426	No Hit
AAAAAAAAAAAAA	90699	0.10402948606642146	No Hit

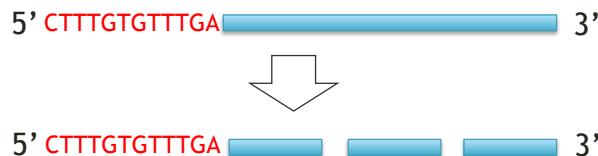
Resource: <http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>



# NuGEN Ovation RNAseq System V2



- Single Primer Isothermal Amplification protocol used in cDNA synthesis
  - SPIA adapters linked to primers
- Fragmentation following cDNA synthesis, so most reads won't have SPIA



Resource: [http://www.nugen.com/sites/default/files/M01114\\_v4.1%20-%20User%20Guide,%20Ovation%20RNA%20Amplification%20System%20V2.pdf](http://www.nugen.com/sites/default/files/M01114_v4.1%20-%20User%20Guide,%20Ovation%20RNA%20Amplification%20System%20V2.pdf)



# Adapter detection

- Checking for adapters in your file:

```
grep -B 1 -A 2 -colour "^\$CTTGTGTTGA" L2_10d.1.raw.fastq
```

- To count sequences with an adapter

```
grep -c "^\u0053\u0043\u0054\u0047\u0047\u0054\u0044\u0047\u0041" L2 10d.1.raw.fasta
```

Resource: [http://linuxcommand.org/man\\_pages/grep1.html](http://linuxcommand.org/man_pages/grep1.html)



## Adapter removal

- Tips:
    - Trimmomatic doesn't work well for short adapter sequences
    - clipping multiple adapters in one pass may not work well
  - Other options for adapter trimming:
    - Flexbar: <http://sourceforge.net/p/flexbar/wiki/Manual/>
      - Adapter detection & removal
      - Barcode detection, removal and read binning
      - Filtering reads with uncalled bases
      - Quality trimming and filtering
      - Length trimming / filtering
    - Cutadapt: <https://pypi.python.org/pypi/cutadapt/>
    - FASTX-Toolkit: [http://hannonlab.cshl.edu/fastx\\_toolkit/seq](http://hannonlab.cshl.edu/fastx_toolkit/seq)
    - Seq crumbs toolkit: [https://bioinf.comav.upv.es/seq\\_crumb](https://bioinf.comav.upv.es/seq_crumb)



# Removing SPIA adapters with Flexbar

- Command

```
flexbar --adapters  
Adapter.fasta --  
adapter-trim-end LEFT  
--min-read-length 60 --  
reads L2_10d.  
1.raw.fastq --reads2  
L2_10d.2.raw.fastq --  
target L2_10d --  
format=sanger --  
adapter-min-overlap 7
```

- Result:

- Clip adapters
- Filter reads with uncalled bases
- Remove any reads <60bp

```
Adapter removal statistics  
=====  
Adapter:          Overlap removal:   Full length:  
SPIA              1426360           1062690  
SPIARC             808949            635324  
  
Min, max, mean and median adapter overlap: 7 / 13 / 11 / 12  
  
Output file statistics  
=====  
Read file:          L2_10d.spia_1.fastq  
written reads       3991748  
skipped short reads 239258  
Read file 2:         L2_10d.spia_2.fastq  
written reads       3991748  
too short reads     249235  
Single read file:   L2_10d.spia_1_single.fastq  
written reads       194531  
Single read file 2: L2_10d.spia_2_single.fastq  
written reads       184554  
  
Filtering statistics  
=====  
Processed reads      8871244  
skipped due to uncalled bases 20170 (12773 uncalled in 10085 pairs)  
short prior adapter removal 0  
finally skipped short reads 488493  
Discarded reads overall 508663  
Remaining reads      8362581 (94% of input reads)  
  
Flexbar completed adapter removal.
```

Resource: <http://sourceforge.net/p/flexbar/wiki/Manual/>



# Quality trimming & filtering with Trimmomatic

- Command:

```
java -jar ~/bin/trimmomatic-0.33.jar PE -phred33  
L2_10d.spia_1.fastq L2_10d.spia_2.fastq L2_10d.1.fb-  
tm.fastq L2_10d.1.junk.fastq L2_10d.2.fb-tm.fastq  
L2_10d.2.junk.fastq ILLUMINACLIP:Adapters.fasta:2:30:10  
SLIDINGWINDOW:5:20 LEADING:20 TRAILING:20 MINLEN:60
```

- Result

- Clipping any remaining Illumina sequencing adapters
- Clipping any bases from the end of the reads with quality score <20
- Sliding window quality trim
- Removing any reads that are <60bp after clipping and trimming

- Program prints basic statistics to standard output

Resource: <http://www.usadellab.org/cms/?page=trimmomatic>



# Complexity filtering with seq-crumbs

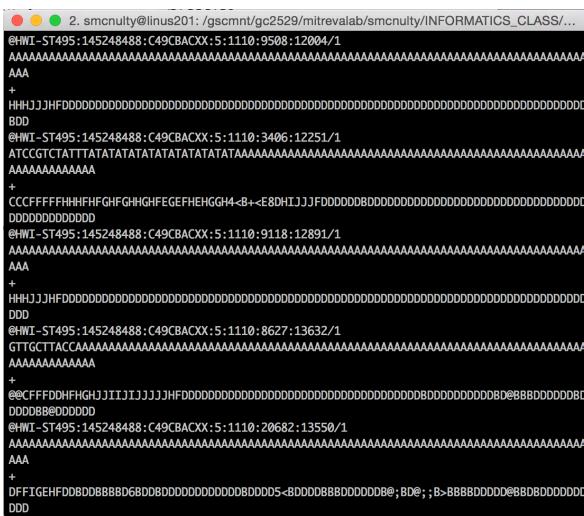
- Seq-crumbs interleave fastq files
    - `interleave_pairs -o L2_10d.int.fb-tm.fastq`  
`L2_10d.1.fb-tm.fastq`  
`L2_10d.1.fb-tm.fastq`
  - Filter low complexity reads
    - `filter_by_complexity -o L2_10d.int.fb-tm-sc.fastq`  
`--paired_reads --`  
`fail_drag_pair`  
`L2_10d.int.fb-tm.fastq`
  - Seq-crumbs de-interleave fastq files
    - `deinterleave_pairs -o L2_10d.1.fb-tm-sc.fastq`  
`L2_10d.2.fb-tm-sc.fastq`  
`L2_10d.int.fb-tm-sc.fastq`

Resource: [https://bioinf.comav.upv.es/seq\\_crumbz/available\\_crumbz.html](https://bioinf.comav.upv.es/seq_crumbz/available_crumbz.html)

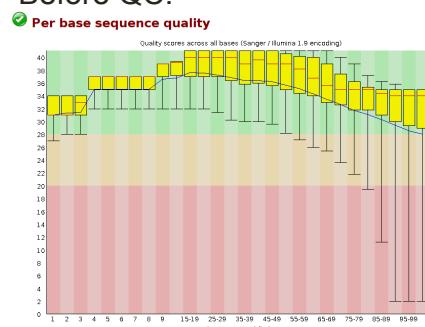
Q

## Quality control, reviewed

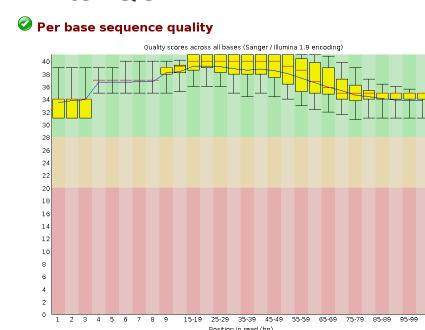
- Quality trimming/filtering
    - Adapter removal
    - Quality trimming
    - Length filtering
    - Complexity filtering
  - Result: confidence in sequence presented



### Before QC:



After QC:



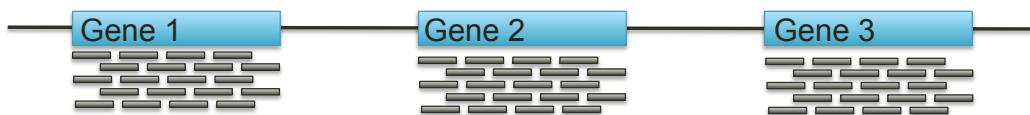
# Contaminant filtering

- Do I need to do contaminant filtering?
- Questions to consider:
  - Where did my worm live?
  - Is the host's genome available?
  - If not, what's the next best thing?
  - Is my worm easily isolated from its host?
  - What does my worm/host eat?
  - Is my worm easily rinsed/cleaned?
- What do you expect to see?



## Contaminant filtering with Bowtie2

- Bowtie for mapping when splicing IS NOT a consideration
  - SILVA rRNA: <http://www.arb-silva.de/>
    - “SILVA provides comprehensive, quality checked and regularly updated datasets of aligned small (16s/18s, SSU) and large subunit (23s/28s, LSU) ribosomal RNA sequences for all three domains of life”
  - Bacteria
    - GenBank bacterial database
    - Custom database (human microbiome project)

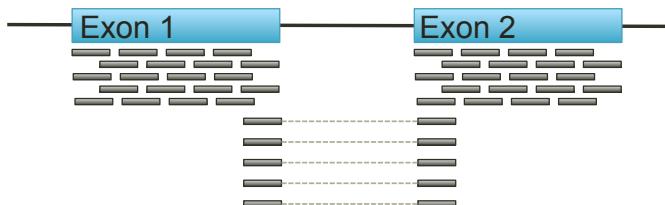


Resource: <http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml>



## Contaminant filtering with Tophat2

- Tophat for mapping when splicing IS a consideration
  - Bowtie aligns reads that fall neatly within exons
  - Tophat splits reads across introns/gaps
- Databases
  - Human
  - Host
    - Intermediate
    - Definitive
- Sources
  - Genbank / Refseq
  - Ensembl.org



Resource: <https://ccb.jhu.edu/software/tophat/manual.shtml>



## Remove contaminant reads

- Index database
  - bowtie2-build `Pig.fasta Pig.fasta`
- Map with bowtie
  - `bowtie2 -x Pig.fasta -1 L2_10d.1.fb-tm-sc.fastq -2 L2_10d.1.fb-tm-sc.fastq -S MapPig.sam`
- Map with tophat
  - `tophat2 -o L2_10d Pig.fasta L2_10d.1.fb-tm-sc.fastq L2_10d.1.fb-tm-sc.fastq`
- Counting mapped reads
  - For BAM file: `samtools view -c -F 4 accepted_hits.bam`
  - For SAM file: `samtools view -c -S -F 4 MapPig.sam`
- Remove contaminant reads and their mates as before
- Result:
  - High quality base calls
  - Confidence in the source of the reads

Resource: <https://broadinstitute.github.io/picard/explain-flags.html> (explanation of sam flags)



# Results of quality control

- Count the number of reads maintained at each step!
  - find . -name “\*1.clean.fastq” | xargs wc -l
  - Divide line count by 4 to get fastq entries

Downsampled read set:

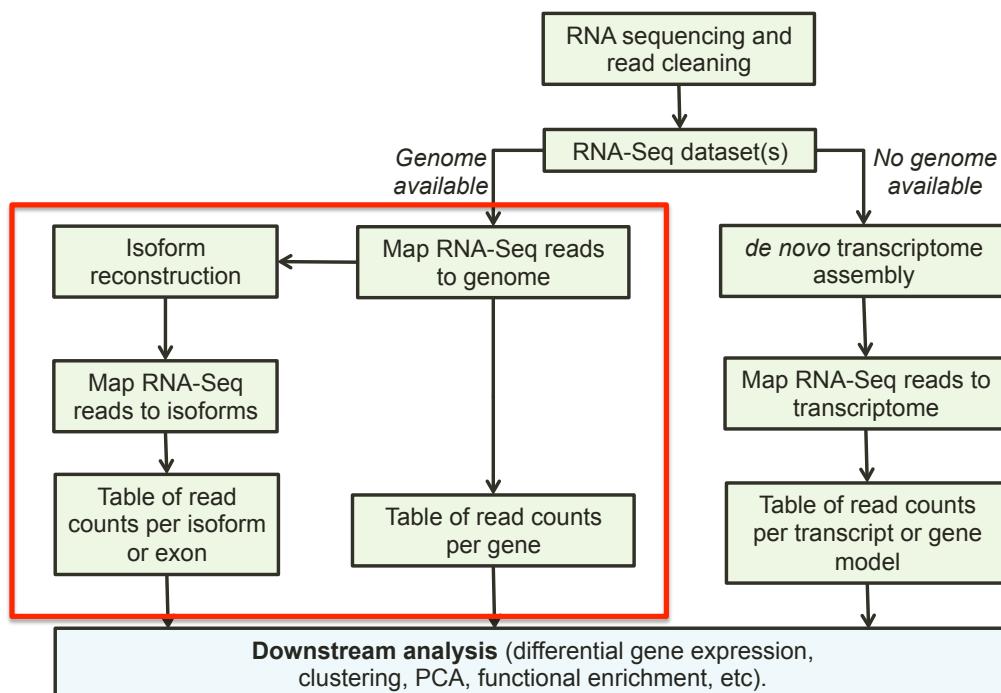
	L2_10d	L3_16d	L3_17d	L4_21d	L5_42d	L5_r163	L5_r179	Total
Raw pairs	4,435,622	5,511,063	4,817,349	5,891,002	5,644,329	3,337,590	4,258,806	33,895,761
Flexbar	3,991,748	4,878,344	4,298,728	5,270,820	5,009,942	2,530,803	3,826,835	29,807,220
Trimmomatic	3,110,420	4,007,562	3,385,936	4,226,000	4,165,397	2,220,509	3,021,273	24,137,097
SeqCrumbs	3,093,078	3,917,497	3,373,150	4,183,440	4,113,913	2,219,777	3,011,416	23,912,271
Contaminants	2,696,239	3,643,862	3,350,928	3,927,395	3,926,103	2,211,368	2,993,460	22,749,355
% maintained	60.80%	66.10%	69.60%	66.70%	69.60%	66.30%	70.30%	67.10%

Full read set:

	L2_10d	L3_16d	L3_17d	L4_21d	L5_42d	L5_r163	L5_r179	Total
Raw pairs	43,592,929	54,459,409	47,371,505	58,231,629	55,800,467	32,809,672	41,902,924	334,168,535
Flexbar	39,229,484	48,195,339	42,272,646	52,090,873	49,524,734	24,877,392	37,657,504	293,847,972
Trimmomatic	30,586,411	40,437,016	33,302,203	42,655,938	41,935,364	21,862,295	29,745,662	240,524,889
SeqCrumbs	30,416,334	39,426,836	33,176,521	42,179,989	41,354,287	21,854,889	29,648,071	238,056,927
Contaminants	26,501,312	36,740,860	32,956,606	39,675,217	39,508,530	21,780,296	29,469,388	226,632,209
% maintained	60.79%	67.46%	69.57%	68.13%	70.80%	66.38%	70.33%	67.82%



## RNA-seq analysis overview



## Section 2: Transcriptome

### Module 1: Genome based RNA-seq analyses

- 1) Splice-aware alignment and verification
- 2) Genome-assisted transcript assembly
- 3) Counting reads in features for differential expression analyses

Resource: <http://www.nature.com/nprot/journal/v8/n9/pdf/nprot.2013.099.pdf>



## Where to find a reference genome

- Sources:
  - Genbank/Refseq
  - Nematode.net
  - Wormbase.org
- Requirements:
  - Assembly fasta
  - GFF3
  - Functional annotation or protein/cds fasta

**Nematode.net**

**Our Mission:**

Parasitic roundworms (nematodes) of humans, livestock and other animals cause diseases of major socio-economic importance globally. They have a major, long-term impact on health, welfare and productivity of humans and animals, particularly in children. The World Health Organization (WHO) estimates that 2.9 billion people are infected with nematodes. Furthermore, the current financial losses caused by parasites to agriculture worldwide (domesticated animals and crops) have a major impact on farm profitability and exacerbate the global food shortage.

Methods available for the control of the parasitic nematode infections are mainly based on chemical treatment (anthelmintics), non-chemical management practices, immune modulation and biological control. However, the incomplete protective response of the host and acquisition of anthelmintic resistance by an increasing number of parasitic nematodes hampered what used to be effective and long-lasting control strategies. Moreover, the use of such drugs poses major risks of residue problems in meat, milk and the environment.

Therefore, the challenges to improve control of parasitic nematode infections are multi-fold and no single category of information will meet them all. However, new information, such as nematode genomics, functional genomics and proteomics, can strengthen basic and applied biological research aimed at developing improvements. Our MISSION is through integrated approaches to accelerate progress towards developing more efficient and sustainable parasitic nematode control programs.

**Bioinformatics Workshop for Helminth Genomics:**

The Bioinformatics Workshop for Helminth Genomics (10-11 September 2015) will teach practical computational skills that should be of value to all nematode biologists. Whether you are an aspiring coder yourself, or you just want to better understand how data is processed from raw sequence to a final result that supports an interesting story in a publication, this course is for you! Click the link above for more information.



# GFF3 format

3. ec2-user@ip-172-31-38-111:~/WORKSHOP_RESOURCES/Section_2/module_1 (ssh)								
#gff-version 3								
T_suis-1_0.Cont72	Final_set	gene	74794	75765	.	-	.	ID=D918_GENE0001:gene;Name=D918_09719
T_suis-1_0.Cont72	Final_set	mRNA	74794	75765	.	-	.	ID=D918_GENE0001.1:mRNA;Parent=D918_GENE0001:gene;Name=D918_09719
T_suis-1_0.Cont72	Final_set	exon	74794	75765	.	-	.	ID=D918_GENE0001.1:exon;Parent=D918_GENE0001.1:mRNA
T_suis-1_0.Cont72	Final_set	CDS	74794	75765	.	-	0	ID=D918_GENE0001.1:CDS;Parent=D918_GENE0001.1:mRNA
T_suis-1_0.Cont40	Final_set	gene	395444	417867	.	+	.	ID=D918_GENE0002:gene;Name=D918_08404
T_suis-1_0.Cont40	Final_set	mRNA	395444	417867	.	+	.	ID=D918_GENE0002.1:mRNA;Parent=D918_GENE0002:gene;Name=D918_08404
T_suis-1_0.Cont40	Final_set	exon	395444	395926	.	+	.	ID=D918_GENE0002.1:exon;Parent=D918_GENE0002.1:mRNA
T_suis-1_0.Cont40	Final_set	exon	396038	396514	.	+	.	ID=D918_GENE0002.1:exon;Parent=D918_GENE0002.1:mRNA
T_suis-1_0.Cont40	Final_set	exon	396937	397314	.	+	.	ID=D918_GENE0002.1:exon;Parent=D918_GENE0002.1:mRNA
T_suis-1_0.Cont40	Final_set	exon	397730	397910	.	+	.	ID=D918_GENE0002.1:exon;Parent=D918_GENE0002.1:mRNA
T_suis-1_0.Cont40	Final_set	exon	398750	399097	.	+	.	ID=D918_GENE0002.1:exon;Parent=D918_GENE0002.1:mRNA
T_suis-1_0.Cont40	Final_set	exon	417155	417867	.	+	.	ID=D918_GENE0002.1:exon;Parent=D918_GENE0002.1:mRNA
T_suis-1_0.Cont40	Final_set	CDS	395444	395926	.	+	0	ID=D918_GENE0002.1:CDS;Parent=D918_GENE0002.1:mRNA
T_suis-1_0.Cont40	Final_set	CDS	396038	396514	.	+	0	ID=D918_GENE0002.1:CDS;Parent=D918_GENE0002.1:mRNA
T_suis-1_0.Cont40	Final_set	CDS	396937	397314	.	+	0	ID=D918_GENE0002.1:CDS;Parent=D918_GENE0002.1:mRNA
T_suis-1_0.Cont40	Final_set	CDS	397730	397910	.	+	0	ID=D918_GENE0002.1:CDS;Parent=D918_GENE0002.1:mRNA
T_suis-1_0.Cont40	Final_set	CDS	398750	399097	.	+	2	ID=D918_GENE0002.1:CDS;Parent=D918_GENE0002.1:mRNA
T_suis-1_0.Cont40	Final_set	CDS	417155	417867	.	+	2	ID=D918_GENE0002.1:CDS;Parent=D918_GENE0002.1:mRNA
T_suis-1_0.Cont17	Final_set	gene	633392	637389	.	-	.	ID=D918_GENE0003:gene;Name=D918_05776
T_suis-1_0.Cont17	Final_set	mRNA	633392	637389	.	-	.	ID=D918_GENE0003.1:mRNA;Parent=D918_GENE0003:gene;Name=D918_05776
T_suis-1_0.Cont17	Final_set	exon	633392	633550	.	-	.	ID=D918_GENE0003.1:exon;Parent=D918_GENE0003.1:mRNA
T_suis-1_0.Cont17	Final_set	exon	633607	633785	.	-	.	ID=D918_GENE0003.1:exon;Parent=D918_GENE0003.1:mRNA
T_suis-1_0.Cont17	Final_set	exon	633840	634033	.	-	.	ID=D918_GENE0003.1:exon;Parent=D918_GENE0003.1:mRNA

- Column 1: contig or scaffold
  - Must match the assembly fasta!
- Column 3: feature
  - CDS, coding\_exon
- Column 9: mRNAs/genes the feature belongs to

Resource: <http://www.usadellab.org/cms/?page=trimmomatic>

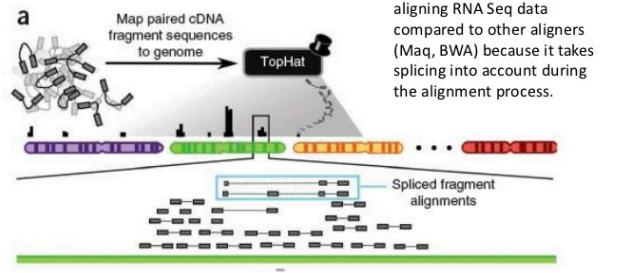


## Aligning reads with Tophat2

- Commands:

```
bowtie2-build  
D918.fa D918.fa
```

```
tophat2 -o L2_10d -G  
D918.gff3  
D918.fa .../module_0/  
L2_10d.  
1.clean.fastq .../  
module_0/L2_10d.  
2.clean.fastq
```



- G option:

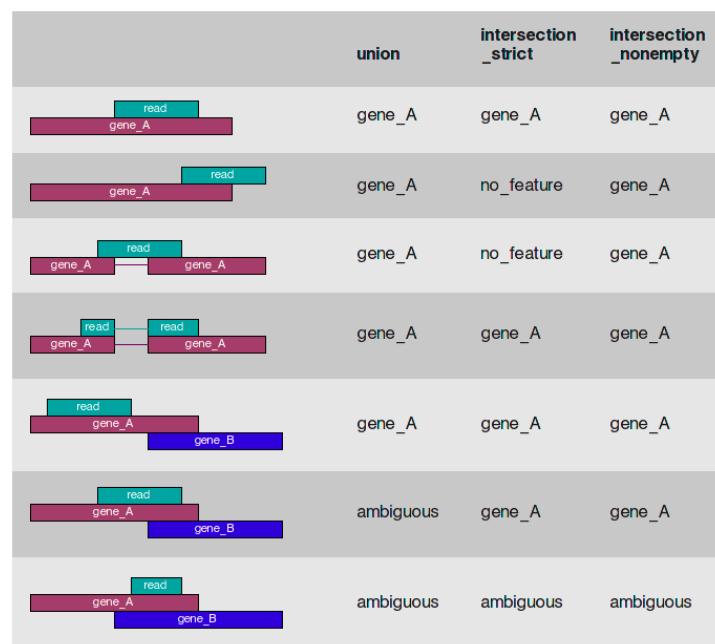
- If this option is provided, TopHat will first extract the transcript sequences and use Bowtie to align reads to this virtual transcriptome. Only the reads that do not fully map to the transcriptome will then be mapped on the genome. The reads that did map on the transcriptome will be converted to genomic mappings (spliced as needed) and merged with the novel mappings and junctions in the final tophat output"

Resource: <https://ccb.jhu.edu/software/tophat/manual.shtml>



# Counting reads within features with htseq-count

- Command:
  - htseq-count -f bam  
-r pos -t CDS -i Parent  
`accepted_hits.bam`  
`D918.gff3` >  
`L2_10d.htseq.txt`
- Arguments
  - f: format
    - sam or bam
  - r: order
    - name or pos
  - t: feature type
    - coding\_exon
    - exon
    - CDS
  - i: feature ID
    - Parent



Resource: <http://www-huber.embl.de/users/anders/HTSeq/doc/count.html>



## htseq-count output

	L2_10d	L3_16d	L3_17d	L4_21d	L5_42d	L5_r163	L5_r179
D918_00003	34	36	28	42	112	163	297
D918_00007	0	3	0	0	97	5	25
D918_00013	273	584	251	372	417	144	232
D918_00014	24	62	39	90	337	381	517
D918_00015	345	615	488	404	638	298	415
D918_00016	1801	1672	3838	1870	2614	1923	3446
D918_00017	3091	3833	4334	4376	3333	2011	2954
D918_00018	706	1680	1252	2430	2285	737	1040
D918_00019	3912	3062	1400	3638	3894	1643	1994
D918_00020	928	2060	2012	1971	3821	6971	3676
//							
alignment_not_unique	221176	839400	567739	890856	1011380	465826	512410
ambiguous	268632	549686	367069	470060	639345	330250	336040
no_feature	2888141	5856583	3677885	4318280	5470650	2710622	3702874
not_aligned	0	0	0	0	0	0	0
too_low_aQual	0	0	0	0	0	0	0

- All values should be integers
- 60-80% mapping rate is considered good
  - Sum counts for all genes and divide by cleaned read pairs

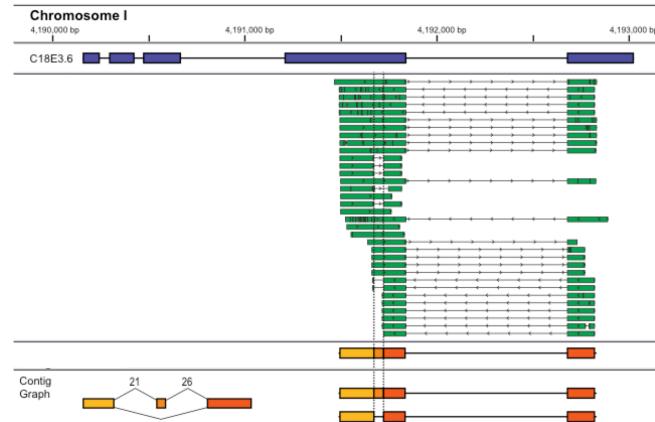
Resource: <http://www-huber.embl.de/users/anders/HTSeq/doc/count.html>



# Cufflinks: genome-assisted transcript assembly

- Assembly transcripts for each sample separately using Cufflinks

```
cufflinks -o CuffOUTPUT  
accepted_hits.bam
```



- Create a file that lists the assembly file for each sample

```
find . -name  
"transcripts.gtf" >  
assemblies.txt
```

- Run cuffmerge to create a single merged transcriptome annotation

```
cuffmerge -g genome.gtf  
-s genome.fasta  
assemblies.txt
```

- Creates an output called merged.gtf

- Use gffread to print a fasta file of our transcripts

```
gffread merged.gtf -g genome.fasta
```

```
-w Transcripts.fa
```

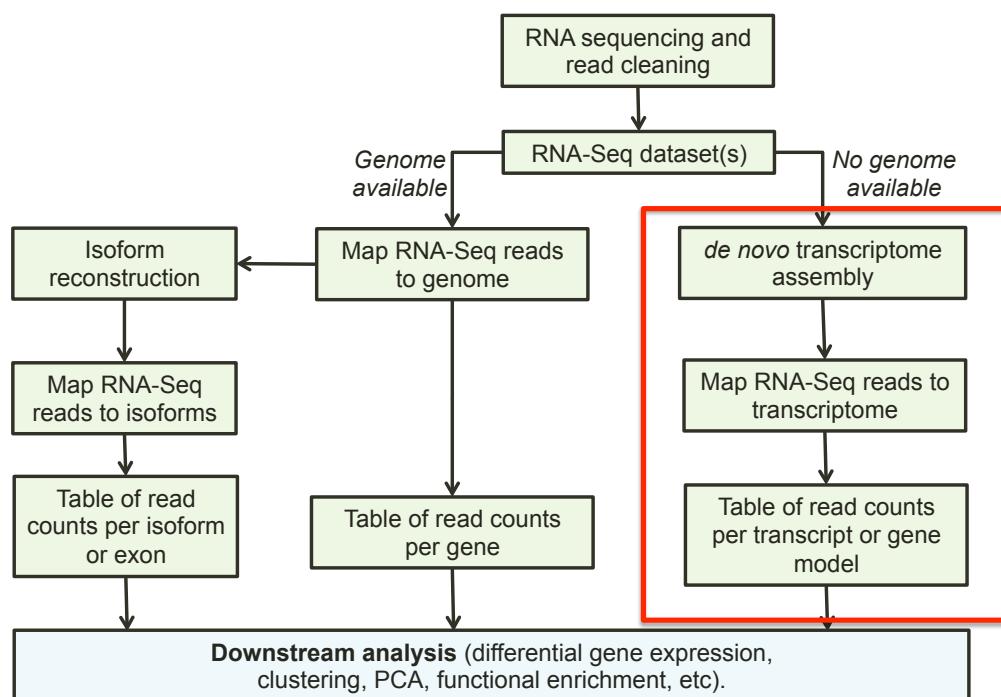
- Options:

- U: discard single-exon transcripts
- M: collapse matching transcripts
- K: collapse shorter, fully contained transcripts

Resource: <http://www.nature.com/nprot/journal/v7/n3/pdf/nprot.2012.016.pdf>



# RNA-seq analysis overview



---

## Section 2: Transcriptome

### Module 2: *De novo* transcript assembly

---

- 1) Digital read normalization
- 2) *De novo* transcript assembly
- 3) Post-assembly filtering
- 4) Mapping raw reads to the assembly



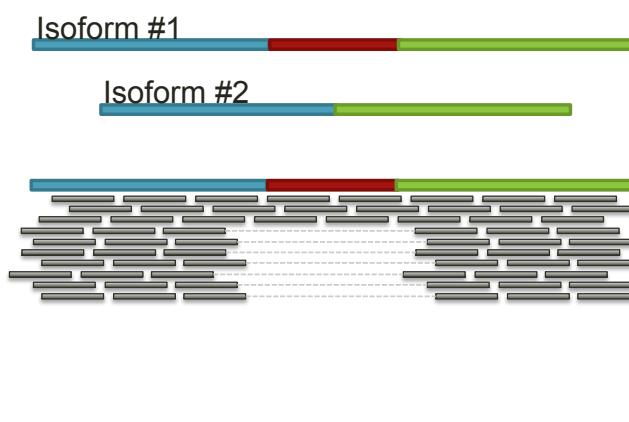
---

## Problems with *de novo* transcript assembly

---

	L2_10d	L3_16d	L3_17d	L4_21d	L5_42d	L5_r163	L5_r179	Total
clean read pairs	26,501,312	36,740,860	32,956,606	39,675,217	39,508,530	21,780,296	29,469,388	<b>226,632,209</b>

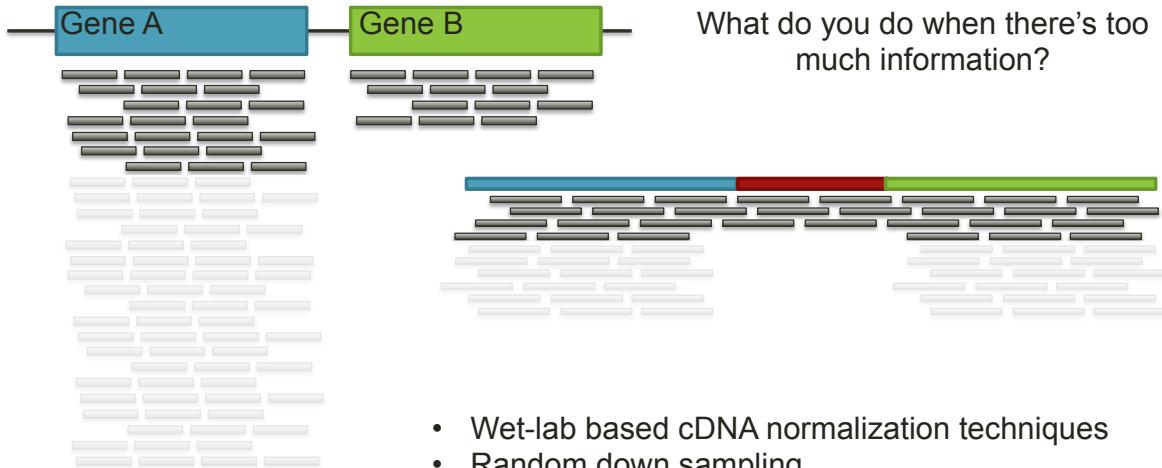
- Lots and lots of “puzzle pieces”
- Varying transcript abundance
- Alternative splicing
- Differential gene expression



Resource: <http://arxiv.org/pdf/1203.4802v2.pdf>



# Data reduction methods

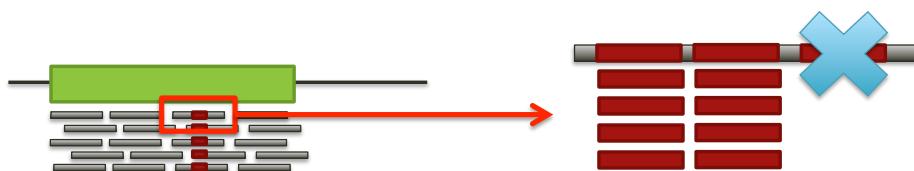


Resource: <http://arxiv.org/pdf/1203.4802v2.pdf>



## Digital read normalization

- Solution: “a computational algorithm that systematizes coverage in shotgun sequencing data sets, thereby decreasing sampling variation, discarding redundant data, and removing the majority of errors”
- Method:
  - K-mer abundance correlates well with mapping-based estimates of read coverage
  - K-mers tend to have similar abundances within a read since they originate from the same DNA/RNA molecule



- Estimate k-mer abundance (i.e., read coverage) to make the following determination

```
for read in dataset:  
    if estimated_coverage(read) < C:  
        accept(read)  
    else:  
        discard(read)
```

Resource: <http://arxiv.org/pdf/1203.4802v2.pdf>



---

## Normalization software

---

- Khmer: <http://khmer.readthedocs.org/en/v1.4.1/>
  - Detailed protocol:  
<http://khmer-protocols.readthedocs.org/en/v0.8.2/mrnaseq/2-diginorm.html>
    - Decide which reads need to be maintained
    - Trim off low abundance parts of high coverage reads (i.e., errors)
    - Re-pair reads
- Trinity implementation:
  - [https://trinityrnaseq.github.io/trinity\\_insilico\\_normalization.html](https://trinityrnaseq.github.io/trinity_insilico_normalization.html)
- For an explanation of the difference, see this blog post:
  - <http://ivory.idyll.org/blog/trinity-in-silico-normalize.html>



---

## *De novo* transcript assembly with Trinity

---

- Trinity approach
  - Inchworm: assembles reads into unique sequences of transcripts, often generating full-length transcripts for a dominant isoform, and reporting unique portions of alternatively spliced transcripts
  - Chrysalis: clusters inchworm contigs into complete de Bruijn graphs for each cluster
  - Butterfly: processes the individual graphs to report full-length transcripts for alternatively spliced isoforms
- Trinity command:  

```
Trinity --seqType fq --max_memory XXG --left AllLeft.fastq  
--right AllRight.fastq --normalize_reads -output TRINITY
```
- Time and memory:
  - Approximately 1G of RAM per million read pairs
  - Approximately 0.5-1h per million read pairs



## Trinity output

- Trinity will create a Trinity.fasta output file in the specified output directory
- Trinity groups transcripts into clusters based on shared sequence content. These clusters are loosely referred to as “genes” or “unigenes”. This information is coded in the trinity accession.

```
● ○ ● 1. ec2-user@ip-172-31-38-111:~/WORKSHOP_RESOURCES/Section_2/mod...
[ec2-user@ip-172-31-38-111 TRINITY]$ head -n 25 Trinity.fasta
>TR1|c0_g1_i1 len=262 path=[240:0-261] [-1, 240, -2]
TAATCTGTTTCGAAATGGTTCTCCCTTTCTGTTGGTACCTACCAAGCAAATGGAC
TGCACTTATCTGCATTCAAGCATTCTAGAGCTTATCGTCCGAAGGAGCATATAGCTG
CTTAATAAGCGTTAATCTTCGTCAGATGTTCTGTTGGTCCCTTATGCTCTG
CTAACGATTCAATAGTTCATCATCGCTCTTATTGACGGCCCTGTTCCGAGCTG
AGCGGATGTCCTGTCTACTG
>TR2|c0_g1_i1 len=255 path=[233:0-254] [-1, 233, -2]
GGAAGCTTTAGGGAAATAAATTCTGCTGATTTGGCTTACGCGTTATCCAACGAGCG
TAGCATTTAGTTGGGCATAAGTAACATCGAATCGAAATCTTGCAGAAATGCTTTT
GTGCATCTACTGTTGTCGCTAGCGCTGCATTAAAATCAAGTAACCTGACAAGTACT
TTGATTATGCTTGAATAATTCTTCGACTTAAACGTATATTATTAGTTGGCTGGT
CATTTAGCCTTGAA
>TR3|c0_g1_i1 len=418 path=[791:0-417] [-1, 791, -2]
GGTAACGCCTTGGGAACCCCTTTCTTAATAAAGACTTTGGTCATGTTCAACGAGC
CTAATTTATCTCTGTTGAAAAGTGAACAAAGATAAGATGGCGTCGCTAAAGGTGAAGC
TGTGTTATCAGACAACTGATAATCCAATAAAAATGTTGATAGATTAAAAGATAACGT
ATGTGCGAGATAATTAAAATTGCTAAAGTACAAGCAATCCCTAGTGTCTCTC
TGCTTGTCTGAGCTTAGGTGATTCACTTGCTAACGCCCTAACCAATCAATGATGGAAGG
AAGCGCTTGTGACCTTGCTGCTGCTGTTAGTCAGTTCGGAACTGTCCTCCCTAT
ATCGCTAGATTCAATATGAATAGTAGATTGAAAGGTACGTCAATTGATTGCTATA
```

[http://trinityrnaseq.github.io/#trinity\\_output](http://trinityrnaseq.github.io/#trinity_output)



## Assembly statistics

- Command:

```
perl ~/bin/
trinityrnaseq-2.0.6/util/
TrinityStats.pl
Trinity.fasta
```

- In a perfect assembly, “unigenes” = expressed genes
- Why are there so many genes/transcripts?
  - Fragmentation
  - Low-confidence transcripts

### “Test” assembly:

```
#####
## Counts of transcripts, etc.
#####
Total trinity 'genes': 48361
Total trinity transcripts: 74070
Percent GC: 45.34

#####
Stats based on ALL transcript contigs:
#####

Contig N10: 2736
Contig N20: 2035
Contig N30: 1646
Contig N40: 1351
Contig N50: 1102

Median contig length: 557
Average contig: 797.01
Total assembled bases: 59034791

#####
## Stats based on ONLY LONGEST ISOFORM per 'GENE':
#####

Contig N10: 2265
Contig N20: 1676
Contig N30: 1302
Contig N40: 1036
Contig N50: 829

Median contig length: 451
Average contig: 648.84
Total assembled bases: 31378557
```

Resource: [http://trinityrnaseq.github.io/#trinity\\_output](http://trinityrnaseq.github.io/#trinity_output)



## Assembly filtering

- Align reads and estimate abundance

```
perl ~/bin/trinityrnaseq-2.0.6/
util/
align_and_estimate_abundance.pl --
transcripts Trinity.fasta --seqType
fq --left ../AllLeft.fastq --
right ../AllRight.fastq --
est_method RSEM --output_dir RSEM
--aln_method bowtie2 --
prep_reference
```

- Filter lowly supported transcripts

```
perl ~/bin/trinityrnaseq-2.0.6/
util/filter_fasta_by_rsem_values.pl
--rsem_output=RSEM.isoforms.results
--fasta=../Trinity.fasta --
output=Trinity.filtered.fasta --
tpm_cutoff=1.0 --isopct_cutoff=1.00
```

*Paragonimus kellicotti* assembly:

	Unfiltered	Filtered
# unigenes	153,461	59,050
# transcripts	251,721	91,029
Ave transcript size	460 bp	563 bp
Alternative splicing	24.8% of unigenes, ave 3.6, max 85	24.4% of unigenes, ave 3.2, max 20
% pairs mapped	68.3%	66.3%

Resource: [http://trinityrnaseq.github.io/analysis/abundance\\_estimation.html](http://trinityrnaseq.github.io/analysis/abundance_estimation.html)



## Feature counting for differential expression

- Prepare reference

```
perl ~/bin/trinityrnaseq-2.0.6/util/
align_and_estimate_abundance.pl --transcripts
Trinity.filtered.fasta --est_method RSEM --aln_method bowtie2
--prep_reference
```

- Align reads and estimate abundance

```
perl ~/bin/trinityrnaseq-2.0.6/util/
align_and_estimate_abundance.pl --transcripts
Trinity.filtered.fasta --seqType fq --est_method RSEM --
aln_method bowtie2 --left ../../module_0/L2_10d.
1.clean.fastq --right ../../module_0/L2_10d.2.clean.fastq
--output_dir L2_10d
```

- Join the abundance values for each sample into matrix for DESeq2

```
perl ~/bin/trinityrnaseq-2.0.6/util/
abundance_estimates_to_matrix.pl --est_method RSEM L2_10d/
RSEM.genes.results L3_16d/RSEM.genes.results ...
```

Resource: [http://trinityrnaseq.github.io/analysis/diff\\_expression\\_analysis.html](http://trinityrnaseq.github.io/analysis/diff_expression_analysis.html)



# Feature counting for differential expression

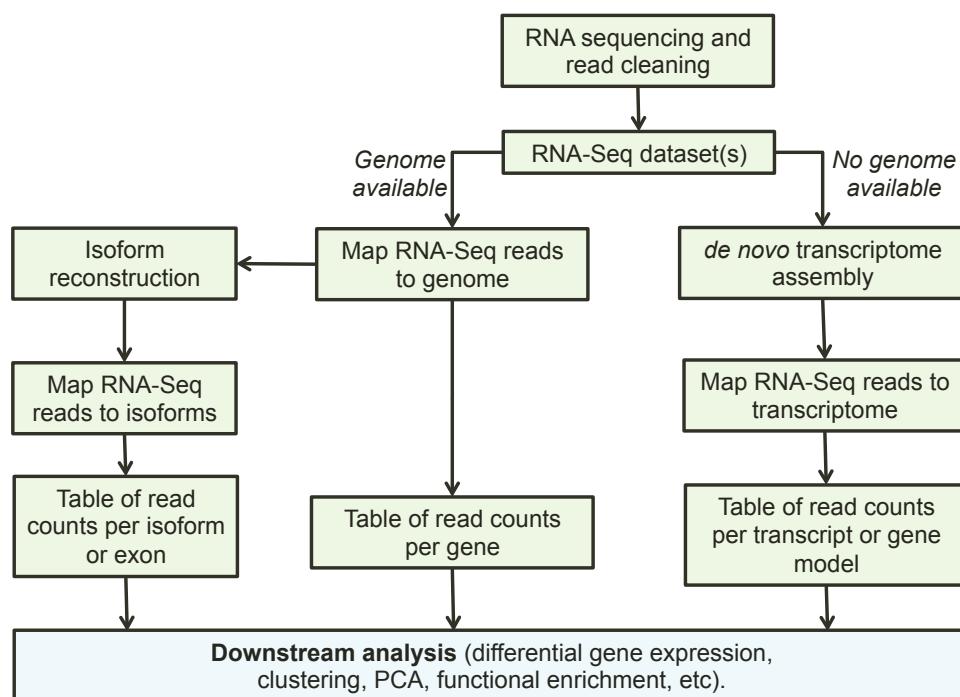
*Cooperia punctata* count table

	HIGH.genes. results	LOW.genes. results	UntreatedA.genes. results	UntreatedB.genes. results
comp197262_c2	53.02	51.97	24	107
comp196358_c0	90	125	104	91
comp194909_c0	3	2	0	79.07
comp189445_c0	15	5	7	15
comp199614_c0	19	23	24.67	18.89
comp191897_c2	16	20	26	3
comp196155_c1	223	283	119	467
comp196537_c0	74.2	98	38.67	200.96
comp194722_c1	11	6	1	33
comp200992_c1	9.24	21.98	27	11
comp189025_c0	57993.94	35917.49	21809.97	76141.69
comp195426_c0	32	74.17	52.45	100.2
comp197998_c0	27	8	12	13
comp201556_c2	22	19	22	25

Resource: [http://trinityrnaseq.github.io/analysis/diff\\_expression\\_analysis.html](http://trinityrnaseq.github.io/analysis/diff_expression_analysis.html)



# RNA-seq analysis overview



---

**Section 2: Transcriptome**  
**Module 3: Expression and differential expression**

---



---

## **Introduction - Expression and differential expression**

---

- For this module, we will be off of the server and working directly on your laptops.
- We will use data files that you downloaded using scp yesterday, which should be saved in ~/Desktop/WORKSHOP\_RESOURCES/Section\_2/module\_3/. Please check that you have downloaded files and folders to this directory.
- Raw data was produced in the previous modules.
- You should already have both RStudio and MS Excel installed on your laptops, as requested before the class started.



# Differential gene expression software

- Calling differentially expressed genes is a complicated statistical problem.
- “Dispersion” of a gene or a sample is used to estimate baseline (within-replicate) variability, and is essential for accurate statistical measurement. Genes with high inter-replicate variability should not be considered “differential”.
- Some measure of dispersion is calculated by all widely-accepted differential callers, but they all calculate it in slightly different ways.
- Three software packages are primarily used: **DESeq**, **EdgeR**, and **CuffDiff**. Others include SAMseq, baySeq, NOIseq, and EBSeq.
- **DESeq** and **EdgeR** are the two most commonly used differential gene expression calculation packages. These produce similar overall results in terms of final gene lists.

## How to choose a differential expression caller

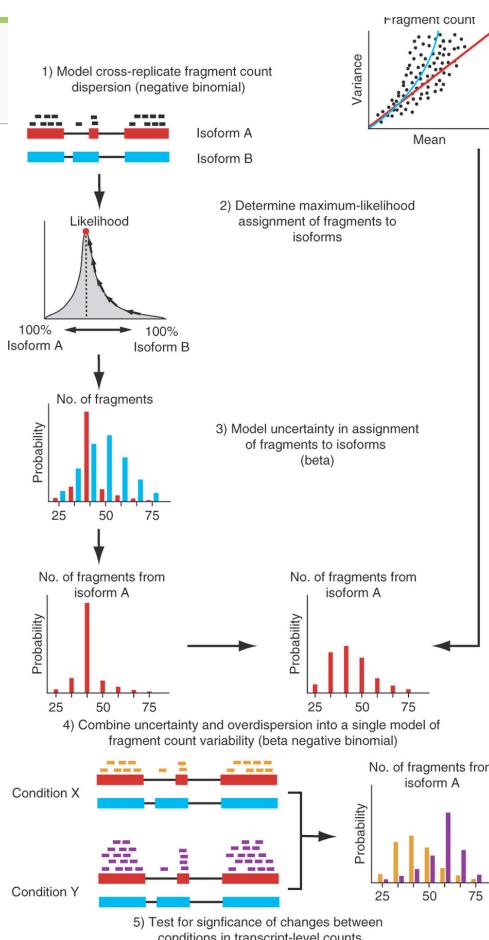
- The primary practical difference between **DESeq** and **EdgeR** is sensitivity (i.e. the number of genes called differential).
- If you are interested in transcript / isoform data, then use **CuffDiff**. CuffDiff tends to be very stringent (fewer differentially expressed genes than DESeq or EdgeR).
- **SAMseq** can be useful for cross-sample differential expression calling, but should not be used for two-sample comparisons.
- Having a larger set of differentially expressed genes is not necessarily better!
- More differentially expressed genes = more false positives, and a larger set of genes to summarize for biological interpretation.

<http://bib.oxfordjournals.org/content/early/2013/12/02/bib.bbt086.long>



## CuffDiff

- CuffDiff considers read counts per exon, and can identify significant changes in exon use and isoform abundance for the same gene.
- This is useful (a) for model organisms where there is known functional significance for specific exons/isoforms or for (b) for studies of a subset of specific genes of interest.
- At a genome-wide level, quantifying differential exon usage complicates downstream analysis without providing practically useful data.
- For example, it is difficult to perform genome-wide functional enrichment testing on differentially expressed isoforms, since multiple isoforms from the same gene can contribute to enrichment scores.



[http://www.nature.com/nbt/journal/v31/n1/fig\\_tab/nbt.2450\\_F2.html](http://www.nature.com/nbt/journal/v31/n1/fig_tab/nbt.2450_F2.html)

## Replicate considerations

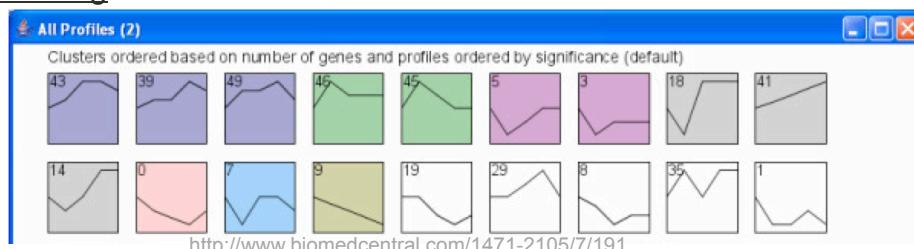
- At least triplicate is preferred for accurate analysis.
- Some samples may be lost due to very high variability from other replicates or low quality RNA, so duplicate is risky (single-replicate produces unreliable statistics).
- Collecting the replicates by repeating an experiment at a later time almost never works for helminth studies.
- Both DESeq and EdgeR *can* be executed with single replicates, but use different statistical models.
- Another program called **GFOOLD** is designed specifically for single-replicate samples, but these comparisons with any software are not confident without additional validation (e.g. qPCR of identified genes).
- Track metadata carefully whenever possible. E.g., the number of worms collected, whether there is a possibility of having mixed samples (male and female, L3 and L4, etc), time of sampling, etc. This may help to explain within-replicate variability in some cases.



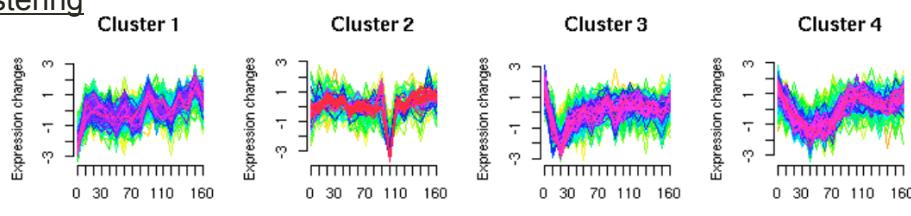
## Gene clustering

- Another analysis approach is to cluster samples based on their overall expression patterns across all available RNA-Seq datasets.
- While this is useful for grouping and classifying genes, the clusters only consider the pattern and do not consider whether the genes are statistically differentially expressed.
- One tool called Short Time Series Expression Miner (STEM) clustering will also identify over-represented patterns, representing clusters of probable biological significance.

### STEM Clustering



### Mfuzz Clustering



<http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2139991/>



## Differential gene expression measurement

Experimental design considerations: What are the samples you want to compare? What approach will you use to compare them?

Example 1: Treatment(s) vs Control

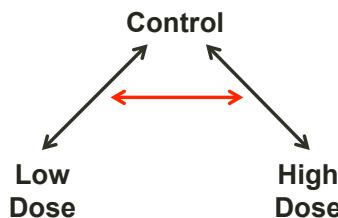
### 1A. Simple treatment / control pair:



- Which genes are high in treatment (upregulated) or lower in treatment (downregulated)?

### 1B. Control vs multiple treatments

(e.g. high and low doses of a drug treatment)



- Which genes are upregulated or downregulated by both treatments, and which ones are only differentially regulated by high-dose treatment but not low?

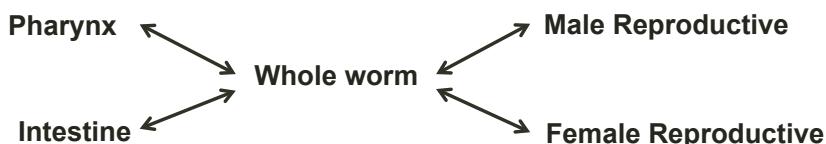


## Differential gene expression measurement

Example 2: Tissue-based (unordered, multiple samples)

e.g. Whole-worm, intestine, pharynx, and male and female reproductive tissue.

### 2A. Each compared to whole-worm:



- What are the tissue-specific overexpressed genes relative to the whole-worm sample?

### 2B. Each compared to all other tissues:



- What are the tissue-specific overexpressed genes relative to the other sampled tissues?

### 2C. Cross-sample combinatorial comparisons

- Some cross-sample differential expression callers (e.g. SAMSeq) can identify combinations of samples with upregulation (e.g. upregulated in both pharynx and intestine relative to other tissues).



## Differential gene expression measurement

Example 3: Stage-based (time series) data  
(e.g. L2, L3, L4, L5 larvae)

3A Pairwise : L2 ↔ L3 ↔ L4 ↔ L5

- Which genes are upregulated in one stage vs its surrounding stage(s)?

3B Grouped : L2    L3 ←→ L4    L5

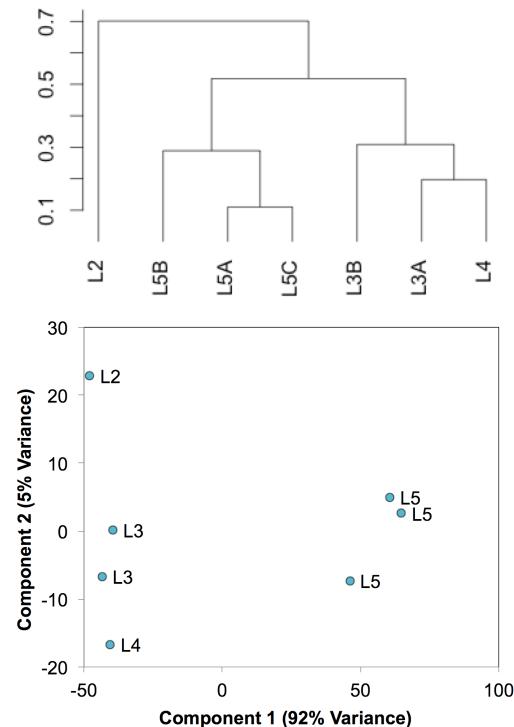
- Which genes are upregulated in early stages relative to late stages?
- Stages are treated as pseudo-replicates for each other.

3C Individual : L2 ←→ L3    L4    L5

L5 ←→ L2    L3    L4

Etc.

- Which genes are upregulated in one stage relative to all others?



## Using RStudio



- R is a free software environment for statistical computing and graphics.
- RStudio is a set of integrated tools to make R much easier to use.
- “Packages” of existing software can be downloaded, installed, and loaded easily.
- Many bioinformatics tools (especially for statistics analysis) are available exclusively in R.
- You can typically work with R by modifying existing scripts, most of which can be downloaded from manuals or other internet resources.
- In this module, we will learn how to use R studio to:
  - Install libraries, set the working directory and input files
  - Run DESeq2 for differential gene expression analysis
  - Run PCA and hierarchical clustering
  - Run GOSTATS for enrichment of differentially expressed genes



# Using RStudio

- The RStudio interface is split into four windows.

- If you only download R, then you will only have the console to work with.

The screenshot shows the RStudio interface with the following components:

- Script Window:** Displays a script named "Helminth\_Genomics\_Workshop\_Script.R" containing R code for installing packages and performing statistical tests.
- Environment pane:** Shows a list of loaded objects, including GO IDs and their corresponding values.
- Packages pane:** Lists available packages with their versions and descriptions.
- Console pane:** Displays the output of running commands, including package installations and test results.

## An example of interacting with RStudio

The screenshot shows the RStudio interface with the "Session" menu open, specifically the "Set Working Directory" option.

- From the menu, select “choose directory” as shown above, to set the working directory where files will be loaded from and saved to. Set to ‘~/Desktop/WORKSHOP\_RESOURCES/Section\_2/module\_3/’ for this course.

```
Console ~ /Desktop/WORKSHOP_RESOURCES/Section_2/Module_3/
>
> setwd("~/Desktop/WORKSHOP_RESOURCES/Section_2/module_3")
```

- When you do this, you will see the “setwd” R command ran in the console. This can then be copied and pasted in the script window.

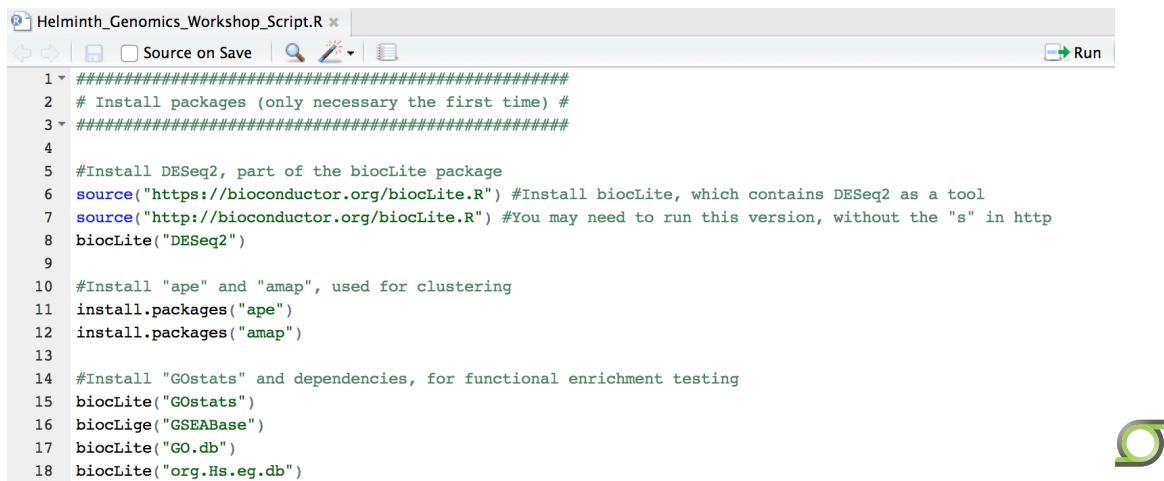
```
1 setwd("~/Desktop/WORKSHOP_RESOURCES/Section_2/module_3")
```

- If you were to save this script in the future, you could now highlight and run this command in order to set the working directory more easily.



## Installing R packages

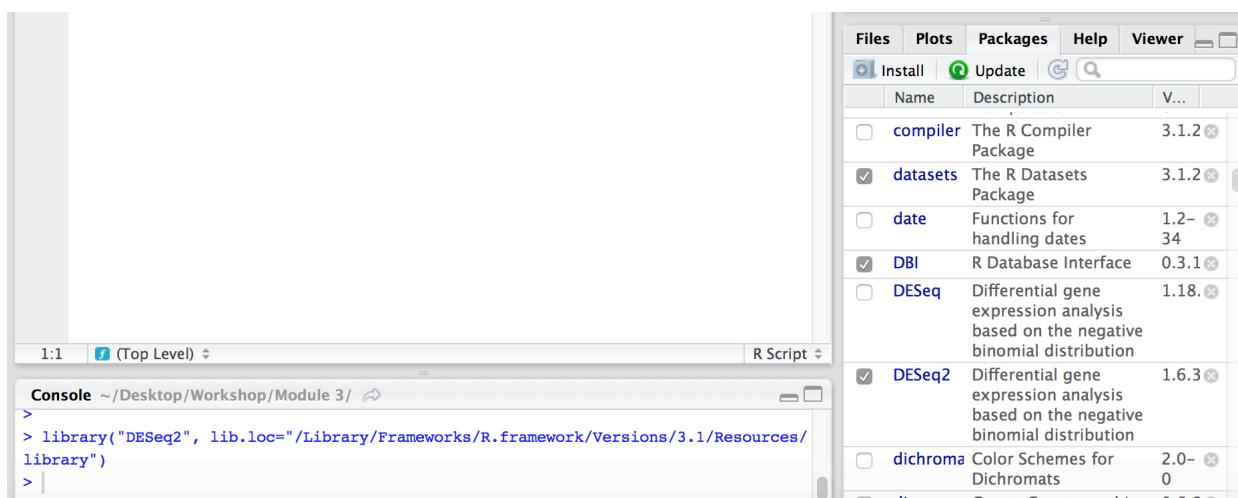
- Now open the “*Helminth\_Genomics\_Workshop\_Script.R*” file. This contains all of the commands we will need for the workshop.
- Any information following a # sign is a comment to clarify what the code is for.
- First, we will install packages. Packages are either installed directly using “`install.packages()`”, or they are loaded through bioconductor (“`biocLite`”).
- Highlight the code shown and click “run” to install all of the necessary packages.
- The manuals for different R packages will include the line necessary to install them.
- Installations only need to be performed one time on each computer, but the packages need to be loaded every time R is restarted.



```
1 #####
2 # Install packages (only necessary the first time) #
3 #####
4
5 #Install DESeq2, part of the biocLite package
6 source("https://bioconductor.org/biocLite.R") #Install biocLite, which contains DESeq2 as a tool
7 source("http://bioconductor.org/biocLite.R") #You may need to run this version, without the "s" in http
8 biocLite("DESeq2")
9
10 #Install "ape" and "amap", used for clustering
11 install.packages("ape")
12 install.packages("amap")
13
14 #Install "GOSTats" and dependencies, for functional enrichment testing
15 biocLite("GOSTats")
16 biocLite("GSEABase")
17 biocLite("GO.db")
18 biocLite("org.Hs.eg.db")
```

## Loading R packages

- After you install packages, they will show up in the “*Packages*” list in your RStudio sidebar. To “load” the packages in the future, you can simply check them off. When you do, you will see the package loading code in the console window.
- This code can also be pasted into scripts. Note that the full path is not necessary (e.g., in the screenshot below, you can just use `library("DESeq2")` instead, which will make your script compatible on other people’s computers).
- Packages can also be searched and installed from this menu, but it is typically easier to paste the install code from a guide.



Console ~ /Desktop/Workshop/Module 3/

```
>
> library("DESeq2", lib.loc="/Library/Frameworks/R.framework/Versions/3.1/Resources/
library")
> |
```

Name	Description	V...
compiler	The R Compiler Package	3.1.2
datasets	The R Datasets Package	3.1.2
date	Functions for handling dates	1.2- 34
DBI	R Database Interface	0.3.1
DESeq	Differential gene expression analysis based on the negative binomial distribution	1.18.
DESeq2	Differential gene expression analysis based on the negative binomial distribution	1.6.3
dichromat	Color Schemes for Dichromats	2.0- 0
digest	Create Cryptographic Hashes	0.6.8

## Preparing and loading input files: DESeq analysis

- Almost all differential expression callers require raw reads as input.
  - We generated read counts per sample from HTSeq output in the previous module.  
  - Open “tsuis\_mnaseq\_htseq\_countstable.txt” from the DESeq directory (in MS Excel)
  - This file contains unprocessed HTSeq count output (from the previous module) for *T. suis* collected from different stages. All downstream work will be performed on this dataset.
  - Note that this is saved as a **tab-delimited text file**. This will be the standard output from most linux programs. If you save in Excel, you will need to specify this format in the “Save as” menu.

	A	B	C	D	E	F	G	H
1	Gene	TSAC-10_day	TSAC-16_day	TSAC-17_day	TSAC-21_day	TSAC-42_day	TSAC-Adult1-	TSAC-adult_v
2	D918_00003	34	36	28	42	112	163	297
3	D918_00007	0	3	0	0	97	5	25
4	D918_00013	273	584	251	372	417	144	232
5	D918_00014	24	62	39	90	337	381	517
6	D918_00015	345	615	488	404	638	298	415

- DESeq requires the genes to be listed in the first columns, the samples labeled in the first row, and read counts in the matrix. This is standard to many of the other differential callers (including EdgeR)



## Loading input files

- After setting the working directory and loading DESeq, we load the input reads file.
  - In R, “objects” are defined using an ‘arrow’ <-
  - We will call the object for the HTSeq counts table “COUNTS”
  - It is important to understand the input command because (a) it is often omitted when you download scripts (they assume you know how to do this) and (b) having the input formatted or loaded incorrectly is a very common reason that scripts don’t work when they are launched. Pay close attention to manuals describing input data.

```
#####
# DESEQ2 #
#####

#Set working directory
setwd("~/Desktop/Workshop/Module 3/DESeq")

#Load library
library("DESeq2") #DESeq

#Read INPUT READS table (1 row of headers with sample names and 1 column with gene names; Saved as tab-delimited
COUNTS <- as.matrix(read.table(file="tsuis_rnaseq_htseq_countstable.txt", sep="\t", header=TRUE, row.names=1))




The diagram consists of seven downward-pointing arrows, each originating from an annotation on the right and pointing to a specific character or word in the R code on the left. The annotations are:



- Object name (points to the first 'as.matrix' call)
- Load as a matrix object (not always necessary) (points to the 'read.table' function call)
- Most typical command for loading data (points to the 'read.table' function call)
- Filename in working directory (points to the 'file' argument)
- Separator for the text file; Can also be comma or space, but '\t' (tab) is the most common (points to the 'sep' argument)
- Set to "FALSE" if there are no headers (points to the 'header' argument)
- Omit if there are no row names (points to the 'row.names' argument)

```

## Loading input files

- For DESeq, you will also need to prepare a metadata file describing your samples.
- This input file is formatted as shown below. Column names can be customized, but the first column must contain sample names corresponding to the counts table.

	A	B	C	D	E
1	Sample ID	Age	Stage	Comparison1	Comparison2
2	TSAC-10_day_larvae-R182	10	L2	Early	L2
3	TSAC-16_day_larvae-R171	16	L3	Early	Early
4	TSAC-17_day_larvae-R181	17	L3	Early	Early
5	TSAC-21_day_larvae-R165	21	L4	Early	Early
6	TSAC-42_day_larvae-R166	42	L5	Late	Late
7	TSAC-Adult1-r163	Adult	L5	Late	Late
8	TSAC-adult_worms-R179	Adult	L5	Late	Late

- The samples that you want to compare should be grouped in one of the columns. Here, we will focus on “Comparison1”, which is early larval stages vs late stages.
- You will need to construct this metadata file yourself prior to running R. We will look at creating tables in Excel later in this module.
- Unlike the read counts table, this input command is not loaded “as.matrix”, but is just a table:

```
#Read META DATA table (Sample names corresponding to input reads file down first column, c  
META <- read.table(file = "tsuis_rnaseq_metadata.txt", sep="\t", header=TRUE, row.names=1)
```



## Managing data

- In RStudio, loaded objects show up in the environment window.
- If you click on the table icon to the right of the object, you can view the object (in the script window) to ensure that files have loaded properly.
- Checking to see if intermediate objects are empty (“NULL”) is a good way to troubleshoot where problems are starting.

The screenshot shows the RStudio interface with the Environment tab selected. The Global Environment pane lists two objects: COUNTS and META. COUNTS is described as a Large matrix (68824 elements, 884.3 Kb). META is described as 7 obs. of 4 variables. To the left, the Script pane shows R code related to reading a file and creating these objects.

```
ed text)  
down other col  
)  
ctor", where 'S
```



## Running DESeq

- First, we will make “dds”, the `DESeqDataSet` object

```
#Make DESeq object ("design" refers to the column header in the meta data defining your comparison of interest)
dds <- DESeqDataSetFromMatrix(countData=COUNTS, colData=META, design = ~Comparison1)
```

Dataset name  
DESeq command (loaded with package)  
COUNTS dataset we previously defined  
META dataset we previously defined  
Header name META that we want to use for the comparison

- In some cases, there are secondary factors to consider. For example, samples may have been collected in two batches, introducing potential variance independent of the comparison.
- This data can be specified in the metadata file, and considered by DESeq using the following syntax:

```
dds <- DESeqDataSetFromMatrix(countData=COUNTS, colData=META, design = ~SecondaryFactor + Comparison1)
```

- This is also useful in cases of paired samples (e.g., the same individuals before and after treatment). DESeq and EdgeR can both utilize secondary factors, but CuffDiff and other software cannot.



## Running DESeq and saving results

- The following line runs the core DESeq code:

```
#Core DESeq code
dds <- DESeq(dds)
```

- Then, this summarizes the results, and writes the summary to a file:

```
#Results summary
res<-results(dds)
summary(res)
sink(file="Comparison2_Early_vs_Late_tsuis_deseq2_results_summary.txt") #Define output file
summary(res)
sink(NULL)
```

The results are also shown in the console:

```
out of 9816 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)      : 1988, 20%
LFC < 0 (down)    : 1525, 16%
outliers [1]       : 299, 3%
low counts [2]     : 0, 0%
(mean count < 0.1)
```

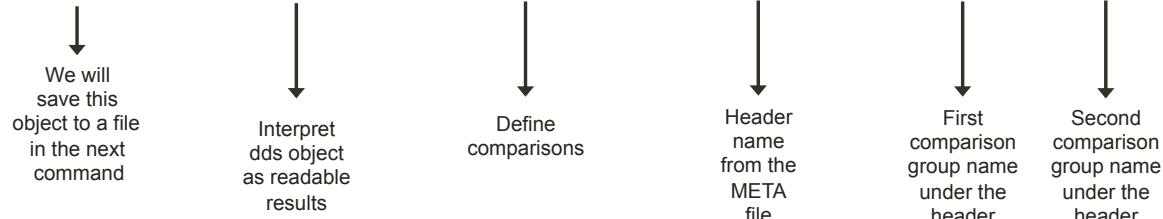
- This shows that at an adjusted p-value of 0.1, ~36% of genes are differentially expressed.
- We will parse the output manually later, with a different p value cutoff.



## Running DESeq and saving results

- Next, we prepare the output data:

```
#Output results from target comparison (enter header name from metadata if applicable)
outputtable <- results(dds, contrast=c("Comparison1", "Early", "Late"))
```



- Finally, the write.table command is used to export the results to a file in the working directory. We'll look at the results later, during the Excel tutorial.

```
write.table(outputtable, file="Comparison1_Early_vs_Late_tsuis_deseq2_output.txt", sep="\t")
```

Object name  
Output filename (try to be descriptive)  
Tab delimited



## Introduction to Microsoft Excel

- Excel is a spreadsheet program which is useful for organizing and visualizing data, calculating statistics, and performing analyses.
- Today we will learn a variety of approaches for using Excel to work with whole-genome data, with a focus on maintaining data integrity and organizing data in the most accessible way possible.
- We will go from several raw data files (generated in previous modules) to a complete database with functional annotation data, expression levels, differential expression data, and more.
- Open “Module 3 Table Completed.xlsx” in the ‘Excel’ folder to view a copy of the completed database, before we create it.

Gene	InterProScan data (Sept 11 2015)		HTSeq output (tsuis_maseq_htseq_countstable.txt, Sept 11 2015)										Gene Lengths (bp)	Sta Age Sar			
	InterPro domains	Gene Ontology Terms	Stage	L2	L3	L3	L4	L5	L5	L5	Sample Na	TSAC-1	TSAC-1	TSAC-1	TSAC-2	TSAC-4	TSAC-A
D918_00007	-	-		0	3	0	0	97	5	25							369
D918_00013	IPR018468:Double-stran	-		273	584	251	372	417	144	232							1230
D918_00014	-	-		24	62	39	90	337	381	517							1059
D918_00015	-	-		345	615	488	404	638	298	415							1341
D918_00016	IPR018972:Something	z GO:0005634:Cellular Co		1801	1672	3838	1870	2614	1923	3446							1410
D918_00017	IPR00793:ATPase, F1/GO:0046034:Biological			3091	3833	4334	4376	3333	2011	2954							1860
D918_00018	IPR001841:Zinc finger, I	GO:0005515:Molecular		706	1680	1252	2430	2285	737	1040							660
D918_00019	-	-		3912	3062	1400	3638	3894	1643	1994							1806
D918_00020	IPR008974:TRAF-like:1	GO:0005515:Molecular		928	2060	2012	1971	3821	6971	3676							2682
D918_00021	IPR011989:Armadillo-like	GO:0005515:Molecular		772	1395	1202	1287	1159	852	883							1983
D918_00022	IPR004947:Deoxyribonucleic	GO:0004531:Molecular		32	422	533	4792	25899	9485	12312							1065
D918_00023	-	-		0	16	25	45	278	1213	315							195
D918_00024	IPR021869:Ribonucleic	GO:0004531:Molecular		72	565	679	3744	15520	3983	9318							1344
D918_00025	IPR006990:Tweety:8.7e-	-		523	872	989	1024	922	1377	673							1059
D918_00026	-	-		960	2019	847	1410	1032	352	363							348
D918_00027	IPR017441:Protein kinase	GO:0005524:Molecular		416	383	435	220	450	427	338							741
D918_00028	IPR000719:Protein kinase	GO:0004674:Molecular		2406	3518	1893	2507	4375	1455	1547							1188



## Introduction to MS Excel: Formulas

- The spreadsheet is laid out in a coordinate system of “cells” with lettered columns and numbered rows. Numbers or string can be entered into any cell just by typing and pressing enter.
- Navigate the spreadsheet using either your cursor or by using the arrows on your keyboard. Multiple cells can be highlighted with the keyboard by holding shift and scrolling with the arrows.
- Formulas can be entered in any cell by entering an “=” sign.
- All formulas follow a specific format of the “=” sign, the formula name, an open bracket, variables, and a closed bracket.
- As you type a formula, a yellow box will pop up to tell you what variables can be entered. Here, I am calculating the average of a series of numbers, in cell B2. The yellow box indicates that I should enter the numbers with commas in between:

	AVERAGE	fx	=AVERAGE(1,2,3,4)
1	A	B	C
2		=AVERAGE(1,2,3,4)	
3		AVERAGE(number1, [number2], [number3], [number4], [number5], ...)	
4			

- After you close the bracket and press enter, the cell value will show the *result* of the formula, but the formula bar will show the formula itself, when cell B2 is selected:

	B2	fx	=AVERAGE(1,2,3,4)
	A	B	C
			2.5



## Formulas in MS Excel

- Formulas can also be calculated on references to cells containing numbers. This is the same formula, but the numbers have been replaced with references to cells containing numbers:

	AVERAGE	fx	=AVERAGE(A2,A3,A4,A5)
1	A	B	C
2		1=AVERAGE(A2,A3,A4,A5)	
3		AVERAGE(number1, [number2], [number3], [number4], [number5], ...)	
4			
5			

- Rather than list all of the cells, *cell ranges* can be used. This follows the format of the first cell, a colon, and then the last cell:

	B2	fx	=AVERAGE(A2:A5)
1	A	B	C
2		1	2.5
3		2	
4		3	
5		4	

- Ranges can span columns and rows (e.g., take the average of a large table).
- Cell references do not need to be typed in manually. You can select the range with your mouse, or you can use the keyboard to select it, after typing the formula and opening the bracket.
- A full list of Excel formulas can be found here:  
<http://www.techonthenet.com/excel/formulas/>



## Working with large datasets

- Open ~/Desktop/WORKSHOP\_RESOURCES/Section\_2/module\_3/Excel/tsuis\_rnaseq\_htseq\_countstable.txt, in Excel.
- This is a large table, with 9,833 rows and 8 columns, but we are going to add more columns as we build the database.
- If you hold down the “command” key on a Mac (⌘) or the “CTRL” key on Windows, and then scroll with your keyboard arrows, the selection will skip to the end of the table. This becomes essential for highlighting all of the cells in a column in a large table, since scrolling with the mouse can take several minutes.
- The first thing we will do is insert four empty rows above the dataset and one below the headers, in order to make room to add more detailed descriptions.
- To do this, right click on the number on the left-hand border, and choose “insert”. New columns or rows will enter above (rows) to the left (columns) of the insertion point.



Gene	TSAC-10_day	TSAC-16_day	TSAC-17_day	TSAC-21_day	TSAC-42_day	TSAC-Adult1	TSAC-adult_worms-R179
Cut	⌘X	0	3	0	0	97	5
Copy	⌘C	273	584	251	372	417	144
Paste	⌘V	24	62	39	90	337	381
Paste Special...	^⌘V	345	615	488	404	638	517
	1801	1672	3838	1870	2614	1923	3446
	1091	3833	4334	4376	3333	2011	2954
	706	1680	1252	2430	2285	737	1040
	912	3062	1400	3638	3894	1643	1994
	928	2060	2012	1971	3821	6971	3676
	772	1395	1202	1287	1159	852	883
	32	422	533	4792	25899	9485	12312
	0	16	25	45	278	1213	315
	72	565	679	3744	15520	3983	9318
	523	872	989	1024	922	1377	673
	960	2019	847	1410	1032	352	363
	416	383	435	220	450	427	336
	406	3518	1893	2507	4375	1455	1547
	32	17	57	27	482	603	754
	692	2083	948	1666	3079	323	981
	D918_00031	507	574	848	463	1233	547
							693

## Sorting data in Excel

- The most important thing when working with these spreadsheets is to never sort the data incorrectly. Not only will all of the results be wrong, but it will be very difficult to tell that something went wrong.
- For this reason, you should never use “Data -> Sort” to sort your data. Instead, always use the “filter” feature.
- In this example, I am highlighting (selecting) the empty row below my headers and then clicking the funnel icon that says “Filter” below it (under the “Data” tab of the ribbon).



Gene	TSAC-10_day	TSAC-16_day	TSAC-17_day	TSAC-21_day	TSAC-42_day	TSAC-Adult1	TSAC-adult_worms-R179
D918_0003	34	36	28	42	112	163	297
D918_00007	0	3	0	0	97	5	25
D918_00013	273	584	251	372	417	144	232
D918_00041	24	62	39	90	337	381	517

- Once this has been clicked, small grey arrows will appear in the row that was highlighted.



## Sorting data in Excel

- When you click on these “sorting arrows”, you can choose to sort a column of your choice, either ascending or descending. All of the data that is underneath an arrow will sort with that data, every time. If you were to sort manually, it is up to you to select the entire dataset every time, so this is the safe option to ensure data integrity.

Gene	TSAC-10_day	TSAC-16_day	TSAC-17_day	TSAC-21_day	TSAC-42_day	TSAC-Adult1	TSAC-adult_worms
D918_00003	34				112	163	297
D918_00007	0				97	5	25
D918_00013	273				417	144	232
D918_00014	24				337	381	517
D918_00015	345				638	298	415
D918_00016	1801				2614	1923	3446
D918_00017	3091				3333	2011	2954
D918_00018	706				2285	737	1040
D918_00019	3912				3894	1643	1994
D918_00020	828				2921	6074	2676

- Since we are going to add more data, we want the arrows to extend very far to the right of the spreadsheet, so that new data will also sort. Excel will only let you add the arrows to columns spanning any actual content, so scroll far to the right with the keyboard and add a space with the spacebar to a cell in row 6 (for example, in cell EA6). Then, hold shift and command/CTRL, and press left to scroll all the way back, highlighting all of the cells along the way. With the entire row selected, press the filter button in the “Data” tab of the ribbon.

- Now, as we add data to the table, all of it will be sortable and will stay organized.

- I do not recommend ever actually using the “Filter” functionality, since this hides rows from view.



## Formatting headers

- Descriptive, organized headers are essential for keeping your data organized, communicating your data to others, and for keeping track of where results came from.

Gene	Stage	L2	L3	L3	L4	L5	L5	L5
	Age (days)	10	16	17	21	42	Adult	Adult
	Sample Name	TSAC-10_day	TSAC-16_day	TSAC-17_day	TSAC-21_day	TSAC-42_day	TSAC-Adult1	TSAC-adult_worms
D918_00003		34	36	28	42	112	163	297
D918_00007		0	3	0	0	97	5	25
D918_00013		273	584	251	372	417	144	232

- Start by inserting a column before the read data, and adding row labels for the metadata. Always retain the original sample names from the raw data so that data can be compared in the future.

- Next, in cell C2, type "HTSeq output (tsuis\_rnaseq\_htseq\_countstable.txt, Sept 11 2015)", because this is a complete, descriptive header for this entire set of columns. Then highlight cells C2:J2, and click “Merge” under the “Home” tab of the ribbon:

HTSeq output (tsuis_rnaseq_htseq_countstable.txt, Sept 11 2015)								
Gene	Stage	L2	L3	L3	L4	L5	L5	L5
	Age (days)	10	16	17	21	42	Adult	Adult
	Sample Name	TSAC-10_day	TSAC-16_day	TSAC-17_day	TSAC-21_day	TSAC-42_day	TSAC-Adult1	TSAC-adult_worms
D918_00003		34	36	28	42	112	163	297
D918_00007		0	3	0	0	97	5	25
D918_00013		273	584	251	372	417	144	232

- This groups all of the columns together, while still allowing them to have separate descriptions. Each set of data with more than one column should be formatted this way to keep it as organized as possible.



## Formatting headers

- Use borders to box off the headers and the different sections of data. To do this, highlight a cell range, then click the borders box in the "home" section of the ribbon.
- For database tables, "Thick Box Borders" make it easier to read. For any table that is to be printed or published, the thinner "outside borders" look better.
- Reminder: Use Command/CTRL + shift and the arrow keys to highlight all of the data to the very bottom, to add borders to the entire data block.

The screenshot shows a Microsoft Excel spreadsheet with the Home tab selected in the ribbon. A dropdown menu is open under the 'Font' group, specifically the 'Borders' dropdown. The menu lists numerous border styles, including 'Outside Borders', 'All Borders', 'Top Border', 'Bottom Border', 'Left Border', 'Right Border', 'No Border', 'Inside Borders', 'Inside Horizontal Border', 'Diagonal Down Border', 'Inside Vertical Border', 'Diagonal Up Border', 'Top and Bottom Border', 'Bottom Double Border', 'Top and Double Bottom Border', 'Thick Bottom Border', 'Top and Thick Bottom Border', and 'Thick Box Border'. At the bottom of the dropdown is a 'Border Options...' button. The spreadsheet itself contains data in columns A, B, and C, with rows numbered from 1 to 34. Column A has 'Gene' and 'Stage' headers, while column B contains numerical values starting from D918\_00003.

## Formatting headers

- Finally, highlight your data, and use the font settings in the ribbon to make it more readable.
- Choose Arial size 10 font, and center the data whenever it's not in a long string format.
- Major headings can be bolded.
- Adjust the column widths by dragging from the edges of the column letters on the outside of the sheet, so that they only use as much space as needed.

The screenshot shows a Microsoft Excel table with the following structure:

Gene	HTSeq output (tsuis_rnaseq_htseq_countstable.txt, Sept 11 2015)						
	Stage	L2	L3	L4	L5	L5	L5
D918_00003	34	36	28	42	112	163	297
D918_00007	0	3	0	0	97	5	25
D918_00013	273	584	251	372	417	144	232
D918_00014	24	62	39	90	337	381	517
D918_00015	345	615	488	404	638	298	415
D918_00016	1801	1672	3838	1870	2614	1923	3446
D918_00017	3091	3833	4334	4376	3333	2011	2954



## Freezing panes

- Under “Layout”, and then “Freeze Panes”, you can choose to ‘freeze’ all of the rows above and all of the columns to the left of the currently selected cell.
- Doing this will lock the headers and gene names in place, so that when you scroll through the table, you will always be able to see this critical data.

The screenshot shows a Microsoft Excel spreadsheet with the ribbon menu open. The 'Layout' tab is selected, and the 'Freeze Panes' button is highlighted. A context menu is open over the header row, showing options like 'Freeze Panes', 'Freeze Top Rows', 'Freeze First Columns', and 'Unfreeze'. The main table has a header row with 'Gene' and 'HTSeq output' labels, followed by columns for 'Stage' (L2-L5) and 'Age (days)' (10-21). Below the header, there are sample names for each row. The first three rows contain gene IDs: D918\_01141, D918\_06007, and D918\_01949. The data starts at row 7.

	A	B	C	D	E	F	G	H	I	J	K	
1												
2			Gene	HTSeq output (tsuis_rnaseq_htseq_countstable.txt, Sept 11 2015)								
3			Stage	L2	L3	L3	L4	L5	L5	L5		
4			Age (days)	10	16	17	21	42	Adult	Adult		
5			Sample Name	TSAC-1	TSAC-1	TSAC-1	TSAC-2	TSAC-2	TSAC-4	TSAC-A	TSAC-a	
6												
7		D918_01141		66638	4E+05	80241	4E+05	74722	5370	35162		
8		D918_06007		77208	3E+05	73788	8E+05	1E+05	19749	45750		
9		D918_01949		2E+05	3E+05	4E+05	7E+05	94375	49435	1E+05		
10												



## Adding additional data: Gene Lengths

- We will use the gene lengths to calculate FPKM values from the raw counts table.
- First, open up “gene lengths.txt” from the Excel folder, select the entire table, and copy it to the clipboard.
- Now, go back to your main file and make a new “sheet” in Excel by clicking the + sign on beside the tabs at the bottom. Paste the data into this second sheet, so that it doesn’t paste mis-aligned into the main table.
- Add a header to your main table for where the new data will go.
- The “wrap text” font feature is helpful when the header name is long but the data will not be wide.

The screenshot shows the same Excel spreadsheet as before, but now with a new column added to the right of the 'Gene' column. This new column is labeled 'Gene Lengths (bp)'. The data in the 'Gene' column remains the same as in the previous screenshot. The 'Gene Lengths (bp)' column is currently empty, indicated by a large white box.

	A	B	C	D	E	F	G	H	I	J	K	
1												
2			Gene	HTSeq output (tsuis_rnaseq_htseq_countstable.txt, Sept 11 2015)								
3			Stage	L2	L3	L3	L4	L5	L5	L5		
4			Age (days)	10	16	17	21	42	Adult	Adult		
5			Sample Name	TSAC-1	TSAC-1	TSAC-1	TSAC-2	TSAC-2	TSAC-4	TSAC-A	TSAC-a	
6												
7		D918_01141		66638	4E+05	80241	4E+05	74722	5370	35162		
8		D918_06007		77208	3E+05	73788	8E+05	1E+05	19749	45750		
9												
10												

Why don't we just sort the two tables by gene name and then copy and paste the data?

- Because even if the same *number* of genes is present, we can't necessarily trust that every gene is present or entered in the same way.
- For example, in an updated genome draft, one gene can be removed and one new gene can be added. The genes at the start and ends of the table will match, but there will be mismatches for every gene in between these two. Any mistakes in a gene name will cause you reach false conclusions about your entire dataset.



## Looking up data in Excel with =VLOOKUP

=VLOOKUP is one of the most useful formulas in Excel, and allows for looking up matching values in a Vertical reference list.

The syntax is:

= VLOOKUP ( [Value to lookup], [Table containing the value in the first column], [column number to return], FALSE)

- In this case, we want to look up the gene length corresponding to each gene name in the main table. We will start with the first gene, which is in cell B7 in this example:

A screenshot of Microsoft Excel showing a formula bar with the formula `=vlookup(B7,`. Below the formula bar is a table titled "HTSeq output (tsuis\_rnaseq\_htseq\_countstable.txt, Sept 11 2015)". The table has columns for Gene, Stage (L2-L5), Age (days), and Sample Name. A second table titled "Gene Lengths (bp)" is shown to the right, containing data for genes D918\_01141, D918\_06007, and D918\_01949. The formula bar also shows the full formula `VLOOKUP(lookup_value, table_array, col_index_num, [range_lookup])`.

- Type “=VLOOKUP(B7,” and then click to the second tab in your file containing the gene lengths. Highlight this entire table using Command/CTRL+Shift and the arrow keys, and then type a second comma. If you make a mistake doing this, just press escape and start over. Then, click back to your main table, and finish the formula with “2” and “FALSE” as the last two entries.



## Looking up data in Excel with =VLOOKUP

- This formula now identifies the gene length of the first gene (in cell B7) by referencing the table in Sheet 2, cells B2:C9834, by matching the gene name in the first column and returning the value in the second column. The last value of “FALSE” is necessary because “TRUE” will allow approximate matches. This should always be false in all cases for any scientific work.

A screenshot of Microsoft Excel showing the completed formula `=VLOOKUP(B7,Sheet2!B2:C9834,2,FALSE)` in the formula bar. The table structure is identical to the previous screenshot, but the formula is now fully entered. The value 969 is displayed in the cell corresponding to the first gene name in the main table.



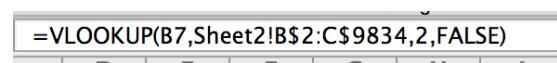
## Copying and pasting formulas in Excel

- Copy and paste the VLOOKUP formula to the cell below it, to look up the value of the second gene. You can right click or use the menus to do this, but I recommend getting used to Command/CTRL+C and Command/CTRL+V to do this.
- Note that in Excel, if you copy and paste a formula down one row, all of the cell references in the formula also move by one row (also with columns). Here, we are now looking up cell B8, to get the value for the second gene instead of the first.
- While this is useful, we have to be careful, because the cell references for the **lookup table** of gene lengths (in sheet 2) has also moved down (from B2:C9834 to B3:C9835).

The screenshot shows an Excel spreadsheet with a table titled "HTSeq output (tsuis\_rnaseq\_htseq\_countstable.txt, Sept 11 2015)". The table includes columns for Gene, Stage (L2-L5, Adult), Age (days), and Sample Name. To the right, there is a separate column labeled "Gene Lengths (bp)" containing values 969 and 975. The formula bar at the top shows =VLOOKUP(B8,Sheet2!B3:C9835,2,FALSE).

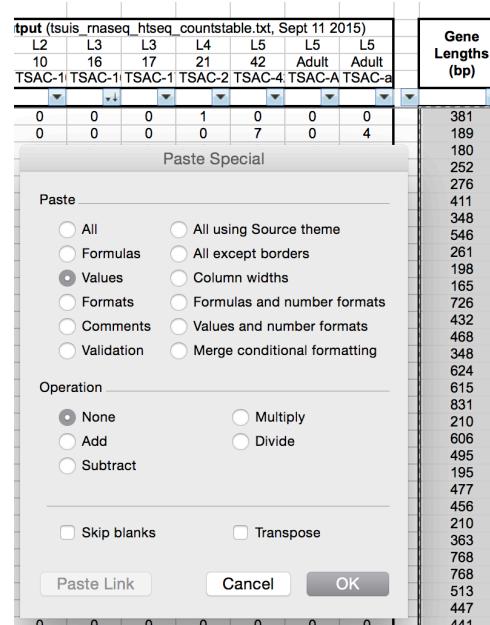
HTSeq output (tsuis_rnaseq_htseq_countstable.txt, Sept 11 2015)							
Gene	Stage	L2	L3	L3	L4	L5	L5
Age (days)	10	16	17	21	42	Adult	Adult
Sample Name	TSAC-1	TSAC-1	TSAC-1	TSAC-2	TSAC-4	TSAC-A	TSAC-a
D918_01141		66638	4E+05	80241	4E+05	74722	5370
D918_06007		77208	3E+05	73788	8E+05	1E+05	19749
D918_01949		2F+05	3F+05	4F+05	7F+05	94375	49435
						1F+05	

- In order to fix this, we can use \$ signs to “lock” the row references in place for the lookup table.
- Any column letter or row number with a \$ in front of it will not change when the formula is copied and pasted.
- Return to the first formula cell and change the reference to B\$2:C\$9834, and paste that down.



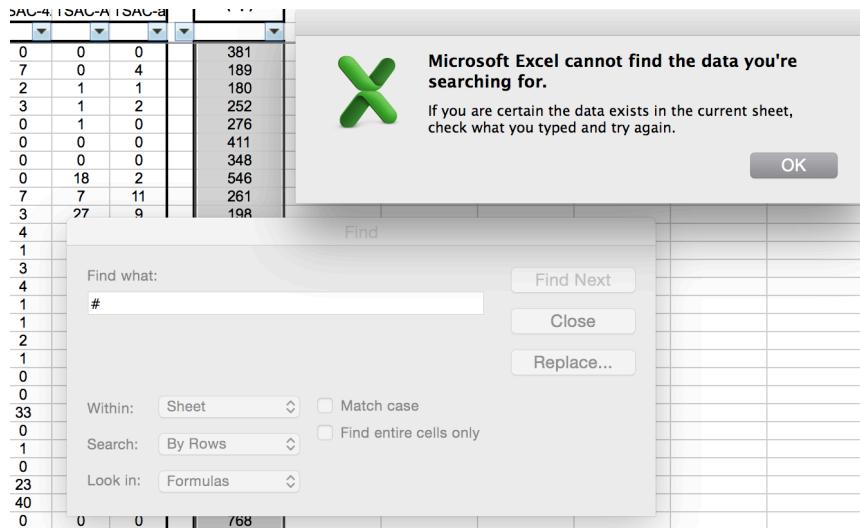
## Filling and ‘clearing’ formulas

- We need to paste the formula down the entire column.
- Copy the formula, then scroll to the bottom of the table by command/CTRL+down on one of the gene count columns.
- Starting at the bottom of the ‘gene lengths’ column, hold shift and command/CTRL and press up, to highlight the entire column. Then, paste with command/CTRL+V.
- Now we have aligned all of the gene lengths.
- The formulas are still “active” and will re-calculate every time the table is sorted or the file is saved. Enough of these active formulas will cause the spreadsheet to slow down or crash eventually.
- We will therefore “clear” the formulas, leaving their values behind.
- To do this, highlight the entire column and copy (command/CTRL+C), and then within the copied cells, right click and choose “paste special”.
- In the “Paste special” dialog, choose “values” and then click “ok”.



## Checking for formula errors

- Formulas in Excel can return errors. In the case of =VLOOKUP, if there is no lookup value in the reference table, it will return '#N/A', indicating that there is no match in the lookup table.
- All errors start with a # sign, so they can be searched easily.
- After clearing the formulas (previous slide), highlight the column and press command/CTRL+F to search.
- If there is no match in this search, then all of the genes were matched up and there is no problem.



## Calculating FPKM values

- We can now calculate FPKM expression values from the raw read counts. Start by copying and pasting the read count headers to the right of the gene lengths, and change the title of the new header set:

HTSeq output (tsuis_rnaseq_htseq_countstable.txt, Sept 11 2015)							Gene Lengths (bp)	FPKM expression values							
Stage	L2	L3	L3	L4	L5	L5		Stage	L2	L3	L3	L4	L5	L5	L5
Age (days)	10	16	17	21	42	Adult	Adult	Age (days)	10	16	17	21	42	Adult	Adult
Sample Name	TSAC-1C	TSAC-1I	TSAC-1	TSAC-2	TSAC-4	TSAC-A	TSAC-a	Sample N	TSAC-10	TSAC-16	TSAC-17	TSAC-21	TSAC-42	TSAC-Ad	TSAC-ad

- FPKM = Fragments (counts from HTSeq) Per Kilobase (gene length / 1000) per Million of reads mapped (the total read count in the sample's column in the HTSeq data).
- This gene expression measure is used because it is normalized both for the gene length and the library size, making the values directly comparable across the entire dataset, and between different experiments.
- We can calculate all of this in a single formula. Start by dividing by the count by the gene length as shown below:

HTSeq output (tsuis_rnaseq_htseq_countstable.txt, Sept 11 2015)							Gene Lengths (bp)	FPKM expression							
Stage	L2	L3	L3	L4	L5	L5		Stage	L2	L3	L3	L4	L5	L5	
Age (days)	10	16	17	21	42	Adult	Adult	Age (days)	10	16	17	21	42	Adult	Adult
Sample Name	TSAC-1C	TSAC-1I	TSAC-1	TSAC-2	TSAC-4	TSAC-A	TSAC-a	Sample N	TSAC-10	TSAC-16	TSAC-17	TSAC-21	TSAC-42	TSAC-Ad	TSAC-ad
34	36	28	42	112	163	297	891	=D7/(L7/1000)							
0	3	0	0	97	5	25	369								
273	584	251	372	417	144	232	1230								
24	62	20	200	227	201	147	1620								

- Using parentheses organizes the formula to ensure that the order of operations is correct (i.e., we are not dividing D7 by L7 first, and then dividing by 1000).



# Calculating FPKM values

- Now all of this needs to be divided by (the library size / 1,000,000). So put the entire existing formula in parentheses, and then divide by (the sum of the sample's column / a million):

- Verify this value to ensure that the formula is typed correctly (D918\_00003 in L2 = 2.6339).
  - We need to lock several things in place in order to copy and paste for the entire table. First, the reference to L7 (the gene length) needs to move down, but not left-to-right, so put a \$ sign in front of the L but not the 7.
  - Second, the “sum” range needs to be locked to the rows but not the columns. So change that to D\$7:D\$9838, so that the columns move with the formula.
  - The final formula should look like this:

= (D7/(\$L7/1000))/(SUM(D\$7:D\$9838)/1000000)

## Aligning additional data

- Copy and paste this formula for the entire FPKM table, and then clear the formulas and check for errors as shown previously.
  - This normalized data will later be used as input for hierarchical clustering (in R), but for now we will continue building the database.
  - Open “secretion data.txt” in the “Excel” directory, and paste into the second sheet of your database file as before.
  - This data is output from two different programs (Phobius and SecretomeP)
  - Create headers for the data in your main table:

FPKM expression values							Secretion Data (Sept 11 2015)		
L2	L3	L3	L4	L5	L5	L5	# TM domains (Phobius)	Secreted (phobius)	Secreted (SecretomeP)
10	16	17	21	42	Adult	Adult			
TSAC-10	TSAC-16	TSAC-17	TSAC-21	TSAC-42	TSAC-Ad	TSAC-ad			

- Set up the =VLOOKUP formula for the first row and column:

B	U	V	W	X	Y
Gene	L5 Adult TSAC-ad	Secretion Data (Sept 11 2015)			
		# TM domains (Phobius)	Secreted (phobius)	Secreted (SecretomeP)	
D918_00003	18.17		1		
D918_00007	3.6932				

- This data needs to be pasted both down and to the right. Using \$ signs, lock the column of the gene name, and the entire table: =VLOOKUP(\$B7,Sheet2!\$F\$2:\$I\$9834,2,FALSE)

## Aligning additional data

- When pasting to the right, we also need to change the “2” to a “3” in the formula, to return the value of the third column in the lookup table instead of the second.
  - Also change this value to a “4” in the last column. Then, copy all three values and paste down for the entire table, clear formulas, and check for errors.

B	U	V	W	X	Y	Z
Gene	L5 Adult TSAC-ad	Secretion Data (Sept 11 2015)				
		# TM domains (Phobius)	Secreted (phobius)	Secreted (SecretomeP)		
D918_00003	18.17	1	-	-		
D918_00007	3.6932					

- Now we will add an additional column, to indicate if each gene is secreted **either** by classical or nonclassical secretion. This should be a “Y” if either of the other two columns are a “Y”. We will use an =IF statement to perform this.

Secretion Data (Sept 11 2015)				
# TM domains (Phobius)	Classically secreted (phobius)	Nonclassically Secreted (SecretomeP)	Secreted (either)	
1	-	-		
0	Y	-		
0	-	-		



=IF formula

=IF is a very useful Excel formula for parsing data. The syntax is:

=IF( [A logical test returning true or false, usually =, <, >, or =>, <=], [value if true], [value if false] )

- So for example, try entering =IF(1=2,"Yes","No").
  - This will return "No" in the cell, because the 'logical test' is false. If you change this to 1=1, then it will return "Yes".
  - Here, we need to check whether either of the cells beside the new column are "Y". In order to accomplish this we will use OR() in the logical test:

	V	W	X	Y	Z	AA
	Secretion Data (Sept 11 2015)					
d	# TM domains (Phobius)	Classically secreted (phobius)	Nonclassically Secreted (SecretomeP)	Secreted (either)		
7	1	-	-	=if(or(X7="Y", Y7="Y"), "Y", "-")		
2	0	Y	-			

- Copy and paste this formula, clear values, and check for errors before moving on.

Secretion Data (Sept 11 2015)				
# TM domains (Phobius)	Classically secreted (phobius)	Nonclassically Secreted (SecretomeP)	Secreted (either)	
1	-	-	-	
0	Y	-	-	Y
0	-	-	-	



## =COUNTIF formula

- In the empty 'sorting' row below your secretion header, use the =COUNTIF formula to count how many genes are secreted according to each criteria.

=COUNTIF( [range of cells to count], [criteria for counting] )

Secretion Data (Sept 11 2015)				
# TM domains (Phobius)	Classically secreted	Nonclassically Secreted	Secreted (either)	
	=countif(X7:X9838,"Y")			
0	-	-	-	-
6	-	-	-	-
0	-	-	-	-
0	-	Y	Y	Y
0	-	Y	Y	Y

- Here, we are counting how many “Y” values there are in the column. Paste this to the right to count for each criteria:

Secretion Data (Sept 11 2015)				
# TM domains (Phobius)	Classically secreted (phobius)	Nonclassically Secreted (SecretomeP)	Secreted (either)	
1	-	-	-	
0	Y	-	-	Y
0	-	-	-	
1	-	-	-	

- This is an easy way to summarize your data. You can also check if values are greater than zero ( “>0”), if values are larger than the value in another cell, etc.
  - =COUNTIFS (with an S) can check multiple criteria in multiple columns.

## Annotation data (lookup with missing values)

- Open “interproscan\_annotations\_per\_gene.txt” from the “Excel” file, and copy and paste into the second sheet as before.
  - Prepare the headers and use =VLOOKUP as before:

Secretion Data (Sept 11 2015)				InterProScan data (Sept 11 2015)	
# TM domains (Phobius)	Classically secreted (phobius)	Nonclassically Secreted (SecretomeP)	Secreted (either)	InterPro domains	Gene Ontology Terms
	863	2676	3539		
1	-	-	-	#N/A	\$4:\$M\$6707,3,FA
0	Y	-	Y		

- This time there is an #N/A value because the lookup table does not contain unannotated genes. Paste the formulas through, and then clear the formulas.
  - Now, replace the #N/A values with “-”, to clean up the table.
  - When long strings “hang” over into the next cell, add an empty space in the column to the right, to cover it up:

InterProScan data (Sept 11 2015)	
InterPro domains	Gene Ontology Terms
-	-
-	-
-	-
IPR018468:Doubl	-
-	-
-	-
-	-
IPR018972:Some GO:0005634:Cell	
IPR000793:ATPase GO:0046034:Biological Process: AT	
IPR001841:Zinc fi GO:0005515:Molecular Function: pr	
IPR008974:TRAF GO:0005515:Molecular Function: pr	
IPR011989:Armac GO:0005515:Molecular Function: pr	
IPR004917:Docy GO:0004531:Molecular Function: d	

## Parsing DESeq results

- Now we will add the DESeq results we calculated in RStudio.
  - Open the “Comparison1\_Early\_vs\_Late\_tsuis\_deseq2\_output.txt” file in the DESeq folder, and paste it into the second sheet of the dataset as before.
  - First, note that the headers are all shifted to the left by 1 column. Cut and paste those to the right to fix this. This problem commonly occurs with R output (row.names has no header entry), so always be sure to check for an empty final column.

baseMean	log2FoldChan	IfcSE	stat	pvalue	padj	
D918_00003	102.244975	-2.2852271	0.54219132	-4.2147983	2.50E-05	0.00019566
D918_00007	13.3896063	-4.7819266	1.08457881	-4.4090172	1.04E-05	8.68E-05
D918_00013	310.784483	0.74493719	0.37667336	1.9776742	0.04796547	0.12076386
baseMean	log2FoldChan	IfcSE	stat	pvalue	padj	
D918_00003	102.244975	-2.2852271	0.54219132	-4.2147983	2.50E-05	0.00019566
D918_00007	13.3896063	-4.7819266	1.08457881	-4.4090172	1.04E-05	8.68E-05
D918_00013	310.784483	0.74493719	0.37667336	1.9776742	0.04796547	0.12076386

From the DESeq manual:

The interpretation of the columns of `data.frame` is as follows.

<b>id</b>	feature identifier
<b>baseMean</b>	mean normalised counts, averaged over all samples from both conditions
<b>baseMeanA</b>	mean normalised counts from condition A
<b>baseMeanB</b>	mean normalised counts from condition B
<b>foldChange</b>	fold change from condition A to B
<b>log2FoldChange</b>	the logarithm (to basis 2) of the fold change
<b>pval</b>	<i>p</i> value for the statistical significance of this change
<b>padj</b>	<i>p</i> value adjusted for multiple testing with the Benjamini-Hochberg procedure (see the R function <code>p.adjust</code> ), which controls false discovery rate (FDR)



## Parsing DESeq results

- We are only interested in the Log2 Fold Change and Adjusted P value, so delete the other columns by right-clicking the column letters on the border and deleting them:

	log2FoldChan	padj
D918_00003	-2.2852271	0.00019566
D918_00007	-4.7819266	8.68E-05
D918_00013	0.74493719	0.12076386

- Set up these headers in the main sheet, and perform the VLOOKUP for these values, then add two headers, for the average FPKM values from the two sample groups:

DESeq results (Early vs Late Larval; L2,L3,L4 vs L5)			
Log2 Fold Change	Adjusted P value	Average FPKM Early	Average FPKM Late
-2.2852271	0.00019566		
-4.7819266	8.6841E-05		
0.74493719	0.12076386		

- Use =AVERAGE to calculate the average value of the sample groups, then paste the formulas down and clear the formulas.

## Parsing DESeq results

- We want to know whether each gene is significantly differentially expressed in either early larval or late larval stages. Start by setting up additional headers:

DESeq results (Early vs Late Larval; L2,L3,L4 vs L5)					
Log2 Fold Change	Adjusted P value	Average FPKM Early	Average FPKM Late	Sig. Higher in Early	Sig. Higher in Late
-2.2852271	0.00019566	2.01222465	11.4886605		
-4.7819266	8.6841E-05	0.08319206	4.73819485		
0.74493719	0.12076386	14.7543182	10.0696598		
-2.7988517	7.828E-07	2.41819608	20.4226954		

- We can see that a negative fold change corresponds to a gene that is higher in the late stages than the early stages (and vice versa for a positive value).
- Therefore, in order to call a gene significantly higher in the early stages: (a) the fold change value needs to be greater than zero, and (b) the P value needs to be less than a threshold value of your choice.
- DESeq recommends a maximum threshold P value of 0.1, but we will parse more conservatively, at 0.01 instead.
- For a very high-confidence small gene set, a threshold of  $10^{-5}$  could be used.
- Generally, 0.05, 0.01, or  $10^{-5}$  are used for publications.
- Fold change thresholds should **not** be used for RNA-Seq data. There is justification for it with microarrays, but the high sensitivity of RNA-Seq data (and high abundance of zero values) invalidates its use for statistical cutoffs.



## Parsing DESeq results

- For the first column, use an =IF statement with an “AND” function to check whether both (a) the Fold change value is greater than zero and (b) the P value is less than or equal to 0.01:

	fx	=IF(AND(AE7>0,AF7<=0.01),"Y","-")				
D	A1	IF(logical_test, [value_if_true], [value_if_false])	AI	AJ	AK	
DESeq results (Early vs Late Larval; L2,L3,L4 vs L5)						
Log2 Fold Change	Adjusted P value	Average FPKM Early	Average FPKM Late	Sig. Higher in Early	Sig. Higher in Late	
-2.2852271	0.00019566	2.01222465	11.4886605	=IF(AND(AE7>0,AF7<=0.01),"Y","-")		
-4.7819266	8.6841E-05	0.08319206	4.73819485			

- Repeat for the second column, but check if the fold change is *less than* zero for it. Then paste the two columns down, clear the formulas, and check for errors.
- Paste the =COUNTIF formula from the secretion columns to count the differentially expressed genes. Note that this doesn’t match the RStudio summary because we are using a different threshold; At a 0.1 threshold, the counts do match.

DESeq results (Early vs Late Larval; L2,L3,L4 vs L5)					
Log2 Fold Change	Adjusted P value	Average FPKM Early	Average FPKM Late	Sig. Higher in Early (0.01)	Sig. Higher in Late (0.01)
-2.2852271	0.00019566	2.01222465	11.4886605	-	Y
-4.7819266	8.6841E-05	0.08319206	4.73819485	-	Y
0.74493719	0.12076386	14.7543182	10.0696598	-	-



## Analyzing data

- Look at the most significantly differentially expressed genes by sorting by P value (A->Z), and then by one of the two categories (Z -> A):

DESeq results (Early vs Late Larval; L2,L3,L4 vs L5)					
Log2 Fold Change	Adjusted P value	Average FPKM Early	Average FPKM Late	Sig. Higher in Early (0.01)	Sig. Higher in Late (0.01)
-10.644221	6.351E-130	2.53810914	5274.44293	-	746
-9.9884623	5.857E-110	0.92325578	1167.77682	-	1229
-11.101479	3.3948E-90	0.52283999	1582.81916	-	
-10.139221	4.3186E-86	3.75579835	5438.42184	-	
-7.6516894	8.9743E-86	1.71464222	411.035881	-	
-8.516144	2.6986E-84	4.88351934	2208.58388	-	
-9.424408	1.6026E-80	2.74299774	2359.01521	-	
-7.9488889	2.5898E-79	22.8418144	6760.78978	-	

- Scroll to the left to see the the InterProScan annotation data, which gives information on the functions of these most significant genes:

InterProScan data (Sept 11 2015)		DESeq results (Early vs Late Larval; L2,L3,L4 vs L5)				
InterPro domains	Gene Ontology Terms	Log2 Fold Change	Adjusted P value	Average FPKM Early	Average FPKM Late	Sig. Higher in Early (0.01)
IPR003587:Hedgehog/intein hint, N-terminal:3.9e-10 Pf GO:0008233:Molecular Function: peptidase activity .7.		9.26599862	3.3309E-68	592.959089	0.92994857	Y
IPR008160:Collagen triple helix repeat:4.4e-09	-	7.23516711	2.9315E-65	130.369762	0.89935279	Y
IPR002486:Nematode cuticle collagen, N-terminal:8.8e-2 GO:0042302:Molecular Function: structural constituent	-	9.88458232	8.5452E-59	463.195311	0.44698347	Y
-	-	7.14546039	1.6124E-54	133.441247	0.94641915	Y
IPR002486:Nematode cuticle collagen, N-terminal:5.1e-2 GO:0042302:Molecular Function: structural constituent		8.68270241	3.2897E-47	1121.30758	2.70186837	Y
IPR003582:Metridin-like ShK toxin:6e-06	-	6.45614714	1.7596E-46	740.679087	8.8752889	Y
IPR002486:Nematode cuticle collagen, N-terminal:6.2e-2 GO:0042302:Molecular Function: structural constituent	-	10.2443388	6.7341E-46	1289.26298	0.83838701	Y
IPR014044:CAP domain:6.7e-05	-	11.0441962	2.0826E-45	957.416983	0.30801891	Y
IPR002181:Fibrinogen, alpha/beta/gamma chain, C-term GO:0007165:Biological Process: signal transduction .7.		3.99722095	1.5752E-39	371.605534	26.0147455	Y

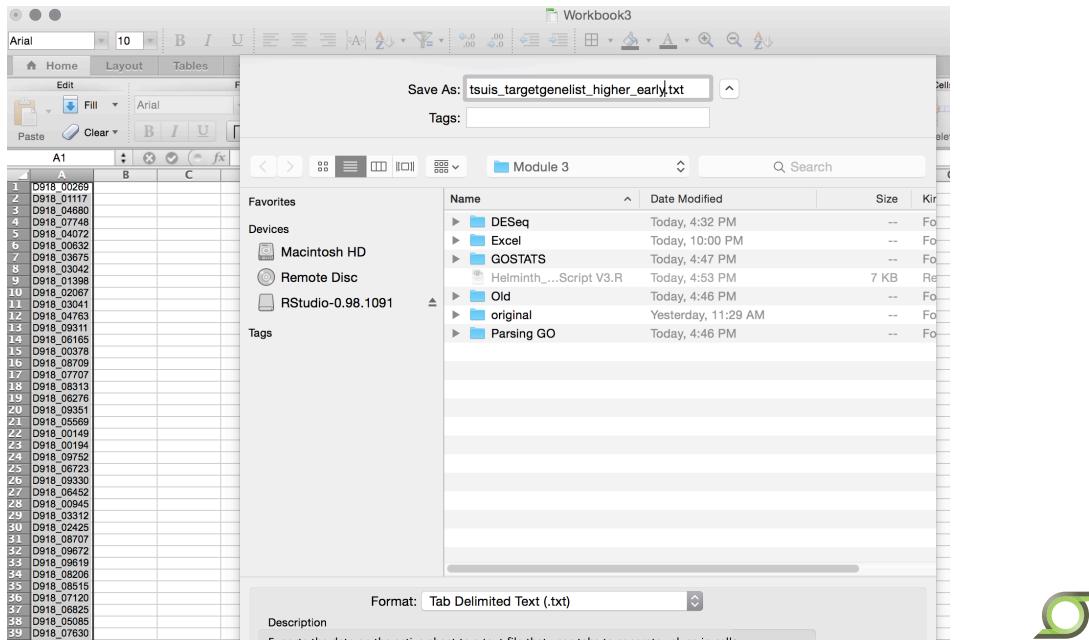
## Saving data for clustering and functional enrichment testing

- For clustering, copy and paste the gene names and the FPKM values for each sample into a new spreadsheet, then save as a tab-delimited text file. Renaming the long sample names to shorter IDs will make the final cluster look nicer:

The screenshot shows an Excel spreadsheet with data in columns A through D. Column A contains gene identifiers, and columns B through D contain FPKM values for samples L1, L2, L3, L4, and L5 respectively. The data is being saved as a tab-delimited text file named "FPKM\_matrix.txt". The "Save As" dialog is open, showing the file name and format. The "Format" dropdown is set to "Tab Delimited Text (.txt)". The "Description" field is empty. The "Favorites" and "Devices" sections of the ribbon are visible on the left, and the "Tags" section is visible on the right.

## Saving data for clustering and functional enrichment testing

- For functional enrichment, we will need a “target” gene list of differentially expressed genes. In the interest of time, we will just save the “higher in early” gene list. Sort the spreadsheet by that column, then copy and paste all of the genes with “Y” values into a new file, then save as a tab delimited text with no headers:



## PCA from DESeq results

- Principal component analysis (PCA) is one approach for visualizing how expression patterns vary across samples.
- Go back to R and find the PCA code section.
- DESeq has a built-in tool for running PCA that utilizes the dds object created earlier.

```
#####
# PCA #
#####

#Log transform deseq object data
rld <- rlogTransformation(dds, blind=TRUE)

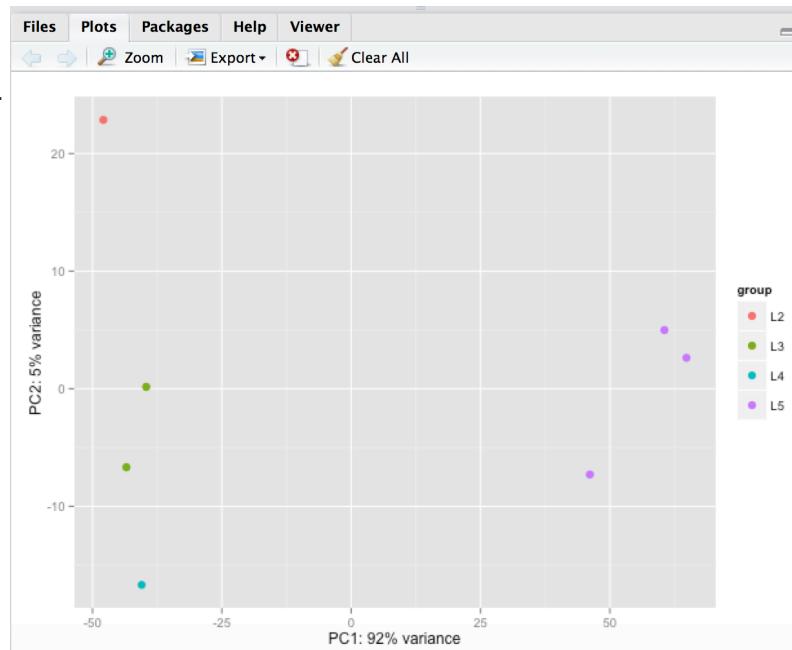
#Perform and plot PCA based on data from top X expressed genes (default 500)
plotPCA (rld, intgroup=c("Stage"), ntop=500)
```

- These commands log transform the data, and then plot the PCA.
- Note that “intgroup” can be any column of the metadata file. Here we use “stage” to give more detail on each sample, as opposed to just the two categories in “Comparison1”.
- “ntop” defines the number of genes to use to calculate the PCA. Using too many low-information genes may add noise to the clustering. The default is 500, but the results are generally not sensitive to changing the number.



## PCA from DESeq results

- After running these commands, the PCA plot will show up in the bottom-right panel.
- Clicking “Export” will allow you save this file. If you save as a PDF, you can edit the plot directly in a vector-based image editing program (Adobe Illustrator, or “Inkscape”, which is free).
- We will also export the plot co-ordinates so that the data can be replotted in Excel later.



## PCA from DESeq results

- The following code will save the PCA coordinates into a file so that the data can be graphed in other programs, and outputs the variance of each component, including those not shown on the plot.

```
#Output PCA coordinates
PCAcoordinates<-plotPCA (rld, intgroup=c("Stage"), ntop=500,returnData = TRUE)
write.table(PCAcoordinates, file="tsuis_PCA_coordinates", sep="\t")

#Output variance per component
rlogMat <- assay(rld)
rv = apply(rlogMat, 1, var)
select = order(rv, decreasing=TRUE)[seq_len(min(500, length(rv)))]
pca = prcomp(t(rlogMat[select,]))
sink(file="tsuis_PCA_variances_per_component.txt") #Define output file
summary(pca)
sink(NULL)
```



## Hierarchical clustering in RStudio

- PCA was calculated directly from the DEseq dataset, but we will use FPKM values for hierarchical clustering.
- Run this code to load libraries and prepare the input files:

```
#####
# CLUSTERING #
#####

setwd("~/Desktop/Workshop/Module 3/")

#Load libraries
library("ape") #for clustering
library("amap") #for clustering

#Input file - Usually FPKM values per gene, genes down the row
x <- read.delim("FPKM_matrix.txt", header=TRUE, row.names=1)
#Transpose matrix (clustering takes genes in columns, samples in rows)
x <- t(x)
```

- If there is an error, check that the file names match.
- Next, we create a distance matrix. The statistic specified here determines the clustering algorithm. Pearson or Spearman correlation is typically used for RNA-Seq data, and “average” linkage is typically best for drawing the clusters:

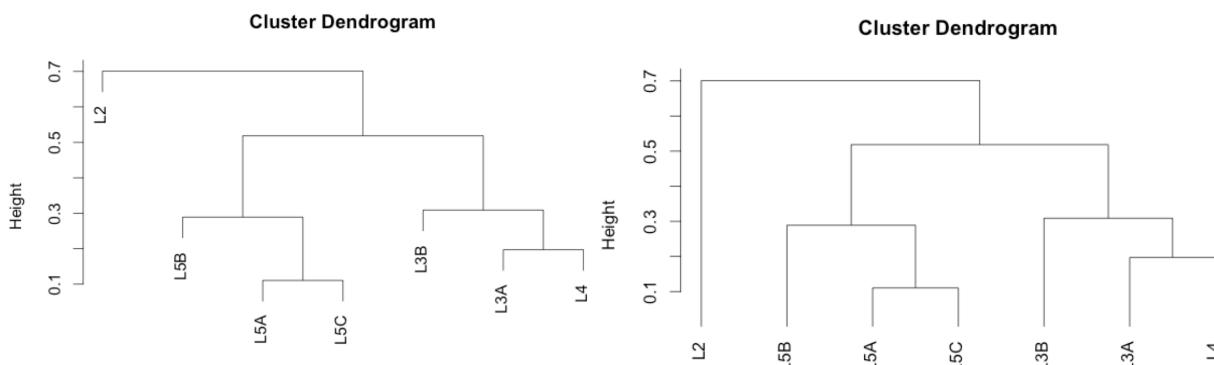
```
#Create distance matrix, can use different clustering methods here instead (pearson, spearman)
dist.mat<-Dist(x, method="pearson", diag = FALSE, upper = FALSE)
#Cluster distance matrix, can use different clustering methods (average, complete, single)
cluster=hclust(dist.mat, method = "average", members=NULL)
```



## Hierarchical clustering in RStudio

- The script includes two approaches for viewing the clustering:

```
#Plot the cluster results with actual distances
plot(cluster)
#Plot the cluster with equal distances for each sample
plot(cluster, hang=-1)
```



- You can export one or both of these as PDF for future reference.
- Finally, the script exports a newick-format file for input into other clustering programs (e.g. FigTree or iTOL):

```
#Optional: Convert cluster plot to newick cluster file format for input to other software
my_tree <- as.phylo(cluster)
write.tree(phy=my_tree, file="tsuis_rnaseq_clustering_pearson_average.newick")
```



---

## Functional enrichment using GOSTATS in RStudio

---

- Run the following to prepare the GO database:

```
#####
# Functional Enrichment (GOSTATS) #
#####

setwd("~/Desktop/Workshop/Module 3/GOSTATS/")

#Load necessary libraries
library("GOSTATS")
library("GSEABase")
library("org.Hs.eg.db")

#Input gene to GO file (tab delimited, three columns: go_ID, evidence [always "IEA"], gene_ID)
genetogo=read.table("GO_to_geneID.txt", sep="\t",header=TRUE)

#Process input GO file

goframeData = data.frame(genetogo)
goFrame=GOFrame(goframeData,organism="Trichuris suis")
goAllFrame=GOAllFrame(goFrame)
gsc <- GeneSetCollection(goAllFrame, setType = GOCollection())
frame = toTable(org.Hs.egGO)
```

- “Go\_to\_geneID.txt” is a pairwise GO and Gene list, generated from InterProScan output in a different module.
- Producing this file is the difficult part about running enrichment on a custom genome. Most tools (including GOSTATS) are designed to be easy to use primarily for model organisms.



---

## Functional enrichment using GOSTATS in RStudio

---

- Here we will input the complete (background) *T. suis* gene set, and our shorter target gene set that we saved from Excel, based on the DESeq output:

```
#Input full gene list and test gene list (no header, just single-column gene name lists)
universe=read.table("tsuis_full_gene_list.txt", sep="\t",header=FALSE)
testgenes=read.table("../tsuis_targetgenelist_higher_early.txt", sep="\t",header=FALSE)
```

- The remaining code runs the enrichment test and produces output. It is ran three times, one for Biological Process (BP), one for Molecular Function (MF) and one for Cellular Component (CC) Gene Ontology terms. Run all of this code to produce the three output files:

```
#BP
params <- GSEAGOHyperGParams(name="My Custom GSEA based annot Params",
                                geneSetCollection=gsc,
                                geneIds = testgenes,
                                universeGeneIds = universe,
                                ontology = "BP",
                                pvalueCutoff = 1,
                                conditional = FALSE,
                                testDirection = "over")
Over <- hyperGTest(params)
write.table(summary(Over), file="GOSTATS_output_BP.txt", sep="\t")
```



## Manual False Discovery Rate (FDR) correction

- Open the “GOSTATS\_output\_MF.txt” file in the GOSTATS folder (Using Excel).
- As with DESeq output, shift the headers to the right by 1 column:

GOMFID	Pvalue	OddsRatio	ExpCount	Count	Size	Term
1 GO:0042302	1.29E-18	63	1.69414405	19	23	structural constituent of cuticle
2 GO:0017171	6.09E-14	6.51922057	7.51316058	33	102	serine hydrolase activity
3 GO:0008236	6.09E-14	6.51922057	7.51316058	33	102	serine-type peptidase activity
4 GO:0004252	1.46E-13	6.78484848	6.85023465	31	93	serine-type endopeptidase activity
5 GO:0008233	4.85E-13	3.62266637	20.2560702	56	275	peptidase activity
6 GO:0070011	1.66E-11	3.41863045	19.5194858	52	265	peptidase activity, acting on L-amino acid, peptidase activity, acting on D-amino acid
7 GO:0005198	2.16E-11	3.90359153	14.3633952	43	195	structural molecule activity
8 GO:0004175	2.39E-11	3.75228763	15.5419302	45	211	endopeptidase activity
9 GO:0003735	5.37E-06	3.36456279	8.32340339	23	113	structural constituent of ribosome

- The list is sorted by P value, with the most significant terms at the top. However, these P values are not population-corrected, and this must be done manually for GOSTATS.
- We need to do correction because there are multiple tests being performed. A 5% chance of being false is not acceptable when performing hundreds of tests.
- Generally, FDR correction is preferred for multiple-testing because it is a reasonable balance of stringency. The most stringent approach is Bonferroni correction (multiplying P values by the number of tests).
- For FDR, the most significant P value is multiplied by the number of tests. The second-most significant P value is multiplied by the number of tests divided by two. The third-most significant P value is multiple by the number of tests divided by three, etc.



## Manual False Discovery Rate (FDR) correction

- This output file contains 314 tests. So the P values need to be recalculated according to:  
P value \* ( 314 / [rank of P value] )
- We can accomplish this using the =RANK formula in Excel:  
=RANK( [value], [range of all values], [0 = Largest first, 1 = Smallest First] )

B	C	D	E	F	G	H	I
GOMFID	Pvalue	OddsRatio	ExpCount	Count	Size	Term	FDR
1 GO:0042302	1.29E-18	63	1.69414405	19	23	structural constituent of cuticle	4.05E-16
2 GO:0017171	6.09E-14	6.51922057	7.51316058	33	102	serine hydrolase activity	9.57E-12
3 GO:0008236	6.09E-14	6.51922057	7.51316058	33	102	serine-type peptidase activity	9.57E-12

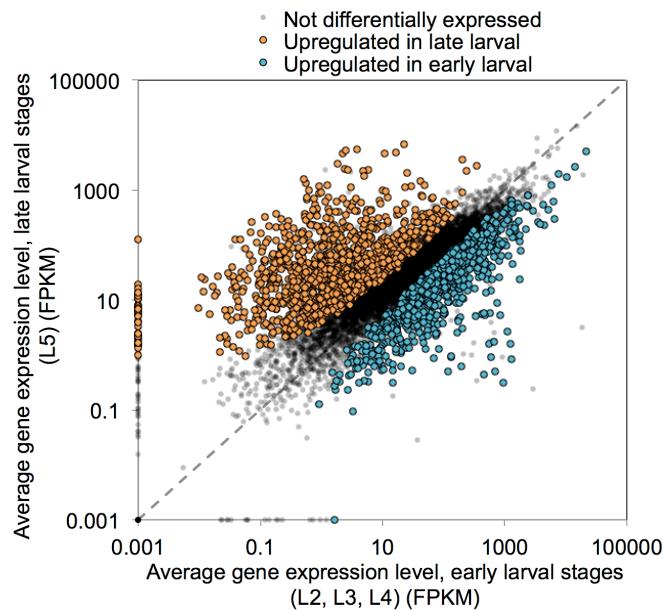
- The formula shown will calculate FDR-corrected P values in column I. The threshold value (0.01) will be applied on these FDR values.
- Some additional formatting will clean up the table and make it ready for publication:

**Table 1:** Molecular Function Gene Ontology terms significantly enriched among genes upregulated in early larval stages compared to late stages

GO ID	Term Description	Expected	Gene Counts		FDR-corrected P
			Observed	Total	
GO:0042302	structural constituent of cuticle	1.7	19	23	4.05E-16
GO:0017171	serine hydrolase activity	7.5	33	102	9.57E-12
GO:0008236	serine-type peptidase activity	7.5	33	102	9.57E-12
GO:0004252	serine-type endopeptidase activity	6.9	31	93	1.15E-11
GO:0008233	peptidase activity	20.3	56	275	3.04E-11
GO:0070011	peptidase activity, acting on L-amino acid peptides	19.5	52	265	8.71E-10
GO:0005198	structural molecule activity	14.4	43	195	9.70E-10
GO:0004175	endopeptidase activity	15.5	45	211	9.37E-10
GO:0003735	structural constituent of ribosome	8.3	23	113	1.88E-04
GO:0061134	peptidase regulator activity	6.7	17	91	8.71E-03
GO:0030414	peptidase inhibitor activity	6.7	17	91	8.71E-03

## Graphing in Excel

- Excel is a very useful program for graphing data, since graphs are easily customizable and interactive.
- We will go through the steps required to create a publication-quality scatterplot image of the previously-generated differential gene expression data.
- Note that within excel, graphs are called “charts”. Also note that Excel, particularly on Macs, can sometimes be prone to crashing when working with graphs. Be sure to save frequently.



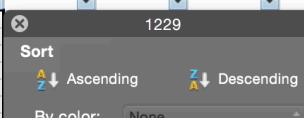
- The points in a graph on it will stay linked to the data you enter. So, if data in the sheet is re-sorted or changed, then the graph will automatically update. For this reason, we will start by moving the data to be graphed onto a new separate sheet, where it won't be changed later:



## Graphing in Excel

- Copy and paste gene names, and all of the DESeq data from the main data sheet into the new graph data sheet.
- Delete the fold change and P value columns by selecting the entire columns (by clicking the letters on the border of the spreadsheet) and then right clicking and “delete”. This data is not required to construct the graph.
- Add the sorting arrows, then sort the sheet by ‘higher in early’ and then ‘higher in late’, so that the three categories of differential expression are in blocks in the table:

Gene	Average FPKM Early	Average FPKM Late	Sig. Higher in Early (0.01)	Sig. Higher in Late (0.01)
D918_00026	184.191624	76.7885724	Y	-
D918_00052	826.869376	50.5122168	Y	-
D918_00061	43.0357892	23.259435	Y	-
D918_00063	113.368905	57.237922	Y	-
D918_00092	65.2295686	31.4537095	Y	-
D918_00093	79.7689055	26.3241948	Y	-

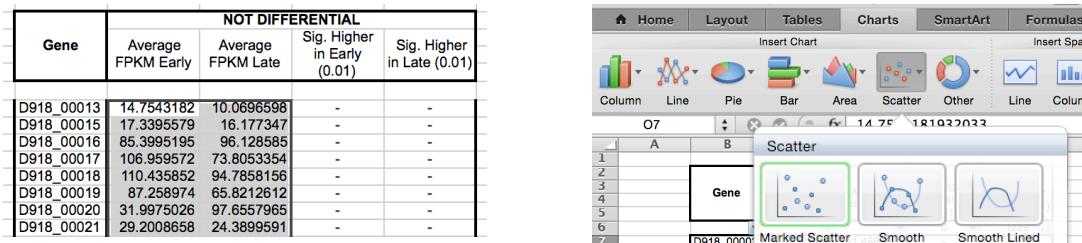


- Cut and paste this table into three sections: Higher in early, higher in late, and not differentially expressed. This isn't strictly necessary to construct the graph, but it is helpful for organization. Copy and paste the headers to organize the data:

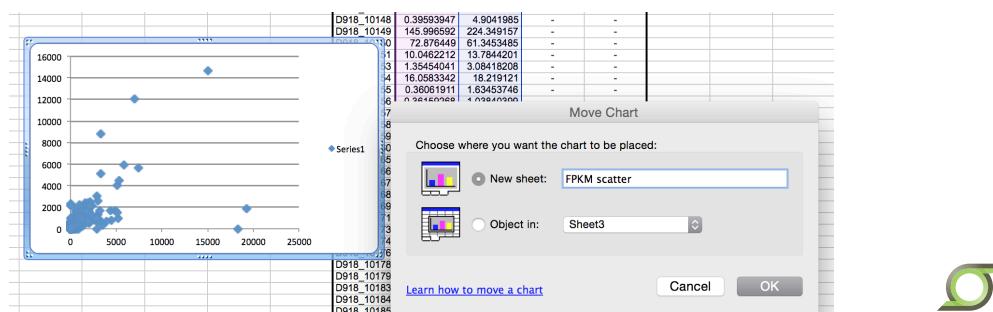
Gene	Higher Late				Gene	Higher Early				Gene	Not. Diff Expressed			
	Average FPKM Early	Average FPKM Late	Sig. Higher in Early (0.01)	Sig. Higher in Late (0.01)		Average FPKM Early	Average FPKM Late	Sig. Higher in Early (0.01)	Sig. Higher in Late (0.01)		Average FPKM Early	Average FPKM Late	Sig. Higher in Early (0.01)	Sig. Higher in Late (0.01)
D918_00003	2.01222465	11.4886605	-	Y	D918_00026	184.191624	76.7885724	Y	-	D918_00013	14.7543182	10.0696598	-	-
D918_00007	0.08319206	4.73819485	-	Y	D918_00052	826.869376	50.5122168	Y	-	D918_00015	17.3395579	16.177347	-	-
D918_00014	2.41819608	20.4226954	-	Y	D918_00061	43.0357892	23.259435	Y	-	D918_00016	85.3995195	96.128585	-	-
D918_00023	5.09743301	179.170881	-	Y	D918_00063	113.368905	57.237922	Y	-	D918_00017	106.959572	73.8053354	-	-
D918_00029	1.34108405	24.2946085	-	Y	D918_00092	65.2295686	31.4537095	Y	-	D918_00018	110.435852	94.7858156	-	-
D918_00034	1.56275633	15.8094969	-	Y	D918_00093	79.7689055	26.3241948	Y	-	D918_00019	87.258974	65.8212612	-	-
D918_00038	0.14974576	7.33730798	-	Y	D918_00102	101.262191	49.5918198	Y	-	D918_00020	31.9975026	97.6557965	-	-
nn18_nn040	0.9838383	1.66467957	-	Y	D918_00113	27.1748171	4.12973106	Y	-	D918_00021	29.2006658	24.3899491	-	-

## Graphing in Excel

- We will start by graphing the “not differentially expressed” genes as an X-Y scatterplot.
- Use Shift + command/CTRL to highlight the FPKM data down this entire column. Then, under “charts”, choose “scatter” and then “Marked scatter” (with no lines):

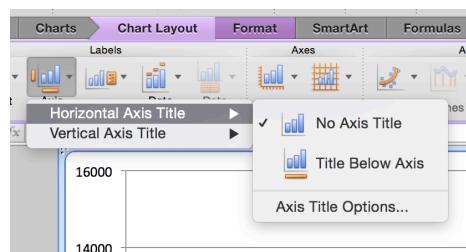


- When you do this, Excel will generate a simple plot of the data, as an object on the sheet. Right click the empty white space on the plot, select “Move Chart”, and then specify a “new sheet” instead, so that it puts the chart on its own sheet:

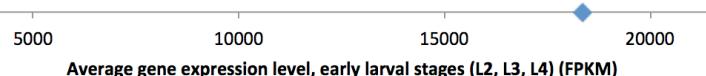


## Graphing in Excel

- The default chart is not formatted nicely, and may vary by version of Excel.
- Note that the order of the following formatting steps doesn’t matter.
- First, we will add axes labels. Under “chart layout”, select “axis titles”, and then click to add a title below the X axis and a rotated title on the Y axis:



- Click on the axes titles to change the labels to something descriptive, usually with units in parentheses:

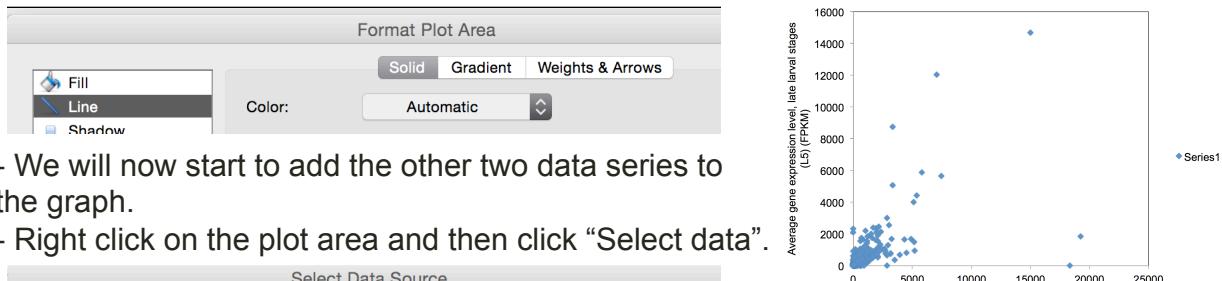


- Next, click on empty white space in the corner of the sheet, to select the entire graph. This will allow you to set a global font without adjusting each component manually. Arial font is always acceptable for publication, so choose it, and choose size 16 font. This large font size is necessary because graphs are rarely printed as a full page, but instead are often shrunk into a single panel.

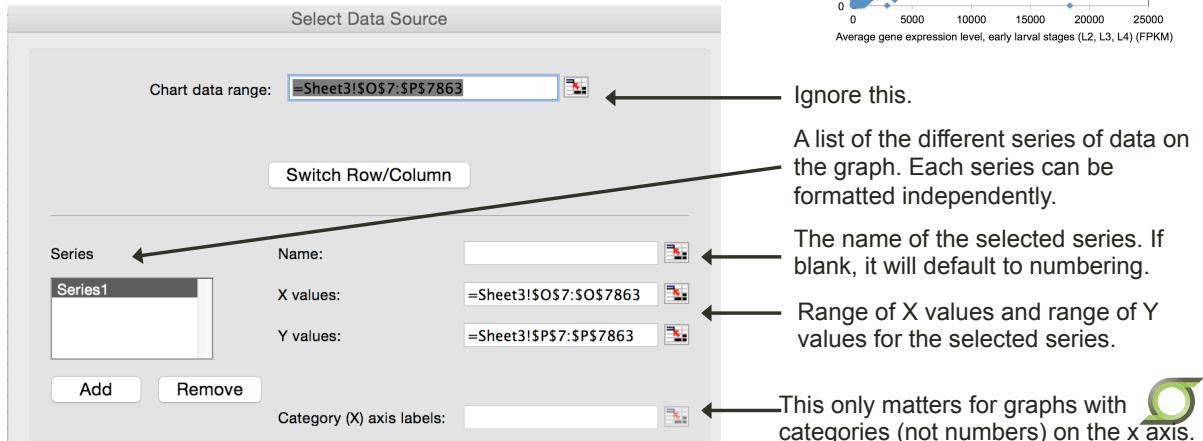


## Graphing in Excel

- Remove horizontal gridlines by clicking on one of them and pressing the “delete” key (backspace on windows). Double-click on the plot area and under “line”, choose black for the color instead of “automatic”. This will put a border around the plot.



- We will now start to add the other two data series to the graph.
- Right click on the plot area and then click “Select data”.



## Graphing in Excel

- First, rename the existing series to “Not differentially expressed” (this is the data we started the graph with).
- Click “add” to add a second series. Title the series (“Upregulated in late larval”), and then click the red arrow beside the “X values” to select the x axis values for this series.



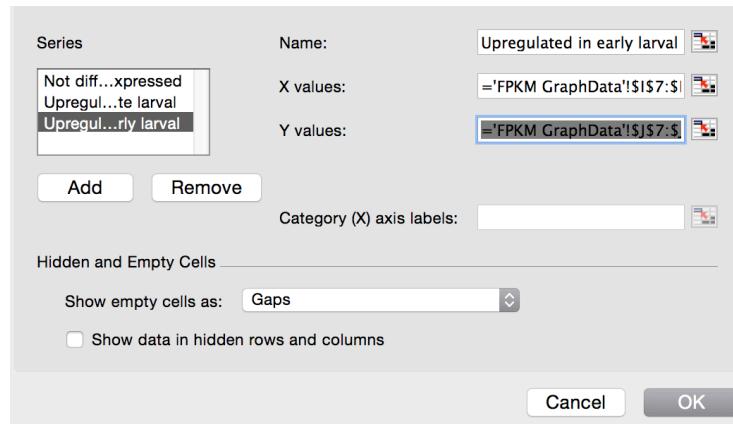
- Click back to the ‘FPKM GraphData’ tab, and highlight the X values (early larval) from the “Upregulated in late larval” columns you previously set up:

A screenshot of the 'FPKM GraphData' sheet in Excel. A column of X values is highlighted with a blue selection bar. The sheet has columns labeled 'Y', 'D918\_00346', '141.984195', and '35.5892882'. Below the sheet is a 'Select Data Source' dialog with the formula '=FPKM GraphData!\$C\$7:\$C\$1235' in the 'Chart data range' field.

- On windows, you can click on the first cell, and shift + CTRL down to select the entire column. This doesn’t always work in the Mac version (a bug), so you may need to either select with the mouse, or type in the range manually.

## Graphing in Excel

- Once the data is selected, press enter or press the red arrow to return to the main data selection menu. Repeat this process to select the Y values, and then add another series for the “Upregulated in early larval” data, and add those x and y values.
- When all of this is finished, click “ok” to return to the graph.
- Note that if an error pops up when entering data, it is probably because you clicked in multiple places, and it is expecting a single range of values. If this happens, delete everything in the white box, and then click the red arrow again.

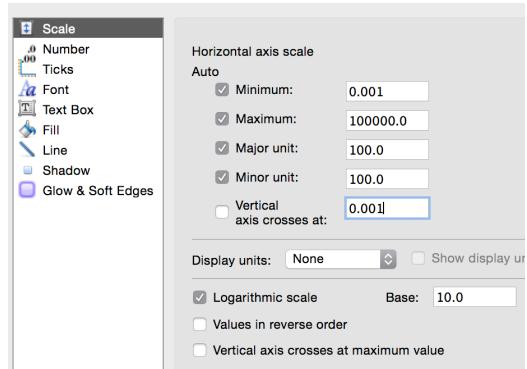
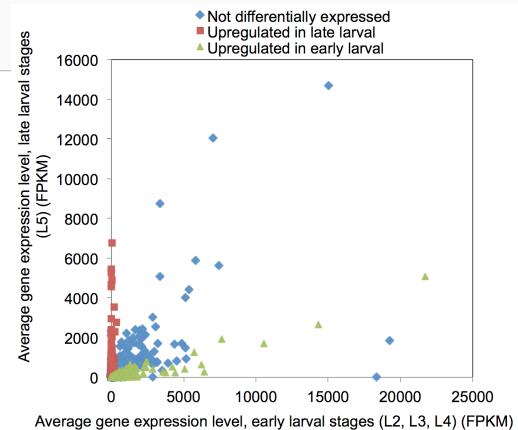


- Click OK to finish the data entry.



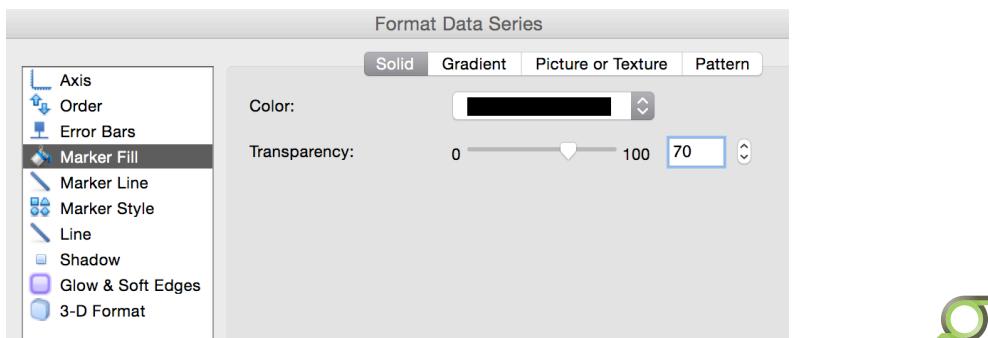
## Graphing in Excel

- Resize and reposition the legend and the graph to reduce empty white space.
- We will format the axes so that they display log values instead of natural values. Start by double-clicking on any of the numbers on the x axis.
- In the “scale” menu, check “Logarithmic scale”. You will get a warning that zero values cannot be displayed, which we will address shortly.
- Set the “vertical axis crosses at” value to 0.001, so that the axes intersect on the corner.
- Repeat both of these steps for the y-axis, except for the y axis, also set the “major unit” to 100, so that it matches the X axis.
- Although we do not need it for this graph, note that this menu is where you can manually set the minimum and maximum values for the plot.



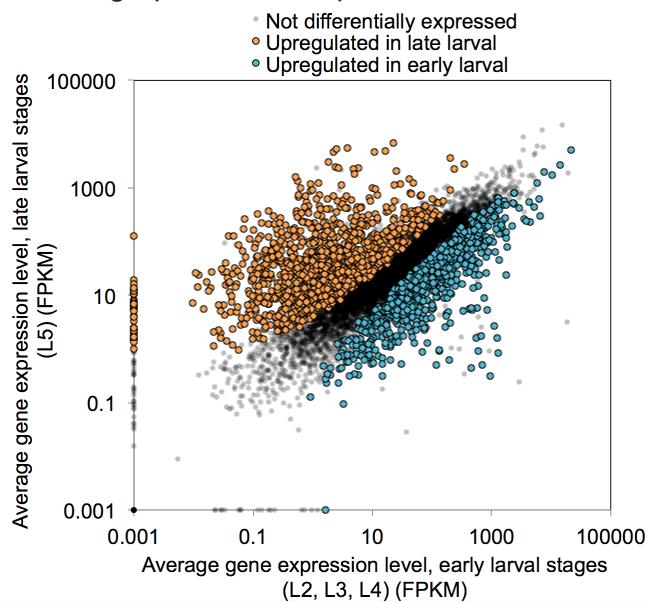
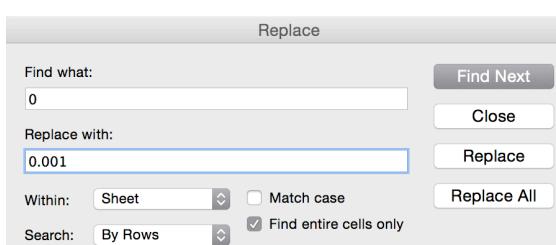
## Graphing in Excel

- Next, we will format the data series points. Start by double clicking on one of the “not differentially expressed” points. Note that if you single-click, and then double-click, you will be formatting a single point and not the entire series. Ensure that the popup window says “format data series” and not “format data point”.
- Go to “Marker style” and choose a circle, then set it to size 4. We make these points small because we want the differentially expressed genes to stand out.
- Now choose “Marker line” and choose “no line”. This is for the border around each point which we don’t want for this series.
- Go to “marker fill”, and set to black with 70% transparency. This will make the points translucent, making it easier to tell where they overlap. Click ok to finish formatting.
- Repeat for the two upregulated gene sets, except choose a size 5 circle, a black marker line, and a solid fill with no transparency (orange and blue).



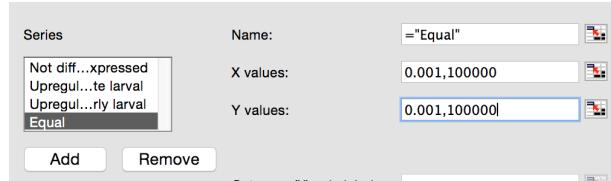
## Graphing in Excel

- Now, we will fix the zero values. Rather than not including points with zero expression, we want them to show up along the axis. We will do this by changing all zero values in the graph data to 0.001.
- Go back to the FPKM GraphData tab, and press “command/CTRL + F” to bring up the “Find” dialog. From here, click “replace”, and check off “find entire cells only”. Use this to replace all zero-value cells with 0.001. The graph will auto-update since the cell references are still linked.
- Now, the points plotted along the axes are zero-value, and not 0.001 as indicated. This can either be mentioned in the figure caption, or the 0.001 values can later be covered up in imaging software and replaced with 0 on the plot.

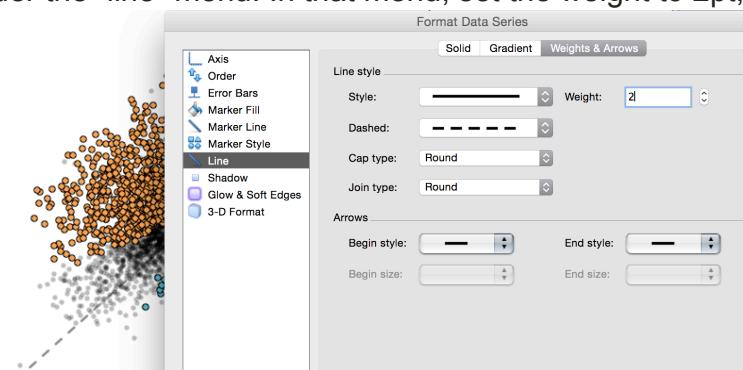


## Graphing in Excel

- Finally, we will add a diagonal line to define where the x and y values are equal. To do this, go back to the “select data” menu (right click the empty space on the graph).
- Now add another series called “Equal”. Manually type in the values 0.001,100000 to both the x and y axis values, then click OK.

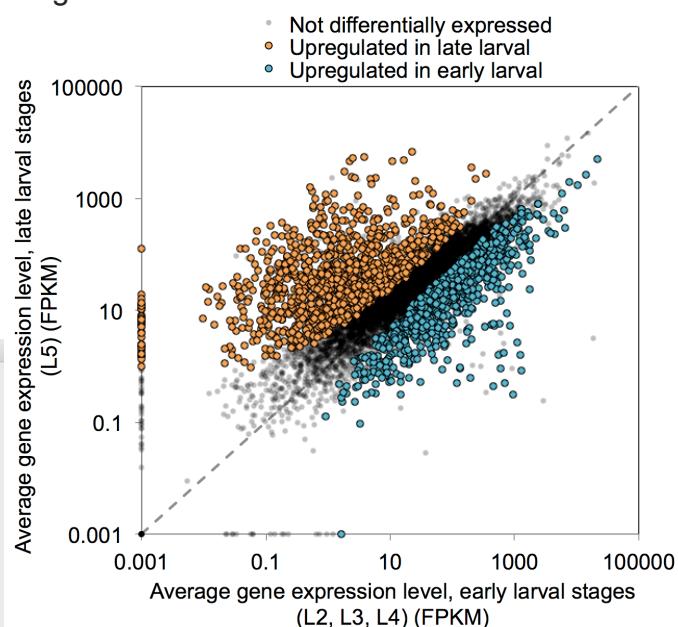
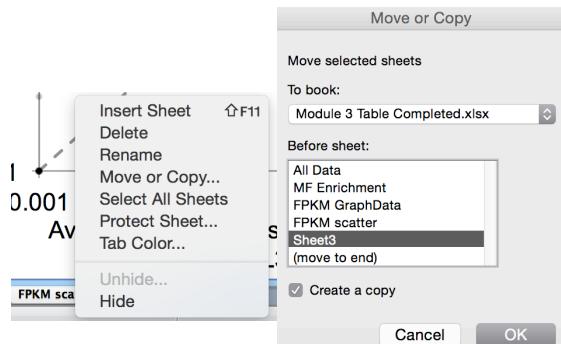


- Two points will show up in the corner. Double click one of them, then set the Marker style to “no marker”, the “line” color to dark grey, and then click to the “weights & arrows” dialog under the “line” menu. In that menu, set the weight to 2pt, and choose a dashed line:



## Graphing in Excel

- If “equal” shows up in the legend, click it and delete it.
- At this point, the graph is complete. This can be saved as a PDF file in the “save as” menu, and imported as a vector-format image into other software.
- You can make a copy of the graph by right clicking the sheet tab at the bottom, and choosing “Move or Copy...”, and then specifying to create a copy. This way, if you make a second scatterplot, you can just change the series data, and keep all of the formatting.



---

## Helpful resources for Section 2

---

- List of RNA-seq bioinformatics tools:
  - [https://en.wikipedia.org/wiki/List\\_of\\_RNA-Seq\\_bioinformatics\\_tools](https://en.wikipedia.org/wiki/List_of_RNA-Seq_bioinformatics_tools)
- khmer website and blog
  - <http://khmer-protocols.readthedocs.org/en/v0.8.2/mrnaseq/index.html>
  - <http://ivory.idyll.org/blog/category/science.html>
- DESeq2
  - <https://www.bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.pdf>
  - <http://www.bioconductor.org/help/workflows/rnaseqGene/>
- GOstats
  - <https://bioconductor.org/packages/release/bioc/vignettes/GOstats/inst/doc/GOstatsHyperG.pdf>



---

## **Section 3: Variome**

### **Module 0: Re-sequencing genomes**

---

Analysis of genetic variation is central to understanding population biology and molecular epidemiology of helminth parasites. Studying genome variations within and between populations can provide insights into geographical differentiation and gene flow, transmission patterns and evolution of parasites. In addition, genome-wide association studies (GWAS) and forward genetic screens (mapping-by-sequencing) can greatly facilitate identification of genetic variants correlated with phenotypes of biomedical interests (e.g., infection behavior, drug resistance, etc.)

NGS provides an unprecedented opportunity to characterize genetic variation in large number of samples at a reasonable cost. Sequencing individuals at a high coverage is the 'gold standard' for obtaining high-quality data, but budget constraints may require alternatives for studying large populations. Reduced representation and pooled sequencing approaches can be cost-efficient, but it is important to understand the strengths and weaknesses of each method to strategically design your experiment.

The following modules in this section will help you understand how we can turn raw sequencing data into reliable information about genetic variation.

#### **Recommended reading:**

DePristo, M. A., E. Banks, R. Poplin, K. V. Garimella, J. R. Maguire, C. Hartl, A. A. Philippakis, G. del Angel, M. A. Rivas, M. Hanna, A. McKenna, T. J. Fennell, A. M. Kernytsky, A. Y. Sivachenko, K. Cibulskis, S. B. Gabriel, D. Altshuler and M. J. Daly (2011). "A framework for variation discovery and genotyping using next-generation DNA sequencing data." *Nat Genet* **43**(5): 491-498.

Nielsen, R., J. S. Paul, A. Albrechtsen and Y. S. Song (2011). "Genotype and SNP calling from next-generation sequencing data." *Nat Rev Genet* **12**(6): 443-451.

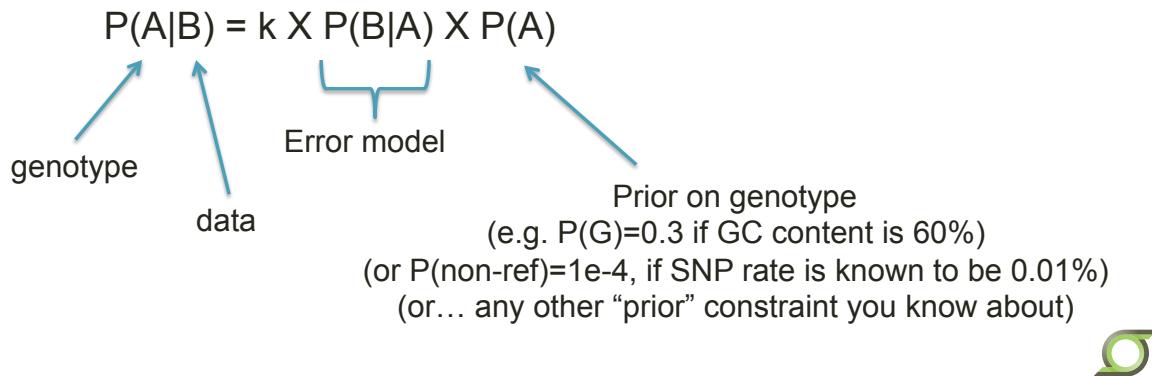
Schlotterer, C., R. Tobler, R. Kofler and V. Nolte (2014). "Sequencing pools of individuals - mining genome-wide polymorphism data without big funding." *Nat Rev Genet* **15**(11): 749-763.

---

## Variome – introduction (cont'd)

---

- Not all mismatches are SNPs!
  - Errors in library preparation/basecalling/mapping etc.
- The basic idea behind finding probability of bases at a locus (genotype likelihoods) using Bayes theorem



---

## Some SNP calling programs

---

	published	citations
--	-----------	-----------

CRISP	2010	92
-------	------	----

SNVer	2011	86
-------	------	----

Samtools	2011	176
----------	------	-----

GATK (Genome Analysis Tool Kit)	2011	>2000
------------------------------------	------	-------

SomaticSniper	2012	128
---------------	------	-----

VarScan-2	2012	404
-----------	------	-----



---

# Genome Analysis Tool Kit

---

Developed at The Broad Institute, Cambridge, MA

Installation: download directly from GATK website

Java Usage: a single jar file (except some preprocessing steps, which use bwa and picard tools)

Help for anything related to GATK, available at GATK website (with Guide, tools documentation and best practices)

Specifically, it is highly recommended to read the best practices before (or while) using GATK:

<https://www.broadinstitute.org/gatk/guide/best-practices>

The use forums (<http://gatkforums.broadinstitute.org/>) are also great, with usually very prompt responses by the GATK team



---

## Before we start...

---

All figures in Module 1 and 2 are courtesy GATK online material (used here with permission)

Our dataset : 4 samples from male *D. viviparus* worms

We selected just 2 contigs for illustration (You will usually do this on the whole genomes of your worm of interest, so your SNP calling will take more than the 2 hours we have here!)

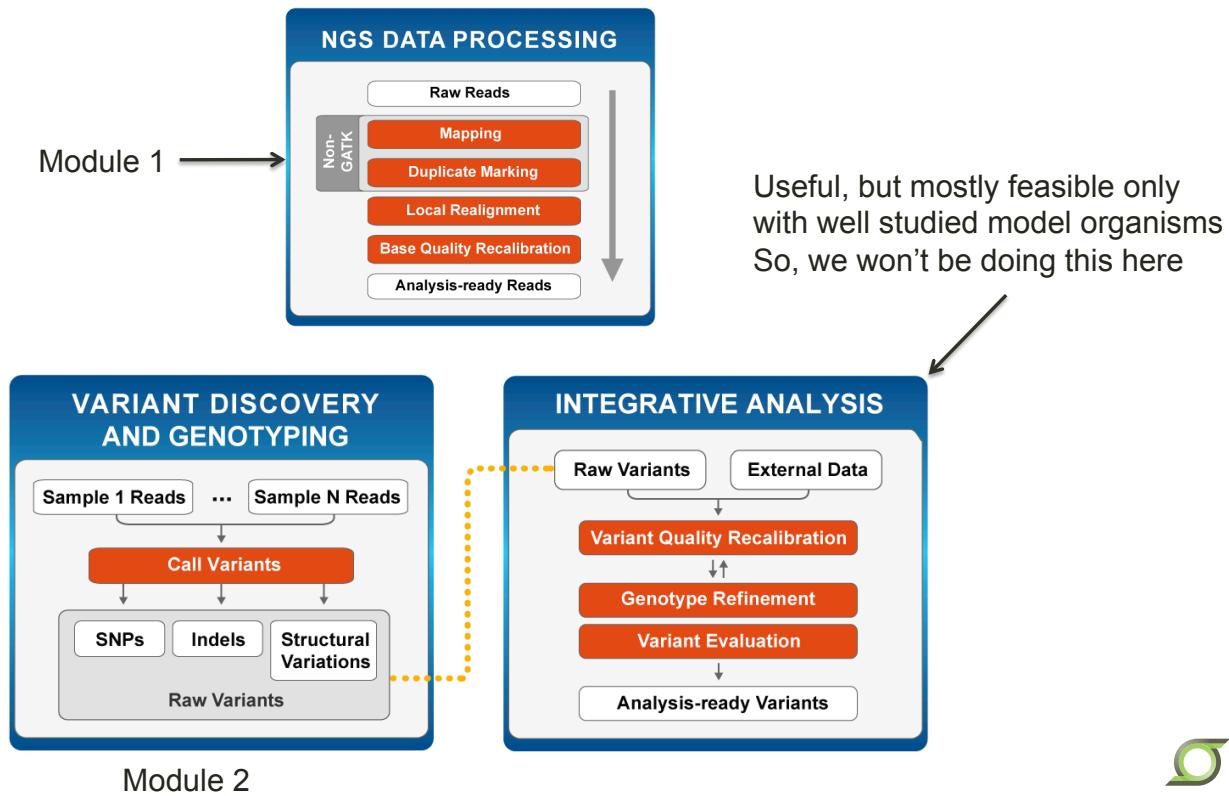
Starting data :

paired end reads in fastq format  
("Section\_3/module\_1/bwa/reads/S1\_1.fastq" etc)

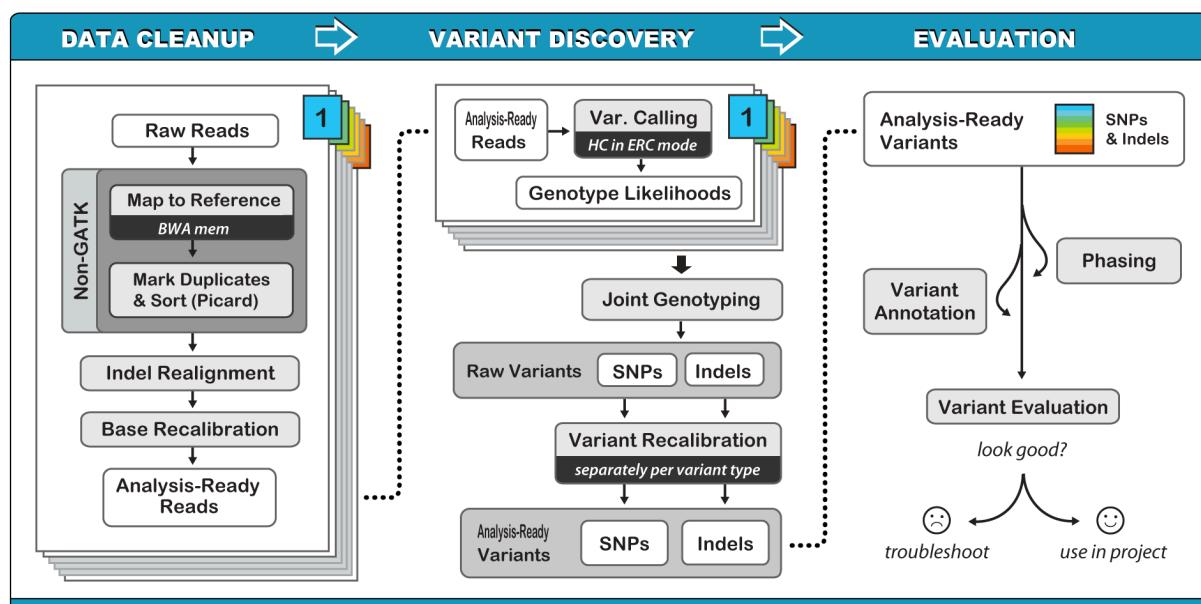
reference sequence and annotation  
(fasta, bed and gff3 files in "Section\_3/reference" directory)



# About GATK: Overall flow



## GATK Process map



---

## **Section 3: Variome**

### **Module 1: Processing and alignment**

---



---

## **Preparing reference file and mapping**

---

**Preparing reference file**  
(**You are here -> “[Section\\_3/reference](#)”**)

```
bwa index reference.fasta  
  
samtools faidx reference.fasta  
  
java -jar ~/bin/picard-tools-1.101/  
CreateSequenceDictionary.jar R=reference.fasta  
O=reference.dict
```

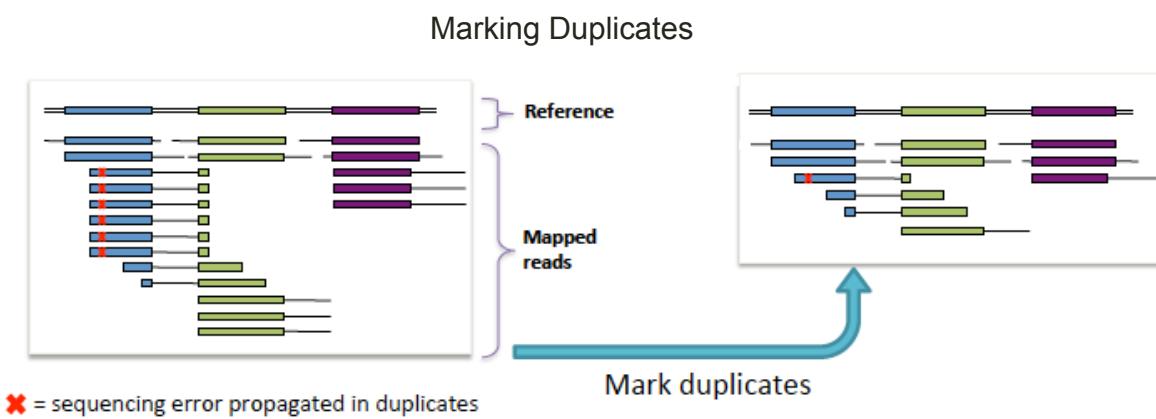
Mapping using bwamem  
([Section\\_3/module\\_1/bwa](#))

Important information about reads is also encoded simultaneously (library name, sample name, read group etc). These are useful for analysis later.

```
cd ../module_1/bwa  
  
for i in S1 S2 S3 S4;do bwa mem -t 8 -M -R "@RG  
\tID:$i\tPL:illumina\tPU:$i\tRG1_UNIT1\tLB:$i-  
lib1\tSM:$i ../../reference/reference.fasta  
reads/$i_1.fastq reads/$i_2.fastq >$i.bwa.1.sam;done
```



## Duplicate reads



For correct estimation of variant likelihoods, we need our reads to represent the correct proportions of molecules in the library. (actually we also want our library to represent the proportions of original biological sample, and should be wary of biases introduced by PCR etc, but right now let's worry only about making sure we don't sequence a molecule more than once). One way of doing this is finding out which sequences are highly likely to originate from the same DNA fragment, and then removing all but one of that set.



## Recognizing duplicates

### Marking Duplicates

Finding reads that start at the same location. And, if paired end, that have their partners also mapping at the same starting location.

We can't simply compare the read sequences because sequencing is error prone and will likely lead to high underestimation of duplicates.

Pos	1	2	3	4	5	6	7	8	9
Ref	T	A	G	C	C	G	A	T	C
r1	T	A	G	C	C	G	A		
r2	T	A	G	C	C	G	A		
r3	T	A	-	C	CAG	A			
r4	T	A	G	C	C	H	H		
r5	T	A	G	C	C	G	A	T	C
r6	S	S	G	C	C	G	A		
r7			G	C	C	G	A		

Blue maps to forward strand  
Orange maps to reverse strand  
Grey bases are clipped

Underlined is the expected 5' start of the read, given the mapping

So...what are the duplicate sets?



# Removing Duplicates

(Section\_3/module\_1/bwa)

Sort and convert to bam

```
for i in S1 S2 S3 S4;do samtools view -bs "$i".bwa.sam | samtools sort - "$i".bwa.sorted;done
```

Removing duplicates with Picard tools

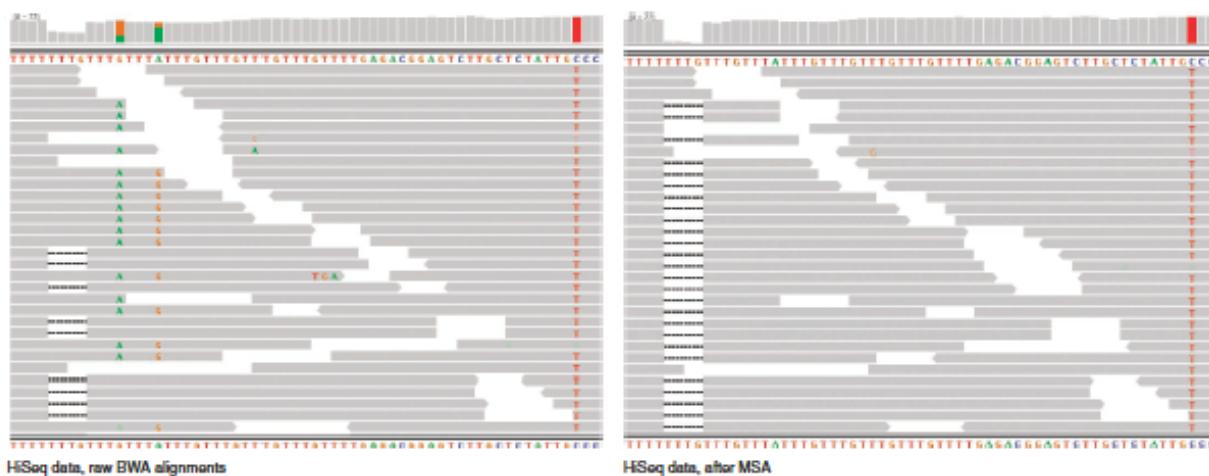
```
for i in S1 S2 S3 S4;do java -Xmx8g -jar ~/bin/picard.jar MarkDuplicates MAX_FILE_HANDLES_FOR_READ_ENDS_MAP=1000 REMOVE_DUPLICATES=true INPUT="$i".bwa.sorted.bam OUTPUT="$i".dedup.bam METRICS_FILE="$i".dedup_metrics ASSUME_SORTED=true;done
```

Then you can look at some examples of before-and-after deduplication reads/alignment using “samtools faidx” and “samtools tview” (or IGV)



# Refining Alignments

Read aligners like bwa etc look at every read independently and try to find the best alignment for every read. This may lead to spurious SNPs because of slightly “off target” mappings, especially in presence of small indels (e.g. left figure below). Realigning all such reads in this region simultaneously by making use of multiple sequence alignment algorithms leads to more concordant alignments. This gets rid of many false positive SNPs which are merely mapping artifacts (right figure below)



# Realignment around indels

([Section\\_3/module\\_1/bwa](#))

Index our de-duplicated bam files

```
for i in S1 S2 S3 S4;do samtools index "$i".dedup.bam;done
```

Find intervals to analyze

```
for i in S1 S2 S3 S4;do java -Xmx8g -jar ~/bin/
GenomeAnalysisTK.jar -T RealignerTargetCreator -R ~/WORKSHOP_RESOURCES/Section_3/reference/reference.fasta -I
"$i".dedup.bam -o "$i".realignment.intervals;done
```

Realign in these loci

```
for i in S1 S2 S3 S4;do java -Xmx8g -jar ~/bin/
GenomeAnalysisTK.jar -T IndelRealigner -R ~/WORKSHOP_RESOURCES/Section_3/reference/reference.fasta -I "$i".dedup.bam -
targetIntervals "$i".realignment.intervals -o
"$i".dedup.realigned.bam;done
```

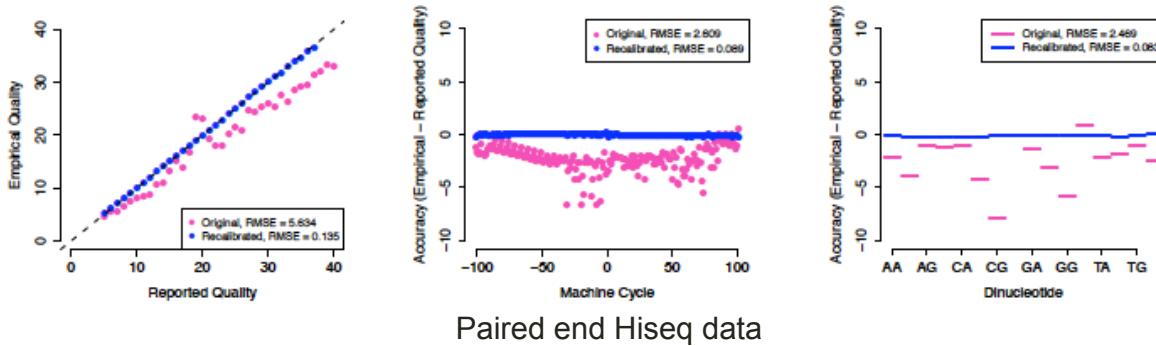


# Base Recalibration (in presence of a truth set)

To improve base quality values, mismatches with reference are analyzed. Assuming that any mismatch which isn't a known SNP is an “error”, base qualities can be readjusted to more closely model the reality (removing systematic errors in original base quality reports).

However, this can only be done in the presence of a substantial set of known True Positives (i.e. a large set of known SNPs). Since we don't have that (yet), we'll skip this and come back to it later...

The figure below shows the result of recalibrating errors from original reported qualities to those obtained using mapping data (after filtering out known SNPs).



---

## Section 3: Variome

### Module 2: Variant calling

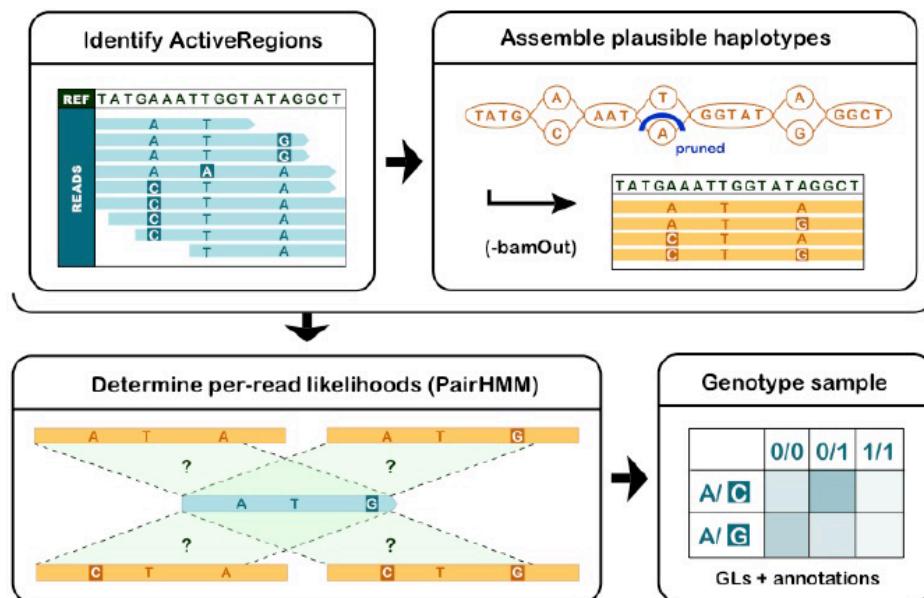
---



---

## Introduction

HaplotypeCaller is the workhorse of GATK's variant calling process. It calls variants by assembling reads in "active regions" into haplotypes (completely independent of reference sequence mapping) and then estimating likelihoods of genotypes at variant loci based on how well each read represents those assembled haplotypes.



---

# Running HaplotypeCaller

---

([Section\\_3/module\\_2/haplo](#))

## Prepare files

```
cd ../../module_2  
mkdir haplo  
cd haplo  
  
for i in S1 S2 S3 S4;do ln -s ../../module_1/  
bwa/"$i".dedup.realigned.bam;done  
  
for i in S1 S2 S3 S4;do ln -s ../../module_1/  
bwa/"$i".dedup.realigned.bai;done
```

## Run HaplotypeCaller with GVCF

```
for i in S1 S2 S3 S4;do java -Xmx8g -jar ~/bin/  
GenomeAnalysisTK.jar -T HaplotypeCaller -R ~/  
WORKSHOP_RESOURCES/Section_3/reference/reference.fasta -I  
"$i".dedup.realigned.bam -ERC GVCF -ploidy 2 -o  
"$i".dedup.g.vcf;done
```



---

# Some default settings for HaplotypeCaller

---

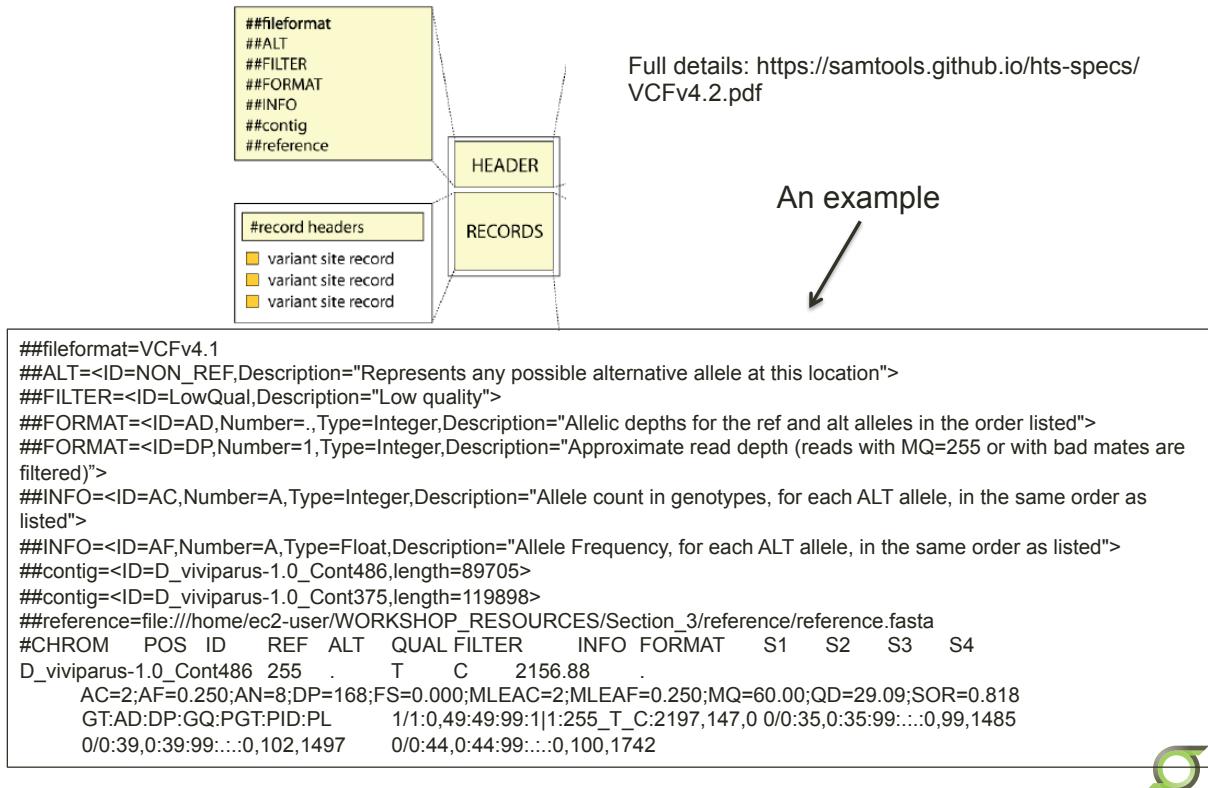
--maxReadsInRegionPerSample	10000
--min_base_quality_score	10
--minReadsPerAlignmentStart	10
--sample_ploidy	2
--standard_min_confidence_threshold_for_calling	30.0
--standard_min_confidence_threshold_for_emitting	30.0
--max_alternate_alleles	6
--maxNumHaplotypesInPopulation	128

See Details at

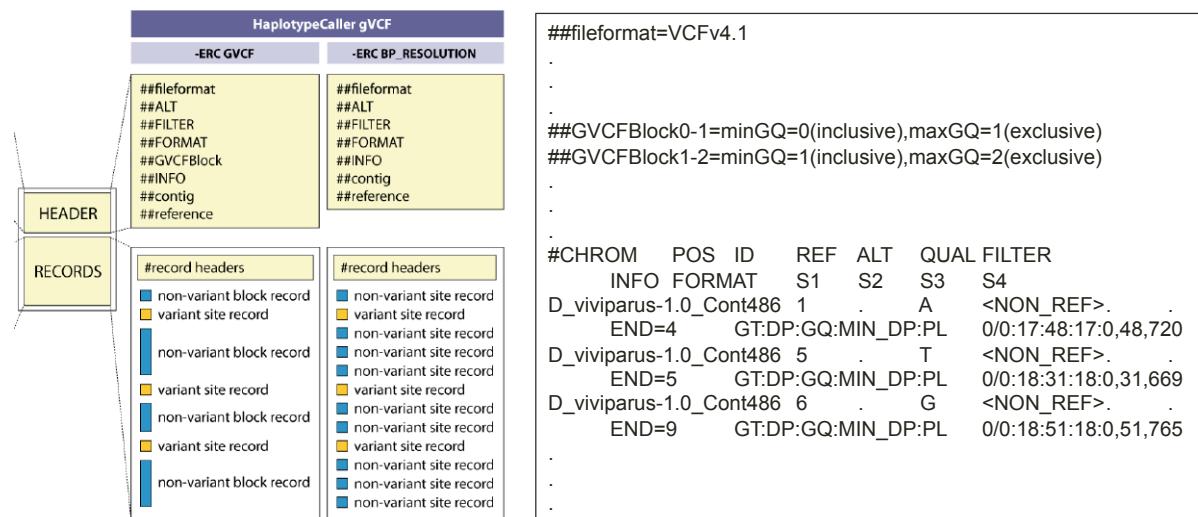
[https://www.broadinstitute.org/gatk/gatkdocs/org\\_broadinstitute\\_gatk\\_tools\\_walkers\\_haplotypecaller\\_HaplotypeCaller.php](https://www.broadinstitute.org/gatk/gatkdocs/org_broadinstitute_gatk_tools_walkers_haplotypecaller_HaplotypeCaller.php)



# The VCF file format



## Using GVCFs to combine sample-wise variants



([Section\\_3/module\\_2/haplo](#))

```

java -Xmx8g -jar ~/bin/GenomeAnalysisTK.jar -T GenotypeGVCFs -R
~/WORKSHOP_RESOURCES/Section_3/reference/reference.fasta $(for
i in S1 S2 S3 S4; do echo -n "--variant \"$i\".dedup.g.vcf "; done)
-o all_raw.vcf

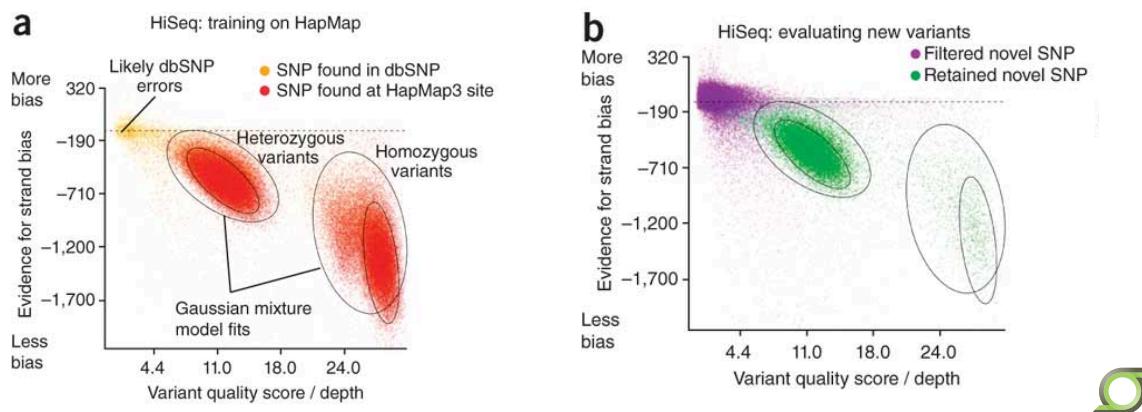
```



# Variant Quality recalibration for refinement

To get a higher confidence set of real SNPs, we can look at a truth set (if we have one) of real SNPs and analyze what values various relevant metrics take for them. e.g. you may just pick up very rare (and potentially spurious) SNPs just because of very high depth of coverage. Looking at various metrics ( Variant quality score/Depth, strand bias etc) may separate real SNPs with False Positives (figures below).

So, first we calibrate using known SNPs, then use those calibrations to filter out potential False Positives and obtain a final analysis-ready variant set.



## Using hard filters

However, last page is useless for us since we don't actually have a truth set.

We still want to set up a filter to refine our raw variant set. So, we'll use some hard filters (i.e. thresholds pre-decided rather than dynamically calibrated based on data). We will use values recommended by GATK best practices, though these numbers can be changed based on any insight you may have into your specific case.

QD	: Quality by Depth	< 2.0
FS	: FisherStrand	> 60.0
MQ	: RMS Mapping Quality	< 40.0
MQRankSum	: Mapping Quality Rank Sum	< -12.5
ReadPosRankSum	: Read Position Rank Sum	< -8.0

In addition, we will also apply a depth of coverage filter (even though GATK team advises that it isn't as critical with HaplotypeCaller as with its older and almost obsolete cousin "UnifiedGenotyper"). We just want high confidence SNPs to generate a raw "truth set". So, we'll apply a relatively strict Depth filter. GATK used to suggest Depth of Coverage (DP) > (mean+5\*sd).

We will use DP > (median + 2\*MAD)



---

## Setting stage for filtering SNPs

---

Collecting SNPs and getting coverage  
([Section\\_3/module\\_2/var\\_filt](#))

### Prepare Files

```
cd ..  
mkdir var_filt  
cd var_filt/  
ln -s ../../haplo/all_raw.vcf
```

### Extract SNPs from the “raw” vcf file

```
java -Xmx8g -jar ~/bin/GenomeAnalysisTK.jar -T  
SelectVariants -R ~/WORKSHOP_RESOURCES/Section_3/  
reference/reference.fasta -V all_raw.vcf -selectType SNP -  
o raw_snps.vcf
```



---

## Getting DP filter threshold

---

Collecting SNPs and getting coverage  
([Section\\_3/module\\_2/var\\_filt](#))

Finding base-wise coverages over the reference contigs (in order to find the DP filter threshold)

```
for i in S1 S2 S3 S4;do ln -s ../../module_1/  
bwa/"$i".dedup.realigned.bam;done  
  
for i in S1 S2 S3 S4;do coverageBed -abam  
"$i".dedup.realigned.bam -b ../../reference/  
reference.fasta.bed -d >"$i".coverage.bed;done
```

We will find the median and MAD (median absolute deviation) in R. This is done after adding the depths over all the samples:

```
S1<-read.table("S1.coverage.bed",header=F,stringsAsFactors=F)  
S2<-read.table("S2.coverage.bed",header=F,stringsAsFactors=F)  
S3<-read.table("S3.coverage.bed",header=F,stringsAsFactors=F)  
S4<-read.table("S4.coverage.bed",header=F,stringsAsFactors=F)  
sum<-S1$V6+S2$V6+S3$V6+S4$V6  
summary(sum[sum<=(median(sum)+(2*mad(sum)))] )
```



---

## Applying SNP filters

---

Since we are only using DP to get a strict set for the purpose of base recalibration, we are sloppy here and using bedtools coveragebed utility to get coverage (also, partly because we want to introduce you to the convenient and useful coveragebed utility). If you really want to get proper depth numbers to set your DP filter, you should use the DepthofCoverage tool of GATK itself (as it takes care of any base filters that are applied in GATK before counting depths).

Also, remember that DP doesn't need to be used with HaplotypeCaller, and we won't use it to get our final SNP set anyway.

Now, we can apply our SNP filter!

```
java -Xmx8g -jar ~/bin/GenomeAnalysisTK.jar -T VariantFiltration -R ~/WORKSHOP_RESOURCES/Section_3/reference/reference.fasta -V raw_snps.vcf -o raw_snps_filtered.vcf --filterExpression " QD < 2.0 " --filterName "QD" --filterExpression " FS > 60.0 " --filterName "FS" --filterExpression " MQ < 40.0 " --filterName "MQ" --filterExpression " MQRankSum < -12.5 " --filterName "MQRankSum" --filterExpression " ReadPosRankSum < -8.0 " --filterName "ReadPosRankSum" --filterExpression " DP > 268 " --filterName "DP"
```



---

## Applying indel filters

---

([Section\\_3/module\\_2/var\\_filt](#))

Now, we'll repeat filtering with indels too (using separate thresholds recommended by GATK best practices)

```
java -Xmx8g -jar ~/bin/GenomeAnalysisTK.jar -T SelectVariants -R ~/WORKSHOP_RESOURCES/Section_3/reference/reference.fasta -V all_raw.vcf -selectType INDEL -o raw_indels.vcf
```

```
java -Xmx8g -jar ~/bin/GenomeAnalysisTK.jar -T VariantFiltration -R ~/WORKSHOP_RESOURCES/Section_3/reference/reference.fasta -V raw_indels.vcf -o raw_indels_filtered.vcf --filterExpression " QD < 2.0 " --filterName "QD" --filterExpression " FS > 200.0 " --filterName "FS" --filterExpression " ReadPosRankSum < -20.0 " --filterName "ReadPosRankSum"
```



---

## Combining variants

---

([Section\\_3/module\\_2/var\\_filt](#))

We can now combine the SNPs and indels into a single variants file that can be used as a “truth set” to recalibrate bases (that we talked about in Module 1)

```
java -Xmx8g -jar ~/bin/GenomeAnalysisTK.jar -T
CombineVariants -R ~/WORKSHOP_RESOURCES/Section_3/
reference/reference.fasta --variant raw_snps_filtered.vcf
--variant raw_indels_filtered.vcf -o
raw_combined_filtered.vcf -genotypeMergeOptions UNSORTED --
printComplexMerges
```



---

## Using Variant set for base quality recalibration

---

([Section\\_3/module\\_1/bwa](#))

Prepare Files, get recalibration data and apply it to update base quality values

```
cd ../../module_1/bwa
ln -s ../../module_2/var_filt/raw_combined_filtered.vcf

for i in S1 S2 S3 S4;do java -Xmx8g -jar ~/bin/
GenomeAnalysisTK.jar -T BaseRecalibrator -R ~/
WORKSHOP_RESOURCES/Section_3/reference/reference.fasta -I
"${i}".dedup.realigned.bam -knownSites raw_combined_filtered.vcf
-o "${i}".recal_data.table;done

for i in S1 S2 S3 S4;do java -Xmx8g -jar ~/bin/
GenomeAnalysisTK.jar -T PrintReads -R ~/WORKSHOP_RESOURCES/
Section_3/reference/reference.fasta -I
"${i}".dedup.realigned.bam -BQSR "${i}".recal_data.table -o
"${i}".recal_reads.bam;done
```



---

## Variant Calling again

With this presumably better set of base qualities, we'll repeat our earlier steps for variant calling (i.e. haplotypecaller followed by combining the sample GVCFs)

([Section\\_3/module\\_2/haplo](#))

```
cd ../../module_2/haplo

for i in S1 S2 S3 S4;do ln -s ../../module_1/
bwa/"$i".recal_reads.bam;done
for i in S1 S2 S3 S4;do ln -s ../../module_1/
bwa/"$i".recal_reads.bai;done

for i in S1 S2 S3 S4;do java -Xmx8g -jar ~/bin/
GenomeAnalysisTK.jar -T HaplotypeCaller -R ~/WORKSHOP_RESOURCES/
Section_3/reference/reference.fasta -I "$i".recal_reads.bam -ERC
GVCF -ploidy 2 -o "$i".recal.g.vcf -bamout
"$i".recal.haplo.bam ;done

java -Xmx8g -jar ~/bin/GenomeAnalysisTK.jar -T GenotypeGVCFs -R ~/
WORKSHOP_RESOURCES/Section_3/reference/reference.fasta $(for i in
S1 S2 S3 S4;do echo -n "--variant \"$i".recal.g.vcf ";done) -o
all_recal.vcf
```

“-bamout” option is just to get a bam file which can then be visualized using IGV or “samtools tview” if you want to look at something closely.



---

## Final SNPs hard filtering

We will again filter the variants with the hard filters introduced before. While we will stop here for the demonstration, usually one wants to see some sort of convergence of results before stopping. So, if you see a significant change in the number of variants detected as compared to the last round, you can do the same cycle all over again (i.e. using SNPs to recalibrate bases followed by calling and filtering variants again)

([Section\\_3/module\\_2/var\\_filt](#))

```
cd ./var_filt/
ln -s ./haplo/all_recal.vcf

java -Xmx8g -jar ~/bin/GenomeAnalysisTK.jar -T SelectVariants -R
~/WORKSHOP_RESOURCES/Section_3/reference/reference.fasta -V
all_recal.vcf -selectType SNP -o recal_snps.vcf

java -Xmx8g -jar ~/bin/GenomeAnalysisTK.jar -T VariantFiltration
-R ~/WORKSHOP_RESOURCES/Section_3/reference/reference.fasta -V
recal_snps.vcf -o recal_snps_filtered.vcf --filterExpression " QD
< 2.0 " --filterName "QD" --filterExpression " FS > 60.0 " --
filterName "FS" --filterExpression " MQ < 40.0 " --filterName "MQ"
--filterExpression " MQRankSum < -12.5 " --filterName "MQRankSum"
--filterExpression " ReadPosRankSum < -8.0 " --filterName
"ReadPosRankSum"
```



---

## Final indel hard filtering

---

We do apply hard filters for indels again.

([Section\\_3/module\\_2/var\\_filt](#))

```
java -Xmx8g -jar ~/bin/GenomeAnalysisTK.jar -T SelectVariants  
-R ~/WORKSHOP_RESOURCES/Section_3/reference/reference.fasta -V  
all_recal.vcf -selectType INDEL -o recal_indels.vcf

java -Xmx8g -jar ~/bin/GenomeAnalysisTK.jar -T  
VariantFiltration -R ~/WORKSHOP_RESOURCES/Section_3/reference/  
reference.fasta -V recal_indels.vcf -o  
recal_indels_filtered.vcf --filterExpression " QD < 2.0 " --  
filterName "QD" --filterExpression " FS > 200.0 " --filterName  
"FS" --filterExpression " ReadPosRankSum < -20.0 " --filterName  
"ReadPosRankSum"
```



---

## Combining variants for further analysis

---

Combining SNPs and indels gives a common variant file which can be used for further analysis. In our case, we have a pre-generated file which will be used in Module 4

```
java -Xmx8g -jar ~/bin/GenomeAnalysisTK.jar -T  
CombineVariants -R ~/WORKSHOP_RESOURCES/Section_3/reference/  
reference.fasta --variant recal_snps_filtered.vcf --variant  
recal_indels_filtered.vcf -o recal_combined_filtered.vcf -  
genotypeMergeOptions UNSORTED --printComplexMerges
```

As said before, you should compare the change in variants after this round of recalibration and calling (but we will move on to Module 4 regardless of the change!)



---

## **Section 3: Variome**

### **Module 2: Variant calling (cont' ed)**

---

#### **Visualization of variants**

Variants in VCF format can be visualized using the Integrative Genomics Viewer (IGV), a high-performance visualization tool for interactive exploration of large, integrated genomic datasets (<http://www.broadinstitute.org/igv/>). IGV supports a wide variety of data types, including next-generation sequence data and genomic annotations.

#### **Options for installing and running IGV**

(<http://www.broadinstitute.org/software/igv/download>)

1. (Mac only) Download and run the Mac application; or
2. (All systems) Use the Java Web Start buttons; or
2. (All systems) Download the binary distribution and run IGV from the command line.

#### **Creating a .genome File**

1. Click Genomes>Create .genome File. IGV displays a window where you enter the information.
2. Enter an ID and a descriptive name for the genome (e.g., D\_viviparus).
3. Enter the path to the FASTA file for the genome (reference.fasta). If the FASTA file has not already been indexed, an index will be created during the import process. This will generate a file with a ".fai" extension which must be in the same directory as the FASTA file.
4. Specify the gene file (reference.gff3).
5. Click Save. IGV displays the Genome Archive window.
6. Select the directory in which to save the genome archive (\*.genome) file and click Save. IGV saves the genome and loads it into IGV.

#### **Loading data**

1. Select File>Load from File. IGV displays the Select Files window.
2. Select one or more data files or sample information files, then click OK.

Please load the following files:

recal\_combined\_filtered.vcf  
S1.recal.haplo.bam  
S1.dedup.realigned.bam

---

---

## **Section 3: Variome**

### **Module 3: Variant annotation**

---

Using SnpEff (<http://snpeff.sourceforge.net>), we will annotate and predict the effects of variants on genes (such as amino acid changes). SnpEff is written in Java and runs on Unix/Linux, OSX and Windows. It accepts input files in VCF/BED format, and can provide consequence terms defined by the Sequence Ontology

(<http://www.sequenceontology.org>) and in HGVS notation (<http://www.hgvs.org/mutnomen/>).

## Building databases

SnpEff needs a database to perform genomic annotations. In order to build a database for a new genome, you need to:

1. Configure a new genome in SnpEff's config file.

1a. Add genome entry to SnpEff's configuration by editing the.snpEff.config file.

```
gedit ~/bin/snpEff/snpEff.config
```

Add the following lines, save the file and exit gedit.

```
# Dictyocaulus_viviparus
D_viviparus.genome : Dictyocaulus_viviparus
```

1b (optional). If the genome uses a non-standard codon table, add codon table parameter. Please see SnpEff documentation for detail ([http://snpeff.sourceforge.net/SnpEff\\_manual.html](http://snpeff.sourceforge.net/SnpEff_manual.html)).

2. Create a directory for this new genome.

```
mkdir ~/bin/snpEff/data/D_viviparus/
```

3. Get the reference genome sequence in FASTA format.

```
ln -s ~/WORKSHOP_RESOURCES/Section_3/reference/reference.fasta
~/bin/snpEff/data/D_viviparus/sequences.fa
```

4. Get genome annotations from GFF file.

```
ln -s ~/WORKSHOP_RESOURCES/Section_3/reference/reference.gff3
~/bin/snpEff/data/D_viviparus/genes.gff
```

5. Build a SnpEff database.

```
java -Xmx8g -jar ~/bin/snpEff.jar build -gff3 -v D_viviparus
```

You can check the database to see if the features (genes, exons, UTRs, etc.) have been correctly incorporated, by taking a look at the database.

```
java -Xmx8g -jar ~/bin/snpEff.jar dump D_viviparus | less
```

## Running SnpEff

1. Change directory to where the SnpEff output files will be saved.

```
cd ~/WORKSHOP_RESOURCES/Section_3/module_3
```

2. You can annotate the vcf file by running the following command. Command line option -v switches on the "verbose" mode, which can be useful for debugging.

```
java -Xmx8g -jar ~/bin/snpEff.jar -v D_viviparus  
~/WORKSHOP_RESOURCES/Section_3/module_2/var_filt/recal_combined_f  
iltered.vcf > recal_combined_filtered.eff.vcf
```

SnpEff adds annotation information ('ANN' tag) to the INFO field of a VCF file. The INFO field is the eighth column of a VCF file. SnpEff updates the header of the VCF file to add the command line options used to annotate the file as well as SnpEff's version, so you can keep track of what exactly was done.

```
less recal_combined_filtered.eff.vcf
```

3. SnpEff creates an additional output file showing overall statistics. This "stats" file is an HTML file, which can be opened using a web browser.

```
chrome.snpEff_summary.html
```

4. SnpEff also generates a (tab separated) TXT file having counts of number of variants affecting each transcript and gene.

```
head.snpEff_genes.txt
```

### **Filter and manipulate annotated VCF files using SnpSift**

1. Once your genomic variants have been annotated, you need to filter them out in order to find the "interesting/relevant variants". SnpSift helps to perform this VCF file manipulation and filtering. It can be used to extract fields from a VCF file to a tab separated TXT format that you can easily load in R, Excel, etc.

```
cat recal_combined_filtered.eff.vcf |  
~/bin/snpEff/scripts/vcfEffOnePerLine.pl | java -Xmx8g -jar  
~/bin/SnpSift.jar extractFields - CHROM POS REF ALT AF  
"ANN[*].ALLELE" "ANN[*].EFFECT" "ANN[*].IMPACT" "ANN[*].GENE"  
"ANN[*].HGVS_C" "ANN[*].HGVS_P" > recal_combined_filtered.eff.txt  
  
head recal_combined_filtered.eff.txt
```

2. You can now easily list, for instance, the coding variants identified in your genes of interest (e.g., DICVIV\_10165 and DICVIV\_11294)

```
cat recal_combined_filtered.eff.txt | grep -v "MODIFIER" | grep -  
E "DICVIV_10165|DICVIV_11294"
```

---

## **Section 4: Final topics**

### **Module 0: Finding sequence resources**

---

If you're looking for genomic sequence data, two of the best public resources available to you are the Ensembl websites and NCBI's GenBank. Ensembl maintains a collection of websites organized by higher order taxonomy: Ensembl, Ensembl Metazoa, Ensembl Bacteria, Ensembl Protists, Ensembl Fungi, Ensembl Plants, etc... while GenBank's website is a single entity. Both provide the ability to locate and download genome sequence and annotation. For most of your data needs these two sites will be your go-to resources

For Helminth specific data, your best bet will be one of the more specialized websites such as WormBase, WormBase Parasite, and our own Helminth.net websites (Nematode.net & Trematode.net). For genomic sequence data and gene annotation, the WormBase sites are very well organized and have a lot of worm data available. For worm model organisms such as *C.elegans*, the original WormBase maintains a trove of curated annotation. WormBase Parasite maintains a broad collection of genomic sequence and gene annotation on most of the currently studied parasitic helminthes. And the Helminth.net sites are a good source for finding higher order analysis and annotation.

*Useful Information:*

(Ensembl) <http://useast.ensembl.org/index.html?redirect=no>

(Ensembl Metazoa) <http://metazoa.ensembl.org/index.html>

(NCBI GenBank) <http://www.ncbi.nlm.nih.gov/genbank/>

(NCBI SRA) <http://www.ncbi.nlm.nih.gov/sra>

(Helminth.net) <http://helminth.net>

(WormBase Parasite) <http://parasite.wormbase.org/index.html>

(WormBase) <http://www.wormbase.org>

---

---

## **Section 4: Final topics**

### **Module 1: Bioinformatics packages**

---

By now you've probably realized that many of the workhorse tasks used by computational biologists already have publicly available, robust solutions. The links provided below take you to documentation for some of the packages we find most useful. When faced with a new analysis task, scan the overviews of these packages, google search, or just ask around. You'll often find that some imposing problem you are faced with has a very simple, packaged tool already available!

One resource of special note is the Galaxy platform. It's a web-based platform for developing and running bioinformatics workflows. While not appropriate for large scale processing of data, for most projects involving just a handful of samples it works very well. The Galaxy project itself developed the Galaxy platform, and their intent is for other labs to take their software and setup their own resources for others to use. But the developers themselves maintain a fantastically full featured Galaxy site themselves!

*Useful Information:*

(samtools) <http://samtools.sourceforge.net>  
(picard) <http://broadinstitute.github.io/picard/>  
(bedtools) <http://bedtools.readthedocs.org/en/latest/>  
(bamtools) <https://github.com/pezmaster31/bamtools/wiki/Using-the-toolkit>  
(list of RNA-Seq bioinformatics tools) [https://en.wikipedia.org/wiki/List\\_of\\_RNA-Seq\\_bioinformatics\\_tools](https://en.wikipedia.org/wiki/List_of_RNA-Seq_bioinformatics_tools)  
(Galaxy) <https://usegalaxy.org>

---

#### **Section 4: Final topics**

##### **Module 2: Sources of help for bioinformaticians**

---

Good places to turn if you need bioinformatics help are:

- 1) SeqAnswers – bioinformatics forum
- 2) Biostars forum
- 3) Google!

*Useful information:*

(SeqAnswers – bioinformatics forum) <http://seqanswers.com/forums/forumdisplay.php?f=18>  
(Biostars forum) <https://www.biostars.org>

---

#### **Section 4: Final topics**

##### **Module 3: Open discussion**

---

Open discussion and specific questions.