From the Schnable Lab:

Yang Zhang and Daniel Ngu's Pipeline for Processing RNA-seq Data (As of November 17, 2016)

yzhang91@unl.edu dngu2@huskers.unl.edu

Pre-processing the reads:

The alignment software used in this protocol (GSNAP) has the capacity to "soft-trim" reads during alignment, aligning only the high quality portion of the read. However, this process is reducing the speed of alignments significantly, so an initial QC step where adapters and low quality sequences are removed can substantially decrease overall runtime.

For paired end reads only:

After adapter and low quality sequence trimming, some paired end reads are completely removed, but in other cases either the forward or the reverse read is completely removed and the other side of the paired end read is retained. Quality trimming forward and reverse read files separately can therefore cause problems because many programs expect the forward and reverse reads should always be both present and present in the same order in the fastq files for forward reads and reverse reads.

We recommend using a piece of software called Trimmomatic to perform quality and adapter trimming of paired end data. (http://www.usadellab.org/cms/?page=trimmomatic)

Example commands (modified from the Trimmomatic website)

Paired End:

```
java -jar /home/linux/software/Trimmomatic-0.33/trimmomatic-
0.33.jar PE -phred33 input_forward.fq input_reverse.fq
output_forward_paired.fq output_forward_unpaired.fq
output_reverse_paired.fq output_reverse_unpaired.fq
ILLUMINACLIP:TruSeq3-PE.fa:2:30:10 LEADING:3 TRAILING:3
SLIDINGWINDOW:4:15 MINLEN:36
```

#Note: To understand more about the adapter trimming, let's go through the command line above.

```
Java -jar
```

This is an important programming language that we need to run trimmomatic (adapter trimming software). It is available from: https://java.com/en/download/faq/whatis_java.xml

```
/home/linux/software/Trimmomatic-0.33/trimmomatic-0.33.jar
```

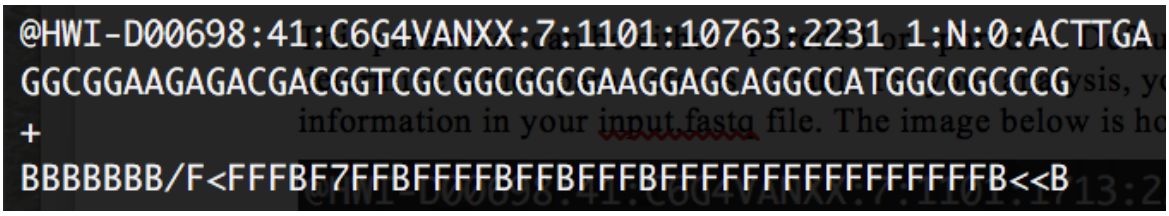
This is telling Java(programming language) the pathway where Trimmomatic that was installed.

`PE`

This can be either PE or SE. PE means Paired-end sequence whereas SE means single end sequence.

`-phred33`

This parameter can be either –phred33 or –phred64. If you do not put anything for this parameter the default will be –phred64. To determine which parameter is suitable for your raw input.fastq data, you have to understand some information in your input.fastq file. The image below is how a typical fastq file looks like.



For the 1st line, it usually starts with a "@" character and a sequence identifier.

For the 2nd line, it shows the sequence of bases.

For the 3rd line, it always begin with a "+" character.

For the 4th line, it encodes the quality for the sequence of bases in 2nd line. You can see that this line contains the same number of symbols as the letters in 2nd line. To check the phred quality score, you can go to the following link: http://www.somewhereville.com/ and refer to the Table 1. inside the website.

For my case, I have to set my parameter to –phred33 to minimize the unwanted errors. The reason why I am choosing –phred33 instead of choosing –phred64 is I can only find the "/" and "<" characters in the column that is named as (Sanger "Q + 33" ASCII GLYPH) of Table 1.

#Note: It is very very important to determine your own parameter for this section to minimize unwanted errors in your analysis.

`input_forward.fq input_reverse.fq`

For paired-end sequence, you have to input your forward and reverse sequence input.fastq files.

`output_forward_paired.fq output_forward_unpaired.fq`
`output_reverse_paired.fq output_reverse_unpaired.fq`

After your two forward and reverse input.fastq files have been trimmed, they will be stored into 4 different temporary output.fastq files. Paired.fastq files will be analyzed in the further process whereas unpaired.fastq files will not.

The above command line will perform the following actions:

Remove sequence adapters based on the sequenced stored in the TruSeq3-PE.fa file (ILLUMINACLIP:TruSeq3-PE.fa:2:30:10)

Remove leading low quality or N bases (below quality 3) (LEADING:3)

Remove trailing low quality or N bases (below quality 3) (TRAILING:3)

Scan the read with a 4-base wide sliding window, cutting when the average quality per base drops below 15 (SLIDINGWINDOW:4:15)

Discard reads less than 36 bases long (MINLEN:36)

Single End:

```
java -jar trimmomatic-0.30.jar SE -phred33 input.fq output.fq
ILLUMINACLIP:TruSeq3-SE:2:30:10 LEADING:3 TRAILING:3
SLIDINGWINDOW:4:15 MINLEN:36
```

Optional

For single end reads, an alternative, python based software tool called cutadapt can also be used. (Source code: https://pypi.python.org/pypi/cutadapt) (docuentation: http://cutadapt.readthedocs.org/en/latest/guide.html), the basic command-line for cutadapt is:

```
cutadapt -q 10 -- a AACCGGTT -- o output.fastq input.fastq
```

Cutadapt uses the same quality trimming algorithm as BWA. In this example command the cutoff score is a phred value of 10, and the trimmed adapter sequence is AACCGGTT. Cutadapt can be used to trim adapters without trimming low quality sequences, or trim low quality sequences without trimming adapters simply by omitting the adapters.

Preparing to align:

Software needed:

The various files and scripts included in the latest version of GMAP/GSNAP

(http://researchpub. gene.com/gmap/)

Reference genome needed:

Your reference genome as a FASTA file.

Annotation file needed:

A GTF file describing the annotated genes in your reference genome.

#Note: A GTF file is _similar to_ a GFF file, but there are crucial differences. If you can't find a GTF file, you will have to write your own script to turn a GFF file into a GTF file.

#Note: One common difference between GTF and GFF formats is how chromosomes are labeled between your FASTA and GTF files. The same chromosome might be called "1" or "chr1" or "Chr1" or "Chr01" in the GTF/GFF file. (Optionally) a list of known SNPs between the genotype used to generate this RNA-seq data and the genotype used to generate the reference genome.

Building the genomic index:

After installing the binaries from GMAP/GSNAP somewhere in your path, you can build the genomic index you'll be aligning reads against using gmap_build. Example command:

```
gmap_build -D ./ -d maizeV3  Zmays_284_AGPv3.hardmasked.fasta
```

This is telling gmap to build the index ("maizeV3") using the sequences from the file "Zmays_284_AGPv3.hardmasked.fasta" stored in the current directory, and saving that index in the same current directory.

#Note: it is important to conduct this step on a computer with enough RAM or bad things will happen.

Aligning Reads to the Reference Genome:

This is the single most computationally intensive step and can take anywhere from minutes to days depending on read length (longer takes longer), genome size (bigger takes longer), number of processors (fewer takes longer), and single end vs paired end (paired end takes way WAY WAY WAY longer). By the way, you have to manage time well when your computer is processing the step, sometimes it will slow down your computer when you want to open another software at the same time.

Example command:

```
gsnap -D /home/linux/Documents/reference_genome/maize/maize/ -d
maizeV3 --nthreads=4 -B 3 -N 1 -n 2 -Q --nofails --format=sam
temp.fastq > results.sam
```

There's a lot of stuff here, so let's walk through it from left to right.

```
-D /home/linux/Documents/reference_genome/maize/maize/
```

This is telling gsnap which directory I store my genomic index files in.

```
-d maizeV3
```

This is telling gsnap that I want to align against an index called maizeV3.

`--nthreads=4`

This particular command came from currently computer I am using. Never use more threads than your computer and processing cores.

#Note: there are two dashes preceding this command instead of just one.

`-B 3`

This sets how much information GSNAP will try to load into RAM (as opposed to loading from your hard drive as it is needed). Possible settings range from 0-5 with five loading the most data into RAM (and being the fastest) and 0 loading the least. This lets you tune GSNAP to run faster (on high memory computers) or run at all (on computers with less ram).

If you don't enter any settings at all, GSNAP defaults to 2. If your computer doesn't have enough RAM to successfully run your alignment at _at least_ a setting of -B 2, it's best to find another computer, because aligning at slower settings is really slow.

`-N 1`

This parameter means looking for novel splicing. 0 means do not look for novel splicing. This is based on your own preference.

`-n 2`

Only report reads with two or fewer “best” alignments in the genome. It's quite ok to increase this setting, but be prepared for the side of your output file to grow substantially larger if you do (since GSNAP has to report a separate alignment for each location a read maps to, a small number of reads which align at many locations add up quickly).

`-Q`

If a read aligns equally well to more than two locations – or however many you specified with –n above -- don't report it at all (the alternative is to report two of the 3-infinite equally good positions at random).

`--nofails`

Don't write out information on reads which fail to align at all. This saves space by making the resulting alignment file smaller.

`--format=sam`

Report alignments in the standardized SAM format instead of gsnap's native alignment format.

`temp.fastq`

The name of the file containing the sequencing reads to be aligned against the genome.

`results.sam`

The name of the file to store alignment data in.

Converting alignment data to a useful format:

Downloaded and installed the samtools package can be downloaded from the following link: http://samtools.sourceforge.net/

The plus side of the SAM (Sequence alignment map) format is that it is basically human readable – if you're one of the rare humans who has spent a long enough time looking at it to understand what you're seeing. The explanation of SAM file can be found by googling " Sequence Alignment/Map Format Specification." The downside of SAM is that the files are huge (sometimes bigger than the raw reads file you started with) and it's not easy for a computer to find just the parts of the file it is interested in.

So the next step is to convert SAM to sorted and (optionally) indexed BAM. BAM is the binary version of SAM, not at all human readable, but much more compact and much easier/faster for computer programs to access.

Example commands (these shouldn't need much explaining):

```
samtools view -bS results.sam > results.bam
samtools sort results.bam results-sorted
```

(optional)

```
samtools index results-sorted.bam
```

At the end of this process you have a sorted binary alignment file (results-sorted.bam) and – if you indexed it – an index file that tells a computer wherein that file to look for the reads that map to a specific part of the genome called "results-sorted.bam.bai"

Downstream analyses:

There are a bunch of different analyses which can be run on aligned sequence data depending on where the RNA or DNA you started with came from and what you are interested in. These analyses will require you to write your own scripts to extract and analyze data. If you use python I highly recommend the pysam library for extracting data from .bam files (http://code.google.com/p/pysam/). The only analysis which is standardized enough to describe here is quantifying gene expression from RNA-seq.

Quantifying Gene Expression:

#Note: This analysis is ONLY for measuring gene expression using RNA-seq data. Using it for any other purpose will lead to terribly misleading and confusing results. So just don't don't do it.

This RNA-seq analysis will be carried out using Cufflinks (available from here: http://cufflinks.cbcb.umd.edu/). So this software must be downloaded and installed.

#Note: Some versions of cufflinks have weird bugs where some expressed genes are reported as having 0 FPKM. To test a new version of cufflinks, carry out this procedure to measure gene expression and independently measure expression using a simple read counting software package such as HTCount. If you have many aligned reads for some genes with zero expression in the cufflinks report you probably have a buggy version of cufflinks. The current version of cufflinks (2.2.1) seems to be reasonably ok in my experience.

Cufflinks takes data from a sorted BAM file, and gene annotation data from a gtf file and measures gene expression in units of FPKM (frequency per kilobase of exon per million aligned reads). You can use the same .gtf file you used to generate your splicing file for GSNAP above.

Example command:

```
cufflinks -p 4 --library-type ff-firststrand --GTF
/home/linux/gsnap_indexes/MaizeV3.gtf results-sorted.bam
```

`-p 4`

This parameter means the number of thread is used during the analysis. Number of thread is varied for each computer.

`--library-type ff-firststrand`

For single-end library, it is important to set this parameter. If not, you might see that your zero fpkm value for fpkm_output file. However, paired-end library does not require to set this parameter because the default setting is (fr-unstranded).

For more information about the about the library type you may refer to the following links:

http://www.nature.com/nprot/journal/v7/n3/fig_tab/nprot.2012.016_T1.html

`--GTF /home/linux/gsnap_indexes/MaizeV3.gtf`

This parameter means the pathway for cufflinks to access annotation file(gtf).

This will create a number of intermediate files, but the one we're interested in is called genes.fpkm_tracking. This image below is an example of how a genes.fpkm_tracking file looks like:

tracking_id	class_code	nearest_ref_id	gene_id	gene_short_name	tss_id	locus	length	coverage	FPKM	FPKM_conf_lo	FPKM_conf_hi	FPKM_status
GRMZM2G059865.v6a	-	-	GRMZM2G059865.v6a	GRMZM2G059865	-	1:4853-9652	-	-	26.7824	24.6502	28.9146	OK
GRMZM5G866996.v6a	-	-	GRMZM5G866996.v6a	GRMZM5G866996	-	1:46227-47746	-	-	0	0	0	OK
GRMZM5G833153.v6a	-	-	GRMZM5G833153.v6a	GRMZM5G833153	-	1:144956-145646	-	-	41.6017	37.1957	46.0078	OK
GRMZM5G809743.v6a	-	-	GRMZM5G809743.v6a	GRMZM5G809743	-	1:144360-144657	-	-	197.663	176.298	219.028	OK
AC177838.2_FG002.v6a	-	-	AC177838.2_FG002.v6a	AC177838.2_FG002	-	1:70594-71919	-	-	0.259558	0	0.627139	OK
GRMZM2G306216.v6a	-	-	GRMZM2G306216.v6a	GRMZM2G306216	-	1:146264-147500	-	-	0.965349	0	2.22733	OK
GRMZM2G059778.v6a	-	-	GRMZM2G059778.v6a	GRMZM2G059778	-	1:76119-76752	-	-	0.803714	0	1.73176	OK
GRMZM2G059745.v6a	-	-	GRMZM2G059745.v6a	GRMZM2G059745	-	1:92353-93541	-	-	11.2986	7.90279	14.6945	OK
GRMZM5G811273.v6a	-	-	GRMZM5G811273.v6a	GRMZM5G811273	-	1:62501-64014	-	-	7.77536	3.54623	12.0045	OK
AC177838.2_FG013.v6a	-	-	AC177838.2_FG013.v6a	AC177838.2_FG013	-	1:154433-154905	-	-	0	0	0	OK
GRMZM2G093399.v6a	-	-	GRMZM2G093399.v6a	GRMZM2G093399	-	1:136306-138929	-	-	0.299039	0	0.708984	OK
AC177838.2_FG009.v6a	-	-	AC177838.2_FG009.v6a	AC177838.2_FG009	-	1:136309-136555	-	-	0	0	0	OK
GRMZM2G093344.v6a	-	-	GRMZM2G093344.v6a	GRMZM2G093344	-	1:109935-111769	-	-	0.646609	0.246057	1.04716	OK
GRMZM2G394757.v6a	-	-	GRMZM2G394757.v6a	GRMZM2G394757	-	1:110764-111465	-	-	3.12207	1.77715	4.46698	OK
GRMZM2G059856.v6a	-	-	GRMZM2G059856.v6a	GRMZM2G059856	-	1:9857-10388	-	-	0.000270862	0	0.340417	OK
GRMZM5G888250.v6a	-	-	GRMZM5G888250.v6a	GRMZM5G888250	-	1:9881-10387	-	-	2.57684	1.1417	4.01198	OK
AC177838.2_FG012.v6a	-	-	AC177838.2_FG012.v6a	AC177838.2_FG012	-	1:150549-150900	-	-	0.187148	0	0.935739	OK
AC177838.2_FG017.v6a	-	-	AC177838.2_FG017.v6a	AC177838.2_FG017	-	1:167329-167866	-	-	0	0	0	OK
GRMZM2G542340.v6a	-	-	GRMZM2G542340.v6a	GRMZM2G542340	-	1:215043-215371	-	-	0	0	0	OK
GRMZM5G837673.v6a	-	-	GRMZM5G837673.v6a	GRMZM5G837673	-	1:240524-240838	-	-	0	0	0	OK

Parsing the *.fpkm_tracking file should be pretty self.explanatory. If you are not comfortable writing parsing scripts, you can even open it in open office by selecting tabs as the delimiter between different columns. #Note: be sure to rename this file or it will be overwritten by the next set of results when you run cufflinks again in the same directory

Citations for Software Used in this Protocol:

Trimmomatic: Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: A flexible trimmer for Illumina Sequence Data. Bioinformatics, btu170.

GSNAP: Wu, Thomas D., and Serban Nacu. “Fast and SNP-tolerant Detection of Complex Variants and Splicing in Short Reads.” Bioinformatics 26, no. 7: 873 –881.

SAMTOOLS: Li, Heng, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. “The Sequence Alignment/Map Format and SAMtools.” Bioinformatics 25, no. 16 : 2078–2079.

CUFFLINKS: Trapnell, Cole, Brian A Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. “Transcript Assembly and Quantification by RNA-Seq Reveals Unannotated Transcripts and Isoform Switching During Cell Differentiation.” Nat Biotech 28, no. 5 : 511–515.