# WORKFLOW

Download all software to computer 🡪 download .fastq.gz 🡪 unzip 🡪 trim & quality filter 🡪 assemble transcriptome with or without reference genome 🡪 … 🡪 data analysis!

Useful tools and tricks (more in SNP pipeline document):

* ctrl u saves what you’ve typed, ctrl y pastes it
* $ find /some/path –name somefile
* $ ls -1 | wc -l counts the number of files, including directories. If you just want to count files of a certain type, change the ls command to look for \*.txt or whatever you want
* replacement using sed: sed 's/old/new/g' input.txt > output.txt
* Check how much space you are taking up on a drive: $ du -hs /directory/

Procedure:

* 1. Assess quality of sequences using FastQC. This will produce an html file showing summary statistics about the quality, & you can see at what point in the read the quality drops off into the red. Look through each read of each sample and decide where you are going to trim each read; record this in a file (how many reads, what you keep, what you trim). For some programs, reads have to be the same length for all samples, but for this pipeline they can be different lengths.
* 2. Even if the sequencing facility clipped adapters, if your target DNA is very short (like many transcripts), there can be artifacts caused by reading through from one side of the read through the other index sequence, so you need to search for and remove all index sequences and their reverse complements, as well as Stubby adapter and transposase sequences. We use Trimmomatic to remove common adapter sequences and/or specific sequences. If I am doing *de novo* assembly, I run Trimmomatic from within Trinity so that I don’t lose the unpaired reads. When aligning to a reference, I run Trimmomatic or TrimGalore and then align.
  + In Trimmomatic, the steps will occur in the order they are typed on the command line, so it is important to identify and trim adapter sequences first so they can be correctly identified.
  + To look for standard Nextera adapters in paired end mode: for i in \*1.fastq.gz; do java -jar /path/to/Trimmomatic.jar PE $i ${i%1.fastq}2.fastq -baseout ${i%1.fastq} ILLUMINACLIP:adaptersequences.fa:2:30:10 LEADING:5 TRAILING:5 MINLEN:35 SLIDINGWINDOW:5:15; done where params after ILLUMINACLIP are filename of adapter sequences, # seed mismatches, PalindromeClipThreshold, SimpleClipThreshold. Optional ILLUMINACLIP parameters are Minimum Adapter Length (default 8; can go as low as 1 b/c false positive rate is so low) and Keep Both Reads (add :true). It is worth trying this both ways (keeping and not keeping both reads).
  + Trimmomatic trims after bases exceed a minimum quality threshold; using SLIDINGWINDOW will allow it to keep the read in the case of 1-2 low-quality bases. Params are window size, phred quality threshold.
  + MAXINFO balances optimal base quality, read length, and error sensitivity. Suggested target read length is 40 bp for successful alignment; strictness = (range 0 - 1). A low value (<0.2) favors longer reads, whereas a higher value (>0.8) favors correct reads.
  + Your adapter file should be a fasta file with the sequences of all adapter **pairs** with names the program can identify as part of a pair (prefix/1 and prefix/2). You can also use a fasta file with any sequences to remove, e.g., transposase & stubby seqs
* 3. Either align reads to reference using bwa now (after trimming/quality filtering), or use Trinity to generate a single *de novo* transcriptome assembly by combining all reads across all samples as inputs. Then, reads are separately aligned back to the single Trinity assembly for downstream analyses. Trinity comprises 3 programs: Inchworm, Chrysalis, and Butterfly. They are internally integrated so you don’t have to run them separately, but you can.
  + Assemble RNA-Seq data in a de novo assembly with trinity\_w\_trimmomatic.sh:

/path/miniconda3/etc/profile.d/conda.sh

conda activate trinity-2.11.0

/trinity-2.11.0/bin/Trinity --seqType fq --max\_memory 10G --samples\_file /path/samples.txt --min\_contig\_length 150 --output /path/allbirds\_trinity --full\_cleanup where --samples\_file z is a tab-delimited text file with columns condition/replicate/leftread/rightread

* 4. After Trinity.fasta alignment is created, you need to map your reads to that alignment (or you map to the reference genome). Either bwa or bowtie will work.
  + To do so, first index the alignment using $ /path/bowtie2-build /path/allbirds.fasta /path/allbirds\_assembly which will create files with the assembly prefix
  + After creating the index, align reads and generate alignment statistics with $ for i in \*1.fastq; do bowtie2 -q --no-unal -k 20 -x /path/allbirds\_assembly -1 $i -2 ${i%1.fastq}2.fastq 2>${i%1.fastq}align\_stats.txt | samtools view -@10 -Sb -o ${i%1.fastq}bowtiealn.bam; done
* 5. Assess the number of full-length coding transcripts by blasting against swissprot and/or nr (the nr database is huge, so the fasta file will need to be split into multiple pieces to make it run faster – e.g., it took 72 hrs on our supercomputer workq to get the first <400 hits)
  + Create a table of counts
* 6. The next step is to estimate the abundance of each transcript (standardized to mean abundance of that individual's transcripts). The Trinity website (<http://trinityrnaseq.github.io/analysis/abundance_estimation.html>) has more details on this process, but I used the following steps:
  + First start with Trinity's perl script to estimate abundance: for i in \*qual.fastq; do perl /home/genetics/trinityrnaseq-2.0.2/util/align\_and\_estimate\_abundance.pl --transcripts ${i%L001\_R1\_001.qual.fastq}trinity\_out/Trinity.fasta --seqType fq --est\_method eXpress --aln\_method bowtie2 --single $i --prep\_reference --thread\_count 8 --output\_dir ${i%L001\_R1\_001.qual.fastq}trinity\_out/ --output\_prefix ${i%L001\_R1\_001.qual.fastq}abund; done;
  + Files generated will include: results.xprs = the primary eXpress output file, containing expression values for transcripts. results.xprs.genes = a 'genes' expression output file, generated by the Trinity toolkit, based on the eXpress transcript results. These 2 files are formatted identically, but .genes is limited in data content to the length, counts, & FPKM expression values. The remainder of the fields are set to N/A. --output\_dir is an important flag, because the 'output prefix' flag refers only to the bam outputs, not the express outputs, so without this you'll just overwrite the eXpress results file each time through the loop.
  + I did the same with RSEM so I would have independent estimates of abundance: $ for i in \*qual.fastq; do perl /home/genetics/trinityrnaseq-2.0.2/util/align\_and\_estimate\_abundance.pl --transcripts ${i%L001\_R1\_001.qual.fastq}trinity\_out/Trinity.fasta --seqType fq --est\_method RSEM --aln\_method bowtie --single $i --thread\_count 8 --prep\_reference --output\_prefix ${i%L001\_R1\_001.qual.fastq}RSEMabund; done
  + Note: both of these will lead to an error that bowtie, bowtie2, express and others are not in the path. I didn't feel comfortable editing the /etc/profile file on the server, so I first had to locate the directory containing those utilities (sudo find /home/genetics/ -name 'rsem-calculate-expression'), and then add that directory to the path on the command line (export PATH=/home/genetics/trinityrnaseq-2.0.2/trinity-plugins/:$PATH). Unfortunately there were errors in running 'make' for RSEM so this is still not working.
  + Now rename your files so they have the names in each line, for future concatenation: for i in \*qual.fastq; do cat ${i%L001\_R1\_001.qual.fastq}trinity\_out/results.xprs | awk -F '\t' -v x=${i%L001\_R1\_001.qual.fastq} '{print x "\t" $0}' > ${i%L001\_R1\_001.qual.fastq}express.name; done --To rename a column header with an \_x in each column and get rid of the rest of the file: cat r43\_S20\_results.cut.txt | head -n 1 | awk -F '\t' -v x=r43 '{print $1 "\t" $2 "\_x" "\t" $3 "\_x" "\t" $4 "\_x"}' > r43\_results.xprs
  + This output will be more than we need, so we'll cut to a subset with FPKM (CI), sample & contig IDs, etc. $ for i in \*express.name; do cat $i | cut -f1,2,3,5,9,12,13,14,15,16 > "${i%name}cut.txt"; done
  + Now, I don't want the names in each line but in the header. I haven't figured out how to modify the header and keep the rest of the file, but I can create header files like this: for i in \*cut.txt; do cat $i | head -n 1 | awk -F '\t' -v x=${i%\_results.cut.txt} '{print $1 "\t" $2 "\_" x "\t" $3 "\_" x "\t" $4 "\_" x}' > $i.head; done
  + After creating the header, sort the files by transcript ID and then you can concatenate the header and then paste all files together. I accidentally did an extra step and concatenated the header before sorting, but to sort: for i in \*results.txt; do cat $i | tail -n +2 $i | sort -d -s -k1,1 > ${i%txt}sort.txt; done
  + Now to concatenate the header: for i in \*sort.txt; do cat ${i%sort.txt}cut.txt.head $i > ${i%sort.txt}sorted.head; done
  + Finally, you can combine them into one file with $ paste \*sorted.head > allsamples\_results.txt