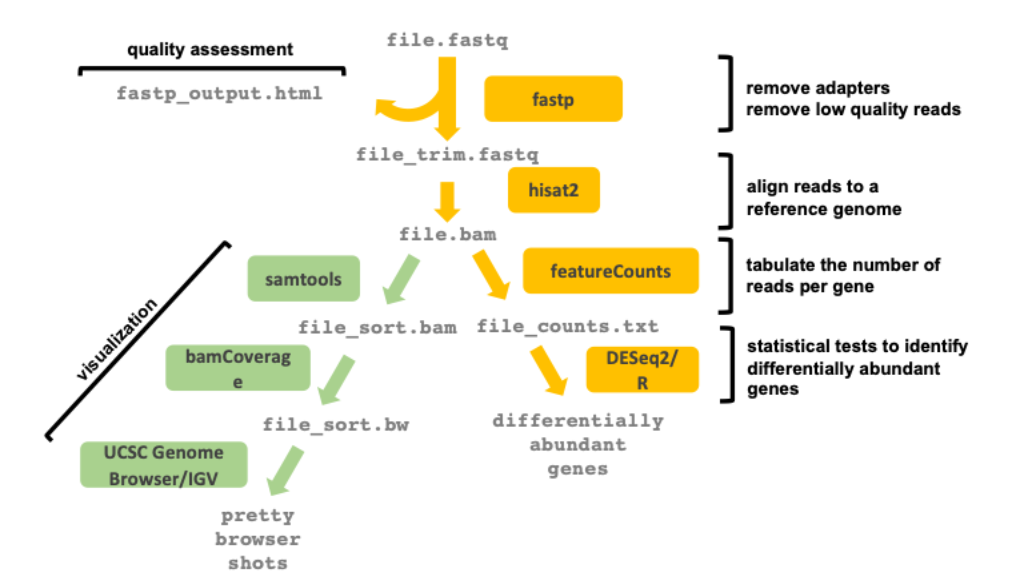
## Pipeline overview:

****

## Installing software on Summit

1. Open Jupyterhub: <https://jupyter2.rc.colorado.edu/>
   1. Username: CSU eid email address
   2. Password: CSU password, a **comma**, and the word **push**.
2. Click “Launch server” → “Summit interactive (12 hr)” → “Spawn”
3. Open a terminal
4. Initiate virtual conda environment:

$ **hostname** *# check where you are*

$ **ssh** scompile *# switch to a compile node*

$ **hostname** *# check where you are (you should see a hostname of* ***shas136*** *or* ***shas137****)*

1. To test whether your conda environment is specified, ensure you are in your home directory and read what is written in the document **.condarc** or whether that document even exists:

$ pwd *# you should be /home/eID@colostate.edu*

$ **ls** -alh *# you may see a file called .condarc*

$ **more** .condarc

pkgs\_dirs:

- /projects/.colostate.edu/<yournamehere>/.conda\_pkgs

envs\_dirs:

- /projects/.colostate.edu/<yournamehere>/software/anaconda/envs

*# If you see those four lines of code, you're all set up! You may also see additional lines, that is fine.*

* 1. If you saw 4 lines of code within the file **.condarc** that specify where your package directories should be stored and where your environment directories should be stored → proceed to step 6.
  2. If you didn’t see 4 lines of code OR don’t have a file called **.condarc**:

*# ONLY IF YOU DIDN'T SEE THE FOUR LINES OF CODE ABOVE*

$ **nano** .condarc

*# then copy and paste the following in. Substitute <yournamehere> with your eID name & remove <>.*

pkgs\_dirs:

- /projects/.colostate.edu/<yournamehere>/.conda\_pkgs

envs\_dirs:

- /projects/.colostate.edu/<yournamehere>/software/anaconda/envs

*# Exit out of nano using*

*# CTRL + S*

*# Type Y*

*# Return*

$ **more** .condarc *# do this to check your .condarc file*

1. Activate conda:

$ source /curc/sw/anaconda3/latest

(base) [eid@colostate.edu@shas0137 ~]$

1. Build a custom virtual environment for the project.

$ hostname. *# Ensure first that you're on an scompile node. It should say shas136 or shas137*

$ conda create -n <projectEnvironmentName> python==3.8

$ conda **env** list

1. Navigate into your new environment:

$ conda activate <projectEnvironmentName>

$ conda **env** list *# This shows you which environments are available and selected*

$ conda list *# This shows the software currently installed in your active environment*

1. Install software packages: fastp, bwa, hisat2, bedtools, and samtools:

$ conda config --add channels conda-forge

*# If you get a warning, that's ok*

*$ conda* ***install*** *-c bioconda sra-tools fastp bwa hisat2 bedtools samtools*

1. For every future login to Summit, do the following:

$ **ssh** scompile

(base) [eid@colostate.edu@shas0136 ~]$

* 1. If you don’t see the “base” tag, initiate conda:

*$* source /curc/sw/anaconda3/latest

$ conda activate <projectEnvironmentName>

1. To install anything new at this point into this project environment, just run the following code:

$ conda config --add channels conda-forge *# may not be necessary*

$ conda **install** -c bioconda <software\_name\_here>

* 1. To find packages: <https://anaconda.org/bioconda/>
  2. If you run into problems using conda or installing software: [rc-help@colorado.edu](mailto:rc-help@colorado.edu)

## Project initiation on Summit

1. Log into Summit and start a terminal.
2. Move to scompile node and activate conda environment:

$ **ssh** scompile

$ source /curc/sw/anaconda3/latest

$ conda activate <projectEnvironmentName>

1. Create a directory for the RNA-seq project within your scratch directory on SUMMIT. **NOTE: Scratch directories provide 10 TB to work with, but files are NOT backed up and are deleted 90 days after their initial creation.**

$ cd /scratch/summit/<eID>@colostate.edu

$ mkdir <projectDirectoryName>

$ cd <projectDirectoryName>

$ mkdir 01\_input 02\_scripts 03\_output

## Data acquisition

*Optional – only if using .fastq files from another publication and not your own raw RNA-seq data.* ***If using your own data, upload .fastq files directly into the appropriate Summit directory using the navigation window on the sidebar.***

1. Downloading datasets from NCBI GEO:
   1. Find GEO entry (accession number usually provided in the publication)
   2. Click on “SRA Run Selector” at the bottom of the page

$ **mkdir** test\_download *# Create a new directory to save the file downloads*

$ cd test\_download *# Move into that directory*

* 1. To download 1 .fastq file:

$ fasterq-dump --split-files --progress <SRR number of .fastq file> *#took me 5 minutes for a 1 GB file.*

$ **vdb-validate** <SRR number> # automates the process of checking md5sums

$ **ls** # should see a \_1.fastq and \_2.fastq file for each SRR number in the dataset.

* 1. To download all .fastq files:
     1. Click “Accession list” → download somewhere on your computer → open file
     2. Copy and paste the contents of that file into a new file on Summit called SRR\_Acc\_List\_<projectName>.txt (you can use nano to do this)
        1. Should be a list of SRR numbers
     3. Make a script in the same directory that will automate the process of downloading every sample in the SRR list using fasterq-dump

*#!/usr/bin/env bash*

*#SBATCH --nodes=1*

*#SBATCH --ntasks=1*

*#SBATCH --time=06:00:00*

*#SBATCH --partition=shas*

*#SBATCH --output=log-download-%j.out*

*# Execute code with: $ sbatch automateSRA.sbatch <file\_listing\_SRR\_files.txt>*

*# loop over each SRR file and import each fastq file:*

while read line

do

echo -e $line

echo "fasterq-dump --split-files --progress **$line**"

time fasterq-dump --split-files --progress $line

echo "vdb-validate **$line**"

vdb-validate $line

done < $1

* + 1. Execute the program:

*$* sbatch automateSRA.sbatch SRR\_Acc\_List\_<projectName>.txt

* + 1. Create a metadata file in which the first two columns are paired fastq file names and the third column is a nickname for each sample (can create in Excel and then export as a .csv or .tsv). Continue to fill in the metadata file with relevant information about each sample. This metadata file will be important for use in future scripts to instruct a series of commands to loop over each field.

## Use fastp to remove adapters and low quality reads

*This step assesses the overall quality of the data, removes low Phred-scoring reads, removes adapter sequences that snuck into the reads, and removes polyA tails or PolyX tracks. PolyA tails won’t align to the genome. PolyX tracks sometimes arise erroneously through the sequencing process.*

*Fastp github repository:* [*https://github.com/OpenGene/fastp*](https://github.com/OpenGene/fastp)

*Fastp publication:* [*Chen S, et al., (2018) fastp: an ultra-fast all-in-one FASTQ preprocessor. Bioinformatics, Volume 34, Issue 17, 01 September 2018, Pages i884–i890*](https://academic.oup.com/bioinformatics/article/34/17/i884/5093234)

1. Log into Summit and start a terminal. Activate conda environment and navigate to the project directory.

$ **ssh** scompile

$ source /curc/sw/anaconda3/latest

$ conda activate <projectEnvironmentName>

$ cd /scratch/summit/[eID@colostate.edu](mailto:eID@colostate.edu)/<projectDirectory>/

1. Create a script within the script subdirectory for this project. Consider using the sample nickname from the metadata file for the outputFileName.

$ cd ../02\_scripts

$ pwd *# make sure you are in your scripts directory*

$ **touch** preProcess\_fastq.sh

*#!/usr/bin/bash*

*# execute fastp*

fastp -i ../01\_input/<inputFileName>\_1.fastq\

-I ../01\_input/<inputFileName>\_2.fastq\

-o ../03\_output/<outputFileName>\_trim\_1.fastq\

-O ../03\_output/<outputFileName>\_trim\_2.fastq\

--thread 1\

-h ../03\_output/<outputFileName>\_report.html\

-j ../03\_output/<outputFileName>\_report.json\

-g -x -p

1. To loop through multiple files: (Replace SRRs with file names as necessary)

$ **touch** preProcessWithLooping\_fastq.sh

*#!/usr/bin/bash*

*# execute fastp on two paired-end reads*

*# Initiate a bash array called SRRIDs with two elements*

SRRIDs="tester\_SRR5832182 tester\_SRR5832183"

*# Loop through the SSRID array and run fastp on each:*

for SRRID in $SRRIDs

do

echo -e $SSRID

fastp -i ../01\_input/*${SRRID}*\_1.fastq\

-I ../01\_input/*${SRRID}*\_2.fastq\

-o ../03\_output/*${SRRID}*\_trim\_1.fastq\

-O ../03\_output/*${SRRID}*\_trim\_2.fastq\

-h ../03\_output/*${SRRID}*\_report.html\

-j ../03\_output/*${SRRID}*\_report.json\

--detect\_adapter\_for\_pe\

--thread 1\

-g -x -p

done

Fastq options:

|  |  |  |
| --- | --- | --- |
| **option name** | **information after** | **description** |
| -i | ../01\_input/<inputName>.fastq | input file for one side of paired-end reads |
| -I | ../01\_input/<inputName>.fastq | input file for the other side of the paired end read |
| -o | ../03\_output/<outputName>\_trim\_1.fastq | trimmed output file #1 |
| -O | ../03\_output/<outputName>\_trim\_2.fastq | trimmed output file #2 |
| -h | ../03\_output/<outputName>\_report.html | create a report |
| -j | ../03\_output/<outputName>\_report.json | create another report |
| --detect\_adapter\_for\_pe |  | to switch the adapter removal into paired-end mode |
| --thread | 1 | Number of cores to use (parallel processing) |
| -g |  | this will remove poly-G sequences |
| -x |  | this will remove poly-X sequences |
| -p |  | this will perform over-represented sequence analysis |

## Building reference genome indices - STAR

STAR manual: <https://github.com/alexdobin/STAR/blob/master/doc/STARmanual.pdf>

1. Install necessary software:

$ **ssh** scompile

$ source /curc/sw/anaconda3/latest

$ conda activate <EnvironmentName>

$ conda **install** -c bioconda star

1. Navigate to your project directory and make a directory for the index build:

mkdir indicesBuild

1. Acquire the appropriate reference genome files from Ensembl. To do this, search Ensembl for the genome you want to use as your reference genome and navigate to the “Download DNA sequence” (FASTA) link. To download all the files on this page to Summit, replace the https:// with rsync:// and place /ensembl/ after the ensembl.org (as described by Ensembl [here](https://uswest.ensembl.org/info/data/ftp/rsync.html)). For example, for the ROS CFam 1.0 genome:

$ rsync -azvP rsync://ftp.ensembl.org/ensembl/pub/release-106/fasta/canis\_lupus\_familiaris/dna/ .

* 1. Don’t forget to add the period at the end to specify you want them downloaded in your current working directory.
  2. Repeat this process for obtaining the .gtf Ensembl files for your selected genome; e.g.:

$ rsync -azvP rsync://ftp.ensembl.org/ensembl/pub/release-106/gtf/canis\_lupus\_familiaris/

1. Try to stick with .gtf files; if you have a .gff file, here’s how to convert:
   1. Go to <https://github.com/gpertea/gffread> and download the software for gffread using their instructions.
   2. Run it with the following:

gffread <original gff3 filename> -T -o <output gtf file name>

gffread Arabidopsis\_thaliana.TAIR10.49.gff3 -T -o gffread\_Arabidopsis.gtf

(-T -o option will output GTF format instead of GFF3)

1. After the data transfer to Summit, delete any unnecessary genome files. It is generally encouraged to use the “primary assembly” files for building indices, so we can delete any masked (“sm”, “rm”) or toplevel files.
2. Check sums & gunzip all remaining .gz genome files, then concatenate the individual chromosome fasta files into one fasta file:

$ cat \*.fa > <concatenatedGenomeFilename>.fa

1. In your indicesBuild directory, create a script called STAR\_buildIndices.sbatch. Modify as necessary and submit as a job.

#!/bin/bash

# Building a STAR index file

#SBATCH --job-name=execute\_STAR-build

#SBATCH --nodes=1

#SBATCH --ntasks=8 # modify this number to reflect how many cores you want to use (up to 24)

#SBATCH --partition=shas

#SBATCH --qos=testing # modify this to reflect which queue you want to use. Options are 'normal' and 'testing'

#SBATCH --time=4:00:00 # modify this to reflect how long to let the job go. This indicates 4 hours.

#SBATCH --output=log\_STAR-build\_%J.txt

#SBATCH --mail-type=END,FAIL,TIME\_LIMIT

#SBATCH --mail-user=edlarsen@colostate.edu

# make output directory

mkdir STAR\_<genomeName>\_index

# Run STAR

STAR \

--runThreadN 8 \

--runMode genomeGenerate \

--genomeDir ./STAR\_<genomeName>\_index \

--genomeFastaFiles ./<concatenatedGenomeFilename>.fa \

--sjdbGTFfile ./<gtffile>.gtf

## Align sequences to the reference genome (Feeding trimmed fastq files to STAR)

1. For paired-end reads, STAR requires that we input a comma separated list of read1 files, followed by a space, followed by a comma separated list of read2 files. To generate these comma separated lists of file names, navigate to the 02\_output folder where our trimmed fastq files are stored, and run this code:

$ ls \*\_trim\_1.fq | tr '\n' ',' > read1ListTest.txt

$ ls \*\_trim\_2.fq | tr '\n' ',' > read2List.txt

1. Delete the trailing comma after the last filename and paste ../03\_output/ before each filename. (Remember, these are the trimmed files generated by FASTP, so they should be located in your 03\_output directory.)
2. Navigate to your 02\_scripts directory and create a script *STAR\_alignReads.sbatch*.
3. Copy & modify the following script.
   1. “genomeDir” specifies the path to the genome directory where genome indices (from the previous step) were generated.
   2. “readFilesIn” indicates the name(s) (with path) of the files containing the sequences to be mapped (i.e., our trimmed RNA-seq fastq files). ~~Multiple files can be mapped in one run by separating with a comma.~~ **~~For paired end reads, use comma separated list for read1, followed by space, followed by comma separated list for read2.~~**
   3. “outSAMtype BAM SortedByCoordinate” outputs a compressed, coordinate-sorted bam file instead of a larger unsorted sam file. Can be used for visualization with IGV and directly indexed with samtools.
   4. “outReadsUnmapped Fastx” creates a fastq file of reads that STAR failed to map.
   5. “quantMode GeneCounts” uses STAR to generate read counts. We’ll still do featureCounts with our output bam file, but I think it’s good to just take advantage of this tool when running STAR anyway, especially if we ever decide to change protocols to use one read counter over the other. This option does require that we also provide a GTF file.

#!/bin/bash

# Running alignment job with STAR

#SBATCH --job-name=execute\_STAR-mapping

#SBATCH --nodes=1

#SBATCH --ntasks=8 # modify this number to reflect how many cores you want to use (up to 24)

#SBATCH --partition=shas

#SBATCH --qos=normal

#SBATCH --time=12:00:00

#SBATCH --output=log\_STAR-mapping\_%J.txt

#SBATCH --mail-type=END,FAIL,TIME\_LIMIT

#SBATCH --mail-user=[edlarsen@colostate.edu](mailto:edlarsen@colostate.edu)

# Create an array of sample IDs for looping.

SampleIDs="CI100641 CI100850 CI100993 … etc."

# Run STAR, looping through each sample. Adjust pathways as needed.

for SampleID in $SampleIDs

do

STAR \

--runThreadN 8 \

--genomeDir ../indicesBuild/<directory of STAR generated genome index files> \

--quantMode GeneCounts \

--outSAMtype BAM SortedByCoordinate \

--outReadsUnmapped Fastx \

--sjdbGTFfile ../indicesBuild/<path to reference genome GTF file> \

--readFilesIn ../03\_output/${SampleID}\_trim1.fq> ../03\_output/${SampleID}\_trim2.fq

done

## Index BAM files with samtools

1. This indexes a coordinate-sorted BAM file for fast random access. Use the coordinate-sorted BAM files generated by STAR. Output will be indexed BAI files. BAI files act as an external “table of contents” for the BAM file and allows programs to jump directly to specific parts of the BAM file without reading through all of the sequences. This kind of indexing is required for genome viewers like IGV.
2. Navigate to the output directory that contains the BAM file generated by STAR and execute this line of code:

$ samtools index <bamfilename>.bam

## Quantify/tabulate reads (featureCounts)

*FeatureCounts online manual:* <http://bioinf.wehi.edu.au/featureCounts/>

*FeatureCounts publication:* Liao et al., Bioinformatics, Volume 30, Issue 7, 1 April 2014, Pages 923–930.

*The input files for featureCounts are .sam/.bam files (alignment files) and a .gtf/.gff annotation file for your reference genome. BAM files need to be sorted for counting (which we accomplish in our STAR script). The output of featureCounts is a counts.txt file containing count data for the entire experiment, and a summary file for the entire experiment. (Note that this uses BAM files, NOT our indexed BAI files!)*

1. Install necessary software:

$ **ssh** scompile

$ source /curc/sw/anaconda3/latest

$ conda activate <EnvironmentName>

$ conda **install** -c bioconda subread

$ conda **install** -c bioconda deeptools

1. Navigate to 02\_scripts directory and start a new script called tabulateCounts.sbatch

#!/usr/bin/env bash

#SBATCH --nodes=1

#SBATCH --ntasks=4

#SBATCH --time=04:00:00

#SBATCH --partition=shas

#SBATCH --output=tabulate-%j.out

#SBATCH --mail-type=END,FAIL,TIME\_LIMIT

#SBATCH --mail-user=edlarsen@colostate.edu

featureCounts -p -T 4 -a ../01\_input/<inputFileName>.gtf -o ../03\_output/<outputFileName>\_feature\_counts.txt ../03\_output/<directory with STAR alignment output>/<bamfilename>.bam

1. Execute the code:

$ sbatch tabulateCounts.sbatch

$ squeue -u $USER *# to check it*

$ **more** log\_features\_%j.txt *# where %j is the job ID #*