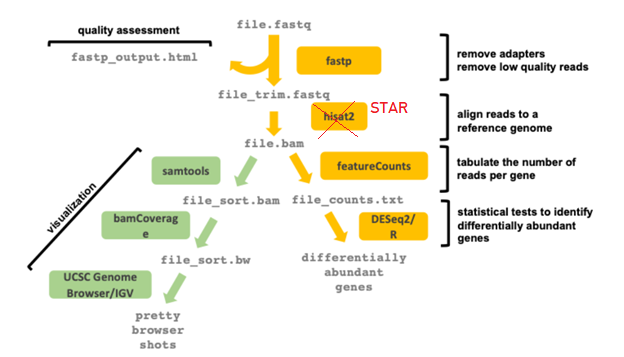
## Pipeline overview:

****

## Notes about this guide

*Last updated 1/17/24 by Eileen Owens*

*This guide was adapted from* [*Erin Nishimura’s DSCI512 course*](https://rna.colostate.edu/2021/doku.php?id=rnaseq_home) *and vignettes from the various software packages utilized in this guide. Links to current documentation for these packages at the time of this guide’s creation are provided throughout, and it is recommended that you consult these to customize scripts appropriately to your project.*

*Highlighted text is used throughout this guide to indicate sections of code that are meant to be replaced with your actual input (CSU eid, file names, directory paths, etc.).*

*Additional recommended resources:*

* [*CURC Research Computing Summer Camp Github*](https://github.com/ResearchComputing/Summer_Camp_2023)*: access PowerPoint lectures related to working on Alpine*
* *CURC support:* [*rc-help@colorado.edu*](mailto:rc-help@colorado.edu)
* *Online introductory courses to R programming:* [*STAT158*](https://csu-r.github.io/Module1/)*,* [*ERHS 535*](https://geanders.github.io/RProgrammingForResearch/course-information.html)*,* [*R for Data Science*](https://r4ds.had.co.nz/)
* *Online walkthroughs of RNA-seq pipelines with example scripts:* [*Sydney Informatics Hub*](https://sydney-informatics-hub.github.io/training-RNAseq/)*,* [*University of Manchester*](https://ycl6.gitbook.io/guide-to-rna-seq-analysis/)

## Accessing Alpine

1. Obtain a CURC account: [https://it.colostate.edu/research-computing-and-cyberinfrastructure/compute/get-started-with-alpine/#](https://it.colostate.edu/research-computing-and-cyberinfrastructure/compute/get-started-with-alpine/)
2. (Optional) Access Alpine from a local terminal:

$ ssh eid@colostate.edu@login.rc.colorado.edu

* 1. You will be prompted for a password. This is your CSU password, a **comma**, and the word **push.** Accept the DUO authentication request.

1. (Recommended) Access Alpine from OnDemand web-based GUI:
   1. Log in to OnDemand with CURC account: <https://ondemand-rmacc.rc.colorado.edu/>
   2. To open a terminal: Clusters 🡪 >\_Alpine Shell
   3. You will be prompted to re-enter your password. This is your CSU password, a **comma**, and the word **push.** Accept the DUO authentication request.
   4. You should see a welcome message displayed in the terminal after you successfully log in.
2. Switch from a login to a compile node.
   1. When you first log in to your Alpine shell, you will notice a prompt that looks like this:

[eid@colostate.edu@**login-ci1** ~]$

* 1. The “login” in your prompt indicates you are on a login node. Alpine offers different node types meant for certain tasks, which are all explained [here](https://curc.readthedocs.io/en/latest/compute/node-types.html). Login nodes are not intended for computational tasks, so it is generally a good idea to switch to a compile node after logging in. To do this, simply type this command on the command line:

$ acompile *# Default time allocated to you is 1 hr; modify with flag “--time=HH:MM:SS” as necessary to avoid interruptions. Max is 24 hours.*

* 1. Now your prompt should look something like this. This indicates that you are now working on a compute node.

[eid@colostate.edu@**c3cpu-a7-u11-3** ~]$

## Overview of basic Unix commands and symbols

This pipeline requires working in a Unix terminal on CU Boulder’s Alpine supercomputer, which requires some basic familiarity with navigating file systems (or “directories”) and executing scripts from the command line. Here is a helpful list of command commands, many of which you’ll see used throughout this guide:

|  |  |  |  |
| --- | --- | --- | --- |
| Command | Function | Common options | Example/Notes |
| pwd | Display current working directory |  | pwd |
| cd | Change directory | .. (up one level) | cd .. cd *DirectoryName* |
| grep | Search for pattern in file | -r (grep directory), -c (return only count), -o (output matches) | grep *pattern files* OR grep -r *directory* |
| gunzip | Uncompress a .gz file |  | gunzip *file.fq.gz* |
| ls | List files in current directory | -l (list file permissions) | ls |
| mkdir | Make new directory |  | mkdir *DirectoryName* |
| cat | Display file contents |  | cat > *file* (place input into *file*, can be used to combine files) |
| head | Output the first 10 lines of a file | -n (custom # of lines to display) | head *file* |
| tail | Output the last 10 lines of a file | -n (custom # of lines to display) | tail *file* |
| touch | Create a file if it doesn’t exist |  | touch *file* |
| cp | Copy file | -R (recursive) -X (ignore attributes) | cp -RX *directory1 directory2* |
| mv | Move or rename file |  | mv *file1 file2* |
| rm | Remove (delete) file | -r (remove directory), -f (force remove file) | rm *file* |
| echo | Write arguments to standard output | -n (don’t add new line after input) | echo Hello, world! |
| | | Pipe – output from one command to another |  | command1 | command2 |
| ./ | Execute program |  | ./*program.pl* |
| ssh | Secure shell – use to access remote computers and servers |  | ssh *username@server.com* |
| scp | Copy something from one machine to another | -r (for copying directories), use . for path to working directory | scp *eid@colostate.edu@login.rc.colorado.edu:/path/to/remote/file /path/to/local/file* |
| rsync | A fast remote (and local) file-copying tool | -a (archive mode) -v (verbose) | rsync -av *eid@colostate.edu@login.rc.colorado.edu:/remote/path /local/path* |
| > | Redirect standard output |  | cat file1.txt file2.txt > file3.txt |
| ; | Separate commands on the same line |  |  |
| / | Separator in a path name |  |  |
| ~ | Your home directory |  |  |
| . | Present working directory |  |  |
| .. | Parent of present working directory |  |  |
| \* | Match one or more occurrences of any character |  | cat \*.txt > summary.txt *# Here, the wildcard is used to capture the output of all files with a .txt extension in a directory and create a new summary.txt file containing that output.* |
| chmod | Change file permissions | Owner, position 1; group, position 2; anyone, position 3. Read = 4, write = 2, execute = 1, none = 0.  1 = execute only; 2 = write only; 3 = write and execute; 4 = read only; 5 = read and execute; 6 = read and write; 7 = read, write, and execute | chmod 777 |
| #!/bin/bash | Directs shell to bash interpreter |  | You’ll see this line at the beginning of every bash script in this guide. |
| # | Note/comment |  | A note to the person reading the code that will be ignored by the computer when the code is run. Information following “#” is not executed. |
| squeue | View information about jobs located in the Slurm scheduling queue | -u (user) | squeue -u $USER |
| scancel | Cancel a job submitted to the Slurm scheduler |  | scancel *jobID* # Can obtain the job ID from squeue |

## Upload raw data to Alpine

*If you are using publicly available fastq files and not your own raw data, skip this section and proceed to the “Data Acquisition” section below instead.*

1. Create a directory for your RNA-seq project within your scratch directory on Alpine. **NOTE: Scratch directories provide 10 TB to work with, but files are NOT backed up and are deleted 90 days after their initial creation.**
   1. Change your working directory to your scratch directory. You can do this either by typing out your full CURC username,

$ cd /scratch/alpine/eid@colostate.edu

or by simply typing *$USER* in place of your full username.

$ cd /scratch/alpine/$USER

* 1. Then, make a project directory, navigate into it, and make 3 additional subdirectories.

$ mkdir projectDirectoryName *# make a project directory*

$ cd projectDirectoryName *# change your working directory to your new project directory*

$ mkdir 01\_input 02\_scripts 03\_output *# make 3 additional subdirectories*

1. Option 1: Transfer files using **Globus**, a web-based GUI for file transfer between a local machine and a remote directory (Alpine, in this case). *Good for dealing with large data files, like RNA-seq fastq files.*
   1. Access Globus: **app.globus.org**
      1. Choose Colorado State University as affiliated institution, then follow the prompts to log in with your eid.
   2. Download Globus Connect Personal onto the local computer where the raw data files are stored.
      1. To include specific directories (like network drives), open “Options” in the Globus Connect Personal application 🡪 “Access” 🡪 “+” 🡪 navigate to desired folder. *Make sure to only grant shareable, not writeable, permissions to RSTOR to avoid tampering with permanently stored raw data.*
   3. In the Globus web app, open the *File Manager* tab 🡪 “Transfer or Sync to”
      1. This should prompt you to select **two** collections; one on the left, one on the right. The order doesn’t matter; this protocol will assume the remote Alpine connection on the left, and the local computer on the right.
   4. Search “CU Boulder Research Computing” in the *Collection* bar on the left.
      1. Enter the path to your scratch directory in the *Path* bar:

/scratch/alpine/eid@colostate.edu/yourProjectDirectory/01\_input

* 1. On the right, enter the Collection name of your local computer (this is something you choose during installation of the Globus Connect Personal application).
     1. Navigate to the directory containing the raw data files to be uploaded.
  2. Select the files to transfer to Alpine, then click “Start” under the local collection on the right side to transfer them to Alpine directory.

1. Option 2: Transfer files using **FileZilla**, a free and open-source software program for file transfer that you download to your local computer and connect remotely to Alpine.
   1. Download FileZilla to the local computer where the raw data files are stored: <https://filezilla-project.org/>
   2. Open FileZilla. In the “Local site” panel on the left, navigate to the directory containing the raw files.
   3. Connect to Alpine as the remote site by entering the following into the bar at the top of the FileZilla application window:
      1. Host: **sftp://login.rc.colorado.edu**
      2. Username: [**eid@colostate.edu**](mailto:eid@colostate.edu)
      3. Password: your CSU password, a **comma**, and the word **push.**
      4. Leave the “Port” field blank. Click Quickconnect. Accept the DUO authentication request.
   4. In the “Remote site” panel to the right, enter **/scratch/alpine/eid@colostate.edu** in the address bar and then double-click through folders to navigate to your desired upload directory (*01\_input*).
   5. Select the files from the Local site panel, right click, and select “Upload.” Accept any additional DUO authentication requests.
2. Option 3: Transfer files using **Jupyter Notebook**, a user-friendly web-based GUI similar to Globus, but less ideal for really large files.
   1. Log in to OnDemand with CURC account: <https://ondemand-rmacc.rc.colorado.edu>
   2. Interactive Apps 🡪 Jupyter Session (Presets) 🡪 1 core, 12 hours 🡪 Check box for “use JupyterLab instead of Jupyter Notebook” 🡪 Launch 🡪 Wait for session to load, then click “Connect to Jupyter” 🡪 File🡪 Open from Path 🡪 type the path to your scratch directory (/scratch/alpine/eid@colostate.edu)
   3. Use the navigation window on the left to upload files directly from your local machine to the *01\_input* folder you created on your Alpine scratch directory.
3. Option 4: Transfer files using the terminal – *efficient but requires a higher level of comfort with Linux.*
   1. For small data transfers, a login node is fine. For large data transfers, use a data transfer node (data transfer nodes will require a VPN connection to CSU network).

$ ssh eid@colostate.edu@dtn.rc.colorado.edu

* 1. Copy local file to RC resources using a login node:

$ scp file1 eid@colostate.edu@login.rc.colorado.edu:/remote/path

* 1. Copy local file to RC resources using a DTN:

$ scp file1 eid@colostate.edu@dtn.rc.colorado.edu:/remote/path

* 1. To copy a file from RC resources to local path:

$ scp -r eid@colostate.edu@dtn.rc.colorado.edu:/remote/path /local/path *# Can just use a period for the local path if already in the desired directory.*

* 1. Scp is more secure and good for first-time file transfers. If files are then changed and we want to sync them again between remote and local directories, rsync is a more efficient option.

$ rsync -av eid@colostate.edu@login.rc.colorado.edu:/remote/path /local/path

1. Once files are transferred to your Alpine scratch directory, check md5sums. They should match the md5sums associated with your raw data files (for Novogene, this is provided in an MD5.txt file). If they do not match, it indicates the file was corrupted during transfer and should be re-uploaded to Alpine.

$ md5sum \*.fq.gz > Date\_md5sums.txt *# Run this line in the directory where raw files were uploaded. This creates a text file containing the md5sums for the uploaded files.*

## Data acquisition

*Optional – only if using publicly available fastq files and not your own raw RNA-seq data.* ***If using your own data, follow steps for “Upload raw data to Alpine” above and skip this section*.**

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=amilan

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

# Execute code with: $ sbatch automateSRA.sh SRR\_Acc\_List\_projectName.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.sh SRR\_Acc\_List\_projectName.txt

## Installing software on Alpine

1. Open an Alpine terminal (either locally or using OnDemand) and log in to CURC, then switch from the default login node to a compile node.

$ acompile *# Default time allocated to you is 1 hr; modify with flag “--time=HH:MM:SS” as necessary to avoid interruptions. Max is 24 hours.*

1. Initiate virtual conda environment.
   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/eid/.conda\_pkgs

envs\_dirs:

- /projects/.colostate.edu/eid/software/anaconda/envs

**# If you see these 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 3**.
  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.*

pkgs\_dirs:

- /projects/.colostate.edu/eid/.conda\_pkgs

envs\_dirs:

- /projects/.colostate.edu/eid/software/anaconda/envs

# Exit out of nano using

# CTRL + X

# Type Y

# Return

* + 1. Now check your .condarc file:

$ **more** .condarc

1. Activate conda:

$ module load anaconda  
  
*# Your output should look like this, with (base) in front of your command prompt:*

(base) [eid@colostate.edu@c3cpu-a2-u34-4 ~]$

1. Build a custom virtual environment for the project.

$ conda create -n projectEnvironmentName python==3.10

$ conda **env** list *# Make sure your new environment shows up in the list of your conda environments.*

|  |  |
| --- | --- |
| **Useful conda commands** | |
| conda env list | # list all environments |
| conda list | # list packages in active environment |
| conda env remove -n envname | # remove an environment |
| conda config --show cannels | # view configured channels |
| conda deactivate | # deactivate environment |
| conda create --name clonedenv --clone envtoclone | # clone an environment |

1. Navigate into your new environment:

$ module load anaconda

$ conda activate projectEnvironmentName

1. Install software packages: sra-tools, fastp, bwa, STAR, 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 bedtools samtools star*

1. After this initial setup, for every future login to Alpine to access the software necessary for this RNA-seq pipeline, run these lines:

$ acompile

$ module load anaconda

$ 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 software packages: <https://anaconda.org/bioconda/>

## Overview of job scheduling on Alpine

The next steps of this guide will require use of the software programs we just installed in our conda environment. These programs often require computational power beyond what we can accomplish directly from the command line, so we instead write scripts that we will submit as *jobs* to Alpine’s job manager (Slurm). The ***#SBATCH*** lines you’ll see at the beginning of the bash scripts below are how we request various resources from the job manager. You can read more about how batch jobs work on Alpine [here](https://curc.readthedocs.io/en/stable/running-jobs/batch-jobs.html).

Note that requesting excessive resources, like number of cores or time, can result in your job being de-prioritized and left to sit in the queue longer. It is best to consider what allocations you actually need based on the number of samples in your project and request accordingly.

**If you think a job will require longer than 24 hours based on the number of files you have to process**, you can switch *#SBATCH --qos=normal* to *#SBATCH --qos=long*.

Alternatively, you can follow the instructions in [this guide](Job%20looping%20with%20Fastp%20and%20STAR.docx) (*T:\Rsch-AAvery\PROJECTS\Eileen\Protocols\RNAseq\Job looping with Fastp and STAR.docx*) to automatically submit a separate job for each sample. This will make processing your files go much faster, because instead of looping through multiple files sequentially in one job, multiple jobs processing one sample each will run in parallel. While this may sound ideal, I generally recommend limiting the volume of jobs you submit whenever possible, because it’s much more difficult to cancel jobs that go wrong if you have hundreds of them going at once, and it’s obviously also more cumbersome to sift through hundreds of log files for debugging than just one.

## 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 Alpine, activate conda environment, and navigate to the project directory.

$ acompile *# Switch to a compile node*

$ module load anaconda

$ conda activate projectEnvironmentName

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

1. Create a script within the script subdirectory for this project (02\_scripts).

$ cd 02\_scripts

$ **touch** fastp.sh

1. Open the fastp.sh script you just created (either with nano or from a Jupyter Notebook), paste the following, modify accordingly, and save.

#!/bin/bash

# execute fastp on two paired-end reads of fastq files

#SBATCH --job-name=execute\_fastp

#SBATCH --nodes=1

#SBATCH --ntasks=16 *# modify this number to reflect how many cores you want to use (FYI, the max fastp can utilize is 16 threads)*

#SBATCH --partition=amilan *# Alpine’s general compute node*

#SBATCH --qos=normal *# If you need more than 24 hours, you can change this to qos=long*

#SBATCH --time=23:00:00 *# modify this to reflect how long to let the job go (HH:MM:SS). Max is 24 hours.*

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

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

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

# Initiate a bash array called SAMPLEIDs.

SAMPLEIDs="CI101495 CI102794 CI104568 etc” *# Space delimited list of Sample IDs in quotation marks; need to match the sample ID in the fastq filename.*

# Loop through the SAMPLEID array and run fastp on each:

for SAMPLEID in $SAMPLEIDs

do

fastp -i ../01\_input/${SAMPLEID}\_1.fq.gz\

-I ../01\_input/${SAMPLEID}\_2.fq.gz\

-o ../03\_output/${SAMPLEID}\_trim\_1.fq\

-O ../03\_output/${SAMPLEID}\_trim\_2.fq\

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

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

--detect\_adapter\_for\_pe\

--thread 16\

-g -x -p

done

1. Submit job to run fastp.sh to the Slurm scheduler. Output will be stored in the *03\_output* directory. Expected run time: ~20-30 minutes per sample.

$ module load slurm/alpine *# Load up the slurm Alpine module*

$ pwd *# Make sure you are still in your scripts directory*

$ sbatch fastp.sh  
$ squeue -u $USER *# Make sure job was submitted successfully*

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. Switch to a compile node on Alpine and request more time than the default 1 hr (these files take a while to download, and the download will be interrupted if you hit your time limit).

$ acompile --time=8:00:00 *# 8 hours*

1. Activate the conda environment where STAR is installed. Then, navigate to your project directory and make a new subdirectory for the index build:

$ module load anaconda

$ conda activate projectEnvironmentName

$ cd /scratch/alpine/$USER/yourProjectDirectory

$ mkdir indicesBuild

1. Acquire the appropriate reference genome files from Ensembl.
   1. To do this, search Ensembl for the genome you want to use as your reference genome and navigate to the “Download DNA sequence” (FASTA) links. To download all the files on this page to Alpine, **replace the https:// with rsync://** and **place /ensembl/ after the ensembl.org** (as described by Ensembl [here](https://useast.ensembl.org/info/data/ftp/rsync.html)). Add a period at the end to specify you want them downloaded in your current working directory.
      1. Then, navigate to the “Download GTF” link under *Gene Annotation* for your selected genome and repeat the same process to grab all the files on this page.
      2. *See Table below for pre-made links to the major canine reference genomes; keep in mind these may need to be updated, as they were based on Ensembl release 109.*

|  |  |  |
| --- | --- | --- |
| Genome | DNA sequence | GTF file |
| ROS CFam 1.0 | $ rsync -azvP rsync://ftp.ensembl.org/ensembl/pub/release-109/fasta/canis\_lupus\_familiaris/dna/ . | $ rsync -azvP rsync://ftp.ensembl.org/ensembl/pub/release-109/gtf/canis\_lupus\_familiaris/ . |
| CanFam 3.1 | $ rsync -azvP rsync://ftp.ensembl.org/ensembl/pub/release-104/fasta/canis\_lupus\_familiaris/dna/ . | $ rsync -azvP rsync://ftp.ensembl.org/ensembl/pub/release-104/gtf/canis\_lupus\_familiaris/ . |
| UU\_Cfam\_GSD\_1.0 (CanFam4) | $ rsync -azvP rsync://ftp.ensembl.org/ensembl/pub/release-109/fasta/canis\_lupus\_familiarisgsd/dna/ . | $ rsync -azvP rsync://ftp.ensembl.org/ensembl/pub/release-109/gtf/canis\_lupus\_familiarisgsd/ . |
| Dog10K\_Boxer\_Tasha (CanFam6) | $ rsync -azvP rsync://ftp.ensembl.org/ensembl/pub/release-109/fasta/canis\_lupus\_familiarisboxer/dna/ . | $ rsync -azvP rsync://ftp.ensembl.org/ensembl/pub/release-109/gtf/canis\_lupus\_familiarisboxer/ . |

1. After the data transfer to Alpine, 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 to ensure the downloaded fasta and gtf files match the associated CHECKSUMS file on Ensembl.

$ pwd # Make sure you’re in your indicesBuild directory; otherwise, navigate there.  
$ sum \*.gz > genomeFileSums.txt

1. Once sums have been confirmed to match, gunzip all remaining .gz genome files, then concatenate the individual chromosome fasta files into one fasta file:

$ gunzip \*.gz

$ cat \*.fa > concatenatedGenomeFilename.fa

1. In your indicesBuild directory, create a script called STAR\_buildIndices.sh.

$ cd /scratch/alpine/$USER/yourProjectDirectory/indicesBuild  
$ touch STAR\_buildIndices.sh

1. Open the script you just created (using nano or a Jupyter Notebook), paste the following, modify as necessary, and save.

*#!/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 32)*

#SBATCH --partition=amilan *# Alpine’s general compute node*

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

#SBATCH --qos=normal

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

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

#SBATCH --mail-user=eid@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

1. Submit the STAR\_buildIndices.sh script to the Slurm scheduler. Output will be stored in the *indicesBuild* directory.

$ sbatch STAR\_buildIndices.sh  
$ squeue -u $USER *# Make sure job was submitted successfully*

1. Once the job has completed, copy the contents of the **geneInfo.tab** file from the output into an Excel spreadsheet on your local computer. Label the first column “probe\_id”, the second column “gene\_name”, and the third column “description.” Save as a .CSV. This will serve as the reference file for matching probe IDs to gene symbols in downstream analyses using R. Repeat for every genome indexed.

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

1. Navigate to your 02\_scripts directory and create a script *STAR\_alignReads.sh*.

$ cd /scratch/alpine/$USER/yourProjectDirectory/02\_scripts  
$ touch STAR\_alignReads.sh

1. Open the script you just created (using nano or a Jupyter Notebook), paste the following, modify as necessary, and save.
   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). Since we are using paired-end reads, read1 and read2 have to be supplied (in that order, separated by a space).*
   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=16 *# modify this number to reflect how many cores you want to use (up to 32), and update the --runThreadN command below to match*

#SBATCH --partition=amilan

#SBATCH --qos=normal

#SBATCH --time=23:00:00

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

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

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

#Make an output directory for results:

mkdir ../03\_output/STAR\_Alignment\_to\_GenomeName

# Create an array of sample IDs for looping.

SampleIDs="CI100641 CI100850 CI100993 … etc." *# Space delimited list of sample IDs in quotation marks.*

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

for SampleID in $SampleIDs

do

STAR \

--runThreadN 16 \

--genomeDir ../indicesBuild/directory-containing-STAR-generated-genome-index-files\

--quantMode GeneCounts \

--outSAMtype BAM SortedByCoordinate \

--outReadsUnmapped Fastx \

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

--readFilesIn ../03\_output/${SampleID}\_trim\_1.fq ../03\_output/${SampleID}\_trim\_2.fq \

--outFileNamePrefix ../03\_output/STAR\_Alignment\_to\_GenomeName/${SampleID}

done

1. Submit the STAR\_alignReads.sh script to the Slurm scheduler. Output will be stored in *03\_output/STAR\_Alignment\_to\_GenomeName*.

$ sbatch STAR\_alignReads.sh  
$ squeue -u $USER *# Make sure job was submitted successfully*

## Index BAM files with samtools

*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.*

1. Navigate to the directory containing the .bam output files from our STAR alignment.

$ cd /scratch/alpine/$USER/yourProjectDirectory/03\_output/STAR\_Alignment\_to\_GenomeName

1. Once in the directory containing all bam files, run this line of code. Output *.bai* files will be stored in the same directory.

$ for bam in $(ls \*.bam); do samtools index $bam; done

## 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:

$ acompile

$ module load anaconda

$ conda activate ProjectEnvironmentName

$ conda **install** -c bioconda subread

1. Navigate to the directory containing your BAM alignment files. Create a vector of space-delimited BAM file names for featureCounts input:

$ cd /scratch/alpine/$USER/yourProjectDirectory/03\_output/STAR\_Alignment\_to\_GenomeName

1. Once in the directory containing your BAM files, run the following line of code to create a vector of space-delimited BAM file names to use as our input for featureCounts:

$ ls \*.bam | tr '\n' ' ' > bamList.txt

1. Create a new script called tabulateCounts.sh in this same directory. Running the script in this directory will allow us to easy paste in the BAM files names without having to also indicate a path to those files.

$ touch tabulateCounts.sh

1. Open the script you just created (using nano or a Jupyter Notebook), paste the following, modify accordingly, and save.
   1. Replace “**BAMfile1**”, “**BAMfile2**”, “**BAMfile3**”, etc. with the pasted list of space-delimited BAM file names from **bamList.txt**. *Pay attention to directory paths, as we are not in the usual 02\_scripts directory for this step. Remember that “..” in a directory path means “go up one level.”*

#!/usr/bin/env bash

#SBATCH --nodes=1

#SBATCH --ntasks=4 *# Make sure this matches the -T command below.*

#SBATCH --time=12:00:00

#SBATCH --partition=amilan

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

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

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

featureCounts -p -T 4 -a ../../indicesBuild/path-to-genome-gtf-file.gtf -o ../FileName\_feature\_counts.txt BAMfile1 BAMfile2 BAMfile3

1. Submit tabulateCounts.sh to the Slurm scheduler. Output will be stored in *03\_output.*

$ sbatch tabulateCounts.sh

$ squeue -u $USER *# Make sure job was submitted successfully*

## Perform QA/QC

1. Install necessary software:

$ acompile

$ module load anaconda

$ conda activate ProjectEnvironmentName

$ conda **install** -c bioconda fastqc multiqc

1. Navigate to your 03\_output directory that contains the trimmed .fq files (fastp output), then create a script called fastQC.sh.

$ cd /scratch/alpine/$USER/yourProjectDirectory/03\_output  
$ ls *# Make sure you see the trimmed fastq files we generated with fastp*$ touch fastQC.sh

1. Open the script you just created (using nano or a Jupyter Notebook), paste the following, modify accordingly, and save.

#!/bin/bash

*# Running fastQC on multiple trimmed fastq files.*

#SBATCH --job-name=execute\_fastQC

#SBATCH --nodes=1

#SBATCH --ntasks=16 *# modify this number to reflect how many cores you want to use (up to 32); make sure it matches the -t option below.*

#SBATCH --partition=amilan

#SBATCH --qos=normal

#SBATCH --time=4:00:00

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

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

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

# Run fastQC on all .fq files in the current directory.

fastqc -t 16 \*.fq

1. Submit the fastQC.sh script to the Slurm scheduler. Output will be stored in *03\_output*.

$ sbatch fastQC.sh  
$ squeue -u $USER *# Make sure job was submitted successfully*

1. Repeat the process for the raw fq.gz files in your *01\_input* directory.

$ pwd *# Make sure you’re still in your 03\_output directory where the fastQC.sh script is*  
$ cp fastQC.sh ../01\_input *# This will make a copy of the script in the 01\_input directory*  
$ cd /scratch/alpine/$USER/yourProjectDirectory/01\_input *# Navigate to your 01\_input directory*  
$ ls *# Make sure you see your .fq.gz files in this directory*$ sbatch fastQC.sh *# Submit a job to run the fastQC.sh script in this direcotry*

* 1. Open the fastQC.sh script in your *01\_input* directory and **modify only the last line** to run on all fq.gz files.

fastqc -t 16 **\*.fq.gz**

* 1. Save the modified script and submit the fastQC.sh script to the Slurm scheduler from your 01\_input directory. Output will be stored in *01\_input.*

1. To make it easier to look at the output for all samples at once, we’re going to use the multiQC tool. Navigate to the base directory for this project and then execute the following in the command line:

$ cd /scratch/alpine/$USER/yourProjectDirectory  
$ multiqc --outdir ./MultiQC .

* 1. This will comb through all of the subdirectories of our project and give a consolidated HTML report that summarizes all the QC data from the fastQC files generated for each sample, as well as the log.out files for STAR alignment and the .summary file for featureCounts.
  2. NOTE: You can use the *ignore* flag if there are any files you want excluded from the multiQC report.

$ multiqc --ignore “file1.txt” --ignore “./entire/directory/\*” --outdir ./MultiQC .

1. Download the resultant multiQC html report to your local computer for your reference. This information is often required when publishing RNA-seq data.

## Prepare metadata for DESeq2

*Note: The instructions for creating these metadata files are intended to ensure compatibility with the code provided in the DESeq2 Rmarkdown file referenced in the “Normalization and modeling with DESeq2” section below. Choosing different column names or adding/excluding columns other than what’s shown here will likely necessitate changes to the DESeq2 script as well.*

1. On **the local computer that will be running R (not Alpine),** start a project directory.
   1. For easiest pathing, it’s best to create this directory in your home directory (Mac) or (C:) directory (Windows), but it’s arbitrary since we can copy our path name from anywhere.
2. Initiate subdirectories in this project directory like we did on Alpine:

*01\_input*

*02\_scripts*

*03\_output*

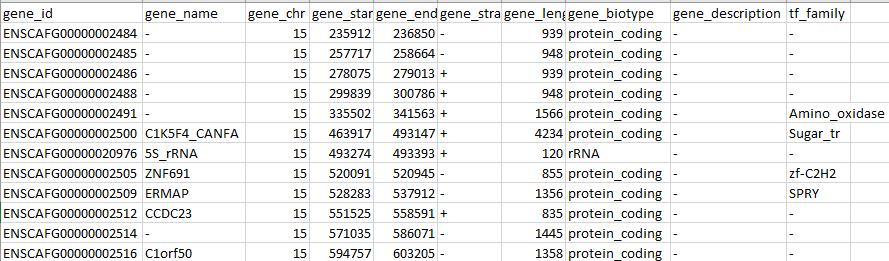
1. Create a metadata file in which **the first column is the sample ID** for each sample, and subsequent columns contain relevant information about each sample (e.g., experiment group vs. controls). You can create this metadata file in Excel and then export as a **tab-delimited .txt** file. **This metadata file should have NO header!!!**
   1. *Example:*

CI101495 CD4\_PTCL

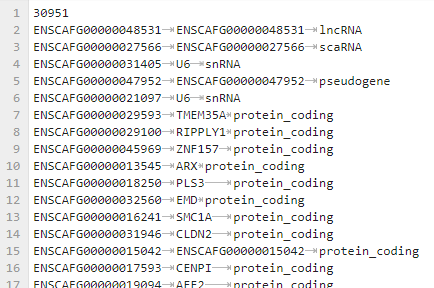
CI156616 CD4\_LN\_CTRL

CI104568 CD4\_PTCL

1. Create a file correlating Ensembl gene ID to gene name.
   1. **Option 1:** Pull this information from the **gene.xls** file in your Novogene results. This file should look something like this:



* 1. **Option 2:** Navigate into the folder of your indicesBuild directory on Alpine that contains your genome index. In this folder, you should see a file called **geneInfo.tab**. It should look something like this, containing both Ensembl IDs and gene names:



* + 1. Download this file and open it in Excel, choosing *tab* as the delimiter for column breaks.
  1. Create a new spreadsheet in Excel where the **first column is labeled *probe\_id***, the **second column is labeled *gene\_name***, and the **third column is labeled *description***. Then, paste the list of Ensembl IDs into the first column, the corresponding list gene names into the second column, and the corresponding description (e.g., lncRNA, protein\_coding) into the third column.
     1. *Example:*

|  |  |  |
| --- | --- | --- |
| probe\_id | gene\_name | description |
| ENSCAFG00000047952 | ENSCAFG00000047952 | pseudogene |
| ENSCAFG00000021097 | U6 | snRNA |
| ENSCAFG00000029593 | TMEM35A | protein\_coding |
| ENSCAFG00000029100 | RIPPLY1 | protein\_coding |

* + 1. Once pasted, sort the gene\_name column A->Z to bring the blanks to the top. Copy and paste the probe\_id into the gene\_name column for these rows.
    2. Save this as a **.csv** file to the local computer that will be running DESeq2 in R.
  1. Save these metadata files in your local **01\_input** project folder.

## Normalization and modeling with DESeq2

1. Download the feature\_counts.txt output file from Alpine to the *01\_input* directory on your local computer.
2. Make sure you have R and Rstudio installed.
   1. Instructions on how to do so can be found here: <https://www.stat.colostate.edu/~jah/talks_public_html/isec2020/installRStudio.html>
3. Obtain the necessary R-Markdown file for running DESeq2 in Rstudio. ***Save a copy to your local 02\_scripts directory and do not modify the originals.***
   1. All of my RNA-seq related R scripts are stored here on the T-drive: T:\Rsch-AAvery\PROJECTS\Eileen\Protocols\RNAseq\
      1. The **Bioconductor Package Install.Rmd** script will automatically install all the required software packages in R. Copy this file to your local computer and run it in Rstudio.
      2. The **2023-RNAseq-Analysis.Rmd** script is the starting place for data normalization and differential gene expression analysis with DESeq2. Copy this file to your local computer and open it in Rstudio.
      3. Scripts for other downstream analyses (e.g., GSEA, GSVA, hierarchical clustering) are also located in this directory for use in Rstudio. See the **README** document in this directory for a description of the application of each script.
      4. ***Read carefully through each script and modify as necessary to fit your project!!!***