# TACC Workshop 1:

Here is a synopsis of our first TACC workshop. By next week, please make sure that you are comfortable logging in, moving between directories, viewing the contents of your directory, making a file, and reading out a file.

Since genomic sequences are massive files, it'd be too difficult to analyze them visually. So to handle our genomic sequences, we'll be using the supercomputer called Lonestar6 in TACC. First off- if you're using Windows, you'll need to download a software called Putty in order to interact with TACC. If you're on a Mac, you can communicate with TACC directly through your terminal (found in Utilities).

To log onto Lonestar6 via your Mac terminal:
**ssh <your username>**
ex. ssh kblack@ls6.tacc.utexas.edu

Then, it will ask for your password and token code (which is a multi-factor authentication that goes to your phone. You will need to set it up in your TACC account online). Once you enter it, and you see a new line starting with:
login1.ls6$
...or something similar to that, then you are now working directly in the Lonestar6 supercomputer.

To log in with Putty, follow this:

If you're using **PuTTY** as your Terminal from Windows:

- Double-click the **Putty** icon
- In the **PuTTY Configuration** window
  - make sure the **Connection type** is **SSH**
  - enter **ls6.tacc.utexas.edu** for Host Name
    - Optional: to save this configuration for further use:
      - Enter **Lonestar6** into the **Saved Sessions** text box, then click **Save**
      - Next time select **Lonestar6** from the **Saved Sessions** list and click **Load**.
  - click **Open** button
  - answer **Yes** to the SSH security question
- In the PuTTY terminal
  - enter your TACC user id after the **"login as:"** prompt, then **Enter**
  - enter the password associated with your TACC account
  - provide your 2-factor authentication code

Once you log in, you are working from a "login node" in your Home directory. From here, you can navigate between directories/folders that hold your sequences, genomes, apps, etc.

You will have three different main directories: Home, Work, and Scratch. Each of these are for storing different files- you are given 12 GB in home, 1024 GB in work, and unlimited GB in scratch.

In Home, you can store your apps, small programs, scripts, and binaries.

In Work, you can store sequences, genomes, final project data, and anything that you don't want to lose. It is stored forever.

In Scratch, you can put anything and store things that you are actively working on. You have infinite storage here, but if you don't touch these files in 2 weeks, they may be deleted forever. So, whenever you're working in Scratch and get a final product that you want to save, move it to Work.

If you want to check which directory/folder you are in, type:
**pwd**
meaning "present working directory" and it will tell you where you are.

You can change directories by typing:
**cd**
meaning "change directory" and you can move around. For example, type **cds** to move to Scratch, **cdw** to move to Work, or **cd** to move to Home. If you want to move to a specific subdirectory, you can also do that by typing: cd <entire pathname>. For example, typing:
**cd /work/06909/cbscott/ls6**
will also take you to work.

You can also use shortcuts like $WORK (ex. **cd $WORK**), $HOME, or $SCRATCH to move around. But choose whichever way that is easiest for you.

If you want to see what is inside a directory, type:
**ls**
meaning "list everything in here". And it will list out all files sitting in that directory.

If you want to make a new directory, type: mkdir <name of new directory>

For example, go to scratch: **cds**
Make a new directory in scratch called "new": **mkdir new**
Go inside your new directory: **cd new**
List everything inside your new directory: **ls new**

You should have nothing listed inside it yet, since it is brand new.

If you want to make a new text file in your new directory, you can type:
**echo "new test file" > myfile.txt**
This means that you are echoing a phrase into a file called myfile.txt. You can echo any phrase into a file called anything you want.

To check if your file is there, type **ls**. Your new file should be listed, along with the date it was created and the size of your file.

To read your file, type:
**cat myfile.txt**
And it should print out everything written inside your text file.

Some important shortcuts to remember:

**ssh=** secure login
**pwd=** present working directory
**cd=** change directory
**ls=** list everything in this directory
**mkdir=** make a new directory
**echo=** paste a phrase into a new file
**cat=** read all contents of a file

# TACC Workshop 2:

Today we learned how to download and install conda. By next time, please make sure you have conda installed and working.

Before we can work with any data, we'll need to set up conda to install the software we'll need to use. Conda is a package manager that can install your software plus all the dependencies it requires.

Step by step instructions to download and use conda are on our github. You can copy those lines and paste them directly to your command line, and they should work. Be sure to change any usernames though.

How to download conda:

Once you're logged into TACC in your terminal, go to your work directory where you should store your software:
**cdw**
make a new directory for your software:
**mkdir software**
check that your new directory exists:
**ls**
go into your new directory:
**cd software**
check that you're in your software directory now:
**pwd**
it should show something like: **kblack/work/software**

***New command to download anything from the internet!***
**wget**

So, to download conda:
**wget https://repo.anaconda.com/archive/Anaconda3-2021.11-Linux-x86_64.sh**
Check that conda is now in your software directory:
**ls**
** If you wanted to remove it and try again, you can do:
**rm <file>**

Make your newly downloaded file executable
**chmod +x <file>**
Your file should now appear green, which means it is executable. To run the file now, do:

**./file**
It should give you a bunch of information. Just scroll to the bottom and accept the terms and conditions. If it asks whether you want to initialize "conda init", choose:
**Yes**
Then conda will try to install itself in your home directory. To specify where you want conda to be installed:
**/work/your number/username/ls6/software/anaconda3**
*your number and username can be checked by: **pwd**
Before you can start using conda, you'll need to logout and log back in.

When you log back in, you should see **(base)** in front of your command line. This means conda is successfully running!
Now go back to work:
**cdw**
And you'll have to tell conda where to install things. Run these next lines in order (these lines are from line 34 in the github instructions):
**conda config --add channels defaults**
**conda config --add channels bioconda**
**conda config --add channels conda-forge**
If it gives you a warning, that's fine. Just ignore.

Now let's set up your first conda environment!
Choose a name for your environment. It's best to relate this to your project name (like STX for example).
**conda create --name STX**
If it asks whether you want to proceed, choose **Yes.**
Now you have a "conda environment" that will store all the software you download to work on this particular project. Before you can use the software in that environment, you'll need to activate it.
**conda activate STX**
You should now see **(STX)** in front of your command line, showing that you're working in your STX conda environment.

Lastly, a couple tips for working from the command line:
1. To abort any process, type ^c three times.
2. When naming your files, use underscores (definitely not spaces!)

# TACC Workshop 3:

Today we learned how to download genetic sequences from NCBI. By next time, please make sure that you have downloaded a dataset of fastq files into your scratch directory.

Before we get started, let's install some software. First, I'd recommend you download some type of script editor to store, build, and edit all your scripts. I use Visual Studio Code and highly recommend it. Make a new blank script for your project and start typing or pasting all your code in there. Then all code from here on out can be copied and pasted from your editor to your terminal.

Within TACC, we installed conda last time (which will definitely come in handy for future installations) but I've never used the conda installations of these particular software, so we'll do it in a way that I'm positive will work.

Software we need: Edirect, SRA toolkit, and Misha's 2brad scripts

To install Edirect, log into TACC and paste:
**cd**
**sh -c "$(wget -q ftp://ftp.ncbi.nlm.nih.gov/entrez/entrezdirect/install-edirect.sh -O -)"**
To check that it's there, type:
**ls**
you should see edirect in your home directory.

To install SRA toolkit, paste this:
**cd**
**wget http://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-ubuntu64.tar.gz**
**tar vxf sratoolkit.current-ubuntu64.tar.gz**
Check that it's there again with **ls**.

To install Misha's 2bRAD scripts, paste this (you can copy and paste the whole thing, including the #comments).
**cd**
**mkdir bin**
**cd ~/bin**
**git clone https://github.com/z0on/2bRAD_denovo.git** # cloning Misha's scripts
**mv 2bRAD_denovo/* .** # move scripts to ~/bin from sub-directories
**rm -rf 2bRAD_denovo** # remove now-empty directory
**chmod +x *.pl.** # designating all .pl and .py files (perl and python scripts) as executable
**chmod +x *.py**
**chmod +x *.R**

Now you've installed all the things. But since you installed them in Home, you can only use them if you're in Home. But we're mostly going to work in Scratch. So we need to set a path to each program so that we can access them from any directory in TACC.  To do this, we will add a path for each program in a text file called .bashrc

******THIS IS REALLY IMPORTANT! BE CAREFUL WHEN EDITING YOUR .BASHRC. ACCIDENTAL TYPOS OR DELETIONS CAN MAKE YOUR WHOLE TACC ACCOUNT STOP WORKING**********

But don't be scared. Here's how you do it:
**cd**
**nano .bashrc**
You are now in a text editor (called nano) looking at your .bashrc file. There is a lot of info here, but you only need to scroll down to Section 2.
First, delete the "#" in front of # export PATH=$HOME/bin:$PATH, so it is now:
**export PATH=$HOME/bin:$PATH**
Right below it, paste:
**export PATH=$HOME/sratoolkit.3.0.1-ubuntu64/bin:$PATH**
**export PATH=$HOME/edirect:$PATH**
You now have three pathways written into your .bashrc. To save it hit
**Ctrl+O**
Hit **Enter** to save.
To exit, hit
**Ctrl+X**
You should be back home. Before the changes to your .bashrc can take effect, you'll need to log out and log back in.
When you're logged back in, let's check if the paths are set. Paste:
**which ls6_launcher_creator.py**
It should show you: ~/bin/ls6_launcher_creator.py, which means that it knows the pathway to this particular python script. If the paths are not set, it will say "no ls6_launcher_creator.py in <a whole paragraph of paths>". If you see this, you'll need to try resetting the paths.

So now we have everything installed, let's download some data!
First go to scratch and make a directory for this practice project (I'm calling my new directory "practice").
**cds** # Go to scratch
**mkdir practice** # Make a new directory called practice
**ls** # Check that your new directory is there
**cd practice** # Go into your new directory
In here, we're going to download a specific 2bRAD dataset of coral samples from NCBI. To do this, paste:

**export BioProject=PRJNA812916**
**$HOME/edirect/esearch -db sra -query $BioProject | efetch -format runinfo |cut -d "," -f 1 | grep SRR > $BioProject.SRR && $HOME/edirect/esearch -db sra -query $BioProject | efetch -format runinfo > $BioProject.fullMeta.csv**
This should download metadata for the coral samples in this bioproject in NCBI's database. Check that it's there (should be a .csv file):
**ls**

Now, let's download the sequences. There are 549 samples in this project, and genomic sequences are giant .fastq files. This is a lot of data to import, so we're going to submit a "job." This is the first time we get to use the power of TACC!

Make a file called "gets" with a for-loop that downloads each sample. Paste:
**>gets**
**for A in `cat $BioProject.SRR`;do**
**echo "fastq-dump-orig.3.0.1 $A">>gets;**
**done**
Check that your new file called "gets" is there.
**ls**
Let's see what's inside:
**cat gets**
There are lines on lines on lines that look like:
fastq-dump-orig.3.0.1 SRRxxx
fastq-dump-orig.3.0.1 SRRxxx
fastq-dump-orig.3.0.1 SRRxxx
...and so on.
This is the expanded for-loop, where every line is a command to dump the fastq file for each sample into your directory.

Now that we've written a job, let's submit it to TACC. First, insert your email into the code below, and paste:
**ls6_launcher_creator.py -j gets -n gets -a IBN21018 -e <your email> -t 20:00:00 -w 12 -q normal**
**sbatch gets.slurm**

It should tell you that you've submitted a job. This means that your job "gets" is in the queue to run for 20 hours. To see if your job is running, type:
**squeue -u <your username>**
It will show your job ID, your username, your job name, the status, and how long it's been running. If your status (ST) says PD, then your job is pending (or it's sitting in the queue and will run once the computer finishes other people's jobs). If your status says R, then it is running! Whether it's running or now, you can now log off and it will passively work in the background. Once your job starts running, it will send you and email to let you know. It will also send you an email once it's finished. It's unlikely that this job will take 20 full hours, but it should take more

than one hour. If you get an email that your job finished in a few seconds, then it failed (probably because of a typo somewhere).

Once your job is finished, you can log back into TACC and go to your project directory to see all your sequences in there.

If your job fails, here's how to check what went wrong:

Log into TACC, and go to your directory in scratch where you submitted the job.
**cds**
**cd practice**
**ls**
You should see some new files here, including:
gets.oxxxxx
gets.exxxxx (where xxxx are some numbers)

The gets.exxxxx is your error file. To see where your job errored, look at the last few lines:
**tail gets.exxxxx**

If the error looks like something you can fix, give it a try and submit the job again. If you're not sure what it means, you can send it to me or Carly and we'll help you fix it.

# TACC Workshop 4:

So by now you should have all your fastq sequences downloaded into your scratch directory.

Now that your directory is starting to fill up with many files, it can be hard to find specific files. So instead of using **ls** to see what's in your directory, let's start using
**ll -tr**
This will list out all the contents of your directory in reverse chronological order. So the newest created (or added) files will be at the bottom.

Let's also look inside our fastq files to see what they look like. To see the first 10 lines of a file, try
**head <file.fastq>**

And it should show you a bunch of text including nucleotide sequences (AGCTCGATACGAT or something like that). These are your raw sequencing reads. Each read has a header with an arbitrary name and a length. Underneath that, you'll see a bunch of weird characters that aren't readable, but these show the quality of the reads. Somehow this code is interpreted by other software down the line so we can filter out low quality reads.

> Note: It's always good to look inside the top 10 lines of new files to make sure nothing looks severely wrong.

This particular dataset contains two species of coral, but we only want to work with one (the lettuce coral *Agaricia agaricites*). So we'll need to split up the dataset by species. Let's take a look at the metadata file. To do this, let's *not* use **head** because we might want to see more than 10 lines. So let's try **less -S** so we can scroll through the file.
**less -S PRJNA812916.fullMeta.csv**
Now you can scroll through it, but it's still a lot. Type **q** to exit.

> Note: If you have multiple metadata files and aren't sure how many you have, you can list them out using * as a wildcard character. For example
> **ls *fullMeta***
> will all files with the string "fullMeta" in the name. Hopefully you only have one (the one associated with this bioproject #). If you have others, you can remove them with
> **rm <file>**

Let's just look for Agaricia in the metadata. We'll use grep to find characters
**grep *Agaricia* PRJNA812916.fullMeta.csv**
And it should only show you the lines containing Agaricia in them.

This is still a lot of information, so let's take those specific lines and pipe them into a smaller file without the other coral species data. Pipe is this character: |

**grep *Agaricia* PRJNA812916.fullMeta.csv | awk -F, '{print $1}' | head**

This line finds all lines with the string "Agaricia" in it, then we pipe it into the next command, which prints the first column (containing sequence names), and then we pipe it into the next command to look at the first 10 files.

And now you should see the files that are for Agaricia. It should look something like
SRRxxx.fastq
SRRxxx.fastq
SRRxxx.fastq

If you see the list of sequences, then let's do it again but this time let's pipe the sequence names into a new file:

**grep *Agaricia* PRJNA812916.fullMeta.csv | awk -F, '{print $1}' > agaricia_names**

Let's check that this new file contains the sequence names
**head agaricia_names**

Now you have the names of all Agaricia sequences in one file. However, these aren't the original names of the samples, these are the names that NCBI gave them. We'll need to rename them with their correct names (that are also listed in the metadata file). Let's first make a new directory to separate all our Agaricia sequences.
**mkdir Agaricia**

Our new Agaricia directory is empty. We'll need to fill it with the Agaricia sequences. Let's make a for loop to move all the Agaricia sequences into our empty directory

**For file in `cat agaricia_names`; do mv ${file}.fastq Agaricia; done**

Let's go inside our directory and make sure they're in there
**cd Agaricia**
**ls**
You should now see fastq files in your Agaricia directory, but not all of them! Only the portion that are for Agaricia should be in there.

The next step in this process is to map all of these reads to a reference genome. Because we're working with a nonmodel organism (*Agaricia agaricites*), there is no reference genome we can simply download. Instead we'll need to make a "de novo" genome, composed of all our reads aligned together.

First let's install some software we'll need in a conda environment.
Let's activate our conda environment:
**conda activate STX**

And now let's install the following software into our environment: cutadapt, samtools, jellyfish, and bowtie2.
**conda install -c bioconda cutadapt**
**conda install -c bioconda samtools**
**conda install -c conda-forge jellyfish**
**conda install -c bioconda bowtie2**

Now your software is installed and ready for mapping! (next week)

Some important shortcuts to remember:

**ll -tr=** list all files in directory in reverse chronological order
**less -S=** view entire file by scrolling. To exit, type q
**grep=** similar to Ctrl+F to find a specific string of characters
**|** = Pipe, take the output of one command and input it into the next command
***=** wildcard character, helps when referring to multiple files with the same prefix/suffix

***** This is as far as we got before working with our own sequences. From here, switch to STX.sh for the remaining code.