Instructions for running many SLiM simulations on the cluster

If you run the same SLiM simulation more than one time, the results can vary from on simulation to the next due to random chance (genetic drift). So, we want to be able to run the same simulation many times so that we can get an idea for the distrubtion of our statistics of interest under neutrality, taking into account genetic drift (e.g. global ancestry, local ancestry, allele frequency).

Later on, we'll also want to run the same general simulation with some variables changing from one simulation to the next (e.g. simulating different selection strengths for a mutation of interest).

In order to do that we have to do a few things:

1) Make sure we can keep track of the output of indepndent simulation runs

2) Set up a way to run independent simulations in parallel on the cluster

1) Modify SLiM outname

First, we have to modify the SLiM script so that our output .trees file is named differently each time we run the simulation A simple way to do that is to add a 'seed' variable to the outname.

In the final generation of your SLiM simulation you can modify your "outname" line to add a seed variable:

```
outname = "" + "/NeutralSimulationName_seed-" + seed + ".trees";
```

Modify `"NeutralSimulationName"` to the shared name you want all your simulations to have. The `seed` variable will take an argument passed from the command line, and add that to the name of the file.

For example, if you run:

```
slim -d seed='"1"' neutralsimulation.slim
```

The output of that would be "/NeutralSimulationName_seed-1.trees". Changing that seed value will be how we keep track of our simulation runs.

2) Set up job array script

We can run each simulation sequentially, but that may take a long time and doesn't take advantage of the many available nodes on the cluster. Instead, we can run them in parallel by launching independent but similar jobs. The best way to do this is through a job array.

For example, you can create a new file and write the following:

```
#!/bin/bash
#SBATCH -p scavenger
#SBATCH -a 1-10

c=$SLURM_ARRAY_TASK_ID
```

```
declare -i c10="${c}0"


for i in {0..9};
do

c_array[$i]=$((c10 - i))

done

for i in "${c_array[@]}"
do

/hpc/home/ih49/home/SLiM_build/slim -d seed=$i /path/to/SLiM_simulation.slim

done
```

This script is doing a few things. Let's go through line by line:

1) `#!/bin/bash` tells the script this is a bash file
2) `#SBATCH -p scavenger` tells SLURM we want to use any node that is available on low-priority, rather than waiting around for the common partition (this is best for running many parallel jobs, but you run the risk your job will be canceled and requeued).
3) `#SBATCH -a 1-10` tells SLURM we are running a job array (the same job run many times, with an independent identifier). We are telling it we want to run the job 10 times, with the task IDs from 1 to 10. The jobs will now be grouped together. When you run `squeue -u netID`, you will see many jobs with the same number followed by an underscore and the task ID. E.g. if the job is 123123123, and we ran this script, the jobIDs would be 123123123_1, 123123123_2, 123123123_3, ... , 123123123_10.
4) `c=$SLURM_ARRAY_TASK_ID` gives us the specific task ID for the current job in the array.
5) The next chunk is a little more complicated. This is because the current SLiM simulations are fairly short, and running many very short jobs on the cluster can be a burden on the job scheduler, so you want to group short jobs together. So rather than running one simulation at a time, we will run 10 simulations at a time. That is what this chunk does:

```
declare -i c10="${c}0"


for i in {0..9};
do

c_array[$i]=$((c10 - i))

done

for i in "${c_array[@]}"
do
```

We are essentially using the task ID to calculate 10 seed numbers to iterate over sequentially. For example, if we are on task ID # 2, the seed numbers would be [20, 19, 18, 17, 16, 15, 14, 13, 12, 11]. We then start a for loop, which states "for every seed number in our list of seed numbers, do the following:"

6) We pass the seed number to the following line of code:

```
/hpc/home/ih49/home/SLiM_build/slim -d seed=$i ~/path/to/SLiM_simulation.slim
```

`$i` is the seed number for the current iteration of the for loop. This tells the SLiM simulation to assign the current seed number to the seed variable in this final line of our SLiM script:

```
outname = "" + "/NeutralSimulationName_seed-" + seed + ".trees";
```

In summary:

- We are running 10 TASKS in the JOB ARRAY.
- For each TASK in the JOB ARRAY, we are running 10 SIMULATIONS.
- This totals to 100 SIMULATIONS across all TASKS.
- For each SIMULATION in a TASK, we are passing a unique SEED to the SLiM script.
- the SLiM script will output a new file for each SIMULATION with an unique SEED number in the filename.

Some things to make sure of before you run the scripts:

- modify the scripts to match your correct directories & filenames
- make sure you are outputting your trees files to your work directory (can create a subdirectory for better organization). the files are large, so we won't have room in our group or home directories.
- when debugging, it's useful to run just one task in an array to confirm everything outputs the way you expect it to. when you run the job array from the command line you can do `sbatch -a 1 jobarray.sh`. This will override the job array line within the script, and tell it to run only task 1. Then, you can confirm you get 10 simulations in the expected output folder and with the expected filenames.