Part 1 – Population genetics and evolutionary forces

Exercise 1

In this exercise, you will explore the effect of genetic drift on allele frequencies over time. This exercise requires that you have RStudio installed on your personal machine.

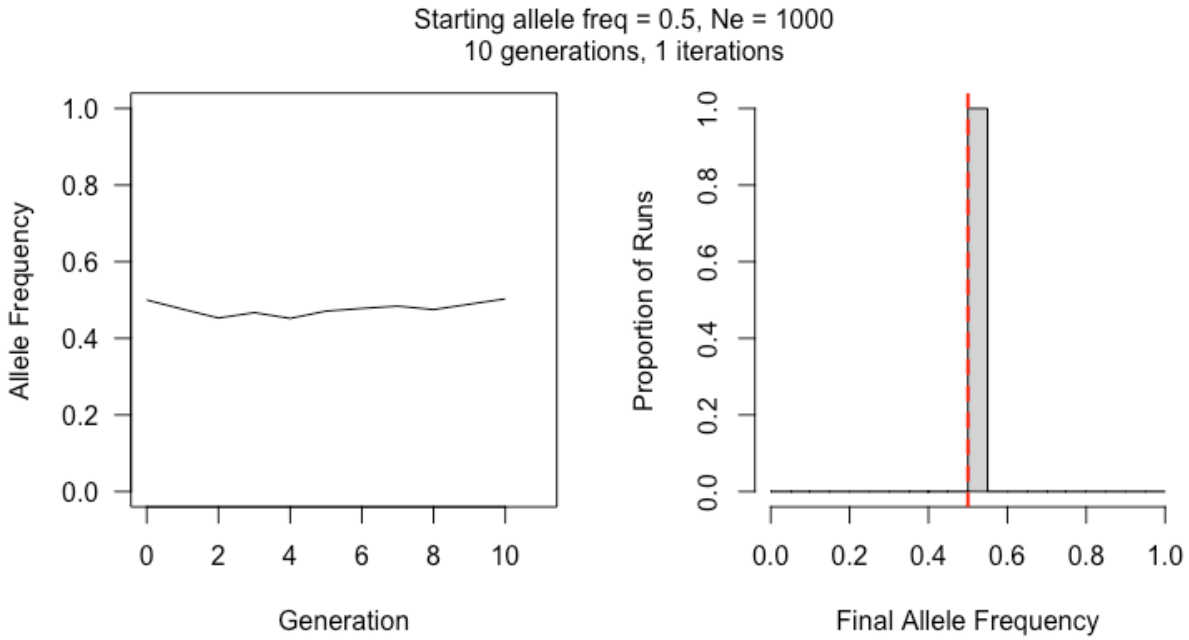
1. Begin by opening RStudio and 'sourcing' the plot_allele_drift.R script. Note that you may need to change your working directory first.

   ```
   > setwd("path/to/script")
   > source("plot_allele_drift.R")
   ```

2. Next, run the **plot.allele.drift** function with the default parameters. This function simulates a population of a given size ('Ne') with a particular starting allele frequency ('f0'). It then samples alleles **with replacement** in the starting generation to produce the next generation. It continues this sampling process for a certain number of generations ('gens') and then stops.

   ```
   > plot.allele.drift()
   ```

3. After running the function, you should see two plots appear in the lower right window of RStudio. These show the result of your drift simulation. They should look similar to this:



The plot on the left shows the **frequency of our allele** (y-axis) **over time**, in generations (x-axis). The plot on the right shows a histogram of our **final allele frequency** compared to 'f0', the **starting allele frequency** (dashed vertical red line). This histogram will be more interesting in subsequent simulations.

4. Start changing parameters in the function. First, **increase the number of runs** ('n_runs') to 25. The 'n_runs' argument allows you to perform and view the results of multiple allele frequency trajectory simulations at once under the same 'Ne', 'f0', and 'gens' parameters.

   ```
   > plot.allele.drift(n_runs=25)
   ```

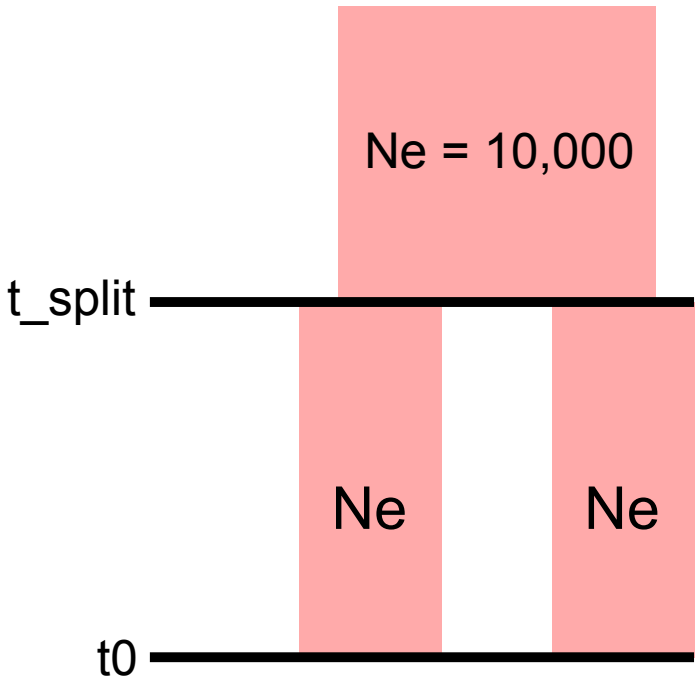
5. Continue exploring this simulation framework by changing the other parameters. Keep in mind that any parameter that you don't explicitly set will take the default value. If any one simulation takes more than a few minutes to run, press the 'stop' icon in RStudio and adjust your parameters (especially n_runs – try to always keep this under 100). As you experiment, think about the following questions:

   - What happens to the distribution of final allele frequencies as you decrease or increase 'Ne'?
   - What happens to the distribution of final allele frequencies as you increase the number of generations?
   - What are the chances that an allele will become fixed (final frequency of 1) or lost (final frequency of 0) in the population? How can you adjust your parameters to increase the occurrence of fixation or loss?

Exercise 2

In this exercise, you will explore how population split time and effective population size affect F_{ST}, a measure of genetic divergence.

1. Take a look at the two_pop.py script - it defines a two-population demographic model and a 1 Mb sequence of DNA evolving neutrally. The mutation rate is 1 x 10^{-8} per base pair per generation. At the end of the simulation (t_0), 100 diploid individuals are sampled from each population and the script calculates F_{ST} between these individuals. See diagram below:



Launch the script with the default parameters. You will need to have python3 and msprime installed on your machine. This is easy to do via conda.

TO INSTALL REQUIRED SOFTWARE

```
> conda install -c anaconda python
> conda install -c conda-forge msprime
```

TO LAUNCH SIMULATION

```
> python3 two_pop_Fst.py
```

The simulation will return some output to the screen. Read the output and answer the following questions:

- When did these populations split?

- What was the ancestral effective population size?
- What are the current effective population sizes?

2. Continue exploring this simulation framework by changing the split time ('t_split') and effective population size ('Ne') parameters. Start by changing only one parameter at a time. How do you think this will change F_{ST}?

   ```
   > python3 two_pop_Fst.py --Ne [your chosen value]
   ```

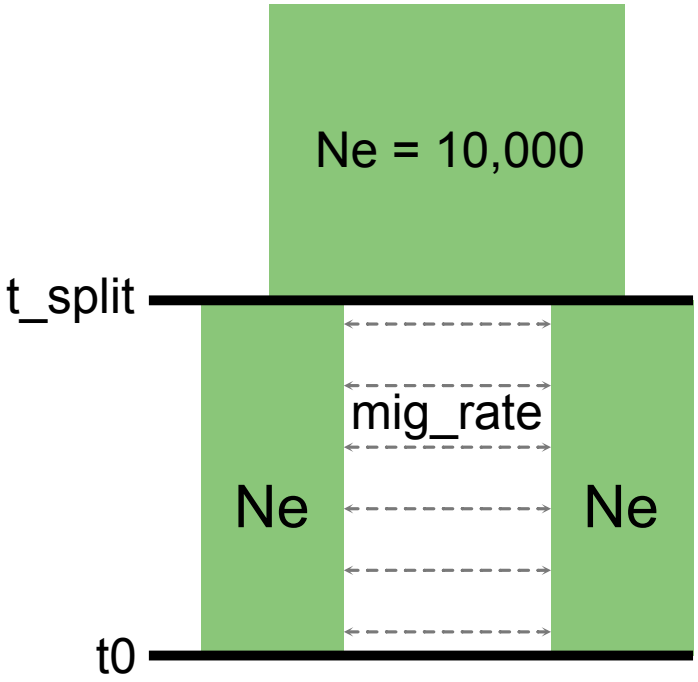
3. Report your starting parameters and results for two simulations using this Google Form. Make sure you are changing **one parameter at a time** and letting the other keep the default value.

Part 2 – Gene flow

Exercise 3

In this exercise, you will add symmetric migration to the two-population model you explored in Exercise 2, and observe the effect on population differentiation.

1. Return to the two_pop.py script. You may have noticed that this script has an additional parameter 'mig_rate'. This parameter defines the symmetric **migration rate** between the two descendent populations. This migration is **symmetric**, meaning the gene flow occurs in both directions. By default, this parameter is set to 0 (no gene flow).



   Change the model so that **at each generation post-split, 0.1% of each descendent population is made up of migrants from the other population**. What migration rate does this translate to? How do you think this will change how the two descendent populations diverge (i.e. what will be the effect on F_{ST})?

   ```
   > python3 two_pop_Fst.py --mig_rate [migration rate value]
   ```

2. Run the model with gene flow 5 times or so and collect the F_{ST} results. Compared to the model **without post-split gene flow**, which parameters produce the **most similar values**?

3. Choose a small (i.e. <0.1) migration rate value and a large (i.e. > 10000) effective population size value. What is the expected F_{ST} at equilibrium? Recall:

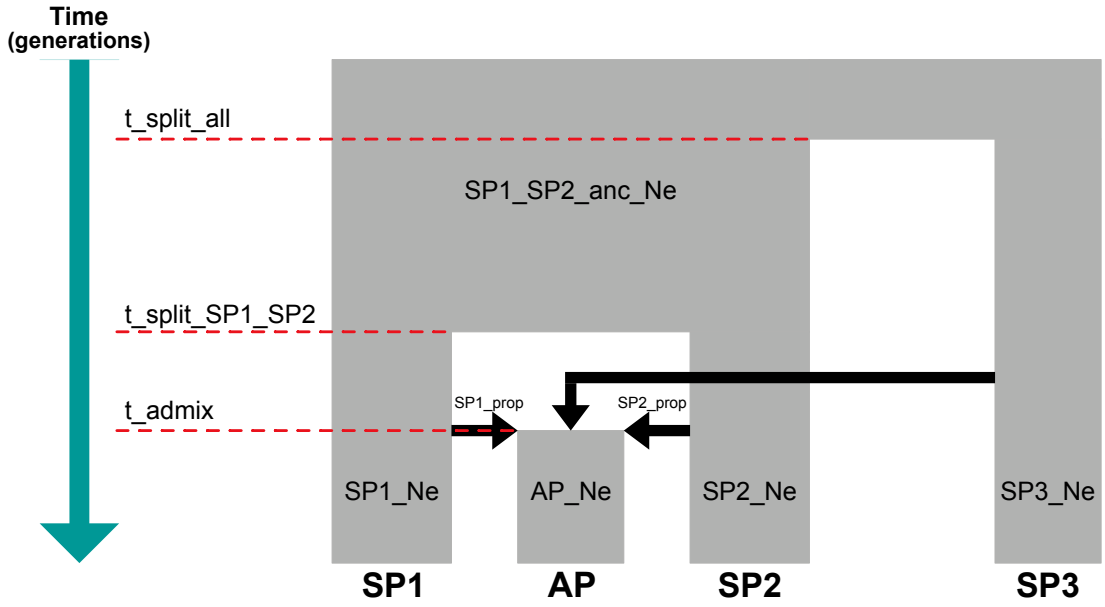
$$F_{ST} \sim \frac{1}{1 + 4N_e m}$$

Run the model with the values you chose and an old (>10000 generation) split time to allow the simulation to reach drift-migration equilibrium. Repeat your model and collect around 10 different F_{ST} values. How do these simulation values compare to your calculated expected value?

Exercise 4

In this exercise, you will explore a 3-way admixture model (i.e. 3 source populations, one admixed population). You will run ADMIXTURE on simulated data, plot your results, and calculate average ancestry proportions. **We will walk through the first 4 steps together, then move into breakout rooms for the rest.**

1. Take a look at the 3_pop_admix.py script – it defines a demographic model of 3-way admixture and a 30 Mb sequence of DNA evolving neutrally. The mutation rate and recombination rate are both 1 x 10^{-8} per base pair per generation. At the end of the simulation (t_0), 100 diploid individuals are sampled from each population and the script produces a .vcf file containing their genomic data. See diagram below:



   There is a lot going on in this model, and this script allows you to freely choose most of the parameter values. You can specify each populations' Ne, each population split time, the timing of admixture, and the ancestry proportions for the admixed population. However, first start by running the script with its defaults. This will print some output that will walk you through each of the parameter values. You can also launch the script with just the --help flag to get further explanation.

```
> python 3_pop_admix.py --help
> python 3_pop_admix.py
```

2. The simulation should complete in under a minute with the default parameters and will generate a file with a .vcf extension. By default, it is called '3_way_admix_output.vcf'. Now, launch the bash script called 'run_admixture.sh' giving it arguments for 1) your input file name and 2) an output name of your choice.

```
> sh run_admixture.sh [your input vcf] \
                      [output file name]
```

This script is doing a lot: some text processing, variant filtering, and then ADMIXTURE for Ks 2 through 5. This script should take no longer than 5 minutes to complete. It will generate .Q files, .P files, .out files, and a run_stats.txt file. Take a look at each type of file with 'less'.

```
> less output.3.Q
```

3. Open RStudio and load the 'plot_admixture.R' script. Start by setting your working directory to the one that contains your data and changing the 'outfile' variable to take the actual name of your file. Step through each section of the script and examine the plots.

4. Report the results of simulation using this Google Form.

5. Explore the effects of changing the parameters of this model. Pick one parameter to vary and choose at least 3 different values. It would be a good idea to coordinate with the other participants in your breakout room so that you can collectively experiment with a few different parameters. A few ideas are below, but feel free to explore whatever you like:

    - Change Ne of the admixed population (AP_Ne)
    - Change the time of admixture (t_admix)
    - Change one of the population split times (eg. t_split_SP1_SP2)
    - Make admixture proportions less even (SP1_prop and/or SP2_prop)

6. Before you run your simulation, predict how changing your parameter will affect the resulting admixture proportion estimates. For example, how well will ADMIXTURE be able to recover your simulated proportions? How will the ancestry standard deviations be affected?

7. Go through steps 1-3 of this exercise for each of your three values. Evaluate the hypothesis you made in step 6, and discuss with the others in your breakout room.

Exercise 5

In this exercise, you will run RFMix on simulated data you generated under a 3-way admixed scenario in the previous exercise. RFMix is a program that identifies local ancestry tracts in a genome of interest given reference data for the putative source populations.

1. Launch the run_RFMix.sh script specifying the .vcf output you want to work with (i.e. the file generated by the msprime simulation) and the specific individual (sample name) you want to analyze. You can choose any individual from AP_1 to AP_50.

   ```
   > sh run_RFMix.sh [your input file name] \
                     [output file name] \
                     [sample ID]
   ```

   This script starts from the msprime vcf file output and performs minor allele frequency filtering to exclude variants occur in less than 1% of the sample. This time, we are using vcftools in order to preserve phase information. For this analysis, we do not want to thin for linkage disequilibrium. Why not?

   The script then does some text processing and data filtering to create the files that RFMix needs. We need to tell the program which individuals comprise our reference panel, and which source populations they belong to. We also need to specify which focal population we want to estimate local ancestry for. Finally, we generate a 'dummy' recombination map with a constant rate so that RFMix can run. In real data, a real recombination map would tell RFMix if there are certain areas of the genome where recombination is more frequent (or not) and therefore where ancestry tracts are more (or less) likely to be 'broken'.

   Once RFMix has finished running, you will see that it has generated 4 different files. They end with the extensions 'msp.tsv', 'fb.tsv', 'sis.tsv', and 'rfmix.Q'. The 'rfmix.Q' file calculates the global ancestry proportions of this individual, and is directly comparable to ADMIXTURE's estimate. The 'msp.tsv' file contains the local ancestry calls for this individual.

2. Open RStudio and load the 'plot_RFMix.R' script. Start by setting your working directory to the one that contains your data and changing the 'infile' variable to take the actual name of your file. Run the script to see the local ancestry plot for this individual by genomic position.

3. Repeat the previous steps for 1-2 more individuals (or as many as you like) to see how individuals from the same population differ in their local ancestry patterns.