Python Programming - Assessed Exercise No.2

Issued: Friday 30 November 2018

Due: Monday 10 December 2018

There are 2 tasks to complete in this assessed exercise.

Background

The objective of this exercise is to write a script containing 2 functions to simulate genetic drift and the plot the results.

Genetic drift is the change in the frequency of an (gene variant) in a population over multiple generations. An individual's alleles are a sample of those in present their parents and chance has a role in determining whether that individual will survive and reproduce. Genetic drift may cause gene variants to disappear completely from a population and thereby reduce genetic variation. The exercise will simulate two simple genetic drift models.

Task 1

Write a function that creates a population of size 100, half of which have allele 'A' and half which have allele 'B'. This can be created as a lists with 50 'A' and 50 'B'. This list will represent the initial population.

The script should then complete 1000 generations using the following rules:

- With each generation an allele from the current population array should be randomly selected and added to a new array representing the new population.
- The random selection should occur the same number of times as the population size (100) so that the new and original populations are the same size.
- **Note that this means that some alleles may reproduce more than once and others may not reproduce at all.**
- If either of the alleles is completely lost from the population then no further generations should be completed.

The script also needs to produce a plot that shows the change in allele frequency with each generation, up to the 1000 generations or when either one was lost. The plot should be produced once, when all of the generations have been completed and display both alleles.

Ensure that the plot is labelled appropriately.

HINTS:

The **random** module can be imported and used to generate a random number. For example, the command `random.randint(1,5)` will generate a random number between 1 and 5.

To get you started the following code can be used to create the list of 50 'A' and 50 'B' is:

```
pop_array = []

for i in range(0, int(popsize/2)):
     pop_array.append('A')
for i in range(int(popsize/2), popsize):
     pop_array.append('B')
```

The `popsize` value is the population size, which is 100. As `popsize` is 100, `pop_array` will contain 50 A and 50 B.

Task 2

Write a second function that has 2 alleles for a gene 'A' present in each individual. The alleles are 'A' and 'a' and the individuals are either 'AA', 'Aa' or 'aa'.

The population size will again be 100.

The initial population has an even distribution of alleles so 25% are "AA", 50% "Aa" (25% 'Aa' and 25% 'aA') and 25% "aa".

An evolutionary event has occurred that means only 80% of 'aa' individuals will survive to maturity to breed, which needs to be accommodated in the script. However, the population size will remain static at 100 after each generation.

With each generation one allele from each random individual should be combined with one allele from another random individual to create a new population of 100. As 'aa' is only 80% successful it means that in the new population you should only include 80% of those created, so 1 in 5 of the aa you create should be rejected and a new individual created in its place. However, make sure that the final population size after each generation is still 100.

Run the simulation for a maximum of 500 generations or until allele "a" disappears from the population, at which point it should stop.

Finally, draw a plot of 'AA', 'Aa', and 'aa'.

IMPORTANT:

Your script should not require or use any command line arguments.
It should not open any files.
Both functions should be in the same script.
When the script is run it should produce 2 plots, one for each function.
The code must be commented.
Your code must run on the PCs in 310/311.
Marks will be deducted if any of the above are not followed.

What to Hand In

You must submit 1 Python code script as a plain text file or PDF via the turnitin Dropbox on Blackboard.

You also need to email your Python script to d.huntley@imperial.ac.uk for testing.

Both the email and file must be submitted by 10am the day of the deadline shown above.