Writing a basic genetic algorithm

1 Store your population

A simple way to store your population is to have a matrix. Each row of your matrix will represent an individual and each column will represent a parameter you are trying to fit. You can then add in an additional column which will correspond to the fitness score. You can initialise this to NULL, or 0.

For example the following would represent a population consisting of 5 individuals.

P	K	ϕ	fitness
3.2×10^7	1.34	0.243	0
5.72×10^7	0.87	4.87	0
9.81×10^6	1.48	6.10	0
3.56×10^7	0.32	3.89	0
2.38×10^7	1.12	1.73	0

2 Evaluate

Write a function called `evaluate` which takes in your population and outputs your population with the fitness score filled in.

For example your function should read in the population above, and output it as below. (Note these fitness scores are entirely made up).

P	K	ϕ	fitness
3.2×10^7	1.34	0.243	120.42
5.72×10^7	0.87	4.87	602.4
9.81×10^6	1.48	6.10	153.45
3.56×10^7	0.32	3.89	384.34
2.38×10^7	1.12	1.73	214.84

3 Tournament selection

This will guide you through implementing tournament selection. Write a function called `tournament`. It will take in the population, a number 'N_tournaments' which will tell it

how many tournaments to host and 'win_rate' (probability between 0 and 1) which gives the probability of the fitter individual winning the tournament.

To begin with, create an empty matrix called gene pool. This will have 'N_tournaments' rows and a column for each parameter and a column for the fitness. For each tournament, pick 2 individuals at random from the population. Generate a random number, if it is small than 'win_rate', add the fitter individual to the gene pool, otherwise add the less fit individual.

For example, your gene pool made by hosting 10 tournaments may looks like this:

P	K	ϕ	fitness
9.81×10^6	1.48	6.10	153.45
2.38×10^7	1.12	1.73	214.84
3.2×10^7	1.34	0.243	120.42
9.81×10^6	1.48	6.10	153.45
9.81×10^6	1.48	6.10	153.45
3.56×10^7	0.32	3.89	384.34
3.2×10^7	1.34	0.243	120.42
5.72×10^7	0.87	4.87	602.4
2.38×10^7	1.12	1.73	214.84
3.2×10^7	1.34	0.243	120.42

For a population of size N, you will want to have $2N$ tournaments because 2 individuals in the gene pool will produce a single offspring.

4 Crossover

We will implement uniform crossover. Write a function called `crossover` which reads in your gene pool.

Start by creating an empty matrix called new_population. This should have N rows, and a column for each parameter as well as a column for the fitness score.To produce a single offspring, take the first two individuals in the gene pool, which we will call the mother and the father. Build a new chromosome by take a gene from the mother with a probability of 0.5, otherwise take the gene from the father. Add this offspring to your new_population. Keep repeating this for every pair of parents in the gene pool.

Using the gene pool above, your new_population may look like this:

P	K	ϕ	fitness
2.38×10^7	1.48	1.73	0
9.81×10^6	1.34	6.10	0
9.81×10^6	0.32	6.10	0
5.72×10^7	1.34	0.243	0
3.2×10^7	1.34	1.73	0

5 Mutation

Here we will use gaussian mutation. Write a function called `mutate` which reads in the new_population and mutates an individual with a probability m. If an individual is chosen to be mutated, then pick a gene at random and add gaussian noise.

For example, mutation might look like this:

P	K	ϕ	fitness
2.38×10^7	1.48	1.73	0
9.81×10^6	1.34	5.23	0
9.81×10^6	0.32	6.10	0
5.72×10^7	1.57	0.243	0
5.28×10^7	1.34	1.73	0

6 Putting it all together

You will want to create an initial population within your designated search space. Then you will probably want to define a number of generations you want your genetic algorithm to run for. For each generation, you will want to do the following:

REPEAT for each generation

- evaluate the population
- perform selection to create your gene pool
- produce offspring from your gene pool using crossover to produce a new population
- perform mutation on your new population
- save your new population over your existing population