Calculating Expected Offspring

Purpose

The purpose of this application is to calculate the number of offspring in the next generation that possess a dominant, or recessive trait.

This is useful in modelling how the distribution of alleles changes over time, or how expressed traits change over time, among other applications.

How did I go about coming to a solution?

To come about a solution, I decided that the most time-efficient method of modelling this is to individually declare each allele configuration as there are only six possible combinations. For example, the variable 'x2' is linked to the allele configuration 'AA-Aa'; where the capital 'A' refers to the dominant allele, and the lowercase 'a' refers to the recessive allele.

The user can then choose to input their own integer value to represent the population for each allele configuration, or have the values randomly assigned using the *random* library that comes with Python. The number of offspring per pair is also asked, which determines how many offspring a single pair will have.

Using an inheritance table (like the one used in GCSE, and A-level biology), the percentage of offspring that express the dominant trait can be calculated for each configuration. For example, if the allele configuration is 'Aa-aa', there is a 50% chance that the offspring will express the dominant trait.

To calculate the population size of the dominant offspring, the following equation was used:

***Sum(Population of Xn x Probability of dominance) x Offspring per pair***

The sum goes up to a value of n=6 as only 6 allele configurations are possible for a monohybrid cross. This value is different for a dihybrid cross which has more possible configurations. The current population size for a particular allele mother-father configuration is entered, and the probability of an offspring being trait-dominant determines the population of dominant offspring in the subsequent generation. The sum is multiplied by the number of offspring per pair.

When larger numbers are at play, it can be difficult to understand how values relate to one another. This is made easier by data visualisation. To visualise the data, the matplotlib library for Python was selected. A bar chart was used due to the simplicity of the data, which does not span a length of time.

Future improvements

This project was a quick attempt at visualising how a population changes when there are different distributions of alleles among the parents. Therefore, a number of improvements can be made to improve this application.

Firstly, a list and loop could be incorporated to assign each allele configuration a value, and then later to process the results. This method would negate having to individually declare each variable which takes up space, and (although computationally irrelevant in the small sample sizes here) can induce inefficient computation due to the nature of Python's line-by-line reading mechanism.

In the real world, not all couples can have a pre-determined quantity of offspring. Therefore, the random operator can be used to select how many offspring a couple will have, from a range of 0 to a user-defined value. This would enable the application to be used for more holistic calculations where different variables can modify the probability with which a certain value is chosen; for example, if there were economic growth in a country, this would increase the probability that a couple will have more children due to financial burden no longer being a limiting factor. Therefore, the chance that more offspring are randomly selected is increased. In vice versa, if there is a poorer economy, it is more likely that the randomly chosen number is 0 because the parents cannot afford a child.

The design of this application was to be used for monohybrid crosses. As such, it presents the largest flaw in the solution. Although possible to adapt it for dihybrid crosses, it is a large hassle. A future solution to this problem would permit an easier method of assigning, storing, and processing of different allele combinations so that dihybrid crosses could be more easily run. The first step to achieving this goal is to separate the mother-father allele combinations as done here (AA-AA, mother-father).