How Genetic Distribution Changes Throughout Generations Using Monte Carlo Simulation

Sophie Wang

Abstract

The research purpose was to observe how genetic traits evolve over time. A program was written to analyze the evolution of genetic traits. It demonstrates that genetic distribution follows a binomial distribution and uses a binomial distribution to simulate what happens to genetic traits in general. The final part is running a trinomial distribution since blood types have 3 alleles (I^A, I^B, and i). The simulation shows that AB blood types will eventually disappear.

1 Motivation

When I was five, my mom told me I had type B blood. I was perplexed by what that meant, how I obtained it, and how the doctor knew. Then I went to school and started seeing peers writing with their left hand and felt the same level of curiosity. Now, as a 15-year-old, I think the best way to answer these questions is through my passion for programming. To see how the distribution of genetic traits evolves, I wanted to create a program to run a simulation. It will start off with an initial distribution, simulate many generations, and calculate the final distribution.

2 Literature Review

This literature review discusses three scholarly articles about probability in genetics. It evaluates the article by Offner which talks about the different blood types, Meyer and Birney's article which describes how genes are simulated, and Harrison's article which explains the Monte Carlo simulation. Blood type will be analyzed for this project. Monte Carlo simulation will be used to analyze how genotypes and alleles pass from one generation to the next generation. Uncovering this information is helpful in predicting genetic distribution changes from generation to generation.

The blood type article by Offner discusses four different blood types and what is included in a blood cell. It provides a very clear and detailed explanation of ABO blood types. The article explains that the oligosaccharides on the glycocalyx of the cell membrane of red blood cells are used to identify blood types (2015). All blood types include galactose, N-acetylglucosamine, and fucose, but they are distributed to different places throughout the blood cell. Offner mentions that "The A and B transferases are both proteins, and each is 354 amino acids long. The O gene is only 117 amino acids long compared to the A and B. They differ from each other by four amino acids. They are coded for by genes that differ by seven base pairs, four of which result in the observed amino acid changes" (2015). This discusses what the blood cell is made out of and how each type is different from the other. A locus, or distinct region of the chromosome, is something that exists on each chromosome and both A transferase and B transferase might be encoded by alleles at the locus. The A transferase is coded for by the A allele, the B transferase is coded for by the B allele, and the O allele does not code for anything. The article mentions that the blood type genes are located on chromosome number 9, which has roughly 1742 genes (2015). Every person has two number 9 chromosomes due to mitosis, so there will be two alleles for the blood types. There are in total three possible alleles: I^A, I^B, and i. This means that a person will

have two out of these three possible alleles. Some people have more than one blood type and have an advantage over other people. This will cause them to have immunity to several pathogens, which makes them more likely to survive a particular epidemic. This article gives a brief summary of what the blood types are made up of and how they are inherited, but it does not talk about the negative blood types which occur due to the Rh protein.

Meyer and Birney (2018) discuss a simulation tool developed for studying genotype-to-phenotype relationships. The authors first point out that the relationship between genotype and phenotype could be very complex, due to the fact that many phenotypes are influenced by more than one genetic locus and one locus can influence many phenotypes. Additionally, a multitude of environmental factors also influences phenotypes. Because of these complexities, researchers require sophisticated simulations of realistic genotype and phenotype structures. Previous simulation packages are either focused on genotype or phenotype evolution with a relatively simple setup. The new package built by the authors provides a flexible simulation of phenotypes with different genetic and non-genetic variance components. It is a framework focusing on the simulation of phenotypes with a particular emphasis on the complexity of both multiple phenotypes and multiple genetic loci, which is not provided by other multi-phenotype simulation software.

Harrison (2010) summarizes how the Monte Carlo simulation works and provides formulas to explain. It shows that Monte Carlo simulation can be used to study statistics. Sometimes it is difficult to derive a formula analytically to calculate the mean and standard deviation of a distribution. In such cases, one can use Monte Carlo simulation to get a numerical estimate of these statistics. More specifically, one can generate 1,000 sample points in the MC simulation according to a distribution, then calculate the mean and standard deviation of these 1,000 samples. Another way you can use Monte Carlo is to estimate π. Buffon's needle is "a method using repeated needle tosses onto a lined background to estimate π" (2010). I created a code based on this that uses the MC simulation. It showed that the more needles you threw, the closer it was to π showing that the simulation is effective. The article also mentions that there are many different types of distributions. The most common one is the normal distribution otherwise known as the Gaussian distribution. Another one is the Poisson distribution, which the article mentions on page 6. The article gives a brief summary of the Poisson distribution and its formula. The article describes the distribution as "a discrete distribution, with non-negative integer values" (2010). I also created a code for the Poisson distribution to potentially show how many kids a parent could have. These distributions are used to describe different scenarios or dynamics and are characterized by two parameters: mean and standard distribution. On page 6, the article provides the probability

density function, (PDF), for normal distribution, which is

$$\frac{1}{\sigma\sqrt{2\pi}}\ e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1}$$

where σ is the standard deviation and μ is the mean (2010). PDFs describe how likely one can find a random number x. For example, this PDF peaks at μ, which means μ is most likely to appear in a random draw. σ tells the dispersion of the drawl, which is the standard deviation or the square root of the variance. The bigger the σ, the more likely one may get a number x far from the mean. The area under the distribution curve equals 1, which means the total probability equals 1. One weakness of this article is that it does not provide examples of how to use Monte Carlo simulation and apply it to real-world examples.

The articles about blood types describe specific information on the trait and how the distribution of genotypes changes over time. They give some background knowledge about genetics and biology, and also show that mathematics could play an important role in such studies. The Monte Carlo simulation article shows a tool to study probability and statistics. It can be used to generate different paths for the evolution of specific traits like blood types. Understanding the evolution and genetic drift (evolution by random chance), as opposed to natural selection (survival of the fittest), is an important aspect of evolution.

3 Genetics

3.1 DNA

DNA plays an important role in genetics since genes are made up of DNA. DNA is made up of nucleotides, which are made up of a phosphate group, a deoxyribose sugar, and a nitrogenous base. The four nitrogenous bases found in DNA are adenine (A), thymine (T), cytosine (C), and guanine (G). A pairs with T and C pairs with G. The specific order of these bases makes up the genetic code of a human being. Nucleotides bond with each other and form a long chain of alternating deoxyribose and phosphate groups with nitrogenous bases sticking out on one side. A second chain with the bases that pair with the first chain and these two chains are held together by hydrogen bonds that pair the bases together. Then, the double chains of DNA twist and form a double helix shape. This is how DNA is formed. DNA replication is also very important. The DNA first unzips itself by breaking hydrogen bonds (which are weak) between the bases of the first chain and the second chain so there are two strands now. Then, for each strand, free-floating nucleotides attach themselves to the complementary base. In the end, there will be two strands of DNA that are identical to

each other. DNA is inside the nucleus of a cell and it is too big to go out so it provides a strand that serves as a template for mRNA (messenger RNA). The mRNA creates a strand complementary to the DNA strand and is small enough to leave the nucleus. The mRNA attaches to a ribosome. tRNA (transfer RNA) attaches specific amino acids based on the mRNA. Each codon (a group of 3 nitrogenous bases) codes for a specific type of amino acid. This eventually creates a long chain of amino acids which is protein.

3.2 Dominance

Dominance is when genes have an allele pair that has opposite effects that are together. Only one trait can show, so the one that is expressed is the dominant trait and the gene that is masked is the recessive allele. Dominant alleles are usually represented with a capital letter and recessive use lowercase letters. Intermediate inheritance is when certain traits do not show a dominant or recessive trait. One type of intermediate inheritance is codominance. Codominance is when they are both expressed.

3.3 Probability and Punnett Square

A punnett square can be used to determine the probability of an offspring obtaining a gene. It can also be used for calculating a specific gene occurring in an offspring and is mainly used for a few genes. There are two types of probabilities: empirical and theoretical. Empirical probability is when results are received from an experiment whereas theoretical is based on an assumption. The main two rules in probability are the sum rule and the product rule. The sum rule is when you add the probability of different outcomes to get the total probability of a condition occurring. The product rule is when you multiply the probability of different outcomes to get the total probability of two events occurring together. These rules can be applied to genetics. For example, if you have a dihybrid cross (a punnett square with two traits), you can use both rules based on what you are trying to solve as shown in Figure 2. If you have two dogs that are both BbCc and they mate, you can use the product rule to determine the probability of the offspring being BbCc. You can use a monohybrid cross to determine the probability of Bb occurring and a monohybrid cross to determine the probability of Cc occurring, as shown in Figure 1. These both have a probability of $\frac{1}{2}$, and $\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$, which means there is a $\frac{1}{4}$ or 25% chance of BbCc occurring. Furthermore, using the same example, if you want to calculate the probability of 2 offspring being BbCc, you take the probability of one offspring getting BbCc, and add it to itself because you are getting the same trait so the probability will stay the same. So the probability of 2 offspring getting BbCc is going to be $\frac{1}{2}$ because the probability of one offspring getting BbCc is $\frac{1}{4}$ and $\frac{1}{4} + \frac{1}{4}$

equals $\frac{2}{4}$, simplifying to $\frac{1}{2}$. See Figure 1 and Figure 2.

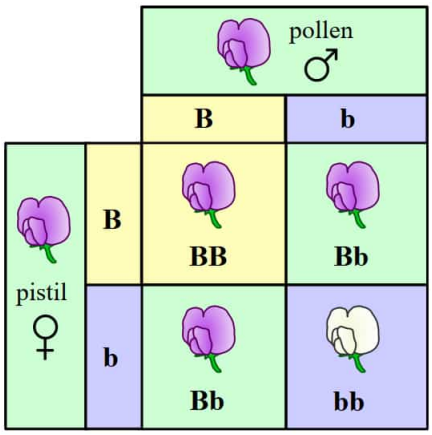

Figure 1: This is an example of a monohybrid cross of flowers where B is the dominant trait and b is the recessive trait. The B and b alleles represent what color the flower is, in this case, purple and white, respectively. If the trait has at least one dominant allele, it will show the color purple and if both of the alleles are recessive, it will show the color white (https://www.pngwing.com/en/free-png-cnmfv, 12-22-2022).

3.4 Blood Types

There are four different blood types: A, B, AB, and O. Each blood type is named after the antigens found on the blood cell. An antigen is a substance that causes the body to make an immune response against that substance. A has A antigens, B has B antigens, AB has A and B antigens, and O has neither. An Rh factor is on the surface of the red blood cell. The blood type is positive if it has an Rh factor and if it's negative, it doesn't have the Rh factor. A person with type A blood will either have an I^A, I^A genotype, or an $I^A i$ genotype. Type B blood would be $I^B I^B or I^B i$. AB would be $I^A I^B$. And O would only be ii. In this case, i is the recessive gene.

In summary, a person can receive blood that contains their own ABO antigens. In addition, Rh- people must receive from people with Rh- blood, while Rh+ can receive either Rh+ or Rh- blood.

The Rh factor is inherited from the RHD gene. A positive person could either be DD or Dd. A negative person can only be dd. The Rh-positive gene is dominant. If both parents are positive, the child could be positive or negative. If one parent is positive and the other parent is negative, the child could also be positive or negative. If both parents are negative, then the child has to be negative. This is because the parents can only be genotype dd so there is no way that you could have a dominant allele in your child's genotype.

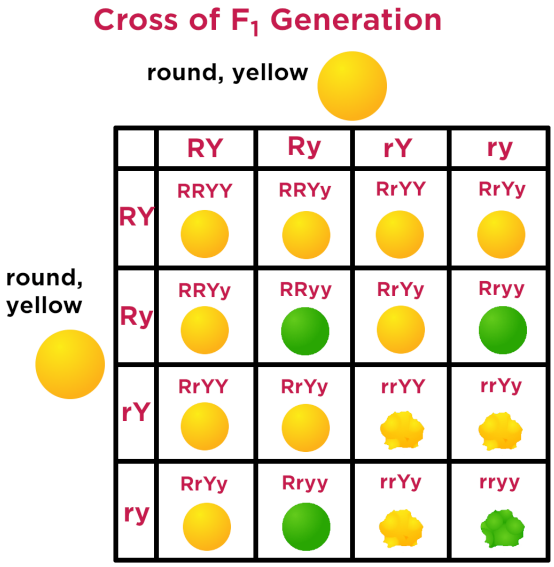

Figure 2: This is an example of a dihybrid cross of peas where R and Y are the dominant traits and r and y are the recessive traits. R means round and Y means yellow. r means wrinkled and y means green. The probability of the pea being round and yellow is $\frac{3}{4}$. The probability of the pea being green and round is $\frac{3}{16}$. The probability of the pea being yellow and wrinkled is also $\frac{3}{16}$. The probability of the pea being green and wrinkly is $\frac{1}{16}$ (https://www.expii.com/t/dihybrid-crosses-definition-examples-11020, 12-22-2022).

4 Simulating Genetic Trait Drift

To study how the genotype and phenotype distribution change over generations, I created a Python program to simulate the process and analyzed the results. The program contains three parts:

- verification that the binomial distribution property of evolution from one generation to another generation
- genotype distribution evolution over time
- blood type distribution evolution over a long time.

The three parts of my program all use Monte Carlo Simulation to study the distribution of genetic traits. This is because many of the distributions as a function of time do not have analytical solutions. It was programmed in Python because it has built-in methods such as binomial and trinomial distribution and the language can graph and plot points. Another advantage of using Python is that it is fast to run the program.

4.1 Binomial Distribution of Genotype Drift

The first part of the simulation program shows that when there are two alleles, say A and a, their frequency in the next generation follows a binomial distribution:

$$P(x) = \binom{2N}{x} p^x (1-p)^{2N-x} \tag{2}$$

where N is total population, p is the frequency of allele A in the current generation (e.g. 30% of all alleles are A), x is the number of alleles in the next generation and $P(x)$ is the probability of getting x number of allele A in the next generation. The number of allele A, x, can range from 0 to 2N. We have some observations of the equation:

- The probability should peak at $x = 2N \times p$. In other words, the next generation mostly likely also has the same number of allele A as the current generation.
- When $x = 0$, i.e. all alleles are a in the next generation, the probability is $(1-p)^{(2N)}$, which follows the product rule of probability.
- When $x = 2N$, i.e. all alleles are A in the next generation, the probability is $(p)^{(2N)}$, which also follows the product rule of probability.

I used an MC simulation to verify this hypothesis. An example is shown in Figure 3. If we assume the population is 20, then there are a total of 40 alleles. In one generation, 75% of them are allele A (i.e. 30) and the remaining 25% of them are allele a (i.e. 10). Therefore, for the next generation, we expect the most likely outcome is still about 30 A and 10 a. Due to randomness, it won't be exact and these numbers would follow some distribution. The bars in the charts are from the MC simulation, which assumes the child's alleles are randomly selected from two parents. The curve is an exact binomial distribution. They overlap each other and both peak around 30. This suggests binomial distribution is a reasonable assumption of child allele distribution.

More generally, if there are three alleles, like the ABO blood type, the child allele distribution would follow a trinomial distribution:

$$P(x,y) = \frac{(2N)!}{x!y!(2N-x-y)!} p_1^x p_2^y (1-p_1-p_2)^{2N-x-y} \tag{3}$$

For example, suppose there are N people in the population and the three alleles are called X, Y, and Z. In this equation, p_1, p_2, and $1-p_1-p_2$ represent the frequency of X, Y, and Z in the current generation. The probability of the child generation having x number of allele X and y number of allele Y follows this formula.

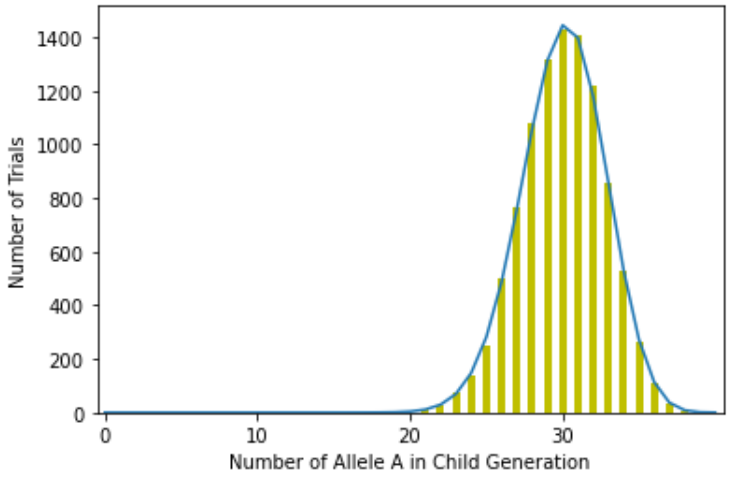

Figure 3: Distribution of Allele A in child generation. In this example, we assume the probability of getting an A is 75 percent in the parent generation, and there are 20 people in the population. The MC simulation generates 10000 scenarios. In each scenario, parents are randomly paired up to calculate how many A and a are present in the child generation. The bars represent the distribution of allele A in the child generation from the 10000 simulations. It is centered around 30. This is because when multiplying 75 percent by 20 (the number of people in the population) and then by two (each person has two alleles) we get 30. The curve represents an exact binomial distribution. The figure validates the binomial assumption because the bars overlap with the curve perfectly (Created by Student Researcher).

To validate that when there are two alleles, the distribution in the child generation is binomial, we use a Monte Carlo (MC) simulation programmed in Python. In this script, we first specify some parameters such as the following: we assume initially 75% of the total 40 (2×20) alleles in the population are allele A and the rest are allele a.

```
1    p=0.75 # probability of allele A (represented by 1)
2    N=20   # number of population
3    Nsim=10000 # number of simulation
```

Listing 1: This code section declares the variables of the probability, the number of people in the population, and how many times the program runs the simulation (Created by Student Researcher).

Next, we run the simulation 10,000 times. In each loop, we first randomly assign alleles A and a to the population. However, the total number of allele A should be 30 ($20 \times 2 \times 75\%$). (Note 1 represents allele A and 0 represents allele a).

```
def Binomial_Proof():
    distr=[0]*(N*2+1)
    for sim in range(Nsim) :
        # randomly generate current generation alleles
        s=0
        while (s!=2*N*p) :
           G1_0=np.random.uniform(size=N)
           G2_0=np.random.uniform(size=N)
           G1=[0 if x>p else 1 for x in G1_0]
           G2=[0 if x>p else 1 for x in G2_0]
           s=sum(G1)+sum(G2)
```

Listing 2: This is a function that first declares a distribution and then loops Nsim times. Each time it loops through, it generates a random gene allele and assigns it to the parent (Created by Student Researcher).

After this, still inside the for loop, we create a child generation and each child randomly comes from a pair of parents (represented by vectors $H1$ and $H2$). From each parent, one of the two alleles is given to the child (represented by vectors $J1$ and $J2$).

```
        # next generation, select parents
        H1=np.random.randint(low=0, high=N, size=N)
        H2=np.random.randint(low=0, high=N-1, size=N)
        # which allele to take from the 2 on the loci
        J1=np.random.randint(low=0,high=2,size=N) # parent 1
        J2=np.random.randint(low=0,high=2,size=N) # parent 2
```

Listing 3: Parents are randomly paired up by selecting a random number between 0 and N, and the next parent is selected between 0 to N-1. Then alleles are selected randomly from the parents that were chosen (Created by Student Researcher).

Now that we know the parent-child relationship, we can finally determine how many allele A's are in the child generation and record them in vector *distr* shown in Listing 4.

```
    # determine child alleles
    K1,K2=([] for i in range(2))
    for i in range(N) :
        [p1,p2]=[H1[i],H2[i]] # parent 1,2
        p2=p2 + (1 if p2>=p1 else 0)
        # allele from parent 1
        K1.append(G1[p1] if J1[i]==0 else G2[p1])
        # allele from parent 2
        K2.append(G1[p2] if J2[i]==0 else G2[p2])
    x=sum(K1) + sum(K2)  # Number of allele A in children
    distr[x]=distr[x]+1
```

Listing 4: Each parent that was chosen from Listing 3 is used for this part of the program. The alleles are taken from parent 1 and parent 2 which will then determine the genotype of the child (Created by Student Researcher).

We can also plot the empirical distribution with exact binomial distortion and they overlap very well as shown in Figure 3 and Listing 4.

```
b_distr=[]
for i in range(N*2+1) :
    b_distr.append(Nsim*math.factorial(N*2)/math.factorial(i)
        /math.factorial(2*N-i)*math.pow(p,i)*math.pow(1-p,2*N-i))
df=pd.DataFrame({'b_distr':b_distr,'emp':distr})
fig, ax1 = plt.subplots()
df['emp'].plot(kind='bar', color='y')
df['b_distr'].plot(kind='line')
ax1.set(xlabel='Number of Allele A in Child Generation',
          ylabel='Number of Trials')
ax1.set_xticks(range(0,2*N,10))
```

Listing 5: The for loop calculate the exact binomial distribution. The dataframe puts exact and empirical distributions of allele A together, and then we overlay them in the same chart (Created by Student Researcher).

With the knowledge that the distribution of genetic traits in the child generation follows a binomial (or multi-nomial) distribution, we can simulate the change over multiple generations. Figure 4 demonstrates 10 simulation paths with each one over a 500-generation

period. One can think of this as there are 10 isolated populations on 10 different islands. Each population has 50 people and hence 100 alleles. We assume initially 60% of the alleles are A on every path. This number evolves differently on different paths since the program is randomly generating numbers. However, in each case, one allele becomes dominant after a period of time. The quickest one takes about 40 generations and the longest one takes about 320 generations.

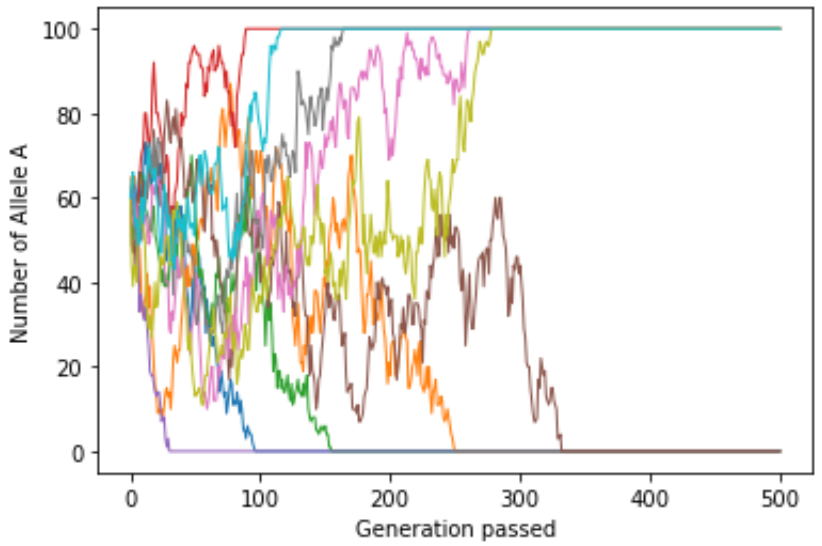

Figure 4: This figure shows the allele probability evolution through long periods of multiple generations for 10 different paths (each path can be thought of as a population on an isolated island). In the code below, all populations start from 60% of allele A. There are 100 alleles in the population because we safely assume there are 50 people and each person has two alleles. In this graph, the brown path takes about 380 generations to become only one allele survives (Created by Student Researcher).

Figure 4 was created by running the function drift_B, as shown in Listing 6. The function first declares the probability of having allele A at the beginning of time. We use 60 percent in this example, which is specified in variable *p0*. For each path, it goes through 500 generations, which is specified in variable *Ngen*. *M* represents the number of sample paths (think of isolated islands). The variable *g* saves how many allele A in each generation. A double loop is used to run the simulation. The first loop runs through generations. The second loop runs through the sample paths. Results are shown in Figure 4.

```
def drift_B() :
    p0=0.6
    N=100 # number of Alleles in populations
    M=10 # number of sample paths
    Ngen=500
    p=p0
    res=[]
    g=[int(N*p0)]*M
    res.append(g.copy())
    for t in range(Ngen) :
         for i in range(M) :
             p=float(res[t][i])/N
             p1=np.random.binomial(N,p,1)[0]
             g[i]=p1
         res.append(g.copy())
    # plot
    res_df=pd.DataFrame(res)
    ax=res_df.plot(legend=False,lw=1)
    ax.set(xlabel="Generation passed", ylabel="Number of Allele A")
```

Listing 6: A graph that follows a binomial distribution is plotted and the lines overlap perfectly, proving that the genotypes follow a binomial distribution (Created by Student Researcher).

4.2 Genotype to Phenotype Calculation

Once we understand how a genetic distribution changes from the above sections, we can calculate phenotype distributions and study how they evolve. Using blood type as an example, we show how to calculate phenotype probability from the genotype probability below.

As mentioned before, blood types consist of three alleles: I^A, I^B, and i. Different combinations of them produce different blood types of A, B, AB, and O. For example, if we assume alleles have the following frequency (i.e. genotype frequency), $P(I^A) = 20\%$, $P(I^B) = 20\%$, and $P(i) = 60\%$, then the blood type frequencies (i.e. phenotype frequencies) are:

- Type A: $P(I^A) \times P(I^A) + 2 \times P(I^A) \times P(i) = 28\%$
- Type B: $P(I^B) \times P(I^B) + 2 \times P(I^B) \times P(i) = 28\%$

Table 1: This table determines what blood type the person will have based on the specific genotype. If the person has at least one I^A allele, then the person will be type A unless the genotype is I^AI^B, then the person will be type AB. If the person has at least one I^B allele, then the person will be type B unless, like stated before, the genotype is I^AI^B (Created by Student Researcher).

	I^A	I^B	**i**
I^A	I^AI^A	I^AI^B	I^Ai
I^B	I^AI^B	I^BI^B	I^Bi
i	I^Ai	I^Bi	ii

- Type AB: $2 \times P(I^A) \times P(I^B) = 8\%$
- Type O: $P(i) \times P(i) = 36\%$

The probabilities should add up to 1, or 100%. To verify, we add the types up: $28\% + 28\% + 8\% + 36\% = 100\%$.

4.3 Phenotype Evolution

In this simulation, we assume all populations start from one-third of allele I^A, one-third of allele I^B, and one-third of allele i. This is because there are three alleles and we would like them to be equally distributed. Each population has 400 people. We choose 400 people since if the number is too big, it would take forever to simulate and if the number is too small, it wouldn't represent a population accurately. We simulate 100 paths of 500 generations by calling the function drift_T(400,100,500). We plot the average probability of each blood type of all the 100 paths. The results are shown in Figure 5. The results demonstrate that, on average, blood type AB will have fewer and fewer people, whereas blood type O will have more and more people.

This figure was created by using a function with three parameters: the number of people in the population, $Nppl$, the number of simulation paths (i.e. number of isolated islands), $Mpath$, and the number of generations, $Ngen$. There are three variables that represent the three alleles in blood types. The number of alleles would be two times the number of people since blood types are made up of combinations of two alleles.

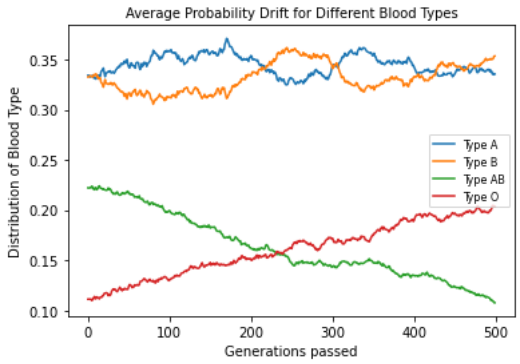

Figure 5: This figure shows the distribution of blood types over many generations. As shown in the graph, the AB blood type will start to decrease and the O blood type will increase. Blood types A and B will stay relatively steady (Created by Student Researcher).

Next, we need to figure out how the distribution of blood types develops over time. In this function, we first initialize variables as shown in Listing 7.

```
1 def drift_T(Nppl,Mpath,Ngen):
2     p0_A=0.333
3     p0_B=0.333
4     p0_i=1-p0_A-p0_B
5     N=2*Nppl # number of Alleles
6     res_A, res_B, res_i, ph_A, ph_B, ph_AB,ph_O=([] for i in range(7))
7     p_A=p0_A
8     p_B=p0_B
9     g_A=[int(N*p_A)]*Mpath
10    g_B=[int(N*p_B)]*Mpath
11    g_i=[N-g_A[i]-g_B[i] for i in range(len(g_A))]
12    res_A.append(g_A.copy())
13    res_B.append(g_B.copy())
14    res_i.append(g_i.copy())
```

Listing 7: Variables p0_A, p0_B, and p0_i are initial probabilities of the three alleles. They are assumed to be equally distributed. Three vectors res_A, res_B, and res_i are used to record genotype simulation history. Another three vectors ph_A, ph_B, and ph_O are used to record phenotype simulation history (Created by Student Researcher).

Then we go over generations and paths to simulate drifts based on the trinomial distribution as shown in Listing 8.

```
1   for t in range(Ngen) :
2       # simulation Phenotype distribution
3       for i in range(Mpath) :
4          p_A=float(res_A[t][i])/N
5          p_B=float(res_B[t][i])/N
6          [p1_A,p1_B,p1_i]=trinomialRand(N,p_A,p_B)
7          [g_A[i],g_B[i],g_i[i]]=[p1_A,p1_B,p1_i]
10      res_A.append(g_A.copy())
11      res_B.append(g_B.copy())
12      res_i.append(g_i.copy())
13      ph_A.append([(g_A[i]*g_A[i]+2*g_A[i]*g_i[i])/float(N*N)
14           for i in range(Mpath)])
15      ph_B.append([(g_B[i]*g_B[i]+2*g_B[i]*g_i[i])/float(N*N)
16            for i in range(Mpath)])
17      ph_AB.append([2*g_A[i]*g_B[i]/float(N*N)
18           for i in range(Mpath)])
19      ph_O.append([g_i[i]*g_i[i]/float(N*N) for i in range(Mpath)])
```

Listing 8: We use double loop to go over generations and simulation paths to determine genotype and phenotyp distribution. The outer loop goes over generations and the inner loop goes over different paths (Created by Student Researcher).

Once the simulation is finished, we calculate the average of the blood type among the simulation paths as shown in Listing 9.

```
1    # initialize dataframes
2    [ph_A_df,ph_B_df]=[pd.DataFrame(ph_A),pd.DataFrame(ph_B)]
3    [ph_AB_df,ph_O_df]=[pd.DataFrame(ph_AB),pd.DataFrame(ph_O)]
4    # calculate average of each blood type
5    ph_mean=pd.DataFrame({'Type A':ph_A_df.mean(axis=1),
6        'Type B':ph_B_df.mean(axis=1),'Type AB':ph_AB_df.mean(axis=1),
7        'Type O':ph_O_df.mean(axis=1)})
```

Listing 9: We first assign the results from vectors to dataframes. Each row represents one generation and each column represents a simulation path. And then we calculate average for every row (Created by Student Researcher).

Finally, we turn the results in the dataframe into plots. The results are shown in Figure 5

```
# making plot
fig, ax=plt.subplots()
ax.set_title(
     'Average Probability Drift for Different Blood Types',
    fontsize=10)
ph_mean.plot(ax=ax)
ax.set(xlabel="Generations passed",
       ylabel="Distribution of Blood Type")
plt.legend(loc='right', prop={'size': 8})
```

Listing 10: Blood type: turn results in dataframe into plots. (Created by Student Researcher)

5 Results

The drift of genetic traits from generation to generation follows a binomial (or multinomial in case of more than two alleles) distribution. This is demonstrated by running a Monte Carlo simulation and observing that the empirical distribution matches the theoretical binomial distribution perfectly. We can then apply this principle to a particular trait, such as blood type. As we know, blood type is determined by three alleles, and therefore, it is natural to assume the drift of their distribution follows a trinomial distribution. From this principle, we ran a multi-path and multi-generation Monte Carlo simulation to determine how the distribution of blood types would evolve after a long period of time. The simulation results show that after many generations, the average population of blood type AB would gradually decrease and the average population of blood type O would gradually increase and reach a plateau. The population of Blood types A and B stay relatively stable compared to the other two blood types.

To understand the speed of the blood type AB population decreasing, I experimented with the simulation with different population sizes. More specifically, I only plotted blood type AB frequency for different sizes of populations. The results show that for a smaller population size, the speed of decrease is faster than that of a larger population. This is illustrated in Figure 6, which is based on running the simulation with different population sizes: *drift_ T(Nppl=400, Mpath=500, Ngen=500)* and *drift_ T(Nppl=800, Mpath=500, Ngen=500)*.

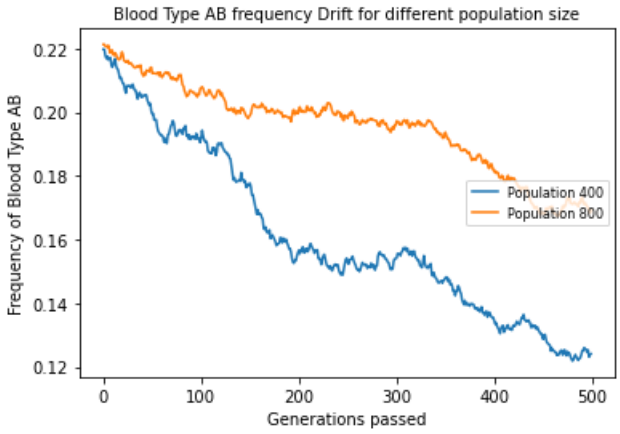

Figure 6: This figure shows the frequency drift of 2 different population sizes of the blood type AB. As shown in the graph, the population of 800 people decreased slower than the population of 400 people (Created by Student Researcher).

6 Discussion

My research shows that the answer to how genetic traits change over from generation to generation is that they follow a multinomial distribution based on how many alleles there are and that some traits may disappear or increase.

The results are congruent with current findings since today, the AB blood type is slowly dying out and O type people are slowly increasing. This is probably because only one of the three alleles I^A, I^B, or i will survive after a long period of time, as demonstrated in Figure 4. There is a small chance that both I^A and I^B will survive, making the AB blood type rare.

7 Conclusion

This research is important to society because people can have a better understanding of how the distribution of blood types changes over generations and how each specific blood type changes. It can be also helpful in animal breeding or vaccine making. For example, if an animal has three alleles but only one of the alleles is useful, the farmer may want to consider breeding it so that only one allele can survive. This project can estimate how long it would take for the other two alleles to disappear. In the future, I can try to simulate the genetic distribution of height since height is a continuous trait, which makes it hard to predict how it evolves. This is because height can be genetically and environmentally affected.

References

Harrison, R. L. (2010). Introduction to monte carlo simulation. In *Aip conference proceedings* (Vol. 1204, pp. 17–21).

Jelenkovic, A., Sund, R., Hur, Y.-M., Yokoyama, Y., Hjelmborg, J. v. B., Möller, S., ... others (2016). Genetic and environmental influences on height from infancy to early adulthood: An individual-based pooled analysis of 45 twin cohorts. *Scientific reports*, *6*(1), 1–13.

Meyer, H. V., & Birney, E. (2018). Phenotypesimulator: A comprehensive framework for simulating multi=trait, multi-locus genotype to phenotype relationship. *Bioinformatics*, *34*(17), 2951–2956.

Offner, S. (2015). Abo blood groups. *The American Biology Teacher*, *77*(8), 583–586.