**Computational Modeling of Allele Frequencies with the TI-84 Calculator**

## Introduction

In many fields of science mathematical modeling is an important complement to laboratory experiments. Mathematical models allow us to simulate the outcomes of experiments that would be difficult or impossible to perform in the lab or in the field. They can generate testable predictions of real-world phenomena that can guide the design of experiments. Good mathematical models symbolically represent true relationships between biological entities and gives us insight into the mechanisms that generate observable phenomena. As the cost of computer processing power goes down, the complexity of models and the sizes of datasets they can use is constantly increasing. However, simple models can still generate valuable insights. In this lesson we are going to create a simulation of allele frequencies in a population using principles from the Hardy-Weinberg model and your TI-84 graphing calculator.

The Hardy-Weinberg model was developed in the early 1900s to debunk the idea that dominant alleles would inevitably take over in a population. The model shows mathematically that under certain conditions, allele frequencies in a diploid, sexually reproducing population will remain constant over time. A population with constant allele frequencies is said to be in Hardy-Weinberg equilibrium. These conditions are highly idealized and never fully realized in nature. Nevertheless, understanding how and why a real population deviates from Hardy-Weinberg equilibrium gives insight into the evolutionary forces acting on it.

The conditions for Hardy-Weinberg equilibrium are:

1. No natural selection
2. Random mating within the population
3. No migration into or out of the population
4. No mutations
5. infinitely large population

The final condition is impossible to satisfy- no population is infinitely large! But there are important distinctions between how large and small populations behave based simply on the rules of probability. The simulations we will implement in this lesson will demonstrate how population size can impact allele frequency with important consequences for the population.

In the first set of questions you will determine whether your class is in Hardy-Weinberg equilibrium. Use the PTC taster genotype data you collected as part of A Taste of Genetics MiniLab (Cat # M6010). In the lab, we referred to the taster allele as T and the non-taster as t, and we continue this convention here. In this scheme, homozygous taster is TT, heterozygous taster is Tt, and tt is homozygous non-taster. $p$ will be the frequency of the T allele and $q$ the frequency of the t allele, and these two variables should always add up to 1 ($p + q = 1$).

Equations and variables will be displayed in *italics*. Snippets of code or other commands entered on the calculator will be in `fixed width font`.

**Part I: Is your class in Hardy-Weinberg equilibrium?**

For this exercise, use the PTC taster genotypes from your classmates that you determined with during A Taste of Genetics MiniLab. If your class did not do A Taste of Genetics, you can download **MiniOne-PTC-example-class-data.xlsx** from https://theminione.com/taste-of-genetics-extension-activities/

1. Record the number of individuals in your class with genotypes TT, Tt, and tt, and the total number of students in your class in the **Number** column of the **Data Table** below. If using the example data, count the students with each genotype from the spreadsheet and record in the **Number** column.

2. From the tabulated genotypes, calculate *p*, the frequency of the T allele. *p* will be a number between 1 and 0 that represents the fraction of total alleles that are T.

    a. Calculate the number of T alleles in your class. Remember that each student with Tt has one copy of the T allele and each student with TT has two copies of the T allele.

    b. Find the total number of alleles in your class by multiplying the total number of students in your class by 2.

    c. Divide the number of T alleles by the total number alleles to find *p*.

3. Calculate *q*, the frequency of the t allele, by subtracting *p* from 1.

    $q = 1 - p =$

4. Using your values for *p* and *q* and the terms of the equation $p^2 + 2pq + q^2 = 1$, calculate the frequency of each genotype you would expect under Hardy-Weinberg equilibrium. Record these numbers in the **Expected Frequency** column of the Data Table.

    $p^2 =$

    $2pq =$

    $q^2 =$

5. Calculate the expected number of each genotype from the expected frequencies you found in Question 5. Multiply the total number of students in your class by each expected frequency. Record the results in the **Expected Number** column of your data table.

6. We will use a **Chi Square Goodness of Fit** test to determine whether the observed numbers and expected numbers of each genotype are significantly different. To calculate statistical significance, we will need to decide on a level of certainty we will accept. In scientific studies, the level of certainty is often set at 0.05, meaning that there must be less than a 5% chance that the difference between observed and expected numbers is due to chance alone.

a. Calculate the squared difference between the observed and expected numbers for each genotype, **$(O-E)^2$**. Record on your data table.

b. Divide each squared difference by the expected number, **$(O-E)^2/E$**, for that genotype, and record on the data table.

c. Add up the numbers in the final column. This is your **Chi Square critical value**.

d. Before we compare our calculated Chi Square critical value to the table of critical values, we need to determine the degrees of freedom (*df*). Degrees of freedom is the number of categories minus the number of independently calculated values minus 1. In this case we have three categories and one independently calculated value (*p*), giving 1 degree of freedom.

e. Find the entry in the **Chi Square Distribution** table that corresponds to the degrees of freedom and desired probability. Is your critical value larger than this number?

7. Is your class in Hardy-Weinberg equilibrium? Justify your answer using the results of the Chi Square test.

| Data Table | | | | | |
|---|---|---|---|---|---|
| | **Observed Number (O)** | **Expected Frequency** | **Expected Number (E)** | **$(O-E)^2$** | **$(O-E)^2/E$** |
| TT | | | | | |
| Tt | | | | | |
| tt | | | | | |
| **Total** | | | | **Critical value** | |

| | **Chi Square Distribution** | | | | |
|---|---|---|---|---|---|
| | **Probability of exceeding the critical value** | | | | |
| **df** | **0.1** | **0.05** | **0.025** | **0.01** | **0.001** |
| **1** | 2.706 | 3.841 | 5.024 | 6.635 | 10.828 |
| **2** | 4.605 | 5.991 | 7.378 | 9.210 | 13.816 |
| **3** | 6.251 | 7.815 | 9.348 | 11.345 | 16.266 |
| **4** | 7.779 | 9.488 | 11.143 | 13.277 | 18.467 |

**Part II: Calculating allele frequencies in a population**

In the remaining part of this lesson, we are going to use computational modeling with your TI-84 calculator to simulate an experiment that could never be done in the lab. We are going to run your class through several generations to see how allele frequencies change over time. We will work up to the full simulation by running some simpler programs.

Before beginning to write the programs, take some time to think about what the inputs and outputs should be. Assuming random mating, how many parameters must be specified in order to calculate allele frequencies and genotype frequencies one generation in the future? What parameters would you have the user supply to a program to calculate allele frequencies and genotype frequencies one generation in the future? (For a hint, look at Questions 2-4 in Part I). What parameters would you have the user supply to a program to calculate the *number* of individuals in a population with a specific genotype?

Before beginning, we strongly suggest completing units 1-4 on the "10 Minutes of Code" so that you are familiar with the TI Basic programming environment.
https://education.ti.com/en/activities/ti-codes/84/10-minutes

If you do not have time to complete the modules, a cheat sheet for common commands can be found here: http://tibasicdev.wikidot.com/sk:cheat-sheet

Several useful tutorials are here, including directions for creating your first program:
http://tibasicdev.wikidot.com/starter-kit

1. Create a new program and name it `PQCALC`.

2. Write a program that does the following:

    a. Asks the user to enter the value of *p* using the `Prompt` command. Assign this value to a variable `P`. (see Unit 2, Skill Builder 1 on the "10 Minutes of Code" webpage).

    b. Create a new variable `Q` and assign a value to it based on the relationship $p + q = 1$. (See Unit 2, Skill Builder 3).

    c. Use the `Output` command to print `"Q ="` along with the value to `Q` to the home screen. (see Unit 2, Skill Builder 1).

    d. Don't forget to start your program with `ClrHome` (Unit 1, Skill Builder 2) and use the `Pause` statement to keep the program from ending too soon (Unit 1, Skill Builder 3).

3. Test your program `PQCALC` by entering the value of *p* you calculated in Question 1 in Part I. Does the returned *q* match the value you calculated in Question 2 in Part I? Tip: If your program cannot run because of an error in the code, you will be given an option to select `GoTo`, which will take you to the line that caused the error.

**Part III: Introducing randomness**

In our simulation we are going to need to keep track of multiple individuals in a population. To do this we will use a list to store multiple values. Lists store data in the form *X = [x1, x2, x3 ,…, xi]*, where each *xi* is a data point. In this exercise you will write a program to make a list of random numbers, one for each individual in the population.

1. Before writing the program, experiment with the `rand` command from the home screen. Find rand by pressing `math` then using the right arrow to select the `PROB` menu. What range of values is produced by this function? How many digits are displayed?

2. Create a new program called `RANDLIST`.

3. Write a program that does the following:

    a. Prompt the user to enter a value for `N`, the number of individuals in the population.

    b. Use the `rand` command to create an `N`-length list of random numbers. In a program, `rand` accepts one argument, the number of random values to generate, and returns a list of those numbers. You can read about the rand command here: http://tibasicdev.wikidot.com/rand.

    c. Assign your list to a variable name of your choosing. Naming lists in TI-Basic code is a little different than naming single elements. All list names must start with the little uppercase L character and cannot have more than five characters. Find the L character by pressing `2nd + list`, right arrow to the `OPS` menu. You can read more about lists here: http://tibasicdev.wikidot.com/sk:lists

    d. Use `Disp` or `Output` to print your list to the screen.

    e. Don't forget to use `ClrHome` in the appropriate locations in your code.

4. Test your program by running it and entering a few different values of *N*. How does the calculator handle displaying your random numbers when the list is too long to fit on the screen?

5. Optional: Use the TI Basic documentation to look up **for loops**. Use a for loop to display each entry of your list on a different line.

**Part IV: From random numbers to alleles and genotypes**

You will now write a program to simulate the production of gametes by the parental generation (Gen 0) that will combine to produce zygotes of the next generation (Gen 1). We will make the simplifying assumption that the number of individuals in the population does not change between generations- there is exactly one offspring for every parent. We will use the value of *p*, two lists of random numbers, an `If` statement, and a `For` loop to assign alleles to each gamete.

In Part III, you generated an *N*-length list with random numbers between 0 and 1. We will use these random numbers to introduce an element of chance into the passing of alleles to the next generation. If

you have a list of 100 numbers and $p = 0.5$, then you would predict that 50 of the random numbers would be less than $p$. However, because the numbers are random, you might not get exactly 50. Imagine flipping a coin 100 times. You would expect to get 50 heads out of 100 flips, but the actual number might not be exactly 50.

The same logic applies to gametes. If $p = 0.4$, out of a sample of 100 gametes you would expect 40 gametes with the T allele. Since there's an element of chance, like the coin flip, the number wouldn't be exactly 40.

Your will loop through two lists of random numbers and assign an allele identity to each one depending on whether the random number is less than $p$ or greater than $p$. If the number is less than p, we will call it a T allele and represent it with a 1, and if the number is greater than $p$, we will call it a t allele and represent it with a 0. These 1s and 0s can be kept in two additional lists. Two lists are necessary because two gametes come together to produce each zygote. Here's an example where $N = 10$ and $p = 0.4$ for Generation 0. Make sure you can follow this calculation and see why, due to the influence of the random numbers, $p$ for Gen 1 is different than $p$ for Gen 1.

```
Lrand1 = { 0.876, 0.984, 0.127, 0.723, 0.653, 0.125, 0.367, 0.757, 0.323, 0.476 }
Lrand2 = { 0.547, 0.398, 0.670, 0.236, 0.764, 0.078, 0.654, 0.342, 0.896, 0.963 }

Lgam1 =  {   0  ,   0  ,   1  ,   0  ,   0  ,   1  ,   1  ,   0  ,  1   ,   1   }
Lgam2 =  {   0  ,   1  ,   0  ,   1  ,   0  ,   1  ,   0  ,   1  ,  0   ,   0   }

zygotes = {  tt  ,  Tt  ,  Tt  ,  Tt  ,  tt  ,  TT  ,  Tt  ,  Tt  ,  Tt  ,  Tt   }

Calculate p for Gen 1:
Total allele count = 20
T allele count (Gen 1) = 9
p (Gen 1) = 9/20 = 0.45
```

Figure 1. Logic used to generate gametes and zygote genotypes based on lists of random numbers. $p$, the frequency of the T allele, is 0.4 for the parental generation, Gen 0. Lrand1 and Lrand2 are two lists of random numbers between 0 and 1. In the lists Lgam1 and Lgam2, a 0 (representing the t allele) is chosen if the number at the corresponding position in the random number list is greater than $p$ and a 1 (representing the T allele) is chosen if it is greater than $p$. The list of zygotes simulates the production of offspring from randomly selected gametes. Two 0s represents the tt genotype, a 0 and a 1 represent the Tt genotype, and two 1s represents the tt genotype. A demonstration of how to calculate $p$ for Gen 1 is given below the lists.

You will implement this logic in a program that will calculate a new $p$ for Gen 1, given $p$ for Gen 0 and the number of individuals in the population, $N$.

1. Create a new program by making a copy of RANDLIST and name it ALLELES. To do this, first create a new program called ALLELES, then press 2nd + rcl. rcl should appear at the bottom of the screen. Press prgm, arrow right to exec, select RANDLIST and press enter twice.

2. Modify the program ALLELES to do the following:

a. Prompt the user to enter a value for *p*.

b. Prompt the user to enter a value for *N*.

c. Create two lists of *N* random numbers and assign them different variable names.

d. Create two new *N*-length lists of zeros and give them unique list names. It will become apparent in later steps what you will do with these. The `Fill` command will come in handy here. Find `Fill` by pressing `2nd + list`, right arrow to the `OPS` menu. `Fill` takes two arguments, the number to fill the list with and the names of the list (Example: `Fill(0, L1)`. You can read more about the syntax here: http://tibasicdev.wikidot.com/fill).

e. Use a `For` loop to go through every element in your *N*-length lists. (`For` loops are covered in Unit 4, Skill Builder 1. To set or get the value of one element of a list, use the name of the list and the desired index as an argument. (Example: `L1(1)` for the value of `L 1` at position `1`. http://tibasicdev.wikidot.com/sk:lists).

f. Use a conditional statement inside your `For` loop to test whether each element of the random number lists is less than *p*. If the current element is less than *p*, change the value of the current element of the corresponding 0s list to a 1. Conditional statements are covered in Unit 3 and here: http://tibasicdev.wikidot.com/sk:conditionals.

g. Calculate *p*, store it as a new variable, and display the value to the user. The math is similar to what you did in Part 1, Question 1, and by now you have all of the tools to add this to your program. Feel free to use any commands that we have not discussed if you think it will improve your code.

h. Use `Pause` and `ClrHome` in the appropriate points of your code.

3. Test your code using a few different values of *N* and *p*. Enter integers for *N* and numbers between 0 and 1 for *p*

4. You will now use your program to simulate the dynamics of allele frequencies over multiple generations of random mating and offspring production.

   a. In the table below, enter the value of *p* you calculated in Part I, Question 1 for your class data for Gen 0 in both columns.

   b. For the first generation, run your program entering this value for *p* and *N* = 20 when prompted.

   a. Record the new value of *p* returned by the program in the first column for Gen 1. Round to three decimal places.

   c. Run the program again, entering *N* = 20 and the value of *p* from Gen 1. Record the result on the table for Gen 2.

d. Repeat for the remaining generations on the table.

e. Repeat this procedure using $N = 200$, again starting with the value of $p$ you calculated in Part I, Question 1.

| Gen(eration) | $p$ for $N = 20$ | $p$ for $N = 200$ |
|---|---|---|
| 0 | | |
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |
| 6 | | |
| 7 | | |
| 8 | | |
| 9 | | |
| 10 | | |

**Part V (optional challenge): Automated looping over generations**
As a bonus exercise, modify your code to loop over multiple generations saving you the trouble of having to input the previous value of $p$ for every generation. Number of generations should be another variable that the user is prompted to enter. Make sure the program displays the number of generations run, the final value of $p$, and the genotype frequencies.

**Part VI (optional challenge): Hardy-Weinberg equilibrium with chi-square test**
Another bonus exercise: write a program with your calculator to perform the allele frequency and chi square calculation you did in Part I. The program should take as input the number of each of the genotype in the population and return the Chi Square critical value.

**Analysis Questions:**

1. Which of the assumptions of Hardy-Weinberg equilibrium might not be met by your class sample?

2. What assumptions do you make in your computational model that are not consistent with the conditions of Hardy-Weinberg equilibrium?

3. What assumptions do you make in your computational model that are not valid for a wild population?

4. In your simulation, did you measure more genetic drift for the large (*N*=200) or the small (*N*=20) population? Support your answer by citing specific numbers from your data.

5. Scientists consider a species to be endangered if its population size falls below a certain threshold. Based on what you learned in this lab about population size, why do you think small populations are a special concern for conservationists?

6. What parameters would you change or add to your model to account for one genotype being favored by natural selection? How would you tell from the output of the model that natural selection is occurring?

For the remaining questions, we have run a similar set of simulations many times using a range of population sizes and summarized the results in a series of graphs.

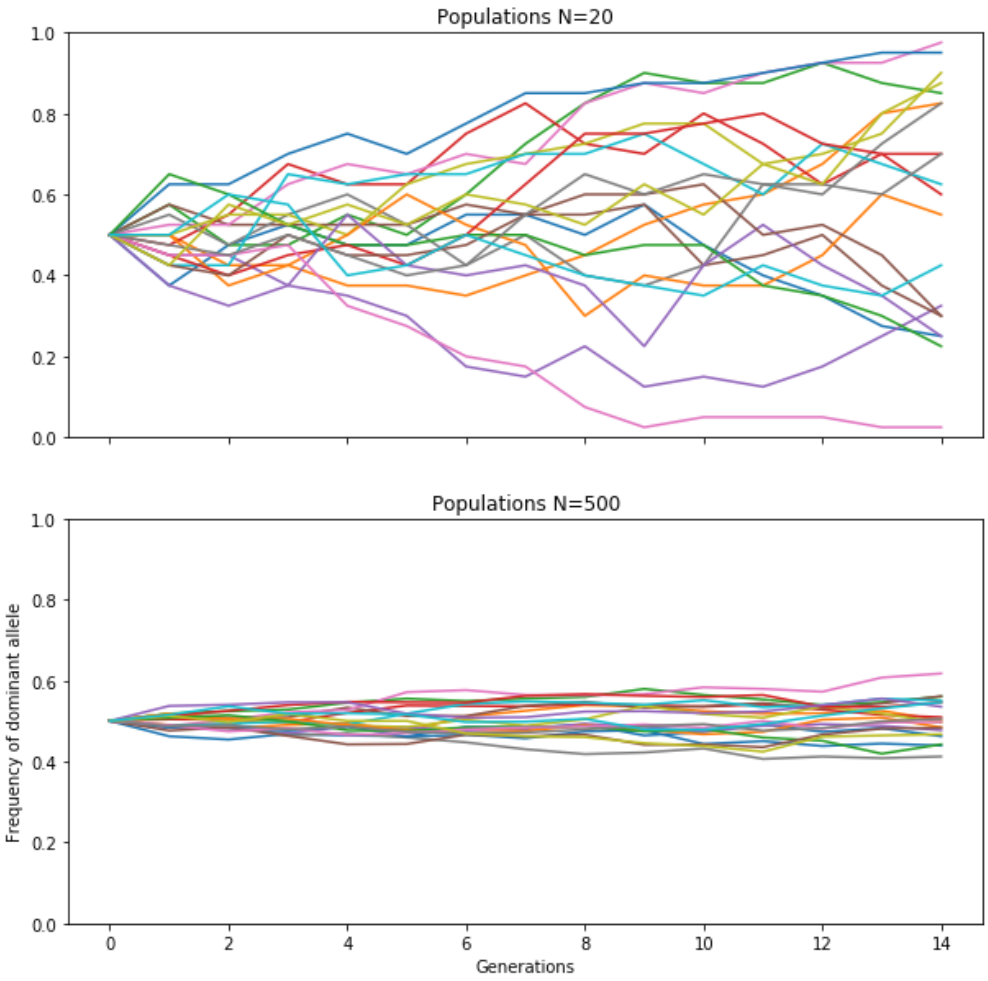


Figure 2. Change in allele frequency over 15 generations for 20 populations with either 20 individuals (top panel) or 500 individuals (bottom panel). X-axis is number of generations and Y-axis is frequency of the dominant allele, which we have been calling *p*. Each line represents a different simulated population.

7. Why are the lines more spread out in the top panel than the bottom panel of Figure 1? How this phenomenon related to the results of your own simulation?

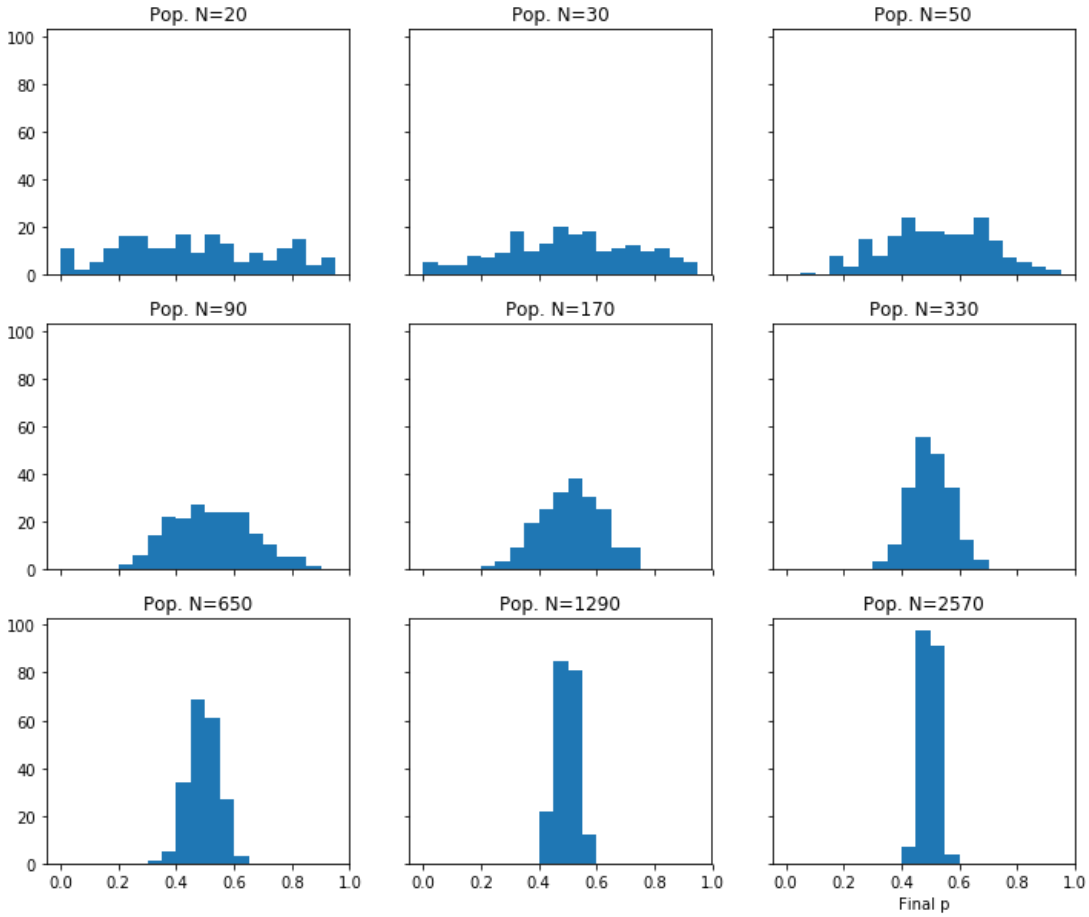


Figure 3. Each histogram represents the distribution of final *p*'s resulting from simulations of 200 populations after 15 generations. The population size (*N*) is listed above each panel. Each population started with $p = 0.5$ and the X-axis is the final value of *p*. The heights of the bars represent the number of final *p*'s within a 0.05 window.

8. In Figure 2, how do the shapes of the histograms change as *N* increases? State your answer in terms of *p* and *N*.

9. Why does a change in *N* cause a change in the distribution of the final value of *p* after 15 generations?

10. **Optional:** Write down the final value of *p* found by every group in the class for the simulations of both population sizes. Use the equation below to calculate the variance of the class data. What is the variance of the *N*=20 populations? What is the variance of the *N*=200 populations? Is one variance larger than the other? If so, provide an explanation based on your knowledge of Hardy-Weinberg theory.

    The equation for calculating variance is:

$$\sigma^2 = \frac{\sum (p_i - \mu)^2}{S}$$

    $\sigma^2$ = variance
    *S* = number of students in the class
    $p_i$ = the i-th value of *p* in the class
    $\mu$ = the mean of all value of *p*

To calculate the variance:

a. Record all the final values of *p* determined by every student in the class for the *N* = 20 case and calculate the mean.
b. Subtract the mean from each individual value.
c. Square each result.
d. Add the squares together.
e. Divide by the number of students in your class.
f. Repeat for *N* = 200.