MIE 124
Homework 6

Red-green colorblindness is a recessive trait linked to the X-chromosome. For this reason, the incidence of RG colorblindness in the male population is as high as 8%, and only 0.5% in the female population.

A quick (simplified) lesson in genetics:
Let's call the gene linked with determining color-blindness the "c-gene."

The c-gene has two variations (called alleles): "c," the allele responsible for colorblindness, and "C," the allele which results in normal vision.

As this gene resides on the X-chromosome, females have two alleles (they have two X-chromosomes) and males have only one (they have one X-chromosome, one Y-chromosome).

The "C" allele is dominant, while the "c" allele is recessive. This means that in order for a given person to present with color-blindness, they must have only "c" alleles. For males, this means that only one "c" allele is necessary, whereas females must possess two "c" alleles.

Generally, parents pass down one allele **each**, providing their offspring with a total of two alleles for each gene. For the current example, this rule reigns true for female offspring, but is slightly different for males.

Since a male by definition must inherit the Y-chromosome from his father, this means that his X-chromosome, and thus singular "c/C" allele comes from his mother.

For this homework you will turn in both a **function** and a **script**.

Your Function
The function will have two inputs: population size (*pop_s*), and number of generations (*n_gen*). The four outputs of the function will be **vectors** describing:

(1) percentage of the **total** population with R-G colorblindness (*total_percent*),
(2) percentage of the **female** population with R-G colorblindness (*female_percent*),
(3) percentage of the **male** population with R-G colorblindness (*male_percent*), and
(4) population **size** (*pop_final*).

Each of these outputs will have length *n_gen* and will track the percentages and population size within each generation.

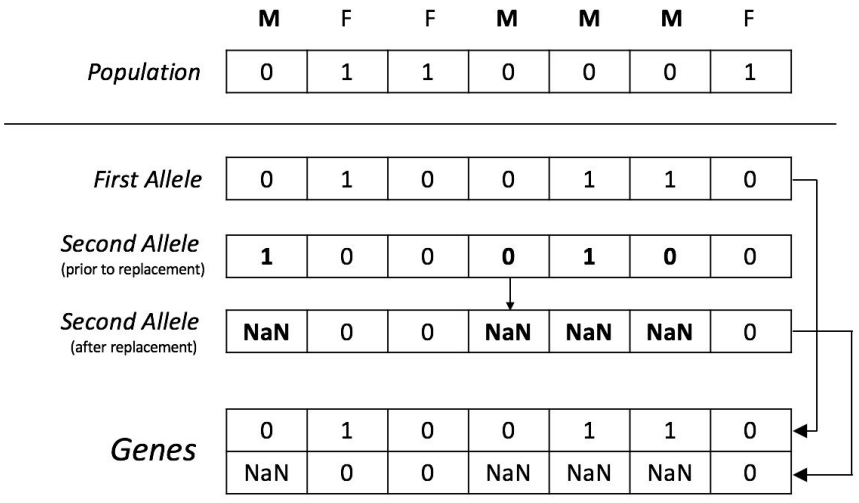
Your objective will be to generate a random population of males and females, each with randomly assigned alleles. You will then randomly pair off males and females as evenly as possible, generate a

number of offspring that each pair will have, and assign alleles to these offspring based on the parent alleles. You will repeat this consecutively (using the new pool of children this time instead of your parent generation) until you have created as many generations as n_gen. Let's get started.

1 - Start by generating a row vector of length pop_s containing evenly distributed random ones and zeros (where a '1' represents a female and a '0' represents a male). Call this vector *population*.

2 - Next, you'll want to generate a random vector of ones and zeros the same length as *population* to represent each person's first allele ('0's will signify the recessive allele and '1's will signify the dominant allele). **HOWEVER**, the recessive allele is less common than the dominant allele (there is an 80% chance that a given person has the dominant allele and thus a 20% chance that they have a recessive allele). Use a method of your choosing to ensure the probability of having a 1 is 80% and the probability of having a 0 is 20%.
You will create two of these vectors, one for the first allele and one for the second. Keep in mind that males will only have one allele while females will have two - for this reason you must eliminate the second value of the randomly generated alleles in every position that corresponds to a **male** (use the *population* vector to determine whether or not the corresponding person is male or female). We suggest denoting the absence of the second allele in males by the designation "NaN" which stands for "Not a Number" (MATLAB help pages can explain this well if you're curious). Make sure that this absence of the second allele only occurs for males, females will ALWAYS have 2 alleles.

3 - From here, you should concatenate (combine) the two allele vectors you have just created. This will result in one master matrix called "*Genes*" of each allele pair for each individual in the population. This matrix has two rows and the number of columns is the length of *population*. Below is a graphic for clarification.



4 - Now you must create a sub-function called "findPercents" with two inputs: the master allele matrix *Genes*, and the *population* vector (keep in mind that both will change from generation to generation; they won't be the same matrix as the generation before). The outputs of this function will be

"*male_percent_CB*," "*female_percent_CB*" and "*overall_percent_CB*," which represent the percentage of the male, female, and total population affected with colorblindness respectively. Note that because NaN is not a traditional number, you cannot just use "sum" to find the sum of each column, as it will result in the wrong answer. Consequently you will have to figure out a way to bypass this. Recall that if the sum of a given column of the alleles matrix is zero, that person is colorblind. Additionally, you will need to find a way to split the allele pairs into groups of male and female in order to determine the incidence of colorblindness within each sex (the "find" function may come in handy here).

5 - Back within your main function, call this sub-function for your starter population (generation 1) and update the first index of your output vectors.

6 - Your next task is to pair off males and females as evenly as possible, generate a random number of offspring that they will produce, and assign an allele pair to each of these children according to the biological laws of inheritance. You will accomplish all of this **within a for-loop**, which will run n_gen-1 times (once for each new generation).

> **a** - To pair off the males and females, you must first split the two genders into different allele matrices. These are temporary matrices, each with three rows and number of columns equal to the number of females and males. Initialize them as all zeros.
> **b** - From here, use the function "randperm()" to assign each individual a random number, the input of your randperm function being the number of females or the number of males depending on the corresponding matrix. Put these values into the first row of each matrix.
> **c** - When you finish this, make the last two rows correspond to the allele pair for each individual. For example, if you have a total population of 13 consisting of seven males and six females, your two matrices may look something like this:

Ordered Male Population

Randomly Assigned #	**1**	**2**	**3**	**4**	**5**	**6**	**7**
First Allele	1	1	0	1	0	0	0
Second Allele	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Ordered Female Population

Randomly Assigned #	**1**	**2**	**3**	**4**	**5**	**6**
First Allele	0	1	1	0	0	1
Second Allele	1	0	1	0	0	0

Keep in mind, you can make as many subfunctions as you want/find necessary. Make sure they are formatted and called correctly and are placed after the main function.

d - Once you have split up the population into male and female matrices, their numerical orders are still completely random so you need to sort them. To do this you will want to use the "sortrows()" function (look in the help section to ensure that you know exactly what sortrows() does) and sort the females and males in terms of their randomly assigned number (the first row).

e - After you have these matrices sorted in ascending order, you need to pair off the corresponding males and females in order to create a new generation of offspring. Before you begin creating these new children, create two empty matrices called *new_pop* and *new_Genes* which will expand each time you generate a new person. These will have one row and two rows respectively, with *new_pop* being similar to your original *population* vector, and *new_genes* being similar to your original *Genes* matrix.

f - Next, use a while loop which runs until all pairs have "mated," (it will run the same number of times as the length of your shorter temporary matrix). For instance, continuing from the example above, your loop would run a total of 6 times, as there are less females than males. At the beginning of this loop define your current male-female pair (for the first iteration it would be male 1 and female 1, for the second it would be male 2 and female 2, etcetera). Then, generate a random **integer between 0 and 5** which will represent the number of offspring this pair will produce. Based on this number (which will be different for each pair), you will run a for-loop which first generates a random gender for the current child using the same number/gender designation as before. Then you will assign this child an appropriate allele/allele pair based on gender and the genome of the parents. (Remember that males must inherit their allele from their mother, while females inherit one allele from each parent). Make sure to update your *new_pop* and *new_Genes* matrices throughout this process.

7 - When you finish these tasks you should **replace** your old *population* and *Genes* variables for generation 1 with *new_pop* and *new_Genes*. Additionally, you should update your *total_percent*, *female_percen*t, *male_percent*, and *pop_size* vectors with the new percentages and size of this population.

8 - Remember that you will repeat this sequence (steps **6**-**8**) for each generation (keep in mind that the first generation is generation 1, so you will actually be creating n_gen-1 new generations). Additionally, you may find it helpful to accomplish the above process using a sub-function. This way, you only need to call the sub-function, replace your *population* and *Genes* vectors, and update your outputs within your exterior loop (the for-loop mentioned in part **6**). When you finish, your four output vectors should be of length n_gen and have the appropriate values for the corresponding generation.

<u>The Script</u>

1 - For the script, make sure to put your name, ID#, and descriptive comment at the top. Next, call your function three separate times, with inputs of **(1000,20)**, **(10000,15**), and **(100000,10)**. You should have 4 outputs for each: *total_percent*, *female_percent*, *male_percent*, and *pop_size*. You are going to graph all of this information on a total of two graphs.

a - On your first graph you will plot all of your percent vectors for all trials. All of your *total_percent* lines should be **solid**, all of your *female_percent* lines should be **dashed**, and all of your *male_percent* lines should be **dotted**. Additionally, all lines corresponding to your **(1000,20)**

trial should be plotted in **green**, all corresponding to your **(10000,15)** should be **red**, and all corresponding to your **(100000,10)** should be **blue**.
b - On your second graph, plot all of your *pop_size* vectors. Make sure to use the same color convention as above.

In both cases your X-axis will represent the generation number - use the length of the output vectors to determine the x-axis spacing. Make sure to call up a new figure for each graph, label your axis, and title your graph with something relevant. Finally, comment on your results at the end of your script. Does the overall population increase, decrease, or stay just about the same? What happens to the incidence level of colorblindness for males, does it go up or down? For females? Does this make sense based on what you know about the laws of inheritance?

2 - At the very bottom of your script, feel free to comment on what you liked/ didn't like about this assignment. What did you find interesting and what did you find boring? Where did you get stuck the most (if you did get stuck)? And anything else you would like to mention.

**These are pretty big inputs so it is expected that your code should take a minute or so to load. However, if your code is taking more than a few minutes you should try to improve the efficiency of your program. (In the case that your code is taking a long time to run, you can use control+c to end manually)