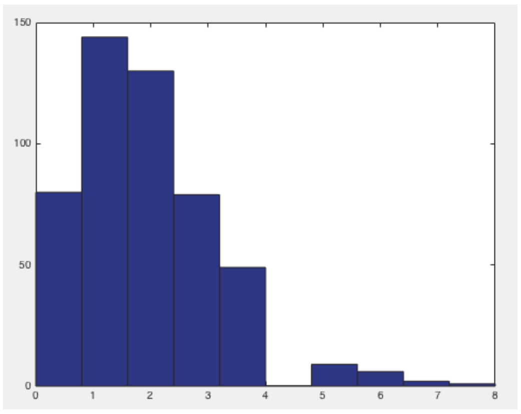
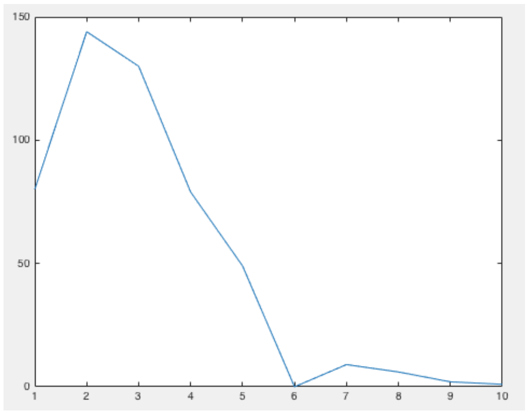
BIOL 8814 HW 1
Hong Seo Lim

PROBLEM 1: SIMULATION OF ONE GENERATION

```
C = 500;
N = 1000;
mu = 1*10^-3;

for i = 1:C
    i;
    ct = 0;
    for j = 1:N
        for k = 1:2
            ct = ct + 1;
            ALREADYONE = 0;
            rn = rand;
            if ALREADYONE == 1
                continue
            end

            if rn < mu
                state = 1;
                ALREADYONE = 1;
            else
                state = 0;
            end
            STATE(ct) = state;
        end
    end
    SUM = sum(STATE);
    SIMUL(i) = SUM;
end
figure
hist(SIMUL)

No_Zeros = sum(SIMUL == 0)
lambda = whatislambda(SIMUL) %labmda = mu * 2 * N (after dividing)
```

Above is the program I wrote. The distribution of resistant mutants that you observed across all 500 cultures can be visualized via histogram. Below is the diagram of the histogram of one simulation. Notice that the distribution is different for each simulation (running the above codes is a simulation), because of the stochasticity of the codes stemmed from *rand* function.





Here y-axis represents the number of cultures, and x-axis represents the number of mutants. As seen from the histogram, they are mostly similar, meaning variance is moderate (Var = 1.9936). If we specify our measurement of c(m), which is exactly visualized by the histogram and the continuous plot next to it, we can see that the distribution looks similar to Poisson distribution. In order to find the parameter lambda of Poisson distribution to fit the data, I used *whatislambda* function that I created last week's lab. Below is the codes for the function.

```
function lambda = whatislambda(X)
poissdata = X;
numberofzeros = nnz(poissdata == 0);
probzero = numberofzeros / (numel(poissdata));
if probzero == 0
    probzero = 0.01;
end
lambda = -log(probzero);

end
```

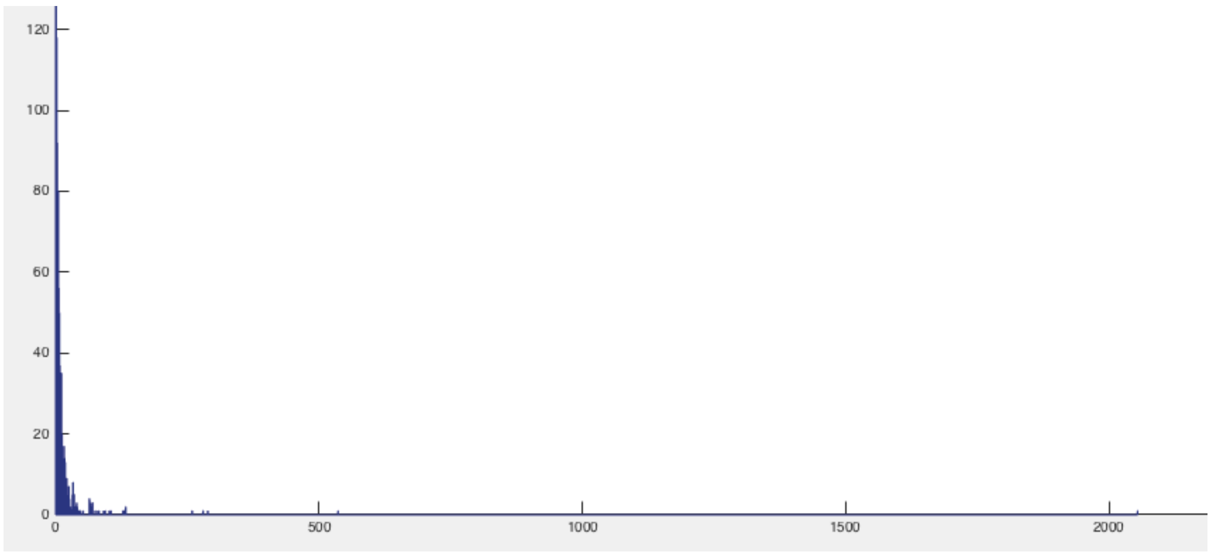
The function utilizes the fact that that the -log(probability of zero) = lambda. Upon using the function, the output was 1.8326, which is the lambda. This is actually works well because when we calculate mean and the variance of the simulation, they were 1.88 and 1.9936 respectively, which are very close to the estimated Poisson lambda. Poisson distribution is approximately close to the distribution of binomial distribution when n is large and p is small. Here our p is very small and n is relatively large. After one generation, our population will double, so N becomes 2000 from 1000. Since our p is 1*10^-3, 2000*p should give you 2. That is the lambda value of Poisson distribution, and since my codes are based on binomial distribution scheme, we can say all of those are related. I would not say the simulation fluctuations are large. If we calculate the index of dispersion, i.e. variance/mean, it is 1.0604. Therefore, fluctuations are not huge for the simulation.

PROBLEM 2: SIMULATION OVER 15 GENERATIONS

Basically, my program for problem 2 is the extensions of program for part 1. We only simulated one generation for part 1, I created a for loop that can simulate 15 generations of that. Of course, for each generation, our population size doubles, so at the end of 15 generations, I have 400*2*15 = 13107200 cells for one sample. We have 1000 independent samples. At each generation, for each cell, a random number is generated using rand function, if the random number is less than mu (10^-7), that cell becomes resistant, and its children cells will also be resistant. I could have model the emergence of resistance in each culture en masse using Poisson distribution. In problem 1, I showed that emergence of resistant cell can be well approximated by Poisson distribution. In order to use Poisson to simulate this, for each generation count how many cells are susceptible, and let's call that number m. Next generation, you will have 2*m susceptible cells. With similar logic from problem 1, the lambda of Poission can be estimated as 2*m*mu. Then you can generate a random number using the lambda of Poisson distribution by using *poissrnd(2*m*mu).* Whatever number you have for that function, those are number of cells that are now resistant. And repeat this process for 15 generations for all samples while counting number of resistant/susceptible cells. Although I could have done this, but I didn't because the first approach sounds more intuitive even though it takes more times to compute. Below is the codes for Problem 2 and Problem 3.

```
C = 1000;
N = 400;
mu = 1*10^-7;
i;
C=1000;
for i = 1:C
    i
    Original_STATE = zeros(1,400);
    age_ct = 0;
    for g = 1:15
        ct = 0;
        for j = 1:length(Original_STATE)
            cell = Original_STATE(j);
            if cell == 0
                for k = 1:2
                    ct = ct + 1;
                    ALREADYONE = 0;
                    rn = rand;
                    if ALREADYONE == 1
                        continue
                    end

                    if rn < mu
                        state = 1;
                        ALREADYONE = 1;
                    else
                        state = 0;
                    end
                    STATE(ct) = state;
                end
            elseif cell == 1
                for q = 1:2
                    ct = ct + 1;
                    state = 1;
                    STATE(ct) = state;
                end
            end
        end
        Original_STATE = STATE;
        ct;
        if sum(STATE) > 0 && age_ct ==0
            age = g;
            age_ct = age_ct+1;
        end
    end
    SUM = sum(Original_STATE);
    SIMUL(i) = SUM;
    AGE(i) = age;
    clear STATE
    clear Original_STATE
end
SIMUL
AGE

%save('prob2_variable')
```

In one sample (287th), number of mutants was 16385. Because of this value, histogram and consequent plots do not look clear, so I decided to consider that sample as an outlier. C(m) generated from above codes is following:



Notice from the above histogram that most of the samples are located at relatively low values, but we see some remarkable 'jackpots'. There were 13 samples that number of mutants are greater than 100. Another way to define a principled way to identify "jackpot" is by order them using sort function, and let the top 5 percentile to be considered "jackpot". Mean of the above data 12.9509, and its variance is whopping 4923.8. The index of dispersion is 381.5119. This huge variance proves does not support the "acquired" hypothesis, because we are supposed to see theoretically same mean and variance for the acquired hypothesis as we discussed in the class. Here, the huge variance, however, could support the spontaneous mutations. I disregarded top 5% (in terms of number of mutants) data points and checked how variance behaves. The variance decreased from 4923.8 to 49.23, a 100 fold change. Below is the code for this calculation.

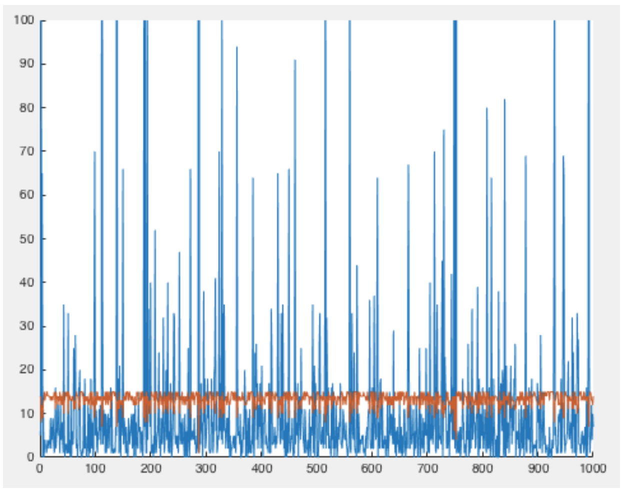
```
DATA = SIMUL;
DATA(287) = [];
n_D = length(DATA)

hist(DATA,1:2054)

DATA_sorted = sort(DATA)
no_jp_DATA = DATA_sorted(1:round(n_D*0.95))
var(no_jp_DATA)
```

PROBLEM3: AGE of MUTANTS

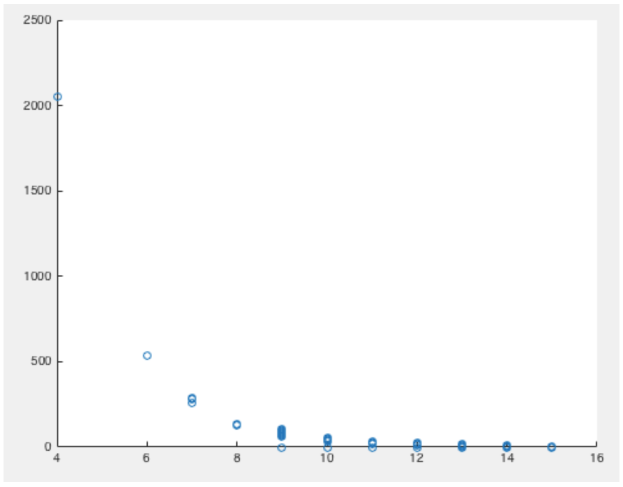
In order to know correlation between "age" of mutants and total number at the end, for each sample, I recorded the first time (generation, g) we see a mutant. The below is code is an excerpt from the codes from Problem 2.

```
        Original_STATE = STATE;
        ct;
        if sum(STATE) > 0 && age_ct ==0
            age = g;
            age_ct = age_ct+1;
        end
```

Below plot shows each sample's age(when it first has a mutant) and total number at the end. Because we have some huge numbers, y-axis goes from 0 – 100 to see the effects of age on the number at the end.



Here red line is the age, and the blue line is the end number. Although it is hard to see, whenever you have lower age, it seems to create "jackpots". To be more precise, I created a scatter plot(below) for this data. As you can see, as the age decreases, the end number increases exponentially. This clearly shows that the time for first appearance of mutant is very important in determining the end number. So early mutants increases the number exponentially, while late mutants' effect is mere. This discrepancy causes the huge variance we see in our data. If, by a chance, mutant appears early, that would cause the "jackpots" we talked about, and the existence of jackpots is the reason for the huge variance we are seeing.



```
%% PROBLEM3 AGE
clear
clc
load prob2_variable.mat
DATA = SIMUL;
DATA(287) = [];
age_DATA = AGE;
age_DATA(287) = [];
figure
hold on
plot(DATA)
plot(age_DATA)
ylim([0,100])
hold off
```

```
figure
scatter(age_DATA, DATA)
```