Project 1

Emmanuel Ogunjirin, Kieran Humphreys

Probability

APMA 3100 - Fall 2020

October 23rd, 2020

Introduction

For this project, a problem statement was provided as context. The problem stated that a representative of a high-speed internet provider calls customers to assess their satisfaction with the internet service. It takes 6 seconds for the provider to turn on a phone and dial a number; then 3 additional seconds to detect a busy signal, or 25 additional seconds to wait for 5 rings and conclude that no one will answer; and 1 second to end a call. After an unsuccessful call, they re-dial (over the course of several days) until the customer answers or they have dialed four times. The outcome of each dialing is determined in an identical way: the customer being called is using the line with probability 0.2; is unavailable to answer the call with probability 0.3; or is available with probability 0.5 and can answer the call within a given amount of seconds we denote as "*X*". *X* is a continuous random variable with a mean of 12 seconds in the exponential distribution.

Tree Diagram

To solve this problem, we needed to solve for the total amount of time spent by the representative on calling one customer. We denoted this time with "*W*". To solve for "*W*", a

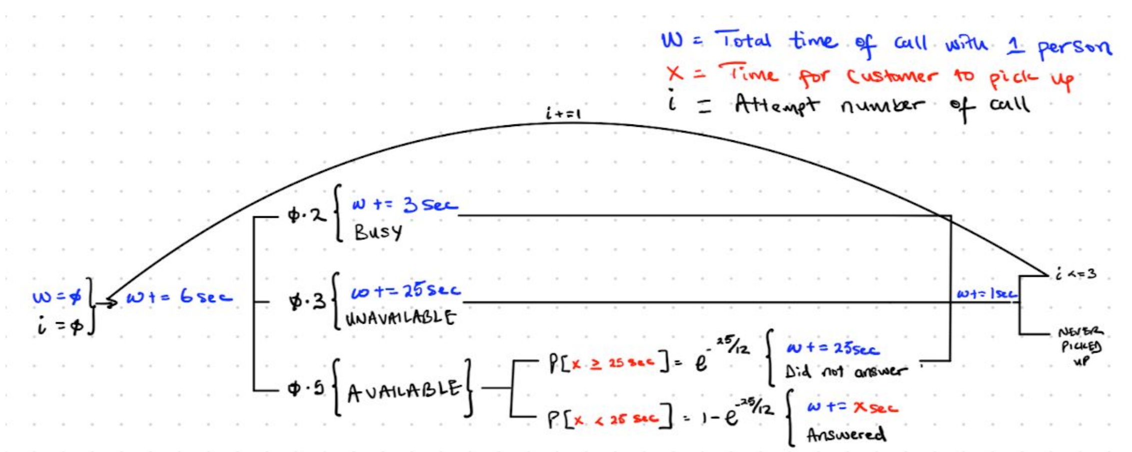

Figure 1: Tree Diagram of all paths Possible in a single call

Monte Carlo Simulation was created to discover the distribution of "*W*" as a function of time. To do this, we created a tree diagram representing all the possible scenarios that could happen and all the possible paths that a single call could go through.

Our tree diagram accounts for the three major situations that can occur in this problem. The first situation is that the person being called is currently using the phone in which case we denoted them to be "busy". This situation happens with a probability of 0.2. Next, we have a scenario that the person being called is unavailable to answer the call, the probability associated with this situation is 0.3. Lastly, there is a situation where the person is available to pick up the phone. This situation is awarded a probability of 0.5. However, given that the user is available, there is a situation where the user picks up or doesn't within a given amount of time. This situation where the user is available and picks up can be modeled using a continuous random variable with a mean of 12 seconds. We used the function to determine the distribution of a continuous random variable, $1 - e^{-\lambda x}$, where $\lambda = 12$ seconds. It's important to note that a representative will re-dial 4 separate times before giving up on the individual. The repetitive call process is represented by the "*i*" in our tree diagram. If the "*i*" reaches 3 (which is the fourth attempt after starting at "*i=0*"), the call process for that individual ends.

$$x_i = (a\, x_{i-1} + c)(modulo\ K),$$

$$u_i = decimal\ representation\ of\ x_i/K,$$

Figure 2: Linear Congruential Random Number Generator

Monte-Carlo Simulation

The design of the Monte-Carlo Simulation algorithm was separated into different sections in order to simplify the process of testing while solving. The first section was the random number generator. The random number generator was modeled after a linear congruential random number generator. The formula provided in the number generator shown in Figure 2 where the starting value (x_0) is 1000, the multiplier (a) = 9429, increment (c) = 3967, and the modulus $(k) = 2^{14}$.

Using this formula, a function was created in python following the pseudocode shown below:

- Define some global variables that hold the previous seed value (X_{i-1})
- Create a function and define the multiplier, and increment values inside the function.
- Using the formula for X_i, write the formula and save it to some variable.
- Reset your global variable to be the value just calculated from X_i.
- Return a value that is as shown in the formula along with U_i.

This pseudocode is the process that was followed in order to create the linear congruential random number generator. Upon running this we got values for the following U_i values.

$$U_{51} = 0.7436\ ,\ U_{52} = 0.3565,\ and\ U_{53} = 0.3159$$

Upon finishing the random number generator, the next step was to create an algorithm that would decide whether a customer is busy, unavailable, or available. The pseudocode that followed is as listed below:

- Define a for loop that iterates from 0 through every number less than 500.
- Call the random number function that was created and store the values to some variable.
- Once this was done, an if-else statement was crafted that checked the random number generated to the ranges for busy, unavailable, or available which were 0 - 0.2, 0.2 - 0.5, and 0.5 - 1.0, respectively.

Once this was done, the final step of the process was to construct the continuous random variable generator that would provide values of how long it took the user to pick up in seconds. This was done by using the same process that was used to create the random number generator, however a different function was made to keep track of the numbers in order to not interfere with the previous function numbers. The process to do this is as follows:

- In the section that accounts for the availability of the user, a new variable is defined to hold the value of the secondary function called.
- Once the function value is obtained, it is plugged into the integral of the CDF function that was acquired.
- That value is then compared against 25 seconds, the amount of time allotted to pick up the phone.
- If the number is more than 25, we go back and restart if we have not called back more than 4 times. If it is less than 25, we record the time it took for the user to pick up the call.

The values from the code were saved after running it 500 times. These values were used later to answer questions. The values saved are in the excel file attached to this submission.

Distribution of W

After generating 500 samples of W, we created an excel spreadsheet to analyze the data. Using built in functions in excel, the mean was determined to be 40.562 seconds. The first quartile was 13.5641 seconds, the median was 27.3817 seconds, and the third quartile was 59.7751 seconds. To determine the probabilities for the next set of questions, $P[W < x]$, the COUNTIF(range,criteria) function was used in excel. This function counts the number of occurrences of an event based on the criteria set. So, to determine $P[W \leq 15]$, we used the COUNTIF function and iterated over the samples of W, with the criteria being less than 15. The values for these events, as well as for $W > w_5$, w_6, and w_7 are shown in Figure 3. These were selected to be 50, 60, and 80 to show an even spread of the data.

Looking at our data, the median is significantly smaller than the mean. This indicates that the probability density function of W would not accurately represent where the mean is. This is because there is a larger cluster of outcomes that have faster times, and a smaller cluster of outcomes that have much slower times. The sample space for W is $6 < W < 128$.

P [W <= 15]	0.282
P [W <= 20]	0.394
P [W <= 30]	0.52
P [W > 40]	0.424
P [W > 50]	0.32
P [W > 60]	0.25
P [W > 80]	0.17

Figure 3: Probabilities of Events

W could be an exponential random variable, because it shows a similar distribution and an exponential random variable can be modeled using the same function we used to find W. The cdf we created could be useful to people who have to make calls for a living, such as call-center workers. The cdf shows the distribution of time it takes for a person to pick up the phone.

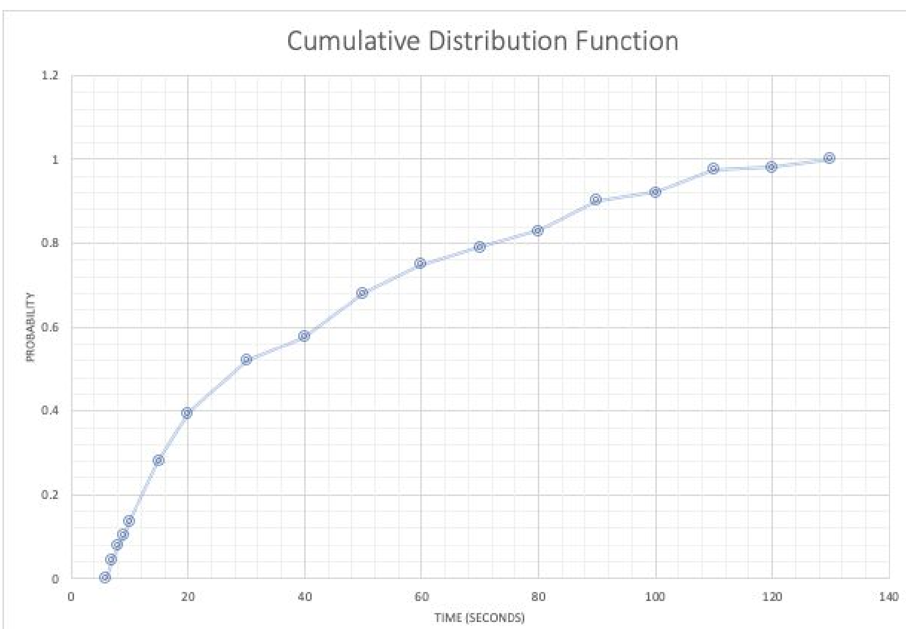

Figure 4: Cumulative Distribution Probability

So, by looking at this distribution call-center workers could maximize their time if a call has gone on too long. If they knew the average time it took for a person to pick up, they could cut their losses by not taking too long on a call.

Conclusion

This project was a valuable lesson in extrapolating data from a real world problem. The first step we took to solve the problem given to us was by modeling the data. A high-level abstraction was attained by creating a tree diagram. Having a visual representation made every other part of the project easier. Next, we had to actually construct the simulation. After much debugging, we had generated code that would accurately simulate the problem statement. Lastly, we had to take the data and draw conclusions from it. With some useful functions in excel, this was done seamlessly. This project was a useful exploration in a real world application of probability.