5 Funnel Simulation

Lesson objectives:

- Understand how time between events affects funnels.
- Use an exponential distribution to simulate funnels.
- Understand the bias of using an MLE estimator when simulating funnels.

Funnels are one of the most common visualization techniques. This exercise explores funnels via simulation with specific attention paid to how the time between events affects funnel visualizations.

1. The first task is to write a funnel simulator. This function, called FSim1, should have the following inputs:

   - n, the number to of users to simulate
   - λ, the parameter

   As an output it should return a list of exponential random variable simulations as a list or an array.

   (a) Using FSim1, create a funnel visualization with 1,000 users and a λ parameter of 2 with stops every .25 to 3. Remember that a funnel simulation isn't a histogram – it is a plot of the number of users who survive beyond the time hurdle. So, if your chart shows the number 775 when the x-axis is at .5, that means that 775 of the 1,000 simulated users had a survival time beyond .5 seconds.

   (b) Using FSim1, repeat the previous assignment, but with λ equal to the values of .2 to 3.0 in step sizes of .2, making sure to plot each.

   What is the relationship between λ and survival times?

2. Now that the basics of the exponential random variable are understood, let's work backward and estimate the λ parameter from some simulated data.

   Starting with FSim1, create another function FSim2, which takes a list of breakpoints and the input from FSim1 to return where users dropped off. FSim2 represents what we would expect to see in a real-world situation, it is a list of the number of users who failed to reach the next step in the user flow. In particular, the input should be:

   - A list from FSim1 of times that users quit
   - A list of breakpoints (of arbitrary length)

the function should then return a list of the number of users who fail to proceed beyond that break-point, but did get to the previous break-point. To make sure that it is working, run the following command and verify your output:

```
>>> FSim2( [.20, .40], [.25, .5])
[1, 1, 0]
```

The output has 2 users, one quitting at .2 seconds and one quitting at .4 seconds with two steps, one at .25 and one at .5. The output represents 1 user quitting before the first step at .25 and one user quitting between the first and second steps. The final zero implies that no user reached beyond the final event.

3. To estimate the λ value we'll use Maximum Likelihood Estimation ("MLE"). The idea behind MLE is to write down the probability of seeing the actual data, conditional on the parameter (in this case our only parameter is λ). We can then look over the possible values of λ and see which value has the highest associated probability. In this case, our data is exponential, which means that the CDF and PDF are:

$$F(x) = 1 - e^{-\lambda x}$$
$$f(x) = \lambda e^{-\lambda x}$$

Assuming that you had the actual quitting times of users (the output of FSim1), to estimate the lambda parameter, via MLE you would need to maximize the following function, assuming the quitting times were defined as $x_1, x_2, ..., x_n$:

$$l = \prod_{i=1}^{n} f(x_i)$$

A trick that is often used is to take the log of this function, which yields:

$$\begin{aligned} L &= \log\big(\prod_{i=1}^{n} f(x_i)\big) \\ &= \sum_{i=1}^{n} \log(\lambda e^{-\lambda x_i}) \\ &= n\log(\lambda) - \lambda \sum_{i=1}^{n} x_i \end{aligned}$$

We want to maximize this expression, so we take the derivative, set it to zero and solve:

$$\begin{aligned} \frac{\partial L}{\partial \lambda} &= 0 \\ 0 &= \frac{n}{\lambda} - \sum_{i=1}^{n} x_i \\ \lambda &= \frac{n}{\sum_{i=1}^{n} x_i} \\ &= \frac{1}{\bar{x}} \end{aligned} \tag{A.1}$$

In other words, if we have the raw quitting times, we just take the inverse of the average to estimate our lambda. Write a function, EstLam1 that uses the output of FSim1 to estimate the λ that generated those values.

(a) EstLam1 is an unbiased estimator. What does this mean?

(b) Generate a sample of 1,000 users using FSim1 with λ equal to 1 and estimate λ using EstLam1.

(c) Using that same sample of 1,000 users, bootstrap a 95% confidence interval for the λ estimate (use 500 bootstraps).

(d) Repeat the above process with number of users equal to 100, 200, 500, 1,000, 2,000, 5,000 and 10,000. Plot the results (both the estimated λ and the confidence intervals on a single plot. How do the results change as the number of users increase?

4. Unfortunately, taking the average won't work when the data is censored, as it often is when event data is observed. In this situation, we'll need to modify our MLE estimator and attempt to maximize it in another way. The following notation will prove helpful when setting this up:

   - Assume that there are b breakpoints and they occur at times equal to $BP_1, BP_2, ..., BP_t$.[4]
   - For each user, define U_i to be the number of steps that each completes.
   - Reindex the users so that the first m_0 represent the number of users who never send the first event (U_i is 0), the next m_1 users to be the users who send any number of events but the last one (U_i is between 1 and $b-1$) and the final m_2 to

[4] We could estimate these based on the average time to each step.

be the number of users who send all events in the funnel ($U_i = b$). This means that $m_0 + m_1 + m_2 = n$, where n is the number of users in the funnel.

- For users in the m_0 group, their likelihood function will be equal to: $F(BP_1|\lambda)$, where F is the previously defined CDF.[5]
- For users in the m_1 group, they survived to a time between BP_{U_i} and BP_{U_i+1}, so their contribution to the likelihood function looks like: $F(BP_{U_i+1}|\lambda) - F(BP_{U_i}|\lambda)$.
- For users in the m_2 group, they survived past the final step so we know that their survival probability is equal to: $1 - F(BP_t|\lambda)$.

Using this notation, the likelihood function is equal to:

$$l = \left\{ \prod_{i=1}^{m_0} F(BP_1|\lambda) \right\} \cdot \left\{ \prod_{i=m_0+1}^{m_0+m_1} F(BP_{U_i+1}|\lambda) - F(BP_{U_i}|\lambda) \right\} \cdot \left\{ \prod_{i=m_0+m_1+1}^{n} 1 - F(BP_t|\lambda) \right\} \tag{A.2}$$

(a) Carefully explain why equation A.1 will not work using event data

(b) Take the log of equation A.2 and, using the distribution function, simplify it by turning it into sums and removing unnecessary expressions.

(c) Using the equation derived in part 4b, write a function, MLE1, which returns the value of the log likelihood function for a given set of data. Its inputs should be:

- A simulated list (the output of FSim2)
- A list of breakpoints, which is the same input of FSim2

FSim2 should return a python lambda function which takes, as its sole input, a λ value and which will return the log likelihood value for the data originally entered in FSim2. An example is shown below:

```
>>> x = [.25, .45, .75]
breaks = [.5]

PRT= MLE1( FSim2(array(x), breaks), breaks)
print PRT(1)
-2.36550425913
```

[5] We are making the dependency on λ explicit

(d) To estimate λ we need to maximize the log likelihood function. To do this, write a function which takes in a range of λ values and then searches over that space for the maximum likelihood. The function should return the λ that maximizes the likelihood. Name this function MaxMLE and example output (continuing from the demonstration above) is shown below:

```
>>> print MaxMLE( FSim2(array(x), breaks), breaks, arange(.1, 3, .05))
2.2
```

5. Now that we have a function that estimates λ as a function of input, let's put it to use by estimating the bias incurred by not having the exact quit times.

   (a) Run 1,000 simulations of 100 users each (using FSim1 with $\lambda = 1$) and pass it through FSim2 with each of the following sets of break points:

   - $[.25, .75]$
   - $[.25, 3]$
   - $[.25, 10]$

   For each simulation calculate the difference between the estimated λ using EstLam1 and MLE1. What is the average difference? How does moving the second breakpoint affect the estimate?

   (b) Using what you learned from doing the above, how should you design breakpoints and the length between them?

What needs to be turned in:

1. A written report answering the questions and containing the appropriate graphs. Note that you can combine multiple graphs into a single chart to preserve paper, especially for 1b and 3d.
2. A python script, which runs top to bottom and generates the output and graphics.