FiveThirtyEight's May 29, 2020 Riddler

Emma Knight

May 31, 2020

This week's riddler, due to Jim Crimmins, is about setting the Zoom server farm on fire:

Question 1. *One Friday morning, suppose everyone in the U.S. (about 330 million people) joins a single Zoom meeting between 8 a.m. and 9 a.m. - to discuss the latest Riddler column, of course. This being a virtual meeting, many people will join late and leave early.*

In fact, the attendees all follow the same steps in determining when to join and leave the meeting. Each person independently picks two random times between 8 a.m. and 9 a.m. not rounded to the nearest minute, mind you, but any time within that range. They then join the meeting at the earlier time and leave the meeting at the later time.

What is the probability that at least one attendee is on the call with everyone else (i.e., the attendees time on the call overlaps with every other persons time on the call)?

Extra credit: What is the probability that at least two attendees are on the call with everyone else?

First off, I will give a heuristic argument that both numbers tend to a fixed constant as the number of people in the call tends to infinity.

To be precise, assume that there are N people in the call. Additionally, I will index time in the call by a number between 0 and 1 (so $t = 0$ is 8 a.m. and $t = 1$ is 9 a.m.). The probability that a particular person leaves before time t is t^2: both random numbers have to be less than t for this to happen. The probability that that same person doesn't leave before time t is then $1 - t^2$. Consequently, the probability that nobody leaves before time t is $(1 - t^2)^N$. We can write this as $\exp(N \log(1 - t^2)) \approx \exp(-Nt^2)$. Thus, one expects the first person to leave when Nt^2 is equal to some constant c, so, on average, the first person to leave leaves at time $c/\sqrt{N}$. Similarly, the first person to enter enters at time $1 - c/\sqrt{N}$.

Now, assume that the first person to leave leaves at *exactly* $c/\sqrt{N}$ and that the last person to enter enters at *exactly* $c/\sqrt{N}$. Then the odds that one particular person overlaps with everyone is $\frac{2c^2}{N}$. The odds that exactly k people overlap with everyone is then $\binom{N}{k} \left(\frac{2c^2}{N}\right)^k (1 - \frac{2c^2}{N})^{N-k}$. If $N \gg k$, then this is roughly $\frac{2^k c^{2k}}{k!} e^{-2c^2}$. If N is not large compared to k, then it is vanishingly unlikely that there are k people that are in the Zoom call at the same time as everyone else.

The probability that there is at least one witness is then $1 - e^{-2c^2}$, and the probability that there are at least two witnesses is $1 - e^{-2c^2} - 2c^2e^{-2c^2}$. This continues on, but there is no mention of numbers larger than 2 in the riddler, so I will pay those numbers no mind.

This argument fails to produce exact numbers, and it is also makes some unjustified assumptions. However, arguments like this typically give answers that are close to reality. To come up with a coherent answer, I will turn to the power of code.

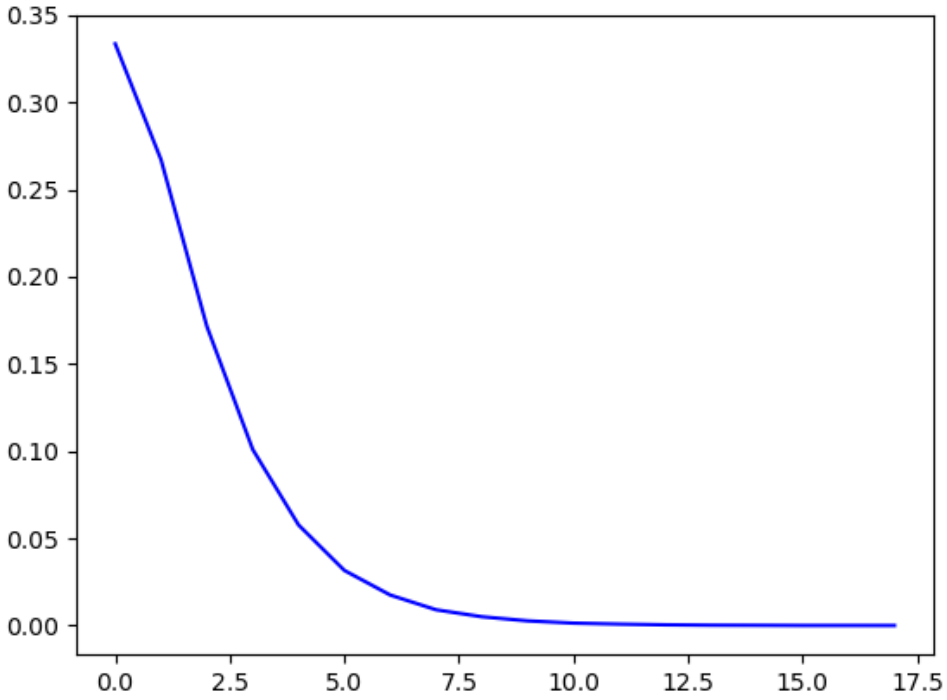
```
import random
import math

##This function does a simulation of a Zoom call
##with population people in it and returns the
##number of people that are in the Zoom call
##with everyone else.

def simulate(population):
    entrances = []
    exits = []
    witnesses = []
    firstexit = 1
    lastentrance = 0

    for i in range(samples):
        x = random.uniform(0,1)
        y = random.uniform(0,1)
        if (y < x):
            z = y
            y = x
            x = z
        entrances.append(x)
        exits.append(y)
        if (x > lastentrance):
            lastentrance = x
        if (y < firstexit):
            firstexit = y

    for i in range(samples):
        if (entrances[i] < firstexit):
            if (exits[i] > lastentrance):
                witnesses.append(i)
    return(len(witnesses))

##These are some variables that are used for
##the simulation.  The code runs simulations
##times, and each simulation is with people
##number of people in the Zoom call.  successes
```

```
##is the number of Zoom calls with a person
##in the call with everyone else, and
##supersuccesses is the number of Zoom calls
##with at least two people in the call with
##everyone else.

simulations = 100000
people = 1000
successes = 0
supersuccesses = 0

for i in range(simulations):
    z = simulate(people)
    if (z > 0):
        successes += 1
    if (z > 1):
        supersuccesses += 1

print(successes)
print(successes/simulations)
print(supersuccesses/simulations)
```

This code runs in $O(\text{simulations} \cdot \text{people})$ time.

Numerically, it appears that there is roughly a $2/3$ chance of at least one person in the Zoom call with everyone else and roughly a $2/5$ chance of at least two people in the Zoom call with everyone else. Running 100000 simulations of 10000 people, I got 66635 calls with at least one person in the Zoom call with everyone else, and 39920 calls with at least two people in the Zoom call with everyone else.

One can play with the code to produce a graph of the distribution on the number of witnesses. With the same simulation as above, I got the following graph of the distribution of number of people in the Zoom call with everyone else:



This also bears out my point about N being large relative to the number of people in the Zoom call with everyone else: here, $N = 10000$, and the most people we got to be in the Zoom call with everyone else was 17.