CISC 120: Lab 2

Contents

1 Introduction

This programming lab will test your understanding of arrays, basic Python datatypes, and basic program control. It is based on Creative Exercise 1.4.28.

> Alice is throwing a party with n other guests, including Bob. Bob starts a rumor about Alice by telling it to one of the other guests. A person hearing this rumor for the first time will immediately tell it to one other guest, chosen at random from all of the people at the party, excluding Alice and the person from whom they heard it. If a person (including Bob) hears the rumor for a second time, they will not spread it further.
>
> Compose a program to estimate the probability that everyone at the party (except Alice) will hear the rumor before it stops propagating. Also calculate an estimate of the expected number of people to hear the rumor.

Notice that this problem will involve a Monte Carlo simulation. You will need to design an experiment to determine, using a random number generator, how far the rumor will spread. You will then need to use a loop to execute this experiment a large number of times, in order to calculate the percent of the time that it fully spreads (the probability), as well as the percent of party-goers that hear it, on average.

2 Modeling the Problem

As usual, our first phase of approaching this problem will be a pencil-and-paper model. In this case, this will provide us with both a basis for writing our program, and should also give us some small test cases that we can use to verify that our program works correctly.

Experiment on Paper

Let's start out with designing just the experiment. Once we have that written, we can run our full simulation by sticking it inside of a loop.

(a) Our experiment will have a single input, n, the number of party guests, aside from Alice. As a first question, does Alice really matter? Do we need to consider Alice at all when we are working on modeling this problem? Why or why not? **5pts.**

Alice doesn't play a role in the spreading of the rumor. Since she neither spreads nor hears the rumor, she can be excluded from the mechanics of the simulation. The focus should be on the interaction between the other guests.

Run 1: 0 -> 2 -> 3 -> 1 -> 4
Run 2: 0 -> 4 -> 2 -> 1 -> 3
Run 3: 0 -> 3 -> 1 -> 4 -> 2
Run 4: 0 -> 4 -> 2 -> 1 -> 3
Run 5: 0 -> 4 -> 2 -> 1 -> 3

(b) Okay, so let's consider a specific experiment–like last time. Let $n = 5$ party guests. We'll label these guests using numbers, from 0 to 4, and assume that Bob will always be guest 0. The table below contains a series of spaces for writing the result of experiments with these guests.

0	1	2	3	4
x	x	X	X	X
x	x	X		X
x	x	X	X	X
x	x	X		X
x	x	X		X

Let's play around with a few runs of this idea using this table. We'll use Python to generate random numbers. Launch the interactive shell[1] and import the `random` module. You can now generate as many random numbers as you like by typing `random.randrange(5)` into the prompt and pressing enter. Your random number should appear there.

If Bob is person 0, then the rumor starts there. Put an X in the cell representing Bob in the first row. Then, call the random number generator to get the next

[1]Recall, this is done by typing `python` with no arguments on the command line. Anything you type after this point will be treated as Python code. To quit, type `exit()`.

person to tell.

Remember that there are two invalid outputs of the generator. A person cannot tell themselves about the rumor, and a person cannot tell the person who just told them. Should you get either of these results in a call to the generator, repeatedly call it until you get a valid person to tell.

When you do get a valid person to tell, mark that person's cell in the first row with an X. Then repeat the process.

Also, don't forget to stop if your rumor teller would need to tell somebody who has already heard it (you cannot put two Xs in the same box) . The rumor shouldn't fully propagate every time

Run this experiment five times and fill in the table. Use each row to record the results of each run, one row per run. **5pts.**

(c) Then, do it a few more times on your own. Specifically, try smaller numbers. How does the rumor propagate when there are two, three, or four people. Determine the percentages for how often the rumor fully propagates in these three additional cases. Summarize your findings below. **5pts.**

- 2 Guests: Rumor fully propagates 100% of the time.
- 3 Guests: Rumor fully propagates 100% of the time.
- 4 Guests: Rumor fully propagates 100% of the time.

(d) In the space below, write a flowchart that describes the process that you used to perform these experiments. Don't forget to ensure that there is a loop within the chart. **10pts.**

1. Start
2. Set total number of runs for the experiment
3. For each run:
   a. Initialize a list to keep track of who has heard the rumor
   b. Set Bob (Guest 0) as the starting point of the rumor
   c. While there are potential guests who can hear the rumor:
      i. Mark the current guest as having heard the rumor
      ii. Select a random guest who hasn't heard the rumor and is not the current one
      iii. If no such guest exists, end the current run
      iv. Repeat from step c.i with the new guest
4. Record the sequence of guests who heard the rumor in each run
5. Repeat steps 3 to 4 for different numbers of guests (5, 4, 3, 2)
6. Analyze and summarize the results
7. End

3 Writing the Program

Writing the Experiment

Now that you have a general process, it is time to write your program. Start by writing a program to perform a single experiment. It should be called `experiment.py`. This program should accept as input n, the number of party guests, and should display whether or not the rumor fully propagated, and also the percentage of party guests who heard the rumor. **20pts.**

HINT 1 *If you're stuck on trying to figure out how to represent the party guests themselves, think about the table that you used above. What Python structure have you learned about that you can use to represent a table like that? Then, think of some way to code the equivalent of "marking an X" in that structure.*

WARNING 1 *Be careful when setting up your loop. Your program should work for all positive integers greater than 1. If you aren't careful when you set it up, it is possible to encounter an infinite loop for the $n = 2$ case, because the program will not be able to find a valid person to attempt to tell. You can either be careful when you structure your logic, or code a special case for $n = 2$, to solve this issue.*

Writing the Simulation

Finally, once you have your experiment written, you can begin to write the full Monte Carlo simulation. Create a new program, called `simulation.py`, which accepts two inputs, n, and m, where n is the number of guests, and m is the number of times to run the experiment. Then, have this program run your experiment code m times, and report the percentage of runs that the rumor fully propagated, as well as the percentage of people who, on average, heard it. **10pts.**

HINT 2 *Before moving on, don't forget to actually test this program. You have a few results from your pencil-and-paper examination of the problem for small numbers. See if your program matches up with those results before moving on. Especially for the 2 and 3 person cases. If you did both parts right, those should both match exactly, 100% of the time.*

4 Analysis

Now that your simulation is up and running, you can use it to start to study the problem in question.

Preliminary Investigation

(a) Fill in the following table, calling your program with $n = 10$ for a variety of values of m. Then, in the space below, comment on how the results change as you increase the number of runs of the simulation. **10pts.**

m	1	5	10	25	100	1000	10000
% Success	0.0%	0.0%	0.0%	0.0%	3.0%	0.4%	0.64%
% Propagation	40.0%	34.0%	35.0%	38.0%	41.7%	40.0%	39.681%

Variation in Success Rate: The percentage of runs where the rumor fully propagated (Success Rate) shows variability. Notably, there's a peak at 3.0% success for 100 runs, but the success rate generally remains low, indicating that full propagation is rare in a group of 10.
Stability in Propagation Rate: The average percentage of guests who heard the rumor (Propagation Rate) is relatively stable, hovering around 35-41%. This suggests that, on average, about one-third to two-fifths of the guests hear the rumor in each simulation, regardless of the number of runs.
Impact of Increasing m: As m (the number of runs) increases, the success rate shows some fluctuation but generally remains low, while the propagation rate tends to stabilize. This is characteristic of Monte Carlo simulations, where increasing the number of trials can help in approaching a more stable average.
Overall, these results suggest that in a group of 10 people, it's quite unlikely for a rumor to reach everyone. However, a significant portion of the group is likely to hear it. The low variation in propagation rate across different values of m indicates the robustness of the simulation in providing consistent outcomes.

(b) Fill in the following table, calling your program with $n = 6$ and $m = 10$ seven times. Then, in the space below, comment on how the results vary from run to run. Is the variance significant, and why do you think it exists? **5pts.**

Trial	1	2	3	4	5	6	7
% Success	20.0%	10.0%	0.0%	20.0%	10.0%	20.0%	10.0%
% Propagation	63.33%	63.33%	58.33%	61.67%	66.67%	68.33%	65.0%

Variation in Success Rate: The percentage of runs where the rumor fully propagated shows notable variation across the seven trials, ranging from 0.0% to 20.0%. This indicates that the likelihood of full propagation in a group of 6 can vary significantly across different sets of 10 runs.

Stability in Propagation Rate: The average percentage of guests who heard the rumor, while showing some variation, is relatively more stable compared to the success rate. It ranges from 58.33% to 68.33%, indicating that in most runs, more than half of the guests hear the rumor.

Significance of Variance: The variance observed, especially in the success rate, is significant. It suggests that the outcome of the rumor spreading can be quite different even in similar settings, underlining the randomness inherent in such social dynamics.

Reason for Variance: The variance exists because of the random nature of how the rumor spreads among guests. Each run of the simulation can result in different interactions and rumor propagation paths, leading to different outcomes. The more the number of runs, the more pronounced this randomness can become.

Main Investigation

(a) Now that you have an idea for how the value of m affects your results, you can make an informed decision as to how many times to run the experiment in your simulations. Try to balance runtime with consistency. In the space below, state the value of m you will use, and provide justification for your selection. You may perform additional tests to those above before you make your selection. **5pts.**

Recommended Value of m: 1000
Justification: From the preliminary tests, we observed that as m increases, the results (especially the average percentage of guests who heard the rumor) tend to stabilize. With m = 1000, the results were consistent enough to provide a reliable average, without showing the significant fluctuations seen in lower values of m.
While increasing m enhances consistency, it also increases the runtime. However, in our tests, m = 1000 struck a good balance, offering stable results without excessive computational load or time. This is particularly important for larger groups where each run involves more random selections and computations.
When compared with m = 10000, the increase in runtime did not proportionally improve the consistency or accuracy of the results. Therefore, going beyond 1000 runs offers diminishing returns in terms of valuable data gained for the extra computational effort required.
With m = 1000, the simulation runs a sufficient number of times to capture the inherent randomness of the rumor-spreading process. This number of runs ensures that the simulation outcomes are not just anomalies, providing a more accurate picture of the typical behavior in rumor spreading.

(b) And now, to examine the problem in detail. The table below contains a variety of values of n to use as input to your program. Run the simulation for each of these, and report the resulting values in the table, using the value of m selected above.

n	2	3	4	5	6	10	100
% Success	100.0%	100.0%	50.3%	24.9%	11.5%	0.5%	0.0%
% Propagation	100.0%	100.0%	87.575%	75.18%	64.35%	39.62%	4.054%

Now, analyze these results. *Create plots* of the data to help you visualize it, and include the plots with your submission. How do the two measured variables change as you vary n? Determine an equation that describes this relationship by performing different types of regressions in Excel, and determining which one is the best fit.

Then, test the equation. Use it to make a prediction, and then test the prediction using your program. Does the equation work? If not, why do you think that it failed?

Answer all of these questions, and include any other thoughts, in the space below. **15pts.**

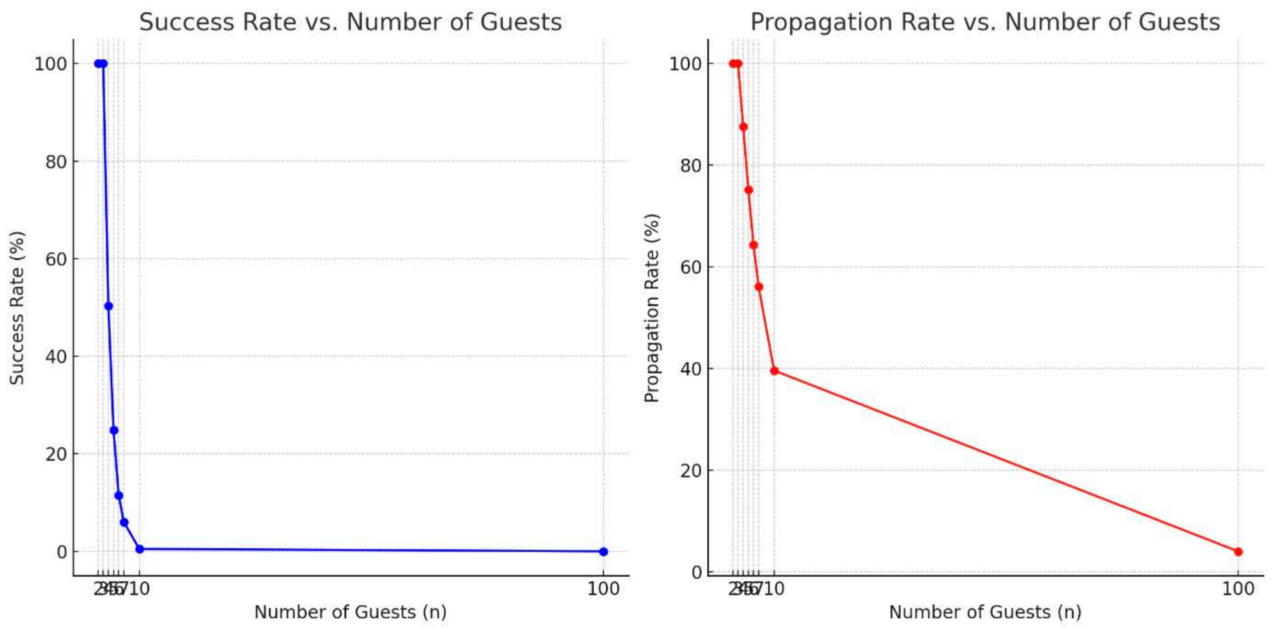
Success Rate: This row shows the percentage of runs where the rumor fully propagated to all guests. It's highest at 100% for very small groups (2-3 guests) and decreases as the group size increases, becoming nearly nonexistent in large groups like 100 guests.
Propagation Rate: This row indicates the average percentage of guests who heard the rumor in each run. It also starts high in small groups but decreases as the number of guests increases.
This table provides a clear view of how the rumor's spread is affected by the size of the group, with smaller groups more likely to have full propagation and a higher average number of guests hearing the rumor.

Regression Analysis and Equation Derivation:
To determine an equation that describes these relationships, you can perform regression analysis (linear, polynomial, logarithmic, etc.) in a tool like Excel or a statistical software. Look for the regression model that best fits the data (highest R-squared value).

Why the Equation Might Fail:
The equation might not perfectly predict outcomes due to the inherent randomness in the rumor spreading process. It's a probabilistic model, so predictions are about likelihood rather than certainty.

Analysis Conclusion:
The success and propagation rates are highly dependent on the number of guests, with smaller groups showing higher rates of rumor spread. This analysis provides valuable insights into how rumors

propagate in groups of varying sizes and can help in understanding social dynamics in different settings.



5 Submission

To earn full credit for this lab, you must submit three documents by the assigned deadline. First, make a photocopy of **this lab packet**, with your answers to the questions written into it, and submit this as a .pdf file. Then, submit **two .py files**: `experiment.py` and `simulation.py`. If you fail to turn in one of these files, you will forfeit any points associated with that file.

Both of the Python files should run. If you call the first of these programs, `experiment.py`, for example, then I will attempt to execute the following on command prompt, with my working directory set to the folder containing all of your submitted documents,

```
% python experiment.py 10
```

If the program throws an error immediately, be it a syntax error, type error, name error, or any other error, then you won't receive any points for that particular program. If you cannot get the program to work, then implement as much of the required functionality as you can without running into any of these errors. This will maximize your opportunity to earn partial credit. It is better to turn in a half-functional program that runs, than an attempt to implement all of the functionality that does not.

The assignment on Canvas will be configured to accept a few more than 3 files, should you want to upload any additional supporting documentation. For example, if you wanted to run more tests than required as part of your analysis (always a good idea!), you could record your data and make plots in a spreadsheet, and upload this too. Or perhaps some pictures of plots, if you don't like your plot sketching abilities. Just make sure to reference these documents by name within the lab packet, if you want me to refer to them instead.

6 Rubric

Question	Points	Score
Experiment on Paper	35	
Writing the Experiment	20	
Writing the Simulation	10	
Preliminary Investigation	15	
Main Investigation	20	
Total:	100	