Final

Bulgaria: Rainfall and Flooding Simulation

Minerva University
CS166: Modeling and Analysis of Complex Systems
Professor Ribeiro
April 19, 2023

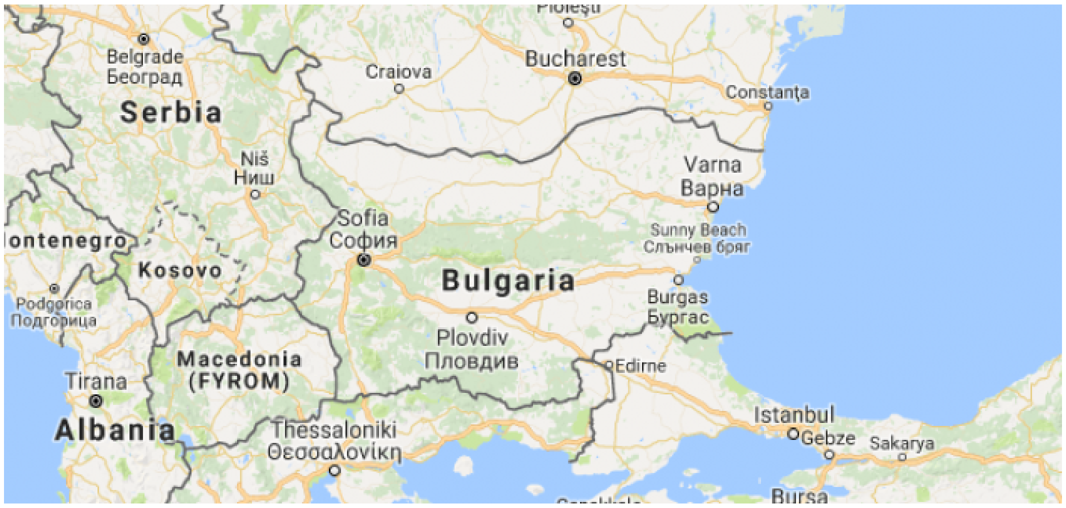
Table of Content

Introduction

In the past 10 years Europe has experienced numerous floods. One region that has been significantly affected is South Eastern Europe, more specifically the Balkan Peninsula and Romania (https://resources.eumetrain.org/data/3/329/hiw_session_8.pdf).

I am from Bulgaria and throughout the past couple of years, my country has experienced devastating floods due to rainfall and storms. In 2022, thousands of people were evacuated from their homes due to torrential rains that caused flooding and overflowing of rivers. The main areas that were impacted were those of Southern Bulgaria, in particular, Plovdiv and the region around it (reference Figure 1) (Davies, 2022).

Figure 1. Map of Bulgaria.



Note: You can see Plovdiv at the center of the southern part of Bulgaria.

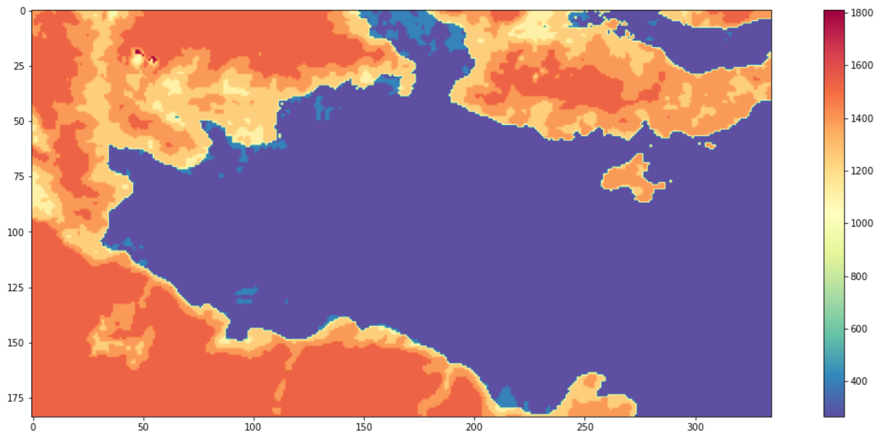
Furthermore, Plovdiv is the second-biggest city in Bulgaria and the oldest continuously-inhabited city in Europe going back to at least 6000 BC (*Plovdiv – Travel Guide at Wikivoyage*, 2023). Various sites from the Roman, Thracian, and Bulgarian empires are located in the Plovdiv

Province. Plovdiv's economic and historical significance is an additional reason why it is important that the region is protected from floods.

Getting Data

For the model I used elevation data from Bulgaria. Since I did not have access to a relevant dataset, I had to do the data processing myself (see the Jupyter Notebook "Elevation map of Bulgaria" section for the code). I got a screenshot of a topographic map and the color bar. After that, I traversed the pixels on the map and calculated the minimum L2 norm between the color bar and the color of the current pixel. This way I created a 2-dimensional array with the elevation of each cell. My computer takes too long to load when considering the whole map of Bulgaria so as I said I am focusing on Plovdiv Province so I computed that map (see Figure 2).

Figure 2. Topographic map of Plovdiv Province and the region around it.



After collecting the elevation data, I also did research on the relevant solid types in the region and their properties. This is important because the soil type would influence the absorption rate of the ground as well as the maximum capacity that can be stored in a cell. However, there is no publicly available data about the soil that I could incorporate.

Model and Simulation

Variables

For the model we will be using a cellular automaton. For the state, I initialize the self.topography attribute of the simulation as a representation of the Plovdiv region (Figure 1). The topography attribute is a 2-dimensional matrix where each cell is represented as a dictionary. Each cell has the following dictionary keys:

- water_content (int): The current water in that cell.
- elevation (int): The value of the elevation of that cell.
- water_capacity (int): The maximum capacity of water that the cell can contain.
- water_absorption_rate (float): The rate of the ground in that cell absorbing water into it.

The other parameter in the model is an evaporation_rate (float), which represents rhe natural water evaporation rate of the rain. It is the average fragment of the water that will evaporate from each cell at each update step.

There are also other attributes in the model that help us keep track of the state:

- water_distribution: Represents the state at the current time step of each cell in terms of water content.
- next_water_distribution: Represents the state at the next time step of each cell in terms of water content.
- water_change: Keeps track of the number of times water moved from one cell to another cell at each time step. Used to track the dynamics of the system. Used to see if the behavior converged.
- step_counter: The current step. Simulation (initial state) begins at step 0

Before updating the model I add rainwater with the help of an add_water() function. I simply specify the segment of the map where it should rain with x and y coordinates and draw a random sample from a distribution for the amount of water to be added in each cell. The distribution can be any distribution we choose and has the potential to be based on real rain patterns. For now, I am using a default value of a uniform distribution.

Update Rules

At each time step the state is updated according to the following rules:

- The simulation traverses the whole topology landscape grid.
- In each cell where there is water:
  - Rule 1: Water evaporation
    - The water evaporates according to the evaporation_rate parameter.
  - Then we find the lower elevation neighbors of the current cell.
  - For each of the neighbors, the incoming water towards it is calculated as the appropriate fraction of water coming from the original cell.
  - Rule 2: Water absorption
    - From the incoming water, the neighboring cell can absorb some. The absorption is based on the amount of incoming water, the water absorption rate of the cell, and the maximum water capacity of the cell. It cannot be more than the maximum capacity of the cell so we take the minimum of the water capacity and the incoming water.
  - Rule 3: Water flow

        - The water that remains after rules 1 and 2 gets added to the cell in the next state.
        - The watter that is added to a cell is also added to the elevation on the topography map to account for the water depth on top of the regular terrain.

Assumptions

The model makes several assumptions:

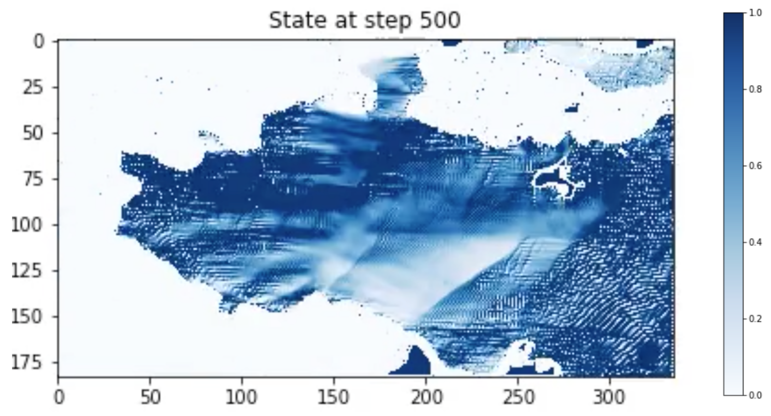
- The model assumes that water moves only to lower level neighbors – in a way using the downwards slope to move. However, it does not take into account water velocity which may help propagate water through uphill landscapes as well. For a region like the Plovdiv Province that is a valley surrounded by mountains this may be a key factor because the water may be coming downwards from steep mountains.
- The model assumes that the evaporation rate is uniform for each cell on the grid. However, in reality the evaporation rate may vary depending on the temperature, soil type, humidity, sunlight strength and coverage, etc.
- The model assumes that each cell absorbs water uniformly but that may not be the case. It may depend on other factors like presence of plants, soil type, and rain duration that impact the way the ground would absorb water.

Considering all of these assumptions, the model represents a limited version of reality but is powerful enough to provide us with insights about the overall flow of the water.

Analysis

This section will explore empirical results by running the simulation. First I wanted to see which areas are the most vulnerable to flooding. I tried out different rainfall scenarios but all of the results show a similar trend to that seen in figure 3 where the valley gets flooded over with the water as it has nowhere else to go.

Figure 3. Flooding pattern from the simulation.

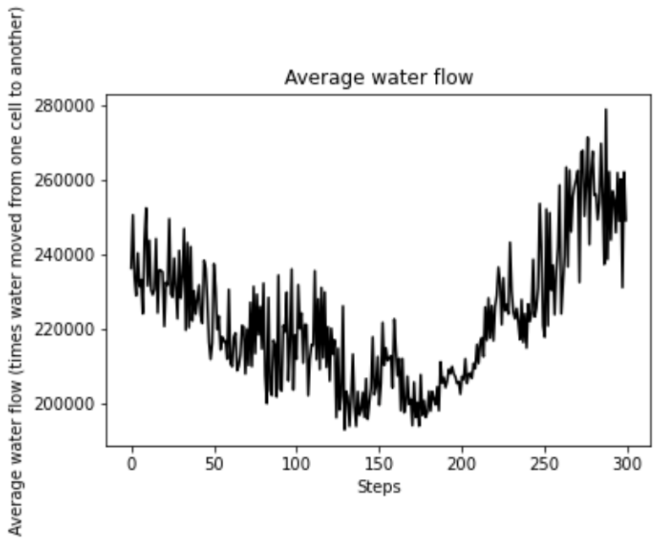


Note: This flooding pattern was produced by adding a uniformly distributed amount of water to all cells in the grid. The water absorption was set to 0 so barely any water remains on the mountains and the water remains congested in the valley. The pattern is similar when theinitial rainfall is only in the mountains as the water quickly falls down from them and into the valley.

Average Water Flow and Water Depth in Plovdiv(without intervention)

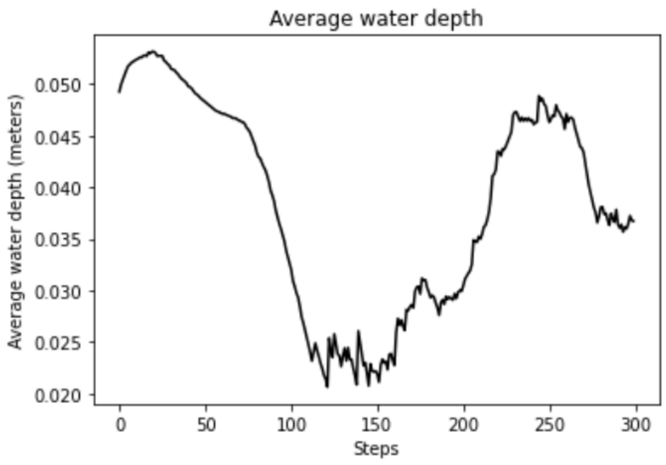
Then I explored how the average water flow changes.

Figure 4. Average water flow as the simulation converges.



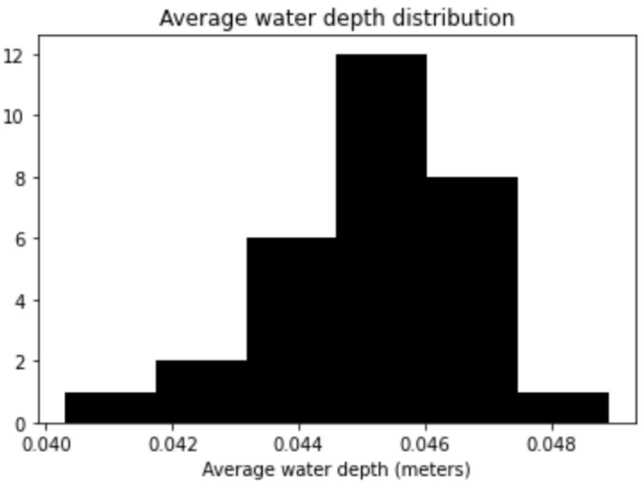
Note: The line plot shows how the average water flow on the whole map changes over time.

Figure 5. Average water depth.



Note: This line plot shows the average water depth of the Plovdiv city (not the whole map) over 300 updates.

Figure 6. Average water depth distribution.



Note: This histogram shows the distribution of the final average water depth in the city (Plovdiv) of 30 random trials with 30 update steps each. The starting rainfall was a uniform distribution between 0 and 0.1. The 95% confidence interval is (0.0446 0.0457); the mean is 0.0451.

From Figures 4 and 5 we can see that it does not seem like the water flow in the map converges at some point. This is not because we are not running the simulation for long enough. If we look at the animations in the Jupyter Notebook, we can see the patterns. The water eventually congregates in the valley. However, since we have set the absorption capacity of the cells to 0, once the water is in the valley it keeps moving around the valley in a pattern that looks like bouncing off the walls of the mountains. In Figure 1 the interactions increase at the end as the water is in the valley because there are simply a bigger number of lower elevation neighbors so the water keeps moving back and forth as it has nowhere to be retained. In Figure 2 we can see that the water depth decreases at first, which is because Plovdiv is built on 7 hills so the water trickles out. However, when the water from the mountains comes down, it goes back into the city area eventually and the water depth increases causing a flood.

In Figure 6 we can also see the distribution of the average water depth in the city of Plovdiv. From the central limit theorem, we expect the distribution to be approximately normal because we take the average value of the water depth. Figure 6 is close to a normal distribution but since we have only 30 trials the sample is too small (more trials took too long to compute0. The 95% confidence interval is (0.0446 0.0457) with a mean of 0.0451. This is actually a very high number considering that at first, we add rain with a uniform distribution between 0 and 0.01, which indicates that the water all goes down to the valley and is likely to form a flood.

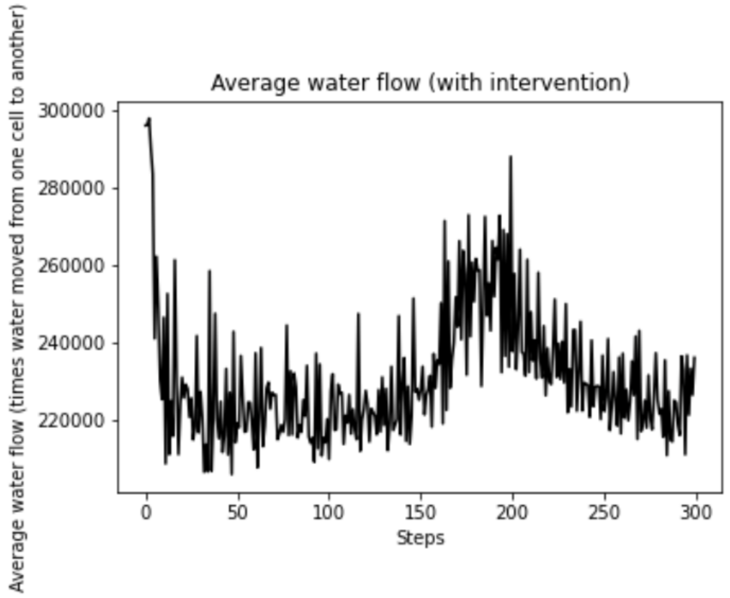
Average Water Flow (with intervention)

As we have seen from the results so far the rainfall water simply has nowhere else to go. I could not pinpoint a particular area where the water comes from or congests; therefore, building a damp does not seem like the optimal solution.

One of the best strategies to deal with urban flooding is the ‘sponge city’ strategy. In this strategy, the city is able to retain the water like a sponge through ecological measures like urban farms, green spaces, grass areas, rooftop gardens, etc (Wavin, 2016).

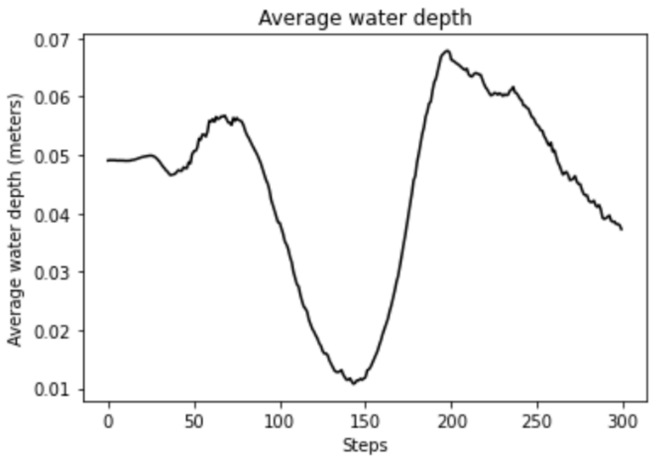
I increased the absorption capacity of the Plovdiv Province and observed if there would be a difference in the results.

Figure 7. Average water flow as the simulation converges.



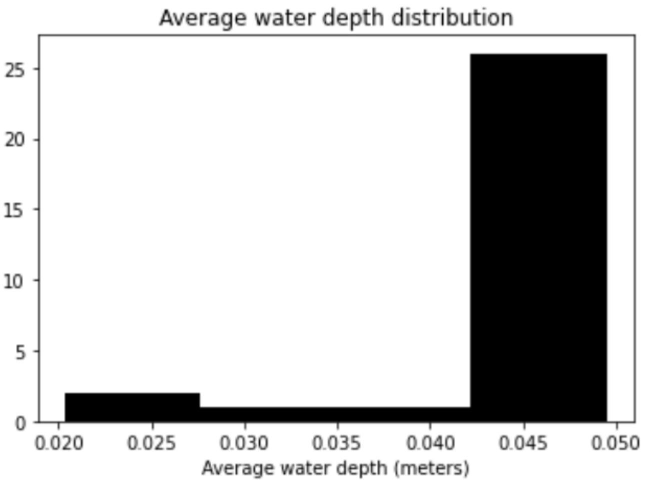
Note: The line plot shows how the average water flow on the whole map changes over time when the water capacity and absorption of Plovdiv city is increased.

Figure 8. Average water depth.



Note: This line plot shows the average water depth of the Plovdiv city (not the whole map) over 300 updates when the water capacity and absorption of Plovdiv city is increased.

Figure 9. Average water depth distribution with intervention.



Note: This histogram shows the distribution of the final average water depth in the city (Plovdiv) of 30 random trials with 30 update steps each when the water capacity and absorption of Plovdiv city is increased. The starting rainfall was a uniform distribution between 0 and 0.1. The 95% confidence interval is (0.0408, 0.0454); the mean is 0.043.

If we compare Figure 7 with Figure 4, we can see that the water flow experiences a decrease as the water is less likely to move between neighboring cells even if slightly. In Figure 4 the interactions increased for reasons explained above and they hover around 26 thousand. However, in Figure 7 they actually decrease and hover around 22 thousand. Then, if we compare Figure 8 with Figure 5 we can see that the water depth follows a very similar trend of decreasing and increasing again. With the intervention, it actually reaches a higher peak around 0.7 compared to 0.5 before. This is because some of the water is contained in the cells permanently as they absorb it. In reality, even if there is more water in the cell, most of it is absorbed and the water that is on top of the surface is less. The distribution in Figure 9 is different from that in Figure 8 in several ways. There is a left skew. Even though most of the points are located on the right at similar values as before, we can observe that the range is now bigger and goes down to much lower water depths of 0.02 compared with the lowest value of 0.04 in Figure 6. There are not that many trials (sample size is again 30) because with my laptop I could not run a bigger number of them. Therefore this strategy can be effective as the city will be able to retain some of the water, if not all, by absorbing it into the ground instead of flooding. Still, it does not seem to be sufficient to fully prevent flooding due to the topology and torrential rain in the region.

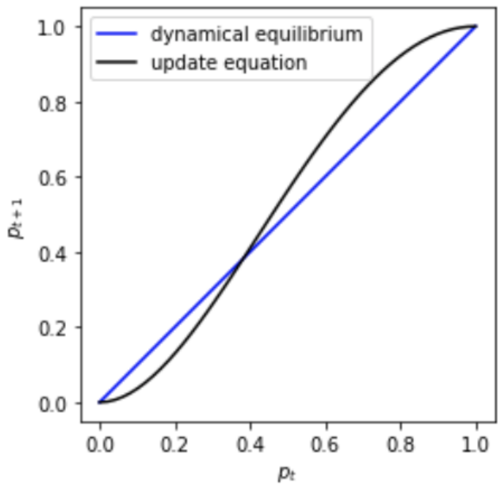
Renormalization: Theoretical Analysis

We can use the renormalization scenario in the context of the flooding model. As per the Sayama textbook, the conductance of a 2 by 2 grid with a Moore neighborhood was given by the following formula:

$$\textit{Equation (1):}\; p_2 = p_1^{\,4} + 4p_1^{\,3}(1 - p_1) + 4p_1^{\,2}(1 - p_1)^2$$

Using the formula we can generate a cobweb plot and find the threshold value (see Figure 10).

Figure 10. Cobweb plot for percolation.



Note: This cobweb plot helps us find the threshold value for when percolation will occur in the system.

In this context, percolation would occur when the water goes through the whole map from one end of it to the other. In the forest fire model, our main parameter was the tree density. Here we can determine this through one of the parameters like the elevation (the density of very high points on the map), soil type (density of low absorbing ground), or the evaporation rate since all of them are relevant for the water to be able to move from one end to the other. Since we do not have data for the other parameters, the most relevant one is the elevation. The elevation of the cells determines whether water moves from one cell to another (just like the existence of a tree determined whether the fire could move to another cell with a tree). If we look at Figure 10, we can see that the threshold is at about 0.4 which means that if at most 0.4 cells on the grid are with an extremely high elevation and serve as walls for the water, then we would observe percolation. In the empirical results above we observe percolation which makes sense because most the cells on the map do not have a very high elevation value. So the theoretical and empirical results are consistent. To test this further we could determine the thresholds for the other parameters as well and compare the theoretical and empirical results on various random topologies to see the fraction of cells that are flooded.

Conclusion

Ultimately, urban flooding is an important problem to be tackled in Bulgaria as it impacts thousands of people and Plovdiv as an economic and historical center. Prevention strategies are complex and they should often be a combination of several different measures. The sponge strategy seems to be impactful but not a full solution to the flooding problem. Still, it may be the first step to tackling it. This project can be extended to explore other strategies like building canals, water tanks, or a dam in the Plovdiv Province. It would also benefit from reliable data of rainfall patterns, soil types, and evaporation rates.

WORD COUNT: 2000 words

Appendix

Appendix A: *LOs*

#Modeling

I applied this LO in the Introduction, Getting Data and Model and Simulation sections. I included a thorough description of the model scenario, rules, parameters, and assumptions. I conducted research and collected elevation data for the topology of the map. Furthermore I explain all of the variables and parameters as well as the update rules. I also critique some of the assumptions of the model.

Word count: 65

#PythonImplementation

I created a topographic map of the elevation of the relevant region using Python. I implemented a Python class for the Flood simulation. It follows all of the relevant update rules and assumptions. I also use appropriate data structures throughout the implementation. For example, for the topology in the simulation I use a 2-dimensional array whee each cell is represented as a dictionary with certain parameters like the absorption and water capacities. I also created an animation that makes it easy to visualize the state of the simulation and to follow through all of the updates as the water moves through the landscape. The code is broken down into different functions that are reused. I also included different tests to show that the model is working as expected (i.e. testing the water coming down the mountains, testing if there is rainfall on all of the map).

Word count: 147

#CodeReadability

All of the code is well documented. There is an appropriate level of comments that makes it easy to follow the logic throughout the code. All of the methods have docstrings. I also formatted the Jupyter Notebook with titles and explanations. The variable names are also descriptive and make it easy to follow the code used to generate the graphs as well.

Word count: 62

#Professionalism

The report follows an appropriate format, with good grammar, and organization. There is a table of contents that allows for easy navigation together with the page numbers. There are also clear and concise titles and subtitles for all sections. All graphs have corresponding figure numbers, titles, labels, units, and descriptions.

Word count: 50

#EmpiricalAnalysis

I applied this LO in the Analysis section. After implementing the model in Python, I used it to explore the scenario and gain valuable insights. I created appropriate plots that allowed me to determine whether the intervention would be effective. I could not run each of the experiments with a very big number of trials because they were very slow to compute. Still I made sure to have at least 30 trials in the sample. I ran each of the experiments with a big number of trials. I also included relevant stats like confidence intervals as well as the mean values. I explain how the central limit theorem was relevant.

Word count: 110

#TheoreticalAnalysis

I applied this LO in the Theretical Analysis section.

Word count: 66

Appendix B: *HCs*

#modeling

See the description of the #Modeling LO for more detail. I created a model by abstracting the details from a real-world scenario. I also implemented and tested the model in Python.

Word count: 114

#audience

My target audience is the government and people of the Plovdiv Province to spread awareness of this issue and provide constructive feedback on action items that can be taken by the local government to protect the region. I first start off by putting the problem into perspective and explaining its significance as well as my personal motivations. Then I prove why the region is prone to flooding and propose a possible intervention, the 'sponge city' strategy, that can help tackle the issue.

Word count: 82

References

Davies, R. (2022, September 5). *Bulgaria – Hundreds Evacuated After Damaging Floods in Plovdiv Province – FloodList*. Floodlist.com. https://floodlist.com/europe/bulgaria-floods-plovdiv-september-2022#:~:text=Hundreds%20of%20people%20have%20evacuated

Plovdiv – Travel guide at Wikivoyage. (2023). En.wikivoyage.org; Wikivoyage. https://en.wikivoyage.org/wiki/Plovdiv

Wavin. (2016, August 26). *10 measures to prevent (urban) flooding*. Wavin. https://www.wavin.com/en-en/News-Cases/News/10-measures-to-prevent-urban-flooding