Homework 3: Detection, Surveillance

Released: Oct 3, 2024; Due: **5pm ET**, Oct 22, 2023

Georgia Tech
College of Computing
B. Aditya Prakash

CSE 8803 EPI Fall 2024
Student NAME: Shikhar Verma
Student GTID: 903497421

Days Late: 2
Slip Days Left: 0

Reminders:

1. *Out of 100 points. 4 Questions. Contains 5 pages.*
2. *If you use Late days, mark how many you are using (out of maximum 4 available) at the top of your answer PDF.*
3. *There could be more than one correct answer. We shall accept them all.*
4. *Whenever you are making an assumption, please state it clearly.*
5. *You will submit a solution pdf* `LASTNAME.pdf` *containing your answers and the plots as well as a tar-ball* `LASTNAME.tgz` *that contains your code and any output files.*
6. *Please type your answers either in LaTeXdocument or in a separate file like a Word document and then convert it into a pdf file. Typed answers are strongly encouraged. Illegible handwriting may get no points, at the discretion of the grader. Only drawings may be hand-drawn, as long as they are neat and legible.*
7. *Additionally, you will submit one tar-ball* `LASTNAME.tgz` *that contains your code and any results files. Code and results for each question should be contained in a separate sub-directory (Eg:* `Q1`*) and there should be a* `README.txt` *file for each sub-directory explaining any packages to install, command to run the code files and location of the expected output. Please follow the naming convention* ***strictly****.*
8. *If a question asks you to submit code please enter the file path (Eg:* `Q1/Q-1.3.1.py`*) in the solution pdf.*
9. *You can download all the datasets needed for this homework from canvas files, you can check the information about the datasets in the* `README.txt` *file.*

1. (14 points) Submodularity

Q 1.1 (4 points) Prove that the coverage function from our lecture is a monotone submodular function. Recall that the coverage function takes a collection of sets $\{S_i\}_{i=1}^n$ and outputs the size of their union $|S_1 \cup S_2 \cup \cdots \cup S_n|$.

Solution: Let's suppose we have 2 sets A and B and a collection of sets S, where $S = S_1, S_2, \ldots, S_n$. If we run the function $f(x)$ on both sets A and B, our notation will look like:

$$f(A) = |\bigcup_{i \in A} S_i|$$
$$f(B) = |\bigcup_{i \in B} S_i|$$

Let's further suppose that $A \subseteq B$. Then, if we run the function on set A and set B, we will find that the union of however many sets for A and B will still cause all the sets unionized in A to still be a subset of the all the sets unionized in B.

$$\bigcup_{i \in A} S_i \subseteq \bigcup_{i \in B} S_i$$

The union operation only adds elements or keeps unique elements, meaning no elements are removed through the union operation. This ensures that $f(B) \geq f(A)$, proving that this function is monotone.

A function is submodular if $\forall A \subseteq B$,

$f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B)$

Let's again suppose that $A \subseteq B$ and $U = \{u\}$, where U is a set unionized with A and B, respectively. Then, we have two possible cases that can occur when we run $f(A)$ and $f(B)$.

Case 1: U is not present in A nor B
If this case is true, the marginal utility of adding U to both A and B will be the exact same: 1. Since we're adding 1 new element to both A and B, the cardinality of those sets will both increase by 1. Therefore, subtracting the new cardinality from the original will be 1, and the definition of a submodular function still holds:

$1 \geq 1$

Case 2: U is present in B but not A
If this case is true, the marginal utility of adding the new set to A will be 1, but the marginal utility of adding the new set to B will be 0. Since B already has the set, unionizing the set with B will just keep the element already present, keeping the cardinality the exact same. Therefore, the marginal utility will be 0 and the definition of a submodular function still holds:

$1 \geq 0$

A third case where U is in A but not in B is not possible since $A \subseteq B$. Furthermore, adding more sets to U won't disprove this proof as the marginal utility still holds because the function is monotone.

$\therefore$ **The coverage function is a monotone submodular function**

Q 1.2 (5 points) Let f be a monotone submodular function. Show that for any two sets S and T: $f(T) - f(S) \leq \sum_{e \in T} (f(S + e) - f(S))$.

Solution: Let's assume that $S \subseteq T$. The function inqueality we have is

$f(T) - f(S) \leq \sum_{e \in T} (f(S + e) - f(S))$

Let's also assume that each element e we add to S is a unique element not currently present in S up to k elements. We can find these elements by calculating the set difference of S and T:

$T - S = \{e_1, e_2, e_3, \ldots, e_k\}$
Let's expand the original expression of $f(T) - f(S)$:

$f(T) - f(S) = [f(S + e_1) - f(S)] + [f(S + e_1 + e_2) - f(S + e_1] + \ldots [f(T) - f(S + e_1 + e_2 + e_3 + \cdots + e_{k-1})]$

Each of the elements in [] can represented as $gain_1$, $gain_2$, etc. For example:

$gain_1 = [f(S + e_1) - f(S)]$
$gain_2 = [f(S + e_1 + e_2) - f(S + e_1)]$
$gain_3 = [f(S + e_1 + e_2 + e_3) - f(S + e_1 + e_2)]$

where

$gain_1 \geq gain_2 \geq gain_3 \geq \cdots \geq gain_k$

This idea is true because we know that $f(x)$ is a monotone submodular function. Let's suppose we are adding a specific element e_i to S and computing the marginal gain of adding that element e_i:

$f(S + e_i) - f(S)$

Using our previous definition, this expression is just $gain_1$. Let's look at the original inequality given to us one more time:

$f(T) - f(S) \leq \sum_{e \in T}^{k} (f(S + e) - f(S))$

Using the our previous equations that we founded we can expand the left and right hand sides into:

$gain_1 + gain_2 + \cdots + gain_k \leq gain_1 + gain_1 + \cdots + gain_1$

Since $gain_1$ is the highest possible value for the marginal utilities, this expression still holds true.

$\therefore$ We have proven the inequality

Q 1.3 (5 points) Let f be a monotone submodular function. Show that the following function is also submodular: $h(A) = \min(f(A), f(V/A))$ where V is the universal set of all elements.

Solution: Let's suppose we have two sets A and B where $A \subseteq B$. Let's run the function $h(x)$ on both sets:

$h(A) = min(f(A), f(V/A))$

$h(B) = min(f(B), f(V/B))$

If we add a new node $\{u\}$ to both A and B, and we run the function $h(x)$, we get:

$h(A \cup \{u\}) = min(f(A \cup \{u\}), f(V/(A \cup \{u\})))$
$h(B \cup \{u\}) = min(f(B \cup \{u\}), f(V/(B \cup \{u\})))$

We have two specific cases in this proof.

Case 1: $f(A) \leq f(V/A)$ **and** $f(B) \leq f(V/B)$
If this case is true, then our equations will look like:

$h(A) = f(A)$
$h(B) = f(B)$
$h(A \cup \{u\}) = f(A \cup \{u\})$
$h(B \cup \{u\}) = f(B \cup \{u\})$

Since we know $f(x)$ to be submodular, then $h(x)$ must also be submodular because they're equal to each other.

Case 2: $f(A) \geq f(V/A)$ **and** $f(B) \geq f(V/B)$
If this case is true, then our equations will look like:

$h(A) = f(V/A)$
$h(B) = f(V/B)$
$h(A \cup \{u\}) = f(V/(A \cup \{u\}))$
$h(B \cup \{u\}) = f(V/(B \cup \{u\}))$

When we subtract from a universal set, the larger the subtrahend, the smaller the result is. Therefore, since we also know that $f(x)$ is a monotone function, we know that

$f(V/(B \cup \{u\})) \leq f(V/B) \leq f(V/(A \cup \{u\})) \leq f(V/A)$

Since $f(X)$ is monotone submodular function, we can display the marginal gain with:

$f(V/B) - f(V/(B \cup \{u\})) \geq f(V/A) - f(V/(A \cup \{u\}))$
Let's multiply both sides by -1:

$-1 * [f(V/B) - f(V/(B \cup \{u\}))] \geq -1 * [f(V/A) - f(V/(A \cup \{u\}))]$
$-f(V/B) + f(V/(B \cup \{u\})) \leq -f(V/A) + f(V/(A \cup \{u\}))$

If we rearrange this inequality, we get:

$f(V/(B \cup \{u\})) - f(V/B) \leq f(V/(A \cup \{u\})) - f(V/A)$

This is the exact definition of submodular function, so if this case is true, $h(x)$ will be submodular:

$h(A \cup \{u\}) - h(A) \geq h(B \cup \{u\}) - h(B)$

Both cases adhere to the submodular definition.
$\therefore h(x)$ is submodular

Q 1.4 (5 points) **[Bonus]** Let f be a monotone submodular function and let g be a concave function. Show that the following function is also submodular: $h(A) = g(f(A))$.

Solution: The submodular function $f(x)$ will take in a set value for its x and output a real number. Additionally, function $g(x)$ will take in a real number for its x and output a real number.

$f(x) : 2^{|V|} \rightarrow \Re$
$g(x) : \Re \rightarrow \Re$

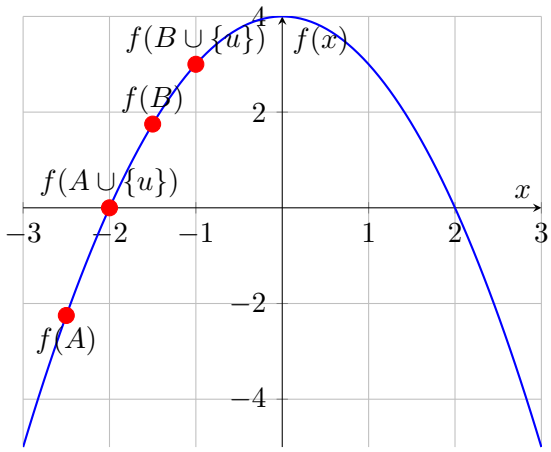
Let's suppose we have two sets A and B where $A \subseteq B$. If we run $h(x)$ on A, B, $A \cup \{u\}$, and $B \cup \{u\}$, we get:

$h(A) = g(f(A))$
$h(B) = g(f(B))$
$h(A \cup \{u\}) = g(f(A \cup \{u\}))$
$h(B \cup \{u\}) = g(f(B \cup \{u\}))$

Since we know $f(x)$ is a monotone function, we can say that

$f(A) \leq f(A \cup \{u\}) \leq f(B) \leq f(B \cup \{u\})$.

The output of the $f(x)$ function will be used as input for the $g(x)$. Let's say $g(x)$ is some arbitrary concave function. We can visualize some arbitrary outputs for the $f(x)$ function in a graph shown below:



Hi

We can clearly see that even though the distances between the two points of the normal set and set added with the new set are the same (0.5), the actual y distance is a lot larger. Hence,

$h(A \cup \{u\}) - h(A) \geq h(B \cup \{u\}) - h(B)$

One idea to keep in mind is that if the distance between $f(B)$ and $f(B \cup \{u\})$ is larger than the distance between $f(A)$ and $f(A \cup \{u\})$, it's possible that the submodularity function will not hold. However, the distance between $f(B)$ and $f(B \cup \{u\})$ can never be larger than the distance between $f(A)$ and $f(A \cup \{u\})$ because $f(x)$ is a monotone submodular function.

$\therefore h(x)$ is a submodular function

2. (30 points) Anomaly Detection

We will try to implement a simple anomaly detection algorithm for detecting outbreaks using ideas similar to the subset scan methods we saw in class. For this question, we will use the data from the CDC about the ILI burden across the 10 HHS regions and US national level for the past 21 years (2000-2020).

Typically, epidemiologists focus on specific weeks of a year where the flu is prevalent called a *flu season*. A flu season starts at week 40 of a given year and ends at week 20 of the next year. We enumerate the weeks of a Flu season as *Epiweeks*.

For example, the 2019-20 season starts at week 40 of 2019 (Epiweek 1) and ends at week 20 of the year 2020 (Epiweek 33). Since a year can have 52 or 53 weeks, a flu season can have 33 or 34 Epiweeks. The data for ILI is uploaded on Canvas as `ILINet.csv`. In this question, we refer to `% WEIGHTED ILI` of the csv file as ILI values.

Q 2.1 (3 points) Plot all the seasons in the dataset for the US National region starting from the 2000-2001 season to the 2019-2020 season. The y-axis is ILI and the x-axis is the Epiweek number of the season. (Plot all the weeks in one graph)

Solution:

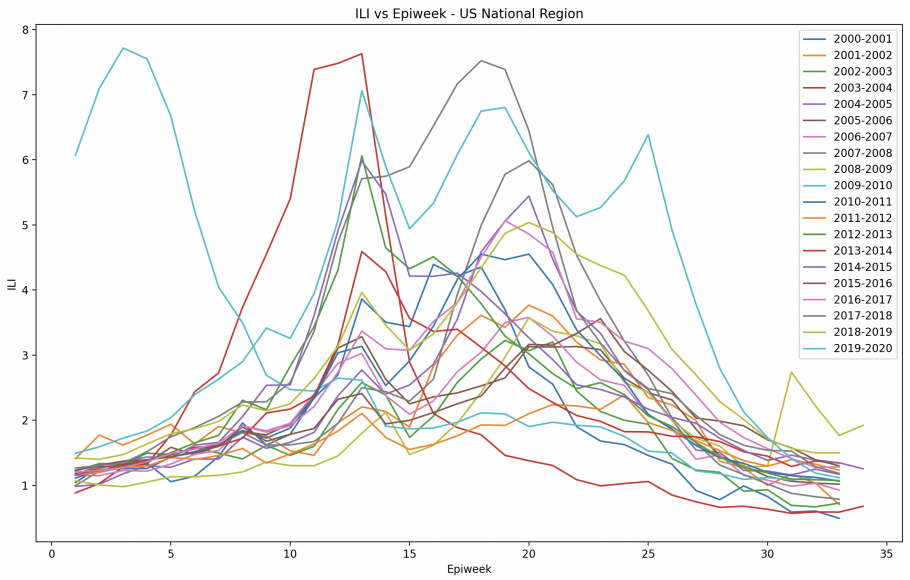


Figure 1: Q2.1 ILI Values for US National Region (all seasons)

Q 2.2 (10 points) We will use the 2019-2020 season as the test season for our outbreak detector. ILI usually varies from 0 to 8. Let's focus on only data from the US National region for this part. For each Epiweek, fit a Gaussian distribution over all values of ILI for that Epiweek from all the training seasons (2000-2001 season to 2018-2019 season), Output the mean and std-deviation of each Epiweek you get in a table. Also, submit the code.

Solution: Code is located in hw3/hw3/Q2/Q2.2

Table for mean and std-deviation of each epiweek is in hw3/hw3/Q2/outputs/Q2.2_Epiweek_mean_std.csv

Q 2.3 (5 points) In statistics, you may have heard about the '3 sigma' rule for anomalies i.e., anything beyond 3 standard deviations from the mean is unlikely and hence an anomaly [1]. Hence, use the mean and std-deviation you get in Q2.2 for each Epiweek (again focusing on US national) and apply this rule to compute which weeks in the 2019-20 season will be marked anomalous.

Solution: Anomalous Weeks:
10
11
12
13
14
15

[1] `https://en.wikipedia.org/wiki/68%E2%80%9395%E2%80%9399.7_rule`

Q 2.4 (2 points) Do these weeks 'make sense'? Anything surprising? Any speculation about what happened during these weeks that made them anomalous? Why were the other weeks possibly not anomalous?

Solution: Yes! These weeks makes sense. The 10th week of 2020 was the first week when COVID really started to pick up (first week of March). Early March isn't a flu season, so the epiweeks for early March in the previous seasons never had such high ILI values. As a result, these weeks were marked as anomalous because they were way above the average.

The other weeks of 2020 were possibly not anomalous because the ILI values were not so outlandish in comparison to previous flu seasons; they were probably a little bit above the mean but not 3 standard deviations over like in the anomalous weeks.

Q 2.5 (10 points) Repeat the same steps in Q2.2, and 2.3 for regions 2, 4, 7, 9, and 10, and finally write down the weeks you found anomalous for each of these regions.

Solution:

Code for Q2.2 portion is located in the folder with the respective region name and labeled as Q2.5-RegionX_Part1.py

Region 2: 11, 12, 13, 14, 15, 16, 17
Region 4: 48, 10, 11, 12, 13, 14
Region 7: 9, 10, 11, 12, 13
Region 9: None
Region 10: 51, 52, 12

3. (30 points) Sensors for detection in Network Model

We are going to empirically look at various strategies for selecting social network sensors to detect epidemic outbreaks. We will use the graph in `facebook.txt` [2] which contains a small subset of friendship network of a Facebook group. We will use the SI model with $\beta = 0.005$ to simulate the infection.

Q 3.1 (4 points) As a warmup, simulate the SI model for up to $T = 100$ time-steps for 100 runs and plot the average fraction of S, I vs t curve. Select 4 nodes at random to be infected at $t = 0$. Also plot the average number of daily infected people for $t = 0, \ldots, 100$. Report the average time t^* at which the maximum daily infections are maximum.

Hint: Use the implementation of SI model in `si_model.py` file. If you use another language you can convert the logic of the code in the file.

Solution:

[2] taken from `https://snap.stanford.edu/data/ego-Facebook.html`

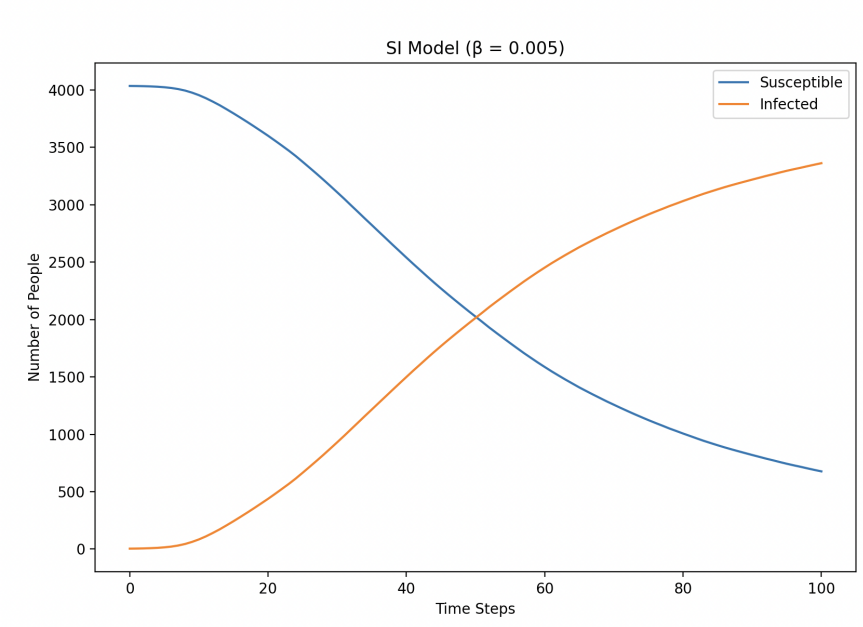


Figure 2: Q3.1 SI Model

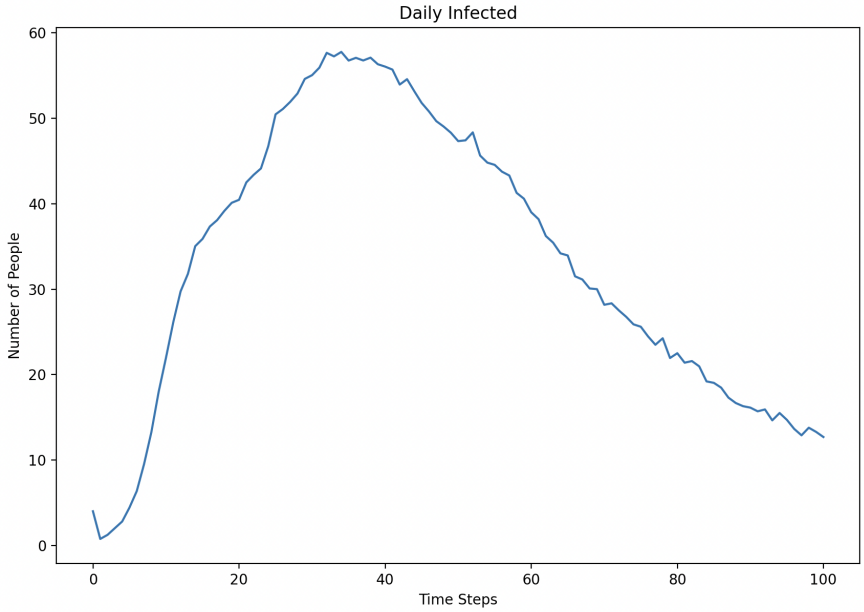


Figure 3: Q3.1 Daily Infections

Average Time $t^* = 34$

Q 3.2 (10 points) Since the graphs are large, it is not always feasible to keep track of the states of all nodes in practice. Therefore, we will select a smaller subset of nodes to track during the epidemic which we call sensors. We will implement three strategies for sensor selection:

- `RANDOM`: We choose k nodes uniformly at from the graph
- `FRIENDS`: We choose k nodes uniformly at random and for each of them we select a random friend. We will use these friends as the sensors.
- `CENTRAL`: We select the top k nodes with largest *eigenvector centrality* [3]. You can use functions like `nx.eigenvector_centrality_numpy`.

Set $k = 100$ for all strategies. The file `RandNodes.npy` contains a random list of nodes. Select the first k nodes from the list to implement the `RANDOM` strategy. For `FRIENDS` strategy, use the k selected nodes from `RANDOM` strategy to randomly select their friends. Simulate the SI model for $T = 100$ steps and note the fraction $\tilde{I}(t)$ of the sensors that are infected at each time step for each strategy. Note that sensors can also be infected at $t = 0$.

Plot average $\tilde{I}(t)$ vs t for all three strategies averaged over 20 runs. Also plot the average number of daily infections $\tilde{I}_d(t) = \tilde{I}(t) - \tilde{I}(t-1)$ over time.

Solution:

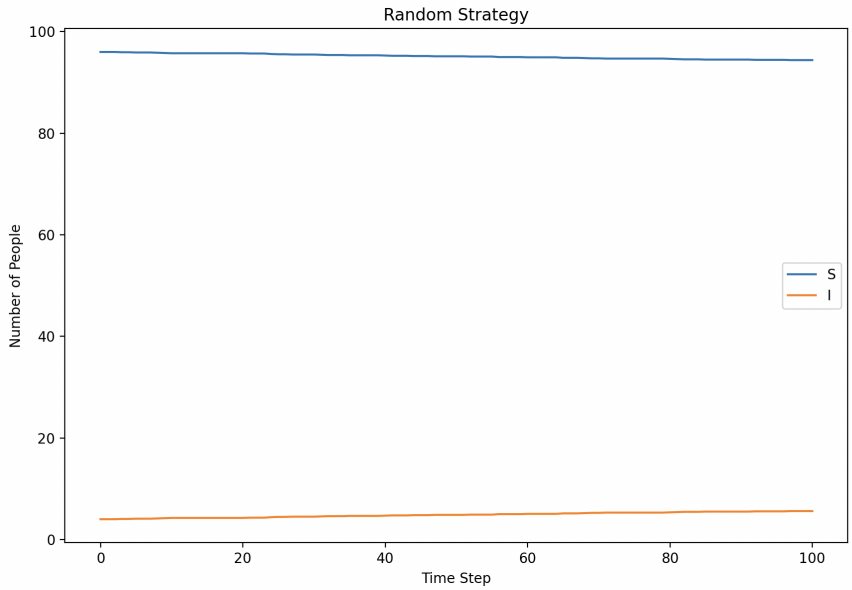


Figure 4: Q3.2 Random Strategy $\tilde{I}(t)$ vs t

[3] `https://en.wikipedia.org/wiki/Eigenvector_centrality`

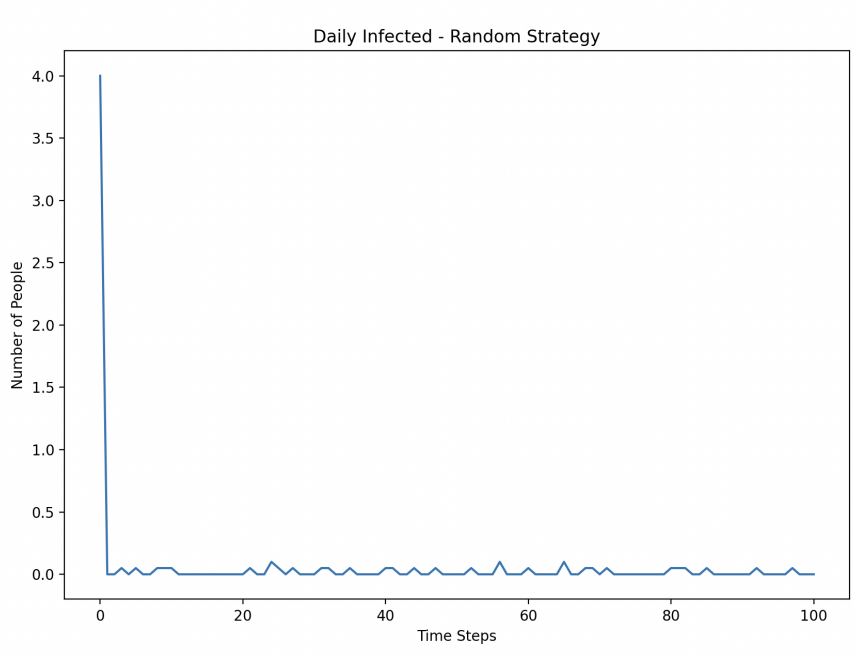


Figure 5: Q3.2 Random Strategy Daily

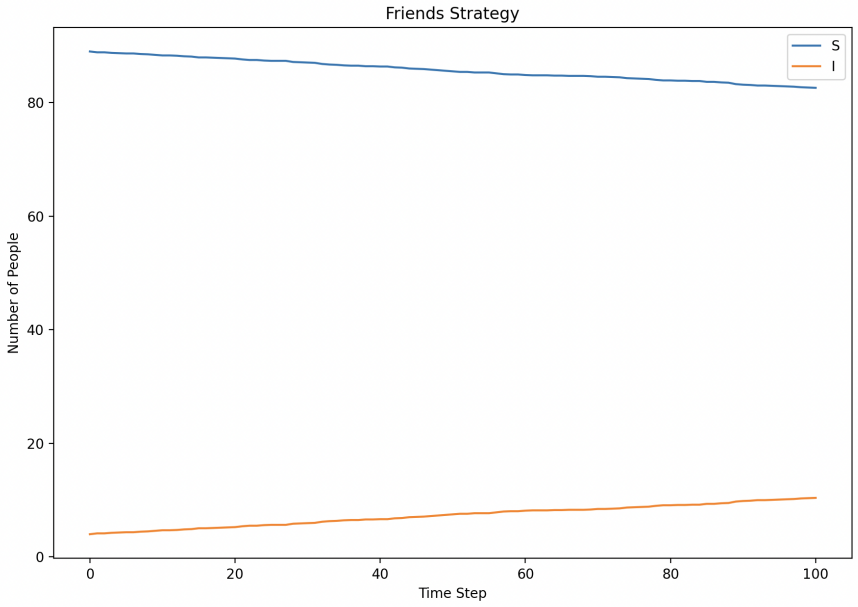


Figure 6: Q3.2 Friends Strategy $\tilde{I}(t)$ vs t

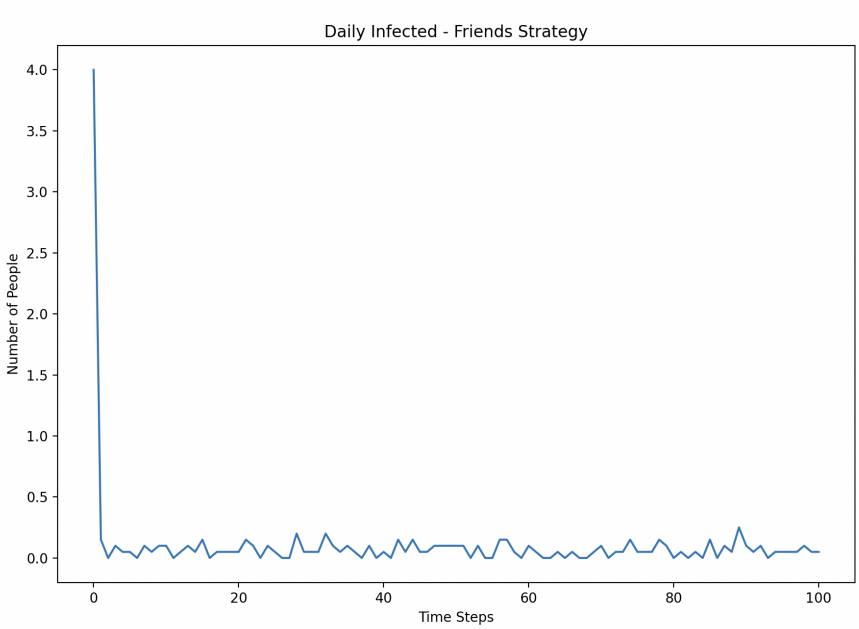


Figure 7: Q3.2 Friends Strategy Daily

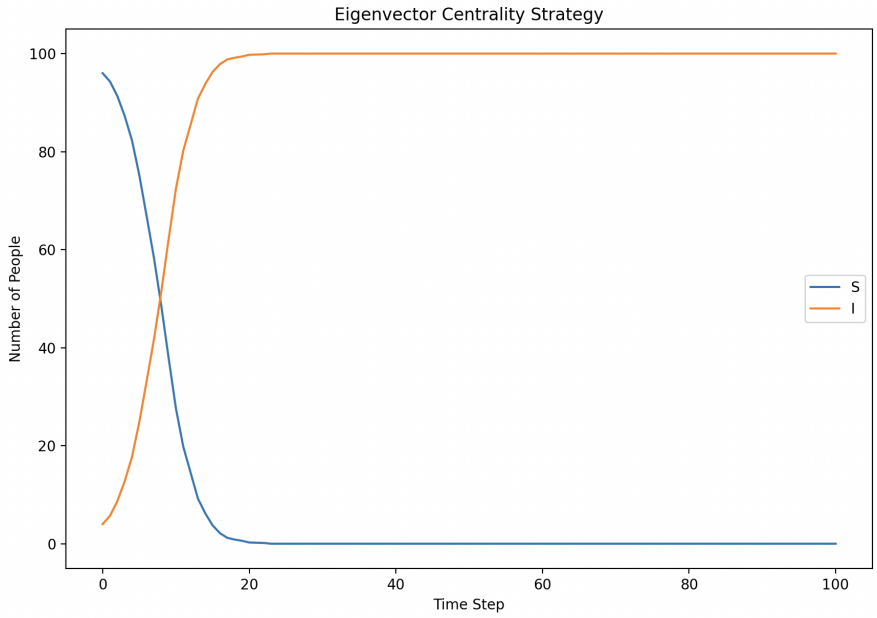


Figure 8: Q3.2 Central Strategy $\tilde{I}(t)$ vs t

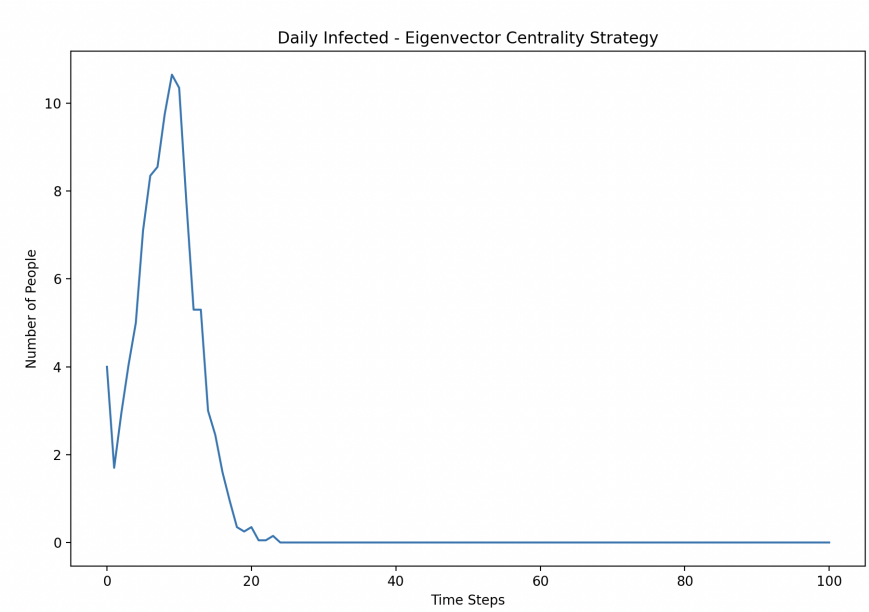


Figure 9: Q3.2 Central Strategy Daily

Q 3.3 (6 points) Report the peak time $\tilde{t}^*$ and peak daily infection for all 3 strategies (the time t where $\tilde{I}_d(t)$ is maximum is peak time).

The value $t^* - \tilde{t}^*$ is the *lead time* i.e., the difference in time between the detection of epidemic peak among sensors and the time when it peaks in the entire population. Report the lead time for all 3 strategies.

Solution:
Peak Time - **Random**: 97
Peak Daily Infection - **Random**: 0
Peak Time - **Friends**: 100
Peak Daily Infection - **Friends**: 0
Peak Time - **Eigenvector Centrality**: 23
Peak Daily Infection - **Eigenvector Centrality**: 9

Lead Time - **Random:** $34 - 97 = -63$
Lead Time - **Friends:** $34 - 100 = -66$
Lead Time - **Eigenvector Centrality:** $34 - 23 = 11$

Q 3.4 (4 points) Compare the lead-time of various strategies and explain the difference in lead-time of various strategies.

Solution:
The lead time for the random strategy was -63 days, meaning the epidemic peak was predicted 63 days later than the peak in the overall population. In this strategy, the sensor nodes are chosen completely randomly. This strategy doesn't prioritize choosing sensors that are well-connected in the population. As a result, these sensors may not be exposed to the epidemic as quickly as sensors that are more central

in the population, leading to delayed epidemic peak detection.

The lead time for the friends strategy was -66 which was slightly worse than the lead time for the random strategy. This strategy falls into the same pitfall as the random strategy because randomly choosing friends of individuals as sensors may not guarantee the friends are well-connected in the population. As a result, we get the late detection time of the peak.

The lead time for the centrality strategy was +11 days, meaning we were able to predict the peak of the pandemic 11 days before the actual peak occurred in the population. This is great because now we know when the peak can possibly happen in the population and way more useful to policy makers. The reason the centrality strategy works so well is because it chooses the most well-connected nodes as the sensors. The sensor nodes essentially experience the epidemic earlier, allowing for earlier detection the epidemic within the population as whole.

Q 3.5 (6 points) Now repeat Q 3.2 for $k = 50$ and $k = 500$. For each strategy submit a $\tilde{I}(t)$ vs t plot comparing different values of k. How does the lead time change with a value of k for each strategy?

Solution:

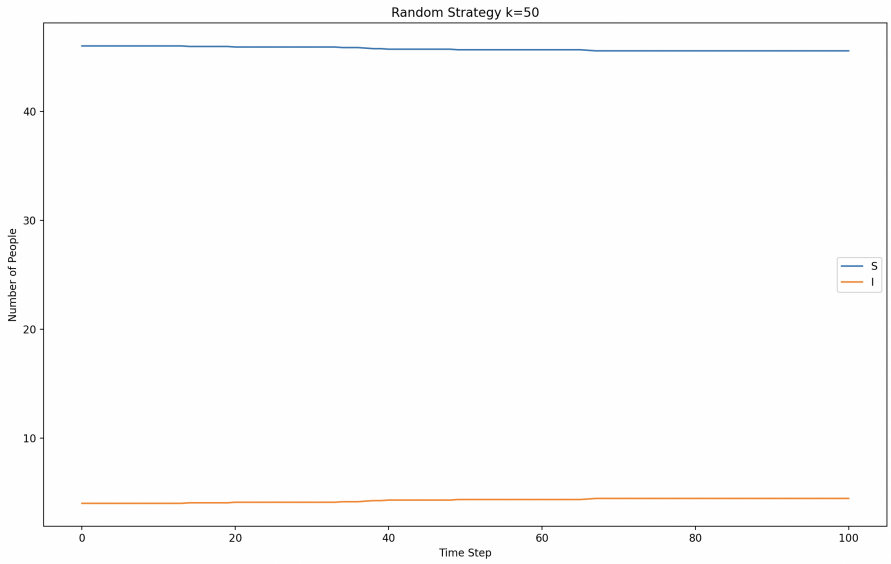


Figure 10: Q3.5 k=50 Random $\tilde{I}(t)$ vs t

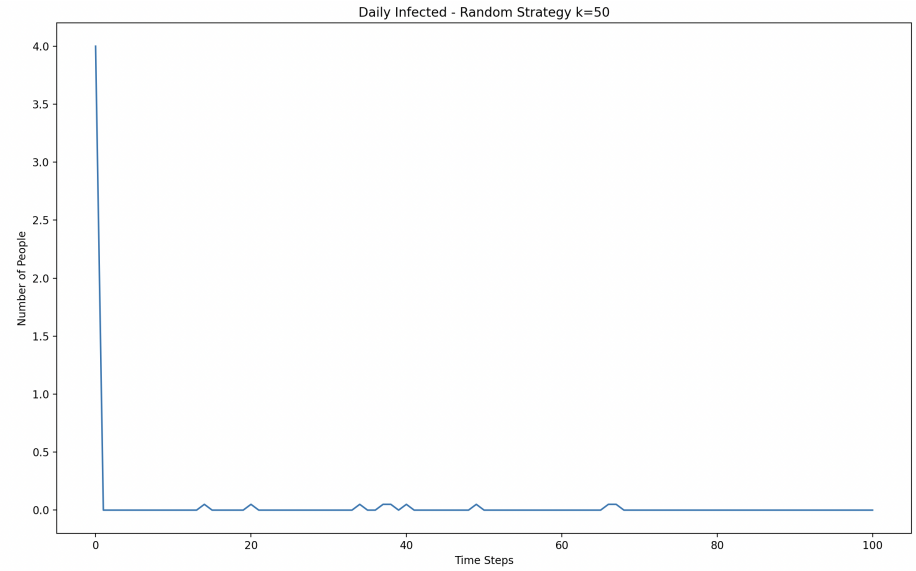


Figure 11: Q3.5 k=50 Random Daily Infections

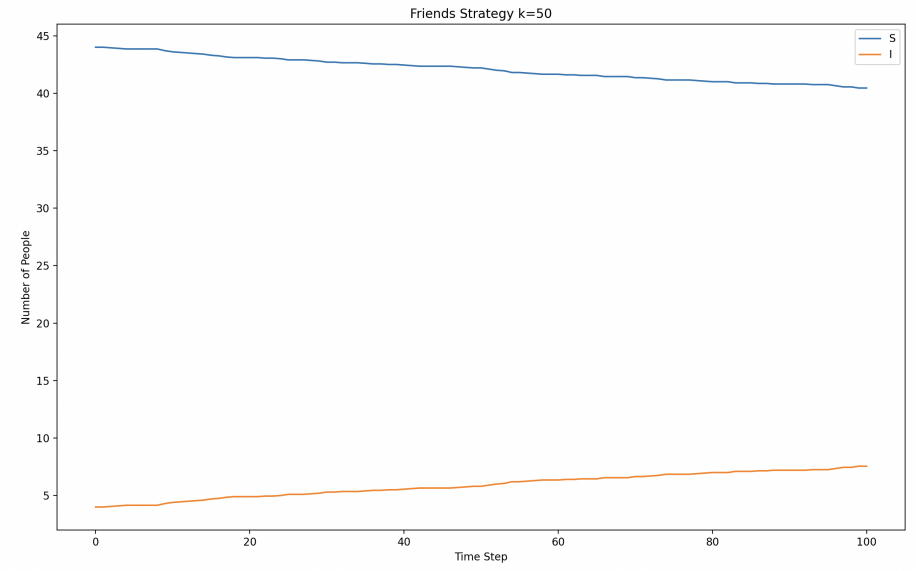


Figure 12: Q3.5 k=50 Friends $\tilde{I}(t)$ vs t

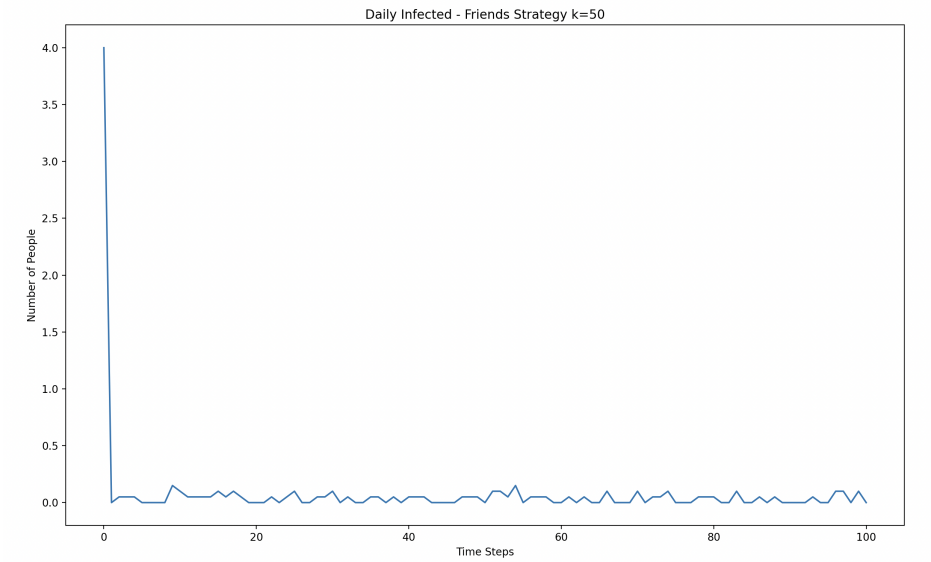


Figure 13: Q3.5 k=50 Friends Daily Infections

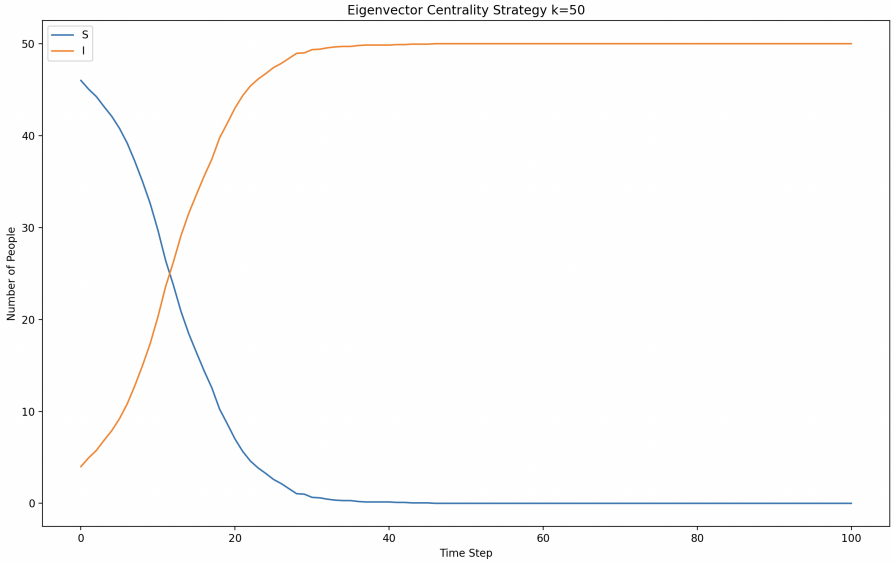


Figure 14: Q3.5 k=50 Centrality $\tilde{I}(t)$ vs t

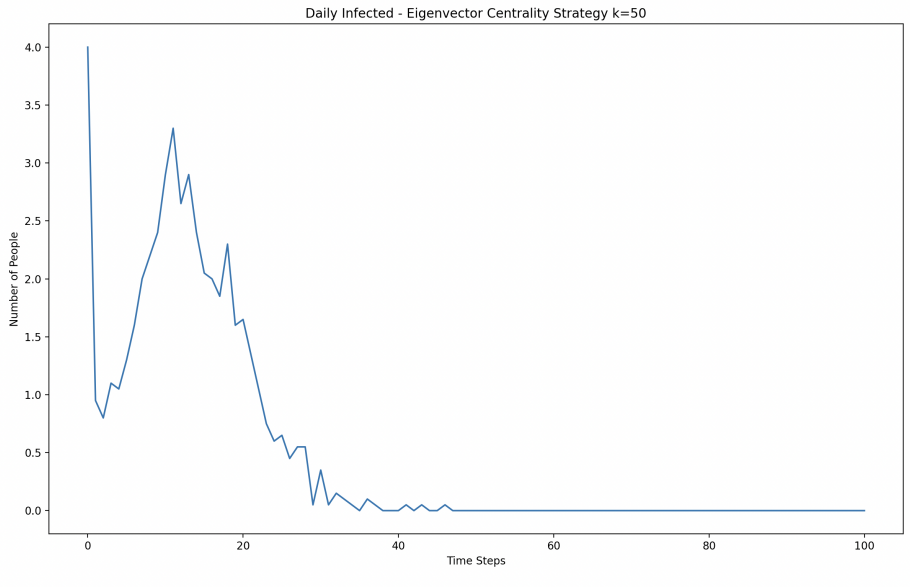


Figure 15: Q3.5 k=50 Centrality Daily Infections

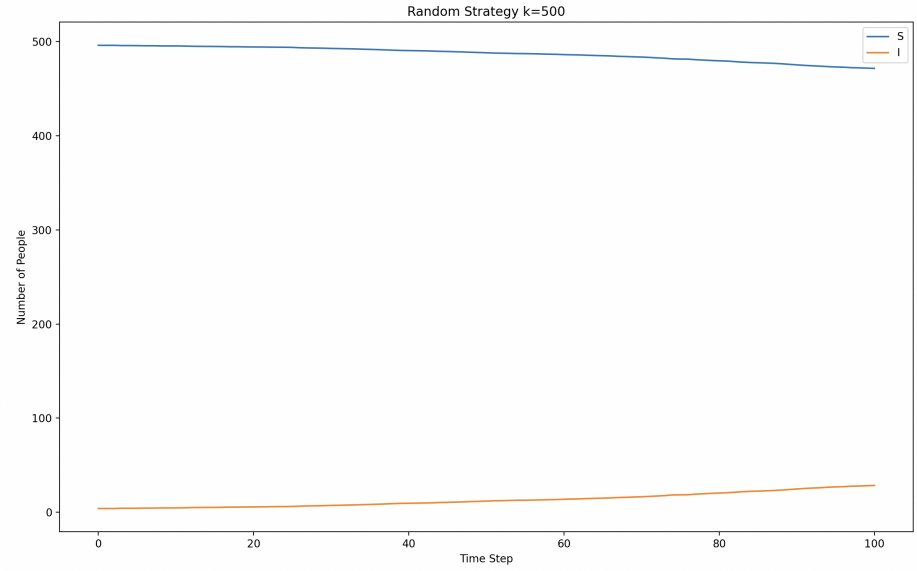


Figure 16: Q3.5 k=500 Random $\tilde{I}(t)$ vs t

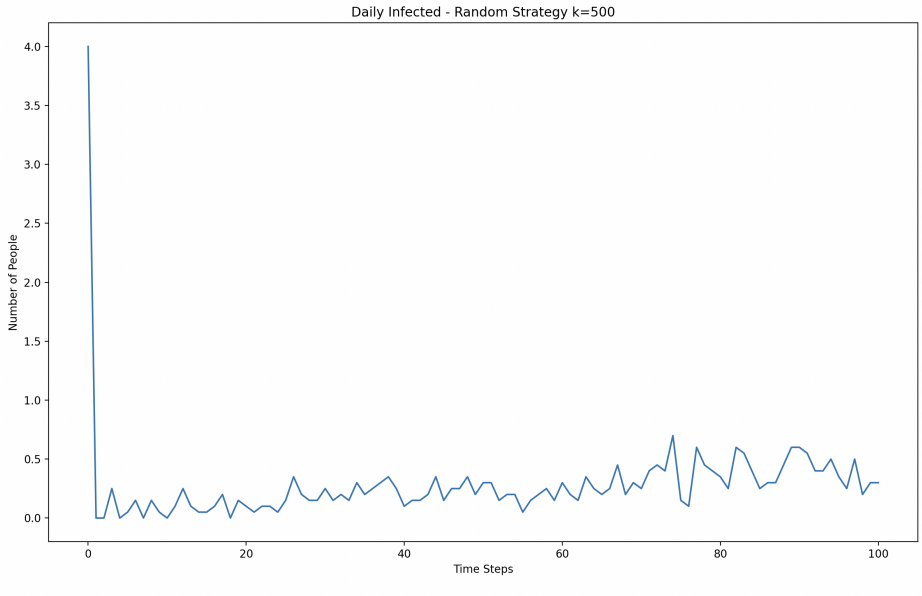


Figure 17: Q3.5 k=500 Random Daily Infections

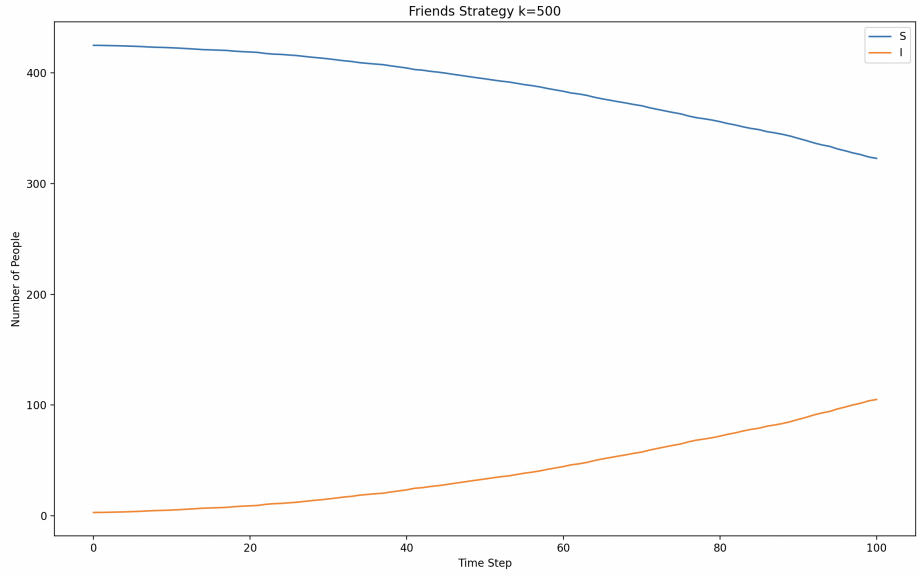


Figure 18: Q3.5 k=500 Friends $\tilde{I}(t)$ vs t

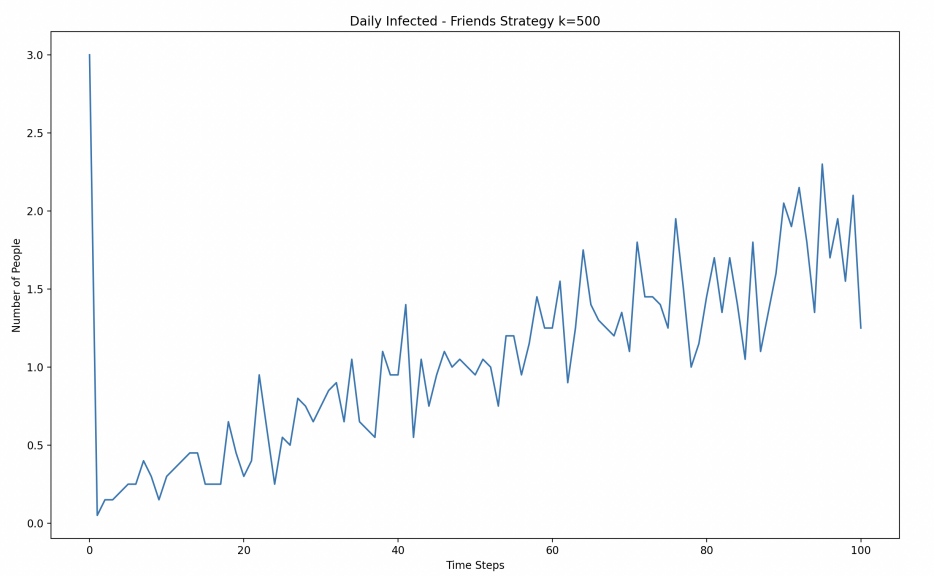


Figure 19: Q3.5 k=500 Friends Daily Infections

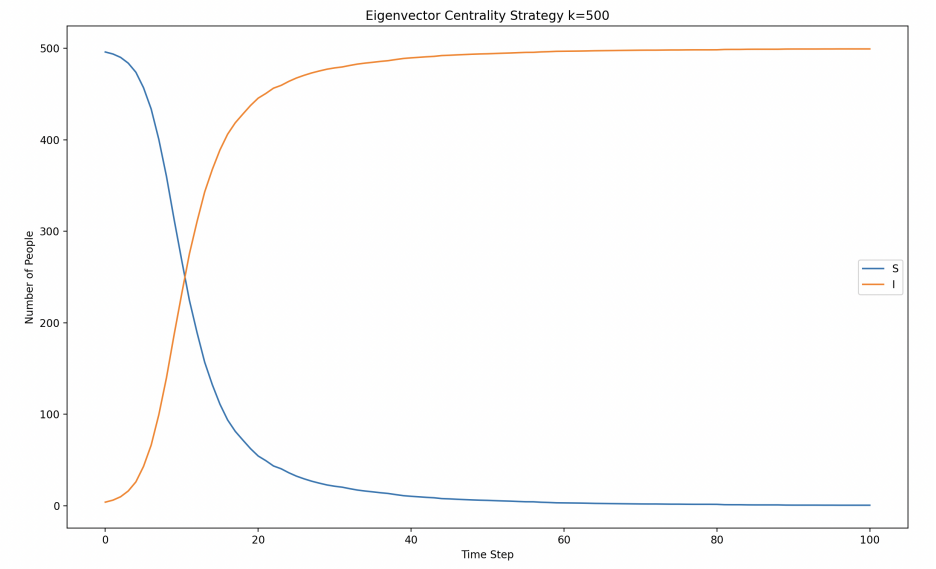


Figure 20: Q3.5 k=500 Centrality $\tilde{I}(t)$ vs t

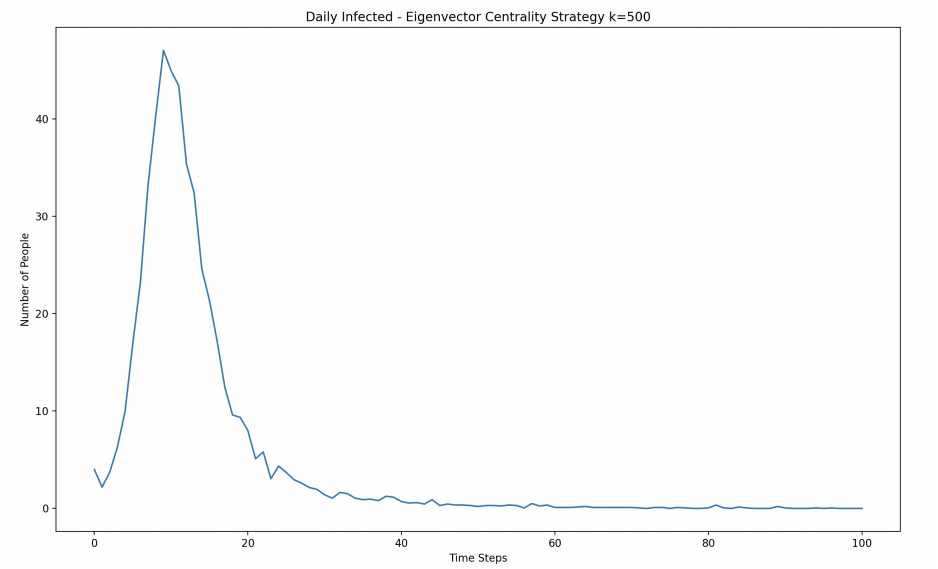


Figure 21: Q3.5 k=500 Centrality Daily Infections

The lead time seems to improve as k increases for each strategy.
Here are the lead time for k=50:

Lead Time - **Random:** $34 - 67 = -33$
Lead Time - **Friends:** $34 - 99 = -65$
Lead Time - **Eigenvector Centrality:** $34 - 46 = -12$

Here are the lead times for k=500:

Lead Time - **Random:** $34 - 100 = -66$
Lead Time - **Friends:** $34 - 100 = -66$
Lead Time - **Eigenvector Centrality:** $34 - 96 = -62$

It seems like the lead times for k=50 are better than the lead time for k=500. This is most likely because the chosen sensors have a closer network with each other when k is lower.

4. (26 points) Flu Surveillance using Google Symptoms Data

We will study the efficacy of Google Symptoms Data [4] as a source of Flu surveillance. We measure the usefulness of a signal using the correlation with ILI signals collected by CDC [5]. Specifically, we will look at the 2018-19 season (datasets can be downloaded from Canvas).

Q 4.1 (3 points) We will start by considering the state of Georgia. Extract the `% Unweighted ILI` from `ILINet_states.csv`. Plot the weekly `Unweighted ILI` of Georgia over the 2018-19 flu season. (Refer to Q2 for details on flu seasons.)

Solution:

[4] `https://pair-code.github.io/covid19_symptom_dataset/`
[5] `https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html`

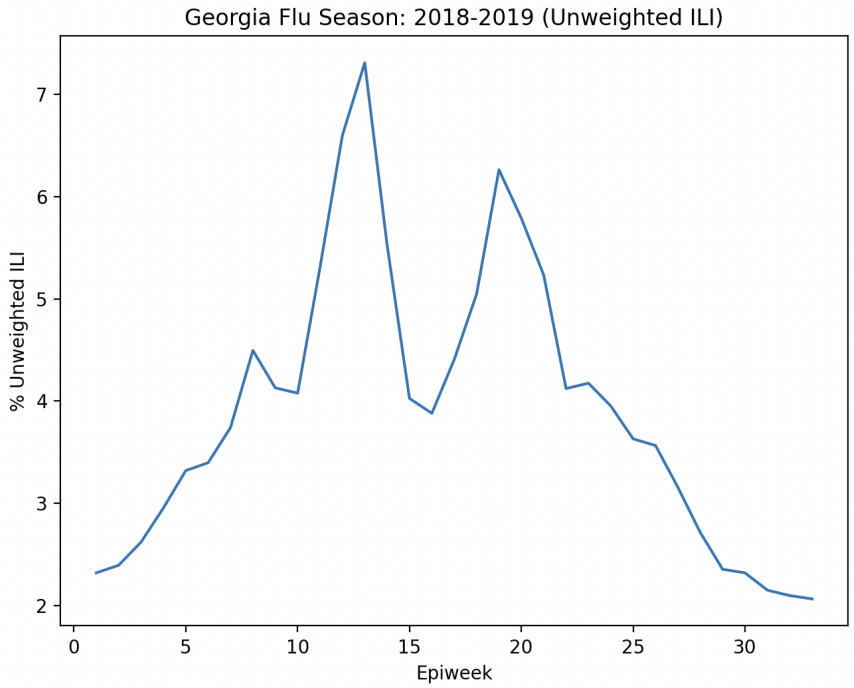


Figure 22: Q4.1 Unweighted ILI GA 2018-2019

Q 4.2 (10 points) CDC lists various symptoms for Flu [6]. We will consider the following symptoms: `Fever, Low-grade fever, Cough, Sore throat, Headache, Fatigue, Muscle weakness`.

Extract the Symptoms trends for each of these sympotms from the files `2018_symptoms_dataset.csv` and `2019_symptoms_dataset.csv` over the weeks of 2018-19 seasons. Submit a single plot showing the trends of all the symptoms over the 2018-19 seasons with the x-axis showing the Epiweeks and the y-axis the symptom trend values.

Solution:

[6] `https://www.cdc.gov/flu/symptoms/symptoms.htm`

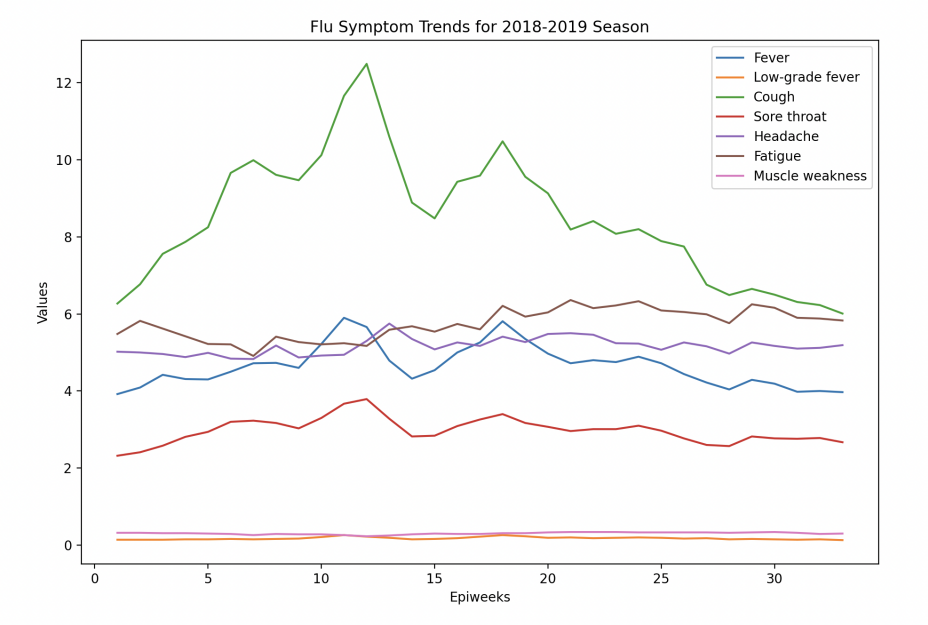


Figure 23: Q4.2 Symptom Trend Values GA 2018-2019

Q 4.3 (3 points) We will use Pearson Correlation Coefficient (PCC) [7] to measure how correlated each of symptom's trend is to ILI. Evaluate PCC for each of the symptoms with ILI for the 2018-19 season in Georgia. You may use functions like `scipy.stats.pearsonr` to evaluate PCC. Also, submit the code.

Solution: Code is located in hw3/hw3/Q4/Q4.3.py

Here are the PCC values for each of the symptom trend values in Georgia:

PCC Fever: 0.74235419732904
PCC Low-grade fever: 0.6901377920327422
PCC Cough: 0.8193518750226271
PCC Sore Throat: 0.7270191635280946
PCC Headache: 0.5721409837315692
PCC Fatigue: -0.10449709655195571
PCC Muscle: -0.45363265775879963

Q 4.4 (10 points) Repeat Q4.1 and Q4.2 for following states: `California, Texas, New York, Alaska` and `Mississippi`. Each state including Georgia, also reports the symptom with the highest PCC along with the value of PCC. Can you think of any reasons for the differences in PCC across these states?

Solution:
Highest PCC Symptom - Georgia: Cough (0.8193518750226271)

[7] `https://en.wikipedia.org/wiki/Pearson_correlation_coefficient`

1 Alaska:

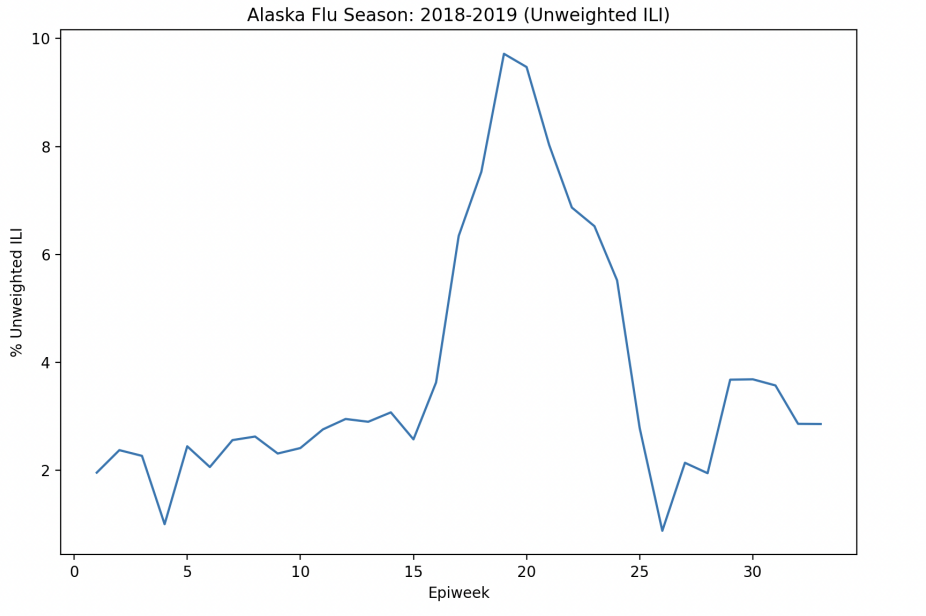


Figure 24: Unweighted ILI Alaska 2018-2019

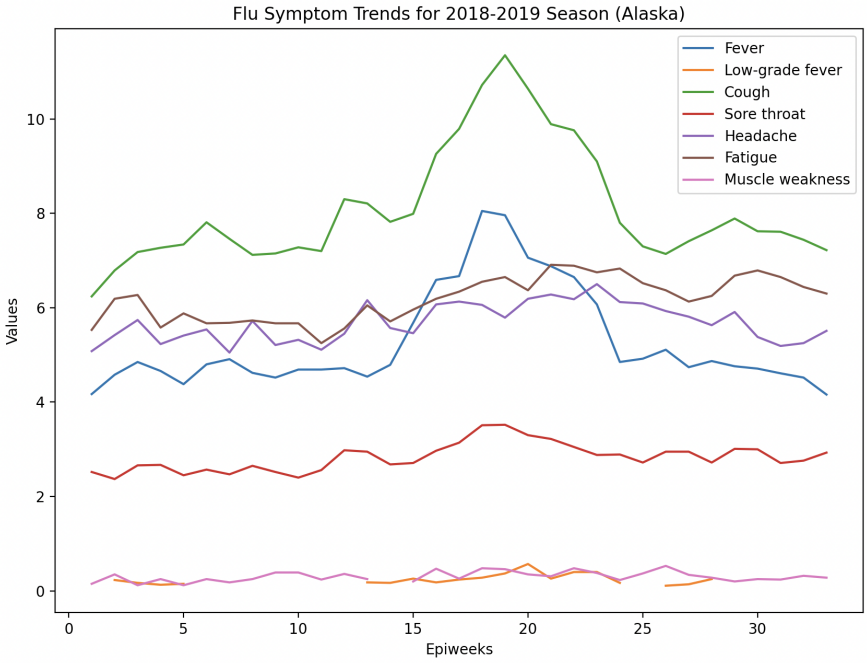


Figure 25: Symptom Trend Values - Alaska

Highest PCC Symptom: Cough (0.9126747036854097)

2 California:

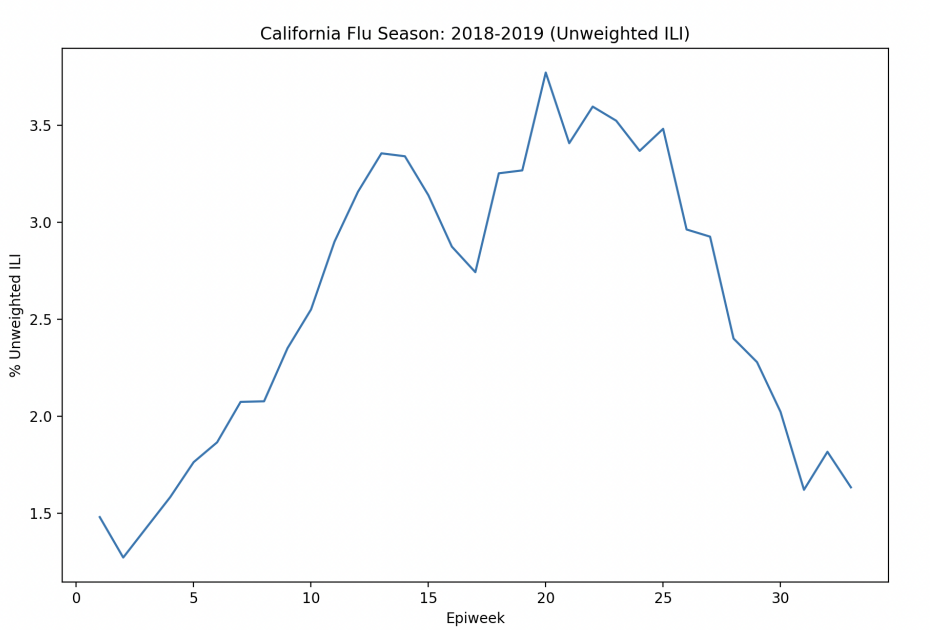


Figure 26: Unweighted ILI California 2018-2019

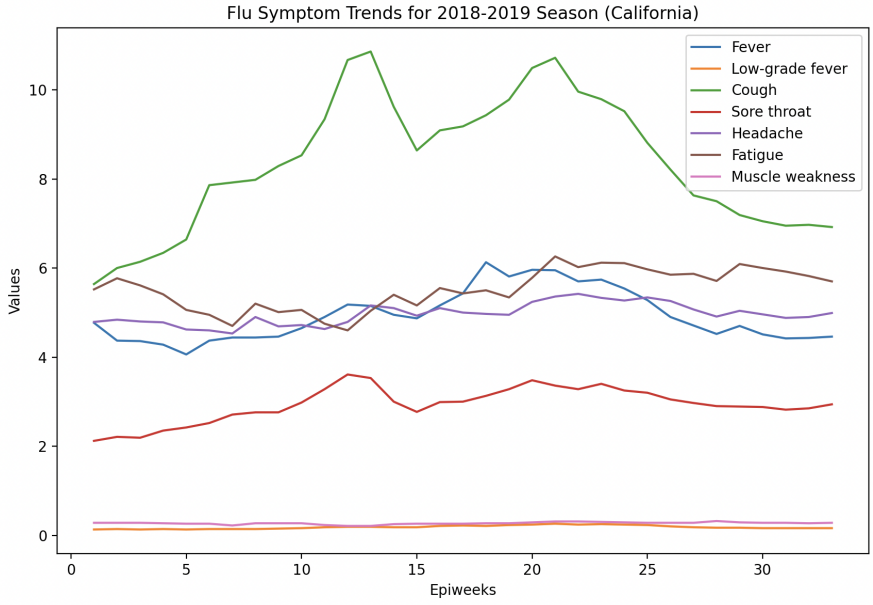


Figure 27: Symptom Trend Values - California

Highest PCC Symptom: Cough (0.9138491381378956)

3 Mississippi:

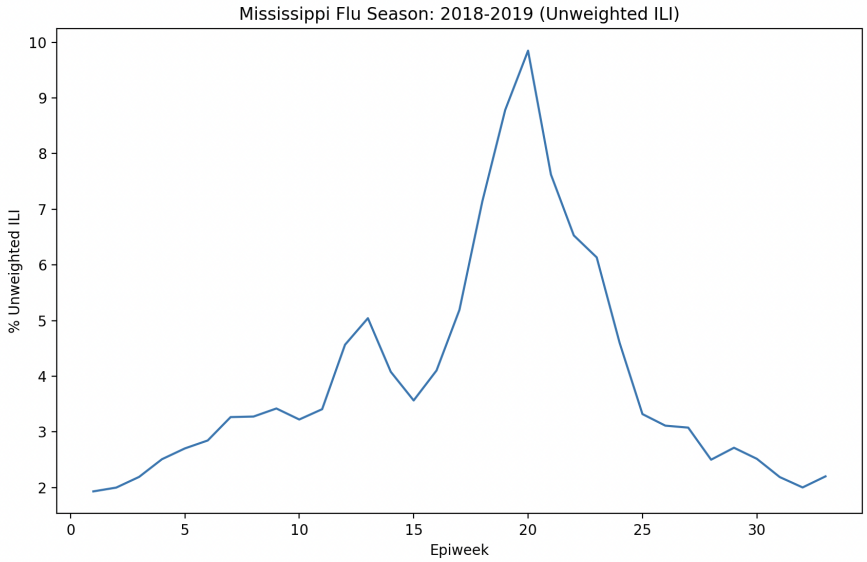


Figure 28: Unweighted ILI Mississippi 2018-2019

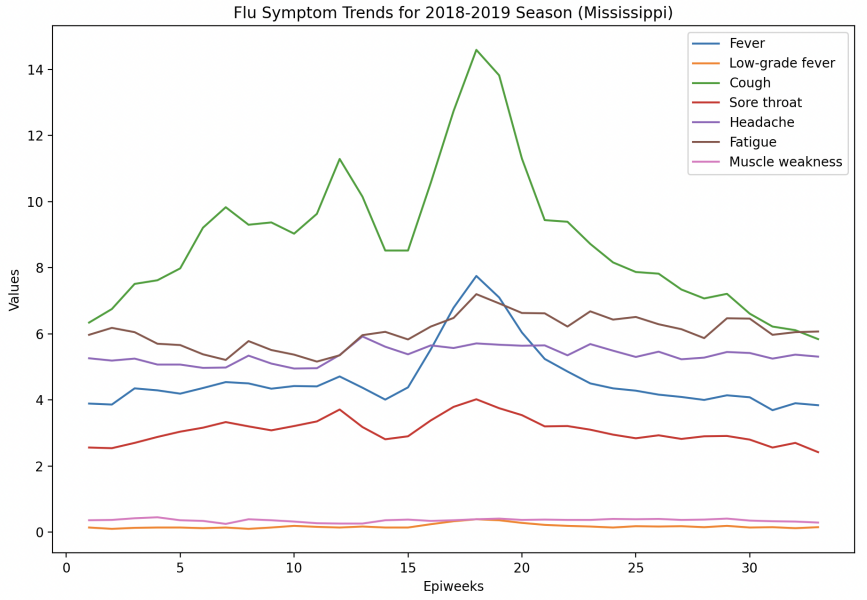


Figure 29: Symptom Trend Values - Mississippi

Highest PCC Symptom: Fever (0.7717618073197053)

4 New York:

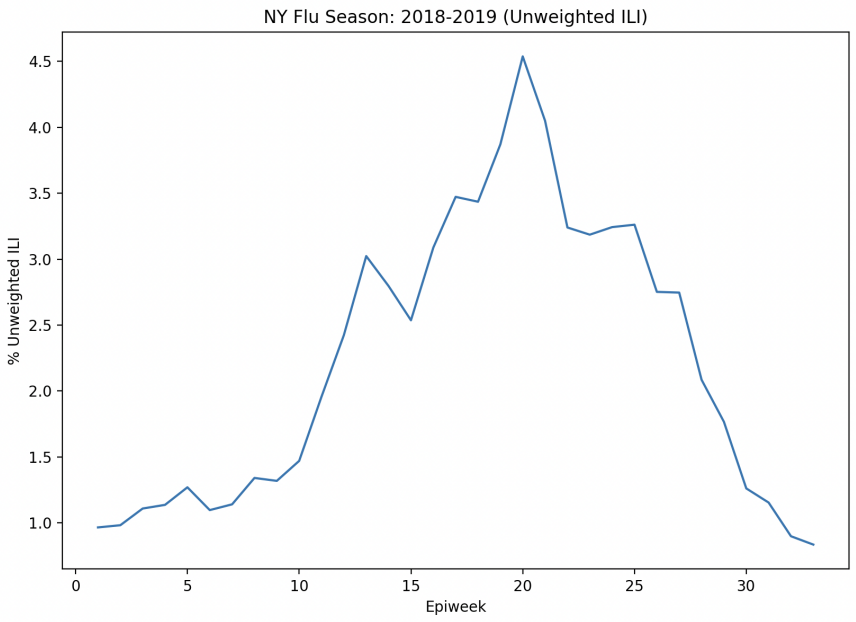


Figure 30: Unweighted ILI NY 2018-2019

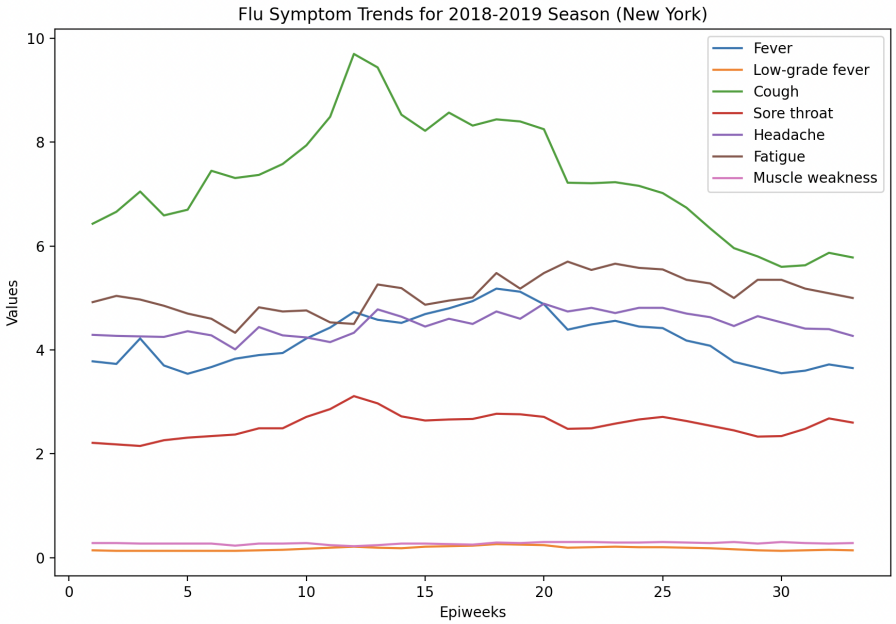


Figure 31: Symptom Trend Values - NY

Highest PCC Symptom: Low-grade Fever (0.8962088820761804)

5 Texas:

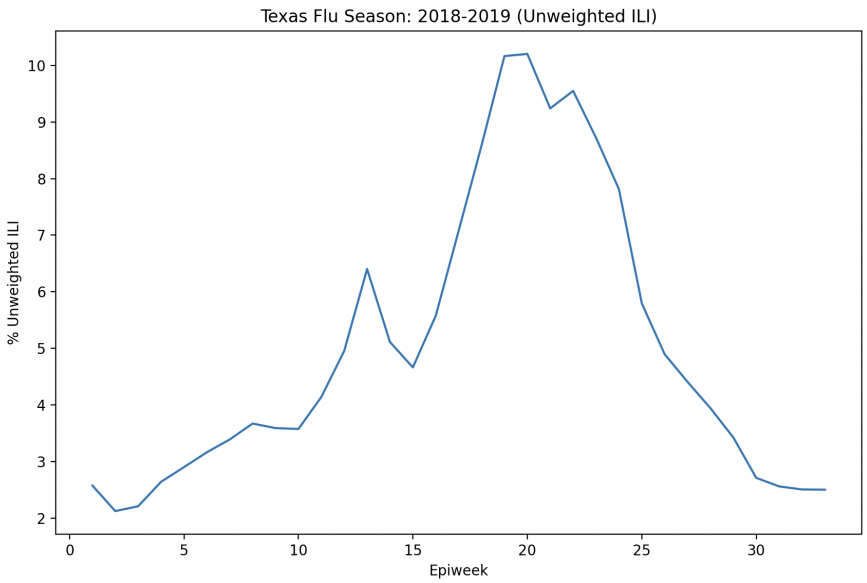


Figure 32: Unweighted ILI Texas 2018-2019

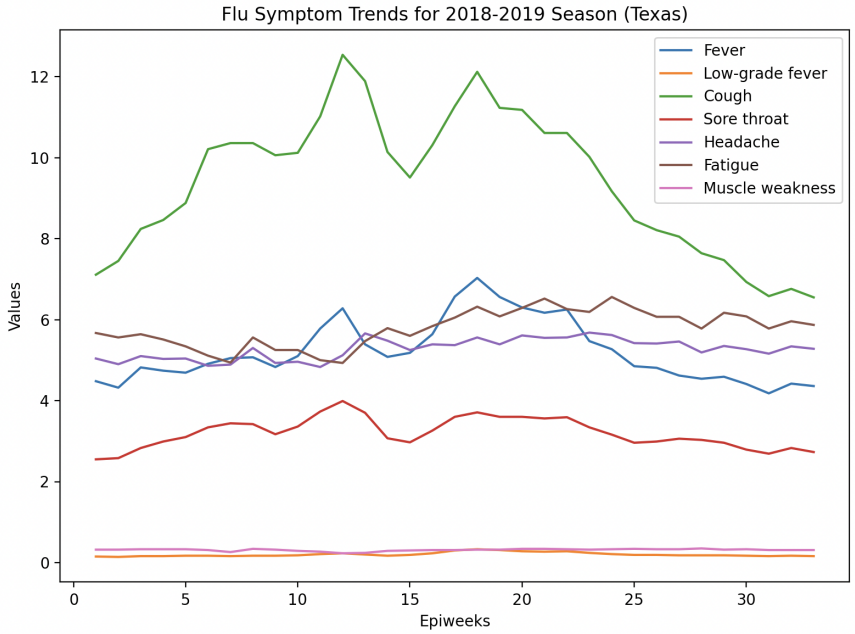


Figure 33: Symptom Trend Values - Texas

Highest PCC Symptom: Low-grade Fever (0.8954566944028768)

Q 4.5 (6 points) **[Bonus]** There may be some lead-time between reported ILI and the symptoms trend. Here, we define the lead-time t_s of a symptom s to be the value of t' that maximizes the PCC between ILI time-series from week $t' + 1$ to T and time-series of symptom s from 1 to $T - t'$ where $T = 52$ week. Intuitively, we measure the delay between the symptoms signal and the ILI signal.

For the most correlated symptoms you calculated in Q4.3 for each of the 6 states, find the lead time for the respective symptom. Submit the code.

Solution: Code for Q4.5 is located in each respective state labeled State_Name/Q4.5_State_Name.py

Georgia Lead Time = 1
Alaska Lead Time = 1
California Lead Time = 1
Mississippi Lead Time = 2
New York Lead Time = 31
Texas Lead Time = 31