Homework #3
Diffusion Models and Network Construction
Due: May 29th, 2020

- The assignment should be delivered digitally by email. Your message should be sent to **pribeiro@dcc.fc.up.pt** with subject *"[NetSci HWK3] - FirstName Last Name StudentNumber"*
- Your delivery should be a **zip file**, containing a **PDF report with the answers** and all additional files that were used for producing those answers
- You may work in a (small) group, but you should do your own **individual writeup**. This means you can collaborate by talking about the exercises, but **you should not copy answers or code**.
- Please **acknowledge any help you got** and state any references you consulted (including internet pages) and any students with whom you talked about the exercises.
- Answers should be **submitted until 23:59 of the due date**. Up to 24h of delay will get you a 25% penalty. 24h to 48h of delay will get you a 50% penalty. After 48h your work will not be counted.

(always explain how you reached each answer, so that I can understand your train of thought; use whatever programming language you prefer, but you must include in the zip any code you created)

---

1. Diffusion and Cascading Behavior

The networks.

The first two groups of questions on information diffusion will take place in a fictitious town for which you have been given two possible versions of the (undirected) social graph of citizens: `graph1.txt` and `graph2.txt`. Each of these graphs has 10,000 nodes (with ids between 0 and 9999), and is given by an edge list file.

(a) Read the two graphs and **plot their degree distributions** in linear scale and in log-log scale. Although both graphs have roughly the same number of edges, degree distributions are actually very different. Make a **brief comment on what distributions you are observing and what graph model might have been used** to create each of the graphs.

---

2. Decision Based Models

(Note: this exercise is heavily based on a homework assignment from Stanford, including the 2 graphs)

The model

The city will have an election between candidates A and B. You are the network scientist working for candidate A, responsible for results forecasting and voters acquisition. Most citizens have already decided who they are voting for: 40% know they will vote for A (nodes with last digit in $\{0, 1, 2, 3\}$), 40% know they will vote for B (nodes with last digit in $\{4, 5, 6, 7\}$), and the remaining 20% are undecided (nodes with last digit in $\{8, 9\}$).

The undecided voters will go through a period where they choose a candidate each day based on the **majority** of their friends. The **decision period** works as follows:

- The graphs are initialized with every voter's initial state (A, B, or undecided).
- In each iteration day, for every undecided voter, if the majority of their friends support A, they now support A. If the majority of their friends support B, they now support B. "Majority" for A means that strictly more of their friends support A than the number of their friends supporting B, and vice versa for B (ignoring undecided friends).
- After deciding to vote for a certain candidate, a citizen will never change his/her mind again. Only undecided voters may change on the next iteration.
- When doing an update, you should always use the voting values from the previous iteration, that is, you should take into account the voting preferences from the previous day.
- The process continues until the votes **converge,** that is, there is no node that changed from undecided to one of the candidates. Note that this means that at most you will have 2000 iterations (i.e., the number of initial undecided nodes).

(a) **Basic forecast**

Make the initial vote configurations and simulate the decision model given above. **Which candidate wins in graph1, and by how many votes? Which candidate wins in graph2, and by how many votes? How many iterations did it take to converge in each graph?**

(b) **TV advertising**

You have 10,000€ to spend on TV advertising. Unfortunately only 100 citizens pay enough attention to TV: those with ids from 3000 to 3099. However, your ads are extremely persuasive, so anyone who sees the ad is immediately decides to vote for candidate A regardless of his/her previous decision. You may spend 1,000€ at a time on ads. The first 1,000€ reaches voters 3000–3009, the second 1,000€ reaches voters 3010–3019, and so on. In other words, the total of k€ in advertising would reach voters with ids from 3000 to $3000 + \frac{k}{100} - 1$. Note that the TV advertising affects all of the voters (not just those undecided) and that after voters are persuaded by your ads, they never change their minds again.

Simulate the effect of advertising spending on the two possible social graphs. First, read in the graphs again and assign the initial configurations as before. Now, before the decision process, you purchase k€ of ads and go through the decision process of counting votes until convergence.

For each of the two social graphs, **plot** k€ (the amount you spend) on the x-axis (for values $k = \{0, 1000, 2000, \ldots, 10000\}$ and the number of votes for A minus the number of votes for B on the y-axis. **Put these on the same plot**. What's the **minimum amount you can spend to win the election** in each of the two social graphs?

(c) **A fancy dinner**

Now, instead of TV advertising, you had another great idea, and decide to invite some of the voters to a fancy dinner hosted by your candidate. The event will cost 500€ per invited person, but after the dinner all of these invited citizens will be persuaded and will vote for candidate A (not just those undecided), never changing their minds again.

You gained access to the social graph and in order to be effective you decide to invite the k people with the highest degree in the social graph (regardless of their initial voting configuration). If there are ties between citizens with the same degree, the lowest id gets chosen first.

For each of the two social graphs, **plot** k€ (the amount you spend) on the x-axis (for values $k = \{0, 500, 1000, 1500, \ldots, 10000\}$ and the number of votes for A minus the number of votes for B on the y-axis. **Put these on the same plot**. What's the **minimum amount you can spend to win the election** in each of the two social graphs?

Give a **brief comment** on how the **topology** of the graphs (as seen on the first question) influences and **explains the results** you obtained in this question.

3. Probabilistic Models

The Model

Consider that for your town the undirected social graph is represented by `graph2.txt`.
For this question we will consider a stochastic network epidemic model, similar to the traditional SIR model, with parameters β (probability of infection) and δ (probability of recover), along with the following **rules of simulation**:

- Nodes can be in one of three states **S***usceptible*, **I***nfected* or **R***ecovered*.
- At the start of the simulation all nodes are *susceptible*
- At each iteration day an *infected* node will have β probability of spreading its infection to each of its *susceptible* neighbors.
- After each iteration day, a node which was already *infected* will have a probability of δ to become *recovered*.

For all the following questions always run at least $k = 3$ simulations and report the average of the required values (please comment on the value of k you used on your results).

(a) **Basic plot**
Initialize 5 random nodes as *infected* (all other nodes are initially *susceptible*) and run the simulation for 100 days with $\beta = 4\%$ and $\delta = 1/14$ (which implies an `expected number` of 14 days with the infection). **Make a plot of the number of *susceptible*, *infected* and *recovered* nodes at each day (all lines in the same plot).** Is the plot what you were expecting? Why? What is the **peak number of infected nodes**? At what day does that peak occur?

(b) **Basic statistics**
For the previous simulation, plot the **number of new infected nodes per day**. At **what day does this number peak**? If is before or after the peak of infected nodes? Can you offer a **brief explanation** of that time difference and why there is a single peak?

(c) **Estimating R_0**
The R_0 indicates the average number of nodes that get infected from each node with an infection (assuming no immunity). Show **how many nodes were infected before day 6** (a) **and how many nodes were infected by those nodes** (b). Give an **estimation of that initial R_0 for those spreader nodes as $\frac{b}{a}$**. Is this R_0 bigger or smaller than 1? Does this help explain what happens? Can you estimate R (the reproductive number) for the nodes that get infected after the peak infection day and comment on the value you obtained?

(d) **Lowering the curve**
Imagine that the city is prepared for the infection and adopts right from the beginning **social distancing that lowers β to 1% (4$\times$ less)**. Keeping all the other simulation parameters the same, **make a new plot** of *susceptible*, *infected* and *recovered* nodes at each day. **How does it compare to the previous plot?** What is the new R_0? What is the new number of peak infected nodes? What could be the advantages for the health care system of the city?

(e) **Effect of vaccines**
Now imagine that γ **percentage of the nodes are initially already immune because they were vaccinated**. We will simulate this by putting γ nodes initially on the *recovered* state. Consider that you repeat the previous simulation with $\beta = 4\%$ and $\delta = 1/14$ (with γ randomly vaccinated nodes and 5 random infected nodes at the start). Iterate the simulation until there are no more infected nodes (which will always eventually happen) and **compute I, the number of nodes that got infected somewhere along the simulation.**
Plot this I value on the y-axis as you change the value of *gamma* from 0% to 100% (with jumps of 5%). Give a **brief comment** on what you are seeing and the effectiveness of vaccination. What does it say about the percentage needed for `herd immunity`?

4. Network Construction

In this problem you will be building a network with data from Unicode Locale Data Repository (UCDL). In particular we will be dealing with **countries and languages**, as UCDL provides statistics on the quantity of people from each country that are able to read and write in each territory.

The data was collected from here, and you have access to a raw pre-prepared **csv file**: `raw_unicode.csv`
In this file, each line represents a language understood in a country and comes in the following format:

Country, Country_ID, Language, Language_ID, Population$_{amount}$, Population$_{percentage}$

For example, the first line indicates that Persian is written and read in Afghanistan by 17 million persons, or 50% of its population

(a) **A first network**
Your first task is to build an **undirected weigthed bipartite network between countries and languages**, where the weight is the percentage of population. You should produce two files `unicode_nodes.csv` and `unicode_edges.csv` respectively with the nodes and edges of your network. Gephi can read these files if you preface each file column titles lines (see this example for a little bit of help). To prove you produced the right network (you can use gephi or any other method you prefer):

i. Indicate **basic statistics**: number of nodes, edges, average degree, the average weighted degree, number of connected components, size of the giant component
ii. Only for the **giant component**: diameter, average path length top, top-3 nodes with highest degree and highest pagerank
iii. Show an **image depicting the network**. Choose the layout that feel best shows the network and use node colors to depict type (country or language) and node size to depict degree (you can omit the node labels).

(b) **Filtering the network**
The network you built is still full of *"noise"*. **Remove all the languages that are in total spoken by less than 10 million persons and all the edges that correspond to less than 10% of the population of a country. Afterwards remove all the nodes that are not from the giant component.** To prove produced the right network indicate the number of nodes and edges of the new graph (and submit a `unicode_filtered.gephi` file)

(c) **Projecting on one mode**
Let's now build a new network only of countries, based on a projection. You want to produce a network of *countries* $\times$ *countries* where the weight is related to the common languages between each two countries. You can opt to either use your own script/code or use the Multimode Networks Transformation Gephi tool.

To prove you produced the right network, and in relation to the new (projected) graph:

- Indicate the number of nodes and edges of the new graph
- Identify communities using a modularity based algorithm. Indicate the number of communities you found (they should be more than five). Show the top-5 nodes of each community in terms of global pagerank and use try to explain what each community represents.
- Show an **image depicting the network**. Choose the layout and features that you feel may best represent the division in communities you saw and the importance of the nodes.
- Submit a `unicode_countries.gephi` file with your network and the built layout.