EPIDEMIOLOGICAL MODELLING OF INFLUENZA

Jennifer Probst 16-703-423 UZH
jennifer.probst@uzh.ch
supervision: Akos Dobay
11.06.19

1. Abstract

Influenza (short flu) is estimated to affect about 10-20% of the world's population annually, resulting in around five million cases of severe disease and roughly 250'000 to 500'000 cases of death. [1]

In this study the spread of influenza viruses in different cities all over the world was modelled. Of particular interest were the influence of population density, the movement of people and the force of infection on the duration of an epidemic/pandemic, the prevalence (total number of infected people) and the maximum number of infected people in these cities.

Therefore, ancient pandemics of different severities were simulated in various cities having different population densities and movement parameters. For modelling purposes, the SIR model and cellular automaton were used. Simulations were run on different grid sizes.

Statistical analysis revealed an overall clear positive trend between force of infection and prevalence as well as maximum number of infected people. The duration of the epidemic was found to be shorter with higher force of infection. Further, the model suggested that societies with less public transportation exhibit lower prevalence but prolonged the time of a pandemic. Impact of population density was not clearly visible.

Further research could deepen the analysis of the impact of model parameters and the influence of population density.

2. Introduction

2.1. Influenza viruses [1], [3], [12], [13], [14]

Influenza belongs to the family of Orthomyxoviruses, which are RNA viruses with seven genera namely Influenza virus A, Influenza virus B, Influenza virus C, Influenza virus D, Thogotovirus, Quaranjavirus and Isavirus. Only the first three genera cause influenza in humans, with influenza A

being by far the most common one. They further affect all kinds of vertebrates, often pigs and other mammals or birds.
Influenza is transmitted by airborne droplets from coughing or sneezing over short distances or by contaminated surfaces such as doorknobs, light switches or banknotes as the virus can persist outside of the body.

Seasonal influenza symptoms are sudden fever and cough, also headache, pain in muscles and joints, sore throat and a runny nose. It can only cause severe illness or worst-case death in high risk groups such as pregnant women, children under five years or people over 60 years, health care workers and individuals with chronic medicals or immunosuppressives.

2.2. Influenza epidemics and pandemics [3], [4], [10], [11]

An epidemic is defined as roughly 10-20% of the population being infected with locally constrained outbreaks. In contrast pandemics spread globally and infect a large proportion of the world population. Epidemics occur in temperate climates mainly during winter and in tropical areas irregularly throughout the whole year. Influenza in humans is nearly always caused by a variety of species and strains of the influenza A virus, as these viruses can constantly change their antigenic surface molecules hemagglutinin (HA) and neuraminidase (NA) (this is called antigenic drift). It slowly creates new strains (still fairly like the already existing ones), that are poorly or not at all recognized by the immune system anymore. These strains then spread rapidly throughout the population, causing an epidemic. Pandemics, in contrast, occur when influenza viruses obtain totally new antigens. This happens mostly through reassortment for example between avian and human strains (this is called antigenic shift). As the antigens of this newly aroused virus are entirely novel, everyone will be susceptible to it and the virus will sweep uncontrolled through the population, causing a pandemic. Pandemics are much less frequent than epidemics; nine influenza pandemics were recorded during the past 300 years. Thereof the Spanish influenza pandemic in 1918 was the worst in recorded history, classified as one of the three most destructive human pandemics along with the plague of Justinian and the Black Death. [2]

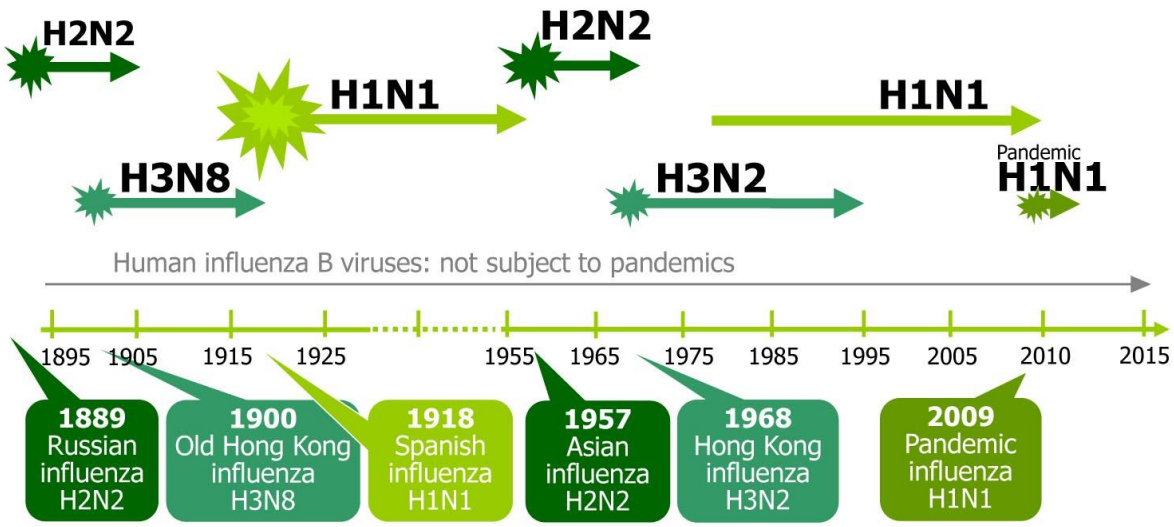

Figure 1: Overview of the most severe historical influenza pandemic outbreaks [5]

With mathematical modelling the progression and spread of infectious diseases can be predicted and modelled. In this study we examined the impact of four historical pandemics shown in Table 1 and a seasonal flu on different cities worldwide.

Pandemic	Year	Influenza virus type	People infected (approx.)	Estimated deaths worldwide	R_0 (Basic reproductive number)
Spanish flu	1918–1919	A/H1N1	33% (500 million)	50-100 million	2.5
Asian flu	1956–1958	A/H2N2	?	2 million	1.65
Hong Kong flu	1968–1969	A/H3N2	?	1 million	1.80
Swine flu	2009–2010	Pandemic H1N1/09	10-200 million	150,000+ (est.)	1.46
Seasonal flu	Every year	mainly A/H3N2 & A/H1N1, rarely B	5–15% (340 million – 1 billion)	250,000 – 500,000 per year	1.28

Table 1: Historical influenza pandemic outbreaks [4], [5]

The basic reproductive number (R_0) stands for the number of people that one infected individual will on average infect during their infection. It is therefore a measure of the transferability of a disease. As a comparison: the estimated basic reproductive numbers for other common infectious diseases range from 12–18 for measles, 12–17 for pertussis, and 4–7 for mumps, polio, rubella, and smallpox.[6]

3. Methods

The spread of epidemiological diseases can be modelled in various ways. In this project we decided to simulate the spread of influenza based on an approach proposed by S. White et al. [28] In their work they introduce a mathematical model to simulate the diffusion of epidemics using cellular automata and the SIR Model. We extend their work to study the influence of population density, transportation dynamics and the force of infection on the duration of an epidemic/pandemic, the prevalence (total number of infected people) and the maximum number of infected people. These questions are examined by a Python based simulation. Eight cities all over the world exhibiting different parameter combinations were chosen and simulations for the in Table 1 displayed flues were run for every city. Using regression in R, we then examined correlations between parameters of interest.

3.1. SIR Modell [7], [17]

The SIR Modell is a compartmental model commonly used in epidemiological modelling. It was developed in 1927 by W. O. Kermack and A. G. McKendrick and widely used with multiple modifications since then. In this model the population is divided into three compartments each representing a specific state of the epidemic. One can either be S, I or R. S stands for susceptible, which are people who have not been infected yet but are susceptible to the disease. People belonging to the compartment 'Infected' are currently infected and capable of passing on the disease to susceptible people. Recovered people have already had the disease and are now immune meaning they can neither infect other people nor get infected again.

$$S \rightarrow^{\beta * c} I \rightarrow^{\gamma} R$$

The population size remains constant, meaning: $N = S^t + I^t + R^t, \forall t$ (where t stands for the timestep). To describe the transition from one compartment to another, the parameters $\beta, \mathrm{c} \text{ and } \gamma$ were used. The force of infection (β) is a measure of transferability of the disease. The contact rate (c) in this model is by default set to one and lowered in cities where transferability of the disease is reduced (e.g. by people wearing mouthpieces). The rate of recovery (γ), calculated as one over the average duration of the disease, stands for the probability to recover from the disease in one timestep (the exact choice of parameter values is described more in detail in 3.3).

3.2. Cellular automata [14], [15]

A cellular automaton is a discrete model that consists of a regular grid of cells of a distinct size. Usually there exist a finite number of states which each cell can adopt. We start with an initial state, which defines a certain state for each cell. This state of each cell is then updated and recorded for each step and cell. For each cell a neighbourhood is defined; here we used the 8 adjacent cells of that cell (Moore neighbourhood), which is commonly used for disease spread modelling. The update of states depends for each cell on its own, its neighbours' states and specific rules (usually mathematical equations). The here used differential equations are almost fully deterministic and can be seen in 3.3.1.

3.3. Implementation of the model [18], [19]

3.3.1. Basic setup

We set up a n x n grid. The simulations were run for grid sizes 15x15, 30x30 and 50x50, each cell representing one square kilometre. The state of a cell in row i and column j is defined by three fractions representing the proportions of susceptible, infected and recovered people in this cell at time t, where one time step represents one day.

$$State_{ij}^{t} = [S^t, I^t, R^t]$$

Following some assumptions are made:

- In the beginning of the simulation there are only healthy people in each cell. The number of people in each cell in the initial state is randomly distributed around the population density of the respective city.

$$State_{ij}^{0} = [n, 0, 0]$$

- One single infected person is introduced into the middle cell of the grid.

$$State_{\frac{n}{2},\frac{n}{2}}^{0} = [n-1, 1, 0]$$

- We assume homogeneous mixing of the population, meaning people making contact at random.
- The total number of people in the grid is constant, implying the epidemics not being lethal. There are no births and no migration out of or into the grid.
- Migration inside the grid is possible. There are different migration rates for healthy and sick people.
- People can only leave the susceptible group by becoming infected and progress further to the recovered group by recovering and receiving immunity.
- No difference in age, sex, race and social status is made.

For every timestep t the fractions S^t, I^t and R^t get updated. The therefore used transition rates between classes are defined here:

$$(1)\ S_{ij}^t = S_{ij}^{t-1} - \beta * c * S_{ij}^{t-1} * I_{ij}^{t-1}$$
$$(2)\ I_{ij}^t = (1-\gamma) * I_{ij}^{t-1} + \beta * c * S_{ij}^{t-1} * I_{ij}^{t-1}$$
$$(3)\ R_{ij}^t = R_{ij}^{t-1} + \gamma * I_{ij}^{t-1}$$

Equation (1) describes the number of susceptible people at time t. It depends on the number of susceptible people at the previous step minus the ones that become infected in this time step. This depends on the contact rate in this city, the force of infection of the virus and the fractions of susceptible and infected people in this cell. The number of susceptible people is therefore monotonically decreasing and steady. Equations (2) and (3) are similarly composed. The number of infected people at time t is composed by the number of infected people at time t-1 plus the number of people getting newly infected minus the number of people recovering. R at time t is monotonically increasing and is computed by adding the people recovering in this time step to the previous ones (this depends on the recovery rate).
The state of a cell in this model therefore only depends on its previous state.

In addition to these basic update rules migration is added at each step. For every city, migration rates for susceptible, infected and recovered were defined. Depending on these rates, after having updated S, I and R for every cell, the corresponding number of migratory people were assigned to one randomly chosen field of the cell's neighbourhood and subtracted in the cell.

3.3.2. Implementation of different cities and pandemics

To test the correlation between the model parameters such as force of infection, migration rates, population density and contact rates, cities were set up with different compositions of these parameters.

The recovery rate was set to γ= 0.1 for all cities, which implies an average sickness duration of 10 days.

In various papers, the basic reproduction number (R_0) of the pandemics defined in Table 1 were estimated. We can calculate the force of infection β as follows:

$$\beta = R_0 * y = R_0 * 0.1,$$

Pandemic	R_0 (basic reproductive number)	β
Spanish Flu	2.5	0.25
Asian Flu	1.65	0.165
Hong Kong Flu	1.80	0.18
Swine Flu	1.46	0.146
Seasonal Flu	1.28	0.128

Table 2: Basic reproductive number and force of infection of modelled pandemics/epidemics [6], [8], [9]

Eight different cities were chosen for the model. The number of people per cell was set up with the actual population density of the corresponding city. The contact rate was either high (=1) if the city does not exhibit any protective action against influenza (then the transition rate from susceptible to infected is exactly = β) or lowered in case of Seoul, as people there wear mouthpieces to prevent from infecting others. For migration rates we chose either high=([0.1, 0.01, 0.1]) or low=([0.01, 0.001, 0.01]), depending on the public transportation in the cities. High transportation rates stand

for 10% of susceptible and recovered people moving per timestep and 1% of the infected ones. In a city with low transportation 1% of susceptible and recovered people and 0.1% of infected move per day.

City	Population density [N/km^2]	Contact rate	Migration rates
Levallois-Perret (suburb of Paris, France)	26'000	1	High
Dhaka (Bangladesh)	29'105	1	Low
Zurich (Switzerland)	4'454	1	High
Raipur (India)	4'500	1	Low
Bergen (Norway)	604	1	High
Menzingen (Switzerland)	164	1	Low
Seoul (South Korea)	16'456	0.95	High
Monaco	18'713	1	High

Table 3: Modelling parameters for different cities [20], [21], [22], [23], [24], [25], [26], [27]

To see the effect of population size and transportation we chose pairs of cities with very high (Levallois-Perret and Dhaka), high (Seoul and Monaco), medium (Zurich and Raipur) and low population density (Bergen and Menzingen). In the groups of very high, medium and low population density there was always one city with low migration rates and one with high migration rates. Seoul had the same parameter setting as Monaco but a lower contact rate.

The simulations were run for each combination of city and pandemic on grid sizes 15x15, 30x30 and 50x50. We were interested in the number of time steps it takes until equilibrium was reached (time of pandemic), in the number of people that got infected in each simulation (prevalence) and the maximum number of people being infected at once. Equilibrium was defined to occur, when the total changes in numbers of infected people and recovered both were smaller than a certain threshold.

3.4. Statistical analysis

The statistical analysis was performed using R. For each plot of force of infection vs. prevalence, max number of infected and time of pandemic a regression line was separately fit. The non-linear least squares method was used and correlation between the function fit and actual data was calculated.

4. Results

4.1. Differences in size of the grid

As already mentioned before, we ran the simulations on grid sizes 15x15, 30x30 and 50x50. Overall differences between grid sizes were only visible in terms of numbers (higher numbers of infected and maximal infected and lower durance of pandemic for bigger grid size). The results displayed in all following figures were obtained from simulations on a 30x30 grid. The patterns described in 4.2 and 4.3 were observed for all grid sizes.

4.2. Prevalence, maximal number of infected people and time of pandemic

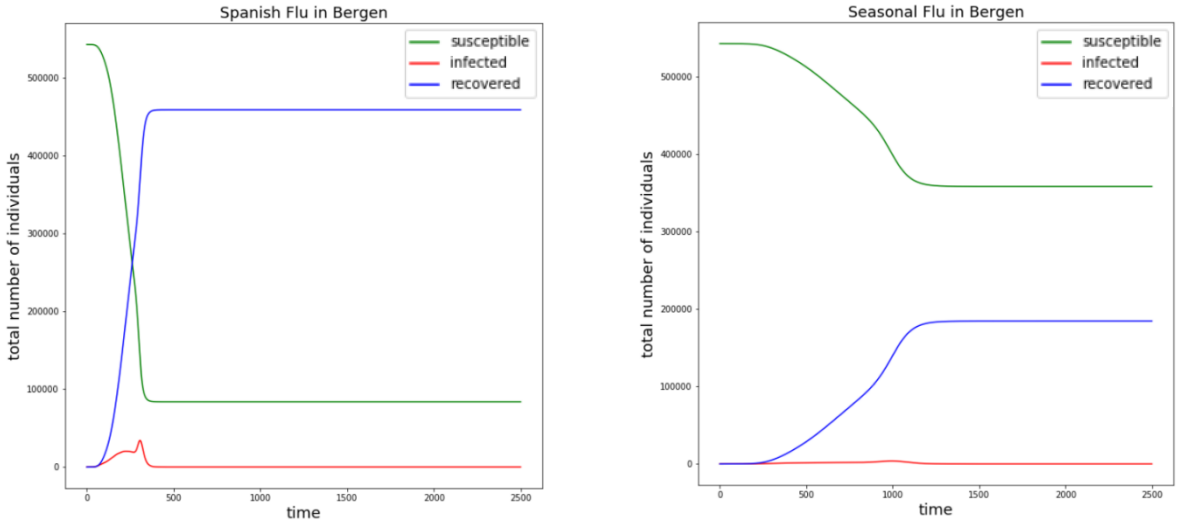

Figure 2: SIR plots of flus with β = 0.25 (Spanish Flu) and β = 0.128 (Seasonal Flu) in Bergen

For each combination of city and pandemic graphs as in Figure 2 were obtained. The graphs show the development of susceptible, infected and recovered people in the cities over time. The number of susceptible always decreased monotonically over time, whereas recovered increased monotonically and infected reached a peak value during infection and dropped off as the pandemic diminished.

4.2.1. Number of infected people

Table 4 shows the total number of infected people for each simulation normalized by the population density such that individual values were comparable.

Overall a clear positive trend between force of infection and prevalence was observed (Figure 3). For the function *f(Force) = a*log(b*Force)*, which predicts the total number of infected people, parameters a and b were estimated using non-linear least squares. Estimates were a = 2058.8010, b=10.2626; both being highly significant (p-value<2e-16). The correlation of this fit to the actual data is 0.9752.

City	Spanish Flu	Asian Flu	Hong Kong Flu	Swine Flu	Seasonal Flu
Levallois-Perret	1899	1390	1209	920	553
Dhaka	1853	1310	1113	793	399
Zurich	1906	1401	1218	929	567
Raipur	1795	1210	1000	661	253
Bergen	1910	1409	1228	940	579
Menzingen	1866	1338	1145	829	440
Seoul	1838	1290	1097	784	402
Monaco	1899	1393	1212	919	557

Table 4: Total number of infected people (normalized) for all simulation scenarios

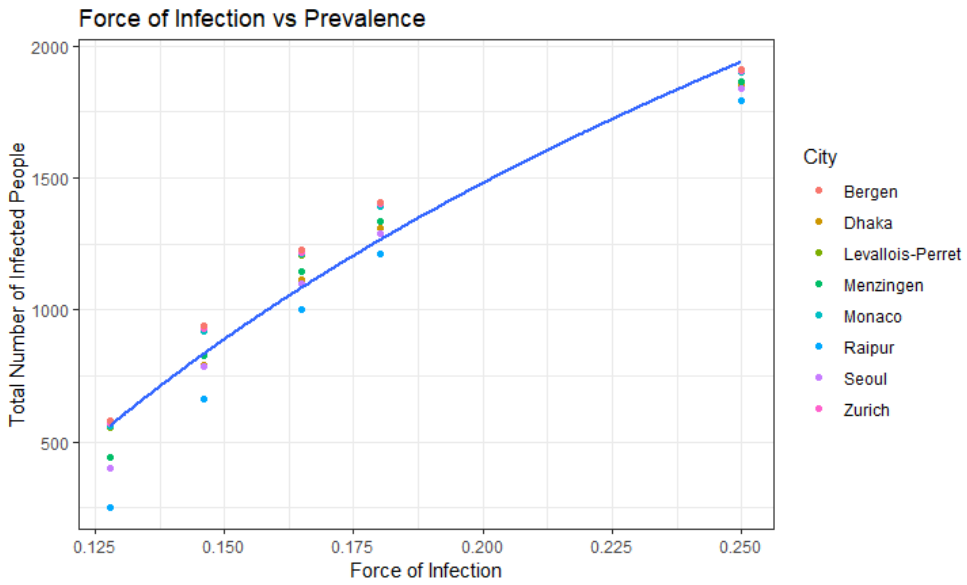

Figure 3: Regression of force of infection vs. prevalence

4.2.2. Maximal number of infected people

Table 5 displays the maximal number of infected people also normalized by the population density.

As shown in Figure 4, a positive correlation between force of infection and maximal number of infected people was found. Equivalently to 4.2.1. we used non-linear least squares to fit the function *f(Force) = a*log(b*Force)*, estimating the maximal number of infected people per simulation. The fit resulted in a correlation of 0.9752 to the actual data with a = 7.5875, b=12.7712 (p-value<2e-16).

City	Spanish Flu	Asian Flu	Hong Kong Flu	Swine Flu	Seasonal Flu
Levallois-Perret	8.47	6.65	5.97	4.88	3.42
Dhaka	8.73	6.94	6.25	5.09	3.58
Zurich	8.46	6.64	5.96	4.85	3.42
Raipur	8.52	6.55	5.79	4.53	2.91
Bergen	8.45	6.63	5.95	4.83	3.39
Menzingen	8.69	6.92	6.23	5.07	3.57
Seoul	8.25	6.27	5.54	4.33	2.78
Monaco	8.47	6.65	5.97	4.87	3.42

Table 5: Maximal number of infected people (normalized) for all simulation scenarios

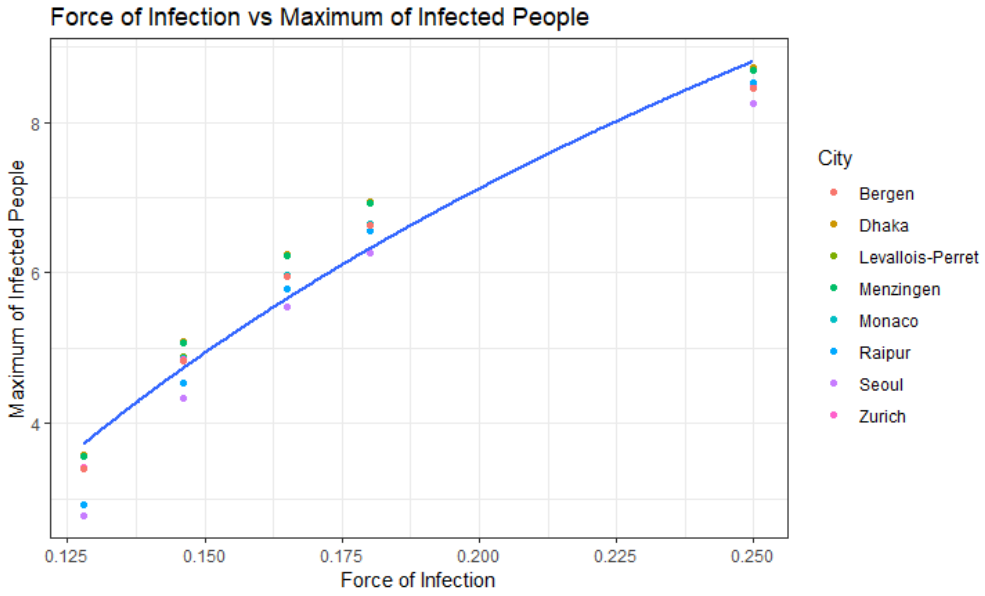

Figure 4: Regression of force of infection vs. max number of infected people

4.2.3. Time of pandemic

In Table 6 the time of pandemic for each of the simulation scenarios can be seen.

Here we found the duration of the epidemic to be shorter with higher force of infection. Non-linear least squares estimated a=7464.911 and b=12.636 (p-value<2e-16) for the function *f(Force) = a*exp(-b*Force)*, predicting the time of pandemic. The correlation of this fit was -0.9834.

City	Spanish Flu	Asian Flu	Hong Kong Flu	Swine Flu	Seasonal Flu
Levallois-Perret	470	711	827	1051	1510
Dhaka	646	1005	1180	1513	2177
Zurich	473	722	839	1063	1528
Raipur	692	1106	1303	1731	2429
Bergen	476	724	839	1075	1560
Menzingen	643	1338	1174	1509	2182
Seoul	497	781	909	1198	1839
Monaco	468	715	823	1059	1508

Table 6: Time of pandemic for all simulation scenarios

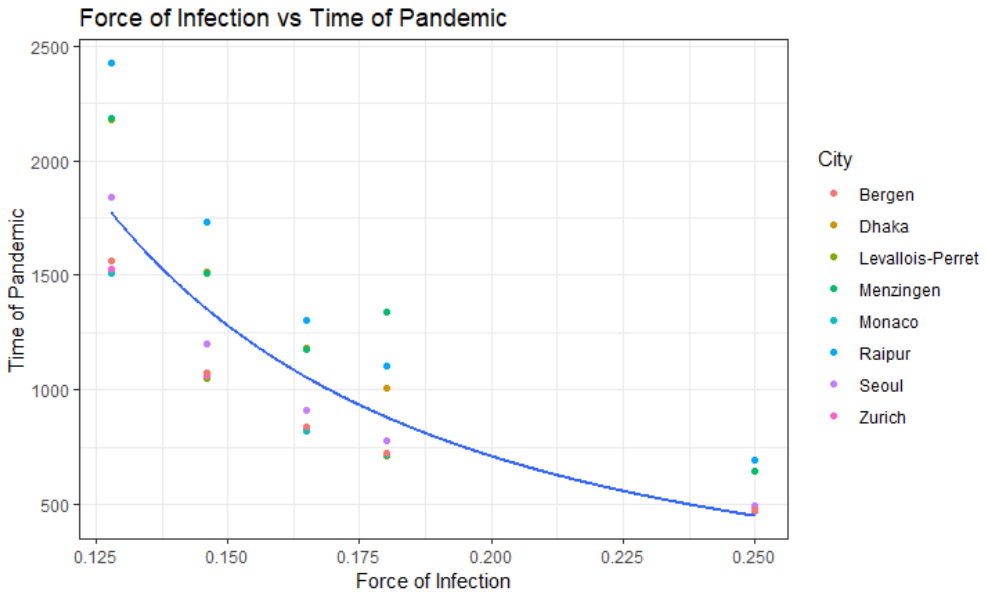

Figure 5: Regression of force of infection vs. time of pandemic

4.3. City effects on individual influenza pandemics on 30x30 grid

In Figure 6 the individual graphs (corresponding to one type of pandemic) show time of pandemic vs. prevalence for the different cities. Pandemics with higher force of infection display an overall lower time of pandemic and higher numbers of infected people as observed in (4.2.). Other than that, the distribution of the cities in the plots seems to follow the same pattern for every flu. Therefore, observations made for one pandemic are representative for all others.

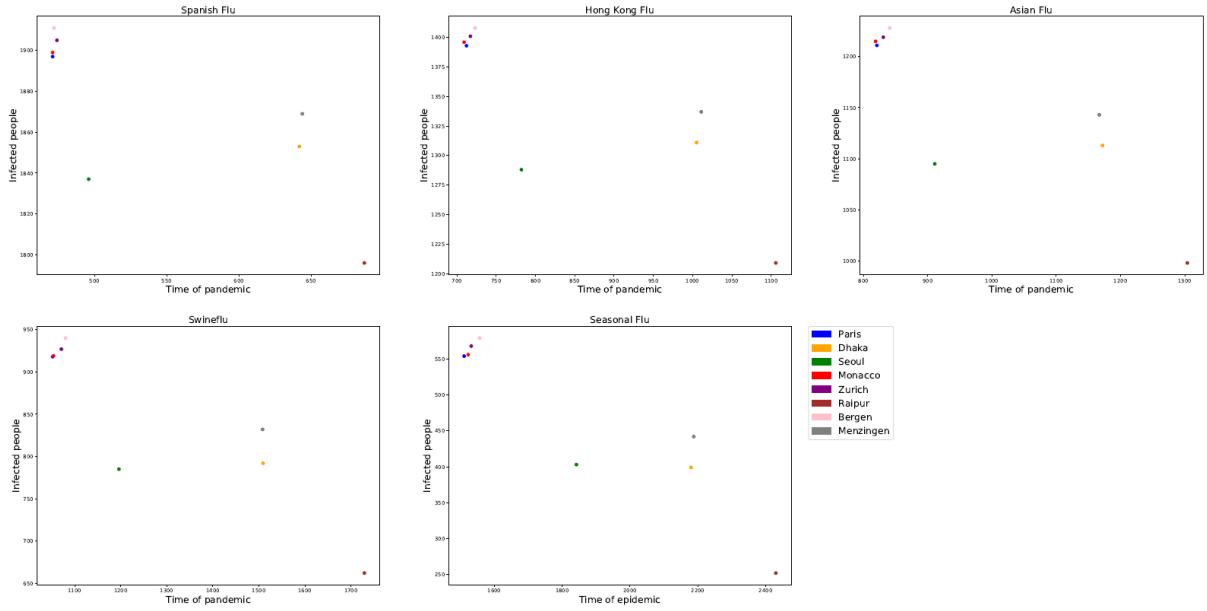

Figure 6: Plots of time of pandemic vs. infected people in different cities for each pandemic

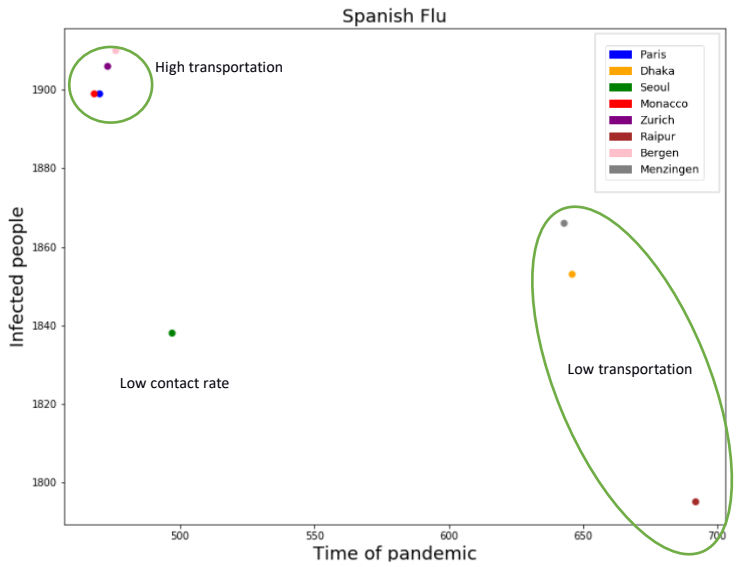

Figure 7: Impact of transportation and contact rate on duration of pandemic and prevalence

The lowered contact rate of Seoul reduced the number of infected people in comparison to other cities with the same high rates of transport.

Furthermore, the model suggested that societies with less public transportation (Menzingen, Dhaka and Raipur) have lower prevalence but extended time of pandemic in comparison to others with higher rates of transport.

This can also be seen in Figure 9 and 10 where individual regression lines for either high or low transportation were fit.

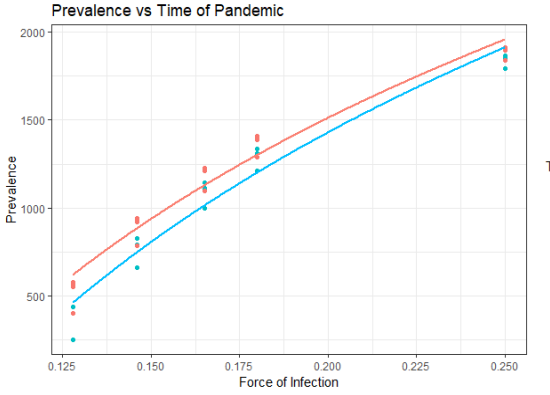

Figure 9: Influence of transportation on prevalence

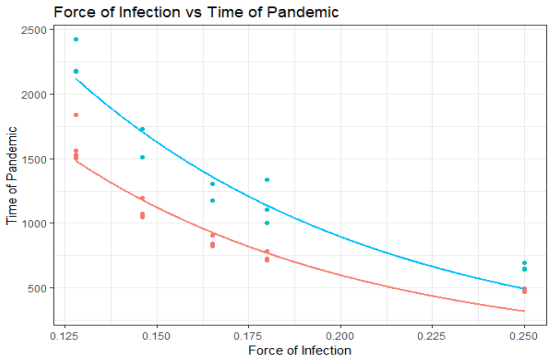

Figure 10: Influence of transportation on time of pandemic

The impact of population density wasn't clearly visible in these simulations. Paris and Bergen for example having the same transportation and contact rates behaved quite similar in this model.

4.4. Animations

To visualize the spread of the pandemics on the grid we also encoded an animation function. Red colour represents high numbers of infected whereas blue stands for low numbers of infected people.

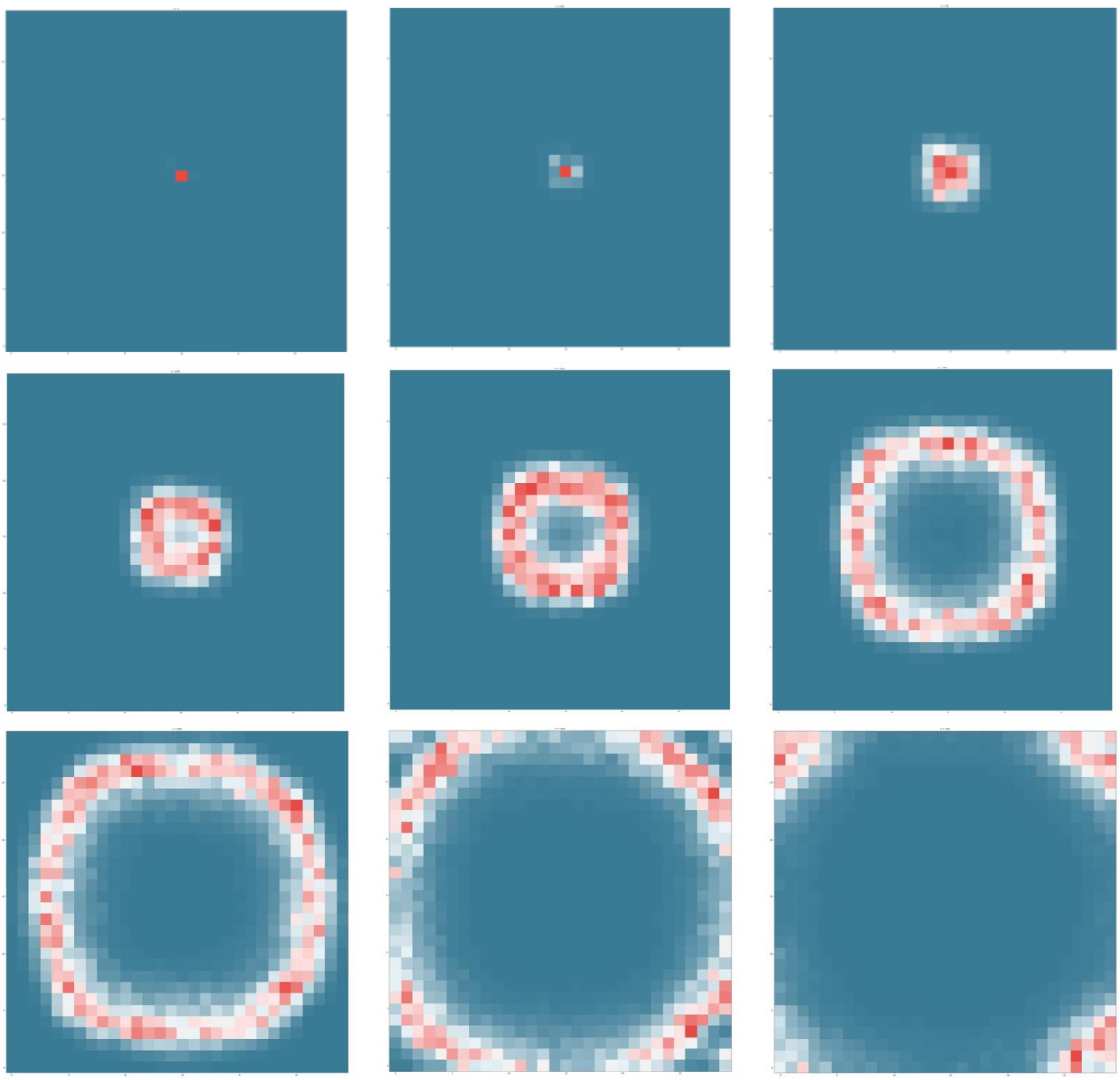

Figure 6: Spread of Spanish Influenza in Levallois-Perret at t = 1, 64, 96, 120, 160, 204, 254, 301, 344 (days)

5. Discussion

General trends observed for high force of infection were high prevalence and high maximal number of infected people. It makes sense that the more people become sick, the stronger the pandemic is. Additionally, strong pandemics in this model spread quicker. This also has a logical explanation. If there are more sick people and a fixed percentage of infected people migrate, this results in more sick people migrating. The pandemic spreads quicker, more people get infected and recover again in shorter time, which explains a lower duration of pandemic with higher force of infection.

Furthermore, we found low transportation rates to result in lower prevalence and high duration of pandemics. This can be explained as follows: Lower transportation rates have the effect that the pandemic does not spread as quickly resulting in high duration of pandemics. The infection also stays more localized in some cells where already recovered persons create a kind of buffer for others to get infected. This effect plus less spread of the disease lead to lower prevalence as outcome.

A lower contact rate reduces the rate of infection and therefore lower prevalence and maximal number of infected people were observed.

These general trends observed seem to be quite reasonable. They closely represent the findings we had expected. But one must keep in mind that this is only a simulation. There were multiple assumptions made in the beginning which simplify the modelling, but don't always apply to reality. Size of the grid was the same for all cities, which is not realistic. This could be improved by modelling the spread of influenza on different grid sizes for individual cities. There were also no births or deaths in this model in order to keep the total population size constant. This could be altered too. Migration in this model was only allowed inside the cities, but for a more realistic model migration into and out of the grid would have to be included as well. Also, the setting of parameters in this simulation did also not always depend on empirical data. As for migration rates, parameters that were thought to be appropriate were chosen.

Looking at the results, we have the theory that the setting of the transportation parameters maybe dominated over some population density effect. This could be the reason why no impact of population density could be observed. It would be interesting to alter them to see, if different patterns were visible with changed migration rates.
The choice of parameter values could also be an explanation why the times of pandemics were quite unrealistic. As every step in this model represents a day, the average duration of seasonal flu for example was 1842 days, which are approximately 5 years.

So, to conclude; the general trends found in this model intuitively make sense. But there remains the question how transferable the model findings are, as a model is always a balance between realistic modelling and simplicity.

6. Self-assessment

Overall, I'm quite satisfied with my project. My main goal, which was primarily to simulate the basic spread of an epidemic, was achieved quite quickly. In the following weeks I kept working on my code making it more efficient and generally usable. Thereby, I learned many new programming skills. Working with classes I could individually combine all different kinds of scenarios and got many plotted outputs. Having never used animations before, I was also very happy to get the animation of the disease spread working. Another thing I learned, was how to structure output data efficiently to easily get good plots and do some statistics on them.

I can't think of anything in particular, that completely failed during my project. I just wished I had more time to investigate if there really is no impact of population density on the outcome of the simulation. Also, a more sophisticated regression in R with more elaborate functions would have been desirable.
Nevertheless, I am overall pleased with the result of my project and I am happy that I had the opportunity to learn many new things in such a practical way.

Therefore, I want to thank Akos Dobay for his advice and constant support throughout my project.

7. Bibliography

[1] *WHO.* From https://www.who.int/en/news-room/fact-sheets/detail/influenza-(seasonal)

[2] Potter, C. (2008). https://onlinelibrary.wiley.com/doi/full/10.1046/j.1365-2672.2001.01492.x.

[3] *Wikipedia*. From Influenza: https://en.wikipedia.org/wiki/Influenza

[4] *Wikipedia*. From Influenza Pandemic: https://en.wikipedia.org/wiki/Influenza_pandemic

[5] From https://emssolutionsinc.files.wordpress.com/2010/03/pandemic-graph.jpg

[6] Matthew Biggerstaff, S. C. (2014). Estimates of the reproduction number for seasonal, pandemic, and zoonotic influenza: a systematic review of the literature. *BMC Infectious Diseases*.

[7] *Wikipedia*. From Compartmental models in epidemiology: https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology#The_MSIR_model

[8] Marc Lipsitch, C. M. (n.d.). *Global Health Security Initiative.* From ESTIMATES OF THE BASIC REPRODUCTIVE NUMBER FOR 1918 PANDEMIC: http://www.ghsi.ca/documents/Lipsitch_et_al_Submitted%2020050916.pdf

[9] Emilia Vynnycky, A. T. (2007). Estimates of the reproduction numbers of Spanish influenza using morbidity data. *International Journal of Epidemiology*.

[10] *Centers for Disease Control and Prevention*. From Past Pandemics: https://www.cdc.gov/flu/pandemic-resources/basics/past-pandemics.html

[11] *History of Vaccines*. From Influenza Pandemics: https://www.historyofvaccines.org/content/articles/influenza-pandemics

[12] *WHO.* From Influenza: https://www.who.int/influenza/en/

[13] *Doccheck.* From Influenza: https://flexikon.doccheck.com/de/Influenza

[14] *BAG (Bundesamt für Gesundheit).* From Saisonale Grippe (Influenza) : https://www.bag.admin.ch/bag/de/home/krankheiten/krankheiten-im-ueberblick/grippe.html

[15] *Wikipedia*. From Cellular automaton: https://en.wikipedia.org/wiki/Cellular_automaton

[16] *Mathworld Wolfram*. From Cellular Automaton: http://mathworld.wolfram.com/CellularAutomaton.html

[17] *Mathematical Association of America*. From The SIR model for spread of disease - The differential equation model: https://www.maa.org/press/periodicals/loci/joma/the-sir-model-for-spread-of-disease-the-differential-equation-model

[18] Rafael Mikolajczyk, R. K. (2009). Influenza – Einsichten aus mathematischer Modellierung. *Deutsches Aerzteblatt*. From https://www.aerzteblatt.de/pdf.asp?id=66813

[19] Florian Miksch, G. Z. (2013). *DWH Simulation Services.* From INFLUENZA – Modellierung und Simulation von Influenza-Epidemien: http://www.hauptverband.at/cdscontent/load?contentid=10008.566581

[20] *Wikipedia.* From Levallois-Perret : https://de.wikipedia.org/wiki/Levallois-Perret

[21] *Wikipedia.* From Dhaka: https://en.wikipedia.org/wiki/Dhaka

[22] *Wikipedia.* From Seoul: https://en.wikipedia.org/wiki/Seoul

[23] *Wikipedia.* From Monacco: https://en.wikipedia.org/wiki/Monaco

[24] *Wikipedia.* From Zurich: https://de.wikipedia.org/wiki/Z%C3%BCrich

[25] *Wikipedia.* From Raipur: https://en.wikipedia.org/wiki/Raipur

[26] *Wikipedia.* From Bergen: https://de.wikipedia.org/wiki/Bergen_(Norwegen)

[27] *Wikipedia.* From Menzingen Zug: https://de.wikipedia.org/wiki/Menzingen_Z

[28] S. White et. Al. (2007). *Applied Mathematics and Computation* Modelling epidemics using cellular automata: https://www.sciencedirect.com/science/article/pii/S0096300306009295

8. Appendix

8.1. Code for the simulation in Python

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Name:    Probst, Jennifer
Email:    jennifer.probst@uzh.ch
Date:    07/06/2019
Semester:  FS19
Topic:    Epidemiological modelling of influenza
"""

import random
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import seaborn
import math
import matplotlib.patches as mpatches

class Society():
  ''' set up societies with their parameters'''

  def __init__(self, name, population, c, y, mS, mI, mR):
    #set variables
    self.name=name
    self.population=population
    self.c=c #contact rate
    self.y=y #recovery rate
    self.mS=mS #migration rate suceptible
    self.mR=mR #migration rate recovered
    self.mI=mI #migration rate infected


class Pandemic():
  '''set up pandemics with their parameters'''

  def __init__(self, name, b):
    #set variables
    self.name=name
    self.b=b #force of infection


class CellularAutomata():

  def __init__(self, society, pandemic, times, n):
    #define variables
    self.society=society
    self.pandemic=pandemic
    self.n=n
    self.times=times
    self.timeepidemic=0
    self.maxI=1
    self.fig = plt.figure(figsize=(self.n, self.n),
frameon=False)
    self.colormap = seaborn.diverging_palette(220, 10,
as_cmap = True)

    #set up arrays and total lists
    self.fractions= np.zeros((self.n,self.n, 3, self.times))
    self.people=np.zeros((self.n,self.n,3, self.times))
    self.plotting=np.zeros((self.n,self.n, 3, self.times))

    #distribute people to the grid
    self.fractions[:,:,:,0]=[1,0,0]
    self.setup_people()

    #choose where infection starts
    self.setup_infection_start()

  def setup_people(self):
    '''sets people in every grid with normal distribution
around n'''
    a=[int(np.random.normal(self.society.population,
0.01*self.n, 1)) for i in range(self.n)]
    self.people[:,:,0,0]=a

  def setup_infection_start(self):
    '''starts the infections with 1 sick individual'''
    self.start_infection = math.floor(self.n / 2)
    a=self.people[self.start_infection,
self.start_infection, :, 0]
    self.fractions[self.start_infection,
self.start_infection, :, 0] = [(a[0]-1)/a[0], 1/a[0], 0]
    self.people[self.start_infection, self.start_infection,
:, 0] = [a[0]-1, 1, 0]

  def loop(self, t):
    '''main loop for animation'''
    self.t=t
    self.simmulate1(self.fractions[:,:,:,t],
self.people[:,:,:,t], t)
    self.update()

  def simmulate_whole(self):
    ''' simmulates n times and returns: total number of
infected, time of pandemic/epidemic, maximum number
of sick people'''

    #simmulate n times
    for t in range(self.times-1):
      self.simmulate1(self.fractions[:,:,:,t],
self.people[:,:,:,t], t)
      Istep=(sum(sum(sum(self.people[:,:,1,:]))))
      if Istep>self.maxI:
        self.maxI=Istep
      if self.Istep>0.001 or self.recover>0.01:
        self.timeepidemic+=1
    self.tI=sum(sum(sum(self.people[:,:,2,:])))

    #add totals
    self.total_S=[(sum(sum(self.people[:,:,0,t]))) for t in
range(self.times)]
    self.total_I=[(sum(sum(self.people[:,:,1,t]))) for t in
range(self.times)]
```

```
        self.total_R=[(sum(sum(self.people[:,:,2,t]))) for t in
range(self.times)]

        self.plot()

        return(self.tI, self.timeepidemic, self.maxI)

    def simmulate1(self, fractions, people, t):
        ''' simulates one round of updates & gives back the
updated fractions and total numbers
            logic:
            1. for all i update fractions in all grids with diff
equ and total numbers
            2. calculate migration people and
            3. store migration in migration matrix
            4. add total numbers and migration matrix
            5. update fractions'''

        #update fractions with diff equations
        self.recover=0
        for i in range(len(fractions)):
            for j in range(len(fractions)):
                fractions[i][j]=self.update_grid(fractions[i][j])

        #update total numbers
        for i in range(len(fractions)):
            for j in range(len(fractions)):
                total_peops=sum(people[i][j])
                frac=fractions[i][j]

people[i][j]=[frac[0]*total_peops,frac[1]*total_peops,fra
c[2]*total_peops]

        #calculate migration people, store migration in
matrix (the ones leaving and coming)
        mm=np.zeros((self.n,self.n,3))
        for i in range(len(people)):
            for j in range(len(people)):
                mm+=self.simmulate_migration(i, j,
people[i][j])

        #add total number and migration matrix
        people+=mm

        #update frequences
        for i in range(len(fractions)):
            for j in range(len(fractions)):
                total_peops=sum(people[i][j])
                peops=people[i][j]
                fractions[i][j]=[peops[0]/total_peops,
peops[1]/total_peops, peops[2]/total_peops]

        #update values
        self.plotting=self.fractions[:, :, 1, t]
        self.fractions[:, :, :, t + 1] = fractions
        self.people[:, :, :, t + 1] = people
        self.Istep=(sum(sum(fractions[:,:,1])))

    def update_grid(self, grid):
        '''gets fractions in one grid as imput and returns
updated fractions'''
        return ([self.update_S(grid), self.update_I(grid),
self.update_R(grid)])

    def update_S(self, grid):
        '''given fractions in one grid, returns updated S'''
        a=grid[0]-
self.society.c*grid[0]*grid[1]*self.pandemic.b
        return a

    def update_I(self, grid):
        '''given fractions in one grid, returns updated I'''
        b=grid[1]-self.society.y*grid[1] +
self.society.c*grid[0]*grid[1]*self.pandemic.b
        return b

    def update_R(self, grid):
        '''given fractions in one grid, returns updated R'''
        self.recover=self.society.y*grid[1]
        c=grid[2]+self.recover
        return c

    def select_neighbor(self, i, j):
        ''' outputs neighbor coordinates as a tuple that
randomly gets migrants
            change in coordinates:
            a= change in i
            b= change in j
            no migrants can get out of the grid
            '''
        while True:
            a=random.randint(-1, 1)
            b=random.randint(-1, 1)
            if a==0 and b==0:
                continue
            if i==(self.n-1) and a==1 and j==(self.n-1) and
b==1:
                return (-1,-1)
            if i==(self.n-1) and a==1:
                return (-1,b)
            if j==(self.n-1) and b==1:
                return (a,-1)
            else:
                return (i+a,j+b)

    def simmulate_migration(self, i, j, grid):
        '''given absolute numbers of people in one grid it
calculates the people to migrate
            and where to and stores them at their destination
grid in an otherwise empty
            matrix which is returned.'''
        #specify temporary migration matrix
        mat=np.zeros((self.n,self.n,3))
        #calculate people to migrate
        migrants_S=grid[0]*self.society.mS
        migrants_I=grid[1]*self.society.mI
        migrants_R=grid[2]*self.society.mR
        #where to migrate
        a,b=self.select_neighbor(i,j)
        mat[a][b]=[migrants_S, migrants_I, migrants_R]
        mat[i][j]+=[-migrants_S, -migrants_I, -migrants_R]
        return mat

    def plot(self):
        ''' generate plots '''
        #timeplot of totals
```

```
        timeline=list(range(self.times))
        fig1, ax1 = plt.subplots(1, 1, figsize=(10,10))
        ax1.plot(timeline, self.total_S, color='green',
label='susceptible')
        ax1.plot(timeline, self.total_I, color='red',
label='infected')
        ax1.plot(timeline, self.total_R, color='blue',
label='recovered')
        plt.legend()
        ax1.set_xlabel('time', fontsize=18)
        ax1.set_ylabel('total number of individuals',
fontsize=18)
        ax1.set_title('{} in {}'.format(self.pandemic.name,
self.society.name), fontsize=18)

        fig1.savefig('{} in {} on {}x{}
grid.png'.format(self.pandemic.name, self.society.name,
self.n, self.n), format='png')
        plt.show()

    def update(self):
        '''plot for the animation'''
        self.fig.clear()
        ax = plt.subplot()
        plt.imshow(self.plotting, cmap = self.colormap)
        ax.grid(False)
        ax.set_xlim([-1/2, self.n-1/2])
        ax.set_ylim([-1/2, self.n-1/2])
        ax.set_title('t = {}'.format(self.t+1))
        #plt.show()

    def animate(self):
        '''animation'''
#       Writer = animation.writers['ffmpeg']
#       self.writer = Writer(fps=15,
metadata=dict(artist='Me'), bitrate=1800)
        cartoon = animation.FuncAnimation(self.fig,
self.loop, frames = (self.times-1), interval = 25, repeat =
False)
#       cartoon.save('basic_animation.mp4', fps=10,
writer=self.writer)


class Szenarios():

    def __init__(self, times, n):
        '''define different flus, forces and societies'''
        self.setup()
        self.flus=[self.spanishflu, self.hongkongflu,
self.asianflu, self.swineflu, self.seasonalflu]
        self.forces=[0.25, 0.18, 0.165, 0.146, 0.128]
        self.societies=[self.paris, self.dhaka, self.seoul,
self.monacco, self.zurich, self.raipur, self.bergen,
self.menzingen]
        self.times=times
        self.n=n

    def simmulate(self):
        self.lists()
        self.plotforce()
        self.plotflu()

    def setup(self):
        '''setup pandemics and societies'''
        #create pandemics
        self.spanishflu=Pandemic('Spanish Flu', 0.25)
        self.hongkongflu=Pandemic('Hong Kong Flu', 0.18)
        self.asianflu=Pandemic('Asian Flu', 0.165)
        self.swineflu=Pandemic('Swineflu', 0.146)
        self.seasonalflu=Pandemic('Seasonal Flu', 0.128)

        #create societies: population, c, y, mS, mI, mR
        self.paris=Society('Paris', 26000, 1, 0.1, 0.1, 0.01,
0.1)
        self.dhaka=Society('Dhaka', 29105, 1, 0.1, 0.01,
0.001, 0.01)
        self.seoul=Society('Seoul', 16456, 0.95, 0.1, 0.1,
0.01, 0.1)
        self.monacco=Society('Monacco', 18713, 1, 0.1, 0.1,
0.01, 0.1)
        self.zurich=Society('Zurich', 4454, 1, 0.1, 0.1, 0.01,
0.1)
        self.raipur=Society('Raipur', 4500, 0.95, 0.1, 0.01,
0.001, 0.01)
        self.bergen=Society('Bergen', 604, 1, 0.1, 0.1, 0.01,
0.1)
        self.menzingen=Society('Menzingen', 164, 1, 0.1,
0.01, 0.001, 0.01)

    def lists(self):
        #set up experiments and store results in list:
        self.loc_list=[]
        for society in self.societies:
            temp=[]
            for flu in self.flus:
                Simmulation=CellularAutomata(society, flu,
self.times, self.n)

                #Simmulation.animate()

                sick, time, maxsick =
Simmulation.simmulate_whole()
                sick=int(sick/(society.population*self.n**2))

maxsick=maxsick/(society.population*self.n**2)
                temp.append([society.name, flu.name, flu.b,
sick, time, maxsick])
            self.loc_list.append(temp)
        print(self.loc_list)

    def plotforce(self):
        #plot for force of infection vs people ill or time of
pandemic or max of infected
        fig1, ax1 = plt.subplots(1, 1, figsize=(10,10))
        fig2, ax2 = plt.subplots(1, 1, figsize=(10,10))
        fig3, ax3 = plt.subplots(1, 1, figsize=(10,10))
        for i in range(len(self.societies)):
            time=[]
            peop=[]
            maxsick=[]
            for j in range(len(self.flus)):
                time.append(self.loc_list[i][j][4])
                peop.append(self.loc_list[i][j][3])
                maxsick.append(self.loc_list[i][j][5])
            ax1.scatter(self.forces, peop,
label='{}'.format(self.societies[i].name))
```

```
        ax1.set_xlabel('Force of infection', fontsize=18)
        ax1.set_ylabel('Infected people', fontsize=18)
        ax1.set_title('Infected people vs. force of
infection', fontsize=18)
        ax2.scatter(self.forces, time,
label='{}'.format(self.societies[i].name))
        ax2.set_xlabel('Force of infection', fontsize=18)
        ax2.set_ylabel('Time of pandemic', fontsize=18)
        ax2.set_title('Time of pandemic vs. force of
infection', fontsize=18)
        ax3.scatter(self.forces, maxsick,
label='{}'.format(self.societies[i].name))
        ax3.set_xlabel('Force of infection', fontsize=18)
        ax3.set_ylabel('Max number of infected people',
fontsize=18)
        ax3.set_title('Max number of infected vs. force of
infection', fontsize=18)

      fig1.legend(loc=0)
      fig1.show()
      fig2.legend(loc=0)
      fig2.show()
      fig3.legend(loc=0)
      fig3.show()

      fig1.savefig('Infected people on {}x{}
grid.png'.format(self.n, self.n), format='png')
      fig2.savefig('Time of pandemic for {}x{}
grid.png'.format(self.n, self.n), format='png')
      fig3.savefig('Max number of infected for {}x{}
grid.png'.format(self.n, self.n), format='png')

   def plotflu(self):
      #plot of time of pandemic vs people for each flu
      fig = plt.figure(figsize=(40,20))
      ax1 = fig.add_subplot(231)
      ax2 = fig.add_subplot(232)
      ax3 = fig.add_subplot(233)
      ax4 = fig.add_subplot(234)
      ax5 = fig.add_subplot(235)
      fig.subplots_adjust(hspace=0.2, wspace=0.2)
      axes=[ax1, ax2, ax3, ax4, ax5]

      times=np.zeros((len(self.flus),len(self.societies)))
      people=np.zeros((len(self.flus),len(self.societies)))
      colors=['blue', 'orange', 'green', 'red', 'purple',
'brown', 'pink', 'grey']

      for i in range(len(self.societies)):
         for j in range(len(self.flus)):
            times[j][i]= self.loc_list[i][j][4]
            people[j][i]=self.loc_list[i][j][3]
      for j in range(len(self.flus)):
         axes[j].set_title('{}'.format(self.flus[j].name),
fontsize=18)
         axes[j].set_ylabel('Infected people', fontsize=18)
         axes[j].set_xlabel('Time of pandemic',
fontsize=18)
         axes[j].scatter(times[j], people[j], color=colors, )
         b = mpatches.Patch(color='blue', label='Paris')
         o = mpatches.Patch(color='orange', label='Dhaka')
         g = mpatches.Patch(color='green', label='Seoul')
         r = mpatches.Patch(color='red', label='Monacco')
         p = mpatches.Patch(color='purple', label='Zurich')
         br = mpatches.Patch(color='brown',
label='Raipur')
         pi = mpatches.Patch(color='pink', label='Bergen')
         gr = mpatches.Patch(color='grey',
label='Menzingen')
         plt.legend(handles=[b,o,g,r,p,br,pi,gr],
bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.,
fontsize=18)
      axes[4].set_xlabel('Time of epidemic', fontsize=18)
      plt.show()

      fig.savefig('Flus on {}x{} grid.png'.format(self.n,
self.n), format='png')



if __name__== "__main__":
   ''' different gridsizes simulated '''

   grid5=Szenarios(800, 5)
   grid5.simmulate()
   grid15=Szenarios(1700, 15)
   grid15.simmulate()
   grid30=Szenarios(2500, 30)
   grid30.simmulate()
   grid50=Szenarios(3500, 50)
   grid50.simmulate()
```

8.2. Code for regression in R

```
library(readr)
library(tidyverse)
library(poweRlaw)

#read in data:
Totaldata <-
read_csv("C:/Users/probs/Desktop/Compbio/Semesterp
rojekt/Totaldatacsv.csv")
Totaldata$City <- as.factor(Totaldata$City)
Totaldata$Transportation <-
as.factor(Totaldata$Transportation)

#model for infected:
fit11 <- nls(Infected ~ a*log(Force*b), data=Totaldata,
start=list(a=700, b=12))
summary(fit11)
cor(Totaldata$Infected, predict(fit11))

new_data <-
expand.grid(Force=seq(min(Totaldata$Force),
max(Totaldata$Force), length=1000))
p1 <- predict(fit11, newdata=new_data,
interval="confidence")
n1 <- cbind(new_data, p1)
n1

ggplot(data=Totaldata) +
  geom_point(aes(x=Force, y=Infected, colour=City)) +
  xlab("Force of Infection") +
  ylab("Number of Infected People") +
  ggtitle("Force of Infection vs Infected People") +
```

```
  theme_bw()+
  geom_smooth(data=n1, mapping=aes(x=Force, y=p1),
stat="identity")

#fit for infected with high and low migration:
migrhigh <- filter(Totaldata, Transportation=='high')
migrlow <- filter(Totaldata, Transportation=='low')

fit1ml <- nls(Infected ~ a*log(Force*b), data=migrlow,
start=list(a=700, b=12))
summary(fit1ml)
xnew <- expand.grid(Force=seq(min(Totaldata$Force),
max(Totaldata$Force), length=1000))
p1ml <- predict(fit1ml, newdata=xnew,
interval="confidence")
n1ml <- cbind(new_data, p1ml)

fit1mh <- nls(Infected ~ a*log(Force*b), data=migrhigh,
start=list(a=700, b=12))
summary(fit1mh)
xnew <- expand.grid(Force=seq(min(Totaldata$Force),
max(Totaldata$Force), length=1000))
p1mh <- predict(fit1mh, newdata=xnew,
interval="confidence")
n1mh <- cbind(new_data, p1mh)

ggplot(Totaldata) +
  geom_point(aes(x=Force, y=Infected,
colour=Transportation)) +
  xlab("Force of Infection") +
  ylab("Prevalence") +
  ggtitle("Prevalence vs Time of Pandemic") +
  geom_smooth(data=n1ml, mapping=aes(x=Force,
y=p1ml), stat="identity", color='deepskyblue') +
  geom_smooth(data=n1mh, mapping=aes(x=Force,
y=p1mh), stat="identity", color='salmon') +
  theme_bw()

#model for maxinfected:
fit21 <- nls(MaxInfected ~ a*log(Force*b),
data=Totaldata, start=list(a=5, b=20))
summary(fit21)
cor(Totaldata$Infected, predict(fit21))

new_data <-
expand.grid(Force=seq(min(Totaldata$Force),
max(Totaldata$Force), length=1000))
p2 <- predict(fit21, newdata=new_data,
interval="confidence")
n2 <- cbind(new_data, p2)
n2

ggplot(Totaldata) +
  geom_point(aes(x=Force, y=MaxInfected, colour=City))
+
  xlab("Force of Infection") +
  ylab("Maximum of Infected People") +
  ggtitle("Force of Infection vs Maximum of Infected
People") +
  theme_bw()+
  geom_smooth(data=n2, mapping=aes(x=Force, y=p2),
stat="identity")

#model for time of pandemic
fit31 <- nls(Time ~ a* exp(-b*Force), data=Totaldata,
start=list(a=10000, b=7))
fit31 <- nls(Time ~ a* Force**b, data=Totaldata,
start=list(a=1000, b=0.1))

summary(fit31)
xnew <- expand.grid(Force=seq(min(Totaldata$Force),
max(Totaldata$Force), length=1000))
p3 <- predict(fit31, newdata=xnew,
interval="confidence")
n3 <- cbind(new_data, p3)

ggplot(Totaldata) +
  geom_point(aes(x=Force, y=Time, colour=City)) +
  xlab("Force of Infection") +
  ylab("Time of Pandemic") +
  ggtitle("Force of Infection vs Time of Pandemic") +
  geom_smooth(data=n3, mapping=aes(x=Force, y=p3),
stat="identity") +
  theme_bw()

#fit for time of pandemic with high and low migration:
migrhigh <- filter(Totaldata, Transportation=='high')
migrlow <- filter(Totaldata, Transportation=='low')

fit3ml <- nls(Time ~ a* exp(-b*Force), data=migrlow,
start=list(a=10000, b=7))
summary(fit3ml)
cor(Totaldata$Infected, predict(fit31))
xnew <- expand.grid(Force=seq(min(Totaldata$Force),
max(Totaldata$Force), length=1000))
p3ml <- predict(fit3ml, newdata=new_data,
interval="confidence")
n3ml <- cbind(new_data, p3ml)

fit3mh <- nls(Time ~ a* exp(-b*Force), data=migrhigh,
start=list(a=10000, b=7))
summary(fit3mh)
xnew <- expand.grid(Force=seq(min(Totaldata$Force),
max(Totaldata$Force), length=1000))
p3mh <- predict(fit3mh, newdata=new_data,
interval="confidence")
n3mh <- cbind(new_data, p3mh)

ggplot(Totaldata) +
  geom_point(aes(x=Force, y=Time,
colour=Transportation)) +
  xlab("Force of Infection") +
  ylab("Time of Pandemic") +
  ggtitle("Force of Infection vs Time of Pandemic") +
  geom_smooth(data=n3ml, mapping=aes(x=Force,
y=p3ml), stat="identity", color='deepskyblue') +
  geom_smooth(data=n3mh, mapping=aes(x=Force,
y=p3mh), stat="identity", color='salmon') +
  theme_bw()
```