Analysis and modelling of tuberculosis epidemiology in Western European cities. The case of Barcelona and Berlin.

Bachelor’s Thesis
Biological Systems Engineering

Author: Miguel Luengo Perez
Director: Clara Prats Soler
Co-Director: Bernat Puig Camps
June 2019
Abstract

Tuberculosis (TB) has been a major health concern for many years and it is still, to this day, one of the top 10 causes of death worldwide. The World Health Organization (WHO) estimates that TB caused 1.3 million deaths and that there were at least 10 million new cases in the year 2017. Even though the situation in Western Europe is not as severe as it is in other regions, the incidence in big cities in this area has been found to be many times higher than the mean for their countries, sometimes reaching values similar to those of high incidence countries.

The control programmes carried out by the local Health Agencies in these cities has been proven crucial to tackle the issue of tuberculosis. In this work, several of these Health Agencies have been contacted in order to analyse the collected data. The analysis of the TB data received from the cities of Berlin and Barcelona form the Robert Koch Institut (RKI) and the Agència de Salut Pública de Barcelona (ASPB) presented three relevant conclusions. The tendency of the disease is different for the two cities: in Barcelona the incidence has been declining for the last 10 years while it has been growing in Berlin. Despite this, the age-distribution of the disease is similar for both cities and in both cases the foreign population group had a great effect in the global incidence of these cities.

Taking these conclusions into account, an agent-based model (ABM) was adapted to reflect these characteristics centred on the effect of migration. The model was also simplified eliminating parameters considered non-essential in order to ease the subsequent process of parameterization. This model, once completed, can be used to carry out virtual experiments that help on the understanding of the situation of TB in these cities.

The adaptation of the model required the optimization of a parametrization process. A random forest machine learning algorithm was used for this purpose to obtain a multivariate function which can be used to analyse different sets of parameters in less time than the compute-intensive ABM.

In the end, although the overall pattern of both cities can be approximately fitted, it seems that the model cannot correctly fit the exact real values. This is probably because some of the parameters eliminated in the simplification process were necessary to explain the real data. On the other hand, the parametrization method proved to be a success reducing greatly the computation time needed to execute the analysis. Thus, the model is now ready to gradually incorporate the simplified parts and to be rapidly parameterized at each step, so that a more robust platform for running in silico experiments is released.
Resumen

La tuberculosis (TB) ha sido una importante preocupación en temas de salud durante muchos años y, aun a día de hoy, es una de las primeras 10 causas de muerte en el mundo. La Organización Mundial de la Salud (OMS) ha estimado que la TB causó 1,3 millones de muertes y que hubo por lo menos 10 millones de casos nuevos en el 2017. Aunque la situación en el oeste de Europa no es tan grave como en otras regiones, la incidencia en las grandes ciudades de esta área supera en gran medida a la media de sus países, en ocasiones llegando a valores similares a aquellos de países considerados de alta incidencia.

Los programas de control desarrollados por las agencias de salud públicas locales de estas ciudades han sido cruciales para abordar el problema de la tuberculosis. En este trabajo se ha contactado con varias de estas agencias para poder analizar los datos recogidos. El estudio de los datos recibidos de las ciudades de Berlín y Barcelona por parte del Robert Koch Institut (RKI) y la Agència de Salut Publica de Barcelona (APSB) mostraron tres conclusiones relevantes. La tendencia de la enfermedad es diferente en las dos ciudades: en Barcelona la incidencia ha ido disminuyendo estos últimos 10 años, mientras que en Berlín está en aumento. A pesar de ello, la distribución de edades de la incidencia es similar para ambas ciudades y en ambos casos la población extranjera tiene un efecto muy grande en la incidencia global de estas ciudades.

Teniendo en cuenta estas conclusiones, se adaptó un modelo basado en agentes (ABM) para reflejar estas características centrado en el efecto de la migración. El modelo también se simplificó, eliminando parámetros considerados no esenciales para facilitar el proceso de parametrización posterior. Una vez completado el modelo se podrá utilizar para realizar experimentos virtuales que nos ayuden a entender la situación de la TB en estas ciudades.

La adaptación del modelo requería una optimización del proceso de parametrización mencionado. Para este propósito se utilizó un algoritmo random forest de machine learning para obtener una función multivariable que se utilizó para analizar diferentes conjuntos de parámetros en un tiempo menor que el ABM que requiere más tiempo de computación.

Al final, aunque el patrón general de ambas ciudades se puede aproximar, parece ser que el modelo no puede ajustar correctamente los valores reales exactos. Esto puede ser debido a que algunos de los parámetros eliminados en la simplificación fuesen necesarios para poder explicar los datos reales. Aun así, el método de parametrización funcionó correctamente, reduciendo en gran medida el tiempo de computación necesario para ejecutar este tipo de análisis. Así pues, el modelo está preparado para incorporar gradualmente las partes simplificadas y poder parametrizarlo rápidamente en cada paso para poder obtener una plataforma fiable para realizar experimentos in silico.
Resum

La tuberculosi ha sigut una important preocupació en temes de salut durant molts anys i fins i tot avui dia és una de les primeres 10 causes de mort al món. La Organització Munidal de la Salut (OMS) estima que la TB va causar 1,3 milions de morts i que hi va haver almenys 10 milions de casos nous l'any 2017. Encara que la situació a l'oest d'Europa no és tan severa com en altres regions, la incidència a les grans ciutats d'aquesta àrea supera per molt la mitjana dels seus països, en alguns casos arribant a valors similars als de països considerats d'alta incidència.

Els programes de control desenvolupats per les agències de salut publica locals d'aquestes ciutats han sigut crucials per abordar el problema de la tuberculosi. En aquest treball s'ha contactat amb algunes d'aquestes agències per poder analitzar les dades recollides. L’estudi de les dades rebudes de les ciutats de Berlín i Barcelona per part del Robert Koch Institut (RKI) i l'Agència de Salut Publica de Barcelona (ASPB) van mostrar tres conclusions rellevants. La tendència de la malaltia és diferent a les dues ciutats: a Barcelona la incidència ha estat disminuint els últims 10 anys a diferència de Berlín, on ha estat augmentant. Així i tot la distribució d'edats de la malaltia és similar a les dues ciutats i en els dos casos la població estrangera té un efecte molt gran a la incidència global d'aquestes ciutats.

Tenint en compte aquestes conclusions es va adaptar un model basat en agents (ABM) per reflectir aquestes característiques centrat en l'efecte de la migració. El model també es va simplificar, eliminant paràmetres considerats no essencials per facilitar el procés de parametrització posterior. Una vegada complet el model es podrà utilitzar per dur a terme experiments virtuals que ens ajudin a entendre la situació de la TB en aquests ciutats.

L’adaptació del model requereix d’una optimització del procés de parametrització comentat prèviament. Per aquest propòsit es va utilitzar un algoritme random forest de machine learning per obtenir una funció multivariable que es va utilitzar per analitzar diferents conjunts de paràmetres en un temps menor que l’ABM que requereix més temps de computació.

Finalment, encara que el patró general de les dues ciutats es va poder aproximar, sembla que el model no es pot ajustar correctament els valors reals exactes. Això pot ser perquè alguns dels paràmetres eliminats a la simplificació fossin necessaris per explicar les dades reals. Així i tot, el mètode de parametrització va funcionar correctament reduint dràsticament el temps necessari per a aquest tipus d’anàlisi. Així doncs, el model esta lletter per incorporar gradualment les parts simplificades i poder parametritzar-lo ràpidament a cadascuna de les passes per obtenir una plataforma fiable per realitzar experiments in silico.
Appendices.................................................................................................................................35
A. Main ABM code.........................................................................................................................35
B. LHS Sampling Code................................................................................................................57
C. Random forest algorithm........................................................................................................59
List of figures

Figure 1: Progression of the cavities formed in the right lung by M. tuberculosis throughout an active tuberculosis disease. ................................................................................................................................. 1

Figure 2: Map of the estimated tuberculosis incidence worldwide in 2017, data corresponding to the Global Tuberculosis Report 2018 (WHO, 2018) ................................................................. 3

Figure 3: Flowchart of a simple SEIR compartmental model. .................................................................................................................. 5

Figure 4: Overview of the excel file obtained with the raw data from the TB cases and contacts from Barcelona from 2010-2015. .......................................................................................... 7

Figure 5: Overview of the excel file obtained with the raw data of the disaggregated population of Barcelona for the year of 2007. 2.1.2 Berlin.................................................................................................................. 8

Figure 6: Overview of the excel file obtained with the raw data from the TB cases and contacts from Berlin from 2008-2017..................................................................................................... 8

Figure 7: Overview of the excel file obtained with the raw data of the disaggregated population of the native men of Berlin, from 2008-2017........................................................................... 9

Figure 8: Evolution of the global incidence of TB disease in Barcelona and Berlin. Incidence was calculated as the number of new cases divided by the total population.............................. 10

Figure 9: Evolution of the specific incidence of TB disease in Barcelona and Berlin disaggregated by country of origin. ............................................................................................................. 11

Figure 10: Specific incidence in Barcelona(a) and Berlin(b) disaggregated by age. The incidences are the result of the mean for the given time period. .......................................................... 12

Figure 11: Population of Barcelona (a) and Berlin (b) disaggregated by age. The population given is the mean for each time period. ............................................................................................... 13

Figure 12: Specific incidence for the native(a) and foreign(b) population divided by the global incidence of each respective city. The incidences used are the mean for each time period .............. 14

Figure 13: Evolution of the incidence in the different districts of Barcelona(a) and Berlin(b) .................. 16

Figure 14: State diagram of the model, where the five states of individuals and possible transitions are shown. Output grey dotted arrows refer to deaths; input dotted arrows are the corresponding entrances of randomly selected individuals in order to keep population constant.............................................. 18

Figure 15: Cassandra interface for ABMs. .......................................................................................................................... 19

Figure 16: Evolution of the number of cases in the city of Barcelona and simulated results. .................. 25

Figure 17: Evolution of the number of cases disaggregated by origin in the city of Barcelona and simulated results. .................................................................................................................. 26

Figure 18: Evolution of the number of cases in the city of Berlin and simulated results. .......................... 26

Figure 19: Evolution of the number of cases disaggregated by origin in the city of Barcelona and simulated results. .................................................................................................................. 27

Figure 20: Evolution of the number of cases in the city of Barcelona and simulated results with migration dynamic and revised initial conditions. .............................................................. 29

Figure 21: Evolution of the number of cases disaggregated by origin in the city of Berlin and simulated results with migration dynamic and revised initial conditions. ........................................ 29

Figure 22: Evolution of the number of cases in the city of Barcelona and simulated results with revised initial conditions. .............................................................................................................. 30

Figure 23: Evolution of the number of cases disaggregated by origin in the city of Barcelona and simulated results with revised initial conditions. .............................................................. 30

Figure 24: Evolution of the number of cases in the city of Berlin and simulated results with migration dynamic and revised initial conditions. .............................................................. 31

Figure 25: Evolution of the number of cases disaggregated by origin in the city of Berlin and simulated results with migration dynamic and revised initial conditions. ................................ 31
List of tables

Table 1: Data used for the simulation of Barcelona. (*) Percentages with respect to the total number of TB sick people. (**) Percentages with respect to the total number of TB infected people..........................22
Table 2: Distribution of the new-infected properties according to the characteristics of the infectious. Adapted from the model for Ciutat Vella [12]. .................................................................22
Table 3: Values for the sickening probability for each year[15]. .................................................................23
Table 4: Maximum and minimum values for the parameters used in the sampling. The values were estimated from the parameters for the model of Ciutat Vella [12]. ................................................24
Table 5: Optimum values found for the studied parameters of the city of Berlin and Barcelona. ............25
Table 6: Migration data for the cities of Barcelona and Berlin considered for the model. Year 1 is 2005 for Barcelona and 2008 for Berlin. ........................................................................28
Table 7: Computation time for a simulation and a parameter analysis for the ABM code and its equivalent function.................................................................32
1. Introduction

1.1 Tuberculosis

Tuberculosis (TB) is an infectious disease caused by the bacillus *Mycobacterium tuberculosis*. It typically affects the lungs (pulmonary TB), but can also affect other sites (extrapulmonary TB). The disease is spread when people who are sick with pulmonary TB expel bacteria into the air, for example by coughing [1].

If the bacteria within the air droplets are inhaled they may reach the alveoli in the lungs. This leads to an immune response which ends with the absorption of the bacteria by macrophages through phagocytosis. This is the starting point of the tuberculosis infection. At this point, depending on the immune response of the individual, the initial infection may enter a control phase or evolve towards an active disease where the person will sicken. Therefore, two stages can be differentiated: the latent stage (latent tuberculosis infection or LTBI) and the active stage (active tuberculosis or TB).

During the latent stage, even though bacteria are present in the lungs, the bacterial load is low enough to remain under control and, therefore, the individual presents no symptoms and is not infectious. Once the immune response of an individual is overrun by the bacteria, the infection can no longer be controlled and the individual develops an active disease. A diagram of the evolution is presented in figure 1.

A relatively small proportion (5–10%) of the estimated 1.7 billion people infected with *M. tuberculosis* will develop TB disease during their lifetime [1]. This probability, however, is higher in people affected by certain risk factors such as undernutrition, diabetes, smoking, alcohol consumption and especially amongst people infected with human immunodeficiency virus (HIV).

During this active stage the sick individual can infect other people and will suffer from symptoms such as cough with blood, fever, weight loss, weakness... However, many of these symptoms may be mild for many months delaying the detection of the disease and resulting in the infection of more people.

![Figure 1: Progression of the cavities formed in the right lung by *M. tuberculosis* throughout an active tuberculosis disease as seen in the thesis of J.Vila [2]](image-url)
1.1.1 Diagnosis and treatment

Latent tuberculosis can be diagnosed rapidly through a tuberculin skin test (Mantoux Test). In case someone has been exposed to the TB bacillus a skin reaction to the antigens will appear after a period of 42 up to 72 hours.

In order to diagnose an active tuberculosis disease, there are two different procedures available: a chest X-ray or a microscopic examination of sputum. Patients who show a positive sputum test are referred to as smear-positive. This has been related to a higher infection power [3]. The cultivation of clinical samples allows the identification of resistant strains which will define the type of treatment. If the strain found is resistant to isoniazid and rifampicin (two of the antibiotics used in its treatment) it is defined as multidrug resistant (MDR). If it is also resistant to at least one other injectable TB drug it is considered extensively drug resistant (XDR).

Treatment against non-resistant strains comprises of a combination of 4 antibiotics and last from 6 to 9 months. The length of the treatment makes it difficult to commit to leading to treatment abandonment. This can aggravate the issue of resistant strains.

Diagnosis and successful treatment of people with TB averts millions of deaths each year (an estimated 54 million over the period 2000–2017) [1].

1.1.2 World situation

Worldwide, tuberculosis (TB) is still one of the top 10 causes of death, and the leading cause from a single infectious agent (above HIV/AIDS) [1]. Figure 2 shows how the estimated incidence of the disease was distributed worldwide in 2017.

TB occurs in every part of the world. In 2017, the largest number of new TB cases occurred in the South-East Asia and Western Pacific regions, with 62% of new cases, followed by the African region, with 25% of new cases. In fact, eight countries accounted for two thirds of the new TB cases: India, China, Indonesia, the Philippines, Pakistan, Nigeria, Bangladesh and South Africa [1].

Two main problems to be faced in high burden countries in order to control tuberculosis are the high percentage of non-diagnosed cases and the emergence of MDR strains. Today, there are still countries with less than 20 % of the cases diagnosed and treated. For instance, notification rate in Nigeria is estimated to be between 16 % and 20 % [4]. Besides, the World Health Organisation (WHO) estimates that, in 2017, there were more than 0.5 million cases worldwide resistant to rifampicin, the most effective first-line drug, 82 % of which were MDR.
1.1.3 Tuberculosis in European Cities

When analysing the situation in Europe, it is the countries in the eastern part of the WHO European Region that are most affected by the TB epidemic partly due to the high presence of MDR strains of the bacteria [5].

Most of the countries in Western Europe are considered to be low-incidence (<20 cases per 100000 population). However, even in these states, several publications have highlighted the higher notification and incidence rates in big cities or metropolitan areas. Cities such as London, Belgium or Barcelona all had an incidence over 20 in 2009 [6]. This is partially due to the high immigration from high-incidence countries which is often concentrated in these big cities.

Local Health Agencies have an essential role in the control of the disease in these cities. Among implemented strategies, Contact Tracing Programs have revealed to be crucial for (1) detecting and treating recent infections, and (2) collecting relevant personal and sociodemographic data that can become an important source of information for epidemiological and policymaking purposes.

1.2. Epidemiological models

Epidemiology studies the factors responsible for the spread of a disease. To achieve this and to fully understand the dynamics of transmission of the infectious agent, tools must be developed. Models are a simplification of reality, a conceptual representation of a complex system. They allow the understanding of complex phenomena by spotting the key aspects of the problem.
This makes them a very powerful tool for epidemiology. The aim of modelling in epidemiology is to better understand the diseases and their dynamics in order to improve the control strategies which can lead to the eradication of the disease[7].

There are two main measures or variables used in epidemiology in order to study the disease in a given population, which are the incidence and the prevalence. The **incidence** is the number of new cases per population at risk in a given time period. It serves as a measure of the probability of the occurrence of the disease. On the other hand, the **prevalence** describes the proportion of individuals in the population which have the disease at a given time.

For the case of TB, since the duration of the disease once it has been diagnosed is around 6 months the relation of the incidence and the prevalence can be an indicator of the efficiency of the healthcare system. If the prevalence is smaller than the incidence it means that most of the new detected cases in that year have already been treated at the end of the year.

Two strategies can be used when modelling a system: The Top-Down and the Bottom-Up approach. The Top-Down approach is based on applying a general theory to the specific, to generate quantitative conclusions. On the other hand the Bottom-Up strategy consists on modelling the low-level interactions, the behaviour of the elements, to observe the dynamics of the global system. [8]

As an example, let us say the flow of a tide must be simulated; a Top-Down approach would start by understanding the movement of a wave, it would then formulate a differential equation able to predict the position of each water molecule at each time. Instead, a Bottom-Up approach would assign each water molecule a trajectory to then observe the general pattern, the wave.

Historically, researchers have been inclined to use the Top-Down strategy and they developed several models that have been extensively used, such as the compartmental models based on differential equations (SIR, SEIRS, SIS...). These models consist of different pools of individuals characterized by the status of the disease. The S stands for susceptible; E stands for exposed, people who are in a latent state of the disease; I stands for infectious who may transmit the disease and R for recovered, individuals who have overcome the disease.

An example of such a model has been presented in figure 3. It is common for these models to maintain a constant population such as in this example, although this may not always be the case. It is for this reason that µ is used both as the death and birth rate so that the flux of mortality is equal to the one of natality.

The flux of individuals from susceptible to exposed is governed by β and depends on the contact rate between infected and susceptible and the infectiousness of the disease. The value of γ depends directly on the mean time it takes to become healed.
Figure 3: Flowchart of a simple SEIR compartmental model. The $S$ stands for susceptible; $E$ stands for exposed, people who are in a latent state of the disease; $I$ stands for infectious and $R$ for recovered. The other parameters are proportional to the flux between compartments.

The corresponding system of ordinary differential equations that mathematically describes these flows is:

\[
\begin{align*}
\frac{dS}{dt} &= \mu N - \mu S - \beta \frac{I}{N} S \\
\frac{dE}{dt} &= \beta \frac{I}{N} S - (\mu + \alpha) E \\
\frac{dI}{dt} &= \alpha E - (\mu + \gamma) I \\
\frac{dR}{dt} &= \gamma I - \mu R
\end{align*}
\]  

On the other hand, the Bottom-Up approach has been gaining importance recently. The Bottom-Up approach includes the heterogeneity in the elements, obtaining more realistic results and allowing a better understanding of the global behaviour derived from the individual diversity. Examples of such strategy are the Agent-based models (ABM) in which the behaviour of each of the agents is described based on its characteristics. One such example in this field is an ABM which simulates the evolution of tuberculosis in Barcelona developed by Prats et al [9]. This paper used ABM simulation in order to evaluate different screening strategies as well as evaluating the effect of interrupting the medical aid provided to irregular migrants could have on the city. It was seen that interrupting this service even if briefly, could have significant effects on the incidence of the whole city for at least ten years.

1.3 Aim, approach and outline

The principal aim of this project is to understand the main features of TB dynamics in Western European cities. In order to achieve this, TB epidemiological and sociodemographic data must be first obtained from control programs carried out by different European local health agencies. These datasets must be analysed to compare the tendencies and characteristics of TB in different cities focusing on sociodemographic factors and migration patterns. Once the results are obtained, an agent-based model must be adapted to incorporate any relevant conclusions and fully understand the system dynamics. Finally, the model developed can be used to conduct virtual experiments that can improve our understanding of the disease. By summary, the specific objectives that must be accomplished in order to achieve the main goal are the following:
- To obtain sociodemographic and TB-related epidemiological data of Western European cities, specifically of Barcelona and Berlin.

- To analyse and compare the tendencies and characteristics of TB incidence in these cities, focusing on the sociodemographic factors and migration patterns

- To adapt an agent-based model to incorporate major findings from the analysis

- To optimize a method for estimating the agent-based model parameters

This project has had the assistance of many different health agencies across Europe who shared the results of their control programmes. Data was received from the Subdirección General de Epidemiología of Madrid, the Rijksinstituut voor Volksgezondheid en Milieu (RIVN) of the Netherlands, the Robert Koch Institut (RKI) of Berlin and the Agència de Salut Publica of Barcelona.

In the end, after analysing the different datasets, the data from Barcelona and Berlin were used to perform the final study based on the similar structure of the data, the number of years recorded and the information for each case that was available. This analysis is presented in Chapter 2.

During the process of adapting the ABM, the project also counted with the collaboration of the Barcelona Supercomputing Center (BSC) since it was implemented in a new software being developed by the HPC modelling and simulation for Societal Challenges research group called Pandora.

In addition to the modification of the dynamics of the model, a parametrization process was required to adapt the model to the real data. A random forest machine learning algorithm was used for this purpose in order to obtain a multivariate function which can be used to analyse different sets of parameters in less time than the compute-intensive ABM. The description of the model and the parametrization process are presented in chapter 3. Finally, conclusions are summarized in chapter 4, as well as perspectives and further work.

The bachelor’s thesis started in June of 2018 and was developed during the following year:
- Data collection: June 2018 – July 2018
- Data analysis: September 2018 – January 2018
- Pandora translation of the model in BSC: September 2018 – March 2019
- Parameterisation and C model design: February 2019 - April 2019
- Parameterisation analysis: April 2019 – June 2019

This project is another step in a line of research developed by the BIOCOM-SC group. It build upon the work of the thesis of J. Vila[2], B. Puig[10] and M. Catala[11] and several publications of the group[9][3][4]. This work is not the end of this research, but it does make two interesting contributions: obtaining TB data from different European cities and optimizing a parameterisation method with the use of machine learning techniques.
2. Data analysis

The aim of the data analysis is to compare how tuberculosis affects two different cities, Berlin and Barcelona, by comparing the evolution of the disease in both cities as well as the pattern of infectiousness.

2.1 Surveillance data

2.1.1 Barcelona

Data from Barcelona was obtained from the surveillance carried out by the Agència de Salut Pública de Barcelona (ASPB) from 2005 to 2015. It contains 4346 cases with their information about age, gender, origin, year of diagnose, district where they live and different risk factors, like HIV, smoking, alcoholism and diabetes. Figure 4 shows a snapshot of the dataset file.

Demographic data from Barcelona were obtained from the Institut d’Estadística de Catalunya\(^1\) (IDESCAT). It was obtained disaggregated by age, sex, nationality and district. Figure 5 shows a snapshot of the Excel file extracted from the database.

---

\(^1\) [https://www.idescat.cat/](https://www.idescat.cat/)

![Figure 4: Overview of the excel file obtained with the raw data from the TB cases and contacts from Barcelona from 2010-2015.](image-url)
2.1.2 Berlin

Data from Berlin was provided by the Robert Koch Institut. It includes data from 2008 to 2017 and contains 3342 cases with their information about the age, sex, country of origin and the district where the case was reported. Figure 6 presents a snapshot of the dataset file.

Demographic data for the city of Berlin regarding the 2008-2017 period was obtained from Das Statistische Informationssystem Berlin-Brandenburg² (StaSi-BBB). It was disaggregated by age, sex, nationality and district. Figure 7 shows a snapshot of the Excel file extracted from the database.

---

² https://www.statistik-berlin-brandenburg.de/datenbank/inhalt-datenbank.asp
2.2 Data treatment

All raw data was provided in an excel format where each row contains the information available for each case such as age, sex or country of origin (see figures 4-7).

Data treatment was done mainly in MATLAB. This software allows to import data directly from excel as vectors and matrices which helps with the data analysis and the determination of epidemiological factors.

As explained in section 1.2, when studying the epidemiology of disease, one of the common variables used as a descriptor is the Incidence. In this work we will present the incidence as the number of new cases in a specific time period as a relative fraction per 100000 inhabitants.

When a population is divided into categories as, for instance, age group or origin, incidence is more meaningful when reported particularly for each group and as a fraction of the population of this group (i.e., TB cases within this category per 100000 inhabitants of this category). For example, when calculating the incidence of tuberculosis in a specific age group among the foreign population of a city, it would be divided by the foreign population in that age group.

In the following analysis, both kinds of incidences will be used. Therefore, we will use the term global incidence when referring to the number of new TB cases (either the total or within a certain group) with respect to the total population of the city; we will use specific incidence when referring to the number of new TB cases within a certain group (typically, age and/or origin) with respect to the population inside this group.
2.3 Results

2.3.1 Evolution of the disease

First off, we can compare the cities by studying the evolution of the incidence of the disease along 10 years. The number of cases in both cities is similar but as can be seen in figure 8, Barcelona has a higher global incidence than Berlin. Despite this the incidence of Barcelona has been decreasing during this time while it has been slightly growing in Berlin.

![Graph showing the evolution of TB incidence in Barcelona and Berlin](image)

*Figure 8: Evolution of the global incidence of TB disease in Barcelona and Berlin. Incidence was calculated as the number of new cases divided by the total population*

It is interesting to see the evolution of the incidence of foreign and native inhabitants separately. Firstly, we can see clearly from figure 9 that in both cases the incidence in the foreign population is much higher. In Barcelona, both the native and foreign incidence have been decreasing. In Berlin however, while the native incidence is almost negligible, the incidence among foreign population has been growing. From these data, we can deduce that the increase in global incidence can be mainly attributed to the incidence among the foreign population.
Besides the country of origin, age is also an important factor to consider when studying the incidence of tuberculosis, since the incidence can vary greatly between different age groups [1]. In figures 10a and 10b the specific incidence for both natives and foreigners is shown for different age groups. The first obvious pattern that can be seen is the higher incidence of the foreign population which was already noted when studying the evolution of the disease.

Another pattern that can be observed is that in both cities the incidence of the foreign population in the age range between 0 and 15 years is lower than for all other age groups. However, when considering the native population, the range from 0 to 5 years has a higher incidence and it is only the range between 5 and 15 which has the lower incidence.

When assessing the rest of the age groups for the native population, the incidence is also found to be greater at older ages. This also happens with the foreign population of Berlin but not with the one in Barcelona. This is due to the fact that the foreign population of Berlin is more aged as it can be seen in figures 11a and 11b.

Figure 9: Evolution of the specific incidence of TB disease in Barcelona and Berlin disaggregated by country of origin.

2.3.2 Age
Figure 10: Specific incidence in Barcelona(a) and Berlin(b) disaggregated by age. The incidences are the result of the mean for the given time period.
After acknowledging the different distributions that the disease shows for the foreign and native population, it is interesting to quantify the differences between both cities.

In order to compare the incidence of both cities, a relative incidence index was calculated by dividing the specific incidence of each age group by the total incidence of its city. The resulting number indicates if the specific incidence of an origin/age group is above (>1) or under (<1) the global incidence of the city. This will allow to compare the data without considering the difference in incidences between Berlin and Barcelona.

The results are shown in figures 12a and 12b. Firstly, for the foreign population of both cities, the incidence is much higher than the global incidence. In Berlin, the incidence among foreign population is approximately 4 times higher than the global incidence, while in Barcelona it is 3
times higher. This would support the ideas discussed in the analysis of the evolution of the disease where the substantial effect the foreign population has in Berlin had been noted.

As it has been discussed previously, the age group between 0 and 15 years has a lower incidence in both cities and in the case of Berlin the incidence is found to be higher in people over 75 years old.

When analysing the native population, similar patterns for both cities can be seen. In both cases in most age groups the value of the factor is below 1 meaning that their incidence is lower than the global one. In Barcelona, however, the factor is slightly higher, meaning that the native population has a greater effect to the global incidence. As had been noted earlier, the incidence is higher at older age groups and at the range between 0 and 5 years.

![Graph showing specific incidence divided by global incidence for native and foreign populations in Barcelona and Berlin.](figure12)

*Figure 12: Specific incidence for the native(a) and foreign(b) population divided by the global incidence of each respective city. The incidences used are the mean for each time period.*
2.3.3 District analysis

In figure 13, the incidence of the different districts of Barcelona (6a) and Berlin (6b) are represented. In the case of Berlin, all districts have similar incidences except Lichtenberg. The case of Lichtenberg can be explained because the Center for TB patients or people at risk of TB (Zentrum für tuberkulosekranke und -gefährdete Menschen) is located in this district. In this centre they carry out the screening of refugees, asylum-seekers and the homeless population of all Berlin which means all these cases are reported as happening in this district when this may not be the case. This can explain its unusually high incidence.

In the case of Barcelona, a district with unexpected high incidence is also found. The incidence of Ciutat Vella has been tried to be explained by attributing it to a number of factors such as mean income or mean surface for inhabitant. An analysis developed by Puig [10], however, found an interesting correlation when comparing the real incidence with a calculated one based on the incidences of the country of origin of the inhabitants of the district.

This would highlight the effect that immigration from high-incidence countries has in these cities. Therefore, one of the objectives of our model will be to evaluate the effect this type of migration really has.
Figure 13: Evolution of the incidence in the different districts of Barcelona (a) and Berlin (b)
2.4 Discussion

From this analysis several conclusions can be drawn. Firstly, that the major difference to be found between both cities is the global evolution of the disease. The incidence in Berlin is lower but it has been growing while in Barcelona the incidence is higher but has been declining. Also, for both cities, the specific incidence of foreign subpopulation greatly affects the evolution of the global incidence. In Berlin specifically, the increase in the global incidence can be attributed to an increasing incidence among foreigners. Finally, it can be concluded that although there are some minor differences in the age distribution of the disease the global pattern for both natives and foreigners is very similar when comparing both cities.

To sum up, the data analysis suggests that migration flows from high-incidence countries may play an important role on TB dynamics in these cities. The next step is to use an agent-based approach to check and quantify such hypothesis.
3. Agent-based Modelling of TB dynamics

Based on the results obtained from data analysis, the modelling analysis is centred in the effect of migration on the incidence. The model has been adapted from an ABM developed by Prats et al [9], which was already used to study TB dynamics in Ciutat Vella (Barcelona). In this model, each agent in the population may belong to the following classes: healthy, latently infected, sick, under treatment, and treated with a probability of relapse (i.e., get sick again without the need of being reinjected). Several variables and parameters such as age, sex and origin (native or foreign) were accounted for each person. The diagram in figure 14 represents the model dynamics.

![State diagram of the model](image)

*Figure 14: State diagram of the model, where the five states of individuals and possible transitions are shown. Output grey dotted arrows refer to deaths; input dotted arrows are the corresponding entrances of randomly selected individuals in order to keep population constant.*

The adaptation of the model maintains most of the dynamics of the previous one but simplifies some parts. Age and sex will no longer be studied as a parameter for the agents and the effects of the risk factors will not be included. These parameters are not considered essential for this analysis since it will be centred on the effect of migration. This will reduce greatly the number of parameters needed to run the simulation and will ease the process of parameterization.

Besides the simplifications the model will also add a migration flow dynamic to study its effect on the global population.

3.1 Pandora

The original model was implemented by Puig [10] in C. For this project it has been translated to C++ in order to work with the Pandora Library. This software was created by the HPC modelling and simulation for Societal Challenges research group of the Barcelona Supercomputing Centre. This tool is designed to implement ABM and to execute them in high-performance computing environments. Furthermore, Pandora is complemented by Cassandra (see figure 15), an application developed to visualize and analyse the results generated by the simulation created with Pandora.

Despite the migration to Pandora was successfully concluded, this software was not used in the final analysis because it is still being developed and the parallelization technology necessary for the high computational needs of the analysis was not ready. However, it remains an interesting tool which may be used in the future.
3.2 Description of the model with the ODD protocol

Agent-based Modelling has been used in multiple fields, from simulation in social science to business, and also biology and epidemiology. As a consequence of this variety, ABM models have had almost as many description procedures as models have been developed. In order to deal with all the different description systems and standardize the description of ABM models the ODD (Overview, Design concepts, and Details) protocol was defined [12]. This protocol consists of three blocks, which are subdivided into seven elements: Purpose, Entities, State variables and scales, Process overview and scheduling, Design concepts, Initialization, Input data and Submodels; nevertheless, not all of the elements have to be used in the description.

In this section, the ODD protocol that was developed for the original model by Gilabert et al. [13] and updated by Vila et al. [2] is presented with the simplifications that were explained previously.

3.2.1 Overview

Purpose:
The objective of this ABM is to analyse the evolution of pulmonary tuberculosis incidence in a community. It will be fitted to the cities of Barcelona and Berlin. The adequacy of the simulation results will be checked and compared to the epidemiological data. The final purpose of this ABM is to achieve through virtual experiments a better understanding of the effect of migration and evaluate the possible effects of epidemiology control strategies and public health decisions.

Entities, state variables and scales:
The fundamental entities in the model are persons. It is considered that persons can go through five infection states: healthy, infected (i.e., with a latent TB infection), sick (i.e., with an active TB), under treatment, and recovered. Persons in four out of the five states, all but healthy, are simulated as individuals. We consider that the characteristics of a healthy population remain
constant, on average (e.g., native/immigrant distribution among others). Moreover, a healthy collective is much larger than infected or sick collectives. Therefore, it is not necessary to control healthy individuals one-by-one; they are considered as a property of space (i.e., the number of healthy people in a spatial cell). A healthy person will acquire individuality once he/she enters the infection cycle. This strategy is an important optimization for drastically reducing the computing time. It was previously tested to provide results comparable to those obtained considering healthy people as individuals [3]. The state variables of the individuals mainly refer to their status in the tuberculosis infection cycle as well as the time spent in such phases and individual diagnostic time when getting sick as well as their origin (native or immigrant). A state diagram of the model is presented in figure 14. The population simulated is approximately 1600000 people, which represents all the people of Barcelona.

The model is partially spatially explicit, i.e., space is considered but it does not mimic the real space of the city. Simulation occurs in a discrete area of 2000 x 2000 spatial cells. Each spatial cell represents a local abstract space where two persons can meet. The time step is set to 1 day, and the simulation may cover up to a period of 1 or more years.

**Process overview and scheduling:**
The model was built in C and in C++ with the Pandora library. The C code is presented in appendix A. The simulation starts with the set-up of the initial configuration, where the population is randomly generated according to the input distributions of parameters and randomly distributed in the 2000 x 2000 grid. The model assumes discrete time steps of 1 day, as mentioned. Each day, all individuals execute a series of actions, and their variables are updated immediately. The individual actions may be: to move, to get infected, to get sick, to be diagnosed and to start a treatment, to abandon or to finish the treatment, to recover and to die. Some of the actions take place daily for all the individuals in the system (e.g., movement) and the other procedures are daily evaluated when necessary (e.g., the possibility of a sick individual to be diagnosed is daily assessed until it finally occurs). When an individual dies, a new healthy is generated adding 1 to the number of healthy people in a random spatial cell. At the end of each time step, global variables are updated.

### 3.2.2 Design Concepts

**Basic principles:**
The model is based on general knowledge of the natural history of tuberculosis. There are two essential characteristics of TB that must be taken into account in any epidemiological model. On the one hand, an infected individual does not necessarily develop an active disease; on average, only 10% of infected people become sick. Moreover, a person remains infected for an extended period and may develop active tuberculosis after several years, but the probability of developing the disease decreases with time [14][15]. Infected people are usually not diagnosed, except those detected in punctual screenings or in a contact tracing study. On the other hand, only TB sick can disseminate the infection. The infection rate increases if the patient is smear positive. Once a TB sick is diagnosed, the pharmaceutical treatment takes 6 months [1]. Once the treatment is finished, the possibility of getting sick again because of a TB reactivation remains at 1% for 2 years. If an infected is detected, it may be treated with preventive drug therapy. This treatment is longer than the one given to persons with active TB. It lasts 9 months and is
administered to infected individuals to prevent the development of an active disease once they have been detected during a screening process [1]. There is also a probability of relapse to the infected state that is calculated similarly to the first treatment.

**Emergence:**
Emerging phenomena are mainly related to long-term dynamics of the infection at the population level. On the one hand, only non-treated people with active TB can spread the disease. Therefore, diagnosis time is an essential parameter for the prevalence of the disease. On the other hand, infected persons may develop active tuberculosis a few years after the infection. Therefore, global consequences of particular conditions at a precise moment may be detected some years later.

**Interaction:**
Local interactions between healthy and sick individuals are explicitly modelled and crucial for the dynamics of the system. They refer to the meeting of two persons favoured by the spatial proximity between them and the possibility that one of those individuals with an active TB may infect the other person.

**Stochasticity:**
Randomness is introduced at all levels of the simulation. The initial distribution of individual properties is randomly executed according to input distributions. Movement is assumed to be random. Each action is associated with a certain probability and thus executed according to a stochastic number.

**Collectives:**
Two different collectives will be distinguished: the foreign and native population. These groups will differ in their diagnosis time and in the initial distribution of the disease. Due to the social patterns the chance for a native to infect a foreigner will be different than the chance to infect another native and the same applies for foreigners.

**Observation:**
The main output data being studied is the incidence for this reason the number of new cases for each year will be shown for the native and migrant population separately in order to compare it with the real data.

**3.2.3 Details**

**Initialization:**
For this particular study, most of the input parameters were taken from official reports. All percentages shown in Table 1 were used for calculating the configuration of initial population: rates sick, under treatment and recovered individuals per 100,000 inhabitants; mean diagnosis delay (MDD); mean treatment abandon rate. Other initial variables are assigned randomly (following the percentages shown in Table 1), and time spent in the infection state assigned.
Table 1: Data used for the simulation of Barcelona. (*) Percentages with respect to the total number of TB sick people. (**) Percentages with respect to the total number of TB infected people.

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infected foreign chance**</td>
<td>73,57</td>
<td>%</td>
</tr>
<tr>
<td>Sick foreign chance**</td>
<td>71,57</td>
<td>%</td>
</tr>
<tr>
<td>Total annual mortality</td>
<td>0,83</td>
<td>%</td>
</tr>
<tr>
<td>Diagnosis Delay (foreign)</td>
<td>46</td>
<td>days</td>
</tr>
<tr>
<td>Diagnosis Delay (native)</td>
<td>63</td>
<td>days</td>
</tr>
<tr>
<td>Diagnosis delay standard deviation</td>
<td>4</td>
<td>days</td>
</tr>
<tr>
<td>Treatment abandon rate*</td>
<td>2,20</td>
<td>%</td>
</tr>
</tbody>
</table>

Submodels:

Move: All persons can move randomly through the surrounding local space, once a day.

Get infected: If there is a number of individuals susceptible to TB (healthy and treated) different from zero in the proximity of a sick individual, meaning one of the 4-neighbouring spatial cells, this sick person may infect one of them with a certain probability. The total of susceptible neighbouring individuals is computed and then the infection process is repeated as many times as healthy and treated people have been found. The infection probability depends on the type of TB disease that the sick person has, either smear-positive or smear-negative. A smear-positive case is considered to double the infection probability. The value of the infection probability is closely linked to the spatial and temporal scales, i.e., the probability of infection is inseparable from the spatio-temporal scale. A change in any of these scales entails the revision of its value. Therefore, it is not a real infection probability when a sick individual meets a healthy person, but an effective infection probability given the particular spatio-temporal constraints. In this case (2000 x 2000 spatial cells and 1600000 individuals), the value of this probability was fixed at 49.7%. Once a person is infected, a newly infected individual is created with the properties assigned according to the characteristics of the infectious. Whether the new person will be set to native or immigrant will depend on the characteristics of the infectious individual. Those probabilities are summarized in table 2. The infection time of the new individual is set at 0 and starts increasing with each time step.

Table 2: Distribution of the new-infected properties according to the characteristics of the infectious. Adapted from the model for Ciutat Vella [2].

<table>
<thead>
<tr>
<th>Origin of cases</th>
<th>Native</th>
<th>Foreign</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native</td>
<td>68%</td>
<td>14%</td>
</tr>
<tr>
<td>Foreign</td>
<td>32%</td>
<td>86%</td>
</tr>
</tbody>
</table>

Get sick: Once infected, the individual may develop active TB according to a particular annual probability that decreases with infection time during the 7 years post-infection.
In order to model this behaviour, the yearly probability of getting sick was modelled based on a particular annual probability that decreases with infection time during the 7 years post infection[16]. It is neglected for the subsequent years(t>7 years).

Table 3: Values for the sickening probability for each year[16].

<table>
<thead>
<tr>
<th>Year</th>
<th>Sickening probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0830</td>
</tr>
<tr>
<td>2</td>
<td>0.0110</td>
</tr>
<tr>
<td>3</td>
<td>0.0016</td>
</tr>
<tr>
<td>4</td>
<td>0.0016</td>
</tr>
<tr>
<td>5</td>
<td>0.0016</td>
</tr>
<tr>
<td>6</td>
<td>0.0016</td>
</tr>
<tr>
<td>7</td>
<td>0.0016</td>
</tr>
</tbody>
</table>

The chance of becoming a TB sick individual is evaluated at each time step for all infected persons. Globally, the average of 10 % of infected developing an active disease is satisfied. The possibility of relapse (getting sick again) for recovered patients is also evaluated daily according to the individual relapse probability (see below). Once a person gets sick, the disease time counter starts running until the individual diagnostic time is reached.

Be diagnosed and start treatment: Each individual has a particular diagnostic time that is randomly assigned when getting sick. These individual times are assumed to be distributed following a normal distribution centred around the mean diagnosis time and standard deviation shown in table 1. When the sick time counter reaches these values, the individual is diagnosed. Once diagnosed, medical treatment is assumed to start and TB to stop spreading. Individual time under treatment is initially fixed at 0 and then updated at each time step.

Abandon the treatment: There is a certain probability that an individual abandons the treatment before finishing it. This possibility is evaluated daily for each patient under treatment, according to the input abandonment probability. If a person leaves treatment during the initial 15 days post-diagnosis, he/she becomes ill again. If he/she abandons the treatment after 15 to 180 days postdiagnosis, the model will consider him/her to be recovered but with a certain probability of relapse during the following 2 years. This probability is assumed to decrease linearly from the 100 % of a 15-day abandonment to the 1 % of the 180-day treatment period.

Recover: When a sick individual is diagnosed and treated for 180 days, he/she becomes recovered and a relapse probability of 1 % is assigned (the chance of getting sick before being considered healthy). 2 years after the diagnosis, the individual is considered to be healthy

Die: Each individual has a certain probability of dying each step. These probabilities are fixed using demographic data from Barcelona. Accordingly, the daily dying probabilities is considered to be $4.278 \cdot 10^{-5}$, which is a simplification of the real mortality distribution. Furthermore, TB sick people have a distinct probability of dying from tuberculosis. This probability is evaluated daily for each sick individual, taking into account that 40 % of non-treated TB sick may die in 5 years.
Each time an individual dies, a new healthy individual is introduced into the simulation world in a random position.

3.3 Parameterization

In order to parameterise the model, the codes developed by Català [11] will be used and optimized. These codes are based on the original C model and have been adapted with the previously discussed changes for this analysis. They use Machine Learning algorithms in order to predict the evolution of the disease by comparing the results of the model with real data.

The parameters being studied are the following:
- Percentage of initial infected
- Probability of infecting
- Probability of getting sick
- $p_{\text{sicken\_all}}$ (term that multiplies the whole distribution of sickening probability increasing or decreasing the total probability. Its default value is 1)

In order to use the Machine Learning algorithms first a data sampling must be done. For each parameter a minimum and maximum value is inputted and a Latin Hypercube Sampling (LHS) method is executed. The LHS code is shown in Appendix B, and the threshold values of each parameter are shown in Table 4. This method assures a better exploration of the parameter space than other methods such as Monte Carlo [11].

Table 4: Maximum and minimum values for the parameters used in the sampling. The values were estimated from the parameters for the model of Ciutat Vella [2].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage of initial infected</td>
<td>0,02</td>
<td>0,1</td>
</tr>
<tr>
<td>Probability of infection</td>
<td>0,05</td>
<td>0,15</td>
</tr>
<tr>
<td>Probability of sickening</td>
<td>0,05</td>
<td>0,08</td>
</tr>
<tr>
<td>$p_{\text{sicken_all}}$</td>
<td>0.5</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Once the sampling is executed, 100 combinations of parameters will be obtained within the maximum and minimum value inputted for each parameter, covering the whole parameters space. Then, for each combination of parameters, 5 simulations are run. In each simulation the number of new native and migrant sick for each year are outputted in a csv format together with the combination of parameters used.

With the data obtained, together with the real data, a random forest regressor algorithm is applied to obtain a multivariate function with all the outputs and all the parameters selected. The corresponding code is shown in Appendix C. This function can be used to analyse different sets of parameters in less time than the compute-intensive ABM.
The outputs of the function of new native and foreign sick for different sets of parameters are compared to the real data and the square error is obtained. Then the set of parameters which has the least error is selected as the optimum.

3.4 Simulation results

Carrying out the parameterization process explained in previous section, the optimum parameters for both Berlin and Barcelona were obtained and are shown in table 5.

*Table 5: Optimum values found for the studied parameters of the city of Berlin and Barcelona.*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Barcelona</th>
<th>Berlin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage of initial infected</td>
<td>0.02</td>
<td>0.02</td>
</tr>
<tr>
<td>Probability of infection</td>
<td>0.1</td>
<td>0.08</td>
</tr>
<tr>
<td>Probability of sickening</td>
<td>0.08</td>
<td>0.02</td>
</tr>
<tr>
<td>p_sicken_all</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The comparison between the real data and the model for Barcelona is presented in figure 16. Both share a similar behaviour although the number of cases is slightly higher in the simulated data.

In figure 17, where the disaggregated data is presented, we can see the cause of this higher values. The model results for native population fit the real data more or less accurately. However, even though both the results from the foreign population and the real data share the same decreasing tendency the values for the model are higher which explains the effects seen in the figure 16.

*Figure 16: Evolution of the number of cases in the city of Barcelona and simulated results.*
When studying the case of Berlin presented in figures 18 and 19, the model does not seem to have captured the general tendency nor the differences between the two groups. This may be caused by the relevance migration has in the evolution of the disease in Berlin since this dynamic was not implemented in the process of parametrization.
Figure 19: Evolution of the number of cases disaggregated by origin in the city of Barcelona and simulated results.

Both models showed an overestimation of the number of cases for the foreign population. In the real data the early years show more cases for the native than the foreign population. However, in the model the opposite thing happens. In order to adjust this, the initial conditions were revised changing the chance of a sick or infected individual to be assigned to foreign or native when initializing the simulation.

As well as this revision of the initial conditions, a migration dynamics was added to better explain the different dynamics of the foreign and native population.

3.5. Initial conditions and incorporation of migration

3.5.1 Initial conditions
The initial proportion of foreign and native individuals with the disease was estimated by dividing the number of foreign cases in the first year by the total cases in that year. That value was considered the chance for a sick or infected individual to be foreign. The values for the probabilities of infected and sick origin were recalculated specifically for Berlin and Barcelona and were changed to 45% and 39%, respectively.

3.5.2 Model’s update to incorporate migration

To better simulate the dynamics of the foreign population a migration dynamic was added into the model. At the setup a frequency of migration in days is defined. If enough time has passed since the last migration, before the step starts a number of individuals will be added with a given probability of pertaining to one of the groups (S, E, I, T or R) and a number of individuals will be selected randomly and eliminated from the simulation to represent emigration.

The yearly number of entrances and exits from Barcelona was obtained from the IDESCAT database of external migration.
The data from Berlin was obtained from the StatIS-BBB database. Since there was no direct information on the external migration it was estimated considering the difference between the foreign population of each year. This difference does not consider the births since they separated in the database as natives with foreign background. The migration data for both cities is presented in table 6.

Table 6: Migration data for the cities of Barcelona and Berlin considered for the model. Year 1 is 2005 for Barcelona and 2008 for Berlin.

<table>
<thead>
<tr>
<th>Year</th>
<th>Barcelona</th>
<th></th>
<th></th>
<th>Berlin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Immigration</td>
<td>Emigration</td>
<td>Immigration</td>
<td>Emigration</td>
</tr>
<tr>
<td>1</td>
<td>58855</td>
<td>39136</td>
<td>48.073</td>
<td>38.019</td>
</tr>
<tr>
<td>2</td>
<td>49496</td>
<td>23388</td>
<td>49.203</td>
<td>41.215</td>
</tr>
<tr>
<td>3</td>
<td>46164</td>
<td>34873</td>
<td>56.028</td>
<td>59.083</td>
</tr>
<tr>
<td>4</td>
<td>40365</td>
<td>34900</td>
<td>61.462</td>
<td>59.091</td>
</tr>
<tr>
<td>5</td>
<td>34593</td>
<td>36374</td>
<td>71.564</td>
<td>45.077</td>
</tr>
<tr>
<td>6</td>
<td>37564</td>
<td>34570</td>
<td>79.360</td>
<td>48.919</td>
</tr>
<tr>
<td>7</td>
<td>46366</td>
<td>36439</td>
<td>87.458</td>
<td>52.376</td>
</tr>
<tr>
<td>8</td>
<td>49178</td>
<td>47161</td>
<td>97.287</td>
<td>61.463</td>
</tr>
<tr>
<td>9</td>
<td>44611</td>
<td>48418</td>
<td>114.338</td>
<td>67.136</td>
</tr>
<tr>
<td>10</td>
<td>59493</td>
<td>29976</td>
<td>133.417</td>
<td>73.829</td>
</tr>
</tbody>
</table>

The distribution of the disease amongst the immigrants was approximated by considering the five most important countries for each city and computing a weighted mean of the incidence in each country depending on the number of immigrants entering for each one.

The number of individuals which had been treated or were undergoing treatment was calculated based on the notification rate for each country.

Since it is difficult to ascertain the number of individuals with a latent disease, it was evaluated as a new parameter substituting the p_sicken_all parameter which proved unnecessary due to it having a value of 1 for both simulations. It was evaluated as the probability of an individual arriving being infected and the maximum and minimum values for the sampling process were 0,05 and 0,3 respectively.
3.5.3 Results of Barcelona

The results of the parameterization with the updated initial conditions and migration are presented in figures 20 and 21. Even though the native population is fitted correctly the number of cases for the foreign population are overrepresented. It seems that the migration dynamics increases the incidence for the foreign population excessively. For this reason, the process was repeated with the new initial conditions but without the migration dynamics.

Figure 20: Evolution of the number of cases in the city of Barcelona and simulated results with migration dynamic and revised initial conditions.

Figure 21: Evolution of the number of cases disaggregated by origin in the city of Barcelona and simulated results with migration dynamic and revised initial conditions.
In figures 22 and 23 the results of this test are presented. With these factors the native and foreign populations of the simulation obtain a similar value throughout the ten years. This also happens for the real populations, but the simulated results have lower values than the real data.

![Figure 22: Evolution of the number of cases in the city of Barcelona and simulated results with revised initial conditions.](image)

![Figure 23: Evolution of the number of cases disaggregated by origin in the city of Barcelona and simulated results with revised initial conditions.](image)

With these analyses we can conclude that the best fit for Barcelona is close to be obtained, but it is not the optimal one, yet. These may be caused because we are assuming a homogenous city when really, as was presented in the district analysis of Barcelona, there are districts like Ciutat Vella which show real different patterns. In this analysis, for example, the migration was considered random through all the surface while really it is concentrated in districts such as Ciutat Vella.
3.5.4 Results of Berlin

Preliminary test showed that the number of cases was being overestimated. A revision of the spatial conditions suggests that the population density had been incorrectly assumed to be the same as in Barcelona, while really density is 4 times smaller. Therefore, the grid was increased from 2000 to 5477 spatial cells.

The results of the parametrization process with new initial conditions, revised population density and the inclusion of migration dynamics are presented in the figures 24 and 25. They show that the patterns of the disease are more accurate adding these factors. Although the fit is still not completely accurate, the tendency of a growing incidence in the foreign population and a constant value for the native one which was noted in the city of Berlin is now correctly described by the model.

Figure 24: Evolution of the number of cases in the city of Berlin and simulated results with migration dynamic and revised initial conditions.

Figure 25: Evolution of the number of cases disaggregated by origin in the city of Berlin and simulated results with migration dynamic and revised initial conditions.
3.6. Discussion

In the end, although the overall pattern of both cities can be approximately fitted, it seems that the model cannot fit the real values closely enough. This could be due to the fact that the model was simplified to a great degree and maybe some of the parameters that were not considered essential were needed to correctly explain the real data. A further analysis adding this factors one by one would be needed to see which of the factors are truly needed.

Another possibility is that the sickening chance which is considered equal for all individuals is being overestimated for the foreign population. The study of TB in Nigeria conducted by the BIOCOM-SC group showed that some individuals have developed a natural resistance to the disease reducing the chance of this group to sicken once infected.

For the city of Barcelona specifically a deep analysis of the relevance of the effect that the difference between districts has would also be interesting to better understand its dynamics.

Besides the parametrization results, the machine learning tools used for this analysis proved an interesting tool. In table 7 the improvement in computation time between using the ABM and the equivalent multivariate function obtained with the random-forest machine learning algorithm on the parameterization process are presented. This method allows to study a large range of different combinations of parameters in very little time. It is also a versatile tool since it can be used to study any number of parameters necessary and analyse the result of one or more outputs.

Table 7: Computation time for a simulation and a parameter analysis for the ABM code and its equivalent function obtained with the random-forest machine learning algorithm. The processor used is an Intel Core i7 of 2.40 GHz and a memory of 6 GB 2401 MHz.

<table>
<thead>
<tr>
<th></th>
<th>Single 10-year simulation</th>
<th>Full parameter analysis (10000 simulations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABM code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barcelona</td>
<td>803 seconds</td>
<td>93 days</td>
</tr>
<tr>
<td>Berlin</td>
<td>2234 seconds</td>
<td>258 days</td>
</tr>
</tbody>
</table>
| Equivalent multivariate function | 0,13 seconds | 20 minutes }


4. Conclusions

In the end several conclusions can be drawn answering the general objectives that were presented in point 1.3:

- Data from different Western European Cities have been obtained. Among them, most detailed and complete are those corresponding to the cities of Barcelona and Berlin, which have been further used for analysis and modelling purposes.

- Data of Berlin and Barcelona have been analysed and compared. From this analysis, we can highlight that:

  - The evolution of the disease in Barcelona and Berlin is different: it has been decreasing in Barcelona and growing in Berlin for the last 10 years.

  - The age distribution of TB incidence in both cities is very similar.

  - In both cities, migration plays an important role in the dynamics of TB incidence, especially in the city of Berlin where it can be the cause of the observed increase.

  - An agent-based model has been adapted and drastically simplified for focusing on the differences between TB incidence among native and foreign. In addition, the possibility of incorporating migration flows have been implemented.

  - A method for estimating ABM parameters by using a random forest machine learning algorithm has been optimized. This method greatly reduces computational cost of parameter estimation. It can be used for estimating any number and kind of parameters and analyse one or more different outputs. In addition, it can be easily exported for the parameterization of other ABMs.

4.1 Perspectives and further work

This work is one step more in a bigger long-term project. It has allowed contact with different health agencies with which we can work in the future to study different cities.

Additionally, several interesting tools have been developed which could be used in the future including the implementation on the Pandora software and the optimization of the parameterization method. The latter will be useful specially with the next steps to be taken in this project. Amongst them, we can highlight the analysis of the model adding more parameters which were not considered during the simplification. With the improvement in computation time this process will now be more feasible. Once the model is accurately parametrized for these cities it could be used to perform virtual experiments to improve our understanding of TB.
References


Appendices
A. Main ABM code

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <assert.h>
#include <math.h>
#include "IBMheader.h"
#include "list.h"

int grid[CELLS][CELLS] = {{0}};
/* Initial values */

/* BARCELONA */
int num_healthy = 1529469;
int num_infected;
float p_ini_infected;
int num_sick = 39;
int num_treatment = 161;
int num_treated = 1840;
*/

/* BERLIN */
int num_healthy = 3380126;
int num_infected;
float p_ini_infected;
int num_sick = 39;
int num_treatment = 161;
int num_treated = 1840;

//CIUTAT VELLA
/*
int num_healthy = 100488;
int num_infected;
float p_ini_infected;
int num_sick = 3;
int num_treatment = 11;
int num_treated = 121;
*/

/* Number of migrants in a year and frequency of arrival */
int migration_frequency = 1;

int yearly_immigrants[10] = {49203,56028,61462,71564,79360,87458,97287,114338,133417,104411}; // berlin
int yearly_emigrants[10] = {41215,59083,59091,45077,48919,52376,61463,67136,73829,62477};

int yearly_immigrants[10] = {51809,58645,61373,59493,44611,45077,48919,52376,61463,67136,73829,62477}; // barcelone

int new_imm;
int new_em;
int imm_counter = 1;

/* Probability for each migrant to pertain to a certain group */

/* Barcelona */
float p_immS;
```

float p_immE;
float p_immI = 0.001118983;
float p_immT = 0.000411291;
float p_immR = 0.001645166;
*/

//Berlin
float p_immS;
float p_immE;
float p_immI = 0.000343013;
float p_immT = 0.000140865;
float p_immR = 0.000563459;

int n_new_nat_sick;
int n_new_for_sick;
int n_new_infected;
int n_heal_infected;
int n_dead_infected;
int n_new_treatment;
int n_new_treated;
int n_recovered;

;/* Probabilities of origin */

//CIUTAT VELLA
//float p_infected_foreign = 0.7357;
//float p_sickened_foreign = 0.7157;

//BARCELONA
//float p_infected_foreign = 0.39;
//float p_sickened_foreign = 0.39;

//BERLIN
float p_infected_foreign = 0.45;
float p_sickened_foreign = 0.45;

/* Characteristic times in days */

int t_infected_max = 7*365;
int t_treatment_min = 15;
int t_treatment_max = 180;
int t_to_healthy = 720; // Time after diagnose to consider healthy.

/* Probability of smear positive case */
float p_smear = 0.22;

/* Diagnose delay mean and std.
 * Mean has two components [autochton,foreign]
 */
int diagnose_mean[2] = {46,33};
int diagnose_std = 4;

/* Probability to abandon treatment before finishing it */
float p_abandon = 0.022;
float p_relapse_min = 0.01;

/* Probability to infect an close individual */
float p_infect;
/* Probability of origin and gender of new infected depending
 * of origin and gender of the infector
 * [Native] [Foreign] (Case)
 * [Native] [Foreign]
 * (Contact)
 */

float p_origin[2][2] = {{0.68, 0.14},
                        {0.32, 0.86}};

/* Probability to become sick depending on time infected */
float p_sicken_all;
float beta;
float p_sicken[7] = {0.0830, 0.0119, 0.0016, 0.0016, 0.0016, 0.0016, 0.0016};
//float f_sicken;

/* Superior value of age group and probability to die for each group */
double p_die = 4.27759e-5;
double p_die_sick = 2.192e-4;

void main(){
    /* Initialize the random seed to generate different simulations */
    clock_t begin = clock();
    srand(time(NULL));
    int ii;
    FILE *inputfile;
    inputfile = fopen("data/input.txt","r");
    fscanf(inputfile,"%f", &p_ini_infected);
    num_infected = (num_healthy+num_sick+num_treatment+num_treated)*p_ini_infected;
    fscanf(inputfile,"%f", &p_infect);
    fscanf(inputfile,"%f", &p_immE);
    p_immS = 1 - (p_immE+p_immI+p_immT+p_immR);
    printf("%f n", p_immS);
    printf("%f n", p_immE);
    fclose(inputfile);

    /*switch(f_sicken) {
     case 1:
     p_sicken[0] = 0.0280 * p_sicken_all;
     p_sicken[1] = 0.0220 * p_sicken_all;
     p_sicken[2] = 0.0177 * p_sicken_all;
     p_sicken[3] = 0.0134 * p_sicken_all;
     p_sicken[4] = 0.0093 * p_sicken_all;
     p_sicken[5] = 0.0060 * p_sicken_all;
     p_sicken[6] = 0.0036 * p_sicken_all;
     case 2:
     p_sicken[0] = 0.0250 * p_sicken_all;
     p_sicken[1] = 0.0250 * p_sicken_all;
     p_sicken[2] = 0.0100 * p_sicken_all;
     p_sicken[3] = 0.0100 * p_sicken_all;
     p_sicken[4] = 0.0100 * p_sicken_all;
     p_sicken[5] = 0.0100 * p_sicken_all;
     p_sicken[6] = 0.0100 * p_sicken_all;
     case 3:
     p_sicken[0] = 0.0285 * p_sicken_all;
     p_sicken[1] = 0.0215 * p_sicken_all;
     p_sicken[2] = 0.0163 * p_sicken_all;
     p_sicken[3] = 0.0123 * p_sicken_all;
    */
case 4:
p_sicken[0] = 0.0250 * p_sicken_all;
p_sicken[1] = 0.0250 * p_sicken_all;
p_sicken[2] = 0.0177 * p_sicken_all;
p_sicken[3] = 0.0126 * p_sicken_all;
p_sicken[4] = 0.0089 * p_sicken_all;
p_sicken[5] = 0.0045 * p_sicken_all;
p_sicken[6] = 0.0034 * p_sicken_all;

case 5:
p_sicken[0] = 0.0270 * p_sicken_all;
p_sicken[1] = 0.0230 * p_sicken_all;
p_sicken[2] = 0.0188 * p_sicken_all;
p_sicken[3] = 0.0145 * p_sicken_all;
p_sicken[4] = 0.0101 * p_sicken_all;
p_sicken[5] = 0.0056 * p_sicken_all;
p_sicken[6] = 0.0010 * p_sicken_all;

FILE *outputfile;
outputfile = fopen("data/output.txt","w");

int i, t, t_max = 365;
int yearmax = 10;
int j,k,count = 0;

/* List for each state */
List *Elist = malloc(sizeof(List));
List *Ilist = malloc(sizeof(List));
List *Tlist = malloc(sizeof(List));
List *Rlist = malloc(sizeof(List));

/* Nodes to read the list */
ListNode *node, *temp;
initialize_simulation(Elist, Ilist, Tlist, Rlist);

/* Initialize grid */
for(j = 0; j < CELLS; j++)
    for(k = 0; k < CELLS; k++)
        count += grid[j][k];

/* Starts */
for(i = 0; i < yearmax; i++)
    /* Years */
    /* Restart counters each year */
    n_new_nat_sick = 0;
    n_new_for_sick = 0;
    n_new_infected = 0;
    n_heal_infected = 0;
    n_dead_infected = 0;
    n_new_treatment = 0;
    n_new_treated = 0;
    n_recovered = 0;

    for(t = 0; t < t_max; t++)
        /* Days */
/* Sick people dynamics */
migration(Elist,Ilist,Tlist,Rlist,i,count);
node = Ilist->head;

while(node != NULL) {
    temp = node->next;
    sick_update(Ilist,node,Elist,Tlist);
    node = temp;
}

/* Infected people dynamics */
node = Elist->head;

while(node != NULL) {
    temp = node->next;
    infected_update(Elist,node,Ilist);
    node = temp;
}

/* Treatment people dynamics */
node = Tlist->head;

while(node != NULL) {
    temp = node->next;
    treatment_update(Tlist,node,Rlist);
    node = temp;
}

/* Treated people dynamics */
node = Rlist->head;

while(node != NULL) {
    temp = node->next;
    treated_update(Rlist,node,Ilist);
    node = temp;
}

/* Healthy people dynamics */
count = 0;
for(j = 0; j < CELLS; j++)
    for(k = 0; k < CELLS; k++)
        count += grid[j][k];
move_healthy();

print_num_states(Elist,Ilist,Tlist,Rlist,count);
printf("\n");
print_yearly_parameters();
printf("\n");

//ALL variables:
//fprintf(outputfile, "%d %d %d %d %d %d %d\n", n_new_nat_sick, n_new_for_sick,
//n_new_infected, n_heal_infected, n_dead_infected, n_new_treatment, n_new_treated, n_recovered);

//New Sick:
//printf("New Native Sick: %i\n", n_new_nat_sick);
//printf("New Foreign Sick: %i\n", n_new_for_sick);
//fprintf(outputfile, "%i %i\n", n_new_nat_sick, n_new_for_sick);

/*printf("Infected: %i, ", Elist->logicalLength);
printf("new sick: %i\n", n_new_sick);*/
}
fclose(outputfile);
clock_t end = clock();
double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
printf("Elapsed: %f seconds \n", time_spent);
}

void print_num_states(List *Elist, List *Ilist, List *Tlist, List *Rlist, int count)
{
    printf("Tot: %d, ", count+Elist->logicalLength+Ilist->logicalLength+Tlist->logicalLength+Rlist->logicalLength);
    printf("S: %d, ", Elist->logicalLength);
    printf("I: %d, ", Ilist->logicalLength);
    printf("T: %d, ", Tlist->logicalLength);
    printf("R: %d\n", Rlist->logicalLength);
}

void print_yearly_parameters(void)
{
    printf("new native sick: %d, ", n_new_nat_sick);
    printf("new foreign sick: %d, ", n_new_for_sick);
    printf("new infected: %d, ", n_new_infected);
    printf("heal infected: %d, ", n_heal_infected);
    printf("new treatment %d, ", n_new_treatment);
    printf("new treated: %d, ", n_new_treated);
    printf("recovered: %d, ", n_recovered);
    printf("dead infected: %d\n", n_dead_infected);
}

void list_new(List *list, int elementSize, freeFunction freeFn)
{
    assert(elementSize > 0);
    list->logicalLength = 0;
    list->elementSize = elementSize;
    list->head = list->tail = NULL;
    list->freeFn = freeFn;
}

void list_destroy(List *list)
{
    ListNode *current;
    while(list->head != NULL) {
        current = list->head;
        list->head = current->next;
        if(list->freeFn) {
            list->freeFn(current->data);
        }
        free(current->data);
        free(current);
    }
}

void list_prepend(List *list, void *element)
{
    ListNode *node = malloc(sizeof(ListNode));
    node->data = malloc(list->elementSize);
    node->prev = NULL;
    memcpy(node->data, element, list->elementSize);
    if(list->logicalLength == 0) {
        list->head = list->tail = node;
    } else {
        node->next = list->head;
        list->head->prev = node;
        list->head = node;
    }
    list->logicalLength++;
void list_append(List *list, void *element)
{
    ListNode *node = malloc(sizeof(ListNode));
    node->data = malloc(list->elementSize);
    node->next = NULL;
    memcpy(node->data, element, list->elementSize);

    if(list->logicalLength == 0) {
        list->head = list->tail = node;
    } else {
        node->prev = list->tail;
        list->tail->next = node;
        list->tail = node;
    }
    list->logicalLength++;
}

void list_for_each(List *list, ListIterator iterator)
{
    assert(iterator != NULL);

    ListNode *node = list->head;
    bool result = TRUE;
    while(node != NULL && result) {
        result = iterator(node->data);
        node = node->next;
    }
}

void list_del_node(List *list, ListNode *node)
{
    assert(list->logicalLength > 0);

    if(node == list->head && node == list->tail) {
        list->head = NULL;
        list->tail = NULL;
    } else if(node == list->head) {
        list->head = node->next;
        list->head->prev = NULL;
    } else if(node == list->tail) {
        list->tail = node->prev;
        list->tail->next = NULL;
    } else {
        node->prev->next = node->next;
        node->next->prev = node->prev;
    }
    list->logicalLength--;
    free(node->data);
    free(node);
}

void list_head(List *list, void *element, bool removeFromList)
{
    assert(list->head != NULL);

    ListNode *node = list->head;
    memcpy(element, node->data, list->elementSize);

    if(removeFromList) {
        list->head = node->next;
        list->logicalLength--;
    }
    free(node->data);
    free(node);
}
void list_tail(List *list, void *element)
{
    assert(list->tail != NULL);
    ListNode *node = list->tail;
    memcpy(element, node->data, list->elementSize);
}

int list_size(List *list)
{
    return list->logicalLength;
}

ListNode* list_random(List *list)
{
    ListNode *head = list->head;
    int r = randint(0,list->logicalLength);
    ListNode *current = head;
    int n = 0;
    while (n!=r)
    {
        current = current->next;
        n++;
    }
    return current;
}

void move(Position *pos){
    double r1, p_lin, p_tot;
    int r2;
    int lin_disp_x[4] = {0,1,0,-1};
    int lin_disp_y[4] = {1,0,-1,0};
    int diag_disp_x[4] = {1,1,-1,-1};
    int diag_disp_y[4] = {1,-1,1,-1};
    int i;
    p_tot = 4 + 4/sqrt(2);
    p_tot = 4 + 4;
    p_lin = 4/p_tot;
    r1 = randdb(0,1);
    r2 = randint(0,4);
    if (r1 < p_lin){
        pos->x += lin_disp_x[r2];
        pos->y += lin_disp_y[r2];
    } else{
        pos->x += diagDisp_x[r2];
        pos->y += diagDisp_y[r2];
    }
    boundary_condition(pos);
}

void move_healthy(){
    int i,j,k,n;
    Position pos;
    for(i = 0; i < CELLS; i++){
        for(j = 0; j < CELLS; j++){
            n = grid[i][j];
            for(k = 0; k < n; k++){
                pos.x = i;
                pos.y = j;
            }
        }
    }
}
```c
void boundary_condition(Position *pos)
{
    /* If a position is out of the grid, returns a new one
     * according to periodic boundary condition.
     */
    if (pos->x > (CELLS-1)){
        pos->x = 0;
    }
    else if (pos->x < 0){
        pos->x = CELLS - 1;
    }
    if (pos->y > (CELLS-1)){
        pos->y = 0;
    }
    else if (pos->y < 0){
        pos->y = CELLS - 1;
    }
}

void migration(List *Elist, List *Ilist, List *Tlist, List *Rlist, int year, int count)
{
    new_imm = yearly_immigrants[year] / (365 / migration_frequency);
    new_em = yearly_emigrants[year] / (365 / migration_frequency);
    double p_emS, p_emE, p_emI, p_emT, p_emR;
    if (imm_counter == migration_frequency){
        imm_counter = 0;
        // IMMIGRATION
        float icounter = 0;
        float ecounter = 0;
        float scounter = 0;
        float rcounter = 0;
        for (int i=0;i<new_imm;i++){
            float r = randdb(0,1);
            if (r<p_immS){
                int randx = randint(0,(CELLS-1));
                int randy = randint(0,(CELLS-1));
                grid[randx][randy] = grid[randx][randy] + 1;
            }
            else if(r<(p_immS+p_immE)){
            }
        }
    }
    // EMIGRATION
    float emcounter = 0;
    float emcounter = 0;
    float emcounter = 0;
    float emcounter = 0;
    for (int i=0;i<new_em;i++){
        float r = randdb(0,1);
        if (r<p_emS){
            int randx = randint(0,(CELLS-1));
            int randy = randint(0,(CELLS-1));
            grid[randx][randy] = grid[randx][randy] + 1;
        }
        else if(r<(p_emS+p_emE)){
        }
    }
```
setup_new_infected(Elist);
Enode *newE = Elist->tail->data;
newE->foreign = 1;
}
else if(r<(p_immS+p_immE+p_immI)){
    setup_new_sick(Ilist);
    Inode *newI = Ilist->tail->data;
    newI->foreign = 1;
    n_new_for_sick++;
}
else if(r<(p_immS+p_immE+p_immI+p_immT)){
    setup_new_treatment(Tlist);
    Tnode *newT = Tlist->tail->data;
    newT->foreign = 1;
}
else{
    setup_new_treated(Rlist);
    Rnode *newR = Rlist->tail->data;
    newR->foreign = 1;
}
}

//EMIGRATION
int total = count+Elist->logicalLength+Ilist->logicalLength+Tlist->logicalLength+Rlist->logicalLength;
p_emS = (double)count/total;
p_emE = (double)Elist->logicalLength/total;
p_emI = (double)Ilist->logicalLength/total;
p_emT = (double)Tlist->logicalLength/total;
p_emR = (double)Rlist->logicalLength/total;
for (int i=0;i<new_em;i++)
{
    float r = randdb(0,1);
    if (r<p_emS){
        int randx = randint(0,(CELLS-1));
        int randy = randint(0,(CELLS-1));
        grid[randx][randy] = grid[randx][randy] - 1;
        scounter++;
    }
    else if(r<(p_emS+p_emE)){
        list_del_node(Elist,list_random(Elist));
        ecounter++;
    }
    else if(r<(p_emS+p_emE+p_emI)){
        list_del_node(Ilist,list_random(Ilist));
        icounter++;
    }
    else if(r<(p_emS+p_emE+p_emI+p_emT)){
        list_del_node(Tlist,list_random(Tlist));
    }
}
icounter++;
}
else{
    list_del_node(Rlist, list_random(Rlist));
    rcounter++;
}
}
imm_counter++;
}

/***************************************************************************/
****  INFECTED DYNAMICS FUNCTIONS  ****
/***************************************************************************/

void infected_update(List *Elist, ListNode *node, List *Ilist)
{
/* Contains everything that has to be considered per infected node
   * each day.
   */

    Enode *infected = node->data;
    /* Check if it dies. If so, end function */
    if (infected_die(Elist, node)){
        return;
    }
    /* If not, move */
    move(&(infected->pos));
    /* Check if it heals. If so, end function */
    if(infected_heal(Elist, node) ) { 
        return;
    }
    /* Check if it gets sick. If so, end function */
    if(infected_sicken(Elist, node, Ilist) ) {
        return;
    }
    /* If it reaches here, advance by one the time infected */
    infected->t_infected++;
}

int infected_die(List *Elist, ListNode *node)
{
/* Check if an infected individual dies. If so, delete it, generate
   * a new healthy individual in a random position and return 1.
   * If it survives, return 0;
   */
    Enode *infected = node->data;
    double r = randdb(0,1);
    double prob = p_die;
    if(r < prob) {
        Position pos = generate_position();
        new_healthy(pos);
        list_del_node(Elist,node);
        n_dead_infected++;
        return 1;
    } else {
        return 0;
    }
}
int infected_heal(List *Elist, ListNode *node)
{
    /* Determine if an infected individual can be considered healthy
     * again. If so, generate a healthy in its position and return 1.
     * If not, return 0.
     */
    Enode *infected = node->data;
    if(infected->t_infected > t_infected_max) {
        new_healthy(infected->pos);
        list_del_node(Elist, node);
        n_heal_infected++;
        return 1;
    } else {
        return 0;
    }
}

int infected_sicken(List *Elist, ListNode *node, List *Ilist)
{
    /* Given an infected individual determine if it gets sick.
     * If it does, delete infected node and generate sick node with
     * same characteristics.
     * Also determine if it can be considered healthy.
     */
    double prob, r;
    int i;
    float f; //factor to fit the desired behaviour
    r = randdb(0,1);
    Enode *infected = node->data;
    for(i = 0; i*365 < infected->t_infected; i++){
    }
    prob = (p_sicken[i-1]/365);
    //prob = (p_sicken_all * exp(-beta*(infected->t_infected)))/365;
    f = 0.9;
    prob *= f;
    if(r < prob) {
        if (infected->foreign == 0){
            n_new_nat_sick++;
        } else{
            n_new_for_sick++;
        }
        new_sick(Ilist, infected);
        list_del_node(Elist, node);
        return 1;
    } else {
        return 0;
    }
}

void new_sick(List *Ilist, Enode *infected)
{
    /* Generate a sick node based on the characteristics
     * of a previously infected node.
     */
    Inode *new = malloc(sizeof(Inode));
    new->foreign = infected->foreign;
    new->smear = define_smear();
new->t_sick = 0;
define_diagnose(new);
new->pos = infected->pos;
list_append(Ilist, new);
}
/*++++++++++++++++++++++++++++++*/
****
SICK DYNAMICS FUNCTIONS
****
++++++++++++++++++++++++++++++*/
void sick_update(List *Ilist, ListNode *node, List *Elist, List *Tlist)
{
/* Contains everything that has to be considered per sick node * each day. */

Inode *sick = node->data;

/* Check if it dies. If so, end function */
if(sick_die(Ilist, node)) {
    return;
}

/* If not, move */
move(&sick->pos);

/* Check if it starts treatment. If so, end function */
if(start_treatment(Ilist, node, Tlist)) {
    return;
}

/* If not, advance sick days and infect others */
sick->t_sick++;
infect(sick,Elist);
}
int sick_die(List *Ilist, ListNode *node)
{
/* Check if sick individual dies. If so, delete it, generate * a new healthy individual in a random position and return 1. * If it survives, return 0; */

double r = randdb(0,1);
Enode *infected = node->data;
if(r < p_die_sick) {
    Position pos = generate_position();
    new_healthy(pos);
    list_del_node(Ilist, node);
    return 1;
} else {
    return 0;
}
}
int start_treatment(List *Ilist, ListNode *node, List *Tlist)
{
/* If the time sick of a sick individual has reached the * diagnose delay time, move the node from sick to node under * treatment. */

Inode *sick = node->data;
if(sick->t_sick == sick->diagnose) {
    new_treatment(Tlist, sick);
    list_del_node(Ilist, node);
    n_new_treatment++;
    return 1;
}
void new_treatment(List *Tlist, Inode *sick) {
/* Generate an individual under treatment from a sick one.
* The sick evolves to treatment.
*/
    Tnode *new = malloc(sizeof(Tnode));
    new->foreign = sick->foreign;
    new->smear = sick->smear;
    new->t_treatment = 0;
    new->pos = sick->pos;
    list_append(Tlist, new);
}

void infect(Inode *sick, List *Elist) {
/* Determine if there are healthy individuals susceptible
* to be infected (in contact) and infect them with a given
* probability.
*/
    Position look;
    double r, prob;
    int i, j, susceptibles;
    int square[4][2] = {{1,0},
                        {1,1},
                        {0,1},
                        {-1,1},
                        {-1,0},
                        {0,-1},
                        {(1,-1)};
    prob = (1 + sick->smear)*p_infect;
    for(i = 0; i < 9; i++) {
        look = sick->pos;
        look.x += square[i][0];
        look.y += square[i][1];
        boundary_condition(&look);
        susceptibles = grid[look.x][look.y];
        for(j = 0; j < susceptibles; j++) {
            r = randdb(0,1);
            if (r < prob) {
                new_infected(sick, Elist, look);
                grid[look.x][look.y] = 1;
                n_new_infected++;
            }
        }
    }
}

void new_infected(Inode *sick, List *Elist, Position pos) {
/* Generate a new infected from a healthy individual according to
* characteristics determined by the case that infects.
*/
    Enode *new = malloc(sizeof(Enode));
    infected_origin(new, sick);
new->t_infected = 0;
new->pos = pos;
}
list_append(Elist, new);

void infected_origin(Enode *node, Inode *infector)
{
    /* Define origin and gender of a new infected according to the
        * origin and gender of the infector.
        */
    int i = 0;
    double r, prob;
    r = randdb(0.0,1.0);
    prob = 0;
    do {
        prob += p_origin[i++][infector->foreign];
    } while(prob < r);
    switch(i-1){
    case 0:
        node->foreign = 0;
        break;
    case 1:
        node->foreign = 1;
        break;
    }
}

/***********************************************************************
**** TREATMENT DYNAMICS FUNCTIONS ****
***********************************************************************

void treatment_update(List *Tlist, ListNode *node, List *Rlist)
{
    /* Contains everything that has to be considered per node under
        * treatment each day.
        */
    Tnode *treatment = node->data;
    /* Check if it dies. If so, end function */
    if (treatment_die(Tlist, node)){
        return;
    }
    /* If not, move */
    move(&treatment->pos);
    /* Check if it finishes treatment. If so, end function. */
    if(finish_treatment(Tlist, node, Rlist)) {
        return;
    }
    treatment->t_treatment++;
}

int treatment_die(List *Tlist, ListNode *node)
{
    /* Check if an individual under treatment dies. If so, delete it, generate
        * a new healthy individual in a random position and return 1.
        * If it survives, return 0;
        */
    Tnode *treatment = node->data;
    double r = randdb(0,1);
double prob = p_die;
if (r < prob) {
    Position pos = generate_position();
    new_healthy(pos);
    list_del_node(Tlist, node);
    return 1;
} else {
    return 0;
}
}

int finish_treatment(List *Tlist, ListNode *node, List *Rlist)
{
    /* Check if a given individual under treatment finishes or abandons
     * treatment. If so, evolve it to treated.
     */
    double r, prob;
    Tnode *treatment = node->data;
    r = randdb(0,1);
    prob = p_abandon/t_treatment_max; // Daily prob to abandon.
    if (r < prob || treatment->t_treatment == t_treatment_max) {
        new_treated(Rlist, treatment);
        list_del_node(Tlist, node);
        n_new_treated++;
        return 1;
    } else {
        return 0;
    }
}

void new_treated(List *Rlist, Tnode *treatment)
{
    /* Move a node under treatment to treated. */
    Rnode *new = malloc(sizeof(Rnode));
    new->foreign = treatment->foreign;
    new->smear = treatment->smear;
    new->t_treatment = treatment->t_treatment;
    new->t_treated = 0;
    define_p_relapse(new);
    new->pos = treatment->pos;
    list_append(Rlist, new);
}

/*************************************************************/
**** TREATED DYNAMICS FUNCTIONS ****
/*************************************************************/

void treated_update(List *Rlist, ListNode *node, List *Ilist)
{
    /* Contains everything that has to be considered per node under 
     * treatment each day.
     */
    int t_left;
    double r, prob;
    Rnode *treated = node->data;
    /* Check if it dies. If so, end function */
    if (treated_die(Rlist, node)) {
        return;
    }
/* If not, move */
move(&(treated->pos));

/* Check either if it recovers, it relapses or it * continues as treated */
t_left = t_to_healthy - treated->treatment;
r = randdb(0,1);
//prob = treated->p_relapse/t_left;
prob = treated->p_relapse/714;

if(treated->t_treated == t_left) {
    new_healthy(treated->pos);
    list_del_node(Rlist, node);
    n_recovered++;
} else if(r < prob) {
    relapse(Ilist, treated);
    list_del_node(Rlist, node);
} else {
    treated->t_treated++;
}

int treated_die(List *Rlist, ListNode *node)
{
    /* Check if a treated individual dies. If so, delete it, generate * a new healthy individual in a random position and return 1. * If it survives, return 0; */
    Tnode *treated = node->data;
    double r = randdb(0,1);
    double prob = p_die;
    if(r < prob) {
        Position pos = generate_position();
        new_healthy(pos);
        list_del_node(Rlist, node);
        return 1;
    } else {
        return 0;
    }
}

void relapse(List *Ilist, Rnode *treated)
{
    /* Move a node under treatment back to sick due to * early abandon of treatment. */
    Inode *new = malloc(sizeof(Inode));
    new->foreign = treated->foreign;
    new->smear = treated->smear;
    new->t_sick = 0;
    define_diagnose(new);
    new->pos = treated->pos;
    list_append(Ilist, new);
}

void initialize_simulation(List *Elist, List *Ilist, List *Tlist, List *Rlist)
{
    /* Initialize all type of individuals. Generate all the initial individuals. * Assign each one at the corresponding list or the grid. */
    int i;
    Position pos;
setup_infected_list(Elist);
setup_sick_list(Ilist);
setup_treatment_list(Tlist);
setup_treated_list(Rlist);

for(i = 0; i < num_healthy; i++) {
    pos = generate_position();
    new_healthy(pos);
}

for(i = 0; i < num_infected; i++) {
    setup_new_infected(Elist);
}

for(i = 0; i < num_sick; i++) {
    setup_new_sick(Ilist);
}

for(i = 0; i < num_treatment; i++) {
    setup_new_treatment(Tlist);
}

for(i = 0; i < num_treated; i++) {
    setup_new_treated(Rlist);
}

Position generate_position(void)
{
    /* Generate a random position in the defined grid */
    Position new;
    new.x = randint(0, CELLS);
    new.y = randint(0, CELLS);
    return new;
}

void new_healthy(Position pos)
{
    /* Generate a healthy individual in a given position. */
    grid[pos.x][pos.y] += 1;
}

void setup_infected_list(List *Elist)
{
    /* Generate the list that will contain the infected individuals */
    list_new(Elist,sizeof(Enode),NULL);
}

void setup_new_infected(List *Elist)
{
    /* Generate an infected individual with the characteristics defined from the general population and append it to the list of infected nodes. */
    Enode *new = malloc(sizeof(Enode));
    setup_infected_origin(new);
    setup_t_infected(new);
    new->pos = generate_position();
    list_append(Elist, new);
}

void setup_infected_origin(Enode *node)
/* Define origin according to the probabilities defined */

double r = randdb(0,1);
node->foreign = 0;
if(r < p_infected_foreign) {
    node->foreign = 1;
}

void setup_t_infected(Enode *node)
{
    /* Define the time that an individual has been in infected state. Uniform between 0 and 7 years. */
    node->t_infected = randint(0,t_infected_max);
}

void setup_sick_list(List *Ilist)
{
    /* Generate the list that will contain the sickened individuals */
    list_new(Ilist,sizeof(Inode),NULL);
}

void setup_new_sick(List *Ilist)
{
    /* Generate a sick individual with the characteristics defined from the cases and append it to the list of sickened nodes. */
    Inode *new = malloc(sizeof(Inode));
    new->foreign = setup_sickened_origin();
    new->smear = define_smear();
    define_diagnose(new);
    setup_t_sick(new);
    new->pos = generate_position();
    list_append(Ilist, new);
}

char setup_sickened_origin(void)
{
    /* Define origin of sickened according to the probabilities defined */
    double r = randdb(0,1);
    if(r < p_sickened_foreign) {
        return 1;
    } else {
        return 0;
    }
}

int define_smear(void)
{
    /* Define if smear positive according to the probabilities defined */
    double r = randdb(0,1);
if(r < p_smear) {
    return 1;
} else {
    return 0;
}

void define_diagnose(Inode *node) {
    /* Define the diagnose delay for a sick individual
     * given its origin.
     */
    int mu = diagnose_mean[node-foreign];
    node->diagnose = randint_normal(mu,diagnose_std);
}

void setup_t_sick(Inode *node) {
    /* Define the time an individual have been sick.
     * Between 0 days and diagnose delay - 1.
     */
    node->t_sick = randint(0,node->diagnose-1);
}

void setup_treatment_list(List *Tlist) {
    /* Generate the list that will contain the individuals
     * under treatment.
     */
    list_new(Tlist,sizeof(Tnode),NULL);
}

void setup_new_treatment(List *Tlist) {
    /* Generate an individual under treatment with the characteristics
     * defined from the cases and append it to the
     * list of treatment nodes.
     */
    Tnode *new = malloc(sizeof(Tnode));
    new->foreign = setup_sickened_origin();
    new->smear = define_smear();
    setup_treatment_t_treatment(new);
    new->pos = generate_position();
    list_append(Tlist, new);
}

void setup_treatment_t_treatment(Tnode *node) {
    /* Define the time since starting treatment of an individual
     * under treatment.
     */
    node->t_treatment = randint(0,t_treatment_max);
}

void setup_treated_list(List *Rlist) {
    /* Generate the list that will contain the treated individuals
     */
    list_new(Rlist,sizeof(Rnode),NULL);
}

void setup_new_treated(List *Rlist) {

/* Generate a sick individual with the characteristics defined from the cases and append it to the list of sickened nodes. */

Rnode *new = malloc(sizeof(Rnode));
new->foreign = setup_sickened_origin();
new->smear = define_smear();
setup_treated_t_treatment(new);
setup_t_treated(new);
define_p_relapse(new);
new->pos = generate_position();
list_append(Rlist, new);
}

void setup_treated_t_treatment(Rnode *node) {
/* Define the time since starting treatment of a treated individual. */

double r = randdb(0,1);
if (r < p_abandon){
  node->t_treatment = randint(t_treatment_min, t_treatment_max);
} else {
  node->t_treatment = t_treatment_max;
}
}

void setup_t_treated(Rnode *node) {
/* Define the time since finishing/abandoning treatment for a treated individual. */

int t_left = t_to_healthy - node->t_treatment;
node->t_treated = randint(0, t_left);
}

void define_p_relapse(Rnode *node) {
/* Define the probability to relapse of a treated individual.
* If t_treated < min, will become sick again.
* p_relapse straight line from 100% at 15 days to 1% at 180 days of treatment.
*/

int num = t_treatment_max - node->t_treatment;
int den = t_treatment_max - t_treatment_min;
node->p_relapse = ((double) num/den)*(1 - p_relapse_min) + p_relapse_min;
}

double randdb(double min, double max) {
/* Generates a double pseudorandom number in [min,max).
* Uniform distribution.
*/

if(max < min) {
  printf("Error in random db: min > max (min: %f, max: %f) \n",min,max);
  exit(1);
}

int r = rand();
return min + (max - min) * (r / (double) RAND_MAX);
}

int randint(int min, int max) {
/* Generates an integer pseudorandom number in [min,max]. */

...
* Uniform distribution.
*/

    int result = 0;

    if(max < min) {
        printf("Error in random int: min > max (min: %d, max: %d) \n",min,max);
        exit(1);
    }

    result = (rand() % (max - min)) + min;
    return result;
}

int randint_normal(int mu, int sig)
{
    /* Generate an integer pseudorandom number with normal distribution.
     * Mean: mu; Std: sig.
     */
    double z1, z2, z, pi;
    z1 = randdb(0,1);
    z2 = randdb(0,1);
    pi = acos(-1.0);
    z = sqrt(-2.0 * log(z1)) * cos(2*pi*z2);
    return z*sig + mu;
}
B. LHS Sampling Code

```c
void shuffle(int *array, size_t n)
{
    if (n > 1)
    {
        size_t i;
        for (i = 0; i < n - 1; i++)
        {
            size_t j = i + rand() / (RAND_MAX / (n - i) + 1);
            int t = array[j];
            array[j] = array[i];
            array[i] = t;
        }
    }
}

double r2()
{
    return (double)rand() / (double)RAND_MAX;
}

int main()
{
    srand(time(NULL));
    int i;
    FILE *f;
    f = fopen("./data/sample.txt", "w");
    int Nsamples=25;
    double a1min=0.01, a1max=0.1;
    double a2min=0.05, a2max=0.14;
    double a3min=0.01, a3max=0.1;
    double a4min=0.05, a4max=0.3;
    double A1[NSamples+1], A2[NSamples+1], A3[NSamples+1], A4[NSamples+1];
    double S[NSamples][NSamples+1];
    int c1[NSamples], c2[NSamples], c3[NSamples], c4[NSamples];
    double Ax_a1, Ax_a2, Ax_a3, Ax_a4;
    Ax_a1=(a1max-a1min)/NSamples;
    Ax_a2=(a2max-a2min)/NSamples;
    Ax_a3=(a3max-a3min)/NSamples;
    Ax_a4=(a4max-a4min)/NSamples;
    for(i=0;i<NSamples;i++){
        c1[i]=i;
        c2[i]=i;
        c3[i]=i;
        c4[i]=i;
    }
    for(i=0;i<NSamples+1;i++){
        A1[i]=a1min+*Ax_a1;
        A2[i]=a2min+*Ax_a2;
        A3[i]=a3min+*Ax_a3;
        A4[i]=a4min+*Ax_a4;
    }
}
```
shuffle(c1,Nsamples);
shuffle(c2,Nsamples);
shuffle(c3,Nsamples);
shuffle(c4,Nsamples);

for(i=0;i<Nsamples;i++)
{
    S[i][0]=A1[c1[i]]+(A1[c1[i]+1]-A1[c1[i]])*r2();
    S[i][1]=A2[c2[i]]+(A2[c2[i]+1]-A2[c2[i]])*r2();
    S[i][2]=A3[c3[i]]+(A3[c3[i]+1]-A3[c3[i]])*r2();
    S[i][3]=A4[c4[i]]+(A4[c4[i]+1]-A4[c4[i]])*r2();
}

fprintf(f,"%d\n",Nsamples);

for(i=0;i<Nsamples;i++)
{
    fprintf(f,"%f %f %f %f\n",S[i][0],S[i][1],S[i][2],S[i][3]);
}
fclose(f);
C. Random forest algorithm

```python
import numpy as np
import pandas as pd
import pickle
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import learning_curve
from sklearn.externals import joblib
from scipy.optimize import minimize
import matplotlib.pyplot as plt

# Data generated by the IBM simulator
data = pd.read_csv('results_mig_bln4.csv')

# Real epidemiological data observed.
truth = pd.read_csv('truth_bln.csv')

def encode_origin(df):
    """Encode origin variable.

    Machine learning models need the input to be numeric. Since the variable
    'origin' is categorical, we need to convert it to numeric. One way to do
    it is to encode its categories, so that:

    origin : {'nat', 'imm'} -> nat : {0, 1}
              imm : {0, 1}

    The 'origin' variable becomes two binary variables, 'nat' and 'imm'.
    """
    df = df.copy()
    df['nat'] = 1*(df['origin'] == 'nat')
    df['imm'] = 1*(df['origin'] == 'imm')
    df.drop(columns=['origin'], inplace=True)
    return df

# Encode origin variable for both the IBM data and the real data.
data = encode_origin(data)
truth = encode_origin(truth)
```
# Split variables between predictors 'X' and response 'y'.
X = data.loc[:, data.columns != 'new_sick']
y = data.loc[:, 'new_sick']

# Define machine learning estimator.
estimator = RandomForestRegressor(random_state=42)

# Define hyper-parameters to try.
param_grid = [
    {
        'n_estimators': [10, 50, 100],
        'max_features': [None, 'sqrt', 'log2'],
        'max_depth': [None, 2, 5]
    }
]

# Define how to split data during cross-validation.
cv = KFold(n_splits=5, shuffle=True, random_state=42)

# Define model to fit.
model = GridSearchCV(
    estimator=estimator,
    param_grid=param_grid,
    scoring='r2',
    cv=cv,
    iid=False,
    return_train_score=True
)

# Fit the model.
model.fit(X, y)

# Report results.
print("Best hyper-parameter set: \n{}\n".format(model.best_params_))
print("Mean cross-validation test score: {}\n".format(model.best_score_))

# Keep only the model with the best hyper-parameter set.
best = model.best_estimator_
# Plot and save learning curves to assess if we need more data.

def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None, 
n_jobs=None, train_sizes=np.linspace(.1, 1.0, 5)):
    
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid(alpha=0.3)
    plt.fill_between(train_sizes, train_scores_mean - train_scores_std, 
        train_scores_mean + train_scores_std, alpha=0.1,
        color="C0")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std, 
        test_scores_mean + test_scores_std, alpha=0.1, color="C1")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="C0",
        label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="C1",
        label="Cross-validation score")
    plt.legend(loc="best")
    return plt

plot_learning_curve(best, "Learning curve", X, y, cv=cv)
plt.savefig('learning_curve.png', dpi=300)

# Save the fitted model.
joblib.dump(best, 'model.pkl')

# To load in another file:
best = joblib.load('model.pkl')

# To make predictions (i.e. function evaluations) input should be as on data
# num_infected, p_infect, p_sicken_all, f_sicken, year, nat, imm.
# Example:

years = 10

c1 = 1

c2 = 0

num_infected = np.arange(0.01,0.2,0.01)
p_infect = np.arange(0.05,0.28,0.01)
p_sicken_all = np.arange(0.01,0.2,0.01)
f_sicken = np.arange(0.05,0.6,0.027)

sqerror = np.zeros(years)
totsqerror = np.zeros(len(num_infected)**4)
param_comb = np.zeros([len(num_infected)**4,4])

new_sick = truth['new_sick']

for l1 in range(0,len(num_infected)):
    for l2 in range(0,len(num_infected)):
        for l3 in range(0,len(num_infected)):
            for l4 in range(0,len(num_infected)):
                for i in range (1,years+1):
                    y_nat_pred =
                    best.predict([[num_infected[l1],p_infect[l2],p_sicken_all[l3],f_sicken[l4],i,1,0]])
                    y_imm_pred =
                    best.predict([[num_infected[l1],p_infect[l2],p_sicken_all[l3],f_sicken[l4],i,0,1]])

                    sqerror[i-1] = [(y_nat_pred - new_sick[i])**2]+[(y_nat_pred - new_sick[c1])**2]

                    print("Year " + str(i))
                    print("Nat Prediction: ",format(y_nat_pred))
                    print("Imm Prediction: ",format(y_imm_pred))
                    c1 = c1+2

                    totsqerror[c2] = np.sum(sqerror)
param_comb[c2,0] = num_infected[l1]
param_comb[c2,1] = p_infect[l2]
param_comb[c2,2] = p_sicken_all[l3]
param_comb[c2,3] = f_sicken[l4]
sqerror = np.zeros(years)
c1 = 1
c2 = c2 + 1

print np.argmin(totsqerror)
print min(totsqerror)

np.savetxt('error.csv',totsqerror)
np.savetxt('params.csv',param_comb)