POLITECNICO DI TORINO

Master Degree course in Mathematical Engineering

Master's Degree Thesis

Modeling the new flu wave using data science and complex networks theory.

Supervisors
Prof. Lorenzo Zino
Prof. Alessandro Rizzo

Candidate
Francesco Celino

Academic Year 2024-2025

Acknowledgements

Ringraziare chi ti ha supportato lungo il tuo cammino è sempre difficile, ma forse è ancora più difficile decidere se fare i ringraziamenti in italiano oppure no in una tesi di laurea che è completamente in inglese.

Battute stupide a parte ho optato per la scelta più intima, e vorrei dunque rivolgermi brevemente alla mia famiglia, che è riuscita a non far arrendere l'uomo più arrendevole esistente (solo dal punto di vista universitario, si precisa), ai miei amici storici, che ho purtroppo trascurato a causa degli studi (avremo modo di rifarci), ai nuovi amici di Torino, che mi hanno insegnato che esistono anche ingegneri simpatici, ai miei relatori Lorenzo, Alessandro e ad Elisa, che mi hanno introdotto al mondo della scienza dei dati e mi hanno fatto partecipare a un progetto di tesi più che stimolante: siete molto in gamba, e adesso che il percorso è finito posso dirvelo "a tu per tu" senza rischiare di risultare di parte.

Il ringraziamento finale va ad Alice, che ha avuto la pazienza di sopportare il sottoscritto e i suoi deliri paranoici mentre stava preparando l'ultimo esame, sei la persona più premurosa che abbia mai incontrato.

Abstract

This Master's thesis explores how the SEINR (Susceptible, Exposed, Infectious, Non-infectious, Recovered) compartmental model can be used to forecast the evolution of influenza-like illness (ILI) cases in Italy during the 2023-24 winter season. The model's innovative aspects lie in its community-based meta-population framework, which simulates *intra-* and *inter*-regional mobility, capturing network dynamics critical to understanding how a disease spreads in Italy's diverse demographic and geographic landscape. This approach, which was previously successful when evaluating the efficacy of NPIs (non-pharmaceutical interventions) during the *COVID-19* pandemic, is then further refined by taking into account features such as class divisions according to age, activity levels, and vulnerability to disease of different age groups, increasing the adaptability of the model to the Italian landscape.

Bayesian inference tools and Monte Carlo methods are then used to improve the estimates of a few fundamental epidemiological parameters such as transmission rate and infection duration. Our empirical results demonstrate the model's effectiveness in capturing flu trends in Italy.

This work emphasizes the role of adaptive modeling in epidemiology, and how public health strategies driven by past data can help in managing seasonal epidemics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Each year, Influenza affects millions of people across the world, posing great challenges for healthcare systems in various developed countries, with great risks for elderly and more vulnerable people. In response, researchers and public health officials work to predict and control the spread of these diseases, using various tools and methods. One of the most powerful tools we have available is mathematical modeling, which allows us to simulate how a disease might spread through a population through the use of "differential equations". This thesis focuses on adopting one of such models [5] for flu-like illnesses in Italy, using a framework called the SEINR model.

The SEINR model is a type of "compartment" model. This means that it divides the population into different groups (or compartments) of people, based on the stages of the disease and whether or not they are infected. In this case, the compartments are *Susceptible* (people who can catch the flu), *Exposed* (people who have been infected but are not yet contagious), *Infectious* (people who can spread the flu), *Non-infectious* (people who do not present any symptom, or are isolated), and *Removed* (people who are either immune to the flu after recovering, or dead due to complications).

The history of compartmental models dates back to 1927, when Kermack and McKendrick [8] defined the SIR model for the first time, in its simplest form.

However, what makes the model we used for this thesis particularly innovative is its "meta-population" structure, already used with great success in 2021 when trying to model the first Covid-19 wave in Italy [5]. Instead of treating Italy as one large group of people, the meta-population approach breaks it down into regions and even considers how people move between these regions. This is crucial for a country like Italy, where people frequently travel between cities and regions for work, school, and other reasons.

Most importantly, when trying to predict the outcomes of a Influenza season, we need to consider other crucial factors, such as vaccines, the age and activity levels of individuals, as well as their vulnerability to illness. These factors are important because some groups, like the elderly or those with pre-existing health conditions, are more at risk during a flu outbreak, but could very well be less exposed to the disease due to having less social contacts. If we want our predictions to be accurate, we need to take into account all of those factors.

Another key feature of this thesis is the use of "Bayesian inference", a statistical

method that allows the model to improve its accuracy (and precision) over time. As new weekly data about flu cases becomes available, the model updates its parameters in a probabilistic fashion: that means, if in one particular week we observe more infected people than we would expect, we probably need to rethink our predictions. That's why, when using Bayesian inference, we do not provide exact estimates for the number of infected people we will have next week, but rather a "probability distribution" that includes reasonable confidence intervals: as more weeks pass and we gather more real data, we expect those probability distributions to get narrower, reflecting our increased confidence on the disease's characteristics.

In short, the goal of my thesis is to provide a more accurate and flexible tool for predicting flu-like illnesses in Italy, one that can adapt to different variants of Influenza each year (as you probably know, each year the flu outbreak is slightly different due to changing vaccines, mobility patterns, timing and public awareness). This tool can help public health officials better prepare for and respond to outbreaks, reducing the strain on healthcare systems and, who knows, maybe also save some lives.

I am also proud to say that, thanks to the invaluable guidance of my supervisors, we were able to use this model to contribute to two major forecasting projects at the European level. Through these projects, I had the opportunity to collaborate with the *ISI Foundation* and the *Istituto Superiore di Sanita'*, as we provided weekly estimates for the evolution of last year's flu season.

[5]

Chapter 2

Methods

The framework which will be used in this project is the one outlined by compartmental models, which aim to divide a population into distinct groups, or compartments, based on their status in relation to the disease. These models provide a simplified yet powerful way to describe the progression of an epidemic through a population using systems of ordinary differential equations (ODEs), in which each variable represents a compartment, and each parameter is modelled according to biological inferences.

The earliest modern compartmental model is the SIR model, introduced by Kermack and McKendrick in 1927 [8]. As implied by the name, the SIR model divides the population into three compartments: *Susceptible* (S), representing individuals who can contract the disease; *Infectious* (I), representing those actively spreading the disease; and *Removed* (R), which includes individuals who have recovered and gained immunity or have died. The dynamics of the SIR model are governed by the following system of ODEs:

$$\begin{aligned} \frac{dS}{dt} &= -\beta \frac{SI}{N}, \\ \frac{dI}{dt} &= \beta \frac{SI}{N} - \gamma I, \\ \frac{dR}{dt} &= \gamma I, \end{aligned} \tag{2.1}$$

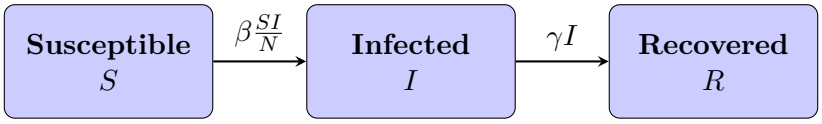


Figure 2.1: Flowchart of the SIR Compartmental Model.

where N is the total population size, β is the transmission rate, and γ is the recovery rate. It is apparent that one of the key hypotheses of the SIR model is that a person cannot become infectious twice: once that person has recovered, they cannot become susceptible again. The model also assumes that our total population remains constant over time: the total sum of $S + I + R$ cannot change over time, and this fact is easily verifiable by integrating these equations with respect to time.

This model's simplicity, while being its greatest strength, is also a major weakness: the model does not contemplate an "asymptomatic period" of sorts, since as soon as you are taken out of the susceptible compartment you can already spread the disease to other people. This behaviour does not describe real-life epidemic phenomena correctly, but makes the model simpler conceptually and easier-to-use.

For my thesis, a more complex model was needed: building upon the SIR framework, the **SEINR model** introduces additional compartments to better capture the complexities of real-world epidemics. Specifically, the SEINR model includes:

- **Susceptible (S):** Individuals who can contract the disease.
- **Exposed (E):** Individuals who have been infected but are not yet infectious, representing the incubation period.
- **Infectious (I):** Individuals who can transmit the disease.
- **Non-infectious (N):** Individuals who no longer spread the disease, either because they are isolated, mildly ill, or simply no longer symptomatic.
- **Removed (R):** Individuals who have either recovered and gained immunity or succumbed to the disease.

The inclusion of the **Exposed** and **Non-infectious** compartments allows the SEINR model to more accurately reflect diseases with an incubation period or asymptomatic cases, which are common in influenza-like illnesses. The same hypotheses we made with the SIR model about the conservation of the population (no births or deaths unrelated to the disease) and the impossibility of becoming ill twice apply here.

An important concept that arises in compartmental models with a **Removed** compartment is **herd immunity**. Herd immunity occurs when a sufficient portion of the population becomes immune, either through infection or vaccination, thereby reducing the probability of disease transmission to susceptible individuals. This phenomenon is closely related to the **basic reproduction number** (R_0), a constant that represents the average number of secondary infections generated by a single infectious individual in a fully susceptible population. In other words, if a disease has a basic reproduction number greater than one, that disease should theoretically spread indefinitely, since each person infects more than one person before recovering (on average).

For simplicity's sake, let us consider the SIR model, where herd immunity is achieved when the fraction of the population that remains susceptible falls below a critical threshold, given by:

$$S_c = \frac{1}{R_0}. \tag{2.2}$$

At this point, the effective reproduction number ($R_t = R_0 \cdot \frac{S}{N}$) drops below 1, causing the epidemic to decline. This principle also holds true for more complex models like SEINR, where, as explained, the dynamics of immunity and disease spread are also influenced by latency periods and non-infectious stages. This behaviour is harder to describe in analitical terms, but will be numerically visible in the following experiments.

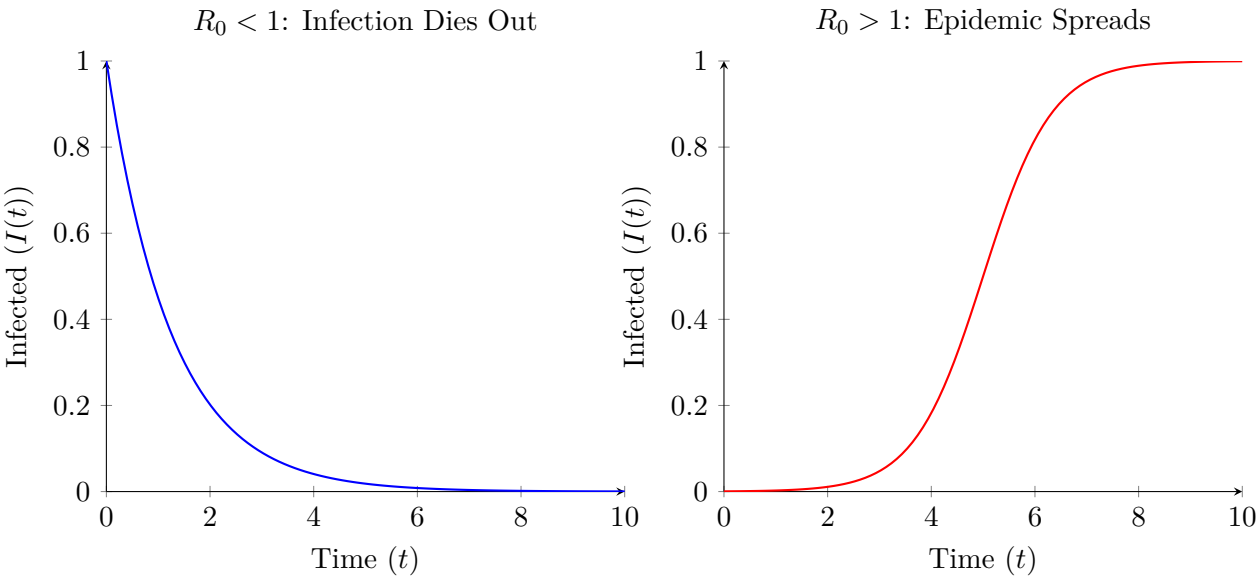


Figure 2.2: Comparison of infection dynamics based on R_0. Left: If $R_0 < 1$, infection decreases and dies out. Right: If $R_0 > 1$, infection spreads and leads to an epidemic.

In the context of seasonal influenza, if we make the assumption that a person can only become infected once each year, the concept of herd immunity plays a significant role in shaping public health strategies, particularly in designing vaccination campaigns (many compartmental models automatically include vaccinated people in the removed compartment). By estimating R_0 and tracking its progression as the epidemic goes on, one can gain insight into how contagious a certain virus or bacteria strain is in a certain place at a certain time, and thus whether implementing non-pharmaceutical interventions (NPIs), such as social distancing or lockdowns, is warranted.

2.1 The Metapopulation SEINR Framework

2.1.1 Activity-Driven Meta-Population Model

In the *Metapopulation* framework, we partition a population of n individuals into K communities, denoted as $\mathcal{H} = \{1, \dots, K\}$: each community represents bounded and well-defined geographical areas (e.g., regions, provinces, or cities). Each community $h \in \mathcal{H}$ contains n_h people. The way a community interacts with other communities in the model is described by a *weighted graph*, where each edge represents a travel path. The weights of this graph are described using the *routing matrix* $\mathbf{W} \in [0,1]^{K \times K}$, which defines the fraction of individuals in a certain community that move to other communities when becoming "active". For example, $\mathbf{W}_{hk}$ denotes the fraction of individuals from community h that travel to community k in a given time unit (as we will see, a time unit represents

a day in our model). The matrix satisfies:

$$\mathbf{W}_{hh} = 0, \quad \sum_{k=1}^{K} \mathbf{W}_{hk} = 1, \quad \forall h.$$

The inhabitants of each community are then divided into P "activity classes", $a_1, a_2, \ldots, a_P$, where $0 < a_i \leq 1$. The baseline activity a_i quantifies the propensity of individuals in class i to interact with other people. At each time step, a fraction a_i of individuals in each class becomes active, either interacting "locally" or traveling to other communities as determined by the *mobility parameter* $b \in [0,1]$. This parameter simply represents the fraction of active individuals commuting to other communities (thus coming into contact with people from other communities), with the remaining $1 - b$ interacting within their own community.

2.1.2 Disease Progression

As explained, the dynamics of the disease will be modeled using the susceptible–exposed–infectious–non-infectious–removed (SEINR) framework. After contagion, susceptible individuals move into the *Exposed* (E) compartment with rate λ, representing the latency period before becoming infectious. The transitions between compartments are defined as follows:

- $E \to I$: Transition to the *Infectious* (I) compartment occurs at rate ν, with $1/\nu$ representing the average latency period.
- $I \to N$: Transition to the *Non-infectious* (N) compartment occurs at rate μ, where $1/\mu$ is the average infectious period.
- $N \to R$: Transition to the *Removed* (R) compartment occurs at rate γ, while $1/\gamma$ represents the average delay before recovery or death.

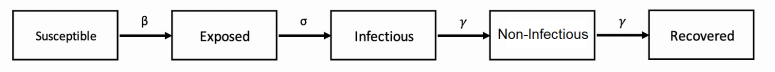


Figure 2.3: A flow chart explaining how the SEINR compartmental model is organized

It is clear that the average time from infectiousness to removal is $1/\mu + 1/\gamma$. The transition between compartments is regulated by the following set of ODEs:

$$\begin{aligned}
S_i^h(t+1) &= \big(1 - \Pi_i^h(t)\big) S_i^h(t), \\
E_i^h(t+1) &= \Pi_i^h(t) S_i^h(t) + \big(1 - \nu\big) E_i^h(t), \\
I_i^h(t+1) &= \nu E_i^h(t) + \big(1 - \mu\big) I_i^h(t), \\
N_i^h(t+1) &= \mu I_i^h(t) + (1 - \gamma) N_i^h(t).
\end{aligned}$$

Where $\Pi_i^h(t)$ represents the contagion probability, which will be defined in the following section.

2.1.3 Contagion Mechanism and Contact Probabilities

If an infectious person comes into contact with a susceptible person, the latter will not always become infected. This uncertainty is captured by the probability of contagion, $\Pi_i^h(t)$, represents the fraction of susceptible individuals in activity class i in community h that transition to the E compartment due to an interaction with infectious individuals. Since we are dealing with large populations, we can assume a thermodynamic limit ($n \to \infty$) and low epidemic prevalence (as we will see, this is in line with real-life data), and $\Pi_i^h(t)$ can be expressed as:

$$\Pi_i^h(t) = m\alpha a_i(1-\beta b)\lambda P_h + m(1-\alpha\beta a_i b)\lambda Q_h + m\alpha\beta a_i b \sum_{k\in\mathcal{H}} \mathbf{W}_{hk}\lambda P_k + m\alpha\beta a_i b \sum_{k\in\mathcal{H}} \mathbf{W}_{hk}\lambda Q_k,$$

where:

$$P_h = \frac{1}{\tilde{n}_h}\left(\sum_{j=1}^{P}(1-\alpha\beta a_j b)I_j^h + \sum_{k\in\mathcal{H}} \mathbf{W}_{hk}\sum_{j=1}^{P}\alpha\beta a_j b I_j^k\right),$$

$$Q_h = \frac{1}{\tilde{n}_h}\left(\sum_{j=1}^{P}(1-\beta b)\alpha a_j I_j^k + \sum_{k\in\mathcal{H}} \mathbf{W}_{hk}\sum_{j=1}^{P}\alpha\beta a_j b I_j^k\right),$$

and $\tilde{n}_h$, is the effective population size in community h. This is defined as

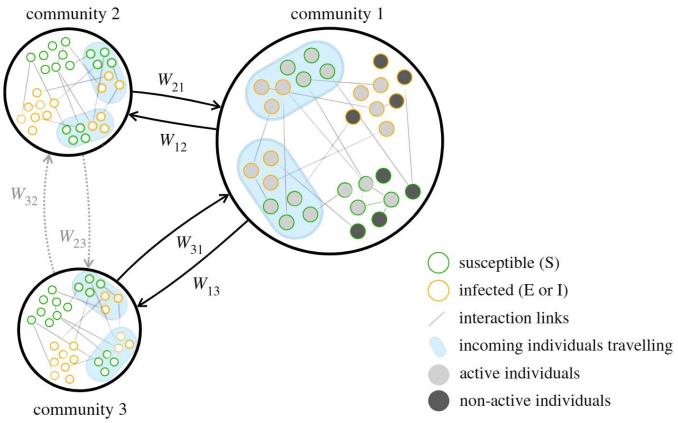


Figure 2.4: Interactions between communities in the Metapopulation framework. Taken from [5]

Thus, we now have a SEINR framework that incorporates interactions between various communities.

2.1.4 Age classes

Some age brackets, such as the elderly, tend to have fewer social interactions compared to younger people. However, they also have a significantly higher probability of developing severe complications in case of infection.

To account for this, the population was partitioned in 2 age classes, the former of which contains every individual who is less than 65 years old, and the latter includes everyone else. The fraction of individuals in each age classes was calibrated using italian census data [12].

2.1.5 Commuting matrix

The commuting matrix is estimated using official census data [12]. Each entry W_{hk} of the matrix denotes the proportion of individuals from region h that commute to region k. The structure of the commuting matrix significantly influences disease spread since regions with high incoming mobility may experience faster outbreaks due to external infections.

Figure 2.5 illustrates the structure of the commuting matrix used in our model.

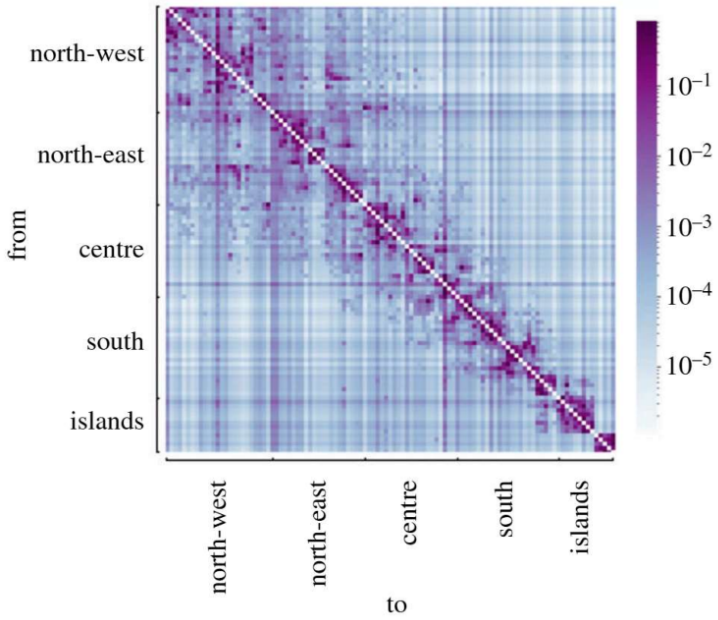


Figure 2.5: Representation of the commuting matrix W showing mobility patterns between different regions. Taken from [5]

2.1.6 Vaccines

In Italy, about two-thirds of the elderly population becomes vaccinated against Influenza each year, with the vaccination campaign typically occurring between October and December [1].

To integrate the concept of vaccination in our model, we created a new class for *vaccinated elderly* people (with its own activity level $a_e lderly$), in addition to the existing classes for the general elderly and young populations. The transition from the *elderly*

class to the *vaccinated elderly* class occurs instantaneously once a predefined threshold date is reached, as an approximation of the timing of the real-world vaccination campaign.

Vaccine efficacy values are taken from [3] and are applied to reduce the probability of transition from the *susceptible* to the *exposed* compartment. This adjustment reflects the protective effect of vaccination against infection. In addition to that, vaccinated individuals retain a lower probability of developing severe complications, in line with empirical data on vaccine effectiveness.

2.2 Other parameters

Meaning	Value(s)	Reference
$1/\nu$	Latency period	
$1/\mu$	Infectiousness period	
$1/\gamma$	Time from infectiousness to reported death	
λ	Per-contact infection probability	$\surd$
η	Class distribution	
a	Baseline activity	
b	Mobility parameter	
α_{low}	Activity reduction	$\surd$
m	Average number of contacts	
β_{low}	Mobility reduction	$\surd$

Table 2.1: Summary of additional model parameters.

2.3 Real-world Data

In order to validate our model, it is necessary to rely on real-world epidemiological data: in our case, we decided to utilize the weekly incidence of influenza-like illnesses (ILI) at the regional level. These data are sourced from the official *Influcast* project repository on GitHub [9], which provides up-to-date weekly reports on influenza incidence across Italian regions. These data are publicly available, and the information provided has its roots in a network of Italian doctors that decided to contribute to the *Influcast* project by sending data about how many of their patients show signs of flu illness.

2.3.1 Data Format

The data are provided as CSV files, where each row represents the recorded incidence for a specific week in a given region. Each file (for example *marche-2023-52-ILI.csv* has the following format):

- **Year** (`anno`): The calendar year of the observation.

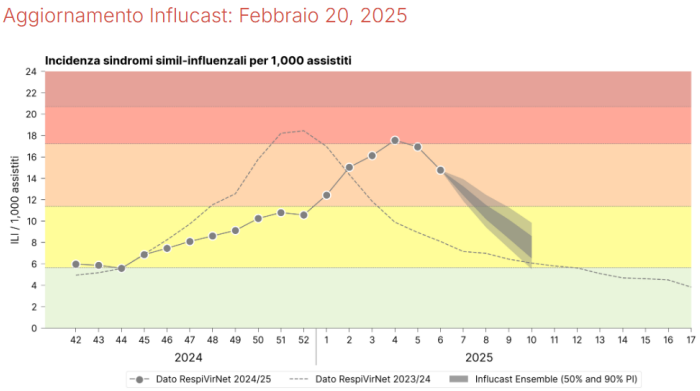


Figure 2.6: Incidence as a function of week number for various years. Taken from [11]

- **Week** (`settimana`): The calendar week number.
- **Number of Cases** (`numero_casi`): The number of reported ILI cases in the region by the surveillance system.
- **Number of Assisted Individuals** (`numero_assistiti`): The total number of patients monitored by the surveillance system.
- **Incidence** (`incidenza`): The estimated incidence per 1000 inhabitants (in our case, this is simply the number of cases divided by the total number of monitored people times 1000).
- **Target** (`target`): The type of disease being monitored (ILI, in this case).

Table 2.2 shows an example of this structure:

Year	Week	Cases	Assisted	Incidence	Target
2023	45	137.0	24447.0	5.6	ILI
2023	46	197.0	29651.0	6.64	ILI
2023	47	216.0	33744.0	6.4	ILI
2023	48	335.0	36037.0	9.3	ILI
2023	49	349.0	30838.0	11.32	ILI
2023	50	524.0	29366.0	17.84	ILI

Table 2.2: Example of CSV file structure for weekly influenza incidence data.

The reason why we decided to rely on real-world data for the calibration of our model is two-fold:

- **Initial Conditions**: Each system of ODEs requires a few initial conditions in order to be simulated. The best thing to do in this case is to provide an imput that makes sense, in order to get an output that can also be applied to the real world.

- **Model Calibration and Training**: during the training phase of our model, we can penalize "bad predictions" and assign good scores to "good predictions" by comparing the output of our model with what we can see in the real world, and adjust the parameters of our model accordingly.

2.3.2 Data Manipulation

An important challenge in integrating real-world data into epidemiological models is interpretation, and the difference in spatial resolution: while the incidence data are available at the **regional level**, our model operates on a **provincial scale**. To bridge this gap, we distribute the reported regional incidence among the provinces proportionally to their respective populations.

Mathematically, given a region r with a total population P_r and an incidence rate I_r, the estimated number of infected individuals in province p (within region r) is computed as:

$$I_p = I_r \cdot \frac{P_p}{P_r},$$

where P_p is the population of province p.

While this proportional allocation is a reasonable first approximation, it introduces a couple of limitations:

- The incidence rate does not necessarily scale linearly with population. Larger cities may experience higher transmission rates due to higher population density and mobility.

- Urban centers may tend to act as hubs for disease spread, meaning that real incidence values may be skewed compared to our proportional model.

Thus, the true relationship between incidence and population might be better captured by a nonlinear function (e.g., polynomial or exponential), an approach that our current model does not consider.

Despite these limitations, our approach ensures that the initial conditions used by the model are at least somewhat demographically consistent with observed epidemiological data. Future improvements of the model may incorporate mobility data or historical patterns of disease spread to refine the distribution of incidence values from the regional level to the provincial level.

2.3.3 Model Calibration

We estimated the initial values of a few epidemiological parameters using virological studies on influenza [2, 4]. These estimates gave us a biologically plausible range for each parameter. We then refined these parameters through a fine-tuning process, utilizing historical data on flu outbreaks of past years.

Initial Estimation from Virological Studies

We initially estimated the order of magnitude of the following parameters based on virological literature:

- μ: The rate at which individuals transition from the *Exposed* (E) compartment to the *Infectious* (I) compartment (i.e., the inverse of the latency period).
- β: The rate at which individuals transition from the *Infectious* (I) compartment to the *Non-infectious* (N) compartment (i.e., the inverse of the infectious period).
- λ: The rate of transmission, governing the transition from *Susceptible* (S) to *Exposed* (E).
- γ: The rate at which individuals transition from the *Non-infectious* (N) to the *Removed* (R) compartment (i.e., the inverse of the recovery/removal period).

Table 2.3 provides a few examples of typical values for these parameters based on the current literature.

Parameter	**Estimated Range (Influenza)**
$1/\mu$ (Latency Period)	1.5-2 days
$1/\beta$ (Infectious Period)	3-5 days
$1/\gamma$ (Recovery/Removal Time)	5-7 days
λ (Per-Contact rate of transmission)	Estimated numerically

Table 2.3: Estimated parameter ranges for influenza based on virological studies [2, 4].

Fine-Tuning with Historical Data

After defining reasonable initial estimates, we refined these parameters using historical influenza incidence data from past seasons. The fine-tuning process involved adjusting μ, β, and λ to minimize the discrepancy between the simulated epidemic curves and observed influenza incidence trends.

The optimization was performed iteratively by running the model with different parameter sets and comparing the output to real-world data. The primary metric used for assessing the quality of fit was the **Mean Absolute Error (MAE)** between the simulated and observed incidence data. Since influenza dynamics vary from year to year, calibration was performed separately for each season to account for changes in viral transmissibility, population immunity, and public health interventions.

By combining virological knowledge with empirical calibration, our model ensures both biological plausibility and high predictive accuracy when applied to real-world influenza outbreaks.

2.4 A different approach: Bayesian estimates

As an additional feature of the model, we tried implementing dynamic interval ranges and probability distributions for a few crucial epidemiological parameters: these distributions will be updated on a weekly basis as new data comes out, using the framework provided by bayesian statistics. This allows our model to adapt more easily to ever-so-slight yearly variations in Influenza infectivity, instead of using fixed parameters distributions (which are probably good enough to give credible results, but do not really fit well training data)

2.4.1 Theory of Bayesian Estimates

The Bayesian framework is founded on Bayes' theorem, which for a vector of parameters θ (e.g., the transmission rate β, the latency rate ν, etc.) and the observed data D (such as weekly influenza incidence) can be written as:

$$p(\theta|D) = \frac{L(D|\theta)\, p(\theta)}{p(D)},$$

where:

- $p(\theta)$ represents the **prior distribution**, capturing our initial beliefs about the parameters (whether those beliefs are informed by virological studies or real world data about flu outbreaks);
- $L(D|\theta)$ is the **likelihood function**, which quantifies the probability of observing some kind of data D given the parameters θ;
- $p(\theta|D)$ denotes the **posterior distribution**, that is, our updated belief about how the parameters are distributed after having observed the data;
- lastly, $p(D)$ is a normalizing constant ensuring that the posterior integrates to 1.

On a more intuitive level, given an initial distribution for our parameters, Bayes' Theorem allows us to ask ourselves how we can update this distribution in order to fit the data we have: the new distribution will of course be proportional to our prior distribution, but also to how likely it is to observe the data we have if we assume that prior distribution to be truthful.

In contrast with the frequentist approach, which provides punctual estimates and confidence intervals based solely on the observed data, the Bayesian method gives a full probability distribution over the parameters. This feature will be of great advantage in our model, allowing greater adaptability.

2.4.2 Numerical approximations for Bayesian methods

Due to the complex nature of the likelihood function in the SEINR model, which results from integrating deterministic differential equations with stochastic observation processes, an analytical solution for the posterior distribution is not attainable. Instead, we employ numerical methods, specifically Markov Chain Monte Carlo (MCMC) techniques, to approximate the posterior. The implementation proceeds as follows:

1. The parameter vector is initialized with estimates derived from existing virological literature and preliminary model calibration.

2. At each iteration of the MCMC algorithm:

   - A candidate set of parameters is generated using a carefully chosen proposal distribution.
   - The candidate is then evaluated against the current parameter set by computing an acceptance probability, which depends on the ratio of their respective posterior probabilities.

3. The acceptance criterion follows the Metropolis-Hastings algorithm:

$$\alpha = \min\left(1, \frac{P(\theta^*|D)}{P(\theta|D)}\right), \tag{2.3}$$

   where:

   - θ^* is the candidate parameter set,
   - θ is the current parameter set,
   - $P(\theta|D)$ is the posterior probability given the observed data D.

   If θ^* yields a higher posterior probability, it is accepted; otherwise, it is accepted with probability α. This ensures that, over many iterations, the chain of sampled parameters converges to the true posterior distribution.

4. Once convergence is achieved, it is assessed using diagnostic measures such as the Gelman-Rubin statistic:

$$\hat{R} = \frac{\mathrm{Var}^+(\theta)}{W}, \tag{2.4}$$

   where $\mathrm{Var}^+(\theta)$ is the pooled variance estimate and W is the within-chain variance. In our case a value of $\hat{R} \approx 1$ indicates convergence.

5. Finally, the resulting posterior sample is used to approximate a probability distribution function, and some of its features such as average, median value and credibility intervals.

6. The approximate posterior distribution is then used in our model to make the predictions we need. In the following cycle, the posterior distribution becomes the new prior, and the numerical algorithm is triggered once again.

While this Bayesian approach is computationally demanding (because we need to run an enormous amount of simulations in order to approximate a continuous probability distribution), its ability to explicitly account for uncertainty and dynamically update forecasts makes it a powerful tool, particularly during periods of rapid epidemiological change (as we will see, each year has different peaks and valleys in flu incidence during the winter season).

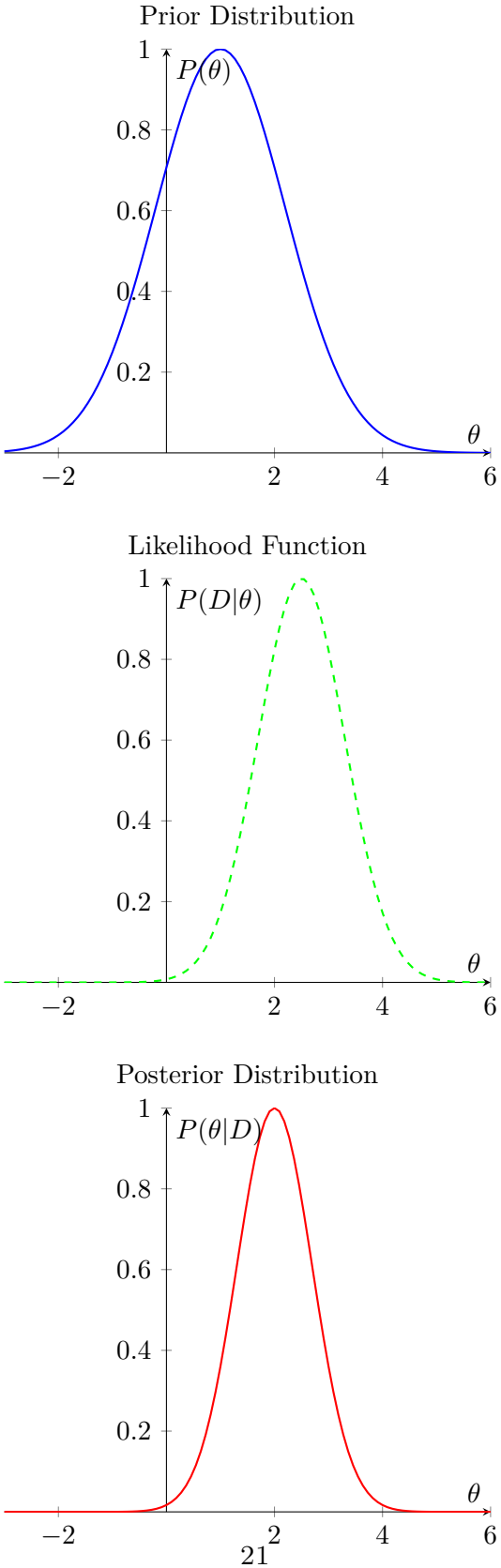


Figure 2.7: Bayesian updating: (Top) Prior distribution, (Middle) Likelihood function, (Bottom) Posterior distribution

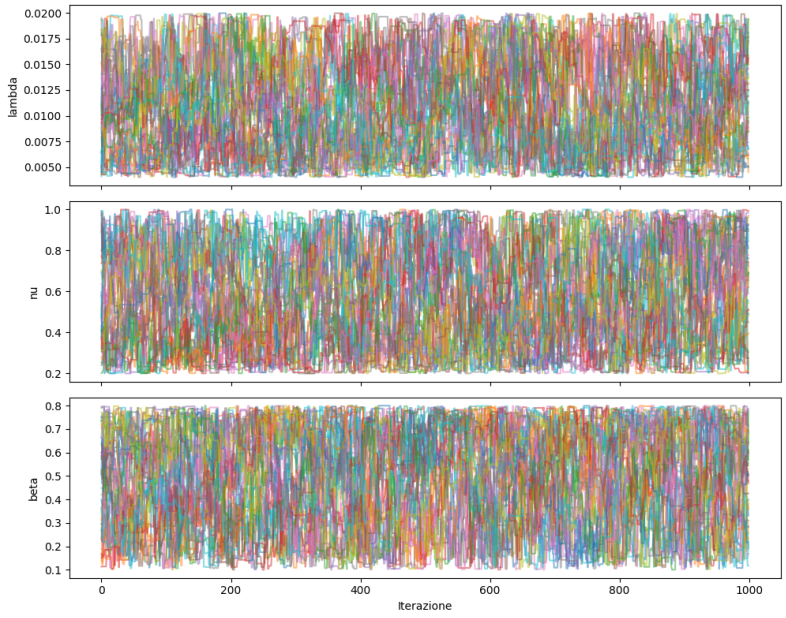
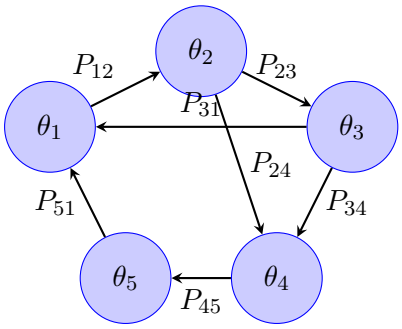
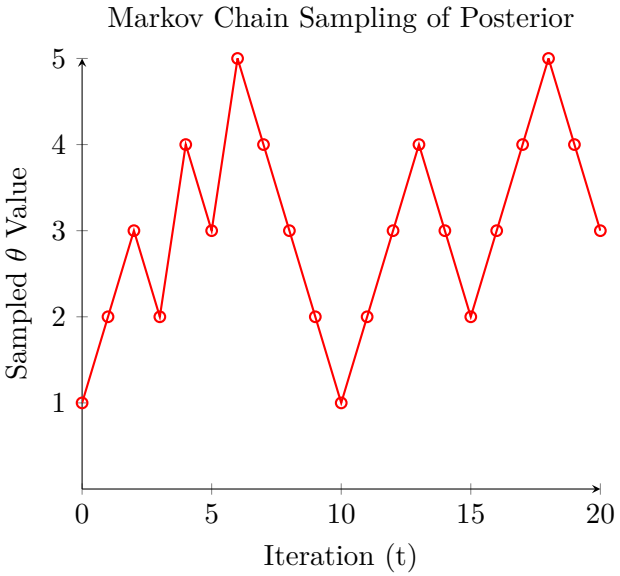


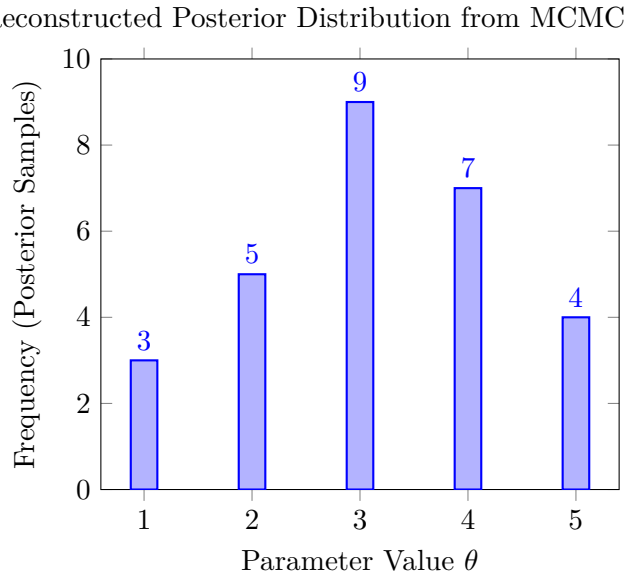
Figure 2.8: An example of MCMC chains that have explored the parameters space in its entirety (notice how dense the plots look)



(a) Markov Chain transitions between parameter states



(b) MCMC sampling process for Bayesian inference



(c) Histogram showing posterior probability reconstruction

Figure 2.9: Markov Chain in Bayesian Inference: (a) A Markov Chain transitions between different parameter states. (b) Over time, an MCMC sampler explores these states, generating samples from the posterior. (c) A histogram of these samples reconstructs the posterior probability distribution.

Chapter 3

Coding and Implementation

Next, we will describe the practical aspects of implementing our SEINR model using Python. The implementation relies on a combination of high-level libraries for numerical computations, data handling, and file management. We will also describe the strategy we employed to numerically solve a time-discrete system of ODEs.

From now on, the non-bayesian strategy of using a fixed distribution of parameters each week (without a posteriori updates) will be considered a particular case of bayesian modelling where the prior distribution is equal to the posterior: this is done to avoid confusion and for brevity's sake, since most sections of the code are similar for both strategies.

3.1 Coding Bayesian Inference

Our approach to Bayesian inference relies on a combination of Python libraries. *NumPy* is used for computationally efficient math operations and array management. For example, `np.linspace` is used to generate uniformly spaced values for parameters like the latency rate (ν), the transmission rate (λ), the infectiousness period (β), and the recovery rate (γ). This allows us to explore a broad range of parameter combinations efficiently.

We rely on *Pandas* for reading and manipulating CSV files containing flu incidence data. A crucial part of the code involves iterating through the available data files, extracting all relevant information (such as region names, year, week, and target values), and aggregating this data into structured dictionaries.

For instance, one section of the code reads regional CSV files, extracts the latest available data for each region, and builds a dictionary containing weekly incidence values. This ensures that the model?s input data is always based on the most recent available records.

3.1.1 Solving the SEINR Model

To numerically solve the system of ODEs, we use a discrete approach, simulating the model on a day-by-day basis. Each time step corresponds to one day, and the numerical integration method used is similar to an explicit Euler integration scheme. At each step,

the state variables ()Susceptible, Exposed, Infectious, Non-infectious, and Removed) are updated according to the differential equations governing the phenomenon (which are explained in detail in chapter 1).

3.1.2 Bayesian Updates and Parameter Estimation

For the Bayesian updates, we integrate prior parameter distributions with simulation results. The implementation follows an iterative approach:

1. A large number of simulations are run for a single week using sampled parameter sets.
2. The model outputs weekly incidence predictions, which are then compared to real incidence data using a Gaussian likelihood function.
3. The likelihood values are used to weight the parameter sets (e.g: which parameters are more likely to cause this data?).
4. The updated posterior is saved and used as the new prior for subsequent iterations.

These steps are repeated each week, and each week our estimates of the parameters should become slightly more adapted to real-world data.

3.2 Forecast File Format for Influcast Repository

To contribute weekly influenza forecasts to the Influcast GitHub repository, our predictions must be formatted according to a predefined structure and saved as CSV files. Each file needs to follow a strict naming convention and contains specific columns to ensure compatibility with the system.

3.2.1 File Naming and Storage

Forecast files are stored within the repository using the following path structure:

`previsioni/Team_X-Modello_Y/2024_05.csv`

where:

- `Team_X` represents the name of the forecasting team.
- `Modello_Y` identifies the model (MetaFlu in our case) used for the predictions.
- `2024_05.csv` refers to the year and week of the forecast.

3.2.2 CSV File Structure

Each forecast file must contain the following columns:

Column Name	Type	Description
`anno`	Integer	Year of the forecast.
`settimana`	Integer	week of the forecast.
`luogo`	String	Location code (national or regional).
`tipo_valore`	String	Always set to "quantile".
`id_valore`	Float	Quantile value (from 0.01 to 0.99).
`orizzonte`	Integer	Forecast horizon (from 1 to 4).
`valore`	Float	Predicted weekly incidence per 1000 patients.
`target`	String	Prediction target (ILI in our case).

Table 3.1: Required columns for forecast CSV files.

3.2.3 Column Details

- **anno, settimana**: The year and epidemiological week of the forecast, stored as integers. These values must match those in the surveillance report and the filename (except for the leading zero in single-digit weeks).
- **luogo**: A two-character code indicating the forecast's geographical scope:
  - `IT`: National forecast.
  - `01 - 21`: Regional codes, following the official mapping:
    - `01`: Abruzzo, `02`: Basilicata, `03`: Calabria, ..., `21`: Veneto.
- **tipo_valore**: This field is always set to "quantile".
- **id_valore**: Represents the quantile for which the forecast is provided. Required quantiles include:

  0.01, 0.025, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.975, 0.99.

- **orizzonte**: An integer indicating the forecast horizon:
  - `-1`, `0`: the most recent and second-most recent surveillance week. Some models include these values in their output file, but they are not necessary in order to participate to the project.
  - `1 - 4`: Predictions for one to four weeks ahead.
- **valore**: A floating-point number representing the predicted weekly incidence (cases per 1000 patients), corresponding to the given week, quantile, and location.
- **target**: Specifies the type of forecast. Allowed values are:
  - `ILI`: Influenza-like illness.
  - `ILI+FLU-A`: Influenza-like illness, including influenza A cases.
  - `ILI+FLU-B`: Influenza-like illness, including influenza B cases.

3.2.4 Example of a Valid Forecast File

Below is an excerpt from a correctly formatted CSV output file:

```
anno,settimana,luogo,tipo_valore,id_valore,orizzonte,valore,target
2023,45,IT,quantile,0.975,1,0.982,ILI
2023,45,IT,quantile,0.975,2,0.995,ILI
2023,45,IT,quantile,0.975,3,1.084,ILI
2023,45,IT,quantile,0.975,4,1.174,ILI
2023,45,IT,quantile,0.5,1,0.934,ILI+_FLU_A
2023,45,IT,quantile,0.5,2,0.956,ILI+_FLU_A
```

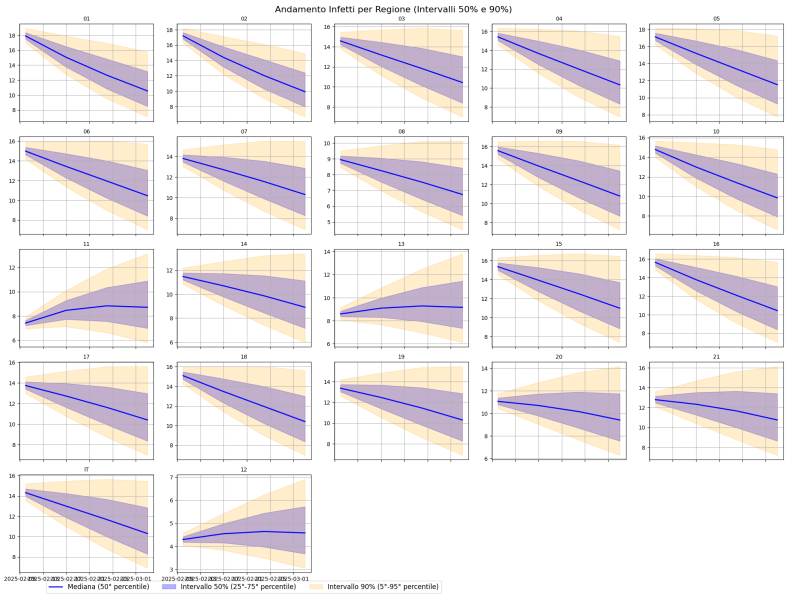


Figure 3.1: A graphical visualization of an output file uploaded to Influcast

Chapter 4

Experiments and Contributions

As explained, the main goal of this thesis was to test whether the SEINR metapopulation model, which had worked well for COVID-19, could also capture influenza dynamics in an effective way. Along the way, I took part in several projects that served as real-world tests for the model and helped shape its development.

4.1 Collaborative Projects

4.1.1 Influcast

Influcast is Italy?s first central hub for epidemiological forecasts, gathering estimates from different research teams on influenza-like illness (ILI) trends at both national and regional levels. The project is coordinated by the ISI Foundation in Turin, and the predictions provided by each team are based on case reports from a network of sentinel doctors, with data provided every Friday by the Italian National Institute of Health (ISS) through the RespiVirNet bulletin.

It should be kept in mind that the reported ILI cases do not just reflect influenza but also include other respiratory viruses like SARS-CoV-2 and Rhinovirus. The platform updates every Wednesday, giving teams enough time to process the latest data, recalibrate their models, and publish forecasts covering the next four weeks.

4.1.2 Respicast

Respicast grew out of the COVID-19 Forecasting Hub, which launched in March 2021 and quickly became an important reference point for European research teams. In November 2023, Respicast expanded to also include forecasts for other respiratory illnesses like ILI and acute respiratory infections (ARI). [10]

Its goal is to give reliable, near-term projections to help public health officials and the general public stay ahead of outbreaks, while also building a strong open-source community of disease modelers.

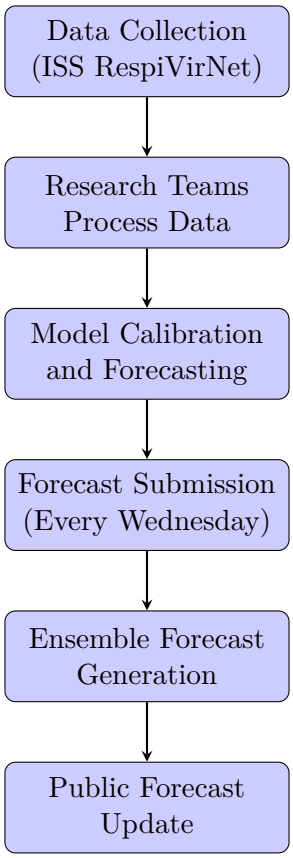


Figure 4.1: Workflow of Influcast: Data is collected from ISS RespiVirNet, processed by research teams, forecasted models are submitted, and an ensemble forecast is generated for public updates.

4.2 Collaborative Paper Published

All this work led to a collaborative paper titled *?Collaborative forecasting of influenza-like illness in Italy: The Influcast experience?* [7]. Over the 2023/2024 winter season, the project carried out 20 forecasting rounds, with five research teams contributing a total of eight different models. These forecasts, which predicted ILI incidence up to four weeks in advances, were combined into an ensemble model.

Our model was labelled as *Mechanistic-1* in this article. As we will see in the section regarding model validation, the ensemble consistently ranked among the best compared to individual models and a baseline forecast. Its performance worsened slightly over longer horizons, but the ensemble still outperformed the baseline across all timeframes.

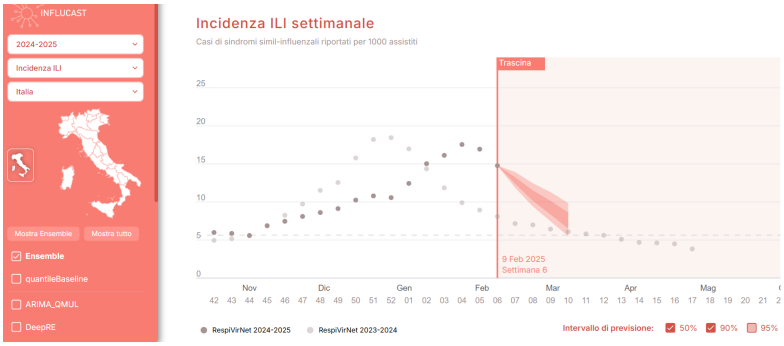


Figure 4.2: Influcast front-end interface, containing past data (both 2023-2024 and 2024-2025 seasons) and forecasts for future weeks.

Figure 4.3: Respicast front-end interface, containing past data (2024-2025 season) and forecasts for future weeks.

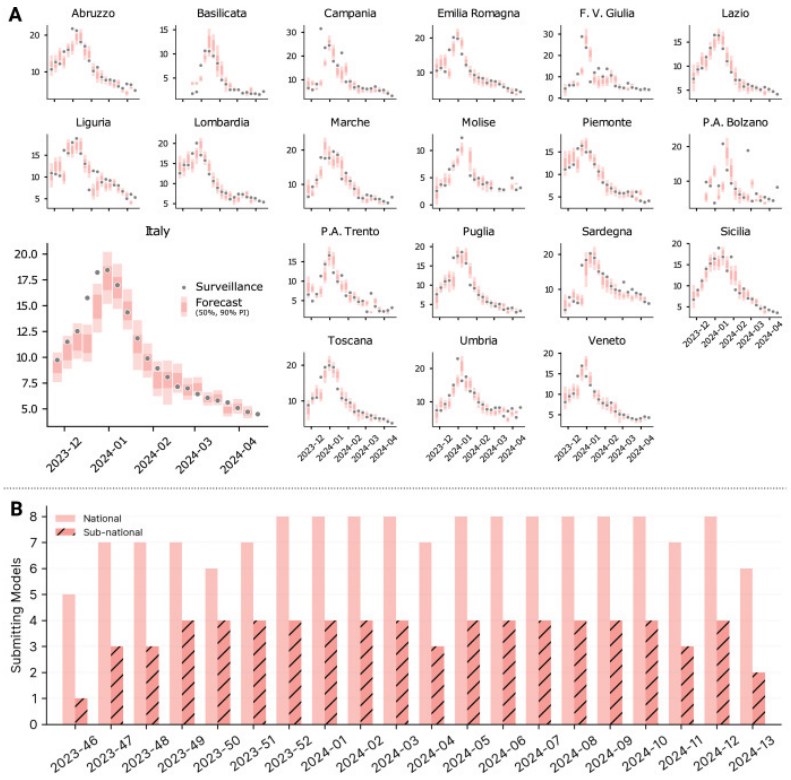


Figure 4.4: (A): Ensemble predictions for a 1-week horizon (with 50 percent and 90 percent confidence intervals) at the national and regional level in different submission rounds. (B) How many models submitted predictions for each submission round. Taken from [7]

Chapter 5

Model Validation

In this chapter, a comparison of two strategies for the SEINR model's parameters update will be given: the non-Bayesian method, which, as explained, operates with fixed parameter ranges, and the Bayesian method, which utilizes data to update parameters weekly. The goal is to determine which of the methods shows better results according to certain error metrics. Specifically, the measures that will be used are:

- the log score
- weighted interval score
- mean relative error
- mean absolute error
- relative MAE
- the 50 percent and 90 percent coverage rates.

First of all, the log score assesses the performance of a probabilistic forecast by taking the logarithm of the probability of the event which actually happened. In other words, if a forecast is better, then its probability will be higher and thus the log score will be higher (less negative). One of the biggest advantages of the log score is its ability to properly penalize?e predictions that are overconfident about the wrong outcome. Nevertheless, it may be too sensitive of a measure, especially if the forecast is trying to predict rare events.

The Weighted interval score (WIS) is intended for measuring both the precision and the focus of a forecast through prediction intervals. This is achieved by comparing the predicted quantiles with the actual observed value. The benefit of the WIS is that it can show the overall quality of the forecast distribution, not just a single point estimate. It is very useful especially when the forecasts are expressed in terms of several quantiles. The negative side, however, is that WIS may be harder to interpret compared to direct measures like MAE, as it brings together information concerning the spread and the central tendency of the forecast.

Next, we have the mean absolute error (MAE) and the mean relative error (MRE). The MAE is a simple metric that gives you the average of the differences of values of the predicted and observed ones. This allows us to understand how severe the average error is without considering whether the forecast is under or over the real value. In comparison, the MRE scales these errors by the observed values, which is very helpful, especially when we are dealing with data that have different scales. Another related metric, the relative MAE, expresses these errors as a ratio, thus, enabling the comparison of the errors across different regions or time periods. However, the drawback of these metrics is that they are based only on point estimates and thus, do not consider the uncertainty that is naturally occurring in probabilistic forecasts, like the ones we are dealing with.

Additionally, coverage percentages (50 percent and 90 percent) are used to evaluate the performance of the forecast intervals in capturing the real results. In other words, if the prediction interval is 50 percent, then about half of the observed values should approximately fall within that range. At the same time, a 90 percent interval should enclose approximately 90 percent of the observations within it. High-coverage shows that the intervals are consistent with the actual realizations, nevertheless, in the case of their being too wide, the forecast may be of lesser quality for decision-making. Low coverage, on the opposite, implies that the intervals are too narrow, hence, they do not cover up the uncertainty.

By examining the Bayesian and non-Bayesian strategies based upon these metrics, a few important findings can be made. The Bayesian method has an advantage because the estimates are the output of a recurrent learning process by utilizing the most recent data, thus allowing for improved calibration and lower errors in the long run. This technique is particularly viable in dynamic scenarios where the circumstances are changing rapidly. However, computational costs increase and the model becomes more complex.

Conversely, the non-Bayesian solution is more straightforward and does not require excessive processing power. However, a larger range of fixed parameters risks providing uninformative, while mostly correct, estimates.

Still, the non-Bayesian method is probably sufficient as a reference point, in order to at least get a rough estimate of the evolution of the contagion.

5.1 Overfitting

An important aspect to keep in mind when trying to model real phenomena is overfitting. The Bayesian approach is in general quite flexible, adapting or refining its parameters when presented new informative data; however, this flexibility can sometimes lead to the model fitting the noise or the peculiarities of the particular data too closely. Overfitting appears when a model not only models the basic trend but also random fluctuations, and this leads to an inability to generalize to new data.

To control overfitting, we suggest resetting the Bayesian optimization at the start of the flu season, which implies "forgetting" old data and starting over.

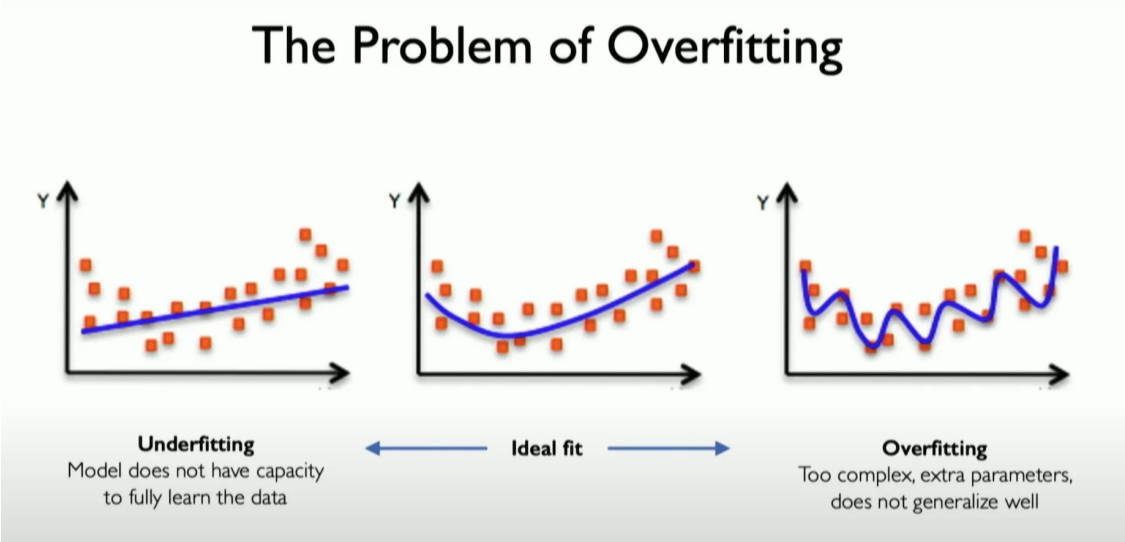


Figure 5.1: Concept of overfitting explained. Taken from [6]

5.2 Christmas Holiday issue

Our models were affected by a major flood of flu-like illnesses that turned out to be much more than we expected in the Christmas season. This increase, which was linked to modifications in social behavior, increased travel, and holiday gatherings, was in contrast to the normal patterns witnessed during non-holiday periods. Consequently, the non-Bayesian model, that uses fixed parameter ranges, or the Bayesian model both faced difficulty in capturing the outbreak accurately. It is important to note that almost every model that participated to the project failed to correctly predict epidemic dynamics in this particular period.

The fixed-parameter, non-Bayesian approach became greatly inflexible in this particular scenario. Due to the fact that it works with pre-set parameter intervals, it could not properly align the change in the epidemic?s dynamics with it, which consequently led to forecasts that were significantly divergent from the real data in this particular period. In contrast, bayesian method, with its weekly parameter updates, was partially capable of dealing with the sudden shift. Despite the fact that both of the models were imperfect in that difficult-to-model period, the Bayesian approach sort of suppressed the error rates by being more dynamic towards the new data.

This is, in fact, the period that will qualify as a reference point for us to be able to give a certain benchmark of how much the Bayesian method has proved to be better than the initial non-Bayesian approach. In the Numerical Results section, we will insert our original model figure among the figures of both the Bayesian and the Non-Bayesian strategies that will show us the performance related to the Christmas season.

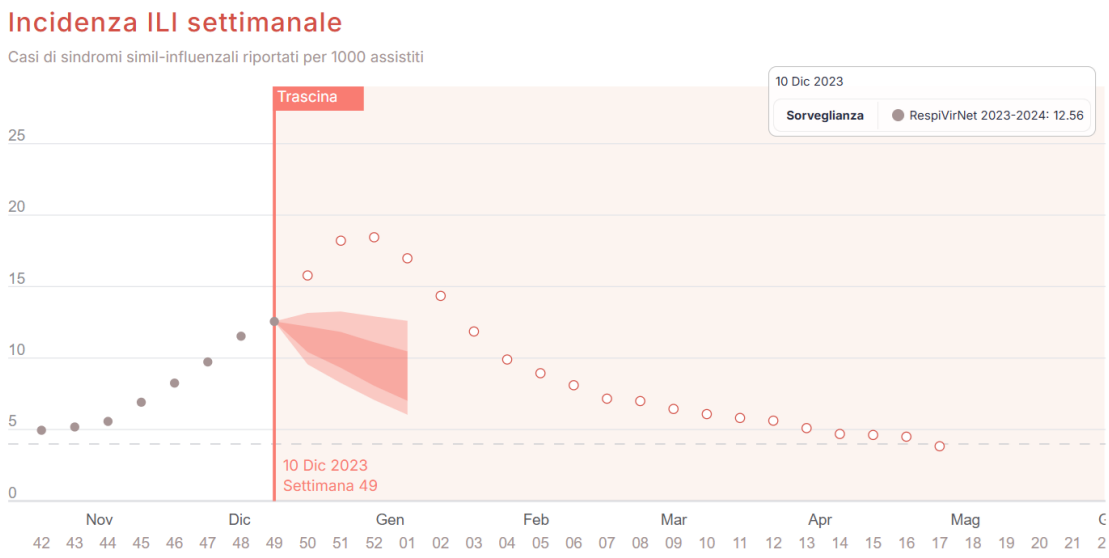


Figure 5.2: Performance of the ensemble model in the Christmas Holydays season. Notice the poor performance and the Incidence surge in the last weeks of the year.

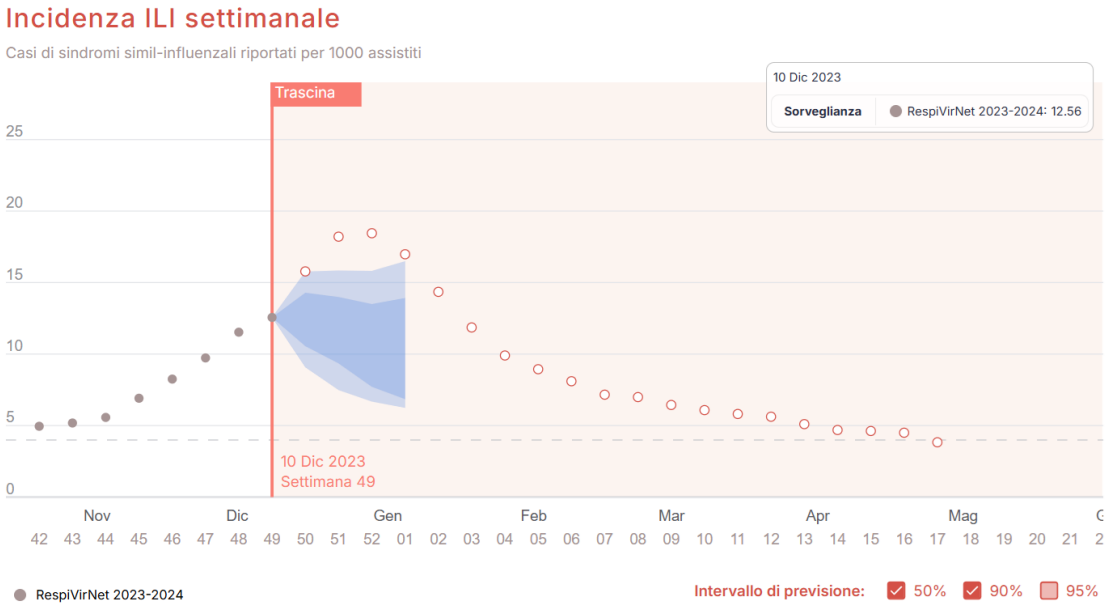


Figure 5.3: Performance of the original, non-bayesian model in the Christmas Holydays season. Notice the poor performance and the Incidence surge in the last weeks of the year, along with the large width of the confidence intervals.

5.3 Numerical results

Below is an overview of the results atttained by our deterministic, non-Bayesian model compared to the other models that participated to the Influcast project

5.3.1 General Overview

As explained, the evaluation will focus on a few performance metrics: relative MAE of the median, relative WIS, and the 50 percent and 90 percent coverage rates. For a thorough explanation of what each of those terms mean, see the beginning of this Chapter.

Model	N. of rounds	Relative MAE	Relative WIS	Coverage (50%)	Coverage (90%
Mechanistic-1	19	**0.56** (1st)	**0.54** (1st)	0.75	**0.89** (1st)
Mechanistic-2	20	1.86	1.64	0.04	0.49
Mechanistic-3	20	0.58 (2nd)	0.58 (3rd)	0.47 (3rd)	0.70 (3rd)
Mechanistic-4	19	0.73	0.73	0.16	0.37
Semi-mechanistic-1	16	1.80	2.14	0.06	0.27
Semi-mechanistic-2	14	0.84	1.04	0.19	0.30
Statistical-1	20	0.99	1.09	0.17	0.43
Statistical-2	19	0.77	0.85	0.11	0.31
Ensemble	20	0.59 (3rd)	0.57 (2nd)	**0.51** (1st)	0.80 (2nd)
Baseline	20	1.0	1.0	0.49 (2nd)	0.66

Figure 5.4: Predictive performance of each model. How different models performed in terms of relative MAE of the median, relative WIS, 50 percent and 90 percent coverage. The best model for each metric is highlighted in bold. Taken from [7]

At first glance, it seems that our non-Bayesian model (Mechanistic-1) performed quite decently overall, ranking first in a few metrics and attaining results that are close to the ensemble model. It should be kept in mind that this general benchmark does not focus to the problematic timeframe we mentioned earlier, which is the Christmas Holidays season. This timeframe will be discussed in detail later. Still, our non-Bayesian model performed reasonably well and serves as a good starting point for future improvements.

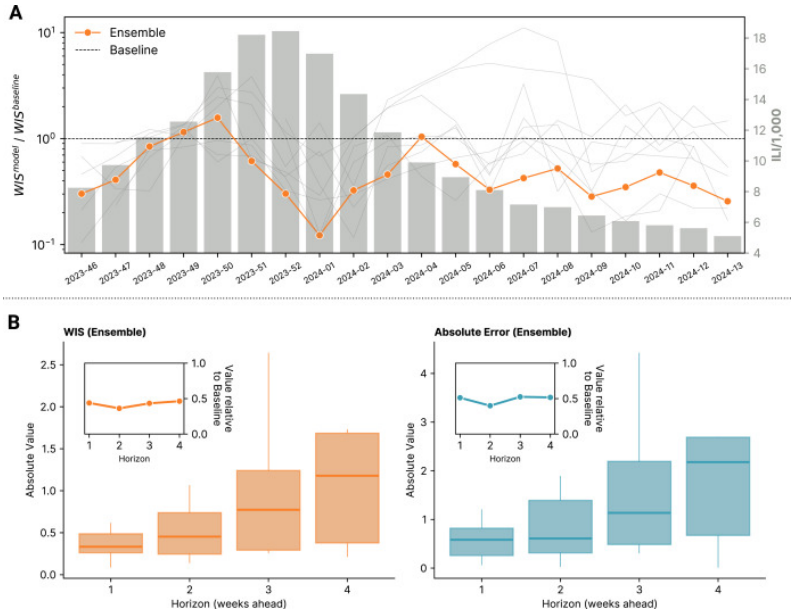


Figure 5.5: Performance of submitted models in time and by horizon. (A) Ratio between average WIS of different models and of the baseline (both obtained averagign for 1 to 4 weeks horizons) for various forecast rounds. If a value is smaller than 1, that model performs better than baseline. The ensemble model is plotted in orange, while baseline model is the black dashed line. The vertical bars in the background show the reported incidence for that week. (B) Absolute WIS values of the Ensemble model for various horizons (from to weeks ahead). On the right, we repeat the analysis considering the absolute error of the median as a performance metric. The box boundaries represent the interquartile range (IQR), the line inside the box indicates the median and the whiskers extend to 1.5 times the IQR from the quartiles. Taken from [7]

5.3.2 Christmas Holidays Benchmark

Having established that our non-Bayesian model yields good results at the seasonal level, let us see how the Bayesian model performs using those same metrics as a benchmark. The attached tables list these performance metrics for every region, pointing out where the non-Bayesian model performs on par with or lags behind the more adaptive Bayesian strategy.

Overall, the results seem to indicate that while our deterministic model can capture general trends, the Bayesian method tends to offer more accurate, narrower and well-calibrated forecasts, especially in regions with high variability.

Table 5.1: Performance Metrics - Bayesian Model

Region	Log Score	WIS	MAE	MRE	Rel. MAE	Cov. 50%	Cov. 90%
abruzzo	−2.86	5.6	1.35	$7.42 \cdot 10^{-2}$	$7.42 \cdot 10^{-2}$	0.89	0.89
basilicata	−4.13	7.21	1.07	0.43	0.43	0.71	0.71
campania	−2.95	6.25	2.57	0.15	0.15	0.44	0.78
emilia-romagna	−2.71	5.46	1.56	$9.54 \cdot 10^{-2}$	$9.54 \cdot 10^{-2}$	0.67	0.89
friuli-venezia-giulia	−2.74	6.33	1.63	0.13	0.13	0.67	0.89
italia	−2.02	4.37	1.15	$7.26 \cdot 10^{-2}$	$7.26 \cdot 10^{-2}$	0.78	1
lazio	−1.74	4.17	0.98	$8.4 \cdot 10^{-2}$	$8.4 \cdot 10^{-2}$	0.89	0.89
liguria	−2.23	4.85	1.77	0.11	0.11	0.56	0.78
lombardia	−1.65	4.29	1.33	$8.84 \cdot 10^{-2}$	$8.84 \cdot 10^{-2}$	0.89	1
marche	−2.82	6.5	1.95	0.12	0.12	0.56	0.78
molise	−3.64	4.7	1.2	0.24	0.24	0.5	0.75
pa-bolzano	−3.23	5.07	1.72	0.14	0.14	0.71	0.86
pa-trento	−3.48	9.33	1.72	0.22	0.22	0.56	0.78
piemonte	−1.97	4.23	1.25	$9.23 \cdot 10^{-2}$	$9.23 \cdot 10^{-2}$	0.56	1
puglia	−2.17	5.19	1.59	0.11	0.11	0.56	0.89
sardegna	−2.75	4.91	1.44	0.13	0.13	0.44	0.89
sicilia	−1.85	4.23	1.19	$7.79 \cdot 10^{-2}$	$7.79 \cdot 10^{-2}$	0.56	1
toscana	−2.06	4.37	1.26	$8.05 \cdot 10^{-2}$	$8.05 \cdot 10^{-2}$	0.67	1
umbria	−2.68	5.17	1.45	0.1	0.1	0.67	0.89
veneto	−1.7	4.06	1.11	$8.86 \cdot 10^{-2}$	$8.86 \cdot 10^{-2}$	0.67	1

After presenting these tables, let us focus on the more problematic timeframe. Next, is a series of figures that compare the results of the two models from December 2023 to the first half of January 2024.

Table 5.2: Performance Metrics - Bayesian Model

Region	Log Score	WIS	MAE	MRE	Rel. MAE	Cov. 50%	Cov. 90%
abruzzo	−6.42	38.05	4.83	0.27	0.27	0.22	0.33
basilicata	−5.37	32.01	3.19	1.39	1.39	0.57	0.57
campania	−6.66	85.23	7.66	0.48	0.48	0.11	0.22
emilia-romagna	−4.94	31.36	3.78	0.23	0.23	0.44	0.44
friuli-venezia-giulia	−5.54	62.76	5.77	0.44	0.44	0.33	0.44
italia	−5.53	26.76	3.61	0.22	0.22	0.22	0.56
lazio	−4.11	13.47	2.71	0.22	0.22	0.22	0.67
liguria	−6.55	50.76	5.52	0.36	0.36	0.11	0.33
lombardia	−5.08	28.26	3.81	0.24	0.24	0.33	0.44
marche	−7.38	42.62	5.08	0.31	0.31	0.22	0.22
molise	−6.14	26.8	3.83	0.79	0.79	0.13	0.38
pa-bolzano	−6.48	68.77	5.64	0.45	0.45	0.14	0.43
pa-trento	−4.99	42.58	4.69	0.64	0.64	0.11	0.56
piemonte	−5.51	23.15	3.28	0.22	0.22	0.33	0.44
puglia	−5.57	24.63	3.51	0.24	0.24	0.44	0.44
sardegna	−8.25	32.93	4.68	0.39	0.39	0.11	0.22
sicilia	−5.95	22.14	3.92	0.26	0.26	0.22	0.33
toscana	−5.55	38.1	4.23	0.25	0.25	0.44	0.44
umbria	−5.81	35.97	4.44	0.31	0.31	0.22	0.44
veneto	−5.47	12.28	2.4	0.18	0.18	0.44	0.56

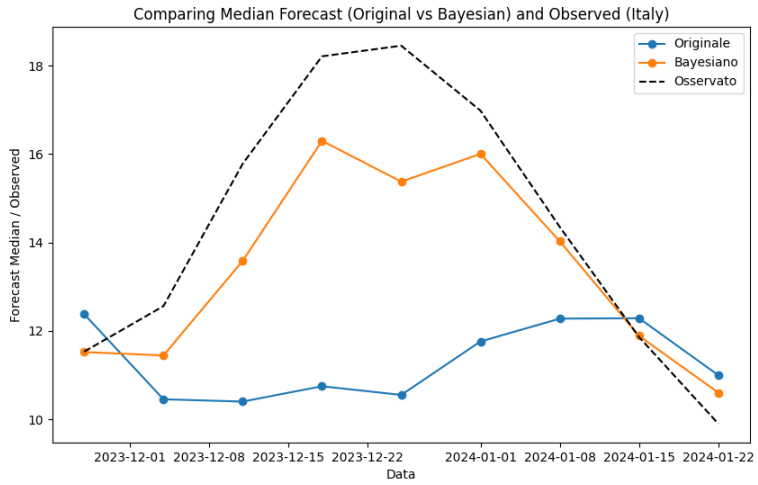


Figure 5.6: Comparison of median forecast values from the Bayesian and non-Bayesian models against observed incidence in Italy. The Bayesian approach tends to better capture peaks and variability.

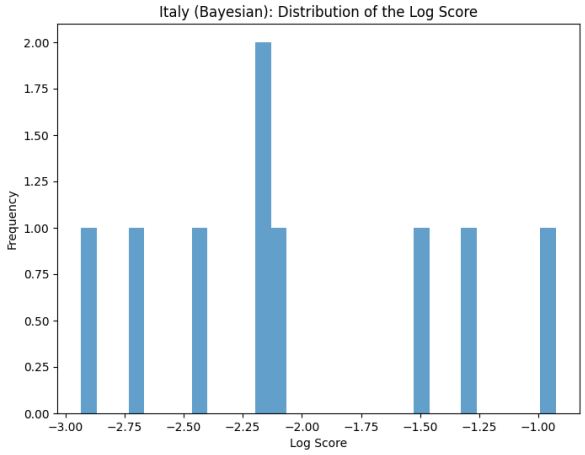


Figure 5.7: Distribution of log scores for the Bayesian model. A higher concentration around less negative values suggests improved forecast reliability.

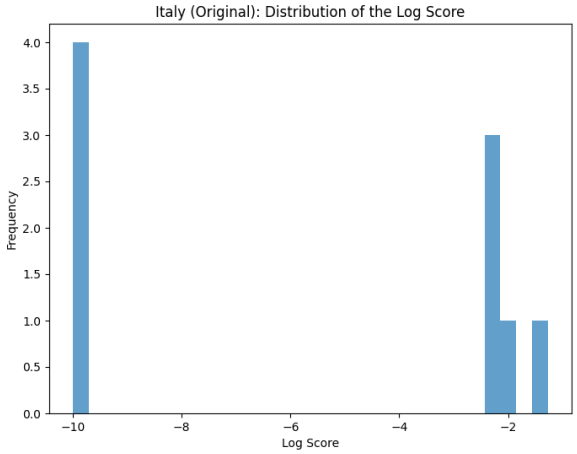


Figure 5.8: Distribution of log scores for the non-Bayesian model. Compared to the Bayesian approach, the log scores exhibit more spread, indicating less consistent predictions.

Although neither model perfectly captured the dramatic holiday spike, the Bayesian forecasts adapted more rapidly to the incoming data, producing results that were generally closer to the observed values and better calibrated than the fixed-parameter method.

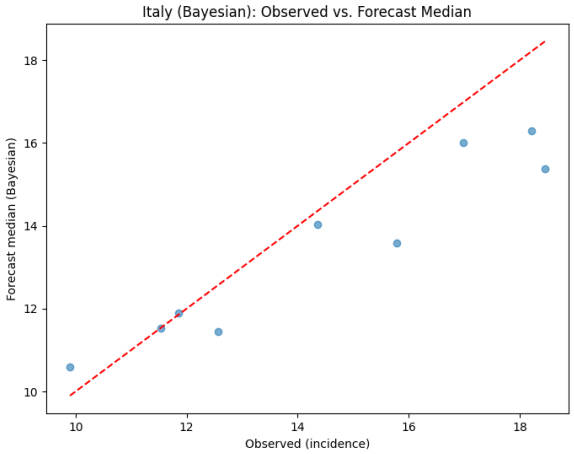


Figure 5.9: Observed incidence vs. Bayesian forecast median. The red dashed line represents a perfect match, showing that the Bayesian model aligns well with real data.

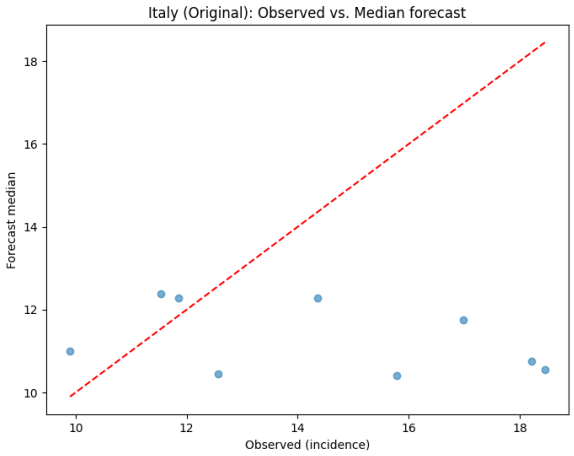


Figure 5.10: Observed incidence vs. non-Bayesian forecast median. The deviations from the diagonal suggest that the deterministic model struggles more with capturing real trends.

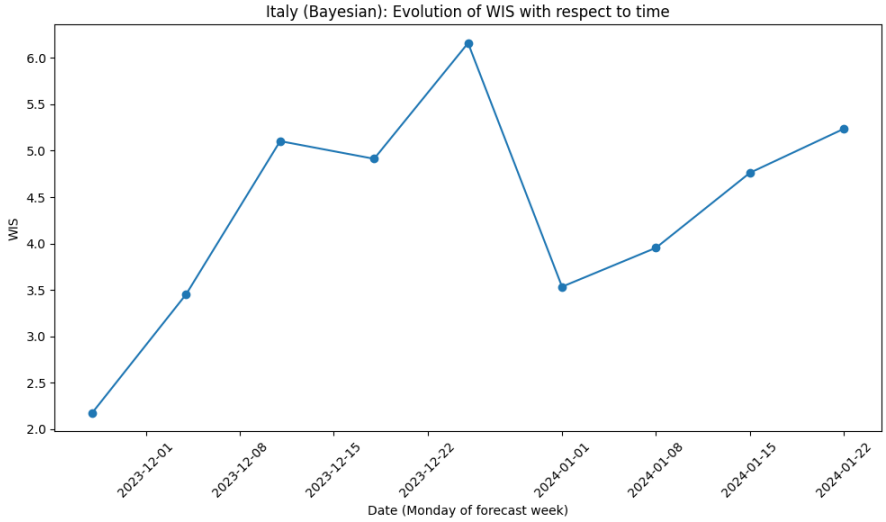


Figure 5.11: Evolution of the Weighted Interval Score (WIS) over time for the Bayesian model. Fluctuations indicate how forecast accuracy varies across different weeks.

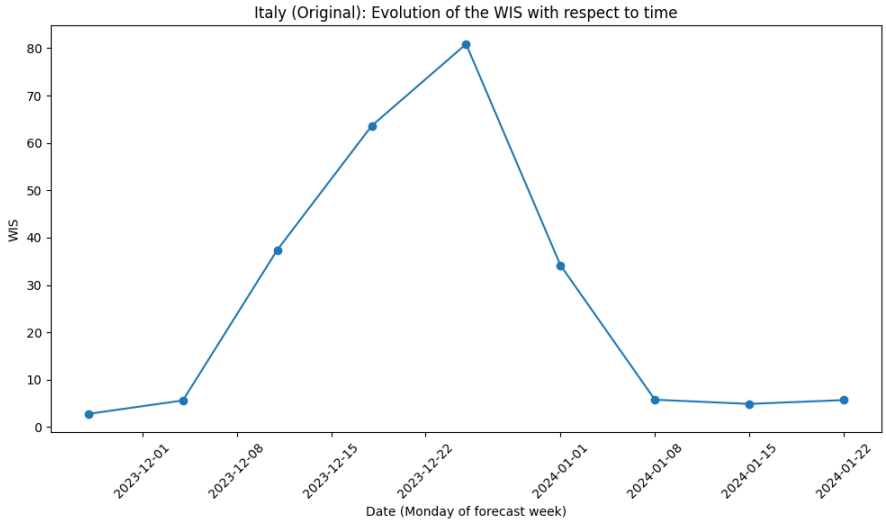


Figure 5.12: Evolution of the Weighted Interval Score (WIS) over time for the non-Bayesian model. The large spikes suggest periods of higher uncertainty and lower forecast reliability.

Chapter 6

Conclusion

This thesis set out to test whether a SEINR metapopulation model that was originally developed for COVID-19, could also be effective in predicting influenza trends in Italy. Along the way, we added a Bayesian approach to refine parameter estimates as weeks went by. The results show that while the deterministic model captures overall seasonal trends decently well, the Bayesian method tends to improve accuracy and confidence intervals, especially in regions where incidence fluctuates more (probably because of small sample sizes). That said, this improvement comes at a cost, quite literally. The computational expense of running Bayesian inference for multiple parameters is significant, which raises the question of how to balance precision and efficiency.

6.1 Limitations and Challenges

A few limitations should be kept in mind if someone wants to expand on my work. One of the biggest challenges is the mismatch between the **provincial** structure of our commuting matrix and the **regional** scale of the real-world data. Since mobility is modeled at a finer level than the available incidence reports, there is an inherent inconsistency that could be affecting the results. Perhaps this could be solved by adapting the metapopulation framework to a regional level of granularity (instead of a provincial one)

Another issue is that the model operates on a **daily** scale, but the real-world data is only available on a **weekly** basis. This means that any short-term fluctuations that the model picks up are essentially smoothed out in the evaluation, which could make it harder to assess the true responsiveness of the forecasts. This issue appears even more important if someone considers that the bayesian model is essentially trained on this kind of weekly data.

Then there is the question of vaccination data. Right now, we assume a fixed effectiveness for the flu vaccine, but in reality, vaccine efficacy depends on a lot of factors, including the circulating strains, population immunity, and even the timing of the vaccination campaign. More granular data on this would help improve the model?s reliability.

Lastly, our model is built for single-season forecasts, meaning it doesn?t account for reinfections or waning immunity over time. This makes it less useful for longer-term

epidemiological planning.

6.2 The Bayesian Trade-off

The Bayesian approach most likely improves forecast precision, but it also adds a heavy computational burden. Running full Bayesian inference for multiple parameters, week after week, is expensive, and if we were to expand this model to a larger scale, the costs would grow even further.

A possible solution would be to limit Bayesian updates to just a few parameters, such as the transmission rate. At the same time, we still wonder whether the non-infectious compartment (N) is actually necessary, or if it just makes the model more complicated without adding much value. We should always try to remember the usual trade-off between underfitting and overfitting, if the model is too simple, it misses key patterns, but if it is too complex, it risks learning random noise rather than real trends.

6.3 Future Directions

There is a lot of room to build on this work in the future. My suggestions include:

- Using machine learning techniques to estimate parameters instead of full Bayesian inference: this could make the model calibration faster while still allowing it to adapt over time.
- Exploring deep learning approaches to complement the compartmental model, making use of past flu seasons to make predictions.
- Expanding the model to account for waning immunity and reinfections, so that it can be used beyond a single flu season.
- Integrating higher-resolution data sources, such as real-time mobility tracking, or at least more exact data concerning Christmas holidays

While this model provides a foundation for flu forecasting in Italy, there is still plenty of room to refine it. Whether that means simplifying some components, adding more data, or finding new ways to estimate parameters, there is a lot to explore in the intersection between epidemiological modeling and data science.

I am personally quite curious to see what the future holds, and if my works serves as a small step in the right direction, whether that means helping real people in their real-life problems using mathematics or simply advancing human knowledge, I will be deeply satisfied. Once again, my acknowledgements go to everyone that helped my in this long journey, and I should especially thank Lorenzo, Alessandro and Elisa, without them I would not have had the occasion to work on such a stimulating project.

Appendix A

Python Implementation

Listing A.1: Initializing the model

```
def initialStatesMemory(p, initialCaseI=None, initialCaseE=None,
    start_date=None, initialN=None, initialR=None):
shape = (3, p.Whk.shape[0])  # Tre classi: giovani, anziani, anziani
    vaccinati

I = np.zeros(shape)
E = np.zeros(shape)
N = np.zeros(shape)
R = np.zeros(shape)

# Distribuzione degli infetti iniziali
if initialCaseI is not None:
I[0] = initialCaseI * p.eta[0]  # Giovani
I[1] = initialCaseI * p.eta[1]  # Anziani
I[2] = initialCaseI * p.eta[2]  # Anziani vaccinati

# Distribuzione degli esposti iniziali
if initialCaseE is not None:
E[0] = initialCaseE * p.eta[0]  # Giovani
E[1] = initialCaseE * p.eta[1]  # Anziani
E[2] = initialCaseE * p.eta[2]  # Anziani vaccinati

#distribuzione non contagiosi iniziali
if initialN is not None:
N[0] = initialN * p.eta[0]  # Giovani
N[1] = initialN * p.eta[1]  # Anziani
N[2] = initialN * p.eta[2]  # Anziani vaccinati

#distribuzione rimossi iniziali
if initialR is not None:
R[0] = initialR * p.eta[0]  # Giovani
R[1] = initialR * p.eta[1]  # Anziani
R[2] = initialR * p.eta[2]  # Anziani vaccinati

# Calcoliamo i suscettibili iniziali
Stot = np.ones(shape) * p.eta.reshape(-1, 1).dot(p.nh.reshape(1, -1))
S = Stot - E - I - N - R
```

```
    if np.any(S < 0):
    raise ValueError("Numero negativo di suscettibili, errore nell'
        inizializzazione.")


    if start_date is not None:
    start_date = datetime.strptime(start_date, '%d-%m-%Y')
    else:
    start_date = datetime.now()

    return S, E, I, N, R, start_date
```

Listing A.2: Advancing the simulation by one time step

```
# this function advances our model by one time step (one day)
def oneStep(SEINR, p):
S, E, I, N, R = SEINR
Lambda, Nu, Beta, Gamma = p.Lambda, p.Nu, p.Beta, p.Gamma
m, eta, ai, alpha, nh, b, Whk = p.m, p.eta, p.ai, p.alpha, p.nh, p.b, p.
    Whk
aAvg = (eta * ai * alpha).sum()
nhTilde = (1 - b * aAvg) * nh + (b * aAvg * (Whk @ nh))
ph = (1 / nhTilde) * (((1 - b) * alpha * ai).reshape(-1, 1) * I).sum(0) +
    (Whk @ (((b * alpha * ai).reshape(-1, 1) * I).sum(0).reshape(-1, 1))).
    sum(0)
PIt1 = (alpha * ai * (1 - b) * Lambda * m).reshape(-1, 1) * ph
PIt2 = ((1 - alpha * ai * b) * Lambda * m).reshape(-1, 1) * ph
PI = PIt1 + PIt2
St1 = (1 - PI) * S
Et1 = PI * S + (1 - Nu) * E
It1 = Nu * E + (1 - Beta) * I
Nt1 = Beta * I + (1 - Gamma) * N
Rt1 = R + Gamma * N
S, E, I, N, R = St1, Et1, It1, Nt1, Rt1

# these conditions make sure that there no compartments with <0 people
if S.min().min() < 0:
E += np.minimum(0, S)
S = np.maximum(0, S)
if E.min().min() < 0:
I += np.minimum(0, E)
E = np.maximum(0, E)
if I.min().min() < 0:
N += np.minimum(0, I)
I = np.maximum(0, I)
if N.min().min() < 0:
R += np.minimum(0, N)
N = np.maximum(0, N)
R = np.maximum(0, R)

_nh = S.sum(0) + E.sum(0) + I.sum(0) + N.sum(0) + R.sum(0)
assert np.isclose(nh, _nh, rtol=0, atol=1e3).all(), f"Population not
    preserved! {nh}, {_nh}"

return S, E, I, N, R
```

Listing A.3: Vaccination and death count functions

```
def calculate_deaths(R, mortality_rate):
deaths = (R * mortality_rate.reshape(-1, 1)).sum(axis=0)
return deaths


def vaccinate_elders(S, p, vaccinate_rate):
vaccinated_elderly_idx = 2  # index for the new class of vaccinated elder
    people
elderly_idx = 1  # index for the class of elder people who choose not to
    vaccinate


num_elderly_to_vaccinate = S[elderly_idx] * vaccinate_rate

# our assumption is that only people in the S compartment get vaccinated
S[vaccinated_elderly_idx] += num_elderly_to_vaccinate
S[elderly_idx] -= num_elderly_to_vaccinate

return S
```

Listing A.4: Defining the parameters class

```
    class parameters(): #parametri che dovremo variare
    def __init__(self, nh, Whk):
    self.nh = nh # numero di persone nella comunità h
    self.Whk = Whk # matrice pendolarismo fra province
    self.ai = np.array([0.149, 0.545, 0.545])  # Livelli di attività: [
        giovani, anziani, anziani vaccinati. Va preso il reciproco perché
        l'unità di misura è 1/days]
    #self.ai = np.array([1.0, 0.5, 0.5])
    self.eta = np.array([0.5, 0.25, 0.25])  # Proporzione di popolazione
        in ciascuna classe iniziale
    #self.b = 0.002  # Tasso di contatto tra province in un caso estremo
        di lockdown draconiano
    #self.b = 0.02 # Tasso di contatto tra province con viaggi fortemente
        ridotti
    self.b = 0.09 # parametrò di mobilità

    self.Lambda = np.array([1, 1.3, 1.2]) * (10**-2) * 1 # Tasso di
        trasmissione
    #self.Nu = 0.1  # Tasso di infezione da esposto a infetto
    #self.Beta = 0.05  # I -> N
    #self.Gamma = 0.01  # Tasso di rimozione (guariti)

    self.Nu = 0.3
    self.Beta = 0.3
    self.Gamma = 0.1
    self.alpha = np.array([1.0, 0.8, 0.9])  # Efficacia dell'
        autoisolamento: [giovani, anziani, anziani vaccinati]. 1 indica no
         autoisolamento
    self.m = 19.77  # numero medio di contatti
```

Listing A.5: Retrieves saved memory from previous weeks of simulation

```
import pandas as pd
import glob
import os

def carica_memoria_recente():
# questa funzione trova il file di memoria più recente fra quelli
    nella cartella di lavoro
file_pattern = "memoria_N_R_giorno_7_*.csv"
files = glob.glob(file_pattern)

if not files:
print("Nessun file di memoria trovato. Inizializzazione da zero.")
return None, None


latest_file = max(files, key=os.path.getmtime)
print(f"Caricamento della memoria da {latest_file}")

# Carica i dati
memoria = pd.read_csv(latest_file)

# Converte i valori di N e R in array NumPy
initialN = memoria["N_medio_giorno"].to_numpy()
initialR = memoria["R_medio_giorno"].to_numpy()

return initialN, initialR

# Carica la memoria più recente
initialN, initialR = carica_memoria_recente()

# se non ci sono file salvati, iniziamo con N e R vuoti
if initialN is None or initialR is None:
initialN = np.zeros(len(orderedPROV))
initialR = np.zeros(len(orderedPROV))
```

Listing A.6: Saves current state to be used in future simulations

```
    # Calcola la data di cui verrà salvato il numero di N e R (di regola,
        una settimana dopo l'inizio della simulazione)
    data_salvataggio = current_date + timedelta(days=giorno_salvataggio -
        1)


    province_results = pd.DataFrame({
        "Provincia": orderedPROV,
        "N_medio_giorno": N_mean,
        "R_medio_giorno": R_mean
    })


    file_name = f"memoria_N_R_giorno_{giorno_salvataggio}_{
        data_salvataggio.strftime('%Y-%m-%d')}.csv"
    province_results.to_csv(file_name, index=False)
    print(f"Risultati di N e R salvati per il giorno {giorno_salvataggio}
        ({data_salvataggio.strftime('%Y-%m-%d')}) in {file_name}")
```

Listing A.7: Bayesian optimization

```
def simulate_one_week(lam, nu, beta, gamma, p, S, E, I, N, R, days=7):
p.Lambda = np.array([lam, lam*1.3, lam*1.2])
p.Nu = nu
p.Beta = beta
p.Gamma = gamma

weekly_new_infections = 0
for _ in range(days):
simulated_exposed = E.sum(axis=0)
daily_new_infected = simulated_exposed * nu
weekly_new_infections += daily_new_infected.sum()
try:
S, E, I, N, R = oneStepCorretto((S, E, I, N, R), p)
#S, E, I, N, R = oneStep((S, E, I, N, R), p)
except AssertionError:
# Se la popolazione non si conserva arrestiamo tutto
return None, None, None, None, None, -1

return S, E, I, N, R, weekly_new_infections

def log_prior(theta, prior_info):
lam, nu, beta = theta
# usiamo delle prior semplici ma leggermente informative
if 4e-3 < lam < 1.5e-2 and 0.2 < nu < 0.7 and 0.1 < beta < 0.6:
return 0.0
return -np.inf

def log_likelihood(theta, data_osservata_per_1000, S, E, I, N, R, p,
    gamma_fixed=0.1, nh=None):
lam, nu, beta = theta
S_f, E_f, I_f, N_f, R_f, weekly_infections = simulate_one_week(lam, nu
    , beta, gamma_fixed, p, S.copy(), E.copy(), I.copy(), N.copy(), R.
    copy())
if weekly_infections == -1:
return -np.inf
if (S_f is None) or (E_f is None):
return -np.inf

if (S_f<0).any() or (E_f<0).any() or (I_f<0).any() or (N_f<0).any() or
     (R_f<0).any():
return -np.inf

total_pop = nh.sum()
incidenza_simulata_per_1000 = (weekly_infections / total_pop) * 1000

sigma = 5.0
ll = norm.logpdf(data_osservata_per_1000, loc=
    incidenza_simulata_per_1000, scale=sigma)
return ll

def log_posterior(theta, data_osservata_per_1000, S, E, I, N, R, p,
    prior_info, gamma_fixed=0.1, nh=None):
lp = log_prior(theta, prior_info)
```

```
    if np.isinf(lp):
    return -np.inf
    ll = log_likelihood(theta, data_osservata_per_1000, S, E, I, N, R, p,
        gamma_fixed=gamma_fixed, nh=nh)
    return lp + ll
```

Listing A.8: MCMC iterations

```
# carichiamo i dati che rimarranno costanti per tutta la calibrazione
pop_df = pd.read_csv('Data/FinalForCommuting/pop.csv')
pop_df['Territorio'] = pop_df['Territorio'].apply(lambda x: x.lower().
    replace(" ", ""))
orderedPROV = pop_df['Territorio'].tolist()
popPROV = dict(zip(pop_df['Territorio'], pop_df['Value']))
nh = np.array([popPROV[prov] for prov in orderedPROV])

Whk = pd.read_csv('Data/FinalForCommuting/A_adj_province.csv', sep=";"
    , index_col=0)
Whk.index = Whk.index.to_series().apply(lambda x: x.lower().replace("
    ", ""))
Whk.columns = Whk.columns.to_series().apply(lambda x: x.lower().
    replace(" ", ""))
Whk = Whk.loc[orderedPROV, orderedPROV].to_numpy()
np.fill_diagonal(Whk, 0)
Whk = Whk / Whk.sum(1).reshape(-1,1)

#df_nazionali = pd.read_csv("path_to_national_data.csv")
df_nazionali = prova[prova["regione"] == 'italia']
df_nazionali = df_nazionali[["anno", "settimana", "incidenza"]]
# df_nazionali: colonne: anno, settimana, incidenza (nuovi casi per
    1000 persone)
df_nazionali = df_nazionali.sort_values(["anno","settimana"])

p = parameters(nh, Whk)
p.Gamma = 0.1  # fissiamo gamma


anno_iniziale, settimana_iniziale = df_nazionali[['anno','settimana'
    ]].values[0]
first_date = date.fromisocalendar(anno_iniziale, settimana_iniziale,
    1)
first_date_str = first_date.strftime("%d-%m-%Y")
start_date = datetime.strptime(first_date_str, "%d-%m-%Y")



initial_case_i, initial_case_e = get_initial_cases_for_week(
    df_dati_provinciali, anno_iniziale, settimana_iniziale,
    orderedPROV, nu=0.22)
initialN=0; initialR=0
S, E, I, N, R, current_date = initialStatesMemory(p, initial_case_i,
    initial_case_e, first_date_str, initialN, initialR)

previous_posterior_samples = None
gamma_fixed = 0.1
sequenza_settimanale = df_nazionali[['anno','settimana']].values
num_settimane = len(sequenza_settimanale)

#queste liste verranno usate per salvare media e intervalli di
    confidenza per i parametri
lam_means, nu_means, beta_means = [], [], []
```

```
lam_lower, lam_upper = [], []
nu_lower, nu_upper = [], []
beta_lower, beta_upper = [], []

# definiamo liste per salvare incidenza simulata e reale
real_incidence = []  # Incidenza reale per 1000 persone (dai dati)
simulated_means = []  # Media dell'incidenza simulata per 1000 persone
simulated_lower = []  # Limite inferiore del 95% CI
simulated_upper = []  # Limite superiore del 95% CI


for week_index in range(num_settimane):
anno_corrente, settimana_corrente = sequenza_settimanale[week_index]
data_osservata_per_1000 = df_nazionali[
(df_nazionali['anno'] == anno_corrente) &
(df_nazionali['settimana'] == settimana_corrente)
]['incidenza'].values[0]

# debug prima di MCMC
print(f"--- Settimana {week_index+1} ---")
print(f"Anno: {anno_corrente}, Settimana: {settimana_corrente}")
print("Data simulazione attuale:", current_date.strftime('%d-%m-%Y'))
print("Stati iniziali della settimana:",
"S:", S.sum(), "E:", E.sum(), "I:", I.sum(), "N:", N.sum(), "R:", R.
    sum())
print(f"Incidenza reale per 1000 persone: {data_osservata_per_1000}")

# definiamo la prior
if previous_posterior_samples is None:
prior_info = {"type":"weak"}
else:
lam_mean = np.mean(previous_posterior_samples[:,0])
lam_std = np.std(previous_posterior_samples[:,0]) or 0.1
nu_mean = np.mean(previous_posterior_samples[:,1])
nu_std = np.std(previous_posterior_samples[:,1]) or 0.1
beta_mean = np.mean(previous_posterior_samples[:,2])
beta_std = np.std(previous_posterior_samples[:,2]) or 0.1
prior_info = {
    "type":"normal",
    "lam_mean": lam_mean, "lam_std": lam_std,
    "nu_mean": nu_mean, "nu_std": nu_std,
    "beta_mean": beta_mean, "beta_std": beta_std
}

def logpost_fn(theta):
return log_posterior(theta, data_osservata_per_1000, S, E, I, N, R, p,
     prior_info, gamma_fixed=gamma_fixed, nh=nh)

n_walkers = 40
initial_guess = [7e-3, 0.3, 0.3]
p0 = [initial_guess + 0.5e-3*np.random.randn(3) for i in range(
    n_walkers)]

sampler = emcee.EnsembleSampler(n_walkers, 3, logpost_fn)
```

```
state = sampler.run_mcmc(p0, 400, progress=True)  # quante iterazioni
    di burn-in?
sampler.reset()
sampler.run_mcmc(state, 1000, progress=True)  # catena principale,
    quante iterazioni?
samples = sampler.get_chain(flat=True)
previous_posterior_samples = samples

lam_val = np.mean(samples[:,0])
nu_val = np.mean(samples[:,1])
beta_val = np.mean(samples[:,2])

# Calcoliamo medie e intervalli di confidenza
lam_samples = samples[:, 0]
nu_samples = samples[:, 1]
beta_samples = samples[:, 2]

lam_means.append(np.mean(lam_samples))
lam_lower.append(np.percentile(lam_samples, 2.5))
lam_upper.append(np.percentile(lam_samples, 97.5))

nu_means.append(np.mean(nu_samples))
nu_lower.append(np.percentile(nu_samples, 2.5))
nu_upper.append(np.percentile(nu_samples, 97.5))

beta_means.append(np.mean(beta_samples))
beta_lower.append(np.percentile(beta_samples, 2.5))
beta_upper.append(np.percentile(beta_samples, 97.5))

print("Parametri medi dopo MCMC:",
f"lambda={lam_means[-1]:.2e}, nu={nu_means[-1]:.3f}, beta={beta_means
    [-1]:.3f}")

# adesso, proviamo a fare una simulazione coi parametri medi trovati
    fra tutte le iterazioni
S, E, I, N, R, weekly_infections = simulate_one_week(lam_val, nu_val,
    beta_val, gamma_fixed, p, S, E, I, N, R, days=7)
print("Dopo simulazione settimana:")
if weekly_infections == -1:
print("Parametri medi portano a popolazione non conservata, -inf
    likelihood")
else:
print("Weekly infections:", weekly_infections)
print("Stati finali settimana:",
"S:", S.sum(), "E:", E.sum(), "I:", I.sum(), "N:", N.sum(), "R:", R.
    sum())

# calcoliamo la likelyhood di alcune combinazioni di parametri
    rispetto ai dati reali. Questo serve ad avere una idea di quanto
# sia realistica la distribuzione di parametri che abbiamo ottenuto.
    Una LL molto negativa e prossima a -inf indica che la
# distribuzione attuale di parametri non riesce a simulare bene i dati
     reali
test_params = [
(lam_val*0.8, nu_val, beta_val),
```

```
    (lam_val*1.2, nu_val, beta_val),
    (lam_val, nu_val*1.1, beta_val),
    (lam_val, nu_val, beta_val*0.9)
    ]
    for (lt, nt, bt) in test_params:
    ll = log_likelihood((lt, nt, bt), data_osservata_per_1000, S, E, I, N,
         R, p, gamma_fixed=gamma_fixed, nh=nh)
    print(f"Parametri test: lam={lt}, nu={nt}, beta={bt}, LL={ll}")

    # incidenza reale
    real_incidence.append(data_osservata_per_1000)

    # simuliamo ora con parametri campionati dalla posterior
    incidenza_simulata_settimanale = []  # Lista temporanea per questa
        settimana
    for theta in samples[np.random.choice(len(samples), 300)]:  # 300
        campioni casuali
    lam, nu, beta = theta
    S_f, E_f, I_f, N_f, R_f, weekly_infections = simulate_one_week(
    lam, nu, beta, gamma_fixed, p, S.copy(), E.copy(), I.copy(), N.copy(),
         R.copy()
    )
    if weekly_infections != -1:  # solo se la simulazione è valida
        salviamo i dati ottenuti
    incidenza_per_1000 = (weekly_infections / nh.sum()) * 1000
    incidenza_simulata_settimanale.append(incidenza_per_1000)

    # calcola media e intervalli di confidenza per l'incidenza simulata
    if incidenza_simulata_settimanale:
    simulated_means.append(np.mean(incidenza_simulata_settimanale))
    simulated_lower.append(np.percentile(incidenza_simulata_settimanale,
        2.5))
    simulated_upper.append(np.percentile(incidenza_simulata_settimanale,
        97.5))
    else:
    simulated_means.append(None)
    simulated_lower.append(None)
    simulated_upper.append(None)


    # ora aggiorniamo E e I secondo i dati reali in preparazione per la
        prossima settimana di simulazione
    new_i, new_e = get_initial_cases_for_week(df_dati_provinciali,
        anno_corrente, settimana_corrente, orderedPROV, nu=nu_val)
    eta = p.eta
    E = eta.reshape(-1,1)*new_e
    I = eta.reshape(-1,1)*new_i
    total_pop = eta.reshape(-1,1)*nh.reshape(1,-1)
    S = total_pop - E - I - N - R
    print("Dopo data assimilation:",
    "S:", S.sum(), "E:", E.sum(), "I:", I.sum(), "N:", N.sum(), "R:", R.
        sum())


    current_date += timedelta(days=7)
```

```
    print("Stima sequenziale completata.")
```

Listing A.9: Non-bayesian optimization

```
df_risultati_regionali = df_risultati_regionali.rename(columns={"
    incidenza": "incidenza_simulata"})
prova = prova.rename(columns={"incidenza": "incidenza_reale"})

df_confronto = pd.merge(
df_risultati_regionali,
prova,
on=["anno","settimana", "regione"],
how="inner"
)

# se vogliamo calcolare l'errore usando solo l'italia
df_confronto = df_confronto[df_confronto["regione"] == 'italia']

# Debug
print("Dati uniti:")
print(df_confronto.head())

#aggiungiamo colonna con la differenza assoluta tra i dati simulati e
    reali
df_confronto["errore_assoluto"] = abs(df_confronto["incidenza_simulata
    "] - df_confronto["incidenza_reale"])

# calcoliamo vare metriche di errore per ogni combinazione di
    parametri
errori_per_parametri = (
df_confronto.groupby(["Lambda", "Nu", "Beta", "Gamma"]).apply(
lambda x: pd.Series({
    "MAE": (x["errore_assoluto"]).mean(),
    "RMSE": ((x["errore_assoluto"] ** 2).mean()) ** 0.5,
    "MedAE": x["errore_assoluto"].median(),
    "R2": 1 - ((x["incidenza_simulata"] - x["incidenza_reale"]) ** 2).
        sum() /
    ((x["incidenza_reale"] - x["incidenza_reale"].mean()) ** 2).sum()
})
).reset_index()
)

# ordiniamo le combinazioni di parametri in base all'errore MAE
errori_ordinati = errori_per_parametri.sort_values("MAE")

# stampiamo la top 10
print("Top 10 combinazioni di parametri per MAE:")
print(errori_ordinati.head(10))


errori_ordinati.to_csv("errori_per_parametri.csv", index=False)

# per ogni metrica di errore stampiamo la combinazione migliore
print("\nMigliori combinazioni per ogni metrica:")
print("Per MAE:")
print(errori_ordinati.loc[errori_ordinati["MAE"].idxmin()])
print("\nPer RMSE:")
```

```
print(errori_ordinati.loc[errori_ordinati["RMSE"].idxmin()])
print("\nPer MedAE:")
print(errori_ordinati.loc[errori_ordinati["MedAE"].idxmin()])
print("\nPer R quadro:")
print(errori_ordinati.loc[errori_ordinati["R2"].idxmax()])




df_confronto.to_csv("confronto_simulato_vs_reale.csv", index=False)
```

Listing A.10: Reading Influcast data

```
'''
Lettura dati reali CON approssimazione nazionale per le province
    mancanti
'''
w = 5
y = 5
import os
import pandas as pd


folder_path = "C:\\Users\\celin\\OneDrive\\Desktop\\polito\\drive-
    download-20241207T142618Z-001\\codice tesi\\dati_reali_24
    -25\\202{}_{:02}".format(y, int(w))
print(folder_path)


weekly_cases_by_region = {}
latest_files = {}

for file_name in os.listdir(folder_path):
if file_name.endswith('.csv'):
base_name = file_name[:-4]
region_name, year_week, target = base_name.split('-')
year, week = map(int, year_week.split('_'))

if region_name not in latest_files or (year, week) > (latest_files[
    region_name]['year'], latest_files[region_name]['week']):
latest_files[region_name] = {'file_name': file_name, 'year_week':
    year_week, 'year': year, 'week': week}

for region_name, file_info in latest_files.items():
file_name = file_info['file_name']
file_path = os.path.join(folder_path, file_name)
print(f"Elaborazione del file per la regione: {region_name}, settimana
    : {file_info['week']} del {file_info['year']}")

df = pd.read_csv(file_path)

if region_name not in weekly_cases_by_region:
weekly_cases_by_region[region_name] = {}

for _, row in df.iterrows():
week = f"{int(row['anno'])}_W{int(row['settimana']):02d}"
cases = row['incidenza']
weekly_cases_by_region[region_name][week] = cases

# Debug: Stampa il dizionario settimanale dei casi
print("Dati settimanali iniziali per regione:", weekly_cases_by_region
    )


week_of_interest = "202{}_W{:02}".format(y, int(w))
```

```
# carichiamo l'incidenza reale per ogni regione
national_file = os.path.join(folder_path, "italia-202{}_{:02}-ILI.csv"
    .format(y, int(w)))
national_df = pd.read_csv(national_file)
incidenza_nazionale = national_df['incidenza'].iloc[-1]  # Ultima
    settimana

for region_name, weeks_data in weekly_cases_by_region.items():
# se influcast non ci ha inviato dati per quella regione
if not weeks_data:
print(f"Attenzione: Nessun dato disponibile per {region_name}. Verrà
    usata l'incidenza nazionale.")
weekly_cases_by_region[region_name] = {week_of_interest:
    incidenza_nazionale}
continue

# se per quella settimana non ci sono dati
if week_of_interest not in weeks_data:
print(f"Attenzione: Nessun dato per la settimana {week_of_interest} in
     {region_name}. Verrà usata l'incidenza nazionale.")
weekly_cases_by_region[region_name][week_of_interest] =
    incidenza_nazionale

print("Dati settimanali aggiornati per regione:",
    weekly_cases_by_region)

def normalize_region_name(region_name):
normalized_name = region_name.lower().replace("_", " ").strip()
return normalized_name

region_name_mapping = {
    "provincia autonoma di bolzano": "P.A. Bolzano",
    "provincia autonoma di trento": "P.A. Trento",
    "valle d'aosta": "valle d aosta",
    "emilia-romagna": "emilia romagna",
    "italia": "Italia",
    "pa trento": "trentino"
}

pop_df = pd.read_csv("popREG2.csv")
pop_df['Regione'] = pop_df['Regione'].apply(normalize_region_name)
pop_df['Regione'] = pop_df['Regione'].replace(region_name_mapping)
pop_dict = pop_df.set_index('Regione')['PopolazioneRegionale'].to_dict
    ()

normalized_weekly_cases_by_region = {
    normalize_region_name(region_name): weeks_data
    for region_name, weeks_data in weekly_cases_by_region.items()
}
normalized_weekly_cases_by_region = {
    region_name_mapping.get(region_name, region_name): weeks_data
    for region_name, weeks_data in normalized_weekly_cases_by_region.
        items()
}
```

```
#calcola i casi totali provinciali
for region_name, weeks_data in normalized_weekly_cases_by_region.items
    ():
if region_name not in pop_dict:
continue

population_region = pop_dict[region_name]
for week, incidence_per_1000 in weeks_data.items():
estimated_cases = incidence_per_1000 * population_region / 1000
normalized_weekly_cases_by_region[region_name][week] = estimated_cases

#distribuzione provinciale dei casi
final_df = pd.DataFrame()
for region_name, weekly_cases in normalized_weekly_cases_by_region.
    items():
region_df = pop_df[pop_df['Regione'].str.strip().str.lower() ==
    region_name.strip().lower()]
if region_df.empty:
continue

for week, total_cases in weekly_cases.items():
region_df['CasiDistribuiti'] = region_df['rapporto'] * total_cases
region_df['Settimana'] = week
final_df = pd.concat([final_df, region_df[['Territorio', 'Settimana',
    'CasiDistribuiti']]], ignore_index=True)

#Salva il file finale
final_df.to_csv('casi_distribuiti_province.csv', index=False)
```

Bibliography

[1] Dati coperture vaccinali influenza, https://www.salute.gov.it/portale/influenza.

[2] Yu Bai and Xiaonan Tao. Comparison of covid-19 and influenza characteristics. *Journal of Zhejiang University-SCIENCE B Biomedicine and Biotechnology*, 2021.

[3] Kissling Esther, Maurel Marine, Emborg Hanne-Dorthe, Whitaker Heather, McMenamin Jim, Howard Jennifer, Trebbien Ramona, Watson Conall, Findlay Beth, Pozo Francisco, Bolt Botnen Amanda, Harvey Ciaran, and Rose Angela. Interim 2022/23 influenza vaccine effectiveness: six european studies, october 2022 to january 2023. *Euro Surveill. 2023;28(21):pii=2300116*, 2023.

[4] Carrat F, Vergu E, Ferguson NM, Lemaitre M, Cauchemez S, Leach S, and Valleron AJ. Time lines of infection and disease in human influenza: A review of volunteer challenge studies. *American Journal of Epidemiology*, 2007.

[5] Parino F., Zino L., Porfiri M., and Rizzo A. Modelling and predicting the effect of social distancing and travel restrictions on covid-19 spreading. *J. R. Soc. Interface 18: 20200875.*, 2021. Distributed under the Attribution 4.0 International license. No changes where made to the cited material unless specified. https://creativecommons.org/licenses/by/4.0/

[6] What are the strategies for addressing Overfitting in neural networks? https://aiml.com/what-are-some-options-to-address-overfitting-in-neural-networks/

[7] Stefania Fiandrino, Andrea Bizzotto, Giorgio Guzzetta, Stefano Merler, Federico Baldo, Eugenio Valdano, Alberto Mateo-Urdiales, Antonino Bella, Francesco Celino, Lorenzo Zino, Alessandro Rizzo, Yuhan Li, Nicola Perra, Corrado Gioannini, Paolo Milano, Daniela Paolotti, Marco Quaggiotto, Luca Rossi, Ivan Vismara, Alessandro Vespignani, and Nicolo Gozzi. Collaborative forecasting of influenza-like illness in italy: the influcast experience. *medRxiv*, 2024. Distributed under the Attribution-NonCommercial-NoDerivatives 4.0 International license. No changes where made to the cited material unless specified. https://creativecommons.org/licenses/by-nc-nd/4.0/

[8] William Ogilvy Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proc. R. Soc. Lond. A115700 721*, 1927.

[9] Predizioni-Epidemiologiche-Italia, Influcast https://github.com/Predizioni-Epidemiologiche-Italia/Influcast

[10] Respicast background https://respicast.ecdc.europa.eu/background/

[11] Influcast serie temporale incidenza per 1000 assistiti https://influcast.org/page/it/insight/

[12] Risultati del Censimento permanente della popolazione https://www.istat.it/statistiche-per-temi/censimenti/popolazione-e-abitazioni/risultati/