Comparative Bayesian Analysis of Epidemiological Data in Disease Outbreaks

Varun V. Datta Dr. Kenneth Butler

Table of contents

Introduction

Bayesian inference has significantly advanced in recent years, propelled by developments in **Markov Chain Monte Carlo (MCMC)** sampling algorithms and the proliferation of **probabilistic programming**. This approach has revolutionized epidemiology by enabling the integration of prior knowledge into the analysis, shifting the role of researchers from mere observers to **active participants** in the inference process. This integration enriches the analysis and tailors it more closely to specific contexts, leveraging the accumulated expertise within the field to achieve nuanced insights(Garcia & Abad, 2014).

The onset of the COVID-19 pandemic further highlighted the need for versatile statistical models capable of handling complex and uncertain data. Bayesian models, particularly those not limited to exponential families or normal distributions, have proven invaluable. They adapt flexibly to the dynamic nature of outbreak data, offering deep and actionable insights into disease spread and informing public health responses.

As emphasized in existing literature, Bayesian methods transform researchers' experiences and prior knowledge into crucial components of data analysis, enhancing both the depth and applicability of epidemiological research. This transformative ability is further evidenced by Gelman's contributions, notably his work on the No-U-Turn sampler and Bayesian data analysis, which underscore the robustness of Bayesian methods in handling complex epidemiological data. These methodologies provide a powerful framework for analyzing outbreaks, demonstrating their applicability and effectiveness in real-world scenarios (Hoffman & Gelman, 2014; Gelman et al., 2013).

References

- Garcia, L. P., & Abad, F. J. (2014). Bayesian inference in public health: A Latin American perspective. Ciência & Saúde Coletiva, 30(4), 703-714. Available at https://www.scielosp.org/article/csp/2014.v30n4/703-714/.
- Hoffman, M. D., & Gelman, A. (2014). The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. Journal of Machine Learning Research, 15(1), 1593-1623.
- Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). Bayesian Data Analysis (3rd ed.). Chapman and Hall/CRC.

Objective

The primary objective of this study is to employ Bayesian statistical methods to analyze epidemiological data from significant disease outbreaks, specifically COVID-19 and influenza. This research aims to:

- **Model Disease Transmission Dynamics**: Utilize Bayesian models to describe the transmission mechanisms of COVID-19 and influenza, capturing the complexity and variability of these disease outbreaks.
- **Estimate Epidemiological Parameters**: Determine crucial parameters such as the basic reproduction number (R_0), effective reproduction number (R_t), infection rate, incubation period, and recovery rates. These parameters are essential for understanding the spread and control of the diseases.
- **Utilize Probabilistic Frameworks**: By integrating Stan and R, this study leverages the probabilistic nature of Bayesian inference to manage uncertainties inherent in outbreak data, providing robust estimates that incorporate prior knowledge and real-time data adjustments.
- **Compare Methodological Approaches**: Contrast the Bayesian estimates with those derived from classical or frequentist statistical methods. This comparison aims to evaluate the robustness and efficiency of Bayesian approaches in real-world epidemiological scenarios, thereby showcasing the potential benefits and limitations of both statistical paradigms.

By focusing on these objectives, the study intends to provide a comprehensive analysis that enhances the understanding of infectious disease dynamics and supports the development of effective public health responses.

Background

What is Bayesian Inference?

Bayesian inference is the process of fitting a probability model to a data set in order to summarize the results by a probability distribution on the parameters of the model and on unobserved quantities such as predictions for new observations. ("Bayesian Data Analysis" by Gelman et al)

Before we divide into fitting a probability model and getting the ouput of this probability distribution that takes care of everything we want to understand or predict, lets define and explain a few Bayesian inference terms that will be used through this study.

Bayesian Inference Fundamentals

The backbone of Bayesian inference or the cornerstone of Bayesian Inference is the **Bayes' Theorem**

Bayes' Theorem: (citation)

$$P(\theta|data) = \frac{P(data|\theta) \cdot P(\theta)}{P(data)}$$

Where:

- $P(\theta|data)$ is the posterior probability of parameter θ given the data.This is what we aim to compute in Bayesian analysis.
- $P(data|\theta)$ is the liklihood: how likely the observed data is under different hypothesis and parameter values.It is the weight given to different parameter values based on how well they explain the observed data.
- $P(\theta)$ is the prior probability of the parameter before seeing the data. It encapsulates our beliefs about the values of θ before seeing the data. In epidemiology, priors can stem from previous studies, expert opinion, or established biological reasoning.
- $P(data)$ is the evidence term or the marginal likelihood that acts as a normalizing constant: ensures that the posterior distribution integrates to 1 or posterior probabilities add to 1. This term is particularly crucial when comparing models, as it integrates over all possible parameter values, weighting them by their likelihood and prior probability.

References:

- Fleet, David (2024). Bayesian Methods. CSCC11 Course Notes. Retrieved from https://www.cs.toronto.edu/~fleet/courses/C11/notesReadings.html.

- Robert, C. P. (2007). The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation (2nd ed.). Springer.

Bayesian vs Frequentist Inference

Definition of Probability

-Bayesian: Probability is a measure of belief or certainty about an event, which updates as new evidence is presented.

-Frequentist: Probability is the long-run frequency of events derived from repeated experiments, avoiding subjectivity.

Handling of Parameters

-Bayesian: Treats parameters as random variables, incorporating prior distributions to reflect prior knowledge or beliefs.

-Frequentist: Treats parameters as fixed but unknown quantities, basing inference solely on the current data without priors.

Incorporation of Prior Information

-Bayesian: Formally includes prior information through prior distributions, useful when historical data or expert knowledge exists.

-Frequentist: No formal mechanism for including prior knowledge; the analysis relies only on data from the current study.

Interpretation of Results

-Bayesian: Results are expressed as a probability distribution (posterior), providing a direct probabilistic interpretation of parameters and credible intervals.

-Frequentist: Uses confidence intervals and p-values, related to the long-run frequency of observing such data under the null hypothesis, not probabilities of hypotheses.

Analysis Flexibility

-Bayesian: Allows for sequential updating of beliefs as new data arrives.

-Frequentist: Typically lacks formal mechanisms for updating estimates with new data without new analysis.

Computational Complexity

-Bayesian: Historically more computationally intensive due to methods like Markov Chain Monte Carlo (MCMC), but recent advancements have improved accessibility.

-Frequentist: Generally simpler computationally, particularly for large samples, though this gap is narrowing with new technologies.

Importance of Bayesian Inference in Epidemiology

In epidemiology, Bayesian methods are invaluable for:

-**Modeling Disease Spread:** Estimating transmission rates and other crucial parameters which often cannot be directly observed.

-**Incorporating Uncertainty:** Handling various sources of uncertainty, such as reporting biases or incomplete data, more explicitly than traditional frequentist methods.

-**Updating Beliefs:** Adapting predictions as new data becomes available, which is crucial in real-time during an outbreak.

The Bayesian framework's ability to merge prior information with new data allows for more nuanced and dynamic interpretations of epidemiological data, making it a powerful tool for public health decision-making.

By using Bayesian inference, epidemiologists can not only estimate the parameters but also quantify the uncertainty of these estimates, providing a more comprehensive understanding of the underlying health phenomena.(Broemeling, 2013)

References:

- Broemeling, L. D. (2013). Bayesian Methods in Epidemiology. Chapman & Hall/CRC Biostatistics Series. Chapman and Hall/CRC.

The 3 steps for Bayesian Data Analysis

In order to fit the probability model on any data and get a resulting probability distribution on everything you want to work with there are 3 high level steps which serve as the blue print for the whole process.

1. Setting up a full probability model— a joint probability distribution for all observable and unobservable quantities in a problem. The model should be consistent with knowledge about the underlying scientific problem and the data collection process.

2. Conditioning on observed data: calculating and interpreting the appropriate posterior distribution—the conditional probability distribution of the unobserved quantities of ultimate interest, given the observed data.

3. Evaluating the fit of the model and the implications of the resulting posterior distribution: how well does the model fit the data, are the substantive conclusions reasonable, and how sensitive are the results to the modeling assumptions in step 1? In response, one can alter or expand the model and repeat the three steps.

References:

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). Bayesian Data Analysis (3rd ed.). Chapman and Hall/CRC.

We will be utilizing two epidemiological models in this study: the Susceptible-Infectious-Recovered (SIR) model and its expansion, the Susceptible-Exposed-Infectious-Recovered (SEIR) model.

SIR model

The Susceptible-Infectious-Recovered (SIR) model to analyze the spread and control of infectious diseases within a population. The Susceptible-Infected-Recovered (SIR) model splits the population in three time-dependent compartments: the susceptible, the infected (and infectious), and the recovered (and not infectious) compartments. When a susceptible individual comes into contact with an infectious individual, the former can become infected for some time, and then recover and become immune. The dynamics can be summarized graphically:

The temporal dyanmics of the subsets of the total population in each of these compartments are established by the following system of Ordinary Differential Equations(ODEs):

$$\begin{aligned}\frac{dS}{dt} &= -\beta S\frac{I}{N}\\ \frac{dI}{dt} &= \beta S\frac{I}{N} - \gamma I\\ \frac{dR}{dt} &= \gamma I\end{aligned}$$

where

- $S(t)$ is the number of people susceptible to becoming infected (no immunity),
- $I(t)$ is the number of people currently infected (and infectious),
- $R(t)$ is the number of recovered people (we assume they remain immune indefinitely),
- β is the constant rate of infectious contact between people,
- γ the constant recovery rate of infected individuals.

Understanding the ODEs

The proportion of infected people among the population is $\frac{I}{N}$. At each time step, given uniform contacts, the probability for a susceptible person to become infected is thus $\beta\frac{I}{N}$, with β the average number of contacts per person per time, multiplied by the probability of disease transmission when a susceptible and an infected subject come in contact.

Hence, at each time step, $\beta S\frac{I}{N}$ susceptible individuals become infected, meaning $\beta S\frac{I}{N}$ people leave the S compartment and $\beta S\frac{I}{N}$ people enter the I compartment.

Similarly, the recovery of an infected individual is taking place at rate γ, and thus the number of infected individuals decreases with speed γI while the number of recovered grows at the same speed.

SEIR Model

We transform the SIR model into a SEIR model, where people are Exposed before being infected. We suppose that Exposed people are not contagious, whereas Infectious people are. Furthermore, we suppose that people are reported as soon as they become Infectious. We add a parameter a, where $\frac{1}{a}$ corresponds to the average time between being infected and becoming infectious (for simplicity we also use $\frac{1}{a}$ as the time between being infected and being reported).

SEIR equations become:

$$\begin{aligned}
\frac{dS}{dt} &= -\beta S \frac{I}{N} \\
\frac{dE}{dt} &= \beta S \frac{I}{N} - aE \\
\frac{dI}{dt} &= aE - \gamma I \\
\frac{dR}{dt} &= \gamma I
\end{aligned}$$

Understanding the SEIR ODEs and transitions

In the SEIR model, there is an additional compartment for exposed individuals (E). The steps are as follows:

1. Susceptible individuals (S) become exposed (E) at the rate of $\beta S \frac{I}{N}$.
2. Exposed individuals (E) become infected (I) at rate σ, representing the incubation period of the disease.
3. Infected individuals (I) recover at rate γ.

Therefore, the key differences and additions in the SEIR model are: - An additional compartment E for exposed individuals. - A new transition rate σ from exposed (E) to infected (I).

The transitions are:

- $S \to E$ at rate $\beta S \frac{I}{N}$.
- $E \to I$ at rate σE.
- $I \to R$ at rate γI.

Assumptions and conditions for SIR and SEIR models

- Births and Deaths are not contributing to the dynamics and the total population $N = S + I + R$ remains constant,
- Recovered individuals do not become susceptible again over time.
- The infection rate β and recovery rate γ are constant.
- The population is homogeneous
- Individuals meet any other individual uniformly at random (homogeneous mixing) and recovery time follows an exponential distribution with mean $\frac{1}{\gamma}$.
- Replacing the integer number of people in each compartement by a continuous approximation is legitimate (the population is big enough).

Using the Bayesian approach on SIR and SEIR Models

By applying Bayesian inference to the SIR model, we aim to estimate the transmission rate and recovery rate, and incorporate prior information and new data to dynamically update our understanding of the epidemic. This approach will allow us to quantify uncertainty in our estimates and make probabilistic predictions about the future course of the epidemic, thus providing valuable insights for public health interventions.

Computational Tools and Techniques for Bayesian Analysis

This study utilizes the R STAN library to design SIR (Susceptible-Infectious-Recovered) and SEIR (Susceptible-Exposed-Infectious-Recovered) models. By leveraging Bayesian inference, we fit these models to epidemiological data using informative and convenience priors for the parameters. The Bayesian approach allows for the incorporation of prior knowledge and uncertainty, making it a powerful tool for understanding and predicting the spread of infectious diseases.

STAN

We will be utilizing a probabalistic framework provided to us by a language STAN. We are able to utilize stan with help of an interface within R.

R has a library Rstan (cite) which helps us compile and run stan models on any data that we load into R.

STAN is a probabilistic programming language for specifying statistical models and performing Bayesian inference. It is particularly well-suited for complex models involving latent variables

and hierarchical structures. In this study, STAN is used to implement SIR and SEIR models, fitting them to real-world data through Bayesian methods. STAN employs Markov Chain Monte Carlo (MCMC) and Hamiltonian Monte Carlo (HMC) techniques to efficiently sample from the posterior distributions of model parameters. The flexibility of STAN allows for the integration of ordinary differential equations (ODEs) to describe the dynamic behavior of epidemiological models.

References:

Hoffman, M. D., & Gelman, A. (2014). The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. Journal of Machine Learning Research, 15(1), 1593-1623.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). Bayesian Data Analysis (3rd ed.). Chapman and Hall/CRC.

Markov Chain Monte Carlo (MCMC)

Markov Chain Monte Carlo (MCMC) is a class of algorithms used to sample from probability distributions when direct sampling is challenging. In the context of this study, MCMC is employed to estimate the posterior distributions of the SIR and SEIR model parameters. By generating a sequence of samples that approximate the target distribution, MCMC provides a practical approach to Bayesian inference, enabling the estimation of model parameters and their uncertainties.

Hamiltonian Monte Carlo (HMC)

Hamiltonian Monte Carlo (HMC) is an advanced MCMC method that uses Hamiltonian dynamics to propose new samples. HMC reduces the correlation between successive samples, improving the efficiency of the sampling process. This method is particularly effective for high-dimensional and complex models, such as the SIR and SEIR models used in this study. By leveraging HMC, STAN can explore the parameter space more effectively, leading to more accurate and reliable posterior estimates.

ODE Solvers: RK45 and Backward Differentiation

Ordinary differential equations (ODEs) are integral to modeling the dynamics of infectious diseases. The RK45 (Runge-Kutta) method and backward differentiation are two numerical techniques used to solve ODEs within STAN. RK45 is an explicit solver that provides a good balance between accuracy and computational efficiency, while backward differentiation is an implicit method suitable for stiff ODE systems. In this study, these solvers are used to simulate

the progression of the disease over time, capturing the complex interactions between different compartments of the epidemiological models.

Basics of STAN Sampling and Hyperparameters

STAN's sampling process involves initializing the model parameters and then using MCMC specifically HMC to generate samples from the posterior distribution. The efficiency and accuracy of this process depend on the choice of hyperparameters, such as step size and number of iterations. Proper tuning of these hyperparameters is crucial for obtaining reliable results. In this study, we carefully select and tune hyperparameters to ensure the convergence and stability of the sampling process, enabling robust Bayesian inference for the SIR and SEIR models.

Exploring Hamiltonian Monte Carlo (HMC) and MCMC

Imagine you are tasked with mapping a complex landscape that represents the probability distribution of model parameters. This landscape has hills (peaks in the probability distribution) and valleys (troughs), and you need to cover it as thoroughly as possible.

One can attempt this task with Traditional MCMC and HMC methods

Traditional MCMC Methods

- Analogy: Walking through this landscape on foot.

-Characteristics: You move randomly, which can be slow and inefficient. This random walk behavior results in high autocorrelation between successive samples and slow convergence, akin to frequently retracing your steps in the same small area before moving on.

- Random Walk: Your movement is somewhat random. You decide the next step based on a combination of your current location and some randomness. This is similar to the Metropolis-Hastings algorithm, a specific type of MCMC.
- Local Decisions: Each step is decided locally, meaning you only look at the immediate surroundings to decide where to go next. Over time, you hope to cover the entire mountain range, building a picture of its overall shape.
- Slow Exploration: This process can be slow because you're often taking small steps and can get stuck in valleys or spend a lot of time on ridges, making it inefficient for exploring large or complex terrains.

Hamiltonian Monte Carlo (HMC)

- Analogy: Riding a bicycle through the terrain.
- Characteristics: The bicycle's momentum helps you glide smoothly over hills and valleys, allowing you to cover more ground quickly and efficiently. In HMC, this momentum is derived from Hamiltonian dynamics, which uses gradients (slopes of the terrain) to guide the sampling process, thereby reducing the randomness of movements compared to traditional MCMC.
- Energy-Based Movement: Your movement is governed by energy principles, similar to how physical systems move. You use potential energy (related to the height of the mountain) and kinetic energy (related to your speed) to guide your path.
- Inertia: You have inertia, meaning you can move across the terrain in a more direct and efficient way. This allows you to traverse large areas more quickly and avoid getting stuck in local features.
- Efficient Exploration: This method allows for much faster exploration of the mountain range because you're not limited to local steps. Instead, you can make large, informed leaps across the terrain, giving you a more comprehensive view in less time.

Application in Bayesian Modeling: SIR and SEIR Models

When using Stan to simulate the posterior distribution of parameters in epidemiological models like SIR (Susceptible, Infected, Recovered) or SEIR (Susceptible, Exposed, Infected, Recovered), HMC provides distinct advantages:

- Efficient Sampling: By leveraging both the current position (current parameter values) and momentum (direction and speed of the proposed moves), HMC navigates the parameter space more effectively than traditional MCMC. This is particularly useful in complex models where parameters interact in non-linear ways, as often seen in epidemiological models.
- Reduced Autocorrelation: The cycling analogy helps us visualize how using momentum allows HMC to escape local optima and explore more of the parameter space without getting trapped. This leads to samples that are less correlated, providing a more varied and representative set of parameter values from the posterior.

Practical Example:

- Model Setup: Consider an SIR model implemented in Stan, starting with a population of 100 individuals, where one individual is initially infected.

- Parameter Estimation: We set priors for the infection rate (β) and recovery rate (γ), perhaps based on historical data or expert opinion. Stan uses HMC to draw samples from the posterior distribution of these parameters, efficiently exploring their feasible values given the observed data.
- Convergence and Diagnostics: As the HMC algorithm runs, it generates multiple chains that explore the parameter space. The efficiency of HMC helps these chains to converge quickly to the true parameter values, providing robust estimates and credible intervals.

Simulating a Posterior in Stan Using Real-World Data

When using Stan to simulate a posterior distribution based on real-world data, the choice between MCMC and HMC can significantly impact the efficiency and accuracy of your sampling:

- MCMC (e.g., Metropolis-Hastings): Similar to exploring a mountain range one step at a time, using MCMC methods in Stan involves making local, incremental updates to your parameter estimates. This can be slow and might require many iterations to adequately explore the posterior distribution, especially if the posterior has complex, high-dimensional structures.
- HMC (Hamiltonian Monte Carlo): Stan's default sampler, NUTS (No-U-Turn Sampler), is based on HMC. Using HMC is like gliding over the mountain range, leveraging the geometry of the parameter space to make informed, efficient moves. This results in faster convergence and more effective exploration of the posterior distribution, providing more accurate estimates with fewer iterations.

By using HMC in Stan, we can take advantage of the algorithm's efficiency to handle complex models and large datasets, making it a powerful tool for Bayesian inference in real-world applications.

Conclusion

Using HMC in Stan for simulating posterior distributions in models like SIR or SEIR offers a substantial improvement over traditional MCMC methods. The cycling analogy underscores how momentum and informed directional choices significantly enhance the exploration of complex statistical landscapes, leading to faster convergence and more reliable estimates. This makes HMC an invaluable tool in the arsenal of Bayesian inference, particularly for real-world data where the underlying dynamics are complex and multifaceted.

Analysis and Results:

To establish best practices for using STAN with epidemiological data, extensive literature searches were conducted. This allowed us to set a baseline approach and understand optimal methods for writing models in STAN.

A key resource was the https://mc-stan.org/users/documentation/case-studies/boarding_school_case_study.ht : **A STAN guide to inference with epidimeology**, which utilized data from the 1978 boarding school influenza outbreak and Swiss Covid-19 wave 1 data. This study provided insights into troubleshooting and building informative models from the ground up.

Drawing from this case study and other resources, we employed the following datasets to fit SIR and SEIR models with various modifications:

- Influenza 1978 Boarding School Outbreak
- New York City Covid 19 Wave 1 Data 2020
- New York State Covid 19 Wave 1 Data 2020

Exploring the Datasets

Influenza 1978 Boarding School Outbreak

Background: https://search.r-project.org/CRAN/refmans/epimdr/html/flu.html https://rdrr.io/cran/outbreak

The daily number of children confined to bed in a boarding school in North England during an outbreak in 1978 of the reemerging A/H1N1 strain. The school had 763 boys of which 512 boys were confined to bed sometime during the outbreak.

Source: Anonymous (1978) EPIDEMIOLOGY: Influenza in a boarding school. British Medical Journal, 4 March 1978 p.587.

- **Least Squares Approach**:
  - Fit a deterministic or frequentist approach using least squares on the 1978 boarding school dataset.
  - Least squares is derived from likelihood estimation.
- **SIR Model with Weakly Informative Priors**:
  - Apply an SIR model using weakly informative priors, following the methodology in the case study.
- **SEIR Model Comparison**:
  - Fit an SEIR model to compare the level of insight gained from SIR vs. SEIR models.

New York City 2020 Covid-19 Data (Wave 1)

Background:

- **SIR Model with Weakly Informative Priors**:
  - Utilize the same weakly informative prior-based SIR model as before.
- **Least Squares Optimization**:
  - Apply a least squares optimization approach.
- **Comparison**:
  - Compare the results of the weakly informative prior-based SIR model with the least squares optimization.
- **Advanced SIR Model**:
  - Develop an advanced SIR model with appropriate priors and mechanisms to adjust for real-world factors such as lockdown measures and underreporting.
- **Model Troubleshooting and Diagnostics**:
  - Implement steps for troubleshooting and apply various diagnostic measures to ensure model robustness.

New York State 2020 Covid-19 Data (Wave 1)

Background:

- **Repeat SIR and SEIR Model Comparisons**:
  - Repeat the same modeling approaches used for the NYC data.
- **Scale Comparison**:
  - Compare the scale and results from school (boarding school outbreak) to city (NYC) to state (New York State).

By following these steps, we aimed to develop a comprehensive understanding of how different models and approaches perform across various datasets and scales.

We will be using the following libraries

```r
# CITE ALL OF THEM

library(deSolve)
library(outbreaks)
library(tidyverse)
library(tidybayes)
library(rstan)
library(gridExtra)
rstan_options (auto_write = TRUE)
options (mc.cores = parallel::detectCores ())
```

Building an SIR model

We utilize the concept of a **sampling distribution / liklihood** represented as

$$p(Y \mid \theta)$$

which basically describes the number of cases or our data (Y is generated)is generated given a set of model parameters θ.

Our objective here is the utilize Bayesian Inference Techniques in determining the most plausible parameter values θ based on observed data $\mathcal{Y}$ and we will be using **posterior distribution**: $p(\theta \mid Y)$ to achieve our goal. Bayesian inference updates our beliefs about the parameters after seeing the data, as described by Bayes' rule:

$$p(\theta \mid \mathcal{Y}) \propto p(\mathcal{Y} \mid \theta) \times p(\theta)$$

Here, $p(\mathcal{Y} \mid \theta)$ is the sampling distribution, $p(\theta)$ is the **prior distribution** which encodes what we know about the parameters before seeing the data, and $\propto$ denotes "proportional to".

Sampling Distribution

In a compartmental model like the SIR model, specific transmission parameters and initial conditions are utilized to predict or simulate the dynamics of an epidemic. The most important prediction any model could make is the number of infected individuals $I_{\text{ODE}}(t)$ at time t. We relate this to actual observed data, such as the number of students reported sick(in the case of 1978 Boarding School Data) $I_{\text{obs}}(t)$.

NOTE: Real world observational data is always noisy

We model the number of students reported sick using a Negative Binomial distribution, which accounts for variability in the data exceeding what would be expected under a simpler Poisson model(Adjusting for overdispersion) :

$$I_{\text{obs}}(t) \sim \text{NegBin}(I_{\text{ODE}}(t), \phi)$$

This formulation gives us $p(Y \mid \theta)$, where θ includes not only the transmission rates (β, γ) but also ϕ, the parameter controlling the degree of dispersion in the Negative Binomial distribution.

ASIDE: Addressing overdispersion within a Poisson Liklihood

Poisson-Gamma Mixture Overdispersion Derivation

To derive the Poisson-Gamma mixture model and show how it accounts for overdispersion, we start by defining the Poisson distribution and the Gamma distribution.

Poisson Distribution

A random variable (Y) follows a Poisson distribution with rate parameter ( ) if its probability mass function (pmf) is given by:

$$P(Y = y \mid \lambda) = \frac{\lambda^y e^{-\lambda}}{y!}, \quad y = 0, 1, 2, ...$$

Gamma Distribution

A random variable Λ follows a Gamma distribution with shape parameter (α) and rate parameter (β) if its probability density function (pdf) is given by:

$$f(\lambda \mid \alpha, \beta) = \frac{\beta^\alpha \lambda^{\alpha-1} e^{-\beta\lambda}}{\Gamma(\alpha)}, \quad \lambda > 0$$

where ($\Gamma(\alpha)$ is the Gamma function.

Poisson-Gamma Mixture

To account for overdispersion in the Poisson distribution, we assume that the rate parameter ( ) itself is a random variable following a Gamma distribution. This leads to the Poisson-Gamma mixture model.

The joint distribution of (Y) *and* Λ) is:

$$P(Y = y, \Lambda = \lambda) = P(Y = y \mid \lambda) f(\lambda \mid \alpha, \beta)$$

Marginalizing over (λ), we obtain the marginal distribution of (Y):

$$P(Y = y) = \int_0^\infty P(Y = y \mid \lambda) f(\lambda \mid \alpha, \beta)\, d\lambda$$

Substituting the pmf of the Poisson distribution and the pdf of the Gamma distribution, we get:

$$P(Y = y) = \int_0^\infty \frac{\lambda^y e^{-\lambda}}{y!} \cdot \frac{\beta^\alpha \lambda^{\alpha-1} e^{-\beta\lambda}}{\Gamma(\alpha)}\, d\lambda$$

Combining terms, we have:

$$P(Y = y) = \frac{\beta^\alpha}{y!\,\Gamma(\alpha)} \int_0^\infty \lambda^{y+\alpha-1} e^{-(\lambda+\beta)\lambda}\, d\lambda$$

Recognizing the integral as the kernel of the Gamma function ($\Gamma(y + \alpha)$), we get:

$$P(Y = y) = \frac{\beta^\alpha}{y!\,\Gamma(\alpha)} \cdot \frac{\Gamma(y + \alpha)}{(\beta + 1)^{y+\alpha}}$$

Simplifying, we obtain the probability mass function of the negative binomial distribution:

$$P(Y = y) = \frac{\Gamma(y + \alpha)}{\Gamma(\alpha)y!} \left(\frac{\beta}{\beta + 1}\right)^\alpha \left(\frac{1}{\beta + 1}\right)^y$$

Overdispersion

The negative binomial distribution naturally incorporates overdispersion. The mean and variance of (Y) are:

$$E(Y) = \frac{\alpha}{\beta}, \quad \text{Var}(Y) = \frac{\alpha}{\beta}\left(1 + \frac{1}{\beta}\right)$$

The variance exceeds the mean, indicating overdispersion.

Thus, the Poisson-Gamma mixture model provides a framework for modeling count data with overdispersion by mixing the Poisson distribution with a Gamma-distributed rate parameter, resulting in a negative binomial distribution.

References:

- Balaji, R., & Czado, C. (n.d.). Overdispersion in Poisson Regression. Lecture notes, University of Colorado. Retrieved from https://civil.colorado.edu/~balajir/CVEN6833/lectures/c-czado-GLM-lectures/lec7-PoissonReg-overdispersion.pdf
- AdamO. (2018, March 7). How to deal with overdispersion in Poisson regression: quasi-likelihood, negative binomial GLM, or subject-level random effect? Cross Validated. Retrieved from https://stats.stackexchange.com/questions/255/negative-binomial-glm-for-overdispersed-poisson-data
- Sakia, R. M. (1992). The Poisson-Gamma mixture model. Journal of Statistical Distributions and Applications, 4, 143-155. Retrieved from https://jsdajournal.springeropen.com/articles/10.1186/015-0046-1

END ASIDE

Prior Distributions

To complete our Bayesian SIR model, we utilize priors for each parameter, $\theta = (\beta, \gamma, \phi)$.

An example is to assume that the recovery rate γ follows a Normal distribution truncated at zero:

$$\gamma \sim N(0.4, 0.5)$$

It is truncated to ensure γ is positive. This reflects a belief that the recovery rate has a high probability of being below 1, meaning most people recover within a day(typical with that strain of infleunza) . These priors can be adjusted as more information becomes available (usually through scientific literature), allowing us to become active participants in our research by updating these prior beliefs to .

Predictions and derived quantities

After fitting the model and obtaining a posterior distribution for θ, we can further derive additional quantities of interests.

We can generate simulated predictions, $ Y_{pred}$, and work out the plausible range of new observations (using the 95% credible intervals) : As these predictions are based on the posterior, $p(\theta \mid Y)$, our simulations account for uncertainty both in the data generating process and in our estimates of θ. Moreover, we compute a posterior distribution of predictions

$$p(Y_{pred} \mid Y) = \int p(Y_{pred}|\theta)p(\theta|Y)d\theta$$

Similarly, we can compute quantities that depend on the model parameters.

We can also evaluate other quantities based on the model parameters. A critical measure is the basic reproduction number, R_0, which quantifies the average number of secondary infections that one infected person will cause in a wholly susceptible population over their infectious period. A value of $R_0 > 1$ implies a potentially widespread epidemic, whereas $R_0 < 1$ indicates that the infection is likely to extinguish over time. Through Bayesian analysis, we also establish a posterior distribution for R_0.

$$p(R_0 \mid Y).$$

Coding SIR models in STAN (Prevelance Model)

Prevalence data reflects the number of infected individuals at a given time.

An ODE takes the form

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y)$$

where y are the *states*, in our example $y = (S, I, R)$, and t is time. We also need an initial condition y_0 at t_0 and the times, τ, at which we evaluate the solution.

To specify an ODE in Stan, we first code f in the `functions` block. This function must observe a strict signature:

```
real[] f(real time, real[] state, real[] theta,
         real[] x_r, int[] x_i)
```

with

- `time`, t;
- `state`, the volumes in each compartment, y;
- `theta`, variables used to compute f, which depend on the model parameters;
- `x_r`, real variables used to evaluate f, which only depend on fixed data;
- `x_i`, integer values used to evaluate f, which only depend on fixed data.

The ODEs for the SIR model is defined as follows:

```
functions {
  real[] sir(real t, real[] y, real[] theta,
             real[] x_r, int[] x_i) {

      real S = y[1];
      real I = y[2];
      real R = y[3];
      real N = x_i[1];

      real beta = theta[1];
      real gamma = theta[2];

      real dS_dt = -beta * I * S / N;
      real dI_dt =  beta * I * S / N - gamma * I;
      real dR_dt =  gamma * I;
```

```
    return {dS_dt, dI_dt, dR_dt};
  }
}
```

We evaluate the solution numerically by using one of Stan's numerical integrators.

We will be using in for the Runge-Kutta 4^{th} /5^{th} order. https://mc-stan.org/docs/2_24/functions-reference/functions-old-ode-solver.html

A call to the integrator is shown below:

```
  y = integrate_ode_rk45(sir, y0, t0, ts, theta, x_r, x_i);
```

with

- `sir`, the name of the function that returns the derivatives, f;
- `y0`, the initial condition;
- `t0`, the time of the initial condition;
- `ts`, the times at which we require the solution to be evaluated;
- `theta`, `x_r`, `x_i`, arguments to be passed to f.

We have now gathered all the elements to solve our systems of ODEs.

Given we are utilizing an SIR model, we assume

- total population remains constant
- the three derivatives $\frac{dS}{dt}$,$\frac{dI}{dt}$, $\frac{dR}{dt}$ sum up to 0

[As mentoined in the background information Assumptions of SIR & SEIR]

We can use this fact to improve computational efficiency of the `sir` function by deriving the value of $\frac{dI}{dt}$ from $\frac{dS}{dt}$ and $\frac{dR}{dt}$

```
    real dS_dt = -beta * I * S / N;
    real dR_dt =  gamma * I;
    real dI_dt =  -(dS_dt + dR_dt);
```

We utilize the `data` block to store our fixed data:

```
data {
  int<lower=1> n_days;
  real y0[3];
  real t0;
  real ts[n_days];
  int N;
  int cases[n_days];
}
```

Subsequently, we utilize the`transformed data` block. In this example, we transform our data to match the signature of `sir` (with `x_r` being of length 0 because we have nothing to put in it).

```
transformed data {
  real x_r[0];
  int x_i[1] = { N };
}
```

Finally, we declare the model parameters.

If you want some parameter to be bounded, and it is not already guaranteed by his prior, you need to specify `<lower=a, upper=b>` when declaring this parameter.

Note: that this is how you put a truncated prior distribution on a parameter.

```
parameters {
  real<lower=0> gamma;
  real<lower=0> beta;
  real<lower=0> phi_inv;
}
```

And then transforms of the parameters

```
transformed parameters{
  real y[n_days, 3];
  real phi = 1. / phi_inv;
  {
    real theta[2];
    theta[1] = beta;
    theta[2] = gamma;

    y = integrate_ode_rk45(sir, y0, t0, ts, theta, x_r, x_i);
```

```
  }
}
```

With all the code above ready the ODE solution, y, is ready, leaving us to define the prior and sampling distribution.

```
model {
  //priors
  beta ~ normal(2, 1); //truncated at 0
  gamma ~ normal(0.4, 0.5); //truncated at 0
  phi_inv ~ exponential(5);

  //sampling distribution
  //col(matrix x, int n) - The n-th column of matrix x. Here the number of infected people
  cases ~ neg_binomial_2(col(to_matrix(y), 2), phi);
}
```

With the inference part of the model complete we can calculate the basic reproduction number, R_0, and predictions for the number of cases in the `generated quantities` block:

```
generated quantities {
  real R0 = beta / gamma;
  real recovery_time = 1 / gamma;
  real pred_cases[n_days];
  pred_cases = neg_binomial_2_rng(col(to_matrix(y), 2) + 1e-5, phi);
}
```

Note: Generated quantities use the results of inference to generate the quantities coded.

The above code is used to generate a stan model called `SIR_Prevelance.stan`

SIR Model (Incidences or Daily counts)

The model described above focuses on representing the prevalence of the disease. However, when dealing with datasets that record the number of daily cases, we shift our focus to incidence. Incidence refers to the number of new cases occurring in a specific period.

To illustrate this, we will fit our basic SIR (Susceptible-Infectious-Recovered) model to COVID-19 data. In previous examples, such as modeling influenza, the data represented the number of students in bed at a given time, which corresponds to disease prevalence. In our new setting with COVID-19, we only have access to the number of new cases reported each day, which constitutes incidence data.

In the SIR model, the incidence of the disease at time `t` is the number of individuals transitioning from the Susceptible compartment to the Infectious compartment at that time. This change requires a modification of our model to reflect the daily incidence instead of cumulative cases.

We add the following code in the `transformed parameters` block:

```
for (i in 1:n_days-1)
  incidence[i] = y[i, 1] - y[i + 1, 1]; //S(t) - S(t + 1)
```

We also modify the negative-binomial likelihood by fitting these incidence parameters to the data:

```
cases[1:(n_days-1)] ~ neg_binomial_2(incidence, phi);
```

This model above will be compiled into a file called `SIR_incidence.stan`

Note: We will use this model in the sectionr relating to covid 19 inference

References: - Stan Development Team. (Leo Grinsztajn, Elizaveta Semenova, Charles C. Margossian, and Julien Riou). Boarding School Case Study. Stan Documentation. Retrieved from https://mc-stan.org/users/documentation/case-studies/boarding_school_case_study.html#stan_codes

https://github.com/charlesm93/disease_transmission_workflow

Building our Least Squares / Non Linear Regression SIR Model

We will be using the `deSolve` library in R to implement our system of ODEs for our SIR model.

SIR Model Function:

This function computes the rates of change for Susceptible (S), Infected (I), and Recovered (R) compartments based on the current state and model parameters.

Model Definition:

- **Define the SIR model dynamics** in the `closed.sir.model` function.
- **Error Function:** Implement the SSE function `sse.sir` that calculates the sum of squared errors between observed cases and model predictions.
- **Model Fitting:** Fit the SIR model to data using the optim function with Nelder-Mead optimization in `fit_and_plot_sir`.

- **Prediction and Plotting:** Generate and plot model predictions alongside actual data for visual comparison.

Sum of Squared Errors (SSE) Function:

```
# Define the SIR model function
closed.sir.model <- function(t, x, params) {
  S <- x[1]
  I <- x[2]
  R <- x[3]
  beta <- params[1]
  gamma <- params[2]
  N <- params[3]  # Total population
  dS <- -beta * S * I / N
  dI <- beta * S * I / N - gamma * I
  dR <- gamma * I
  return(list(c(dS, dI, dR)))
}
```

This function is used to calculate the fit of the model to the actual data by computing the sum of the squared differences between the observed data and the model's predictions.

```
# Sum of Squared Errors (SSE) function
sse.sir <- function(params0, data, N) {
  times <- as.numeric(data$date - min(data$date)) + 1
  cases <- data$cases
  I0 <- cases[1]  # Initial number of infected individuals
  S0 <- N - I0  # Initial number of susceptible individuals
  R0 <- 0  # Initial number of recovered individuals

  # Simulate the model
  out <- as.data.frame(ode(y = c(S = S0, I = I0, R = R0), times = times, func = closed.sir.m
  sse <- sum((out$I - cases)^2)
  return(sse)
}
```

Parameter Optimization:

Uses optimization techniques to find the best-fit parameters (beta and gamma) that minimize the SSE, meaning they provide the best match between the model predictions and actual data.

Simulation and Plotting:

After determining the best-fit parameters, the SIR model is simulated over the observed period to generate predictions. These predictions are then plotted alongside the actual data to visually assess the fit.

```
# Main function to fit SIR model and plot results
fit_and_plot_sir <- function(data, N, initial_params = c(1.5e-3, 0.44)) {
  # Optimization using Nelder-Mead method
  fit <- optim(par = initial_params, fn = function(params) sse.sir(params, data, N), method =

  # Generate predictions using the optimized parameters
  days <- as.numeric(data$date - min(data$date)) + 1
  model_output <- as.data.frame(ode(y = c(S = N - data$cases[1], I = data$cases[1], R = 0), t
 # Print optimized parameters beta and gamma
  cat(sprintf("Optimized beta: %f\nOptimized gamma: %f\n", fit$par[1], fit$par[2]))

  # Plot actual vs. predicted
  ggplot() +
    geom_line(data = model_output, aes(x = days, y = I, colour = "Predicted"), size = 1) +
    geom_point(data = data, aes(x = days, y = cases, colour = "Actual"), size = 2) +
    labs(title = "Actual vs. Predicted Infected Individuals",
         x = "Day from Start",
         y = "Number of Infected Individuals") +
    scale_color_manual(values = c("red", "blue"), labels = c("Actual", "Predicted")) +
    theme_minimal()
}
```

References:

- Choisy, M. (n.d.). SIR Model Simulation. RPubs. Retrieved from https://rpubs.com/choisy/sir
- Moroney, M. (2020, March 19). How to model an epidemic with R. freeCodeCamp. Retrieved from https://www.freecodecamp.org/news/how-to-model-an-epidemic-with-r/
- Bolker, B. (2011). EEID 2011 Simulation. McMaster University. Retrieved from https://ms.mcmaster.ca/~bolker/eeid/2011_eco/EEID2011_Simulation.pdf

Now that we have our base models, let's fit them with the 1978 Influenza Boarding School Data and work on expanding the models from there to initiate our comparative Bayesian analysis to get an idea of Bayesian inference.

1978 Influenza Boarding School (SIR Approach)

```
# Access the dataset
data(influenza_england_1978_school)
flu_data <- influenza_england_1978_school
# Data wrangling

flu_data <- influenza_england_1978_school

flu_data <-flu_data %>%  select(date:in_bed) %>% rename(cases = in_bed )
flu_data
```

```
         date cases
1  1978-01-22     3
2  1978-01-23     8
3  1978-01-24    26
4  1978-01-25    76
5  1978-01-26   225
6  1978-01-27   298
7  1978-01-28   258
8  1978-01-29   233
9  1978-01-30   189
10 1978-01-31   128
11 1978-02-01    68
12 1978-02-02    29
13 1978-02-03    14
14 1978-02-04     4
```

Fitting an SIR model using STAN(Bayesian Approach)

We will be using our prevalence model, which tracks the number of people infected on a given day.

```
model <- stan_model("/Users/varundatta/Desktop/STAD94/Bayesian_Repo/paper/Models/1978 Influe
```

```
# total count of students
N <- 763;
cases <- flu_data$cases
# times / number of days
n_days <- length(cases)
```

```r
t <- seq(0, n_days, by = 1)
t0 = 0
t <- t[-1]

#initial conditions
i0 <- 1
s0 <- N - i0
r0 <- 0
y0 = c(S = s0, I = i0, R = r0)

# data for Stan
data_sir <- list(n_days = n_days, y0 = y0, t0 = t0, ts = t, N = N, cases = cases)

# number of MCMC steps
niter <- 2000
```

```r
fit_sir_negbin <- sampling(model,
                 data = data_sir,
                 iter = niter,
                 chains = 4,
                 seed = 0)
```

Reviewing the Posterior quantiles of our parameters and generated quantities

```r
pars=c('beta', 'gamma', "R0", "recovery_time")
print(fit_sir_negbin, pars = pars)
```

```
Inference for Stan model: anon_model.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

              mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
beta          1.74    0.00 0.06 1.64 1.70 1.73 1.77  1.85  1929    1
gamma         0.54    0.00 0.05 0.46 0.51 0.54 0.57  0.64  2370    1
R0            3.23    0.01 0.29 2.73 3.05 3.21 3.39  3.87  1828    1
recovery_time 1.86    0.00 0.16 1.56 1.76 1.86 1.96  2.19  2147    1

Samples were drawn using NUTS(diag_e) at Mon Jul 29 13:25:29 2024.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Where

`mean`: The average value of the posterior samples for the parameter. It provides an estimate of the central tendency.

`se_mean`: The standard error of the mean, indicating the variability in the estimate of the mean. A lower value suggests more precise estimates.

`sd`: The standard deviation of the posterior samples, reflecting the overall variability in the parameter estimates.

`2.5%`: The 2.5th percentile of the posterior distribution, representing the lower bound of the 95% credible interval. This value indicates that there is a 2.5% chance that the parameter is below this value.

`25%`: The 25th percentile (first quartile) of the posterior distribution. This value indicates that 25% of the posterior samples are below this value.

`50%`: The median (50th percentile) of the posterior distribution, representing the middle value of the samples. It is a robust measure of central tendency.

`75%`: The 75th percentile (third quartile) of the posterior distribution. This value indicates that 75% of the posterior samples are below this value.

`97.5%`: The 97.5th percentile of the posterior distribution, representing the upper bound of the 95% credible interval. This value indicates that there is a 2.5% chance that the parameter is above this value.

`n_eff`: The effective sample size, which indicates the number of independent samples in the posterior. Higher values suggest better mixing and more reliable estimates.

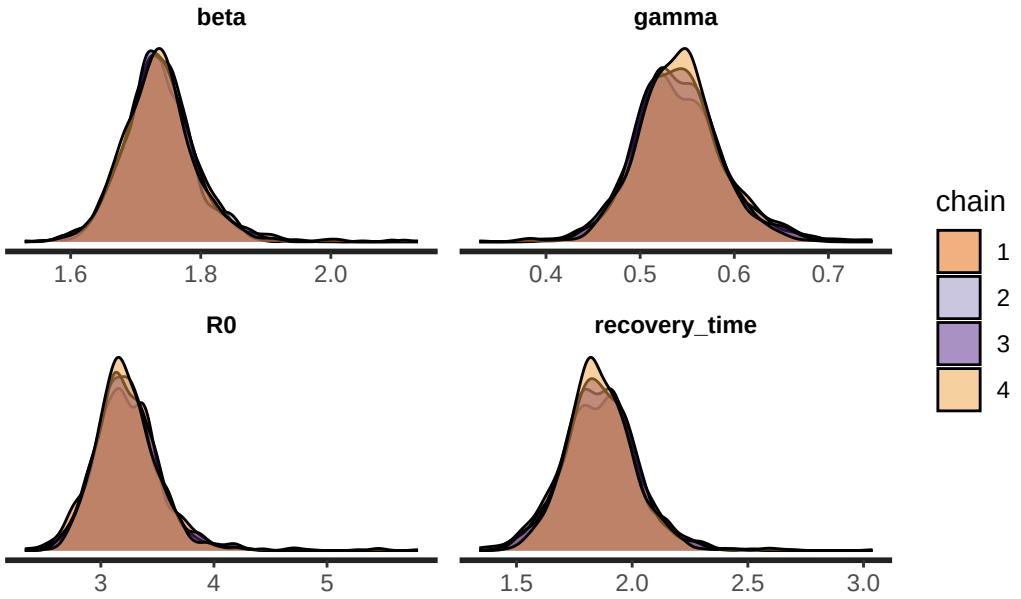
`Rhat`: The potential scale reduction factor, which assesses the convergence of the Markov chains. Values close to 1 indicate good convergence; values significantly greater than 1 suggest potential issues with convergence.

Mixing of markov chains and inference

In Bayesian inference, it is crucial to assess the mixing of Markov chains to ensure that they are converging to the target posterior distribution. Poor mixing can indicate that the chains are not adequately exploring the parameter space, which can result in unreliable estimates.

We use the stan_dens function to visualize the density plots of the parameter estimates from different chains. By examining these plots, we can assess whether the chains are mixing well. If the chains are well-mixed, the density plots for each chain should overlap significantly, indicating that each chain has converged to the same posterior distribution.

```
stan_dens(fit_sir_negbin, pars = pars, separate_chains = TRUE)
```



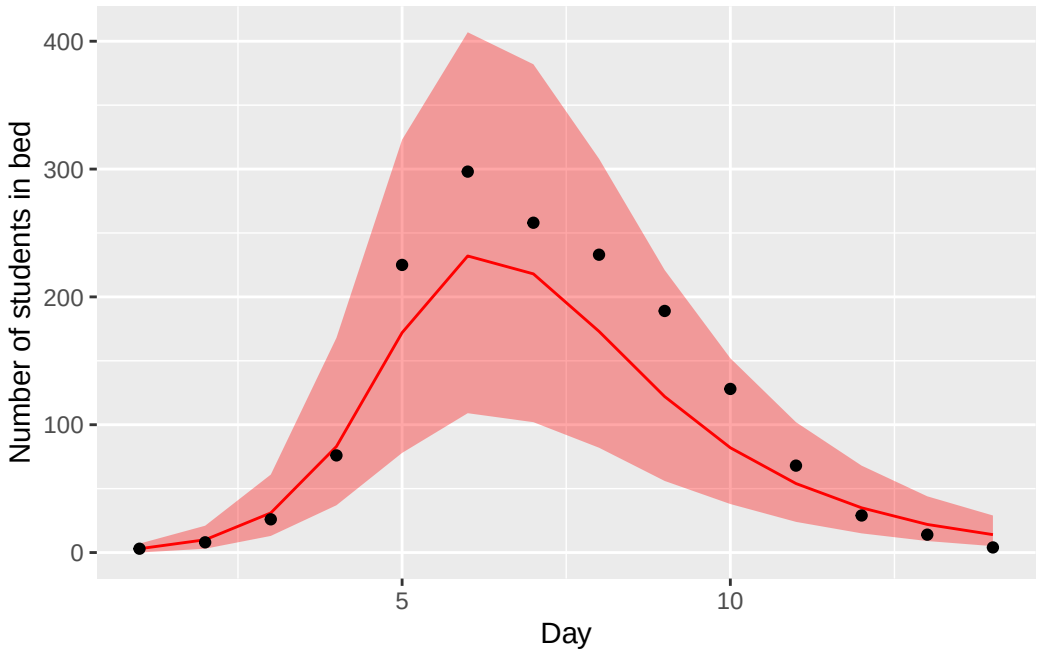
In our case the chains have converged to the same posterior distribution with minor differences in the modes.

Plotting the predicted cases vs actual cases

```
# Create a data frame with summary statistics for the predicted cases
smr_pred <- cbind(as.data.frame(summary(
  fit_sir_negbin, pars = "pred_cases", probs = c(0.05, 0.5, 0.95))$summary), t, cases)

# Clean up column names to remove special characters
colnames(smr_pred) <- make.names(colnames(smr_pred)) # to remove % in the col names

# Plot the predictions with credible intervals
ggplot(smr_pred, mapping = aes(x = t)) +
  geom_ribbon(aes(ymin = X5., ymax = X95.), fill = 'red', alpha = 0.35) +
  geom_line(mapping = aes(x = t, y = X50.), color = 'red') +
  geom_point(mapping = aes(y = cases)) +
  labs(x = "Day", y = "Number of students in bed")
```



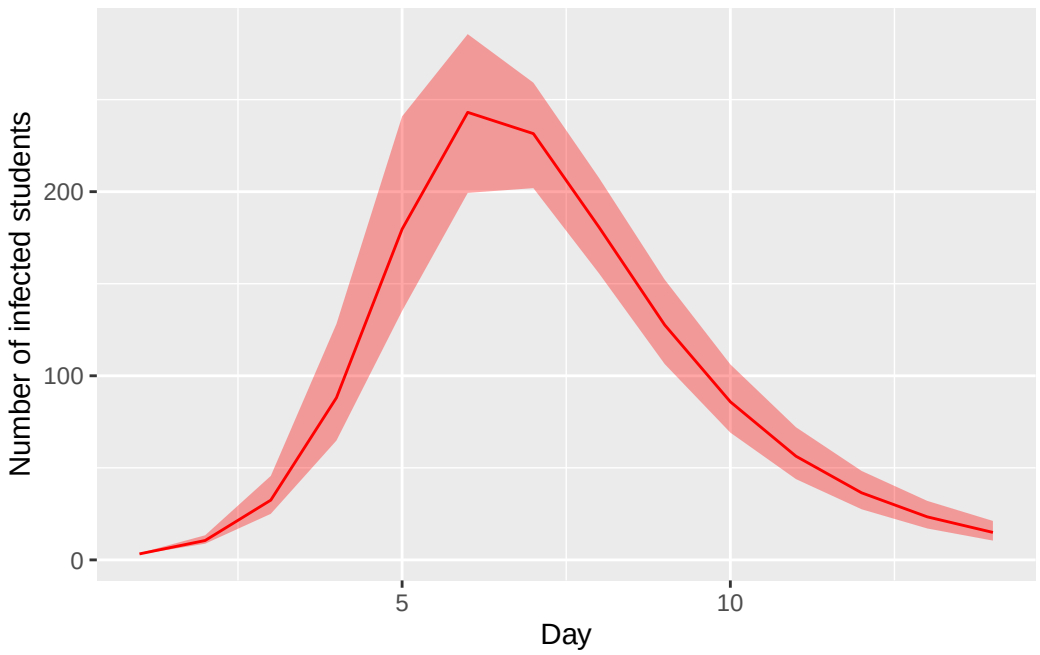
We can clearly see that our posterior mean (the red line) very closely fits the actual prevalence counts over the days. The wide intervals highlight the uncertainty of our SIR model parameters and the predictions generated by them. We are using a negative binomial generating process here, which includes a special parameter to account for overdispersion. This is one reason why our intervals are so wide. If we were to use a Poisson data generating process, our intervals would be less wide but might result in a slightly different fit, potentially being less cautious depending on how one were to percieve this outbreak.

Plotting the number of cases per day

```
# Parameters for the number of infected for each day
params <- lapply(t, function(i){sprintf("y[%s,2]", i)})

# Create a data frame with summary statistics for the number of infected students
smr_y <- as.data.frame(summary(fit_sir_negbin,
                               pars = params, probs = c(0.05, 0.5, 0.95))$summary)
# Clean up column names to remove special characters
colnames(smr_y) <- make.names(colnames(smr_y)) # to remove % in the col names

# Plot the number of infected students with credible intervals
ggplot(smr_y, mapping = aes(x = t)) +
  geom_ribbon(aes(ymin = X5., ymax = X95.), fill = 'red', alpha = 0.35) +
  geom_line(mapping = aes(x = t, y = X50.), color = 'red') +
  labs(x = "Day", y = "Number of infected students")
```



What does this mean for an epidemiologist?

In the analysis above, we use a Bayesian approach to predict the number of students in bed and the number of infected students over time.

The Bayesian framework provides public health officials with credible intervals, which offer a range of plausible values for our predictions. These intervals are essential for informed decision-making.

Bayesian credible intervals allow officials to understand the uncertainty associated with the predictions. If the interval is wide, it indicates greater uncertainty in our estimates, which might prompt more cautious decision-making. Conversely, a narrower interval suggests more confidence in the predictions, which might lead to less conservative actions.

The length of the interval not only reflects the uncertainty about the predicted number of cases but also provides insights into the reliability of the model parameters. Understanding this uncertainty is crucial for public health officials as they balance the need for caution with the desire to avoid unnecessary restrictions.

Experimenting with a different liklihood

As our Analysis in this study is a comparative Bayesian Analysis let's experiment with a Poisson generating process

For a second, let's assume that overdispersion is non existent and let's modify the model to reflect that:

- We can take our existing prevlance model and simply modify the `model` section

```
//sampling distribution
  //col(matrix x, int n) - The n-th column of matrix x. Here the number of infected people
  cases ~ poisson(col(to_matrix(y), 2));
```

- remove `phi_inv` from the `parameters` block (the overdispersion parameter)

```
parameters {
  real<lower=0> gamma;
  real<lower=0> beta;
}
```

- We modify the `generated quantities` block to

```
generated quantities {
  real R0 = beta / gamma;
  real recovery_time = 1 / gamma;
  real pred_cases[n_days];
  for (i in 1:n_days) {
    pred_cases[i] = poisson_rng(y[i, 2]);
  }
}
```

We name this new model `SIR_Prevelance_Poisson.stan` and compile it

Sampling from the Poisson Model

```
model_pois <- stan_model("/Users/varundatta/Desktop/STAD94/Bayesian_Repo/paper/Models/1978 I
```

```r
# Fitting the Poission generating proccess model with the same data
fit_sir_pois <- sampling(model_pois,
                 data = data_sir,
                 iter = niter,
                 chains = 4,
                 seed = 0)
```

Summary of our inference

```r
pars=c('beta', 'gamma', "R0", "recovery_time")
print(fit_sir_negbin, pars = pars)
```

```
Inference for Stan model: anon_model.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

              mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
beta          1.74    0.00 0.06 1.64 1.70 1.73 1.77  1.85  1929    1
gamma         0.54    0.00 0.05 0.46 0.51 0.54 0.57  0.64  2370    1
R0            3.23    0.01 0.29 2.73 3.05 3.21 3.39  3.87  1828    1
recovery_time 1.86    0.00 0.16 1.56 1.76 1.86 1.96  2.19  2147    1

Samples were drawn using NUTS(diag_e) at Mon Jul 29 13:25:29 2024.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

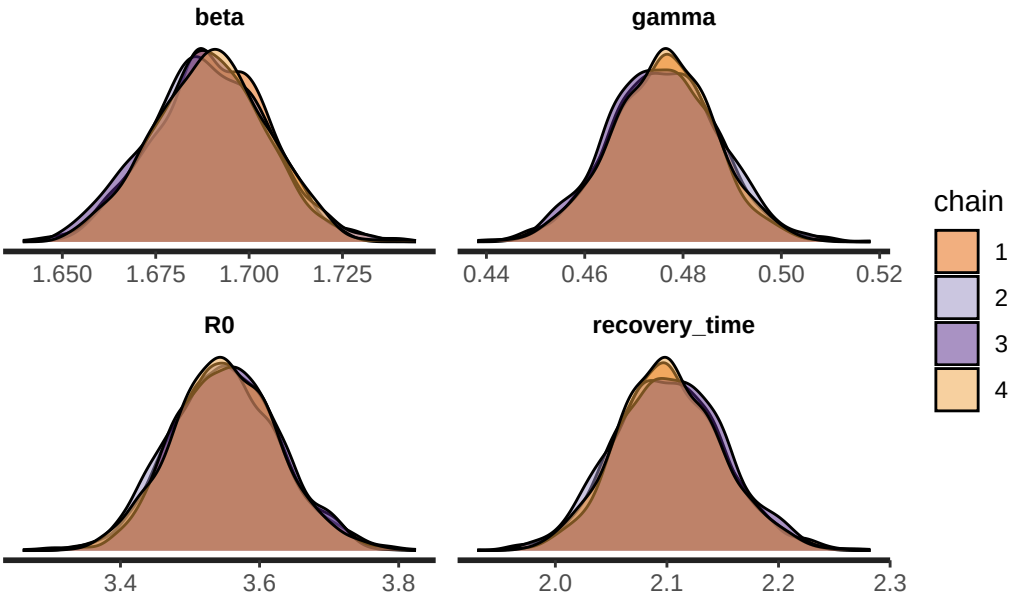
With the values being in a similar range as our Negative Binomial generating model, we get another perspective on our outbreak from a less conservative generating process.

We will compare the two in detail shortly at the end of this section.

With the Rhat $= 1$ for our parameters and transformed/generated parameters we have a good indication that the markov chains have mixed.

A visual check on the chains

```r
stan_dens(fit_sir_pois, pars = pars, separate_chains = TRUE)
```



We can see that the chains have mixed with slight differences in the modes.

Least Squares: Why not use Least Squares as well?

Before we proceed with a comparision of our two Bayesian models, let's utilize the much simpler deterministic least squares approach which finds its roots from non linear regression.

We utilize the function, defined earlier, to fit a least squares SIR model to our dataset.

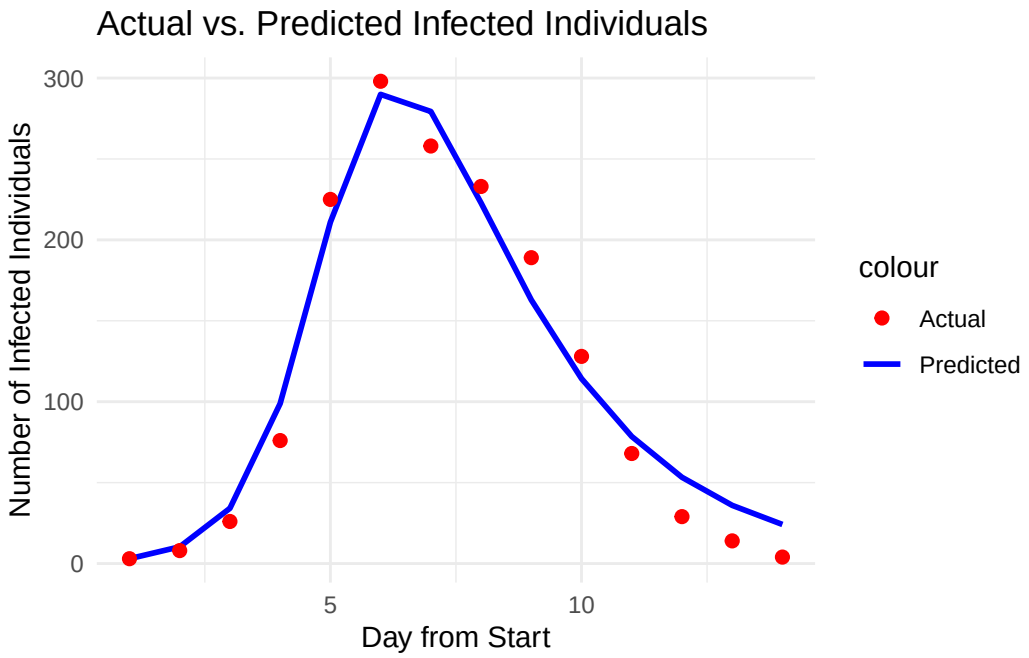
This approach uses an initial guess for parameters $beta = 1.5 \cdot 10^{-3},\ \gamma = 0.44$

```
# Using the function defined above

fit_and_plot_sir(flu_data,763,c(1.5e-3,.44))
```

```
Optimized beta: 1.699742
Optimized gamma: 0.446839
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



The model fits very closely to the actual data points and accomplishes the goal of reducing the squared distance between the fitted line and the actual data.

Using the optimized values of our parameters above let's find our optimized R0 and recovery period

Least Squares Result

```
# Given optimized values
optimized_beta <- 1.699742
optimized_gamma <- 0.446839
# Compute R0
R0 <- optimized_beta / optimized_gamma

# Compute recovery period
recovery_period <- 1 / optimized_gamma

# Given optimized values
optimized_beta <- 1.699742
optimized_gamma <- 0.446839

# Compute R0
R0 <- optimized_beta / optimized_gamma
```

```r
# Compute recovery period
recovery_period <- 1 / optimized_gamma

# Create a data frame
least_squares_estimate <- data.frame(
  Parameter = c("Beta", "Gamma", "R0", "Recovery Period"),
  Estimate = c(optimized_beta, optimized_gamma, R0, recovery_period)
)

# Print the data frame
#print("Least Square's Estimates")
#print(least_squares_estimate)
```

Let's compare the least square estimates with the two bayesian posterior based estimates

Stan SIR Fits Negative Binomial and Poisson

```r
# Extract summaries for the negative binomial model
summary_negbin <- summary(fit_sir_negbin, pars = pars)$summary

# Extract summaries for the Poisson model
summary_pois <- summary(fit_sir_pois, pars = pars)$summary

# Combine the summaries into a data frame
summary_df <- data.frame(
  Model = rep(c("Negative Binomial", "Poisson"), each = length(pars)),
  Parameter = rep(pars, 2),
  Mean = c(summary_negbin[, "mean"], summary_pois[, "mean"]),
  SE_Mean = c(summary_negbin[, "se_mean"], summary_pois[, "se_mean"]),
  SD = c(summary_negbin[, "sd"], summary_pois[, "sd"]),
  `2.5%` = c(summary_negbin[, "2.5%"], summary_pois[, "2.5%"]),
  `25%` = c(summary_negbin[, "25%"], summary_pois[, "25%"]),
  `50%` = c(summary_negbin[, "50%"], summary_pois[, "50%"]),
  `75%` = c(summary_negbin[, "75%"], summary_pois[, "75%"]),
  `97.5%` = c(summary_negbin[, "97.5%"], summary_pois[, "97.5%"]),
  N_Eff = c(summary_negbin[, "n_eff"], summary_pois[, "n_eff"]),
  Rhat = c(summary_negbin[, "Rhat"], summary_pois[, "Rhat"])
)

# Print the data frame
summary_df
```

```
              Model     Parameter      Mean      SE_Mean         SD     X2.5.
1 Negative Binomial          beta 1.7364223 0.0012863076 0.05649140 1.6362239
2 Negative Binomial         gamma 0.5409365 0.0009387305 0.04569846 0.4559596
3 Negative Binomial            R0 3.2324767 0.0068578089 0.29321208 2.7297104
4 Negative Binomial recovery_time 1.8618839 0.0034195232 0.15845549 1.5601973
5           Poisson          beta 1.6894320 0.0002831813 0.01525918 1.6592678
6           Poisson         gamma 0.4759487 0.0002270592 0.01074399 0.4546680
7           Poisson            R0 3.5511950 0.0013182687 0.07660214 3.4071413
8           Poisson recovery_time 2.1021389 0.0010034774 0.04751995 2.0119121
       X25.      X50.      X75.    X97.5.    N_Eff      Rhat
1 1.7019819 1.7338825 1.7670303 1.8542253 1928.746 1.0003568
2 0.5112220 0.5386854 0.5679715 0.6409446 2369.852 0.9998946
3 3.0466173 3.2056679 3.3891321 3.8740529 1828.071 0.9997046
4 1.7606518 1.8563709 1.9560972 2.1931768 2147.254 0.9995802
5 1.6791142 1.6892505 1.6998408 1.7191732 2903.577 1.0015322
6 0.4685564 0.4760004 0.4831899 0.4970396 2238.995 1.0022048
7 3.4983329 3.5497898 3.6029222 3.7087326 3376.556 1.0008918
8 2.0695796 2.1008386 2.1342147 2.1994071 2242.522 1.0022272
```

Least Square's Estimates

```r
print(least_squares_estimate)
```

```
        Parameter Estimate
1            Beta 1.699742
2           Gamma 0.446839
3              R0 3.803925
4 Recovery Period 2.237943
```

What does this all mean? Comparitive Bayesian vs Determinsitic Inference

Talk about the 3 results

Building an SEIR Model(SEIR Prevelances)

Let's take out intiial model with weakly informative priors (`SIR_Prevelance.stan`) and modify it into an SEIR model.

In simple words, we are accounting for the incubation period of the disease by adding another compartment to our ODE based model. [Refer to older section]

Let's break this process down into simple steps

- Modify the `functions` block to design an SEIR model instead an SIR model

```
functions {
  real[] seir(real t, real[] y, real[] theta,
              real[] x_r, int[] x_i) {

    real S = y[1];
    real E = y[2];
    real I = y[3];
    real R = y[4];
    real N = x_i[1];

    real beta = theta[1];
    real sigma = theta[2];
    real gamma = theta[3];

    real dS_dt = -beta * I * S / N;
    real dE_dt =  beta * I * S / N - sigma * E;
    real dI_dt =  sigma * E - gamma * I;
    real dR_dt =  gamma * I;

    return {dS_dt, dE_dt, dI_dt, dR_dt};
  }
}
```

1. dE_dt: The rate of change for the exposed population is added. People move into this compartment from susceptible at a rate β * I * S / N and transition to the infected state at a rate σ * E. This captures the dynamics where exposed individuals become infectious.
2. dI_dt: Now incorporates the inflow from the exposed state (σ * E) and outflow to the recovered state (γ * I), correctly describing the transition from exposure to infection.

- We add sigma as a parameter in the `parameters` block

```
parameters {
  real<lower=0> beta;
  real<lower=0> sigma;
  real<lower=0> gamma;
  real<lower=0> phi_inv;
}
```

- We add prior(weakly informative) on sigma in our `model` block

```
model {
  // priors
  beta ~ normal(2, 1);
  sigma ~ normal(0.5, 0.5);
  gamma ~ normal(0.4, 0.5);
  phi_inv ~ exponential(5);

  // sampling distribution
  cases ~ neg_binomial_2(col(to_matrix(y), 3), phi);
}
```

- We modify the call to the `integrate_ode_rk45` function to call SEIR instead of SIR in the `transformed parameters` section

```
transformed parameters {
  real y[n_days, 4];
  real phi = 1. / phi_inv;
  {
    real theta[3];
    theta[1] = beta;
    theta[2] = sigma;
    theta[3] = gamma;

    y = integrate_ode_rk45(seir, y0, t0, ts, theta, x_r, x_i);
  }
}
```

Least Squares appraoch (SEIR)

Let's re-write our older SIR least squares model and modify it to accommodate the `E` compartment

- **Model Definition**: Define the SEIR model dynamics within the `seir.model` function. This function accounts for four compartments:
  - *Susceptible (S)*
  - *Exposed (E)*
  - *Infected (I)*
  - *Recovered (R)* The model parameters include the transmission rate (`beta`), incubation rate (`sigma`), and recovery rate (`gamma`), with the dynamics applied to a population of size `N`.

- **Error Function**: Implement the sum of squared errors (SSE) function `sse.seir` that calculates the discrepancy between observed cases and model predictions. This function uses the initial number of exposed and infected individuals derived from the data and simulates disease progression to evaluate fit.
- **Model Fitting**: Fit the SEIR model using the `optim` function with Nelder-Mead optimization, encapsulated in the `fit_and_plot_seir` function. Optimization starts from initial parameters, which may be based on empirical data or literature.
- **Prediction and Plotting**: Generate and plot model predictions alongside actual data for visual comparison using `ggplot2`. The function plots:
  - Predicted number of infected individuals as a line graph.
  - Actual data as points.

```r
# SEIR model definition
seir.model <- function(t, x, params) {
  S <- x[1]
  E <- x[2]
  I <- x[3]
  R <- x[4]
  beta <- params[1]
  sigma <- params[2]  # Rate of movement from E to I (incubation rate)
  gamma <- params[3]
  N <- params[4]  # Total population

  dS <- -beta * S * I / N
  dE <- beta * S * I / N - sigma * E
  dI <- sigma * E - gamma * I
  dR <- gamma * I
  return(list(c(dS, dE, dI, dR)) )
}

# Sum of Squared Errors (SSE) function for SEIR
sse.seir <- function(params0, data, N) {
  times <- as.numeric(data$date - min(data$date)) + 1
  cases <- data$in_bed
  E0 <- 0   # Initial number of exposed individuals
  I0 <- cases[1]  # Initial number of infectious individuals
  S0 <- N - I0 - E0  # Initial number of susceptible individuals
  R0 <- 0  # Initial number of recovered individuals

  # Simulate the model
  out <- as.data.frame(ode(y = c(S = S0, E = E0, I = I0, R = R0), times = times, func = seir
```

```
  sse <- sum((out$I - cases)^2)
  return(sse)
}

# Main function to fit SEIR model and plot results
fit_and_plot_seir <- function(data, N, initial_params = c(0.5, 1/3,  0.56 )) {
  # Optimization using Nelder-Mead method
  fit <- optim(par = initial_params, fn = function(params) sse.seir(params, data, N), method

  # Print optimized parameters beta, sigma, and gamma
  cat(sprintf("Optimized beta: %f, sigma: %f, gamma: %f\n", fit$par[1], fit$par[2], fit$par[3

  # Generate predictions using the optimized parameters
  days <- as.numeric(data$date - min(data$date)) + 1
  model_output <- as.data.frame(ode(y = c(S = N - data$in_bed[1], E = 0, I = data$in_bed[1],

  # Plot actual vs. predicted
  ggplot() +
    geom_line(data = model_output, aes(x = days, y = I, colour = "Predicted"), size = 1) +
    geom_point(data = data, aes(x = days, y = in_bed, colour = "Actual"), size = 2) +
    labs(title = "Actual vs. Predicted Infected Individuals (SEIR Model)",
         x = "Day from Start",
         y = "Number of Infected Individuals") +
    scale_color_manual(values = c("red", "blue"), labels = c("Actual", "Predicted")) +
    theme_minimal()
}
```

1978 Influenza Boarding School (SEIR Approach)

Utilizing the SIER model that we just built above let's fit it to our data to determine whether we gain additional insights or not.

SEIR (STAN/ Bayesian Approach with Prevelance)

Compiling and simulating from the model

```
model_SEIR_prevelance <- stan_model("/Users/varundatta/Desktop/STAD94/Bayesian_Repo/paper/Mod
```

```
recompiling to avoid crashing R session
```

```
Trying to compile a simple C file


Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: 'Apple clang version 15.0.0 (clang-1500.3.9.4)'
using SDK: 'MacOSX14.4.sdk'
clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
#include <cmath>
         ^~~~~~~
1 error generated.
make: *** [foo.o] Error 1
```

```r
# total count of students
N <- 763;
cases <- flu_data$cases

# times
n_days <- length(cases)
t <- seq(0, n_days, by = 1)
t0 = 0
t <- t[-1]

# initial conditions
i0 <- 1
e0 <- 2  # Assuming some exposed individuals initially
s0 <- N - i0 - e0
r0 <- 0
y0 = c(S = s0, E = e0, I = i0, R = r0)

# data for Stan
data_seir <- list(n_days = n_days, y0 = y0, t0 = t0, ts = t, N = N, cases = cases)

# number of MCMC steps
niter <- 2000
```

```r
fit_seir_negbin <- sampling(model_SEIR_prevelance,
                data = data_seir,
                iter = niter,
```

```
                chains = 4,
                seed = 0)
```

Summary of our Fit and Posterior Quantiles

SEIR (Least Squares Approach)

New York City Covid 19 Wave 1

New York City will be henceforth refered to as NYC.

Initial SIR FIT using STAN and Least Squares

New York State Covid 19 Wave 1

New York State will be hencforth refered to as NYS.

Conclusion:

Areas of future exploration or possible extensions to this work

- Utilizing Data from Sereological surveys to make a better model
- Create a R Library which works as a front end for the functions that were implemented in here

Web Appendix