2023

ANALYSIS OF COVID-19 IN RURAL AMERICA

Ian Gabriel Derickson

ANALYSIS OF COVID-19 IN RURAL AMERICA

By

IAN GABRIEL DERICKSON

B.S. Corban University, Salem, OR, 2014
M.S. University of Montana, Missoula, MT, 2022

Dissertation

presented in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in Mathematics

The University of Montana
Missoula, MT

May 2023

Approved by:

Ashby Kinch
Graduate School Dean

Dr. Leonid Kalachev, Chair
Department of Mathematical Sciences

Dr. Johnathan Bardsley
Department of Mathematical Sciences

Dr. Javier Perez Alvaro
Department of Mathematical Sciences

Dr. Frederick Peck
Department of Mathematical Sciences

Dr. Douglas Brinkerhoff
Department of Computer Science

Derickson, Ian, Ph.D, May 2023 *Mathematics*

Analysis of COVID-19 in Rural America

Chairperson: Dr. Leonid Kalachev

The outbreak of the novel coronavirus (SARS-CoV-2) in Wuhan, China and subsequent global pandemic illustrated the need of accurate forecasting and a greater understanding of the underlying dynamics of infectious diseases.

In this dissertation, we examine the six most populous towns in the State of Montana through the lens of the classic SIR model. With this, we show that the six towns in question exhibit similar disease dynamics and population behavior within each wave. Furthermore, we conduct analysis on the age demographics of COVID-19 cases and deaths. This analysis corroborates the findings of the SIR model fits, in that the age demographics of cases, deaths, and case fatality rates are remarkably similar across all six towns, lending credence to the hypothesis that the dynamics of infectious diseases are fundamentally the same within rural towns in America.

Acknowledgements

The last five years leading up to this point have been an unforgettable experience. From the people who I have met, both professors and colleagues, as well as the multitude of experiences, I look forward to the next chapter of my life. Looking back on my time here, I would like to take a moment to acknowledge the individuals who made this achievement possible.

First and foremost, I would like to thank my advisor, Leonid Kalachev, for putting up with me for the past three years. I am greatly appreciative of all the advice and support he has given to me, in spite of an often great distance, throughout these past years. I am very thankful for his willingness to give me advice and suggestions whenever I would ask, while also giving me the room to figure things out for myself. A major factor of your experience in graduate school is dictated by your advisor, and I am thankful for having such a great one.

It is often said that you learn as much from your fellow students as from your professors. If I were to list all of them out, the list would be far too long. Instead, I would like to thank you all for putting up with my terrible puns and helping me in my mathematical journey. I have learned so much from all of you. Thank you for always having a smile and an encouraging word, no matter the circumstance.

The research done here would have been impossible without data. For this reason I would like to thank Erin Landguth for supplying me with critical parts of the data used for this dissertation. Along with this, I would also like to thank the Montana Department of Public Health and Human Services, Communicable Disease Epidemiology Section for collecting, aggregating, and making this data available. Without you, this research would not have been possible.

I would like to thank my parents, whose love and support have sustained me throughout my time here. They have been with me every step of the way, through the highs, the lows, and everywhere in between. Without you, I would have never made it to this point.

Lastly, I would like to thank you, the reader, for taking the time to read however much of this you get through. I hope that some of what I have researched and written here can be edifying for you.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In December of 2019, an outbreak of the novel coronavirus (SARS-CoV-2) in Wuhan, China would rapidly spread and lead to a global pandemic being declared by the World Health Organization in early 2020. This disease and its mutations have been known colloquially as COVID-19. With the incidence of the most severe virus outbreak in around a century, great emphasis was put on mathematical models of COVID-19 and its spread in a desire to accurately inform policy decisions. It is important to discuss the purposes of mathematical models of disease, how different models work both from a philosophical and a design standpoint, why different models come to different conclusions, and how a model's trustworthiness is gauged. It should be noted, however, that discussions of mathematical models of disease will be restricted to models that were used during the COVID-19 outbreak to predict the disease dynamics.

In this paper we will be examining the dynamics of the COVID-19 disease outbreak in rural America, in particular the state of Montana. Attention will be primarily given to the six most populous counties of Montana, those being Yellowstone, Gallatin, Missoula, Flathead, Cascade, and Lewis and Clark. Through examination of both fitting of data to SIR models and death cases data, we will show that the different large cities in Montana, in spite of being disparate, exhibit similar disease dynamics. Using this, we will also show how the disease and population dynamics from one city can be used to approximate the disease and population dynamics of another city.

Chapter 2

Infectious Disease Modeling

2.1 Purposes of Mathematical Models of Diseases

Mathematical models of disease serve different purposes depending upon when in the relation to the occurrence of the outbreak they are being employed. During a disease outbreak, mathematical models of disease allow us to extrapolate the state and progress of the outbreak from current information, predict the future dynamics of the disease, and to quantify the uncertainty of these decisions. This information is incredibly important, as policymakers need this information in order to make educated decisions. During the COVID-19 pandemic, several different models were created for this exact purpose, the most well-known ones being the models developed by the Imperial College of London and the University of Washington. These models were known as CovidSim [1] and IHME [2] or the ’Ferguson’ and ’Chris Murray’ models respectively, the second names being after the heads of the programs that developed those models.

A report to the British government (Report 9 [1]) that used the CovidSim model to make predictions was a major factor in the decision of the first UK lockdown as well as a contributing factor to the US shutdowns. Report 9 predicted that without intervention around 500,000

UK citizens would die as well as around 2.2 million Americans, and that at its peak, the ICU and critical care bed requirement would be around 30 times greater than the maximum of both counties combined. With this we can see how during an outbreak, extrapolation of the (then) current state and progress of the outbreak as well as future states of the disease were used to help make policy decisions for two nations. However, the uncertainty of these estimates was not properly quantified [3].

The ending of an outbreak does not imply the end of the use of a mathematical disease model, but the purpose of the model changes. Instead of trying to understand the present and predict the direct future, mathematical models can be used to analyze what happened over the course of the disease. Population dynamics, disease virulence, and rate of recovery of the disease can all be inferred from models of a full outbreak. In the case of COVID-19, such models can also be used to understand the effect of other factors, such as the mask mandate, social distancing, and quarantining. In addition, we can also simulate different 'what if' scenarios for the disease such as different vaccination rates. The information gained from this analysis can be used the next time a similar outbreak occurs in order to have a better starting point for the model.

2.2 The Susceptible - Infected - Recovered Model

Epidemilogical modeling has several categories and classifications, the original and most notable being the Susceptible - Infected - Recovered (SIR) model. Developed in 1927 by A. G. McKendrick and W. O. Kermack, the SIR model is used to systematically describe various types of epidemics such as influenza and COVID-19, as well as inform policy decisions [4] [5] [6]. The SIR model is a compartmental model, in which the population is split into different classes, or compartments, these being Susceptible, Infected, and Recovered/Removed. The susceptible population would be those at risk of being infected by the disease, the infected population would be those infected by the disease, and the recovered/removed population would be those no longer at risk of being infected. It has been widely assumed that the rate at which a susceptible individual is infected is proportional to the product of the number of susceptible (S) and number of infected (I) [4] [7] [8] [9]. This is known as the Law of Mass Action [10] [11].

2.2.1 The Law of Mass Action

The Law of Mass Action states that the rate of a chemical reaction in a solution in dynamic equilibrium is proportional to the concentrations of the reactants [12]. Specifically, it states that the proportion between the reactants and products is constant once equilibrium is achieved [13]. If we let the rate of new infections be C and α be a constant, then the Law of Mass Action is expressed as

$$C = \alpha S I. \tag{2.1}$$

Use of the Law of Mass Action for the SIR model comes with the assumption of homogeneous mixing of the susceptible and infected populations. However, it can be observed that the rate of infection for certain diseases is higher within a household than without.

The constant α is then assumed to be an aggregate that implicitly describes the range of contact among different groups of susceptible and infected [11].

2.2.2 Derivation of the SIR model

If we assume that for a given infection, an individual once infected stays infected, (2.1) can be expressed as (2.2).

$$C = \frac{\delta I}{\delta t} = \alpha S(t) I(t) \tag{2.2}$$

$S(t)$ and $I(t)$ are the susceptible and infected populations at time t. If we assume that the new infections are coming purely from the susceptible population, then we can express the rate of change of the susceptible population in (2.3):

$$\frac{\delta S}{\delta t} = -\alpha S(t)I(t) \tag{2.3}$$

Along with a description of the movement from the susceptible population to the infected population, we also need a description of the movement of the infected population to the recovered/removed population. Several factors impact the movement of the infected population to the recovered/removed population. Much like with α, let the constant β be an aggregate that expresses the range of rates at which the infected population becomes no longer able to infect the susceptible population, whether by removal or recovery. Modifying (2.2) to reflect this gives us (2.4).

$$\frac{\delta I}{\delta t} = \alpha S(t)I(t) - \beta I(t) \tag{2.4}$$

Since βI describes the rate of recovery/removal, we can express the rate of change of the recovered/removed population in (2.5):

$$\frac{\delta R}{\delta t} = \beta I \tag{2.5}$$

Combining (2.3 - 2.5), we can also add an equivalence relation for the total population N.

$$\frac{\delta S}{\delta t} = -\alpha S(t)I(t) \tag{2.6}$$

$$\frac{\delta I}{\delta t} = \alpha S(t)I(t) - \beta I(t) \tag{2.7}$$

$$\frac{\delta R}{\delta t} = \beta I \tag{2.8}$$

$$N = S(t) + I(t) + R(t) \tag{2.9}$$

Equations (2.6-2.9) are the basic SIR model. As stated previously, the variables of S(t), I(t), and R(t) are the time-dependent values of the susceptible, infected, and recovered/removed populations respectively. N is the total population that is being analyzed.

The parameter α represents the aggregation of several factors, such as the virulence of the disease, as well as population interaction/dynamics. We can express these factors with $\alpha = v \cdot d$ where the parameter v is virulence, or the probability that exposure to the disease results in infection, and the parameter d represents population dynamics, or the number of exposures an infected individual will cause prior to removal/recovery. In a standard SIR model, α is not separated into v and d. Several studies have been done regarding incidence rate, however the incidence rate described is equivalent to or a slight modification of our C from (2.2) [14] [15] [16] [17] [18] [19].

The parameter β represents the rate at which an infected person is removed from the pool of people able to infect others (susceptibles). Furthermore, β is the reciprocal of this average time it takes for recovery/removal.

As a system of differential equations, the SIR model represents how S, I, and R change over time. When using these equations, different curves are formed for the Susceptible, Infected, and Recovered/Removed populations, and any number of curve-fitting algorithms are used to fine-tune α and β to fit the observed data.

2.3 SIR Model Assumptions

In any mathematical model of natural processes, assumptions must be made. Understanding these assumptions, both implicit and explicit, aid in interpretation of and appropriate utilization of results. As stated previously, an assumption implicit to SIR models is that the susceptible and infected population mix homogeneously. While we know that this is untrue [11], as stated previously, the parameter α is taken as an average of the rate of mixing.

An explicit assumption made with the basic SIR model is that is a closed system. In equations (2.6)-(2.9), there is no provision made for births or deaths. The assumption is that the rate of births and deaths (sometimes referred to as 'vital dynamics' [20] [21]) has a negligible effect on the overall dynamics. Births and deaths can be further extrapolated to include entry and exit from the population in question, such as via travel [22] [23].

Another assumption made is one of immunity. By the nature of (2.6)-(2.9), it can be seen that it is assumed that once someone has become recovered/removed, they are no longer susceptible to infection. The validity of this assumption generally depends on the disease in question and the definition of the disease in question. For example, vaccination against polio, measles, mumps, and chicken pox has caused a near-eradication of the diseases in the United States [24] [25] [26]. However, influenza continues to be a seasonal disease, as evidenced by the need for yearly vaccinations [27] [28] [29]. As a result of this, the SIR model also assumes eventual eradication of the disease in question, since it is a closed system, and everyone will eventually become non-susceptible to it.

There is also no provision made in the SIR model for asymptomatic patients. While this is certainly a factor within disease dynamics, it is difficult to model for it when fitting data to the SIR model, while still having an impact upon disease dynamics [30] [31].

2.4 SIR Model Variations

The SIR model, while being the most notable and original, has led to many variations over time. These variations are developed to either compensate for assumptions of the SIR model,

model more complex behavior, or utilize the data on hand, which might not work well in a basic SIR model. We will discuss several variations of the SIR model that are used in the modeling of COVID-19.

2.4.1 SIRS

The Susceptible - Infected - Recovered - Susceptible (SIRS) model does not assume that immunity is permanent. This is a reasonable assumption, as diseases mutate over time, causing immunity gained through either vaccination or infection to be ineffective. A result of the SIRS model is a disease reaching an endemic steady-state, one in which the disease is always present [32]. In recent times, an example of an epidemic becoming endemic was HIV, while an older example was tuberculosis [33]. A disease can be prevented from reaching an endemic state by vaccines, as seen with polio [34]. In the context of COVID-19, the SIRS model was infrequently used to simulate the pandemic [35] [36].

2.4.2 SEIR

A highly popular variation of the SIR model is the Susceptible - Exposed - Infected - Recovered (SEIR) model. Frank defined the SEIR model with a series of differential equations [37]:

$$\frac{\delta S}{\delta t} = -\beta S(t)I(t) \tag{2.10}$$

$$\frac{\delta E}{\delta t} = \beta S(t)I(t) - \alpha E(t) \tag{2.11}$$

$$\frac{\delta I}{\delta t} = \alpha E(t) - \gamma I(t) \tag{2.12}$$

$$\frac{\delta R}{\delta t} = \gamma I(t) \tag{2.13}$$

$$N = S(t) + E(t) + I(t) + R(t) \tag{2.14}$$

Here, S(t), E(t), I(t), R(t), and N represent the number of susceptible, exposed, infected, recovered, and total population respectively. β, α, and γ represent the exposure, infection, and recovery rates, respectively. Li and Wang examined the SEIR model and a few variations and showed the positivity, boundedness, and asymptotic smoothness of the solutions. After this, they showed the existence and local stability of disease-free and endemic steady states [38]. Mwalili et. al used the SEIR model with modifications for pathogen in the environment and interventions to compute R_0 and used the result to predict the persistence of COVID-19 without strong control measures [39]. Shin uses the SEIR model to examine "carefully distinctive stages" of COVID-19 in South Korea with an additional death compartment and shows the difficulty of using a single model for the three waves [40]. Wang et. al proposed a modification of the SEIR model taking hospitalizations into account [41]. Yang et. al proposed a modified SEIR that uses move-in and move-out parameters to simulate external influences [42]. Saikia et. al used an SEIR model to predict the dynamics of COVID-19 in India in late 2020 [43]. There are many instances of the SEIR model or variations being used for prediction [44] [45] [46] [47] [48].

2.4.3 SIRD

The Susceptible - Infected - Recovered - Dead (SIRD) model adds in the possibility of death as a result of infection. Calafiore et. al proposed a SIRD model to predict COVID-19 in Italy [49]. Bailey et. al defined the SIRD model with a series of differential equations [50]:

$$\frac{\delta S}{\delta t} = -\beta S(t)I(t) \tag{2.15}$$

$$\frac{\delta I}{\delta t} = \beta S(t)I(t) - \gamma I(t) - \mu I(t) \tag{2.16}$$

$$\frac{\delta R}{\delta t} = \gamma I(t) \tag{2.17}$$

$$\frac{\delta D}{\delta t} = \mu I(t) \tag{2.18}$$

$$N = S(t) + I(t) + R(t) + D(t) \tag{2.19}$$

Here, S(t), I(t), R(t), D(t), and N represent the number of susceptible, infected, recovered, deceased, and total population respectively. β, γ, and μ represent the infection, recovery, and mortality rates, respectively. Haghrah et. al proposed a "fuzzy" SIRD model as a way to predict COVID-19 with consideration for government intervention [51]. Sebbagh and Kechida used an SIRD model to predict parameters such that relevant information could be given to hospital supervisors to hopefully reduce mortality rate [52]. Shringi et. al use a modified SIRD model to predict COVID-19 in India [53]. Athayde and Alencar used a dynamic SIRD model, in which infection, recovery, and mortality rates are allowed to vary over time [54]. In fact, many papers that utilized the SIRD model were attempting to make data-driven predictions of COVID-19 [55] [56] [57] [58] [59].

2.5 The Basic Reproduction Number R_0

In any discussion of infectious diseases, the basic reproduction number (R_0) is utilized as a measurement of the severity of an epidemic [60]. An epidemic is defined as when the population of infected begins to grow when an infected individual is introduced. In the SIR model, this can be interpreted as when $dI/dt > 0$. Using (2.7) we can derive a formulation for this as follows:

$$\begin{aligned} \frac{dI}{dt} &> 0 \\ \alpha \cdot S \cdot I - \beta \cdot I &> 0 \\ \alpha \cdot S \cdot I &> \beta \cdot I \\ \alpha \cdot S &> \beta \\ \frac{\alpha S}{\beta} &> 1 \end{aligned}$$

When considering the beginning of an epidemic, generally we consider the addition of a single infected individual to the overall population. Thus, we can assume that $S(0) \approx N$ with N being defined in (2.9). In some papers R_0 is computed as simply α/β [61] while others compute it as $\alpha N/\beta$ [62] [63]. Olivera and Rivera as well as Manrique-Abril et. al guess R_0 and analyze the approximate fit [64] [65]. Other papers simply use the values for R_0 as generated by others for their computations [66] [67].

2.6 Agent-Based Modeling

Another popular method for mathematical disease modeling is Agent-Based Modeling (ABM). As mentioned in Section 2.3, one of the assumptions of the SIR model is the homogeneous mixing of susceptible and infected populations. ABM seeks to overcome this particular assumption by simulating the actions and interactions of multiple autonomous 'agents.' [68] As a result, while the SIR model looks at the population as a single entity, ABM can separate the population into multiple entities and simulate the disease dynamics in that manner, leading to greater accuracy in comparison to traditional compartmental models [69]. De-Leon and Aran utilized a Monte-Carlo ABM to model the spread of COVID-19 in Israel [70]. Cuevas developed an ABM to model the spread of COVID-19 within facilities [71]. Dong et. al utilized an ABM to model the spread of COVID-19 in the Huangpu District of Shanghai [72].

A common practice is to utilize the SIR compartmental concept in the framework of ABM. Geng et. al used a kernel-based SIR model to introduce a spatial element to the spreading of COVID-19 to compensate for different population densities [73]. Nguyen et. al used an agent-based SIR model that attempted to take into account mobility within an urban environment [74]. Lima and Atman examined the effect of mobility restriction in superspreading event via an agent-based SIR model [75]. Gunaratne et. al used a similar to examine the efficacy of non-pharmaceutical interventions on indoor spreading [76].

The most well-known ABM is CovidSIM, developed by the Imperial College in London [77]. It was the source of the predictions used in Report 9 that lead to the lockdown of the UK and contributed to the lockdown in the US. In addition, it is used as a tool by others to model COVID-19 [78] [79]. Other groups have used the CovidSIM model as a base, developing extensions for it [80] [81]. While there are concerns regarding its veracity [82], it stands as one of the most notable ABM models for COVID-19. Other publicly available agent-based models are OpenABM-Covid19 [83] and Covasim [84].

Chapter 3

COVID-19 in Montana

3.1 Geographical Area of Study

As stated in the introduction, the focus of this dissertation is on the dynamics of COVID-19 in rural America, with the focus being on the State of Montana. The State of Montana is located in the northern United States and features vast geography, from the Rocky Mountains, to the Great Plains, as well as Glacier National Park and Yellowstone National Park. In comparison to other states, while having a population only around 8000 greater than Rhode Island, one could fit the states of Main, Vermont, New Hampshire, Massachusetts, Rhode Island, Connecticut, New Jersey, and New York all in Montana and still have room left over. For this reason, the State of Montana is considered a rural state.

For the sake of study, the main work has been focused on the six most populous counties and their corresponding largest cities. These counties are Yellowstone, Gallatin, Missoula, Lewis and Clark, Cascade, and Flathead. These counties each contain a major population center for Montana and all have at least 50 thousand residents in the county, not necessarily the cities. Table 3.1 displays the relevant information regarding the different counties and cities of interest.

	Yellowstone	Gallatin	Missoula	Flathead	Cascade	Lewis & Clark
County Pop.	167146	122713	119533	108454	84511	72223
City Name	Billings	Bozeman	Missoula	Kalispell	Great Falls	Helena
City Pop.	117445	54539	74822	26110	60403	33120
City Area	43.41	19.12	27.51	11.64	21.79	16.35
City Pop. Density	2705.482	2852.458	2719.811	2243.127	2772.051	2025.688

Table 3.1: County and City information for six most populous counties and cities.

For visual reference in Figure 3.1 the counties being examined are shaded and the locations of corresponding cities are shown.

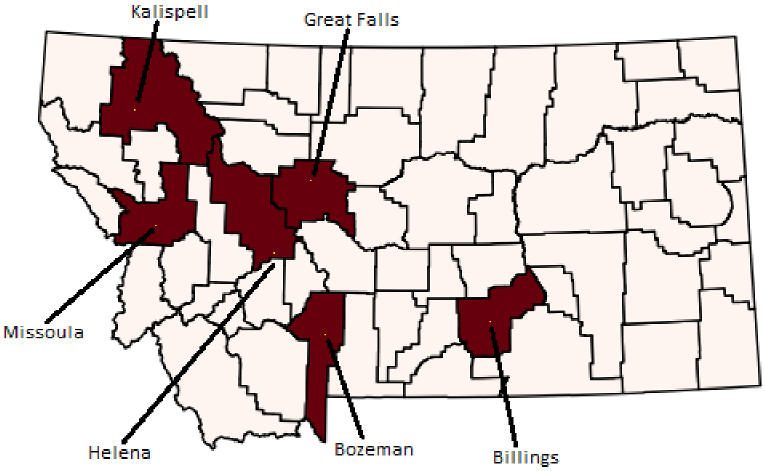

Figure 3.1: Montana Cities and Counties of interest.

Along with the regular information, a few points are good to make about specific counties and cities. The city of Missoula, located in Missoula county, as well as Bozeman, located in Gallatin County, are home to the University of Montana and Montana State Universities, respectively. As college towns, there are a high number of individuals of average age 18 to 22 residing within these towns. In addition, the existence of the universities has the possibility of a potential influence on the dynamics of the disease that we will explore further below.

It is important to note that the six cities being examined are all quite similar in terms of population density. While Billings, Bozeman, Missoula, and Great Falls are closest, Kalispell and Helena are also similar. It is the opinion of the author that in spite of the differences in locale and population demographics, the six cities discussed retain many similarities in terms of urban construction and general culture. While there are certainly differences, in comparison to the wider United States, the author believes that these differences are negligible.

3.2 Heuristic examination of the three waves of COVID-19

Visual examination of any graphic of COVID-19 elucidates three separate waves of infection, or outbreaks, caused by three different strains of COVID-19. While some debates may exist of a wave prior to the declaration of the pandemic and current strains, the focus of this research shall be on the three main waves. These waves are known, and shall be referred to throughout this paper, as the Alpha, Delta, and Omicron waves. Let us now discuss the three waves, their characteristics, and in particular how they manifested in the six counties in Montana.

The Alpha wave, which took place between approximately 6/10/2020 and 3/17/2021, was the first major outbreak. One thing to note is that within these counties of Montana, the vaccines were made available and were being distributed about halfway through the wave. Along with this, for many people the vaccine came in two doses, and while the first dose reportedly had a high rate of effectiveness, there was still a multiple-week wait in full effectiveness, as well as not everyone became fully vaccinated. In addition, the different counties had approximately the same wave structure throughout the Alpha wave. (Refer to appropriate graph here). In other words, the Alpha wave hit all 6 counties at approximately the same time.

The Delta wave, which took place between approximately 8/6/2021 and 12/24/2021, was the second major outbreak. Unlike the Alpha wave, in which immunization began only halfway through and had the issue of two doses, a large part of the population was vaccinated by this time. In addition, the vaccines seemed to be effective against the Delta variant, which was reflected in the systematically lower total number of cases. Like the Alpha wave, all six counties had similar timings of the waves. By this time, many practices, such as mask wearing and social distancing, had become mostly normalized. However, travel restrictions were gradually being lifted, and people were traveling more than during the Alpha wave. In addition, the relationship between the Delta wave and Omicron wave cannot be ignored. The Delta and Omicron waves overlapped slightly, as the Delta wave had yet to fully cease before the Omicron wave began. Therefore, cases from both waves are inexorably mixed, which will be discussed further below.

The Omicron wave, which took place between approximately 12/24/2021 and 3/4/2022, was the last major outbreak. The Omicron wave set itself apart from the other two for a few reasons. The first reason was that while the other two had a component of vaccination and immunization involved, these measures appeared to be completely inert regarding Omicron. For all intents and purposes, the prior vaccinations had no effect upon the Omicron variant. The second point of interest was that the Delta wave had not quite finished yet. As a result, there is a certain degree of overlap between the two in the data. Unfortunately, since information regarding which variant a particular patient was infected with is unavailable, we can only make rough estimation. The last point of difference with the other two variants comes with the timing. As you can see from the dates provided, the Omicron wave hit during the time of year with the greatest amount of travel. As COVID-19 had been going for nearly two years by this point, people were less concerned with travelling. While attempts at social distancing, mask wearing, and other steps to help prevent the spread were still employed, the holiday travel season still contributed heavily to the massive spread of the Omicron variant.

Chapter 4

Goals, Motivations, and Differences

The goal of this dissertation is to present an accurate data-driven methodology for modeling COVID-19. There are thousands of papers regarding COVID-19, with many different models being presented. While some models, such as the IHME model [2] and CovidSIM, are more well known, there have been criticisms of them and their role in policymaking [85] [86]. One issue we see with these models, and several models that are used, is the inherent complexity in the models when compared to the data being used to fit them. Commonly this data is the number of infected. Because of this, overfitting is common, leading to excellent model fits but poor extrapolation. In addition, a common practice is to utilize parameters by other for model predictions [64] [87].

The model and methodology presented in this dissertation is different from standard SIR modelling practice in three major categories. First, the models and data fits are purely data-driven. We do not borrow parameters from other studies, nor do we borrow any additional information. Second, in the context of the SIR model, we fit to the recovered/removed compartment. Most papers fit their models to the infected compartment. We will show how this practice is highly erroneous. Lastly, when using the SIR model, we utilize population density instead of straight population numbers. We will now explain the reasoning and implications of these choices.

4.1 Data Driven Modeling

Data driven modeling is nothing new, as most models are often fit to data to understand natural phenomena. However, many of the models used for COVID-19 attempted to account for complex behaviors and thus made a certain number of assumptions that, while they have a basis in data, are often not strictly data-driven. At best, these assumptions are made based off of extrapolations of data, while at worst, parameters for other fits, such as those derived from Wuhan in March 2020, are used as fits for other locations. The paper by Li et. al detailing the dynamics at Wuhan is one of the most referenced sources for model parameters [88]. For our model, we do not use any model parameters for information that is not directly derived from data. In the next chapter, we will briefly discuss the sources of data used. All model parameters used are purely derived from our fitting. Through this, we will show that an accurate COVID-19 model can be purely data-driven.

While there are some assumptions that we make, which will be discussed subsequently, we will show that those assumptions are necessary. However, beyond those assumptions, we make no assumptions about disease dynamics, population dynamics, or vaccination rates. All of these factors are derived from the data. Because of this, we are using the basic SIR model (2.6)-(2.9) and thus are only fitting α and β, or the rate constant for infection and rate constant for recovery/removal, respectively. Because we are fitting only two parameters, we are less prone to overfitting.

4.2 Fitting to Recovered/Removed Compartment

It is a common practice with COVID-19 modeling to fit to the infected data [89] [90] and with SIR models in particular to the infected compartment [91] [92]. This is a highly erroneous practice. Maronov and Marinova discuss the fact that the data reported is often the newly infected [93]. Law et. al for their model to both the infected and recovered compartments and noted the difficulty in fitting to the infected compartment [94]. Furthermore, Cooper et. al propose a method of fitting data to the recovered compartment [95] that is also utilized by Singh et. al [96]. For our examination, we are going to be fitting to the recovered/removed compartment.

As mentioned above, most papers fit the data to the infected compartment of the SIR model. This is an understandable mistake, as the data is often reported as simply infected, or total cases [97]. Some places reported on newly infected instead of total infected [98] [99] [100]. However, the newly reported cases should not be considered as infected. The infected compartment of the SIR model represents the total number of infected individuals at the given time point. The issue that arises is that the data collected and reported is only of newly infected. While some smaller instances, such as the British boarding school [101], have the data for total infected, it is usually untenable to collect the data for total infected, as that requires not only knowledge of time of infection but also of recovery. In addition, we must also consider population behavior.

Because of its widespread nature and attention it received both from the media and from the public, the behavior of the average person was quite different during the COVID-19 pandemic in comparison to even the H1N1 outbreak. During the usual influenza season and even during the H1N1 outbreak, if someone got sick they did not typically quarantine themselves, instead continuing on with their lives, leading to schools, public transportation, and other public places becoming superspreader locations for these diseases. However, during the COVID-19 pandemic, when people either confirmed or believed themselves to have COVID-19, they would generally immediately quarantine themselves. We can assume that this is generally true for the recorded cases, as the recorded cases were gathered from official testing sites and medical facilities. The implication of this is that we can assume that if someone was recorded, they very quickly quarantined themselves. This requires an adjustment in consideration of our model and data. It is for this reason that we propose for the case of COVID-19 that instead of a recovered compartment, we consider the compartment the removed compartment. Because infected individuals quarantined themselves well before they fully recovered, it is more appropriate to consider them removed.

Considering the removed compartment allows for a different treatment of the available data. As stated above, generally the publicly available data records the newly infected. Since we know that the people getting tested and coming up positive swiftly quarantine, we instead consider the 'newly infected' as 'newly removed' and thus create a removed dataset by summing the number of newly removed each day/ time period. To reiterate, we can consider recorded cases in such a manner only because it was a nearly universal practice to quarantine upon infection confirmation. It will be shown in the data fitting later that the practice of quarantining upon infection is supported by the data.

4.3 Population Density

The last major departure from standard COVID-19 SIR modeling practice that we employ is examination through the lens of population density instead of straight population. If we return to the Law of Mass Action, we can see that it states that the rate of chemical reaction is proportional to the concentrations of the reactants. Population density is equivalent to concentrations of reactants. Yusuf and Shesha show population density is an important factor to consider for pandemic dynamics [102]. In addition to considering population density, we consider the population densities of cities. Since cities are not only where the majority of population interactions take place but also where most of the facilities that record the data utilized are located, it seems sensible to scale the data such that the N in (2.9) is the city population density. Eilersen and Sneppen discuss the difference in superspreading in cities as opposed to countrysides, showing that when modeling dynamics, it is much more useful to model cities instead of the countryside [103]. Utilization of the city population density also implicitly acknowledges and accounts for the fact that different population densities will lead to different rates of infection.

Chapter 5

Data Processing and Methodology

5.1 Data Sources

As mentioned in the Acknowledgements, the main set of data used for this dissertation was provided by Erin Landguth and was from the Montana Department of Public Health and Human Services, Communicable Disease Epidemiology Section. The vaccination dataset was acquired from the Centers for Disease Control and Prevention (CDC) dataset repository. County age demographics, county/city area, and county/city populations was acquired from the 2020 United States Census Bureau. The dataset containing total cases and deaths from COVID-19 for the counties in Montana were acquired from the Montana State Library dataset repository. None of the data or parameter values used in this dissertation were from other papers, and all of the parameter fits are derived purely from the data.

5.2 Analysis of Data

We begin analysis of the data with the Alpha wave. The reason for this is several-fold. While the three waves carried multiple differences, the fundamental mechanics, at least through the lens of an SIR model, were all the same. Therefore, techniques and initial conditions used for the Alpha wave could be carried over to the other two waves with minimal adjustment.

The first item to discuss when analyzing the data is the choice of timescale used. While many data sets and reports utilize a daily update, we have chosen to examine the data on a week-by-week basis. The first reason for this concerns the shape of the daily data and the inherent nature of the data collection. The primary data type used is the number of newly infected individuals. This information was recorded in all sorts of places, whether that be the COVID-19 testing sites that sprung up all over the place, or by hospitals, or any number of other professional medical institutions recording COVID-19 cases. As a result, any recording of newly infected individuals comes with an inherent voluntary and convenience bias. Infected individuals had to be willing to go and get tested in order for this information to be available, and this was not consistently the case. That being said, the combination of societal pressure and fear worked well to cause people to go, get tested, and then the information being subsequently recorded. Figure 5.1 shows the daily rates of newly infected over the course of four weeks for a single county.

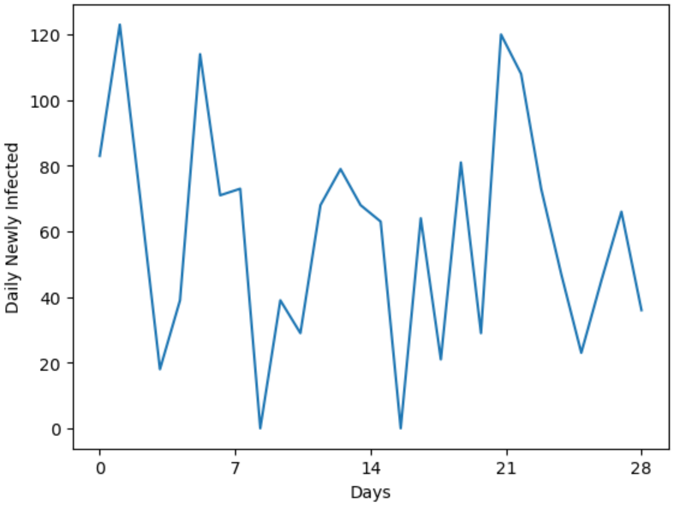

Figure 5.1: Daily New Cases for COVID-19.

A weekly pattern in the data is easily seen. Because individuals had to go somewhere to both get tested and recorded, this naturally caused a weekly cycle to develop. While this pattern in the data would be enough on its own to warrant use of a weekly timescale, another reason comes with the vaccination data. At around six months into the vaccination data, the choice was made to record vaccination rates on a weekly basis instead of a daily basis. Therefore, in order to use vaccination data, a weekly timescale is necessary.

With the choice of using a weekly timescale, the data were condensed into weekly data. Instead of dates being used, data were labelled based on how many weeks from the start of data collection, which was 3/13/2020. By examination of Figure 5.2, we can see that the Alpha wave took place between approximately week 10 and week 70.

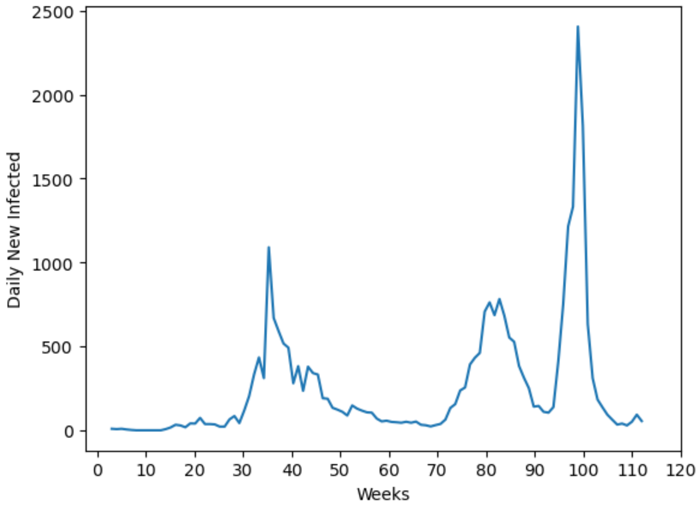

Figure 5.2: Daily New Infected for COVID-19.

Another choice that must be made is the choice of which compartment to fit to data in SIR model. As discussed in Section 4.2, we are fitting to the removed compartment. The data used are time-series, with the column headings detailing dates, counties, regions, hospitalization status, daily cases, cumulative cases, and other county information. The label of the column that we are interested in is 'Daily total cases.' Reading just the column name might lead one to believe that it is the total number of infected cases, however examination of the explanatory file and other columns shows that it is the number of newly infected. It is labels 'Daily total cases' because of three other columns that are labeled 'Recovered,' 'Deceased,', and 'Active' which refer to the counts and statuses of the cases that were newly recorded that day. A cumulative summation of this column gets us the number of removed cases per week which we can fit to our SIR model.

5.3 Algorithm Methodology

The data used for the initial analysis were from the Montana Department of Public Health and Human Services and were provided by Dr. Erin Languth of the University of Montana in the form of a .csv file. The data covered the daily cases of COVID-19 in Montana starting from March 13, 2020 up to April 26th, 2022. Data processing and modeling code were all done in Python by use of the various libraries. The appendices at the end of this dissertation contain the code used. We will discuss general code strategies here, and will leave it to the reader to examine the appendices at the end for greater details within the code.

The first major step we took in our code was creating the dataset that represented the removed compartment, as discussed in Section 4.2. To do this, we first found the weekly newly infected. After this, we cumulatively summed up the newly infected for each week, creating a dataset that represented the total number of removed cases each week, as discussed in Section 4.2.

To fit the SIR model to the removed dataset, a Trust Region Reflective algorithm was used for minimization [104]. For the Alpha wave, the best start and end weeks were unclear, and so all possible combinations were run through, with the Mean Squared Error (MSE) of each fit being used to determine the optimal start and end weeks.

The Delta and Omicron waves had clearly defined start and end weeks. Because of vaccination and its proximity to the Alpha wave, the Delta wave required all the infected people in the Alpha wave to be classified as already removed from the susceptible population. Because of the lack of immunity and ineffectiveness of the vaccines at the time, the Omicron wave did not require similar adjustments.

In order to scale the data properly to simulate city population density, city population and area were utilized as well as county population figures gathered from census data. The formulation for this is expressed in Equation 5.1.

$$\overline{r} = r \cdot (C/P)/A \tag{5.1}$$

In Equation 5.1, r is the total number of removed cases at a given week, with $\overline{r}$ representing the total number of removed cases in a square mile of a specific city at a given week. C and P represent the city and county populations, respectively. (C/P) is the ratio of city population to county population, thus being used to take into account the relationship of the city population to the county population. A represents the area of the city. The assumptions necessary for this are discussed in the next section.

For analysis of the case-fatality rates based on age demographics, data about COVID-19 cases from the official data hub for the Montana State Library was downloaded. After this the data were filtered based on county and age groups initially within appropriate time windows in order to get the total number of cases. After that, the data were additionally filtered based on whether the patients were deceased.

5.4 Assumptions

There are many assumptions being make with our methodology that must be addressed. The first assumption made is that unreported and asymptomatic infected individuals did not have an appreciable effect on the data. The SIR model that we are using is modeling the reported data. The implication of this is that there is a group of individuals who either were never reported or were asymptomatic, and thus their data were never recorded. In order to model this, we would need to make a second SIR model that run concurrently to the one modeling the reported data. In addition, a number of assumptions, about the number of asymptomatic and non-reported would have to be made. There is no data that allows for reasonable estimations of these. However, we can think of cases such as these in terms of underreporting. Underreporting is the phenomenon in which the number of cases is higher than what is reported. Therefore, asymptomatic and unreported cases would be a part of this underreporting factor. It has been shown that in the alpha wave of COVID-19 in Montana that the rate of underreporting was nearly 0%. While this does not hold for later waves, particularly Omicron, we will show the effects of underreporting on the model fitting and a way to account for underreporting.

We also make the assumption that every person who tested positive to COVID-19 was swiftly quarantined, whether in a medical institution or at home. Thus, they would likely have not further spread COVID-19 after testing positive and being recorded. As discussed above, given public policies and public perception, it is reasonable to assume such. In addition, this assumption allows us to treat the numbers of newly infected individuals as newly removed for the sake of the SIR model.

Another assumption that we make is that the newly infected people are homogeneously spread throughout a given county. This is done for two reasons. First, while we are examining a single town from each of the six most populous counties, those towns are not the only population centers within those counties, and since the data do not specify the town from which it was recorded, we cannot make assumptions about the distribution, and thus assume they are spread homogeneously. In addition, while 44% of the population of Montana is rural, it has been shown by Eilersen and Sneppen that infectious diseases spread far more in cities than in the county [103]. It is also reasonable to assume that while there were certainly cases of COVID-19 in the county, it is likely those infected individuals were not tested and thus were not recorded. In addition, it would be reasonable to assume that the number of cases of people from the county getting infected and coming to town to get tested is low enough to not significantly affect the data. For this reason, we can simulate the number of reported cases being in a specific town by multiplying by the ratio of that town's population and the county population.

5.5 Town Comparison Methodology

As discussed in the Introduction, we will be showing that the towns exhibit similar disease dynamics. In order to do so, we need a method of showing that the dynamics between the towns are similar. We assume that the fitted parameters α and β are normally distributed. We acknowledge that this assumption is incredibly naive. However, as there are only six towns with which we are working, we are making this assumption in lieu of further data. Therefore, for a given wave, we can compute the mean and standard deviation for these parameters and examine them. Furthermore, we can also do this with the basic reproduction number R_0. As discussed in Section 2.5, R_0 is a measure of the severity of an epidemic. We will compute R_0 as follows:

$$R_0 = \frac{\alpha N}{\beta} \tag{5.2}$$

We can see from (5.2) that R_0 can be utilized as an expression of the combination of the disease and population dynamics for the disease. However, we note that while N in (2.9) is the total population, as explained in Section 4.3, we are utilizing city population density. Because of this, when computing R_0 later, we will use the city's population density in N.

Chapter 6

Data Fitting Results

6.1 Alpha Wave

Here, we present and discuss the SIR model fit for the six most populous counties in Montana for the Alpha wave. The analysis of the Alpha wave is presented for two cases, the first being the base case in which no scaling is done. The window of data being used for this is from week 20 to week 60. In Figure 6.1 we can see the new cases for the different counties superimposed upon each other.

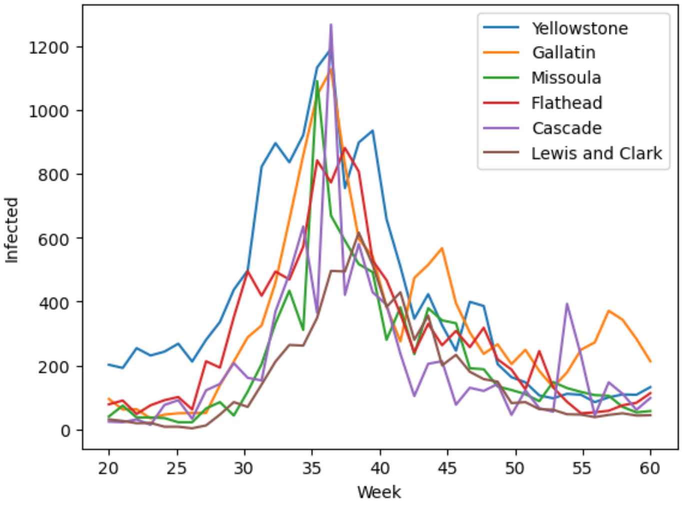

Figure 6.1: Total Cases for Alpha Wave Without Any Scaling.

With the SIR model, we can fit the removed dataset. The results of these fits is illustrated are Figure 6.2.

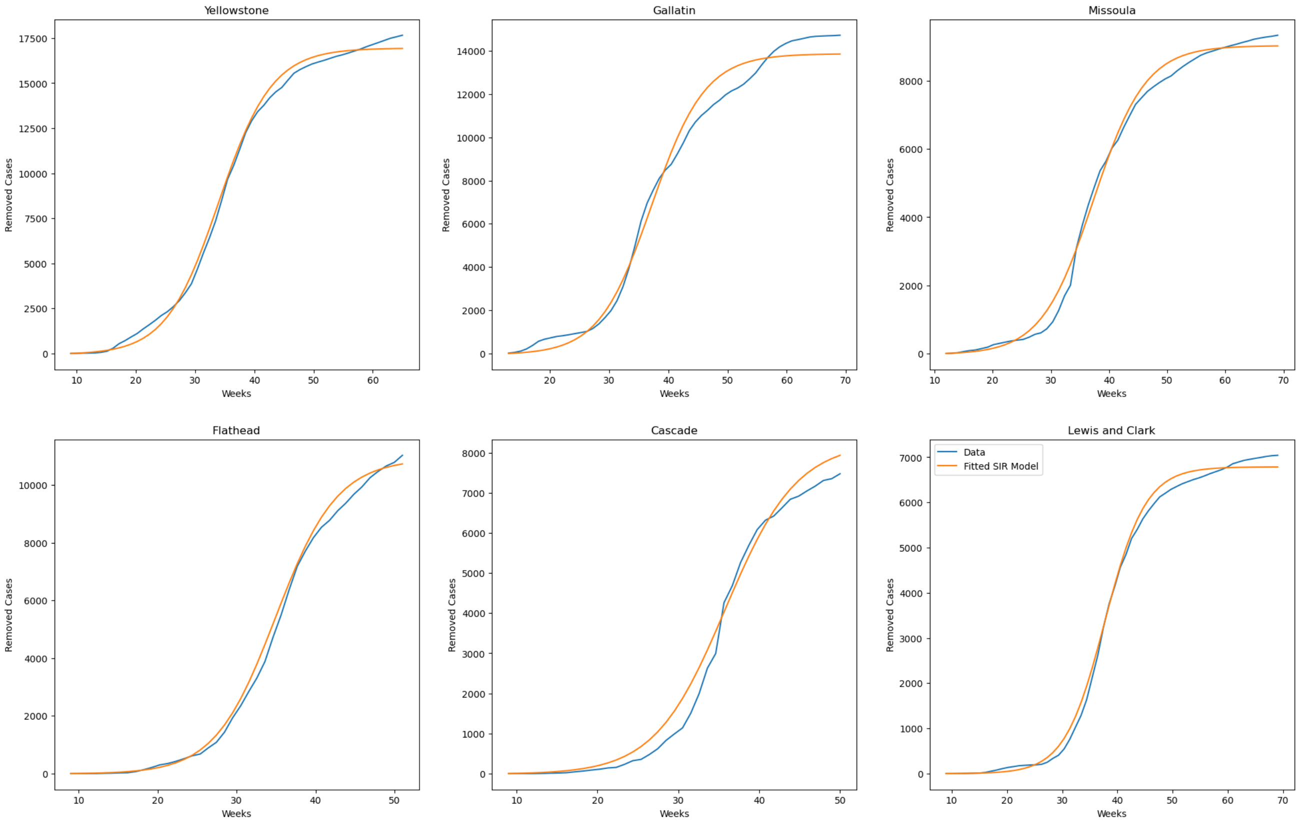

Figure 6.2: Alpha Wave Removed Fits

Table 6.1 contains the estimated model parameter values for the different counties.

Parameters	**Yellowstone**	**Gallatin**	**Missoula**	**Flathead**	**Cascade**	**Lewis and Clark**
Start Week	10.0	15.0	13.0	10.0	10.0	10.0
End Week	66.0	70.0	70.0	50.0	50.0	61.0
α	0.000026	0.000032	0.000049	0.000047	0.000055	0.000089
(SE)	0.000001	0.000002	0.000004	0.000001	0.000003	0.000003
β	4.155278	3.712655	5.658801	4.862012	4.4042	5.533329
(SE)	0.104227	0.250737	0.43881	0.155649	0.28247	0.175168
Initial infected	3.24176	3.471051	1.197023	0.654905	0.895353	0.118342
(SE)	0.845416	0.422385	0.098124	0.368289	0.09965	0.039753
City	Billings	Bozeman	Missoula	Kalispell	Great Falls	Helena

Table 6.1: Alpha Wave Fitted Parameter Values

Let us discuss and interpret the meaning of the model parameters and the estimated parameter values in this context. The parameter α in equations (2.6)-(2.9) represents the appearance of a newly infected individual per time unit (a week in this case) and also per presently infected individual that is intermingling with the susceptible population (in other words is not currently quarantining). In the case of Missoula, the characteristic time associated with the appearance of a newly infected individual during the Alpha wave, computed via $1/(\alpha \cdot N)$, is approximately 0.23011 or 1.61 days. Now, note this is averaged over the course of 56 weeks. The parameter β, as stated previously, is the reciprocal of the length of time it takes for removal of an individual from the pool of infected individuals mingling with the susceptibles, whether this be due to isolation, quarantining, or recovery.

The fitted β values exhibit an interesting pattern. As discussed in Section 4.2, we are making an assumption that individuals when diagnosed with COVID-19 swiftly quarantine themselves. We can see from the fitted β values that this assumption is validated. Given that β is the reciprocal of the length of time of recovery/removal, and the fact that our time intervals are weeks, it can be seen in the case of Missoula to be 1.237 days. While the numbers are different for different counties, there is a pattern of length of time for removal being less than the average length of time of sickness from COVID-19, given that the Alpha variant had on average around a two week lifespan for a patient [105].

We have included the standard errors (SE) for each parameter fit below the corresponding fit. These standard errors are computed via the covariance matrix as generated by the Moore-Penrose pseudoinverse [106]. The standard errors are a measure of the uncertainty of our parameter fit, and cannot be used to contrast the parameters for separate counties with each other. We can see that the uncertainty for our parameter fits is not unreasonable.

6.1.1 Scaled Alpha wave analysis

For the second case of Alpha wave analysis, we consider several scaling factors. As discussed in Section 4.3, we will be utilizing city population density instead of straight population and scaling the data to reflect that. However, for the Alpha wave, we will show other scaling factors as contrast to city population density.

As seen in Figure 6.1, we have the new cases from the top six counties for the Alpha wave. If we consider this in terms of population density across the county, we get Figure 6.3, part A.

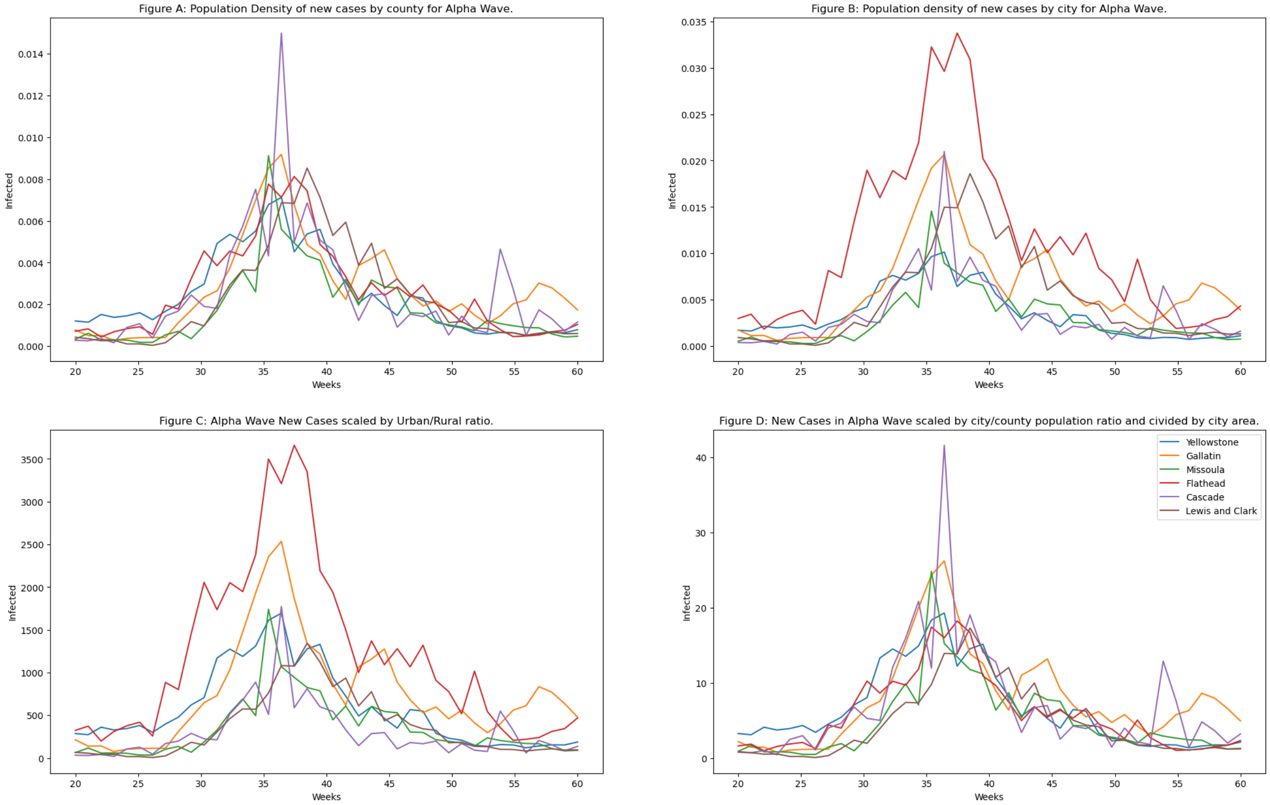

Figure 6.3: Alpha wave newly infected scaling method comparison.

Here we have simply divided the data by the total area of each county, thus examining the data in terms of people per square mile. However, an issue with this is that the majority of the people in most counties, and the majority of reported cases, are located in the largest towns within each county. Thus, it is more sensible to divide by the area of the largest city in the county instead of county area. The results are shown in Figure 6.3, part B.

The issue with this, however, is the inherent assumption that all the cases within the county are taking place within the confines of the given city. While the majority of those being recorded would be, we must account for the existence of other towns within the county as well as cases that are outside the city. Therefore, as discussed in Section 5.4 we assume that infections are distributed homogeneously across the population of a county, and thus we scale the data by the ratio of the population of the largest city and the county. The result of this approach is shown in Figure 6.3 part C.

Now let us combine these ideas together. First, since the Law of Mass Action, and thus by extension the SIR model, assume a homogeneous mixture, we need to concentrate our focus into areas where people are going to be mixing as much as possible. These places are, naturally, cities, and thus scaling the data by the city-county population ratio is the first step. Second, once again, since the Law of Mass Action and SIR models assume mixing of densities instead of just straight populations, we should also transform the data into measurements of density. The result of all this is seen in Figure 6.3 part D.

This is an interesting result, as looking at population density and having been scaled by the city-county population ratio is acting as a normalizing factor. If we look at Figure 6.3 part D, it seems to show that if we focus on purely the city population densities, the population dynamics between the different cities follows a similar path.

As stated previously, examining the removed dataset is preferred. This can be seen in Figure 6.4.

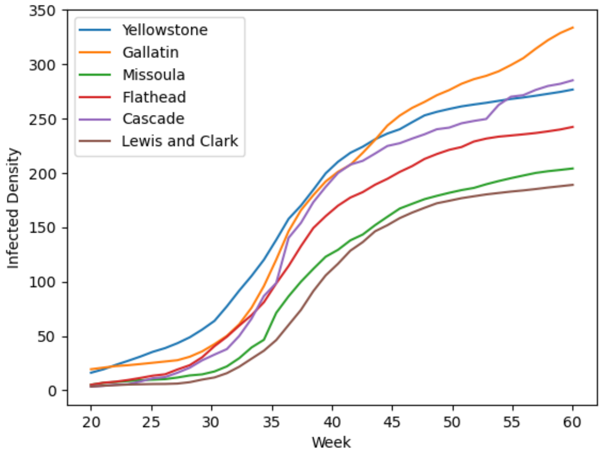

Figure 6.4: Alpha Wave removed data scaled by city/county population ratio and divided by city area.

While this may not appear as compelling as Figure **??**, one can see the difference when compared with Figure 6.5.

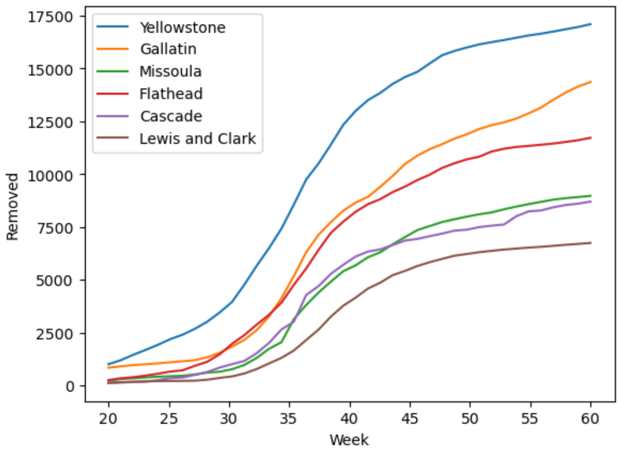

Figure 6.5: Alpha Wave removed data without scaling.

Now with this scaling in place, we can run the SIR model fit again. The results are presented in Figure 6.6.

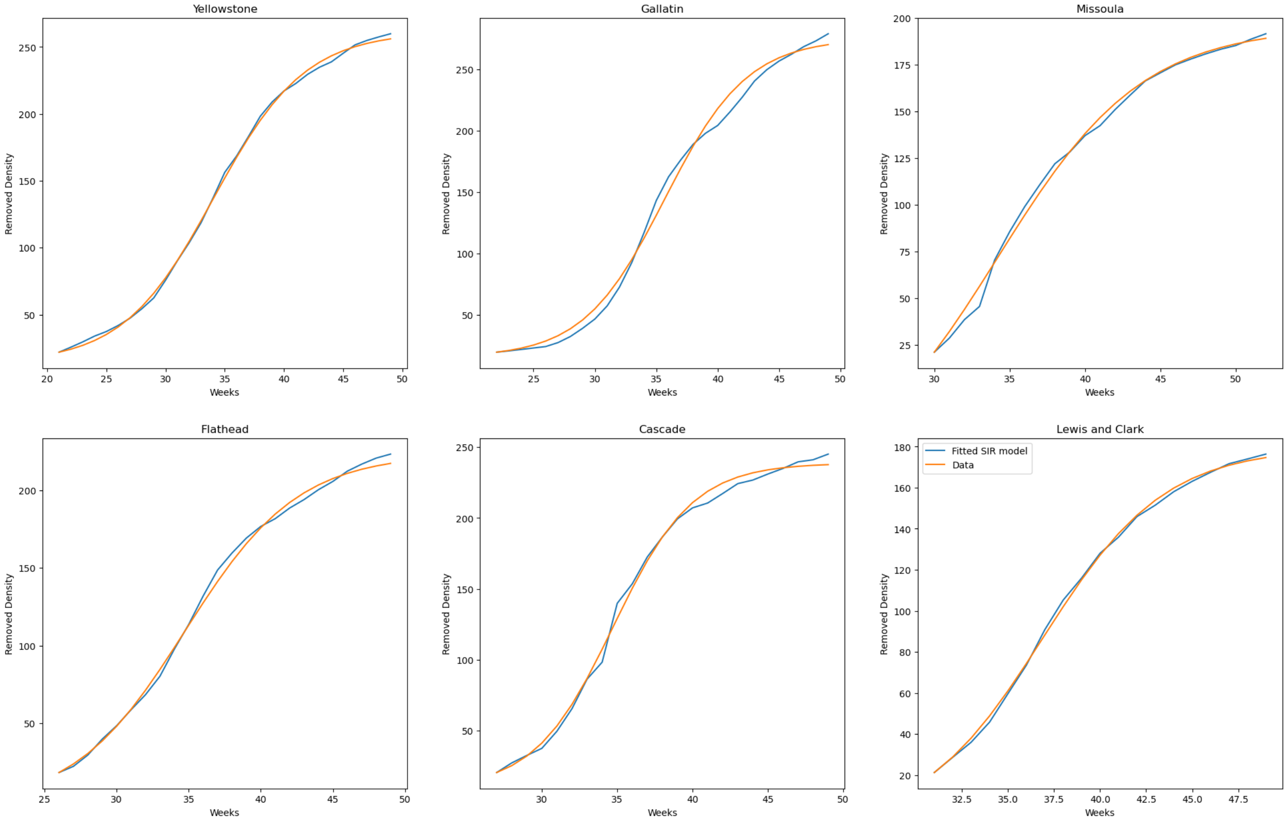

Figure 6.6: Alpha Wave SIR models fitted to removed data scaled by city/county population ratio and divided by city area.

While these may not look too different, the interesting part comes in the fitted parameter values included in Table 6.2.

Parameters	**Yellowstone**	**Gallatin**	**Missoula**	**Flathead**	**Cascade**	**Lewis and Clark**
Start Week	21	22	30	26	27	31
End Week	50.0	50.0	53.0	50.0	50.0	50.0
α	0.002098	0.002301	0.001707	0.002271	0.003314	0.003294
(SE)	0.00061	0.000177	0.000284	0.000158	0.000225	0.000205
β	5.382404	6.222861	4.514582	4.838584	8.770344	6.378984
(SE)	0.157993	0.482194	0.740162	0.33681	0.596111	0.395579
Initial infected	0.367672	0.194525	2.352696	1.026875	0.464807	1.067178
(SE)	0.029325	0.04816	0.558906	0.144611	0.084453	0.119168
R_0	1.0547621	1.0547409	1.0283826	1.0528166	1.0474591	1.0460312
City	Billings	Bozeman	Missoula	Kalispell	Great Falls	Helena

Table 6.2: Alpha Wave fitted parameter values with city/county population and city area scaling.

A visual inspection of the α's and β's for the 6 counties suggest a high degree of similarity between then. However, as discussed in Section 5.5, we assume that α and β are normally distributed. We can thus compute the mean and standard deviation for α, β, and R_0 across the six counties. We will also show the intervals that are within one and two standard deviations of the computed mean for each of these. All of these are shown in Table 6.3.

Parameter	μ	σ	$\mu \pm \sigma$	$\mu \pm 2\sigma$
α	0.0024975	0.0006596	(0.0018379, 0.0031571)	(0.0011783, 0.0038167)
β	6.0177932	1.5365036	(4.4812895, 7.5542968)	(2.9447859, 9.0908004)
R_0	1.0473654	0.0100096	(1.0373558, 1.0573750)	(1.0273462, 1.0673846)

Table 6.3: Alpha wave fitted parameter values with city/county population and city area scaling statistical analysis.

From Table 6.3 we can see that most fitted parameters are within one standard deviation of the mean, with all of them being within two standard deviations. As a reminder, the data used for these fits have been scaled in such a way that we are examining just the population densities of the infected populations in the six towns. From this, a few details can be elucidated. Most importantly, in spite of demographic differences, when it comes to COVID-19, at least during the Alpha wave, the six towns exhibited approximately the same patterns. Now, a few caveats must be stated. These towns are all in the same state, and in the same approximate geographic location in the United States. Thus, similarities will naturally exist in the culture and population dynamics between the six towns. However, with the evidence of the fitted parameters, it can be stated that in spite of the towns being disparate entities, they exhibit similar disease propagation dynamics when it comes to COVID-19.

Rural towns having similar patterns during a disease outbreak is an important point to make. Oftentimes when developing countermeasures to disease outbreaks, separate counties and towns will have separate policies and measures in place to counteract the spread of infection. In regard to the present study, this would imply six different counties and towns independently developing different countermeasures for the outbreak. If the overall population dynamics is similar between the towns, then separate development of countermeasures is unnecessary. Effective countermeasures for one city will be similarly effective in another. However, in order to properly show this, we need to look at the other waves.

6.2 Delta Wave

Let us now discuss the fitting of the data for the COVID-19 Delta wave to the SIR model. One thing to note is that the model for the Delta wave necessitates modification. While one might be tempted to treat the Delta wave similarly to the Alpha wave, they simply cannot. As stated previously, the Delta wave had additional features of both vaccination and prior immunization. Now, while neither immunization nor vaccination feature 100% removal, they still must be taken into account in the model fitting.

For prior immunizations, it is known that infection of COVID-19 granted a level of immunity for several months afterward [107]. In order to account for this, the prior infected/removed must be counted as already removed from the susceptible population. There are multiple studies that suggest various figures for continuing immunity, anywhere from early figures of three months to later figures of 500 days. In addition to this, several studies were done on the effectiveness of 'hybrid immunization,' in which people not only contracted COVID-19, but also received the vaccine, leading to longer periods of immunization. For the sake of simplicity, people who were either previously infected or vaccinated are removed from the pool of susceptibles for the initial time point specified in the model. While this is done simply via data manipulation, in order to account for vaccinations happening during the Delta wave, we have to also adjust the model itself.

As a reminder, the SIR model contains three compartments labelled S, I, and R, which are short for the following: Susceptible (the population who are at risk of being infected by the disease), Infected (the population spreading the disease), and for our purposes Removed (the population who is neither at risk of contracting the disease nor spreading it). The Removed population not only includes people who either previously contracted the disease or were vaccinated, but also includes people who quarantined themselves. In order to model this, we need to introduce a vaccination factor. Using data from the CDC vaccination tracker, we know that the time dependent vaccinations in different counties has followed an approximately linear path. We can see this illustrated in Figure 6.7.

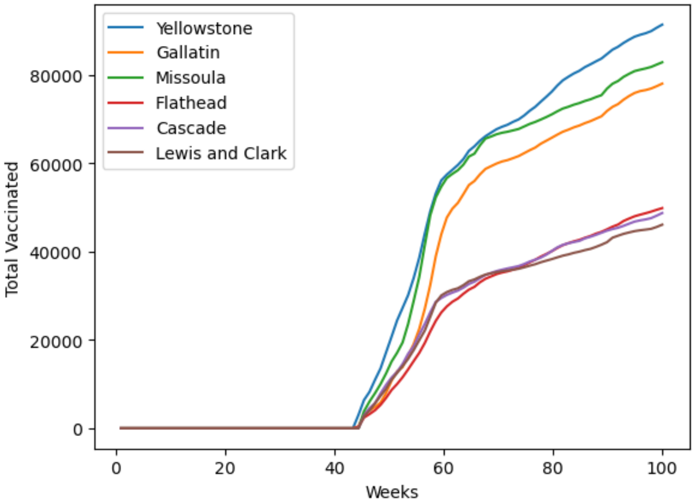

Figure 6.7: Total vaccinations throughout all waves.

We can see from Figure 6.7 that there are approximately three phases of the vaccination. Prior to weeks 43-45 there was no vaccinations, as the vaccine had yet to be developed. Up until around week 60 there was a rapid deployment of vaccines, after which the rate of vaccine deployment slowed down. However, in each stage, the rate of deployment followed an approximately linear rate, allowing them to be modeled with a linear function. Let us introduce a rate of vaccination γ, which is measured in vaccines distributed per week. As a result, the modified SIR model is as follows:

$$\frac{dS}{dt} = -\alpha \cdot S \cdot I - \gamma \tag{6.1}$$

$$\frac{dI}{dt} = +\alpha \cdot S \cdot I - \beta \cdot I \tag{6.2}$$

$$R(t) = N - S(t) - I(t) \tag{6.3}$$

γ is measured as number of vaccinations per time period (in our case, weeks). Since the data to which we fit are scaled, we will also scale γ such that it represents vaccinations per square mile per week within the town of interest. As with α and β, γ is separate for each city. In addition, γ is not a parameter that needs to be fit. As we have the actual data for vaccination rates, we can directly compute γ. While it's not a perfect fit, since the data are approximately linear, we can approximate the rate of vaccination with a high degree of accuracy. In addition, certain assumptions are made here. First, we assume that the vaccinations are homogeneously spread throughout the population, across all demographics and between the city and rural populations. While this is a somewhat simplistic assumption, it is necessary, as information regarding demographics and residence is not available for vaccinations. In addition, we know for certainty that the distribution of vaccines was not homogeneous across demographics, as the FDA had a rolling timeline of approval of administration to progressively younger age groups. However, knowing what the true distribution was is an impossibility as this naturally comes from the manner in which many vaccination clinics were carried out, in which minimal paperwork was done, and while insurance information was taken, the data are not available in public datasets.

In addition to the assumption of homogeneous distribution of vaccines throughout the population, assumptions of the effectiveness of the vaccine are necessary. As is fairly well known, three major vaccines were developed for COVID-19. Named after the companies that developed them, the Pfizer, Moderna, and Johnson and Johnson, vaccines were made widely available to the public. While there is a fourth vaccine that is available currently, developed by Novavax [108], it was not approved for adult use by the FDA until July 2022 [109]. A point of interest here as well is that the Pfizer and Moderna vaccines, known as Comirnaty and Spikevax respectively, required a series of two shots for effectiveness. Unlike these two, the Johnson and Johnson vaccine was a single dose. In the case of modified SIR model (6.1)-(6.3), it is assumed that the vaccine is fully effective, or that once someone has been vaccinated, they are fully immune, and are thus removed from the susceptible population pool. Now, there are plenty of papers and studies done to show that the vaccines were not fully effective. However, for the sake of the model this assumption is being made. In addition, while Comirnaty and Spikevax each require a series of two injections, weeks apart, only the incidences of the first vaccinations are taken into account, since many people who got the first dose also got the second dose. While there was a noticeable percentage of people who got the first dose and chose to not get the second dose, we are making the choice to assume that everyone who got the first dose also got the second dose (and immunity).

While not an assumption in the same manner as above, we acknowledge the fact that a constant rate of vaccination is untenable for perpetuity, as eventually the number of susceptibles will eventually be negative. We would note that in this case, we are building the model off of existing data, rather than attempting to make predictions into the future. In addition, we are working in a given time frame for which the number and rate of vaccinations is known. Because of the limited time frame, we will not have to deal with a negative susceptible population. We should also note the fact that vaccinations are not immediately effective, as the body takes time to develop the immunity. However, examination of Figure 6.7 shows that this is not a concern. While common estimates for the length of time it takes for the vaccine to be effective are around 1-2 weeks [110] even if we allow for 4 weeks before effectiveness, we can see that the rate of vaccination is approximately linear. Therefore, we do not need to take into account a change in vaccination rates in our model. With this new model and the acknowledged assumptions, let us look at the data for the Delta wave as shown in Figure 6.8.

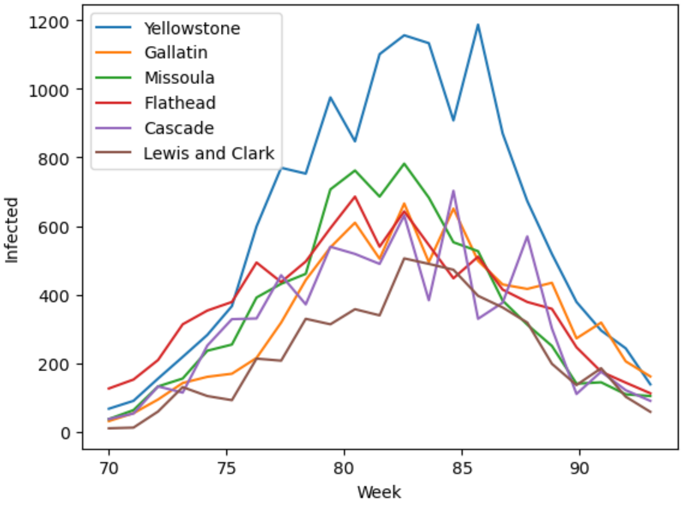

Figure 6.8: Delta wave newly infected cases.

If we apply the same scaling procedure to the Delta wave as we did in the case of the Alpha wave, we get the re-scaled data shown in Figure 6.9.

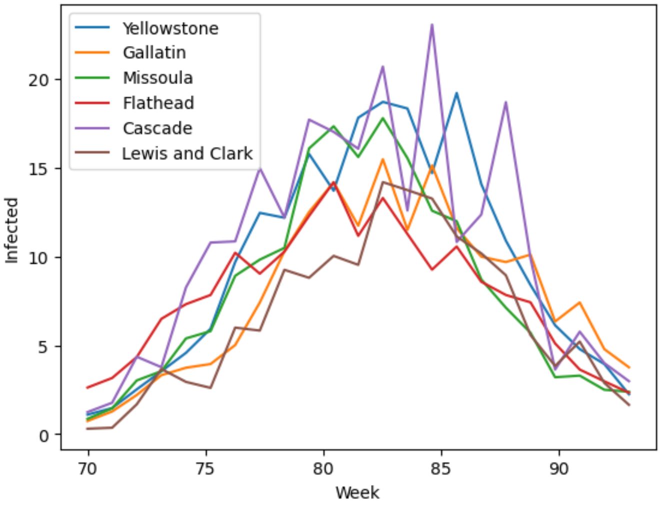

Figure 6.9: Data for Delta wave scaled by city/county population ratio and divided by city area.

While not as striking of a change as the Alpha wave, as illustrated by Figures 6.1 and **??**, it can be seen that the scaling of the data shows similarities in the population dynamics of the disease. There are a couple of additional details about the Delta wave that should be discussed. First, it can be seen that the Delta wave was significantly shorter than the

Alpha wave. While studies have shown that not all the people who were infected during the Delta wave were infected with the Delta variant [111], without widespread testing it will have to be assumed that the effect of such a group of people is negligent to the dynamics of the larger population. As a final note of observation, the endpoint of the Delta wave is fairly abrupt, as the Omicron wave began in the latter stages of the Delta wave. For this reason, the newly infected in the Delta wave had not fully flattened out yet. Table 6.4 has the SIR model parameter fits to the Delta wave without the vaccination constants being used. From a model standpoint, we are using Equations (2.6)-(2.9) for this model fit instead of (6.1)-(6.3).

Parameters	**Yellowstone**	**Gallatin**	**Missoula**	**Flathead**	**Cascade**	**Lewis and Clark**
Start Week	73	73	73	73	73	73
End Week	93	93	93	93	93	93
α	0.003673	0.004355	0.006439	0.004559	0.003213	0.008558
(SE)	0.000079	0.00013	0.000071	0.000071	0.000116	0.000121
β	9.517167	12.009187	16.874758	9.815223	8.506713	16.617304
(SE)	0.206442	0.359141	0.185376	0.15385	0.308034	0.236585
Initial infected	0.32021	0.163241	0.084976	0.295204	0.446264	0.02944
(SE)	0.056364	0.05767	0.02429	0.065652	0.175276	0.024723
City	Billings	Bozeman	Missoula	Kalispell	Great Falls	Helena

Table 6.4: Delta wave fitted parameter values.

As with Table 6.2, α and initial infected is measured in population per square mile, and $1/\beta$ is measured in weeks. While the values for α and β appear compelling, as stated prior to Table 6.4, this set of fits was done without taking vaccination into account. Thus, any statistical similarities are meaningless, as the fitted parameter values are inaccurate. In addition, we can tell there is an issue as the values for β are a bit beyond reasonable expectations. The results of these fits are illustrated in Figure 6.10.

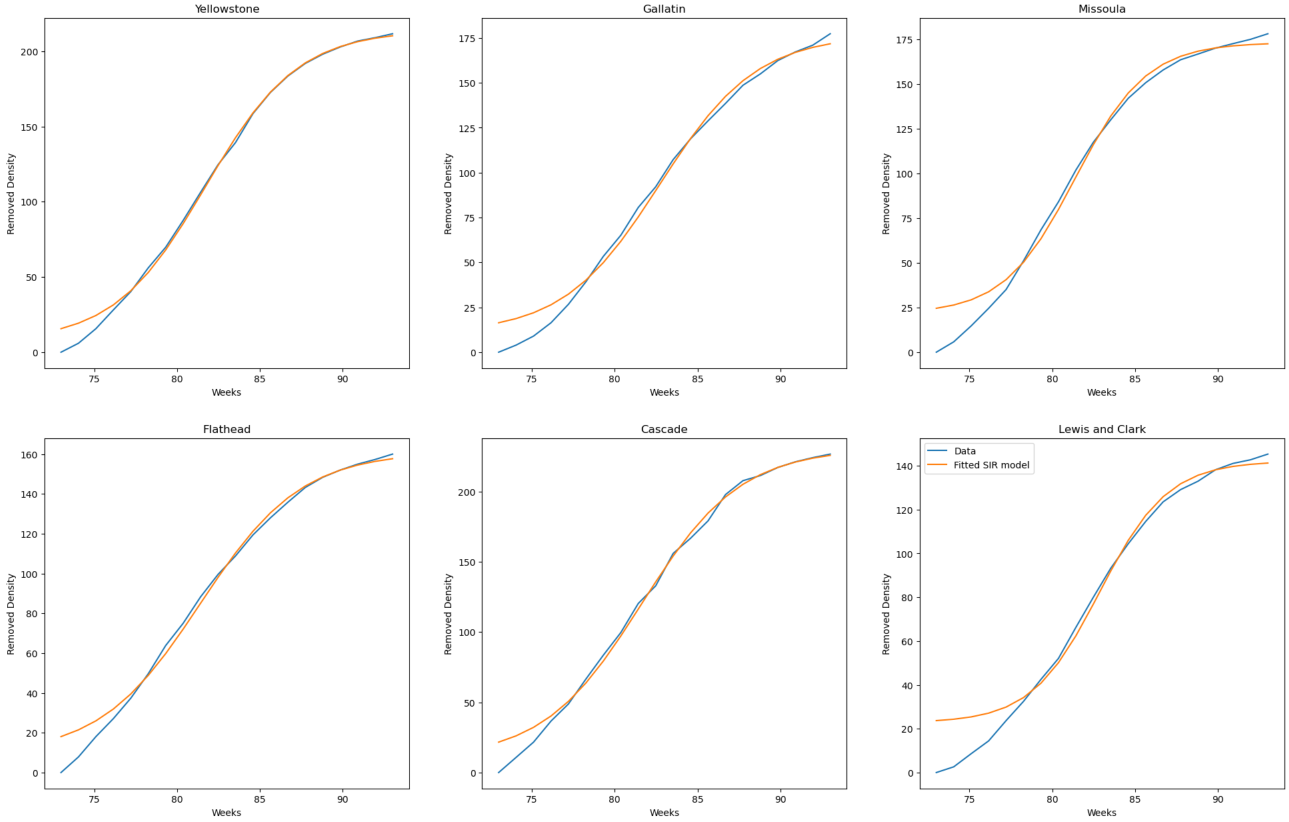

Figure 6.10: Delta Wave non-vaccination SIR models fitted to removed data scaled by city/-county population ratio and divided by city area without.

It can be seen from Figure 6.10 that the fits are poor. While the fits from weeks 78 to 89 follow the data fairly well, outside these weeks the model tends to deviate from the data fairly dramatically. This is to be expected, as shown in Figure 6.7, vaccinations began between Weeks 40 and 45, implying vaccination administration for around 30 weeks prior to the Delta wave. Since we are assuming that a single dose of the vaccine is fully effective, the vaccinated population must be taken into account. Since the removed data we are using is based off of newly infected, we cannot include the vaccination data into the removed data. However, since vaccinated individuals are being treated as being no longer susceptible, Equations (6.1)-(6.3) can be used to model this. Utilization of the modified SIR model shown in (6.1)-(6.3) is illustrated in Figure 6.11.

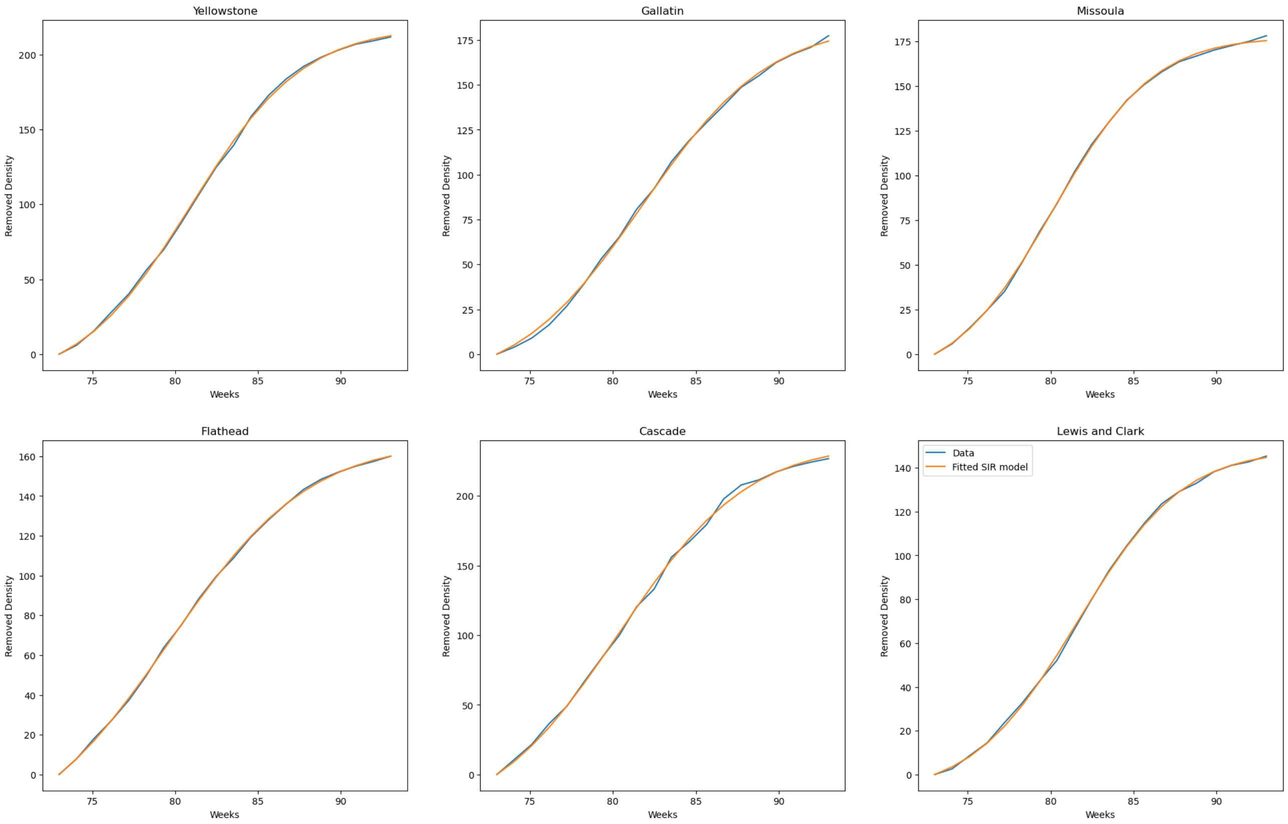

Figure 6.11: Delta Wave vaccination SIR models fitted to removed data scaled by city/county population ratio and divided by city area without.

Parameters	**Yellowstone**	**Gallatin**	**Missoula**	**Flathead**	**Cascade**	**Lewis and Clark**
Start Week	73	73	73	73	73	73
End Week	93	93	93	93	93	93
α	0.001331	0.001192	0.001859	0.001171	0.000968	0.002268
(SE)	0.000042	0.0000065	0.000052	0.000035	0.00006	0.000066
β	3.316738	3.146865	4.741428	2.458812	2.472195	4.263643
(SE)	0.10342	0.171934	0.132995	0.07272	0.151178	0.124938
γ	14.503305	14.982544	12.591779	11.515052	15.889752	8.94272
Initial infected	1.758827	1.439025	1.112842	2.962869	3.44218	0.693541
(SE)	0.109974	0.147279	0.065486	0.139756	0.365281	0.045052
R_0	1.0857042	1.0804817	1.0663725	1.0682808	1.0854101	1.0775434
City	Billings	Bozeman	Missoula	Kalispell	Great Falls	Helena

Table 6.5: Delta wave vaccination SIR model fitted parameter values.

Figures 6.10 and 6.11 illustrate the importance of accounting for vaccinations. With the vaccination constant γ being used, we have excellent fits. Table 6.5 has the fits with the vaccination rates. α and initial infected are measured in population per square mile, while $1/\beta$ is measured in weeks. γ is computed directly from the vaccination data, and is in population per square mile. As with the removed data, this vaccination rate is scaled by the ratio of city and county population, then divided by the corresponding city area to get a population density. The parameter γ, then, represents the population per square mile being vaccinated per week within a given city.

As with the Alpha wave, we can examine α, β, and R_0 via their means and standard deviations. We do not include γ in this analysis as that is neither a fitted parameter value, being directly computed from vaccination data, nor is it a natural process. The means, standard deviations, as well as ranges one and two standard deviations from the means are shown in Table 6.6.

Parameter	μ	σ	$\mu \pm \sigma$	$\mu \pm 2\sigma$
α	0.0014648	0.0004952	(0.0009697, 0.0019600)	(0.0004745, 0.0024552)
β	3.3999468	0.9341280	(2.4658189, 4.3340748)	(1.5316909, 5.2682027)
R_0	1.0772988	0.0083348	(1.0689640, 1.0856336)	(1.0606293, 1.0939683)

Table 6.6: Delta wave vaccination SIR model fitted parameter values statistical analysis.

Similarly to the Alpha wave, we can see from Table 6.6 that most fitted parameters are within one standard deviation of the mean, with all of them being within two standard deviations. With vaccinations being accounted for, we can see that the dynamics illustrated by α and β show similarities across the six cities. Comparing the results shown in Table 6.5 with those in Table 6.2 shows some interesting patterns. Except for Missoula, the values of α and β is lower for the Delta wave in comparison to the Alpha wave. From this it can be concluded that with vaccinations, the Delta wave did not spread as readily as the Alpha wave, and that people did not quarantine themselves as quickly. It should be noted, however, that if vaccinations are taken into account, the α and β values for Missoula hardly changed. While some reasons for this might be theorized, there is a lack of data to be able to make reasonable guesses for reasons.

6.3 Omicron Wave

Let us now discuss fitting the SIR model to the Omicron wave data. As stated previously, while the Alpha and Delta waves both contained immunization and vaccination components, the Omicron wave appeared to have none. Studies at that time showed that prior vaccinations had little effect against the Omicron strain [112]. Therefore, the Omicron wave can be treated much in the same manner as the Alpha wave. Since prior vaccinations and immunities were ineffective, we get something of a clean slate again.

With Figure 6.12 and Figure 6.13, we can see a similar pattern as to the previous two waves, with scaling causing most of the counties to exhibit similar disease dynamic patterns to each other.

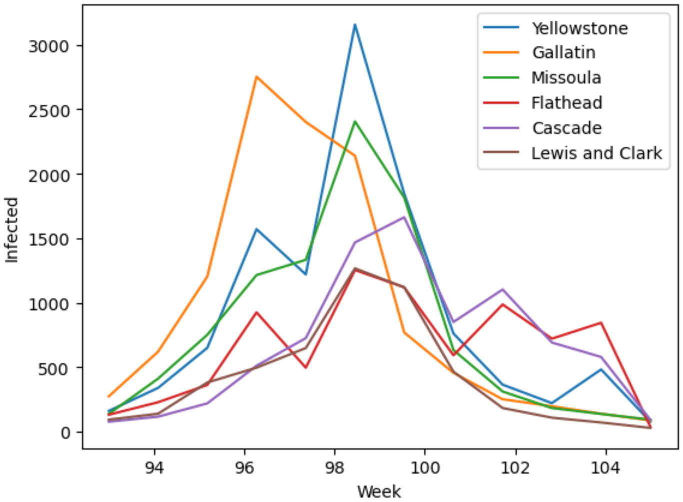

Figure 6.12: Newly infected cases data for Omicron wave.

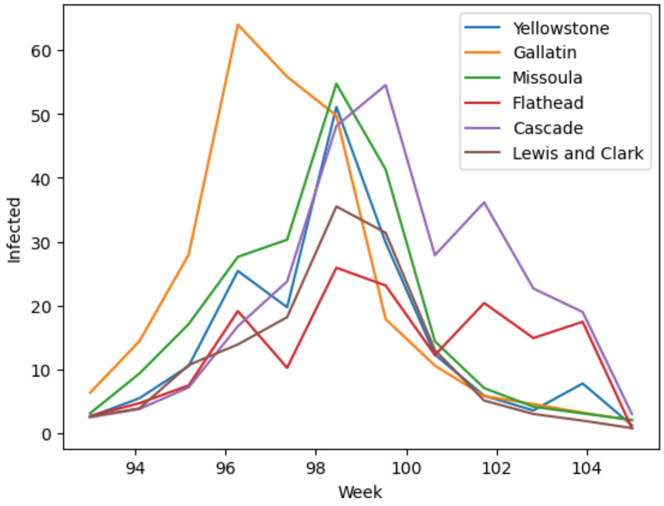

Figure 6.13: Newly infected case data for Omicron wave scaled by city/county population ratio and divided by city area. Units of infected are people/mi^2

Once again, we can see that with scaling the data by city/county population ratio and dividing by city area, most of the cities follow a similar pattern. The one difference is Gallatin, or the city of Bozeman. This difference can be explained, at least in part, by the existence of Montana State University in Bozeman. Given that the Omicron wave happened at the beginning of the spring semester when students were returning, it is reasonable to assume that the large boom in cases is a result of students getting infected. In fact, looking at the chart of total cases during the Omicron wave lends credence to this hypothesis.

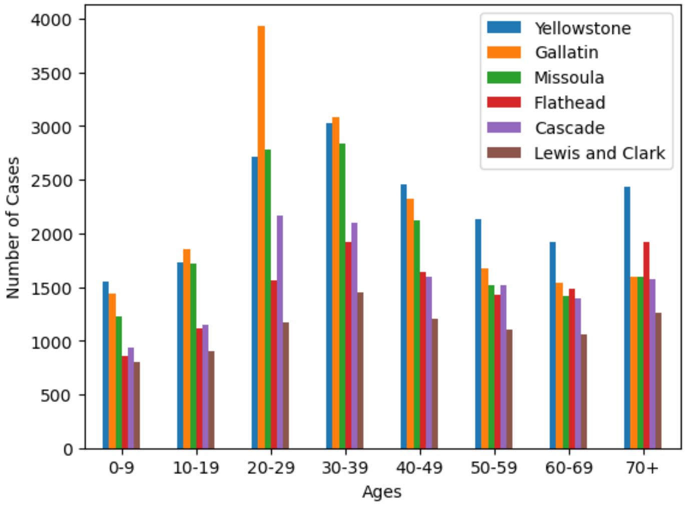

Figure 6.14: Omicron Wave Total Cases by Age Demographic

From this it can be observed that a large number of the cases in Gallatin were in the 20-29 age range, thus a hypothesis of the school having an effect seems reasonable.

Out of the three waves, the Omicron wave was certainly the shortest. In comparison with the previous two waves, Omicron had a much greater spike in new infected cases. A similar note to the Delta wave is that while most people who were infected during the Omicron wave were assailed with the Omicron variant, there is a certain portion of the population that was infected with the Delta variant during this time. Like with the Delta variant, without widespread testing, we are forced to assume that the disease dynamics are mainly dominated by the Omicron variant as opposed to the Delta variant. To this end, we initially fit the Omicron wave data to the SIR model outlined in Equations (2.6) - (2.9). With these assumptions, we have the model parameter value fits for the Omicron variant shown in Table 6.7.

Parameters	**Yellowstone**	**Gallatin**	**Missoula**	**Flathead**	**Cascade**	**Lewis and Clark**
Start Week	93	93	93	93	93	93
End Week	106	106	106	106	106	106
α	0.010292	0.006766	0.008654	0.00558	0.00514	0.01366
(SE)	0.001018	0.000548	0.000606	0.00073	0.000339	0.000919
β	27.003324	18.55421	22.665828	12.09627	13.562876	26.749617
(SE)	2.665465	1.485763	1.583405	1.579629	0.895934	1.796784
Initial infected	0.250403	1.324331	0.378248	4.967668	0.471008	0.155178
(SE)	0.072164	0.241333	0.077189	0.168815	0.090899	0.033609
R_0	1.0436562	1.0576192	1.2846941	0.8653000	1.0610880	1.0525638
City	Billings	Bozeman	Missoula	Kalispell	Great Falls	Helena

Table 6.7: Omicron wave fitted model parameter values.

The model used for the Omicron wave is the original SIR model given in Equations (2.6) - (2.9). As with Tables 6.2 and 6.5, α and Initial infected are population densities, measured in people per square mile, and $1/\beta$ is measured in weeks. The fits of the SIR model to the data are shown in Figure 6.15.

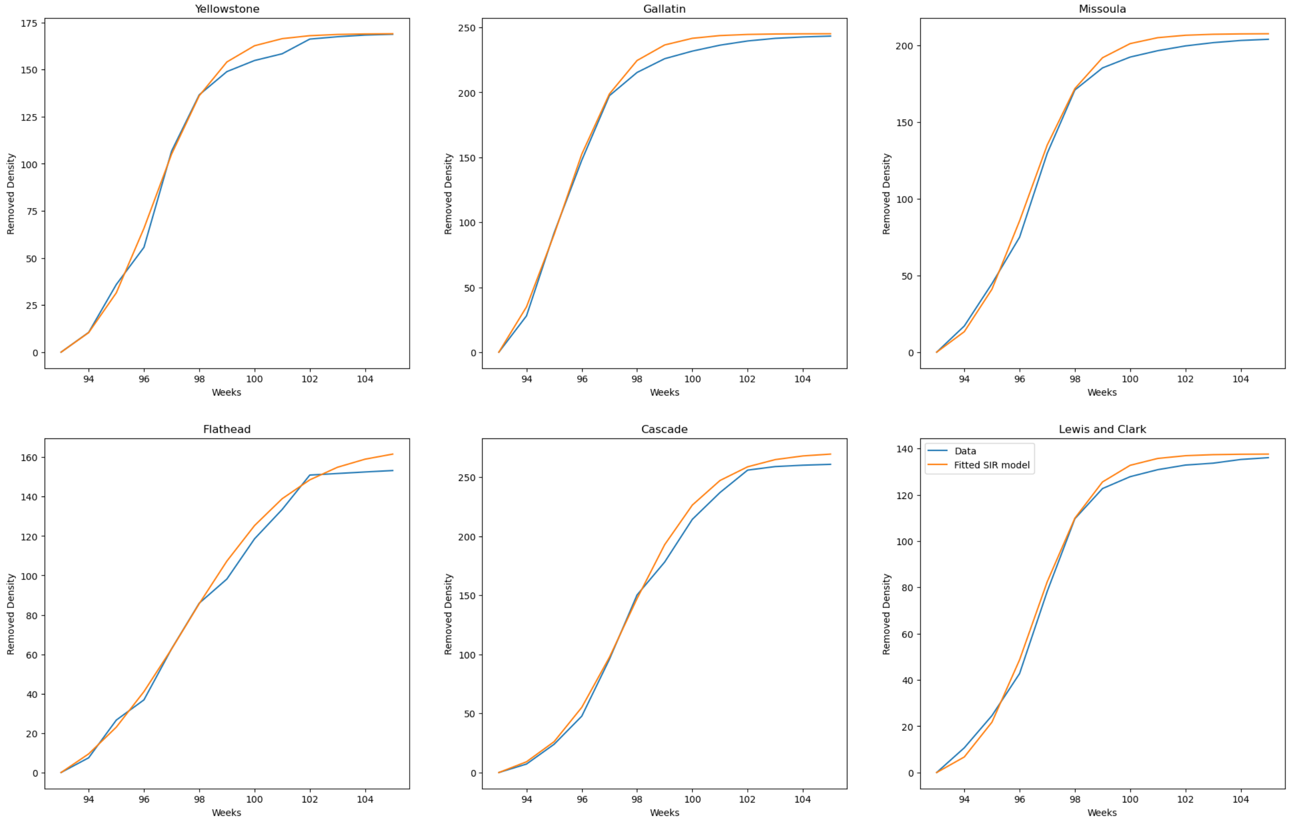

Figure 6.15: Omicron wave removed data scaled by city/county population ratio and divided by city area fitted to SIR model.

While the graphical fits of the fitted model parameters to the data appear reasonable, examination of the actual fits themselves show issues. As a reminder, the reciprocal of the fitted parameter β represents the average time that it takes for an infected individual to be removed from interaction with the susceptible population, or in the case of COVID-19, quarantine. Taking that into consideration, the fitted β parameter values shown in Table 6.7 suggest that individuals were quarantining themselves within hours after recording. There was not a large enough shift in population dynamics from the Alpha and Delta waves to the Omicron wave to support such a shift in fitted parameter values.

The issue that we see arising from this is the issue of underreporting. As discussed in Section 5.4, while the Alpha wave had little underreporting, underreporting was much higher during the Omicron wave [113] [114]. While vaccinations did not have a factor in the Omicron wave, underreporting did, and thus it must be accounted for.

6.3.1 Modified SIR model for underreporting.

In order to account for underreporting, we define parameter k to represent the proportion of infections that were reported as cases. These are people who both got infected and also had their infections recorded and thus reported. We then consider the case in which the disease dynamics are accurately modeled by Equations (2.6) - (2.9). Let us assume that for the length of the Omicron wave, that the proportion of reported cases, $0 \leq k \leq 1$, is constant. This means that the available data represent $k \cdot R$ instead of R. We then rescale our variables such that $\overline{S} = k \cdot S$, $\overline{I} = k \cdot I$, $\overline{R} = k \cdot R$. We assume here that the proportion k of reporting is the same among the three compartments. The modified SIR model is written as follows:

$$\frac{d\overline{S}}{dt} = -\alpha/k \cdot \overline{S} \cdot \overline{I} \tag{6.4}$$

$$\frac{d\overline{I}}{dt} = +\alpha/k \cdot \overline{S} \cdot \overline{I} - \beta \cdot \overline{I} \tag{6.5}$$

$$\overline{R}(t) = k \cdot N - \overline{S}(t) - \overline{I}(t) \tag{6.6}$$

$$\overline{S}(0) = k \cdot S(0) \qquad \overline{I}(0) = k \cdot I(0) \qquad \overline{R}(0) = k \cdot R(0) \tag{6.7}$$

As before, N here is the population density of the town we are examining. We assume that the parameter k is the same across all counties. This is a reasonable assumption, as since there is evidence from the Alpha and Delta waves of similarities in disease and population dynamics between the towns, and there was no significant difference between the towns with regard to public policy or availability of resources. To achieve this, the model given in

Equations (6.4)-(**??**) is fit to the Omicron data, with the parameter k being set. We iterate through values of k from 0.11 to 1.0 with step size of 0.01. From this, a value for k was chosen such the cumulative MSE of the model fits was minimized. Since we assume that the parameter k is the same across all counties, the best fit minimizes cumulative MSE. With these assumptions, the best fit for k was found to be $k = 0.16$. This implies that only 0.16 (16%) of the infected individuals were reported as being sick. The fits for this are shown in Table 6.8.

Parameters	**Yellowstone**	**Gallatin**	**Missoula**	**Flathead**	**Cascade**	**Lewis and Clark**
Start Week	93	93	93	93	93	93
End Week	106	106	106	106	106	106
α	0.001677	0.00117	0.001421	0.000918	0.000847	0.002217
(SE)	0.00016	0.000078	0.000102	0.00012	0.000051	0.000157
β	3.621773	2.452779	2.89265	1.58889	1.545334	3.48428
(SE)	0.341053	0.146284	0.203601	0.204642	0.095404	0.244819
Initial infected	1.7136	9.0832	2.651671	4.559771	3.387722	1.069243
(SE)	0.50248	1.452199	0.594795	1.252555	0.649445	0.25684
R_0	1.252534	1.360597	1.336078	1.295873	1.51892	1.28888
MSE	15.041259	9.6067	12.716526	12.001011	14.809027	6.046455
MSE μ	15.197888	16.608643	13.195711	11.617066	16.628763	7.03363
MSE σ	0.849219	25.75724	15.933198	0.19522	0.803234	13.644741
City	Billings	Bozeman	Missoula	Kalispell	Great Falls	Helena

Table 6.8: Omicron wave modified SIR model fitted parameter values with $k = 0.16$.

It can be seen that the fits for the parameter β are far more reasonable than those in Table 6.7. We have also included the statistics involved with the MSE values for the various choices of k. While the fits for Yellowstone, Flathead, and Cascade appear to be fairly insensitive to variations in k, Gallatin, Missoula, and Lewis and Clark have a high degree of variance in regard to changes in k. With these fits that at least seem more reasonable, we can compute the means, standard deviations, and ranges. These are shown in Table 6.9.

Parameter	μ	σ	$\mu \pm \sigma$	$\mu \pm 2\sigma$
α	0.0013750	0.0005161	(0.0008589, 0.0018911)	(0.0003427, 0.0024073)
β	2.5976177	0.9019888	(1.6956289, 3.4996065)	(0.7936401, 4.4015953)
R_0	1.3422914	0.0945989	(1.2476925, 1.4368903)	(1.1530936, 1.5314891)

Table 6.9: Omicron wave vaccination modified SIR model (k=0.16) fitted parameter values statistical analysis.

We can see much like with the Alpha and Delta waves, most fitted parameters are within one standard deviation of the mean, with all of them being within two. With this, we can see that the patterns of similarities between towns continues in the Omicron wave. In addition, this supports the assumption that the value of k is relatively similar between the towns, as if it was not, these differences would have been reflected in the fits for the other parameters. Examination of R_0 shows that the Omicron wave had a greater severity throughout. We will discuss the results of the SIR model fits for each wave and their relationships in the next section.

6.4 Analysis of Results of Parameter Fitting

We can draw several conclusions from the fitted SIR model parameters for the three waves. First, despite differences in demographics and locations, the 6 towns exhibited similar patterns in disease dynamics. Differences in initial infected can be explained by outside factors, as by its very definition, the initial infected are the initial number of people infected when the model is initialized. Factors that affect this are, generally, either travellers coming to the area and bringing the disease with them, or people travelling away and returning with the disease. In addition, we note that Montana has a high degree of tourism, and especially so for the larger towns in the state, which would be expected to have an effect on initial infected. This can be seen when examining new cases, as the number of new cases in Montana sharply increased after July 4, 2020. Thus, it is unreasonable to expect the initial infected number or density in the different cities to be equal. It can be observed from Table 6.1 that the initial number of infected is not an integer, while in real life the number of infected is an integer. Since we are fitting to imperfect data and on a non-continuous time frame, the fitted initial number of infected is necessarily a non-integer. However, we have shown, across the three waves, that the α and β parameters, which describe the dynamics of the disease, are similar within each wave across the six towns.

Parameters α and β represent an amalgamation of the dynamics of the disease and population interactions and behavior. Since we can reasonably assume that not only did everyone have the same variant but that the disease distribution between different variants was the same, we know that any effect of disease dynamics upon α and β should be the same regardless of city. This means that any significant difference between α and β can be attributed to differences in population dynamics. However, we can see that within each wave, the α and β parameters between different cities are similar. This implies that there is not a lot of difference between the cities when it comes to population dynamics. Now, we can see this for both α and β, which implies that not only are population dynamics similar prior to infection, but that the response of the populace to infection followed similar patterns.

6.5 Comparison of Waves

Let us now compare the differences between different waves. We can see from Table 6.10 that there is a pattern of severity throughout all six cities. We can see that from the Alpha wave to the Delta wave, the severity of both the disease dynamics and the response to the disease reduced. A simple reason for this would be vaccinations. As stated previously, by the time of the Delta wave, vaccinations had been underway for around 30 weeks, and continued throughout the Delta wave. It would appear that from a disease dynamics standpoint, the vaccinations had a marked effect. Moving from Delta to Omicron, it can be seen that both

α and β increase dramatically. These patterns of change are consistent throughout the six cities.

Parameters	**Alpha**	**Delta**	**Omicron**	**Alpha**	**Delta**	**Omicron**	**Alpha**	**Delta**	**Omicron**
	Yellowstone			Gallatin			Missoula		
Start Week	21	73	93	22	73	93	30	73	93
End Week	50	93	106	50	93	106	53	93	106
α	0.002098	0.001331	0.001677	0.002301	0.001192	0.00117	0.002271	0.001859	0.001421
(SE)	0.00061	0.000042	0.00016	0.000177	0.0000065	0.000078	0.000284	0.000052	0.000102
β	5.382404	3.316738	3.621773	6.222861	3.146865	2.452779	4.838584	4.741428	2.89265
(SE)	0.157993	0.10342	0.341053	0.482194	0.171934	0.146284	0.740162	0.132995	0.203601
Initial infected	0.367672	1.758827	1.7136	0.194525	1.439025	9.0832	1.026875	1.112842	2.651671
(SE)	0.029325	0.109974	0.50248	0.04816	0.147279	1.452199	0.558906	0.065486	0.594795
R_0	1.0547621	1.0857042	1.252534	1.0547409	1.0804817	1.360597	1.0283826	1.0663725	1.336078
	Flathead			Cascade			Lewis and Clark		
Start Week	26	73	93	27	73	93	31	73	93
End Week	50	93	106	50	93	106	53	93	106
α	0.001707	0.001171	0.000918	0.003314	0.000968	0.000847	0.003294	0.002268	0.002217
(SE)	0.000158	0.000035	0.00012	0.000225	0.00006	0.000051	0.000205	0.000066	0.000157
β	4.514582	2.458812	1.58889	8.770344	2.472195	1.545334	6.378984	4.263643	3.48428
(SE)	0.33681	0.07272	0.204642	0.596111	0.151178	0.095404	0.395579	0.124938	0.244819
Initial infected	2.352696	2.962869	4.559771	0.464807	3.44218	3.387722	1.067178	0.693541	1.069243
(SE)	0.144611	0.139756	1.252555	0.084453	0.365281	0.649445	0.119168	0.045052	0.25684
R_0	1.0528166	1.0682808	1.295873	1.0474591	1.0854101	1.51892	1.0460312	1.0775434	1.28888

Table 6.10: Comparison of fitted parameter values for all three waves of COVID-19 for all counties.

To further illustrate the consistency of dynamics from one change to the next, Table 6.11 shows the percentage change in the α and β parameters. It can be seen that within each town, the change in α and β from one wave to the next were quite similar, the notable exception being Missoula in the change from Delta to Omicron. In the change from the Alpha wave to the Delta wave, with the exception of Cascade, the other towns had fairly similar changes in parameters.

Parameters	**Alpha to Delta**	**Delta to Omicron**	**Alpha to Delta**	**Delta to Omicron**
	Yellowstone		Gallatin	
% α change	63.4413727	125.9954921	51.8035637	98.15436242
% β change	61.6218701	109.1968374	50.5694246	77.94357241
% R_0 change	102.933562	115.3660454	102.4404856	125.9250388
	Missoula		Flathead	
% α change	81.8582122	76.43894567	68.5998828	78.39453459
% β change	50.8167679	61.00799169	54.4637798	64.62023123
% R_0 change	103.6941407	125.2918656	101.4688408	121.3045297
	Cascade		Lewis and Clark	
% α change	29.2094146	87.5	68.852459	97.75132275
% β change	28.188119	62.50858043	66.8389041	81.72072568
% R_0 change	103.6231486	139.9397334	103.0125488	119.6128156

Table 6.11: Comparison of fitted parameter values for all three waves of COVID-19 for all counties.

The towns of Billings, Bozeman, Missoula, and Kalispell all exhibited similar dynamics across the three waves, while Great Falls and Helena's dynamics has stood out as not only being different from the other four, but also generally different from each other as well. As mentioned earlier, the towns of Bozeman and Missoula are home to the two largest universities in Montana (Montana State University and University of Montana, respectively). The town of Kalispell is the largest town near Glacier National Park, and Billings is the largest town in Montana, with the highest rate of population growth of any town within the state. Unlike these four, the town of Great Falls, has multiple colleges, but none of them have the levels of national recognition similar to those of Montana State University or University of Montana. Because of this, external influx of students, athletes, and other visitors is lessened considerably in comparison. In addition, the town of Helena is the capital of Montana. That being said, the towns of Billings, Bozeman, Missoula, and Kalispell have a significant component of out-of-state residents. For Bozeman and Missoula, this is related to the universities attracting a certain number of students, while Kalispell has visitors heading to Glacier National Park, and Billings has the natural influx of people as the largest growing town in Montana. In addition, as mentioned earlier, there is a significant tourist component that cannot be ignored. While Montana is only the 40th in terms of total number of tourists [115], it is 44th in terms of population according to the 2022 US census. Along with tourists, Bozeman Yellowstone International Airport (BZN) and Missoula International Airport (MSO) should be mentioned as they are located in Bozeman and Missoula respectively, and are often used by tourists to get to the state. In particular, BZN is the closest major airport to the northern entrance to Yellowstone National Park. On the other hand, neither Great Falls nor Helena have a similar common mechanism for influx of out-of-state people. Another thing to note is that the counties that contain Billings, Bozeman, Missoula, and Kalispell, namely Yellowstone, Gallatin, Missoula, and Flathead, respectively, all have a total population of over 100 thousand. While Kalispell has a much smaller population compared to Great Falls (26100 vs 60403), the total population of Flathead (Kalispell's county) is greater than that of Cascade (Great Fall's county). If we were to group the counties by similar dynamics of the cities, it would appear that we can use county population as a method of this suggests that there might be a relation between county population and cross-correlation.

In conclusion, we can see that at least for the four most populous towns, the fitted model parameter values are quite close, meaning that in the future, the population dynamics of the towns can be assumed to be equitable, implying homogeneity in the efficacy of preventative measures.

Chapter 7

Age Demographic Analysis of COVID-19

While the number of new infected is valuable information to keep track of, and the corresponding data type are most often modelled, it is not the data that people are truly interested in. While getting sick is an unpleasant experience, we are ultimately more concerned about the deaths as a result of the disease. The high mortality rate of COVID-19 was a main cause for the reaction that followed. The worldwide response to the H1N1 virus, commonly known as the swine flu, was quite a bit more tame in comparison [116].

The H1N1 virus, known as the swine flu, was the cause of the previous pandemic that gripped the world, and yet unlike COVID-19, it is perceived as minor in comparison. The H1N1 virus was declared a pandemic by the World Health Organization in June 2009 and lasted officially until August 2010. During its lifetime it is estimated that there were around 60.8 million cases of H1N1 with somewhere between 151,700- 575,400 deaths [117]. In comparison, COVID-19 has had 753,001,888 confirmed cases, with 6.807,572 deaths [118]. From this, we can see that COVID-19 was far more virulent and widespread than H1N1. Besides this, there were several other differences. The largest difference was related to the general demographics affected. H1N1 mainly affected the younger demographic [116], while COVID-19 primarily affected the elderly demographic. To examine the dynamics of deaths in COVID-19, we need to look at the case-fatality rates.

The case-fatality rate of a disease is the chance of death if a person is infected with the disease. This can be easily computed by simply dividing the total number of deaths by the total number of cases. This is illustrated in Equation 7.1. Now, if we make the assumption that the cases are spread across the population based on demographics, we would like to compare the demographics for different counties illustrated in Figure 7.1.

$$\text{Case Fatality Rate} = \frac{\text{Total Deaths}}{\text{Total Cases}} \tag{7.1}$$

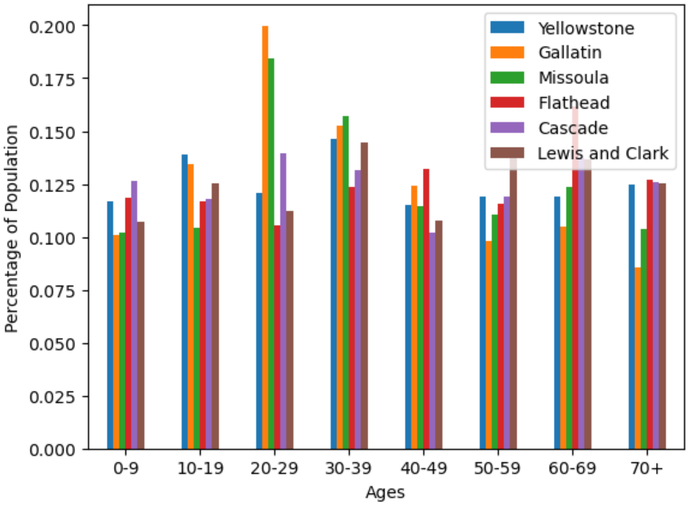

Figure 7.1: County age demographics.

The bar graph in Figure 7.1 shows the age demographics of the 6 counties in Montana based on the latest census data. However, a simple look at the cases shows a different picture, contrary to a heuristic assumption on proportionality of cases and population in various age groups; see bar graph in Figure 7.2.

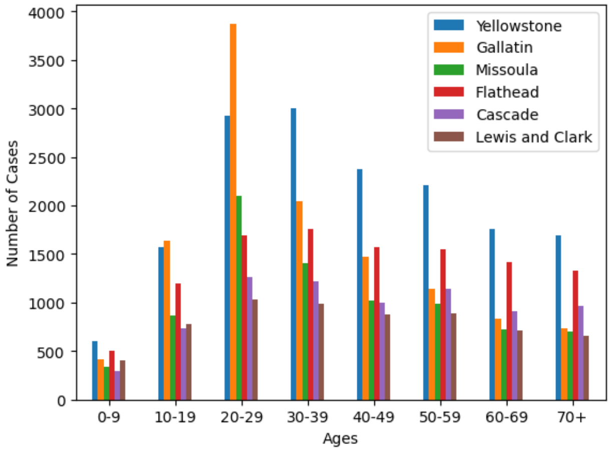

Figure 7.2: Alpha wave total cases for different age demographics.

If we adjust the numbers as to have percentages of the population that were infected instead of the original counts, we get the situation presented in Figure 7.3.

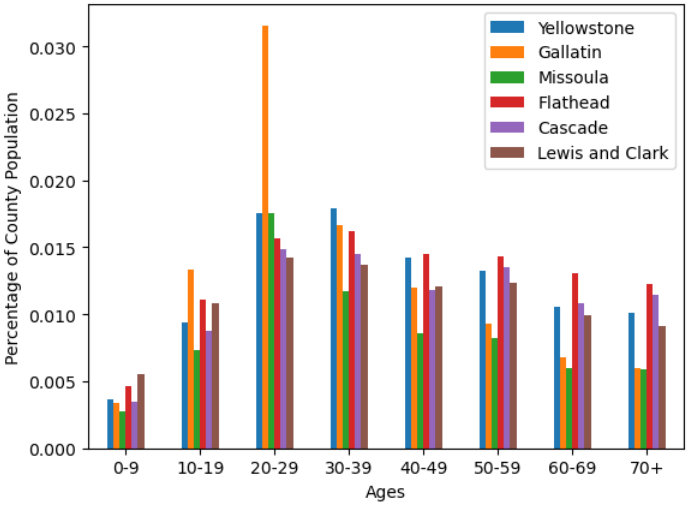

Figure 7.3: Alpha wave cases age demographics expressed as percentages of each age population group for various counties.

We can also look at this in a different manner, as the percentages of total cases in each age demographic, see Figure 7.4.

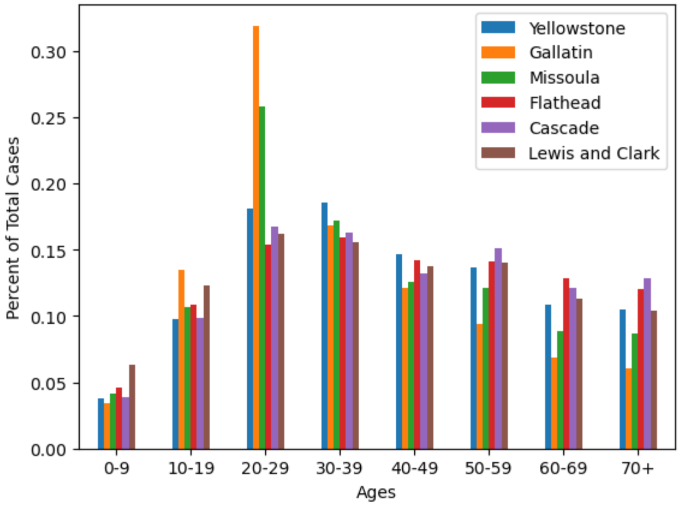

Figure 7.4: Alpha wave cases age demographics expressed as percentages of total cases.

From this we can see that for the Alpha wave there is a general pattern being followed by the towns, however, this is most certainly not reflected in the demographics, as can be seen from the comparison presented in Figure 7.5.

In spite of the correlation not being perfect, there are some patterns that can be observed. For the 20-60 year old age ranges, we can see that except for the 40-60 age range in Gallatin and the 50-60 age range in Lewis and Clark, that the percentage of COVID-19 cases in that age range was greater than the percentage of people belonging to that age range. In other words, the number of COVID-19 cases for those age ranges was overrepresented in those age groups. However, it can also be observed that for the most part in the 0-20 and 60+ age ranges, the percentage of COVID-19 cases was generally less than that of the demographic percentage. In addition, it is important to note that in cases where this was not the case, the differences in percentages between demographics and cases were very small. With this, we can see some consistent patterns across the six towns.

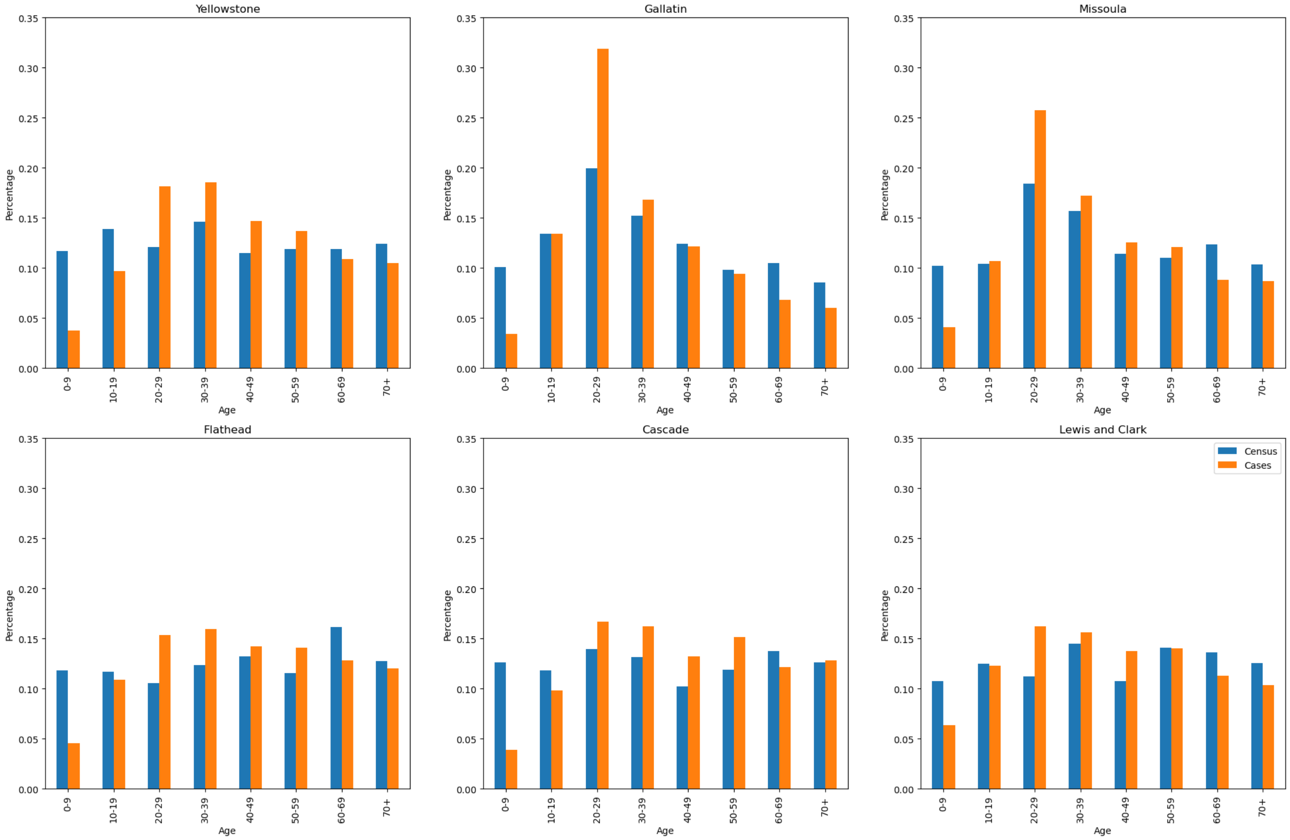

Figure 7.5: Comparison of percentages of each age group vs percentages of total cases in the Alpha wave, for different counties.

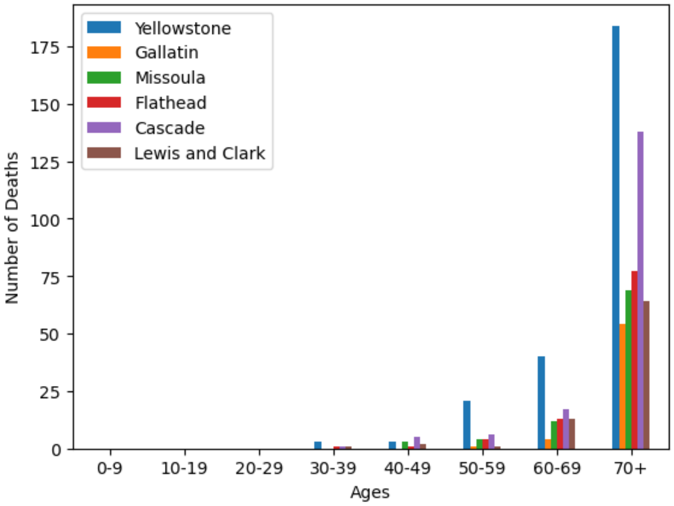

Figure 7.6: Alpha wave total deaths by age demographic for different counties.

Let us next consider the number of deaths during the Alpha wave (see Figure 7.6). A simple visual comparison with Figure 7.2 shows an extreme disparity between the age distribution of cases and distribution of deaths. Dividing the number of deaths by the number of cases to get the case-fatality rates for the Alpha wave, shown in Figure 7.7.

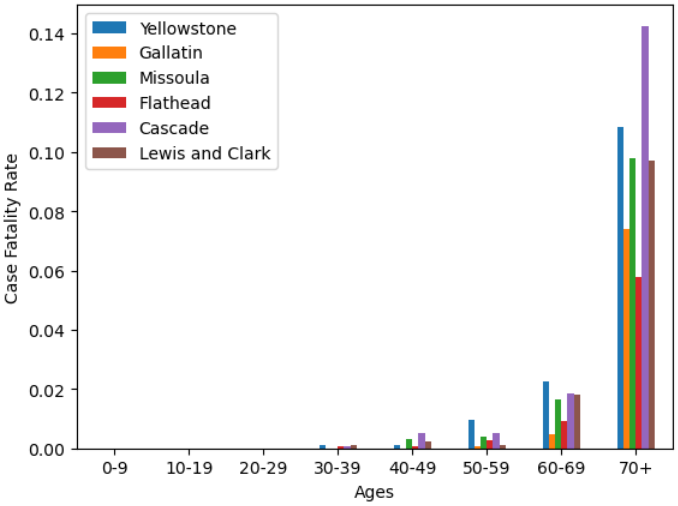

Figure 7.7: Alpha wave case fatality rates by age demographic.

It is evident from Figure 7.7 that the case-fatality rates drastically increase in the 70+ age range. Data scaling has no affect upon case fatality rates, as scaling would be done to both the total cases and total deaths, resulting in cancellation of scaling terms. This is illustrated with the equation Case Fatality Rate = (Total Deaths · S)/(Total Cases · S) in which S represents the scaling factor used on the data. Let us now examine the other two waves.

For the Delta wave, the cases data for different age groups are shown in Figure 7.8.

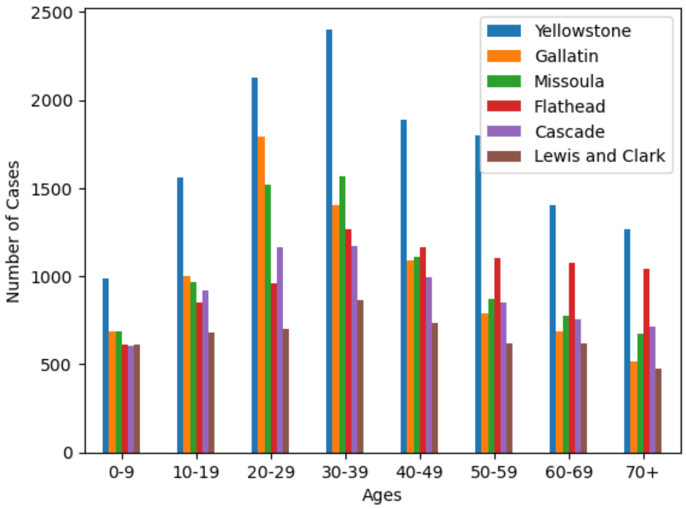

Figure 7.8: Delta wave total cases by age demographic

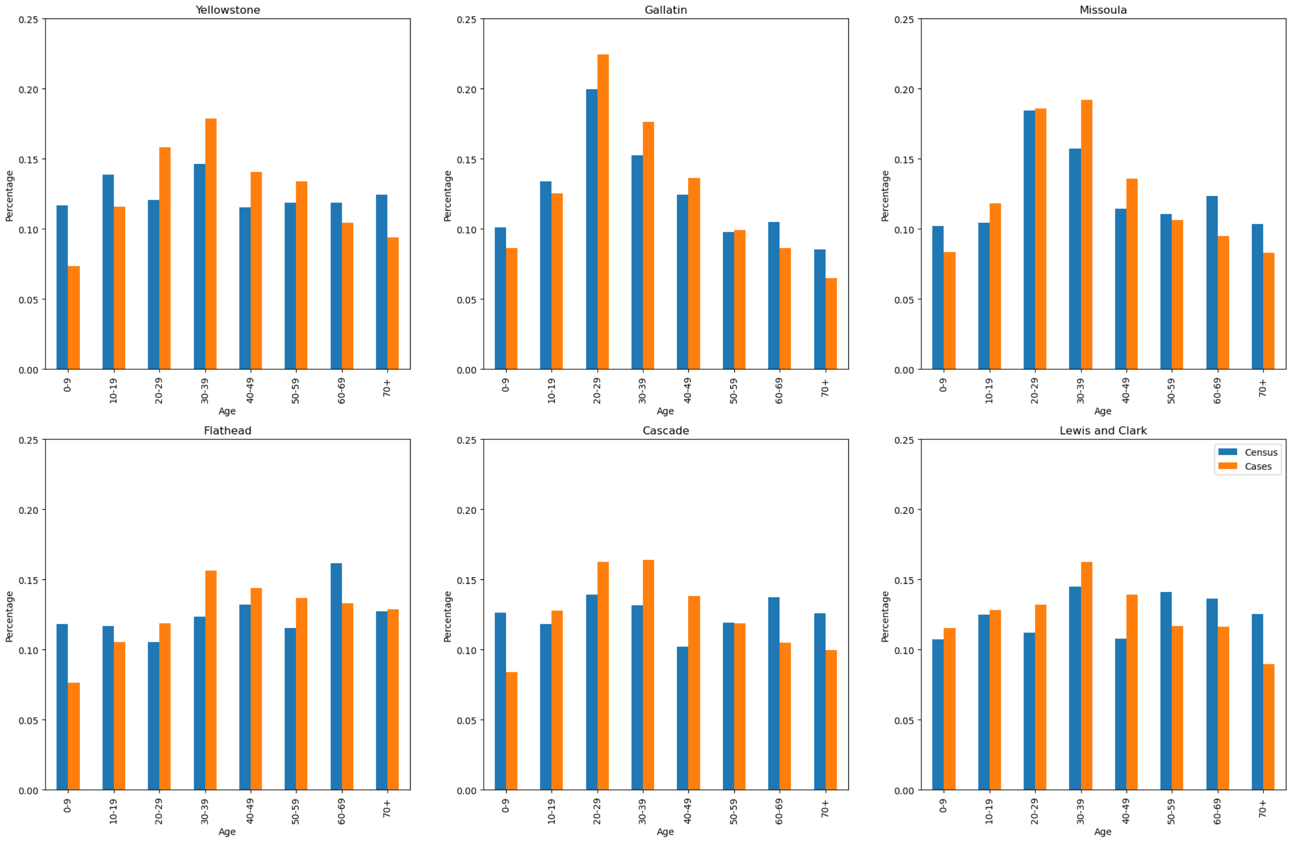

Figure 7.9: Comparison of age demographics of counties vs Delta wave cases.

From the bar graphs in Figure 7.8 we can see that there were fewer cases during the Delta wave in comparison to the Alpha wave. We note that the county of Yellowstone as a whole has an overall higher number of cases than others. In Figure 7.9, we show another comparison of the demographics of each age with the percentages of cases in each age group for different counties.

Comparison of county age demographics with the age-based demographics of the Delta wave, as illustrated in Figure 7.9, show a similar pattern to that displayed in the Alpha wave as illustrated in Figure 7.5. While the county of Lewis and Clark displays further differences, the lower county population and existence of the state capital Helena could be contributing factors. The age-based case-fatality rates for the Delta wave are shown in Figure 7.10.

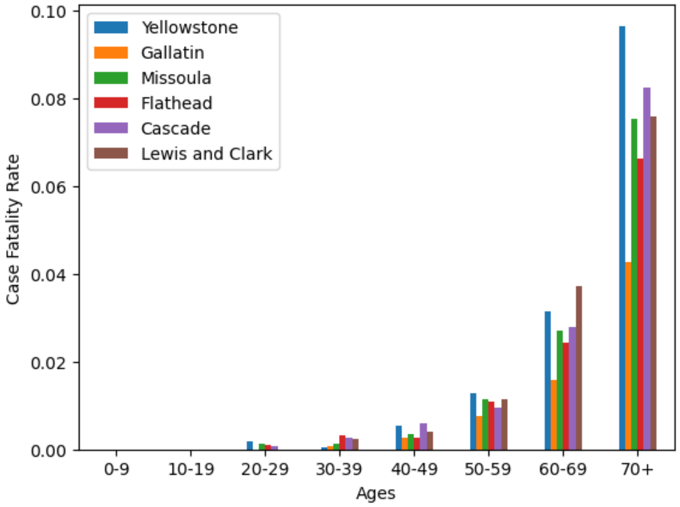

Figure 7.10: Delta wave case fatality rates by age demographic

In the bar graphs shown in Figure 7.10 we can notice several features that are different compared to the Alpha wave. While the overall pattern is similar, we can see that in the 70+ age range, the general fatality is much lower. While in the 60-69 age range we can see that the case-fatality rate went up slightly, is it a change from about 0.02 to 0.03. For the Omicron wave, the total cases by age group are shown in Figure 7.11.

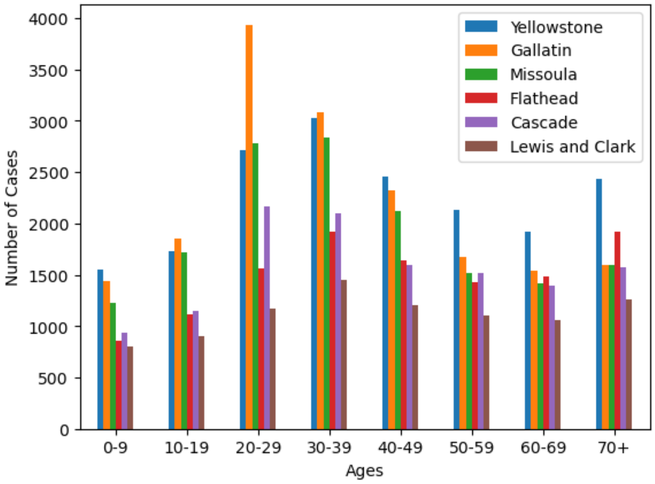

Figure 7.11: Omicron wave total cases by age demographic

As shown in earlier parameter fitting, there is a dramatic increase in the number of cases compared to the Delta wave, with the numbers being similar to the Alpha wave, particularly if we compare to Figure 7.2. We can also see the large uptick in cases from the 20-29 age group in Gallatin, which corresponds with the anomalous pattern shown in Figure 6.12 and Figure 6.13. Comparison of age demographics to the percentages of cases by ages for Omicron is shown in Figure 7.12.

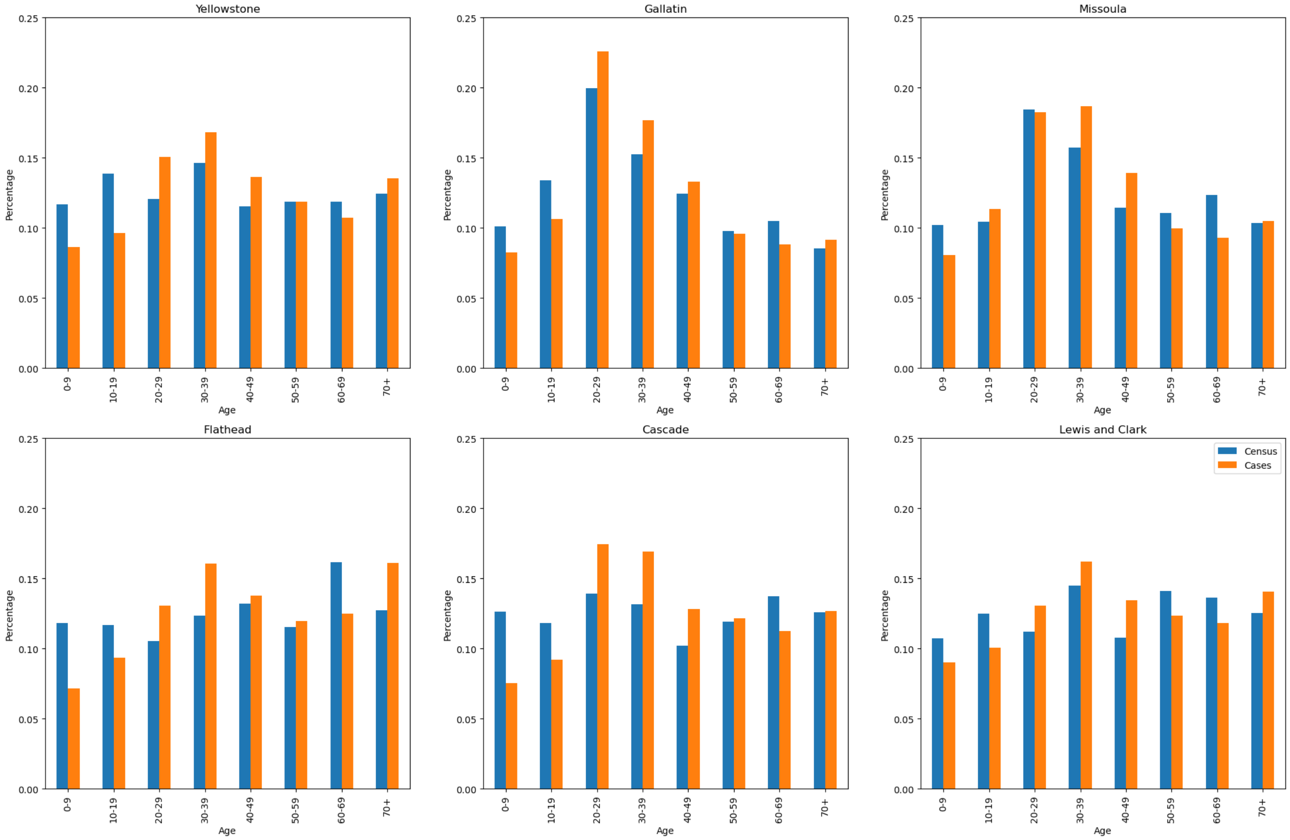

Figure 7.12: Comparison of age demographics of counties with total Omicron wave case age demographics.

There are patterns that can still be seen here. For the 20-60 and 70+ year old age ranges, we can see that with the exceptions of the 50-59 age ranges in Yellowstone, Gallatin, Missoula, and Lewis and Clark, and the 20-29 age range in Missoula, that the percentage of COVID-19 cases in that age range was greater than the percentage of people belonging to that age range. In other words, the number of COVID-19 cases for those age ranges was overrepresented in those age groups. We can see for the 0-19 and 60-69 age ranges, the percentage of COVID-19 cases was generally less than that of the demographic percentage. This mirrors the patterns in both the Alpha and Delta waves, although it should be noted that the pattern in the 70+ age range was the opposite of that of the Alpha wave. The case-fatality rates for the Omicron wave is shown in Figure 7.13.

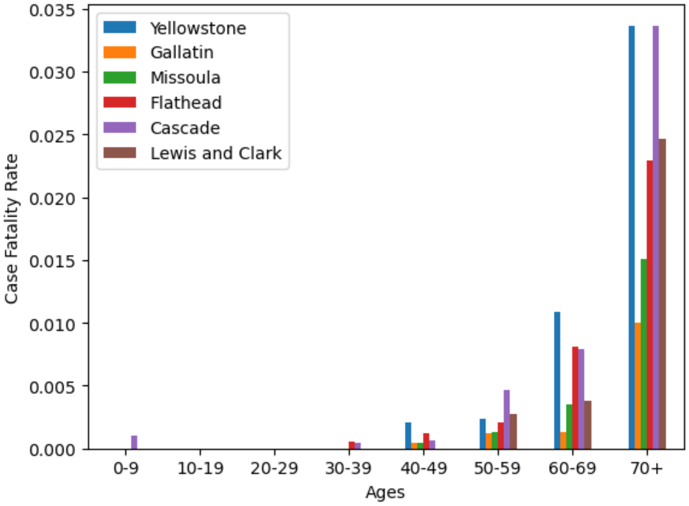

Figure 7.13: Omicron wave case fatality rates by age demographic.

From this we can see that the case-fatality rates of the Omicron wave follow a similar pattern to the case-fatality rates of the Alpha and Delta wave as illustrated in Figure 7.7 and Figure 7.10. However, if we compare the actual case fatality rates shown in Figure 7.13 with those shown in Figure 7.7 and Figure 7.10 we can see that the Omicron case-fatality rates were lower than those of the Alpha and Delta counterparts.

Chapter 8

Discussion

COVID-19, in spite of the millions of deaths that resulted, has been valuable for showing several details regarding infectious disease modeling. Prior to COVID-19, one of the best case studies for infectious disease modeling was an outbreak of influenza at an all-boys boarding school in England. The reasons for this were several fold. One, the residents of the boarding school, by the very nature of the institution, interacted with each other regularly, thus allowing for the spread of the disease naturally. Two, consistent records were kept, with the main data recorded being the number of boys who were sick. Three, the population in question (boys at the boarding school) was a closed system. Combining these factors together made the boarding school example the best case study for infectious disease modeling.

COVID-19 offers us similar opportunities now as a result of the record keeping that has been done throughout the pandemic. While there are potential concerns in regard to accurate record-keeping, there is far more data for COVID-19 than any other pandemic. The previous pandemic, the H1N1 virus or swine flu, was not as well recorded as COVID-19.

As discussed in Chapter 4, there are three major ways in which this dissertation separates itself from the main body of work being done in this space. The first is an adherence to the concept of simple data-driven modeling. While many assumptions must be made in order to accomplish this, we have shown that these assumptions are reasonable within our context of study. Our use of data-driven modeling shows that an accurate model can be derived directly from the data of interest, instead of needing to rely on fitted parameter values from other locations.

The second way this dissertation separates itself is by fitting to the removed compartment of the SIR model. As discussed before, there is a lack of clarity in both literature and data reporting as to the nature of the data. We discussed how the data typically given is that of newly infected, not total infected, and that there is no consistent method for determining recovery time. In addition, we noted that the data given is that of newly infected who got tested, and the test results were recorded. We pointed out that most people after testing positive for COVID-19 would quarantine themselves, leading us to the conclusion that it would be appropriate to consider these as removed instead of infected. This conclusion is at odds with most of the current COVID-19 literature. However, the results achieved show the merit of such an approach.

The third manner in which this dissertation separates itself is by use of the population density. In Section 2.2 we discussed the Law of Mass Action and the subsequent derivation of the SIR model. From this in Chapter 4 we discussed the merits of using the SIR model within the framework of population density instead of straight population, as the Law of Mass Action specifies concentrations of reactants. In addition, it acknowledges the fact that differences in population densities will lead to differences in infection rates. Furthermore, we scaled the data not only to take into account population density, but city population density, as the majority of the cases will take place within the confines of a city. After scaling the data in order to represent city population density, it can be seen from Figures **??**, 6.9, and 6.13 that the towns have similar patterns.

With these three changes, this dissertation has shown a different manner to utilize the SIR model with COVID-19. First, we stick strictly to a simple data-driven SIR model, limiting the assumptions we need to make and the issues of overfitting that arise from an overabundance of parameters. Second, the data broadly available is more appropriate to fit to the removed compartment. Third, population density is more appropriate to utilize as it reflects the physical differences in different locations. The quality of fits that have been achieved with these methods as well as the standard errors with our parameter estimates show the validity of these choices.

We have also shown that the parameter fits for the 6 towns within each wave are similar to each other. With this, we can potentially use the fits from one town to approximate the disease dynamics of another town. In addition, we can see from the fitted parameter β that the general practice of general quarantining had an effect upon the model, and that it is a primary factor in the rate of removal from the susceptible population.

Regarding similar dynamics, we showed in Chapter 6 the similarities in age demographics for total cases and deaths across the six counties. From this we conclude that rural counties and towns share many similarities not only from a disease dynamics perspective, but also from an case age structure standpoint. Because of these similarities, we suggest that knowledge of the disease dynamics and age structure of one county/town should allow for reasonable predictions of the disease dynamics and age structure of other towns. The subsequent question that naturally arises from this is how much can these similarities be used for such predictions.

8.1 Future Research

We have shown in this dissertation a method for a data-driven simple SIR model that accurately models COVID-19. In addition, we have shown that the six towns we examined exhibited similar disease and population dynamics in response to COVID-19. A question we have going forward is how far does this similarity extend. We acknowledge the fact that the six towns examined were all in Montana, meaning that from a state legislature, geographic, and cultural standpoint, they are vastly different from a similar town in a state such as New York. To understand both the extent of this similarity and further details that might affect it, we suggest examinations of more towns, first from Wyoming and the Dakotas, and then further expanding outwards. We would also like to note the town of Butte in Montana, as it was not included in this dissertation. Butte is a town of 35,411 but was not included as we examined the most populous towns in the six most populous counties, instead of the most populous towns. We also suggest examining the town of Butte in future studies. Fitting of data from other towns, particularly from other states, will be useful in understanding additional variables that may have an effect on disease dynamics.

In addition to understanding additional variables that affect disease dynamics, the similarities in disease and population dynamics suggest that we can predict the parameters and dynamics of the disease in another town. Another future research topic would be to develop a method of predicting the model for one rural town based on the fitted model of a different town. Like above, additional towns will need to be examined to understand more of the variables that affect the disease dynamics.

Chapter 9

Conclusion

In this dissertation we have shown that fitting the data to the Removed compartment of a simple data-driven SIR model, while examining the data in terms of city population density, produces excellent results. Using these results, we have shown similarities in disease and population dynamics among the largest cities in the six most populous counties in Montana. The process for doing this has been outlined primarily in Chapter 5. The initial parameter fits, as shown in Table 6.1 and illustrated in Figure 6.1 suggested parallels in disease and population dynamics between the six counties. It was then found that scaling the data by the ratio of the city population and county population, as well as by the area of the largest city, resulted in very similar disease and population dynamics. This scaling was done to simulate COVID-19 within the cities instead of the county as a whole. The results of this are shown in Figure 6.3. Fitting this scaled data to the SIR model showed with greater certainty the parallels in disease dynamics and population behavior between the six cities for the Alpha wave.

This methodology was continued with the Delta and Omicron waves. While these waves had their own separate considerations, discussed in Sections 6.2 and 6.3, parameter fitting similar to that of the Alpha wave was done. As with the Alpha wave, scaling the data by the ratio of the city population and county population as well as by the area of the largest city in order to simulate the dynamics of the cities themselves and then fitting the scaled data to the SIR model showed similar parallels across cities as the Alpha wave. In addition, with a couple of exceptions, the general change in fitted parameter values from wave to the next was a positive growth.

From this, we can conclude that the underlying disease dynamics and population behavior is similar among the six cities. A result of this conclusion, then, is that external forces upon disease dynamics or population behavior should have similar effects upon the six cities. In addition, if a city experiences an outbreak ahead of the other cities, it can be used as a template of what to expect with the disease outbreak.

This similarity in disease and population dynamics is further illustrated by analysis of age demographic data. Comparison of age demographics of total cases for each wave with the corresponding age demographics of each county showed a pattern in the relationship between the two that is mostly consistent among all six counties and throughout all three waves. In addition, the overall shape of the case fatality rate bar charts as shown in Figure 7.7, Figure 7.10, and Figure 7.13 is consistent, in spite of average case fatality rates decreasing in each subsequent wave. From this, it can be seen that the disease and population dynamics are equitable between counties/cities, even down to age groups.

We have shown that with the proper use of data and population density instead of straight population, COVID-19 can be accurately modeled with a simple SIR model. As every place in the world that has COVID-19 is different, it is insensible that parameters computed from one place are blindly used in others. We hope that with this, we can improve COVID-19 modeling. In addition, there it appears that rural towns, at least those in Montana, exhibit parallels in disease and population dynamics in regard to COVID-19. This suggests that external factors, such as governmental policies and vaccinations, should have equitable effects on separate rural towns. We believe that with further study, the fitted model of one town could be used to accurately predict the model of a separate town, and thus enable more accurate modeling and informed policy decisions in the future.

Appendix A

SIR Model Code

A.1 Libraries and Data Processing

The code for this dissertation was written entirely in Python. This first section contains the code that loads the libraries and data being used.

```
import numpy as np
import scipy as sp
from scipy import integrate, optimize
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import mean_squared_error as MSE
from sklearn.metrics import r2_score
import datetime

import geopandas as gpd


### RUN THIS AFTER KERNEL RELOAD ###

# read in data and create lists that are used later

df = pd.read_csv('county_data_combined.csv', low_memory = True)
v_df = pd.read_csv('COVID-19_Vaccinations_in_the_United_States_County.csv',
                   low_memory = False)
v_df_MT = v_df[v_df['Recip_State'] == 'MT']
county_map = gpd.read_file("MontanaCounties_shp/County.shp")
county_list = df['county'].unique()
county_pop = [119533, 122713, 167146, 11283, 10821, 84511, 13785,
             16179, 1946, 5994, 5011, 12957, 10847, 11617, 831, 4896,
             9044, 2059, 9524, 7288, 9491, 12470, 72223, 8917, 1964,
            17473, 35411, 108454, 32033, 20525, 45959, 8124, 11916,
            8904, 7537, 768, 3344, 3017, 1209, 3527, 6269, 1686,
            1718, 1702, 934, 6980, 5916, 2044, 3723, 12959, 6999,
            4192, 1091, 4860, 1428, 519]

# urban_pop is ordered to be added to combined_map, not with the
# order of county_pop

urban_pop = [2294, 0, 0, 4261, 3383 + 1657, 2039 + 1023 + 1000,
             7784, 28532 + 10460 + 5901 + 2053 + 1822 + 1382, 1682,
             34993, 0, 9421, 0, 109550 + 7195 + 6766, 2918 + 1851 +
             1267 + 1141, 0, 1038, 4809 + 2025 + 1125, 1863, 1910,
             2098, 2266 + 1897 + 1823, 0, 8393, 0, 0, 0, 0, 0, 0,
             2916, 74428 + 5377 + 4399 + 2465 + 1958 + 1690 + 1682 +
             1018, 59701 + 1615, 5818, 32315 + 9168 + 8670 + 4706 +
             3813 + 2089 + 1331, 0, 5018 + 2138 + 2088, 0, 1713,
             1443, 1394 + 1107, 2500, 2743, 9715, 1567 + 1273, 0,
             1915, 3089, 3328, 1027, 3022 + 2653 + 1970 + 1018,
             23938 + 8149 + 7870 + 5575 + 5118 + 2705 + 1119 + 1049,
             1786, 2737 + 1118, 4960 + 1998, 6246]

# urban_area has the information for the six most populous towns in
# Montana

urban_area = {'City': ['Billings', 'Missoula', 'Great␣Falls',
                       'Bozeman', 'Butte','Helena', 'Kalispell'],
              'County': ['Yellowstone', 'Missoula', 'Cascade',
                         'Gallatin', 'Silver␣Bow', 'Lewis␣and␣Clark',
                         'Flathead'],
              'Area': [43.41, 27.51, 21.79, 19.12, 716.25, 16.35,
                       11.64],
              'Populations': [117445, 74822, 60403, 54539, 35411,
                              33120, 26110]}

ua_df = pd.DataFrame(data = urban_area)

usaf_death_data = pd.read_csv('data_files/covid_deaths_usafacts.csv',
                              low_memory = True)
usaf_cases_data = pd.read_csv('data_files/covid_confirmed_usafacts.csv',
                              low_memory = True)

### RUN THIS AFTER KERNEL RELOAD ###

# This block creates a dataframe with the weekly new cases for the
whole dataset

weekly_h = []
weekly_removed_h = []

for i in range(len(county_list)):
    c_data = DataLoad(i)
    daily, weekly = InfectedData(c_data)
    weekly_h.append(weekly)
    removed = np.zeros(len(weekly))
    removed[0] = weekly[0]
    for j in range(len(weekly)-1):
        removed[j+1] = removed[j] + weekly[j+1]
    weekly_removed_h.append(removed)
k = 111
column_names = []

for i in range(111):
    column_names.append(str(i))

# This is a dataframe with weekly rates for all counties
weekly_h_df = pd.DataFrame(weekly_h, columns = column_names)

column_names = []
for i in range(111):
    column_names.append('r'+ str(i))


#This is a dataframe with weekly removed rates for all counties
weekly_removed_h_df = pd.DataFrame(weekly_removed_h, columns =
column_names)

# This next bit adds in the COUNTYNUMB column to make these numbers
# talk to the geopandas data.

CN = np.zeros(56)

for i in range(len(county_list)):
    c_name = county_list[i]
    CN[np.where(county_list == c_name)[0][0]] =
    county_map[county_map['NAME'] == c_name.upper()]
    ['COUNTYNUMB'].values[0]


weekly_h_df.insert(0, "COUNTYPOP", county_pop, True)
weekly_h_df.insert(0, "COUNTYNUMB", CN, True)

weekly_removed_h_df.insert(0, "COUNTYNUMB", CN, True)

intermediate_map = county_map.merge(weekly_h_df, on = 'COUNTYNUMB')
combined_map = intermediate_map.merge(weekly_removed_h_df, on =
'COUNTYNUMB')
combined_map['POP_DENSITY'] = combined_map['COUNTYPOP'] /
combined_map['SQMILES']

combined_map.insert(0, "URBANPOP", urban_pop, True)
```

A.2 Functions

The following are functions that are called throughout the rest of the code. Some of the functions (DataLoad and InfectedData) are called in the previous section, but grouping was done to keep data processing all in one section.

```
### RUN THIS AFTER KERNEL RELOAD ###

# Run this to load data from a specific county.
def DataLoad(county, weekly = True):
    county_data = df[df['county'] == county_list[county]]
    # the weekly is in there if a weekly portion is needed for this.
    return county_data


# This takes the data from DataLoad and turns it into daily and weekly
# datasets
def InfectedData(county_data):
    k = len(county_data)
    daily_infected = county_data['Daily total cases'].values
    weekly_infected = np.add.reduceat(daily_infected, np.arange(0, k, 7))
    return daily_infected, weekly_infected


# Basic SIR model to be used in the odesolveR1 function below
def deriv(y, t, alpha, beta):
    S, I,R = y
    dSdt = -alpha * S * I
    dIdt = alpha * S * I - beta * I
    dRdt = beta * I
    return dSdt, dIdt, dRdt


# SIR model modified to account for vaccination rates, used in the
# odesolveR3 function below
def deriv2(y, t, alpha, beta):
    S, I,R = y
    dSdt = -alpha * S * I - gamma
    dIdt = alpha * S * I - beta * I
    dRdt = beta * I
    return dSdt, dIdt, dRdt


def odesolveR1(x, alpha, beta, I0):
    S1, I1, R1 = integrate.odeint(deriv,
                                  (S0, I0, R0),
                                  x,
                                  args=(alpha, beta)).T
    return R1
```

```
def odesolveR3(x, alpha, beta, I0):
    S1, I1, R1 = integrate.odeint(deriv2,
                                  (S0, I0, R0),
                                  x,
                                  args=(alpha, beta)).T
    return R1
```

A.3 Infection Visualization

The following codes are used to create some of the graphs shown in the dissertation.

Figure 5.1 code.

```
county_name = 'Yellowstone'
start = 40
end = 44

# Find index for the county
county_index = np.where(county_list == county_name)[0][0]

# pull the daily infected rates and transform it into a weekly
# average
county_data = DataLoad(county_index)
daily, weekly_0 = InfectedData(county_data)

daily1 = daily[start*7-1:end*7-1]
d_time = np.linspace(0, (end-start)*7, (end-start)*7)

plt.plot(d_time, daily1, label = county_list[top_counties[i]])
plt.xticks([0,7,14,21,28])
plt.xlabel('Days')
plt.ylabel('Daily Newly Infected')
plt.show()
```

Figure 5.2 code.

```
county_name = 'Yellowstone'
start = 1
end = 112

# Find index for the county
county_index = np.where(county_list == county_name)[0][0]

# pull the daily infected rates and transform it into a weekly
# average
county_data = DataLoad(county_index)
daily, weekly_0 = InfectedData(county_data)

weekly = weekly_0[start-1:end-1]
t_time = np.linspace(start,end,end-start)
```

```
plt.plot(t_time, weekly, label = county_list[top_counties[i]])
plt.xticks([0,20,40,60,80,100,120])
plt.xlabel('Weeks')
plt.ylabel('Weekly Newly Infected ')
plt.show()
```

Figure 6.1, ??, ??, ??, ??, 6.8, 6.9, 6.12, 6.13 code.

The following code specifically creates Figure **??**, however it can be easily reconfigured to produce the other figures.

```
# number of counties to look at
n = 6

# These are start and end weeks to examine.
start = 20
end = 60

top_counties = np.argsort(county_pop)[:: -1][:n]
for i in range(n):
    c_data = DataLoad(top_counties[i])
    daily, weekly_n = InfectedData(c_data)
    county_name = county_list[top_counties[i]]
    city_area = ua_df[ua_df['County'] == county_name]['Area'].iloc[0]
    city_pop = ua_df[ua_df['County'] == county_name]
                    ['Populations'].iloc[0]

    county_area = combined_map[combined_map['NAME'] ==
                    county_name.upper()]['SQMILES'].iloc[0]

    rural_pop = combined_map[combined_map['NAME'] ==
                    county_name.upper()]['COUNTYPOP'].iloc[0]

    pop_ratio = city_pop/rural_pop
    weekly = weekly_n[start-1:end-1] * pop_ratio / city_area

    t_time = np.linspace(start,end,end-start)

    plt.plot(t_time, weekly, label = county_list[top_counties[i]])
plt.legend()
plt.show()
```

A.4 SIR Model Fitting

The following codes are for the different waves. While a single function could be made for this, in the operating code different instances of this were utilized for different waves. The goal of these codes is to produce the dataframe results_df that contains the fitted parameters and other relavent information, some of which is not referenced in the dissertation but is useful when modelling.

Alpha Wave Fitting

The Alpha wave fitting code here is designed to find a starting week by targeting a starting infection density.

```
# number of counties to look at
n = 6

start_density = 20
end_weeks = [50, 70]

# Fitting to Susceptible Data instead of Removed
susceptible = False

# Initial Parameters [alpha, beta]
ip = [3.714e-05,6.5]

### Setting codes:
#
#   0: No scaling whatsoever
#   1: Scaling by COUNTY area (county density)
#   2: Scaling by CITY area (city density via area of largest city)
#   3: Scaling by CITY/COUNTY ratio (largest city pop / county pop)
#   4: Scaling by URBAN/COUNTY ratio (urban population / county pop)
#   5: Scaling by CITY/COUNTY ratio and city area(largest city pop
#                             / county pop / city area)
#   6: Scaling by URBAN/COUNTY ratio and city area(urban population
#                             / county pop / city area)
#
###

setting = 5


results_df = pd.DataFrame(['Start Week',
                           'End Week',
                           'alpha',
                           'beta',
                           'Initial Infected',
                           'City',
                           'City Population',
                           'Urban Population',
                           'County Population',
                           'City Area',
                           'County Area',
                           'City/Rural Population Ratio',
                           'Urban/Rural Population Ratio',
                           'City Pop Density',
                           'Rural Pop Density',
                           'City/Rural Ratio divided by city area',
                           'MSE'], columns = ['Parameters'])


top_counties = np.argsort(county_pop)[::-1][:n]

end_array = np.linspace(end_weeks[0],
                        end_weeks[1],
                        end_weeks[1]-end_weeks[0]+1)
scores = np.zeros(len(end_array))

for h in range(n):
    # Find name of county
    county_name = county_list[top_counties[h]]
    county_info = combined_map.loc[combined_map['NAME'] ==
                                   county_name.upper()]

    # Specific city population and area, not the overall urban one.
    city_pop = ua_df[ua_df['County']
                     == county_name]['Populations'].iloc[0]
    city_area = ua_df[ua_df['County']
                     == county_name]['Area'].iloc[0]

    # County (rural) area
    rural_area = combined_map[combined_map['NAME']
                      == county_name.upper()]['SQMILES'].iloc[0]

    # Find index for the county
    county_index = np.where(county_list == county_name)[0][0]

    # pull the daily infected rates and transform it into a weekly
    # average
    county_data = DataLoad(county_index)
    daily, weekly_0 = InfectedData(county_data)

    weekly_1 = weekly_0[10:70]

    ### This creates the removed dataset
    removed = np.zeros(len(weekly_1))
    removed[0] = weekly_1[0]
    for k in range(len(weekly_1)-1):
        removed[k+1] = removed[k]+weekly_1[k+1]
    ###

    # county TOTAL population
    rural_pop = county_pop[county_index]
    # county URBAN population
    urban_p = combined_map[combined_map['NAME']
                      == county_name.upper()]['URBANPOP'].iloc[0]
```

```
ur_pop_ratio = urban_p/rural_pop            # urban/rural pop ratio
cr_pop_ratio = city_pop/rural_pop             # city/rural pop ratio

if setting == 0:
    # this is for absolutely no scaling whatsoever
    removed_scaled = removed * 1
    start, init_density = min(enumerate(removed_scaled),
                       key=lambda x: abs(x[1]-start_density))
    initial_i = weekly_0[start]
    population = rural_pop
elif setting == 1:
    # this is for scaling only by county area (county density)
    removed_scaled = removed / rural_area
    start, init_density = min(enumerate(removed_scaled),
                       key=lambda x: abs(x[1]-start_density))
    initial_i = weekly_0[start] / rural_area
    population = rural_pop / rural_pop
elif setting == 2:
    # this is for scaling by city area only
    removed_scaled = removed / city_area
    start, init_density = min(enumerate(removed_scaled),
                       key=lambda x: abs(x[1]-start_density))
    initial_i = weekly_0[start] / city_area
    population = rural_pop / city_area
elif setting == 3:
    # this is for scaling by city population
    # (ratio of CITY to county)
    removed_scaled = removed * cr_pop_ratio
    start, init_density = min(enumerate(removed_scaled),
                       key=lambda x: abs(x[1]-start_density))
    initial_i = weekly_0[start]
    population = city_pop
elif setting == 4:
    # this is for scaling by URBAN population
    # (ratio of URBAN to county)
    removed_scaled = removed * ur_pop_ratio
    start, init_density = min(enumerate(removed_scaled),
                       key=lambda x: abs(x[1]-start_density))
    initial_i = weekly_0[start]
    population = urban_p
elif setting == 5:
    # this is for scaling by city population
    # then by area of city
    removed_scaled = removed * cr_pop_ratio / city_area
    start, init_density = min(enumerate(removed_scaled),
                       key=lambda x: abs(x[1]-start_density))
    initial_i = weekly_0[start] / city_area
    population = city_pop / city_area
elif setting == 6:
    # this is for scaling by URBAN population
    # then by area of largest city
    removed_scaled = removed * ur_pop_ratio / city_area
    start, init_density = min(enumerate(removed_scaled),
                       key=lambda x: abs(x[1]-start_density))
    initial_i = weekly_0[start] / city_area
    population = urban_p / city_area

if susceptible == True:

    # this is to create the susceptible data
    removed_scaled = population - removed_scaled

    ## This runs through the different possible endpoint to
    ## show the MSEs for each in order to find the best fit
    for j in range(len(end_array)):

        end = end_array[j]
        removed_new = removed_scaled[start:int(end)-10]

        time = np.linspace(0,len(removed_new)-1, len(removed_new))

        I0,R0 = initial_i, init_density
        S0 = removed_new[0] - I0 - R0

        initialParameters = [ip[0],ip[1], I0]

        fittedParameters,pcov = optimize.curve_fit(odesolveS1,
                                                time,
                                                removed_new,
                                                p0=tuple(initial
                                                Parameters))
        y0 = S0, fittedParameters[2], R0
        ret = integrate.odeint(deriv, y0, time,
                            args=tuple(fittedParameters[:2]))
        S, I, R = ret.T

        scores[j] = MSE(removed_new, R)


    min_index = scores.argmin()

    end = end_array[min_index]

    ## This section redoes the fit using the min_index to
    ## output to the results dataframe

    removed_new = removed_scaled[start:int(end)-10]

    time = np.linspace(0,len(removed_new)-1, len(removed_new))

    I0,R0 = initial_i, init_density
    S0 = removed_new[0] - I0 - R0

    initialParameters = [ip[0],ip[1], I0]

    fittedParameters,pcov   = optimize.curve_fit(odesolveS1,
                                                time,
                                                   removed_new,
                                                   p0=tuple(initial
                                                   Parameters))
else:

    ## This runs through the different possible endpoint to
    ## show the MSEs for each in order to find the best fit
    for j in range(len(end_array)):

        end = end_array[j]
        removed_new = removed_scaled[start:int(end)-10]

        time = np.linspace(0,len(removed_new)-1, len(removed_new))

        I0,R0 = initial_i, init_density
        S0 = population - I0 - R0

        initialParameters = [ip[0],ip[1], I0]

        fittedParameters,pcov = optimize.curve_fit(odesolveR1,
                                                   time,
                                                   removed_new,
                                                   p0=tuple(initial
                                                   Parameters))
        y0 = S0, fittedParameters[2], R0
        ret = integrate.odeint(deriv, y0, time,
                              args=tuple(fittedParameters[:2]))
        S, I, R = ret.T

        scores[j] = MSE(removed_new, R)


    min_index = scores.argmin()

    end = end_array[min_index]

    ## This section redoes the fit using the min_index to
    ## output to the results dataframe

    removed_new = removed_scaled[start:int(end)-10]

    time = np.linspace(0,len(removed_new)-1, len(removed_new))

    I0,R0 = initial_i, init_density
    S0 = population - I0 - R0

    initialParameters = [ip[0],ip[1], I0]

    fittedParameters,pcov    = optimize.curve_fit(odesolveR1,
                                                   time,
                                                   removed_new,
                                                   p0=tuple(initial
                                                   Parameters))
results_df[county_name] = [start + 10,
                           end,
                           fittedParameters[0],
                           fittedParameters[1],
                           fittedParameters[2],
                           ua_df[ua_df['County']
                                == county_name]['City'].iloc[0],
                           city_pop,
                           urban_p,
                           rural_pop,
                           city_area,
                           rural_area,
                           cr_pop_ratio,
                           ur_pop_ratio,
                           city_pop/city_area,
                           rural_pop/rural_area,
                           cr_pop_ratio/city_area,
                           scores.min() ]
```

Delta Wave

The code shown here is for the fitting the modified SIR model outlined in Equations (5.1)-(5.3). The code that does not account for vaccination is not included here. In addition, it should be noted that while the majority of the code is the same, there are some changes to take into account the Delta wave's location in the timeline as well as the vaccination constant.

```
# number of counties to look at
n = 6

susceptible = False
override = False
start_week = 73
end_week = 93

# Initial Parameter Guesses
ip = [3.714e-05,7] # alpha, beta

### Setting codes:
#
#   0: No scaling whatsoever
#   1: Scaling by COUNTY area (county density)
#   2: Scaling by CITY area (city density via area of largest city)
#   3: Scaling by CITY/COUNTY ratio (largest city pop / county pop)
#   4: Scaling by URBAN/COUNTY ratio (urban population / county pop)
#   5: Scaling by CITY/COUNTY ratio and city area(largest city pop
#                                          / county pop / city area)
#   6: Scaling by URBAN/COUNTY ratio and city area(urban population
#                                          / county pop / city area)
#
###
```

```
setting = 5


results_df = pd.DataFrame(['Start Week',
                           'End Week',
                           'alpha',
                           'beta',
                           'gamma',
                           'Initial Infected',
                           'City',
                           'City Population',
                           'Urban Population',
                           'County Population',
                           'City Area',
                           'County Area',
                           'City/Rural Population Ratio',
                           'Urban/Rural Population Ratio',
                           'City Pop Density',
                           'Rural Pop Density',
                           'City/Rural Ratio divided by city area',
                           'MSE'], columns = ['Parameters'])


top_counties = np.argsort(county_pop)[::-1][:n]

for h in range(n):
    # Find name of county
    county_name = county_list[top_counties[h]]
    county_info = combined_map.loc[combined_map['NAME']
                        == county_name.upper()]

    # Specific city population and area, not the overall urban one.
    city_pop = ua_df[ua_df['County']
                        == county_name]['Populations'].iloc[0]
    city_area = ua_df[ua_df['County']
                        == county_name]['Area'].iloc[0]

    # County (rural) area
    rural_area = combined_map[combined_map['NAME']
                        == county_name.upper()]['SQMILES'].iloc[0]

    # Find index for the county
    county_index = np.where(county_list == county_name)[0][0]

    # pull the daily infected rates and transform
    # it into a weekly average
    county_data = DataLoad(county_index)
    daily, weekly_0 = InfectedData(county_data)

    weekly_1 = weekly_0[10:]

    # Vaccination Rates
    v_df_MT_county = v_df_MT[v_df_MT['Recip_County']
    == county_name + str(' County')].reset_index(drop = True)
county_vac_daily = v_df_MT_county['Administered_Dose1_Recip']
                                            .iloc[20:].values
county_vac_weekly = np.add.reduceat(county_vac_daily,
                    np.arange(0, len(county_vac_daily), 7)) /7
county_vac_weekly = np.concatenate((v_df_MT_county
    ['Administered_Dose1_Recip'].iloc[:20].values.astype(int),
    county_vac_weekly.astype(int),
    np.zeros(40).astype(int)))
county_vac_weekly = county_vac_weekly[::-1]


### This creates the removed dataset
removed = np.zeros(len(weekly_1))
removed[0] = weekly_1[0]
for k in range(len(weekly_1)-1):
    removed[k+1] = removed[k]+weekly_1[k+1]
###

# this takes into account that we arn't doing the first wave

removed -= removed[start_week-10]

# county TOTAL population
rural_pop = county_pop[county_index]
# county URBAN population
urban_p = combined_map[combined_map['NAME']
                == county_name.upper()]['URBANPOP'].iloc[0]

ur_pop_ratio = urban_p/rural_pop      # urban/rural pop ratio
cr_pop_ratio = city_pop/rural_pop       # city/rural pop ratio

if setting == 0:
    # this is for absolutely no scaling whatsoever
    removed_scaled = removed * 1
    initial_i = weekly_0[start_week]
    population = rural_pop
elif setting == 1:
    # this is for scaling only by county area (county density)
    removed_scaled = removed / rural_area
    initial_i = weekly_0[start_week] / rural_area
    population = rural_pop / rural_pop
elif setting == 2:
    # this is for scaling by city area only
    removed_scaled = removed / city_area
    initial_i = weekly_0[start_week] / city_area
    population = rural_pop / city_area
elif setting == 3:
    # this is for scaling by city population
    # (ratio of CITY to county)
    removed_scaled = removed * cr_pop_ratio
    initial_i = weekly_0[start_week]
    population = city_pop
elif setting == 4:
        # this is for scaling by URBAN population
        # (ration of URBAN to county)
        removed_scaled = removed * ur_pop_ratio
        initial_i = weekly_0[start_week]
        population = urban_p
    elif setting == 5:
        # this is for scaling by city population
        # then by area of city
        removed_scaled = removed * cr_pop_ratio / city_area
        initial_i = weekly_0[start_week-10] * cr_pop_ratio / city_area
        population = city_pop / city_area
    elif setting == 6:
        # this is for scaling by URBAN population
        # then by area of largest city
        removed_scaled = removed * ur_pop_ratio / city_area
        initial_i = weekly_0[start_week] / city_area
        population = urban_p / city_area

    if susceptible == True:
        # this is to create the susceptible data
        removed_scaled = population - removed_scaled

        end = end_week
        removed_new = removed_scaled[start_week:int(end)-10]

        time = np.linspace(0,len(removed_new)-1, len(removed_new))

        I0,R0 = initial_i, init_density
        S0 = removed_new[0] - I0 - R0

        initialParameters = [ip[0],ip[1], I0]

        fittedParameters,pcov    = optimize.curve_fit(odesolveS1,
                                           time,
                                           removed_new,
                                           p0=tuple(initialParameters))
        score = MSE(removed_new, R)

    else:

        end = end_week
        c_vac_new = county_vac_weekly[10:111]
                              - county_vac_weekly[start_week-10]

        removed_new = removed_scaled[start_week-10:int(end)-10]

        time = np.linspace(0,len(removed_new)-1, len(removed_new))

        vac_rates = county_vac_weekly[start_week-10:int(end)-10]
        slope, intercept = np.polyfit(time, vac_rates, 1)
        gamma = slope * cr_pop_ratio / city_area
        gamma = 0 if override == True else slope *
                                           cr_pop_ratio / city_area
        I0,R0 = initial_i, removed_new[0]
        S0 = population - I0 - R0

        initialParameters = [ip[0],ip[1], I0]

        fittedParameters,pcov   = optimize.curve_fit(odesolveR3,
                                              time,
                                              removed_new,
                                              p0=tuple(initialParameters))
        y0 = S0, fittedParameters[2], R0
        ret = integrate.odeint(deriv2,
                               y0,
                               time,
                               args=tuple(fittedParameters[:2]))
        S, I, R = ret.T

        score = MSE(removed_new, R)

    results_df[county_name] = [start_week,
                               end,
                               fittedParameters[0],
                               fittedParameters[1],
                               gamma,
                               fittedParameters[2],
                               ua_df[ua_df['County']
                                   == county_name]['City'].iloc[0],
                               city_pop,
                               urban_p,
                               rural_pop,
                               city_area,
                               rural_area,
                               cr_pop_ratio,
                               ur_pop_ratio,
                               city_pop/city_area,
                               rural_pop/rural_area,
                               cr_pop_ratio/city_area,
                               score ]
```

Omicron Wave

The code for the Omicron wave is nearly identical to that of the Alpha wave with a couple of exceptions. Much like with the Delta wave, since it was not the first wave, because of how the removed dataset was computed, we only want to look at the removed cases for the current wave.

```
n = 6 #number of counties to look at
susceptible = False
start_week = 93
end_week = 106

# Intial Parameter Guesses
ip = [3.714e-04,7]
### Setting codes:
#
```

```
#   0: No scaling whatsoever
#   1: Scaling by COUNTY area (county density)
#   2: Scaling by CITY area (city density via area of largest city)
#   3: Scaling by CITY/COUNTY ratio (largest city pop / county pop)
#   4: Scaling by URBAN/COUNTY ratio (urban population / county pop)
#   5: Scaling by CITY/COUNTY ratio and city area
#      (largest city pop / county pop / city area)
#   6: Scaling by URBAN/COUNTY ratio and city area
#      (urban population / county pop / city area)
#
###

setting = 5

results_df = pd.DataFrame(['Start Week',
                           'End Week',
                           'alpha',
                           'beta',
                           'Initial Infected',
                           'City',
                           'City Population',
                           'Urban Population',
                           'County Population',
                           'City Area',
                           'County Area',
                           'City/Rural Population Ratio',
                           'Urban/Rural Population Ratio',
                           'City Pop Density',
                           'Rural Pop Density',
                           'City/Rural Ratio divided by city area',
                           'MSE'], columns = ['Parameters'])


top_counties = np.argsort(county_pop)[::-1][:n]
scores = np.zeros(len(end_array))

for h in range(n):
    # Find name of county
    county_name = county_list[top_counties[h]]
    county_info = combined_map.loc[combined_map['NAME']
                                    == county_name.upper()]

    # Specific city population and area, not the overall urban one.
    city_pop = ua_df[ua_df['County']
                       == county_name]['Populations'].iloc[0]
    city_area = ua_df[ua_df['County']
                       == county_name]['Area'].iloc[0]

    # County (rural) area
    rural_area = combined_map[combined_map['NAME']
                       == county_name.upper()]['SQMILES'].iloc[0]

    # Find index for the county
    county_index = np.where(county_list == county_name)[0][0]
    # pull the daily infected rates and
    # transform it into a weekly average
    county_data = DataLoad(county_index)
    daily, weekly_0 = InfectedData(county_data)

    weekly_1 = weekly_0[10:]

    ### This creates the removed dataset
    removed = np.zeros(len(weekly_1))
    removed[0] = weekly_1[0]
    for k in range(len(weekly_1)-1):
        removed[k+1] = removed[k]+weekly_1[k+1]
    ###
    # this takes into account that we are not doing the first wave

    removed -= removed[start_week-10]

    # county TOTAL population
    rural_pop = county_pop[county_index]
    # county URBAN population
    urban_p = combined_map[combined_map['NAME']
                        == county_name.upper()]['URBANPOP'].iloc[0]

    ur_pop_ratio = urban_p/rural_pop       # urban/rural pop ratio
    cr_pop_ratio = city_pop/rural_pop        # city/rural pop ratio

    if setting == 0:
        # this is for absolutely no scaling whatsoever
        removed_scaled = removed * 1
        initial_i = weekly_0[start_week+10]
        population = rural_pop
    elif setting == 1:
        # this is for scaling only by county area (county density)
        removed_scaled = removed / rural_area
        initial_i = weekly_0[start_week+10] / rural_area
        population = rural_pop / rural_pop
    elif setting == 2:
        # this is for scaling by city area only
        removed_scaled = removed / city_area
        initial_i = weekly_0[start_week+10] / city_area
        population = rural_pop / city_area
    elif setting == 3:
        # this is for scaling by city population
        # (ratio of CITY to county)
        removed_scaled = removed * cr_pop_ratio
        initial_i = weekly_0[start_week+10]
        population = city_pop
    elif setting == 4:
        # this is for scaling by URBAN population
        # (ration of URBAN to county)
        removed_scaled = removed * ur_pop_ratio
        initial_i = weekly_0[start_week+10]
        population = urban_p
    elif setting == 5:
        # this is for scaling by city population
        # then by area of city
        removed_scaled = removed * cr_pop_ratio / city_area
        initial_i = weekly_0[start_week+10] / city_area
        population = city_pop / city_area
    elif setting == 6:
        # this is for scaling by URBAN population
        # then by area of largest city
        removed_scaled = removed * ur_pop_ratio / city_area
        initial_i = weekly_0[start_week+10] / city_area
        population = urban_p / city_area

    if susceptible == True:
        # this is to create the susceptible data
        removed_scaled = population - removed_scaled

        end = end_week
        removed_new = removed_scaled[start_week:int(end)-10]
        time = np.linspace(0,len(removed_new)-1, len(removed_new))
        I0,R0 = initial_i, init_density
        S0 = removed_new[0] - I0 - R0

        initialParameters = [ip[0],ip[1], I0]

        fittedParameters,pcov    = optimize.curve_fit(odesolveS1,
                                        time,
                                        removed_new,
                                        p0=tuple(initialParameters))
        y0 = S0, fittedParameters[2], R0
        ret = integrate.odeint(deriv,
                               y0,
                               time, args=tuple(fittedParameters[:2]))
        S, I, R = ret.T

        score = MSE(removed_new, R)


    else:

        end = end_week
        removed_new = removed_scaled[start_week-10:int(end)-10]

        time = np.linspace(0,len(removed_new)-1, len(removed_new))

        I0,R0 = initial_i, removed_new[0]
        S0 = population - I0 - R0

        initialParameters = [ip[0],ip[1], I0]

        fittedParameters,pcov    = optimize.curve_fit(odesolveR1,
                                time,
                                removed_new,
                                p0=tuple(initialParameters))
        y0 = S0, fittedParameters[2], R0
        ret = integrate.odeint(deriv,
                               y0,
                               time,
                               args=tuple(fittedParameters[:2]))
        S, I, R = ret.T

        score = MSE(removed_new, R)



    results_df[county_name] = [start_week,
                               end,
                               fittedParameters[0],
                               fittedParameters[1],
                               fittedParameters[2],
                               ua_df[ua_df['County']
                                    == county_name]['City'].iloc[0],
                               city_pop,
                               urban_p,
                               rural_pop,
                               city_area,
                               rural_area,
                               cr_pop_ratio,
                               ur_pop_ratio,
                               city_pop/city_area,
                               rural_pop/rural_area,
                               cr_pop_ratio/city_area,
                               score ]
```

A.5 SIR Model Fitted Parameter Visualization

The following codes are used specifically to create the multigraphs that overlay the SIR models with the fitted parameters with the data.

Alpha Wave Fitted Parameter Visualization Code

This code is used to generate Figure 6.6

```
cols = results_df.columns[1:].tolist()
fig, axes = plt.subplots(nrows=2, ncols=3, figsize = (24,15))
count = 0
for col in cols:
    start, end = results_df[col][0], results_df[col][1]
    county_name = col
    county_info = combined_map.loc[combined_map['NAME']
                        == county_name.upper()]
```

```
# Specific city population and area, not the overall urban one.
city_pop = ua_df[ua_df['County']
                    == county_name]['Populations'].iloc[0]
city_area = ua_df[ua_df['County']
                    == county_name]['Area'].iloc[0]

# County (rural) area
rural_area = combined_map[combined_map['NAME']
                    == county_name.upper()]['SQMILES'].iloc[0]

# Find index for the county
county_index = np.where(county_list == county_name)[0][0]

# pull the daily infected rates and
# transform it into a weekly average
county_data = DataLoad(county_index)
daily, weekly_0 = InfectedData(county_data)

weekly_1 = weekly_0[10:70]

### This creates the removed dataset
removed = np.zeros(len(weekly_1))
removed[0] = weekly_1[0]
for k in range(len(weekly_1)-1):
    removed[k+1] = removed[k]+weekly_1[k+1]
###

# county TOTAL population
rural_pop = county_pop[county_index]
# county URBAN population
urban_p = combined_map[combined_map['NAME']
                == county_name.upper()]['URBANPOP'].iloc[0]

ur_pop_ratio = urban_p/rural_pop      # urban/rural pop ratio
cr_pop_ratio = city_pop/rural_pop       # city/rural pop ratio

removed_scaled = removed * cr_pop_ratio / city_area
start, init_density = min(enumerate(removed_scaled),
                         key=lambda x: abs(x[1]-start_density))
initial_i = weekly_0[start] / city_area
population = city_pop / city_area
removed_new = removed_scaled[start:int(end)-10]



time = np.linspace(int(start)+10,
                   int(end)-1,
                   int(end)-int(start)-10)
I0,R0 = results_df[col][4], init_density
S0 = population - I0 - R0
y0 = S0, results_df[col][4], R0
ret = integrate.odeint(deriv,
                    y0,
                        time,
                        args=tuple(results_df[col][2:4].tolist()))
    S, I, R = ret.T

    axes[int(count/3), count%3].plot(time,
                                     removed_new,
                                     label = 'Data')
    axes[int(count/3), count%3].plot(time,
                                     R,
                                     label = 'Fitted Model
                                                    Parameters')
    axes[int(count/3), count%3].set_title(county_name)
    axes[int(count/3), count%3].set_xlabel('Weeks')
    axes[int(count/3), count%3].set_ylabel('Removed Density')
    count += 1

plt.legend()
```

Delta Wave with Vaccination Fitted Parameter Visualization Code

This code is used to generate Figure 6.11

```
cols = results_df.columns[1:].tolist()
fig, axes = plt.subplots(nrows=2, ncols=3, figsize = (24,15))
count = 0
for col in cols:
    start, end = results_df[col][0], results_df[col][1]
    county_name = col
    county_info = combined_map.loc[combined_map['NAME']
                        == county_name.upper()]

    # Specific city population and area, not the overall urban one.
    city_pop = ua_df[ua_df['County']
                        == county_name]['Populations'].iloc[0]
    city_area = ua_df[ua_df['County']
                        == county_name]['Area'].iloc[0]

    # County (rural) area
    rural_area = combined_map[combined_map['NAME']
                        == county_name.upper()]['SQMILES'].iloc[0]

    # Find index for the county
    county_index = np.where(county_list == county_name)[0][0]

    # pull the daily infected rates and
    # transform it into a weekly average
    county_data = DataLoad(county_index)
    daily, weekly_0 = InfectedData(county_data)

    weekly_1 = weekly_0[10:]

    ### This creates the removed dataset
    removed = np.zeros(len(weekly_1))
    removed[0] = weekly_1[0]
```

```
for k in range(len(weekly_1)-1):
    removed[k+1] = removed[k]+weekly_1[k+1]
###

removed -= removed[start_week-10]
# county TOTAL population
rural_pop = county_pop[county_index]
# county URBAN population
urban_p = combined_map[combined_map['NAME']
                == county_name.upper()]['URBANPOP'].iloc[0]

ur_pop_ratio = urban_p/rural_pop      # urban/rural pop ratio
cr_pop_ratio = city_pop/rural_pop       # city/rural pop ratio

removed_scaled = removed * cr_pop_ratio / city_area
start, init_density = min(enumerate(removed_scaled),
                         key=lambda x: abs(x[1]-start_density))
initial_i = weekly_0[start] / city_area
population = city_pop / city_area

removed_new = removed_scaled[start_week-10:int(end)-10]

time = np.linspace(0,len(removed_new)-1, len(removed_new))

vac_rates = county_vac_weekly[start_week-10:int(end)-10]
slope, intercept = np.polyfit(time, vac_rates, 1)
gamma = slope * cr_pop_ratio / city_area
gamma = 0 if override == True else slope
                                 * cr_pop_ratio / city_area
initial_infected = weekly_1[start_week]
                                 * cr_pop_ratio / city_area

I0,R0 = weekly_0[start_week-10] * cr_pop_ratio / city_area,
        removed_new[0]
S0 = population - I0 - R0

initialParameters = [ip[0],ip[1], I0]

fittedParameters,pcov   = optimize.curve_fit(odesolveR3,
                                     time,
                                     removed_new,
                                     p0=tuple(initialParameters))
y0 = S0, fittedParameters[2], R0
ret = integrate.odeint(deriv2,
                       y0,
                       time,
                       args=tuple(fittedParameters[:2]))
S, I, R = ret.T
time = np.linspace(start_week,int(end), int(end)-start_week)
axes[int(count/3), count%3].plot(time,
                                  removed_new,
                                  label = 'Data')
axes[int(count/3), count%3].plot(time,
                                  R,
                                label = 'Fitted SIR model')
    axes[int(count/3), count%3].set_title(county_name)
    axes[int(count/3), count%3].set_xticks((75,80,85, 90))
    axes[int(count/3), count%3].set_xlabel('Weeks')
    axes[int(count/3), count%3].set_ylabel('Removed Density')
    count += 1

plt.legend()
```

Omicron Wave Fitted Parameter Visualization Code

This code is used to generate Figure 6.15

```
cols = results_df.columns[1:].tolist()
fig, axes = plt.subplots(nrows=2, ncols=3, figsize = (24,15))
count = 0
for col in cols:
    start, end = results_df[col][0], results_df[col][1]
    county_name = col
    county_info = combined_map.loc[combined_map['NAME']
                    == county_name.upper()]

    # Specific city population and area, not the overall urban one.
    city_pop = ua_df[ua_df['County']
                    == county_name]['Populations'].iloc[0]
    city_area = ua_df[ua_df['County']
                    == county_name]['Area'].iloc[0]

    # County (rural) area
    rural_area = combined_map[combined_map['NAME']
                    == county_name.upper()]['SQMILES'].iloc[0]

    # Find index for the county
    county_index = np.where(county_list == county_name)[0][0]

    # pull the daily infected rates and
    # transform it into a weekly average
    county_data = DataLoad(county_index)
    daily, weekly_0 = InfectedData(county_data)

    weekly_1 = weekly_0[10:]

    ### This creates the removed dataset
    removed = np.zeros(len(weekly_1))
    removed[0] = weekly_1[0]
    for k in range(len(weekly_1)-1):
        removed[k+1] = removed[k]+weekly_1[k+1]
    ###

    removed -= removed[start-10]

    # county TOTAL population
    rural_pop = county_pop[county_index]
    # county URBAN population
    urban_p = combined_map[combined_map['NAME']
                    == county_name.upper()]['URBANPOP'].iloc[0]

    ur_pop_ratio = urban_p/rural_pop      # urban/rural pop ratio
    cr_pop_ratio = city_pop/rural_pop       # city/rural pop ratio

    removed_scaled = removed * cr_pop_ratio / city_area
    population = city_pop / city_area
    removed_new = removed_scaled[start-10:int(end)-10]


    # time = np.linspace(0,len(removed_new)-1, len(removed_new))
    time = np.linspace(int(start),int(end)-1, int(end)-int(start))
    I0,R0 = results_df[col][4], removed_new[0]
    S0 = population - I0 - R0
    y0 = S0, results_df[col][4], R0
    ret = integrate.odeint(deriv,
                           y0,
                           time,
                           args=tuple(results_df[col][2:4].tolist()))
    S, I, R = ret.T

    axes[int(count/3), count%3].plot(time,
                                     removed_new,
                                     label = 'Data')
    axes[int(count/3), count%3].plot(time,
                                     R,
                                     label = 'Fitted SIR model')
    axes[int(count/3), count%3].set_title(county_name)
    axes[int(count/3), count%3].set_xlabel('Weeks')
    axes[int(count/3), count%3].set_ylabel('Removed Density')
    count += 1

plt.legend()
```

Appendix B

Age Demographic Analysis Code

The following sections of code were used to create all of the figures used in Chapter 6. Not all of the figures produced were used in the dissertation, as they were used to explore certain aspects of the data that were not discussed.

Data Loading and Processing

The figures in Chapter 6 used separate datasets from those of previous chapters, and thus required their own import and processing.

```
# This loads the case and death data as well as
# census proportions for Montana

full_data = pd.read_csv('data_files/arcgis_data.csv',
                          low_memory = True)
full_data.rename(columns =
               {'Date_Reported_to_CDEpi':'Date_Reported'},
               inplace = True)

deaths_df = pd.read_csv('Provisional_COVID-
                          19_Deaths_by_Sex_and_Age.csv',
                          low_memory = True)

census_data = {'Labels': ['Total Population', '<5', '5-9', '10-14',
                   '15-19', '20-24', '25-29', '30-34', '35-39',
                   '40-44', '45-49', '50-54', '55-59','60-64',
                   '65-69','70-74','75-79','80-84','>85'],
              'United States': [331893745, 18661245, 20010813,
              21821492, 21824088, 21382643, 22100453, 22978685,
              22371398, 21362163, 19782325, 20891392, 21141152,
              21673882, 18351785, 15426419, 9872768, 6278369,
              5962673]}
cd_df = pd.DataFrame(data = census_data)
```

```
census_df = pd.read_csv('census_age_data.csv', low_memory = False)
census_df1 = census_df[census_df.GeographicAreaName.str
                                .contains('|'.join(["Montana"]))]
census_df2 = census_df1[census_df1.columns.drop(
                        list(census_df1.filter(regex='Margin')))]
census_df3 = census_df2[census_df2.columns.drop(
                    list(census_df2.filter(regex='Annotation')))]
census_MT_dict = {census_df3.iloc[x][1]:
                census_df3.iloc[x][2:21].tolist() for x in range(7)}
census_MT_df = pd.DataFrame(data = census_MT_dict)

demo_df = pd.concat([cd_df, census_MT_df], axis=1)

label_dict = {'Labels': ['<5', '5-9', '10-14', '15-19', '20-24',
                        '25-29', '30-34', '35-39', '40-44',
                        '45-49', '50-54', '55-59','60-64',
                        '65-69','70-74','75-79','80-84','>85']}
l_df = pd.DataFrame(data = label_dict)
prop_dict = {col: demo_df[col][1:].tolist()/demo_df[col][0]
                                for col in demo_df.columns[1:]}
prop1_df = pd.DataFrame(data = prop_dict)

# PROPORTION DATAFRAME for demographics
prop_df = pd.concat([l_df, prop1_df], axis=1)

deaths_df1 = deaths_df[deaths_df['State'] == 'United States']
deaths_df1 = deaths_df1[deaths_df1['Sex'] == 'All Sexes']
deaths_df_totals = deaths_df1[deaths_df1['Group'] == 'By Total']

case_data = {'Labels': ['Total','<5', '5-17','18-29', '30-39',
                            '40-49','50-64','65-74','75-84','>85'],
                        'Total US Cases':[96599571,3488372,
                        6191919+4289218+2500564,19428602,
                        15871170, 13509412,17745282, 7062119,
                        3728044, 1834789],
                        'Total US Deaths': [1105405, 424+235,
                        1471-424-235, 6820, 19403, 45410, 198955,
                        249280, 287768, 296298]}
case_df = pd.DataFrame(data = case_data)

case_df['Case-Fatality Rates'] = case_df['Total US Deaths'] /
                                    case_df['Total US Cases']

prop_df_new = pd.DataFrame(index = ['0-9','10-19','20-29',
                                    '30-39','40-49','50-59',
                                    '60-69','70+'],
                                    columns = prop_df.columns)
prop_df_new.iloc[0][0] = '0-9'
prop_df_new.iloc[0][1:] = sum(prop_df.iloc[i][1:] for i in [0,1])
prop_df_new.iloc[1][0] = '10-19'
prop_df_new.iloc[1][1:] = sum(prop_df.iloc[i][1:] for i in [2,3])
prop_df_new.iloc[2][0] = '20-29'
prop_df_new.iloc[2][1:] = sum(prop_df.iloc[i][1:] for i in [4,5])
prop_df_new.iloc[3][0] = '30-39'
prop_df_new.iloc[3][1:] = sum(prop_df.iloc[i][1:] for i in [6,7])
prop_df_new.iloc[4][0] = '40-49'
prop_df_new.iloc[4][1:] = sum(prop_df.iloc[i][1:] for i in [8,9])
prop_df_new.iloc[5][0] = '50-59'
prop_df_new.iloc[5][1:] = sum(prop_df.iloc[i][1:] for i in [10,11])
prop_df_new.iloc[6][0] = '60-69'
prop_df_new.iloc[6][1:] = sum(prop_df.iloc[i][1:] for i in [12,13])
prop_df_new.iloc[7][0] = '70+'
prop_df_new.iloc[7][1:] = sum(prop_df.iloc[i][1:]
                                          for i in [14,15,16,17])


prop_df_new = prop_df_new.drop(columns = ['United States',
                                          'Montana',
                                          'Labels'])
prop_df_new.rename(columns = {'Cascade County, Montana':'Cascade',
                              'Flathead County, Montana':'Flathead',
                              'Gallatin County, Montana':'Gallatin',
                              'Lewis and Clark County, Montana':
                              'Lewis and Clark', 'Missoula County,
                              Montana':'Missoula','Yellowstone
                              County, Montana':'Yellowstone'},
                              inplace = True)
prop_df_new = prop_df_new[['Yellowstone', 'Gallatin', 'Missoula',
                        'Flathead', 'Cascade', 'Lewis and Clark']]

prop_df_new.plot.bar(rot=0)
plt.xlabel('Ages')
plt.ylabel('Percentage of Population')
```

Alpha wave

```
# Alpha wave case and death visualization

start_date = '2020/06/10 12:00:00+00'
end_date = '2021/03/17 12:00:00+00'
setting = 2    # deaths
setting2 = 2   # cases
n = 6

start_index = full_data.Date_Reported.values.searchsorted(start_date,
                                                  side = 'left')
end_index = full_data.Date_Reported.values.searchsorted(end_date,
                                                  side = 'right')

trunc_data = full_data[start_index:end_index]


time_range_data_deaths = pd.DataFrame(index = ['Total Cases',
                               '0-9','10-19','20-29','30-39',
                               '40-49','50-59','60-69','70+'] )
top_counties = np.argsort(county_pop)[::-1][:n]
for h in range(n):
    county_name = county_list[top_counties[h]]

    city_area = ua_df[ua_df['County']
                    == county_name]['Area'].iloc[0]
    city_pop = ua_df[ua_df['County']
                    == county_name]['Populations'].iloc[0]
    rural_pop = combined_map[combined_map['NAME']
                    == county_name.upper()]['COUNTYPOP'].iloc[0]
    pop_ratio = city_pop/rural_pop
    if setting == 0:
        mult = 1
    elif setting == 1:
        mult = 1/city_pop
    elif setting == 2:
        mult = 1/rural_pop
    elif setting == 3:
        mult = pop_ratio
    elif setting == 4:
        mult = pop_ratio / city_area

    col_entry = np.zeros(9)
    temp_data = trunc_data[trunc_data['County'] == county_name]
    col_entry[0] = temp_data.shape[0]
    col_entry[1] = temp_data[temp_data['Age_Group'] == '0-9']
[temp_data[temp_data['Age_Group'] == '0-9']
                    ['Outcome'] == 'Deceased'].shape[0]
    col_entry[2] = temp_data[temp_data['Age_Group'] == '10-19']
                    [temp_data[temp_data['Age_Group'] == '10-19']
                    ['Outcome'] == 'Deceased'].shape[0]
    col_entry[3] = temp_data[temp_data['Age_Group'] == '20-29']
                    [temp_data[temp_data['Age_Group'] == '20-29']
                    ['Outcome'] == 'Deceased'].shape[0]
    col_entry[4] = temp_data[temp_data['Age_Group'] == '30-39']
                    [temp_data[temp_data['Age_Group'] == '30-39']
                    ['Outcome'] == 'Deceased'].shape[0]
    col_entry[5] = temp_data[temp_data['Age_Group'] == '40-49']
                    [temp_data[temp_data['Age_Group'] == '40-49']
                    ['Outcome'] == 'Deceased'].shape[0]
    col_entry[6] = temp_data[temp_data['Age_Group'] == '50-59']
                    [temp_data[temp_data['Age_Group'] == '50-59']
                    ['Outcome'] == 'Deceased'].shape[0]
    col_entry[7] = temp_data[temp_data['Age_Group'] == '60-69']
                    [temp_data[temp_data['Age_Group'] == '60-69']
                    ['Outcome'] == 'Deceased'].shape[0]
    col_entry[8] = temp_data[temp_data['Age_Group'] == '70-79']
            [temp_data[temp_data['Age_Group'] == '70-79']
            ['Outcome'] == 'Deceased'].shape[0] +
            temp_data[temp_data['Age_Group'] == '80-89']
            [temp_data[temp_data['Age_Group'] == '80-89']
            ['Outcome'] == 'Deceased'].shape[0] +
            temp_data[temp_data['Age_Group'] == '90-99']
            [temp_data[temp_data['Age_Group'] == '90-99']
            ['Outcome'] == 'Deceased'].shape[0] +
            temp_data[temp_data['Age_Group'] == '100']
            [temp_data[temp_data['Age_Group'] == '100']
            ['Outcome'] == 'Deceased'].shape[0]

    time_range_data_deaths[county_name] = col_entry * mult

time_range_data_cases = pd.DataFrame(index = ['Total Cases',
                    '0-9','10-19','20-29','30-39',
                    '40-49','50-59','60-69','70+'] )
top_counties = np.argsort(county_pop)[::-1][:n]
for h in range(n):

    county_name = county_list[top_counties[h]]

    city_area = ua_df[ua_df['County']
                    == county_name]['Area'].iloc[0]
    city_pop = ua_df[ua_df['County']
                    == county_name]['Populations'].iloc[0]
    rural_pop = combined_map[combined_map['NAME']
                    == county_name.upper()]['COUNTYPOP'].iloc[0]
    pop_ratio = city_pop/rural_pop
    if setting2 == 0:
        mult = 1
    elif setting2 == 1:
        mult = 1/city_pop
    elif setting2 == 2:
        mult = 1/rural_pop
    elif setting2 == 3:
        mult = pop_ratio
    elif setting2 == 4:
        mult = pop_ratio / city_area

    col_entry = np.zeros(9)
    temp_data = trunc_data[trunc_data['County'] == county_name]
    col_entry[0] = temp_data.shape[0]
    col_entry[1] = temp_data[temp_data['Age_Group']
                    == '0-9'].shape[0]
    col_entry[2] = temp_data[temp_data['Age_Group']
                    == '10-19'].shape[0]
    col_entry[3] = temp_data[temp_data['Age_Group']
                    == '20-29'].shape[0]
    col_entry[4] = temp_data[temp_data['Age_Group']
                    == '30-39'].shape[0]
    col_entry[5] = temp_data[temp_data['Age_Group']
                    == '40-49'].shape[0]
    col_entry[6] = temp_data[temp_data['Age_Group']
                    == '50-59'].shape[0]
    col_entry[7] = temp_data[temp_data['Age_Group']
                    == '60-69'].shape[0]
    col_entry[8] = temp_data[temp_data['Age_Group']
            == '70-79'].shape[0] + temp_data[temp_data['Age_Group']
            == '80-89'].shape[0] + temp_data[temp_data['Age_Group']
            == '90-99'].shape[0] + temp_data[temp_data['Age_Group']
            == '100'].shape[0]
    time_range_data_cases[county_name] = col_entry * mult

time_range_data_cases_percent = pd.DataFrame(index =
                ['Total Cases','0-9','10-19','20-29','30-39',
                '40-49','50-59','60-69','70+'] )
top_counties = np.argsort(county_pop)[::-1][:n]
for h in range(n):

    county_name = county_list[top_counties[h]]
    col_entry = np.zeros(9)
    temp_data = trunc_data[trunc_data['County'] == county_name]
    col_total = temp_data.shape[0]
    col_entry[0] = col_total
    col_entry[1] = temp_data[temp_data['Age_Group']
                == '0-9'].shape[0] / col_total
    col_entry[2] = temp_data[temp_data['Age_Group']
                == '10-19'].shape[0] / col_total
    col_entry[3] = temp_data[temp_data['Age_Group']
                == '20-29'].shape[0] / col_total
    col_entry[4] = temp_data[temp_data['Age_Group']
                == '30-39'].shape[0] / col_total
    col_entry[5] = temp_data[temp_data['Age_Group']
                == '40-49'].shape[0] / col_total
    col_entry[6] = temp_data[temp_data['Age_Group']
                == '50-59'].shape[0] / col_total
    col_entry[7] = temp_data[temp_data['Age_Group']
                == '60-69'].shape[0] / col_total
    col_entry[8] = (temp_data[temp_data['Age_Group']
            == '70-79'].shape[0] + temp_data[temp_data['Age_Group']
            == '80-89'].shape[0] + temp_data[temp_data['Age_Group']
            == '90-99'].shape[0] + temp_data[temp_data['Age_Group']
            == '100'].shape[0]) / col_total


    time_range_data_cases_percent[county_name] = col_entry


trdc = time_range_data_cases.copy().drop('Total Cases')
trdcp = time_range_data_cases_percent.copy().drop('Total Cases')
trdd = time_range_data_deaths.copy().drop('Total Cases')


ratios = trdd/ trdc

trdc.plot.bar(rot=0)
plt.xlabel('Ages')
plt.ylabel('Percentage of County Population')
trdcp.plot.bar(rot=0)
plt.xlabel('Ages')
plt.ylabel('Percent of Total Cases')
trdd.plot.bar(rot=0)
plt.xlabel('Ages')
plt.ylabel('Number of Deaths')
```

```
ratios.plot.bar(rot=0)
plt.xlabel('Ages')
plt.ylabel('Case Fatality Rate')
```

Delta wave

```
start_date = '2021/08/06 12:00:00+00'
end_date = '2021/12/24 12:00:00+00'
setting = 0   # deaths
setting2 = 0  # cases
n = 6

start_index = full_data.Date_Reported.values.
                          searchsorted(start_date, side = 'left')
end_index = full_data.Date_Reported.values.
                          searchsorted(end_date, side = 'right')

trunc_data = full_data[start_index:end_index]


time_range_data_deaths = pd.DataFrame(index = ['Total Cases',
                      '0-9','10-19','20-29','30-39','40-49',
                      '50-59','60-69','70+'] )
top_counties = np.argsort(county_pop)[:: -1][:n]
for h in range(n):

    county_name = county_list[top_counties[h]]

    city_area = ua_df[ua_df['County']
                      == county_name]['Area'].iloc[0]
    city_pop = ua_df[ua_df['County']
                      == county_name]['Populations'].iloc[0]
    rural_pop = combined_map[combined_map['NAME']
                      == county_name.upper()]['COUNTYPOP'].iloc[0]
    pop_ratio = city_pop/rural_pop
    if setting == 0:
        mult = 1
    elif setting == 1:
        mult = 1/city_pop
    elif setting == 2:
        mult = 1/rural_pop
    elif setting == 3:
        mult = pop_ratio
    elif setting == 4:
        mult = pop_ratio / city_area

    col_entry = np.zeros(9)
    temp_data = trunc_data[trunc_data['County'] == county_name]
    col_entry[0] = temp_data.shape[0]
    col_entry[1] = temp_data[temp_data['Age_Group'] == '0-9']
                  [temp_data[temp_data['Age_Group'] == '0-9']
                  ['Outcome'] == 'Deceased'].shape[0]
    col_entry[2] = temp_data[temp_data['Age_Group'] == '10-19']
              [temp_data[temp_data['Age_Group'] == '10-19']
```

```
            ['Outcome'] == 'Deceased'].shape[0]
    col_entry[3] = temp_data[temp_data['Age_Group'] == '20-29']
            [temp_data[temp_data['Age_Group'] == '20-29']
            ['Outcome'] == 'Deceased'].shape[0]
    col_entry[4] = temp_data[temp_data['Age_Group'] == '30-39']
            [temp_data[temp_data['Age_Group'] == '30-39']
            ['Outcome'] == 'Deceased'].shape[0]
    col_entry[5] = temp_data[temp_data['Age_Group'] == '40-49']
            [temp_data[temp_data['Age_Group'] == '40-49']
            ['Outcome'] == 'Deceased'].shape[0]
    col_entry[6] = temp_data[temp_data['Age_Group'] == '50-59']
            [temp_data[temp_data['Age_Group'] == '50-59']
            ['Outcome'] == 'Deceased'].shape[0]
    col_entry[7] = temp_data[temp_data['Age_Group'] == '60-69']
            [temp_data[temp_data['Age_Group'] == '60-69']
            ['Outcome'] == 'Deceased'].shape[0]
    col_entry[8] = temp_data[temp_data['Age_Group'] == '70-79']
            [temp_data[temp_data['Age_Group'] == '70-79']
            ['Outcome'] == 'Deceased'].shape[0] +
            temp_data[temp_data['Age_Group'] == '80-89']
            [temp_data[temp_data['Age_Group'] == '80-89']
            ['Outcome'] == 'Deceased'].shape[0] +
            temp_data[temp_data['Age_Group'] == '90-99']
            [temp_data[temp_data['Age_Group'] == '90-99']
            ['Outcome'] == 'Deceased'].shape[0] +
            temp_data[temp_data['Age_Group'] == '100']
            [temp_data[temp_data['Age_Group'] == '100']['Outcome']
            == 'Deceased'].shape[0]


    time_range_data_deaths[county_name] = col_entry * mult

time_range_data_cases = pd.DataFrame(index = ['Total Cases',
                        '0-9','10-19','20-29','30-39','40-49',
                        '50-59','60-69','70+'] )
top_counties = np.argsort(county_pop)[::-1][:n]
for h in range(n):

    county_name = county_list[top_counties[h]]

    city_area = ua_df[ua_df['County']
                == county_name]['Area'].iloc[0]
    city_pop = ua_df[ua_df['County']
                == county_name]['Populations'].iloc[0]
    rural_pop = combined_map[combined_map['NAME']
                == county_name.upper()]['COUNTYPOP'].iloc[0]
    pop_ratio = city_pop/rural_pop
    if setting2 == 0:
        mult = 1
    elif setting2 == 1:
        mult = 1/city_pop
    elif setting2 == 2:
        mult = 1/rural_pop
    elif setting2 == 3:
        mult = pop_ratio
    elif setting2 == 4:
        mult = pop_ratio / city_area

    col_entry = np.zeros(9)
    temp_data = trunc_data[trunc_data['County'] == county_name]
    col_entry[0] = temp_data.shape[0]
    col_entry[1] = temp_data[temp_data['Age_Group']
                == '0-9'].shape[0]
    col_entry[2] = temp_data[temp_data['Age_Group']
                == '10-19'].shape[0]
    col_entry[3] = temp_data[temp_data['Age_Group']
                == '20-29'].shape[0]
    col_entry[4] = temp_data[temp_data['Age_Group']
                == '30-39'].shape[0]
    col_entry[5] = temp_data[temp_data['Age_Group']
                == '40-49'].shape[0]
    col_entry[6] = temp_data[temp_data['Age_Group']
                == '50-59'].shape[0]
    col_entry[7] = temp_data[temp_data['Age_Group']
                == '60-69'].shape[0]
    col_entry[8] = temp_data[temp_data['Age_Group']
            == '70-79'].shape[0] + temp_data[temp_data['Age_Group']
            == '80-89'].shape[0] + temp_data[temp_data['Age_Group']
            == '90-99'].shape[0] + temp_data[temp_data['Age_Group']
            == '100'].shape[0]


    time_range_data_cases[county_name] = col_entry * mult

time_range_data_cases_percent = pd.DataFrame(index =
                ['Total Cases','0-9','10-19','20-29','30-39',
                '40-49','50-59','60-69','70+'] )
top_counties = np.argsort(county_pop)[::-1][:n]
for h in range(n):

    county_name = county_list[top_counties[h]]
    col_entry = np.zeros(9)
    temp_data = trunc_data[trunc_data['County'] == county_name]
    col_total = temp_data.shape[0]
    col_entry[0] = col_total
    col_entry[1] = temp_data[temp_data['Age_Group']
            == '0-9'].shape[0] / col_total
    col_entry[2] = temp_data[temp_data['Age_Group']
            == '10-19'].shape[0] / col_total
    col_entry[3] = temp_data[temp_data['Age_Group']
            == '20-29'].shape[0] / col_total
    col_entry[4] = temp_data[temp_data['Age_Group']
            == '30-39'].shape[0] / col_total
    col_entry[5] = temp_data[temp_data['Age_Group']
            == '40-49'].shape[0] / col_total
    col_entry[6] = temp_data[temp_data['Age_Group']
            == '50-59'].shape[0] / col_total
    col_entry[7] = temp_data[temp_data['Age_Group']
```

```
            == '60-69'].shape[0] / col_total
    col_entry[8] = (temp_data[temp_data['Age_Group']
            == '70-79'].shape[0] + temp_data[temp_data['Age_Group']
            == '80-89'].shape[0] + temp_data[temp_data['Age_Group']
            == '90-99'].shape[0] + temp_data[temp_data['Age_Group']
            == '100'].shape[0]) / col_total


    time_range_data_cases_percent[county_name] = col_entry


trdc = time_range_data_cases.copy().drop('Total Cases')
trdcp = time_range_data_cases_percent.copy().drop('Total Cases')
trdd = time_range_data_deaths.copy().drop('Total Cases')


ratios = trdd/ trdc

trdc.plot.bar(rot=0)
plt.xlabel('Ages')
plt.ylabel('Number of Cases')
trdcp.plot.bar(rot=0)
plt.xlabel('Ages')
plt.ylabel('Percent of Total Cases')
trdd.plot.bar(rot=0)
plt.xlabel('Ages')
plt.ylabel('Number of Deaths')
ratios.plot.bar(rot=0)
plt.xlabel('Ages')
plt.ylabel('Case Fatality Rate')
```

Omicron wave

```
start_date = '2021/12/24 12:00:00+00'
end_date = '2022/3/25 12:00:00+00'
setting = 0   # deaths
setting2 = 0  # cases
n = 6

start_index = full_data.Date_Reported.values.searchsorted(start_date,
                                               side = 'left')
end_index = full_data.Date_Reported.values.searchsorted(end_date,
                                               side = 'right')

trunc_data = full_data[start_index:end_index]


time_range_data_deaths = pd.DataFrame(index =
                ['Total Cases','0-9','10-19','20-29','30-39',
                '40-49','50-59','60-69','70+'] )
top_counties = np.argsort(county_pop)[::-1][:n]
for h in range(n):

    county_name = county_list[top_counties[h]]
```

```
city_area = ua_df[ua_df['County']
                == county_name]['Area'].iloc[0]
city_pop = ua_df[ua_df['County']
                == county_name]['Populations'].iloc[0]
rural_pop = combined_map[combined_map['NAME']
                == county_name.upper()]['COUNTYPOP'].iloc[0]
pop_ratio = city_pop/rural_pop
if setting == 0:
    mult = 1
elif setting == 1:
    mult = 1/city_pop
elif setting == 2:
    mult = 1/rural_pop
elif setting == 3:
    mult = pop_ratio
elif setting == 4:
    mult = pop_ratio / city_area

col_entry = np.zeros(9)
temp_data = trunc_data[trunc_data['County'] == county_name]
col_entry[0] = temp_data.shape[0]
col_entry[1] = temp_data[temp_data['Age_Group'] == '0-9']
            [temp_data[temp_data['Age_Group'] == '0-9']
            ['Outcome'] == 'Deceased'].shape[0]
col_entry[2] = temp_data[temp_data['Age_Group'] == '10-19']
            [temp_data[temp_data['Age_Group'] == '10-19']
            ['Outcome'] == 'Deceased'].shape[0]
col_entry[3] = temp_data[temp_data['Age_Group'] == '20-29']
            [temp_data[temp_data['Age_Group'] == '20-29']
            ['Outcome'] == 'Deceased'].shape[0]
col_entry[4] = temp_data[temp_data['Age_Group'] == '30-39']
            [temp_data[temp_data['Age_Group'] == '30-39']
            ['Outcome'] == 'Deceased'].shape[0]
col_entry[5] = temp_data[temp_data['Age_Group'] == '40-49']
            [temp_data[temp_data['Age_Group'] == '40-49']
            ['Outcome'] == 'Deceased'].shape[0]
col_entry[6] = temp_data[temp_data['Age_Group'] == '50-59']
            [temp_data[temp_data['Age_Group'] == '50-59']
            ['Outcome'] == 'Deceased'].shape[0]
col_entry[7] = temp_data[temp_data['Age_Group'] == '60-69']
            [temp_data[temp_data['Age_Group'] == '60-69']
            ['Outcome'] == 'Deceased'].shape[0]
col_entry[8] = temp_data[temp_data['Age_Group'] == '70-79']
            [temp_data[temp_data['Age_Group'] == '70-79']
            ['Outcome'] == 'Deceased'].shape[0] +
            temp_data[temp_data['Age_Group'] == '80-89']
            [temp_data[temp_data['Age_Group'] == '80-89']
            ['Outcome'] == 'Deceased'].shape[0] +
            temp_data[temp_data['Age_Group'] == '90-99']
            [temp_data[temp_data['Age_Group'] == '90-99']
            ['Outcome'] == 'Deceased'].shape[0] +
            temp_data[temp_data['Age_Group'] == '100']
            [temp_data[temp_data['Age_Group'] == '100']
```

```
                ['Outcome'] == 'Deceased'].shape[0]


    time_range_data_deaths[county_name] = col_entry * mult

time_range_data_cases = pd.DataFrame(index =
                    ['Total Cases','0-9','10-19','20-29',
                    '30-39','40-49','50-59','60-69','70+'] )
top_counties = np.argsort(county_pop)[::-1][:n]
for h in range(n):

    county_name = county_list[top_counties[h]]

    city_area = ua_df[ua_df['County']
                    == county_name]['Area'].iloc[0]
    city_pop = ua_df[ua_df['County']
                    == county_name]['Populations'].iloc[0]
    rural_pop = combined_map[combined_map['NAME']
                    == county_name.upper()]['COUNTYPOP'].iloc[0]
    pop_ratio = city_pop/rural_pop
    if setting2 == 0:
        mult = 1
    elif setting2 == 1:
        mult = 1/city_pop
    elif setting2 == 2:
        mult = 1/rural_pop
    elif setting2 == 3:
        mult = pop_ratio
    elif setting2 == 4:
        mult = pop_ratio / city_area

    col_entry = np.zeros(9)
    temp_data = trunc_data[trunc_data['County'] == county_name]
    col_entry[0] = temp_data.shape[0]
    col_entry[1] = temp_data[temp_data['Age_Group']
            == '0-9'].shape[0]
    col_entry[2] = temp_data[temp_data['Age_Group']
            == '10-19'].shape[0]
    col_entry[3] = temp_data[temp_data['Age_Group']
            == '20-29'].shape[0]
    col_entry[4] = temp_data[temp_data['Age_Group']
            == '30-39'].shape[0]
    col_entry[5] = temp_data[temp_data['Age_Group']
            == '40-49'].shape[0]
    col_entry[6] = temp_data[temp_data['Age_Group']
            == '50-59'].shape[0]
    col_entry[7] = temp_data[temp_data['Age_Group']
            == '60-69'].shape[0]
    col_entry[8] = temp_data[temp_data['Age_Group']
            == '70-79'].shape[0] + temp_data[temp_data['Age_Group']
            == '80-89'].shape[0] + temp_data[temp_data['Age_Group']
            == '90-99'].shape[0] + temp_data[temp_data['Age_Group']
            == '100'].shape[0]
    time_range_data_cases[county_name] = col_entry * mult

time_range_data_cases_percent = pd.DataFrame(index =
                ['Total Cases','0-9','10-19','20-29','30-39',
                '40-49','50-59','60-69','70+'] )
top_counties = np.argsort(county_pop)[::-1][:n]
for h in range(n):

    county_name = county_list[top_counties[h]]
    col_entry = np.zeros(9)
    temp_data = trunc_data[trunc_data['County'] == county_name]
    col_total = temp_data.shape[0]
    col_entry[0] = col_total
    col_entry[1] = temp_data[temp_data['Age_Group']
            == '0-9'].shape[0] / col_total
    col_entry[2] = temp_data[temp_data['Age_Group']
            == '10-19'].shape[0] / col_total
    col_entry[3] = temp_data[temp_data['Age_Group']
            == '20-29'].shape[0] / col_total
    col_entry[4] = temp_data[temp_data['Age_Group']
            == '30-39'].shape[0] / col_total
    col_entry[5] = temp_data[temp_data['Age_Group']
            == '40-49'].shape[0] / col_total
    col_entry[6] = temp_data[temp_data['Age_Group']
            == '50-59'].shape[0] / col_total
    col_entry[7] = temp_data[temp_data['Age_Group']
            == '60-69'].shape[0] / col_total
    col_entry[8] = (temp_data[temp_data['Age_Group']
            == '70-79'].shape[0] + temp_data[temp_data['Age_Group']
            == '80-89'].shape[0] + temp_data[temp_data['Age_Group']
            == '90-99'].shape[0] + temp_data[temp_data['Age_Group']
            == '100'].shape[0]) / col_total


    time_range_data_cases_percent[county_name] = col_entry


trdc = time_range_data_cases.copy().drop('Total Cases')
trdcp = time_range_data_cases_percent.copy().drop('Total Cases')
trdd = time_range_data_deaths.copy().drop('Total Cases')


ratios = trdd/ trdc

trdc.plot.bar(rot=0)
plt.xlabel('Ages')
plt.ylabel('Number of Cases')
trdcp.plot.bar(rot=0)
plt.xlabel('Ages')
plt.ylabel('Percent of Total Cases')
trdd.plot.bar(rot=0)
plt.xlabel('Ages')
plt.ylabel('Number of Deaths')
```

```
ratios.plot.bar(rot=0)
plt.xlabel('Ages')
plt.ylabel('Case Fatality Rate')
```

Comparison Visualization Code

This code is used three times to produce Figures 7.5, 7.9, and 7.12. To produce each one some adjustments to the y-axis scale were made, but other than that the codes used were all the same, so a single instance will be shown here.

```
top_counties = np.argsort(county_pop)[:: -1][:n]
fig, axes = plt.subplots(nrows=2, ncols=3)
for h in range(n):

    county_name = county_list[top_counties[h]]
    county_demos = pd.DataFrame(index = trdd.index,
                                columns = ['Census', 'Cases'])
    county_demos['Census'] = prop_df_new[county_name]
    county_demos['Cases'] = trdcp[county_name]
    county_demos.plot.bar(ax = axes[int(h/3), h%3],
                          legend = False,
                          figsize = (24,15),
                          title = county_name,
                          ylim = [0,0.25],
                          xlabel = 'Age',
                          ylabel = 'Percentage')

plt.legend()
```

Bibliography

[1] Report 9 - impact of non-pharmaceutical interventions (npis) to reduce covid-19 mortality and healthcare demand.

[2] IHME COVID-19 health service utilization forecasting team and Christopher JL Murray. Forecasting the impact of the first wave of the covid-19 pandemic on hospital demand and deaths for the usa and european economic area countries. *medRxiv*, 2020.

[3] Wouter Edeling, Hamid Arabnejad, Robbie Sinclair, Diana Suleimenova, Krishnakumar Gopalakrishnan, Bartosz Bosak, Derek Groen, Imran Mahmood, Daan Crommelin, and Peter V. Coveney. The impact of uncertainty on predictions of the covidsim epidemiological code. *Nature Computational Science*, 1(2):128–135, Feb 2021.

[4] W O Kermack and A G McKendrick. A contribution to the mathematical theory of epidemics. *Proc. R. Soc. Lond. A Math. Phys. Sci.*, 115(772):700–721, August 1927.

[5] L. Edelstein-Keshet. *Mathematical Models in Biology.* Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1987.

[6] J.D. Murray. *Mathematical Biology.* Biomathematics. Springer Berlin Heidelberg, 2013.

[7] H. E. Soper. The interpretation of periodicity in disease prevalence. *Journal of the Royal Statistical Society*, 92(1):34–61, 1929.

[8] Ronald Ross. An application of the theory of probabilities to the study of a priori pathometry. part i. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 92(638):204–230, 1916.

[9] Alfred J. Lotka. *Contribution to the Analysis of Malaria Epidemiology. I. General Part*, pages 302–347. Springer Berlin Heidelberg, Berlin, Heidelberg, 1978.

[10] Theodore Kolokolnikov and David Iron. Law of mass action and saturation in sir model with application to coronavirus modelling. *Infectious Disease Modelling*, 6:91–97, 2021.

[11] E B Wilson and J Worcester. The law of mass action in epidemiology. *Proc Natl Acad Sci U S A*, 31(1):24–34, January 1945.

[12] P. Érdi and J. Tóth. *Mathematical Models of Chemical Reactions: Theory and Applications of Deterministic and Stochastic Models.* Nonlinear science : theory and applications. Manchester University Press, 1989.

[13] Norske videnskaps-akademi i Oslo and O. Bastiansen. *The Law of Mass Action: A Centenary Volume, 1864-1964.* Universitetsforlaget, 1964.

[14] Shi Zhao, Qianyin Lin, Jinjun Ran, Salihu S Musa, Guangpu Yang, Weiming Wang, Yijun Lou, Daozhou Gao, Lin Yang, Daihai He, and Maggie H Wang. Preliminary estimation of the basic reproduction number of novel coronavirus (2019-nCoV) in china, from 2019 to 2020: A data-driven analysis in the early phase of the outbreak. *Int J Infect Dis*, 92:214–217, January 2020.

[15] Shi Zhao, Salihu S Musa, Qianying Lin, Jinjun Ran, Guangpu Yang, Weiming Wang, Yijun Lou, Lin Yang, Daozhou Gao, Daihai He, and Maggie H Wang. Estimating the unreported number of novel coronavirus (2019-nCoV) cases in china in the first half of january 2020: A Data-Driven modelling analysis of the early outbreak. *J Clin Med*, 9(2), February 2020.

[16] Igor Nesteruk. Statistics based predictions of coronavirus 2019-ncov spreading i n mainland china. *MedRxiv*, 4(1):1988–9, 2020.

[17] Wai-Kit Ming, Jian Huang, and Casper JP Zhang. Breaking down of healthcare system: Mathematical modelling for controlling the novel coronavirus (2019-ncov) outbreak in wuhan, china. *BioRxiv*, 2020.

[18] Narendra S Goel, Samaresh C Maitra, and Elliott W Montroll. On the volterra and other nonlinear models of interacting populations. *Reviews of modern physics*, 43(2):231, 1971.

[19] Ghaus ur Rahman, Kamal Shah, Fazal Haq, and Naveed Ahmad. Host vector dynamics of pine wilt disease model with convex incidence rate. *Chaos, Solitons & Fractals*, 113:31–39, 2018.

[20] Benjamin Wacker and Jan Schlüter. An age- and sex-structured SIR model: Theory and an explicit-implicit numerical solution algorithm. *Math Biosci Eng*, 17(5):5752–5801, August 2020.

[21] Gustavo A Muñoz-Fernández, Jesús M Seoane, and Juan B Seoane-Sepúlveda. A SIR-type model describing the successive waves of COVID-19. *Chaos Solitons Fractals*, 144:110682, January 2021.

[22] Hadeel AlQadi and Majid Bani-Yaghoub. Incorporating global dynamics to improve the accuracy of disease models: Example of a COVID-19 SIR model. *PLoS One*, 17(4):e0265815, April 2022.

[23] Sheryl Le Chang, Mahendra Piraveenan, and Mikhail Prokopenko. The effects of imitation dynamics on vaccination behaviours in SIR-Network model. *Int J Environ Res Public Health*, 16(14), July 2019.

[24] E Canouï and O Launay. [history and principles of vaccination]. *Rev Mal Respir*, 36(1):74–81, December 2018.

[25] Stanley A Plotkin. Vaccines: past, present and future. *Nat Med*, 11(4 Suppl):S5–11, April 2005.

[26] Ananda S Bandyopadhyay, Julie Garon, Katherine Seib, and Walter A Orenstein. Polio vaccination: past, present and future. *Future Microbiol*, 10(5):791–808, March 2015.

[27] David Y Gaitonde, Faith C Moore, and Mackenzie K Morgan. Influenza: Diagnosis and treatment. *Am Fam Physician*, 100(12):751–758, December 2019.

[28] Linda J Keilman. Seasonal influenza (flu). *Nurs Clin North Am*, 54(2):227–243, June 2019.

[29] Robert G Webster and Elena A Govorkova. Continuing challenges in influenza. *Ann N Y Acad Sci*, 1323(1):115–139, May 2014.

[30] Zhiru Gao, Yinghui Xu, Chao Sun, Xu Wang, Ye Guo, Shi Qiu, and Kewei Ma. A systematic review of asymptomatic infections with COVID-19. *J Microbiol Immunol Infect*, 54(1):12–16, May 2020.

[31] Anshika Sharma, Isra Ahmad Farouk, and Sunil Kumar Lal. COVID-19: A review on the novel coronavirus disease evolution, transmission, detection, control and prevention. *Viruses*, 13(2), January 2021.

[32] Lee W Riley. Differentiating epidemic from endemic or sporadic infectious disease occurrence. *Microbiol Spectr*, 7(4), July 2019.

[33] Graham F Medley and Anna Vassall. When an emerging disease becomes endemic. *Science*, 357(6347):156–158, July 2017.

[34] Philip Minor. The polio endgame. *Hum Vaccin Immunother*, 10(7):2106–2108, 2014.

[35] Jun Inamo. How should we overcome the threat by the pandemic of 2019-nCoV? epidemic simulation using the SIRS model. *Clin Rheumatol*, 39(8):2471–2473, May 2020.

[36] Ilaria Renna. Homogenous mixing and network approximations in discrete-time formulation of a SIRS model. *J Biol Dyn*, 15(1):635–651, December 2021.

[37] Till D. Frank. *COVID-19 epidemiology and virus dynamics—nonlinear physics and mathematical modeling.* Understanding Complex Systems. Springer, Cham, [2022] ©2022.

[38] Huaixing Li and Jiaoyan Wang. Global dynamics of an seir model with the age of infection and vaccination. *Mathematics*, 9(18):2195, 2021.

[39] Samuel Mwalili, Mark Kimathi, Viona Ojiambo, Duncan Gathungu, and Rachel Mbogo. SEIR model for COVID-19 dynamics incorporating the environment and social distancing. *BMC Res Notes*, 13(1):352, July 2020.

[40] Hee-Young Shin. A multi-stage SEIR(D) model of the COVID-19 epidemic in korea. *Ann Med*, 53(1):1159–1169, December 2021.

[41] Yanjin Wang, Pei Wang, Shudao Zhang, and Hao Pan. Uncertainty modeling of a modified SEIR epidemic model for COVID-19. *Biology (Basel)*, 11(8), August 2022.

[42] Zifeng Yang, Zhiqi Zeng, Ke Wang, Sook-San Wong, Wenhua Liang, Mark Zanin, Peng Liu, Xudong Cao, Zhongqiang Gao, Zhitong Mai, et al. Modified seir and ai prediction of the epidemics trend of covid-19 in china under public health interventions. *Journal of thoracic disease*, 12(3):165, 2020.

[43] Debashis Saikia, Kalpana Bora, and Madhurjya P Bora. COVID-19 outbreak in india: an SEIR model-based analysis. *Nonlinear Dyn*, 104(4):4727–4751, June 2021.

[44] Delia Ortega-Lenis, David Arango-Londoño, Edgar Muñoz, Daniel E Cuartas, Diana Caicedo, Jorge Mena, Miyerlandi Torres, and Fabian Mendez. [predictions of a SEIR model for COVID-19 cases in Cali-Colombia]. *Rev Salud Publica (Bogota)*, 22(2):132–137, March 2020.

[45] Nicola Picchiotti, Monica Salvioli, Elena Zanardini, and Francesco Missale. COVID-19 pandemic: a mobility-dependent SEIR model with undetected cases in italy, europe, and US. *Epidemiol Prev*, 44(5-6 Suppl 2):136–143, September 2020.

[46] Gavin M Abernethy and David H Glass. Optimal COVID-19 lockdown strategies in an age-structured SEIR model of northern ireland. *J R Soc Interface*, 19(188):20210896, March 2022.

[47] Suwardi Annas, Muh Isbar Pratama, Muh Rifandi, Wahidah Sanusi, and Syafruddin Side. Stability analysis and numerical simulation of SEIR model for pandemic COVID-19 spread in indonesia. *Chaos Solitons Fractals*, 139:110072, July 2020.

[48] Mubashara Wali, Sadia Arshad, and Jianfei Huang. Stability analysis of an extended SEIR COVID-19 fractional model with vaccination efficiency. *Comput Math Methods Med*, 2022:3754051, September 2022.

[49] Giuseppe C Calafiore, Carlo Novara, and Corrado Possieri. A time-varying sird model for the covid-19 contagion in italy. *Annual reviews in control*, 50:361–372, 2020.

[50] Norman TJ Bailey et al. *The mathematical theory of infectious diseases and its applications*. Charles Griffin & Company Ltd, 5a Crendon Street, High Wycombe, Bucks HP13 6LE., 1975.

[51] Amir Arslan Haghrah, Sehraneh Ghaemi, and Mohammad Ali Badamchizadeh. Fuzzy-SIRD model: Forecasting COVID-19 death tolls considering governments intervention. *Artif Intell Med*, 134:102422, October 2022.

[52] Abdennour Sebbvagh and Sihem Kechida. EKF-SIRD model algorithm for predicting the coronavirus (COVID-19) spreading dynamics. *Sci Rep*, 12(1):13415, August 2022.

[53] Sakshi Shringi, Harish Sharma, Pushpa Narayan Rathie, Jagdish Chand Bansal, and Atulya Nagar. Modified SIRD model for COVID-19 spread prediction for northern and southern states of india. *Chaos Solitons Fractals*, 148:111039, May 2021.

[54] Gustavo M Athayde and Airlane P Alencar. Forecasting covid-19 in the united kingdom: A dynamic SIRD model. *PLoS One*, 17(8):e0271577, August 2022.

[55] Jin-Xuan Yang. A SIRD epidemic model with community structure. *Chaos*, 31(1):013102, January 2021.

[56] Hakimeh Mohammadi, Shahram Rezapour, and Amin Jajarmi. On the fractional SIRD mathematical model and control for the transmission of COVID-19: The first and the second waves of the disease in iran and japan. *ISA Trans*, 124:103–114, April 2021.

[57] Jesús Fernández-Villaverde and Charles I Jones. Estimating and simulating a SIRD model of COVID-19 for many countries, states, and cities. *J Econ Dyn Control*, 140:104318, January 2022.

[58] Luisa Ferrari, Giuseppe Gerardi, Giancarlo Manzi, Alessandra Micheletti, Federica Nicolussi, Elia Biganzoli, and Silvia Salini. Modeling provincial covid-19 epidemic data using an adjusted Time-Dependent SIRD model. *Int J Environ Res Public Health*, 18(12), June 2021.

[59] Saptarshi Chatterjee, Apurba Sarkar, Swarnajit Chatterjee, Mintu Karmakar, and Raja Paul. Studying the progress of COVID-19 outbreak in india using SIRD model. *Indian J Phys Proc Indian Assoc Cultiv Sci (2004)*, 95(9):1941–1957, June 2020.

[60] Elvia Karina Grillo Ardila, Julián Santaella-Tenorio, Rodrigo Guerrero, and Luis Eduardo Bravo. Mathematical model and COVID-19. *Colomb Med (Cali)*, 51(2):e4277, June 2020.

[61] Mohamed Lounis and Dilip Kumar Bagal. Estimation of SIR model's parameters of COVID-19 in algeria. *Bull Natl Res Cent*, 44(1):180, October 2020.

[62] Muhammad Taimoor, Sajid Ali, Ismail Shah, and Fred Roland Muwanika. COVID-19 pandemic data modeling in pakistan using Time-Series SIR. *Comput Math Methods Med*, 2022:6001876, June 2022.

[63] Rahim Ud Din and Ebrahem A Algehyne. Mathematical analysis of COVID-19 by using SIR model with convex incidence rate. *Results Phys*, 23:103970, February 2021.

[64] Ronald Eleazar Huarachi Olivera and Antonio Mateo Lazarte RIvera. [SIR model of the pandemic trend of COVID-19 in peru]. *Rev Fac Cien Med Univ Nac Cordoba*, 78(3):236–242, August 2021.

[65] Fred G Manrique-Abril, Carlos A Agudelo-Calderon, Víctor M González-Chordá, Oscar Gutiérrez-Lesmes, Cristian F Téllez-Piñerez, and Giomar Herrera-Amaya. [SIR model of the COVID-19 pandemic in colombia]. *Rev Salud Publica (Bogota)*, 22(2):123–131, March 2020.

[66] Bernd Salzberger, Felix Buder, Benedikt Lampl, Boris Ehrenstein, Florian Hitzenbichler, Thomas Holzmann, Barbara Schmidt, and Frank Hanses. Epidemiology of SARS-CoV-2. *Infection*, 49(2):233–239, October 2020.

[67] B Malavika, S Marimuthu, Melvin Joy, Ambily Nadaraj, Edwin Sam Asirvatham, and L Jeyaseelan. Forecasting COVID-19 epidemic in india and high incidence states using SIR and logistic growth models. *Clin Epidemiol Glob Health*, 9:26–33, June 2020.

[68] U. Wilensky and W. Rand. *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo*. The MIT Press. MIT Press, 2015.

[69] S.F. Railsback and V. Grimm. *Agent-Based and Individual-Based Modeling: A Practical Introduction, Second Edition*. Princeton University Press, 2019.

[70] Hilla De-Leon and Dvir Aran. MAM: Flexible Monte-Carlo agent based model for modelling COVID-19 spread. *J Biomed Inform*, 141:104364, April 2023.

[71] Erik Cuevas. An agent-based model to evaluate the COVID-19 transmission risks in facilities. *Comput Biol Med*, 121:103827, May 2020.

[72] Tao Dong, Wen Dong, and Quanli Xu. Agent simulation model of COVID-19 epidemic Agent-Based on GIS: A case study of huangpu district, shanghai. *Int J Environ Res Public Health*, 19(16), August 2022.

[73] Xiaolong Geng, Gabriel G Katul, Firas Gerges, Elie Bou-Zeid, Hani Nassif, and Michel C Boufadel. A kernel-modulated SIR model for covid-19 contagious spread from county to continent. *Proc Natl Acad Sci U S A*, 118(21), May 2021.

[74] Tri K Nguyen, Nam H Hoang, Graham Currie, and Hai L Vu. Enhancing covid-19 virus spread modeling using an activity travel model. *Transp Res Part A Policy Pract*, 161:186–199, May 2022.

[75] L L Lima and A P F Atman. Impact of mobility restriction in COVID-19 superspreading events using agent-based model. *PLoS One*, 16(3):e0248708, March 2021.

[76] Chathika Gunaratne, Rene Reyes, Erik Hemberg, and Una-May O'Reilly. Evaluating efficacy of indoor non-pharmaceutical interventions against COVID-19 outbreaks with a coupled spatial-SIR agent-based simulation framework. *Sci Rep*, 12(1):6202, April 2022.

[77] David Adam. Special report: The simulations driving the world's response to COVID-19. *Nature*, 580(7803):316–318, April 2020.

[78] Nick Wilson, Michael G Baker, Tony Blakely, and Martin Eichner. Estimating the impact of control measures to prevent outbreaks of COVID-19 associated with air travel into a COVID-19-free country. *Sci Rep*, 11(1):10766, May 2021.

[79] Ken Rice, Ben Wynne, Victoria Martin, and Graeme J Ackland. Effect of school closures on mortality from coronavirus disease 2019: old and new predictions. *BMJ*, 371:m3588, October 2020.

[80] Nessma Adil Mahmoud Yousif, Henri Christian Junior Tsoungui Obama, Yvan Jordan Ngucho Mbeutchou, Sandy Frank Kwamou Ngaha, Loyce Kayanula, George Kamanga, Toheeb Babatunde Ibrahim, Patience Bwanu Iliya, Sulyman Iyanda, Looli Alawam Nemer, Kristina Barbara Helle, Miranda Ijang Teboh-Ewungkem, and Kristan Alexander Schneider. The impact of COVID-19 vaccination campaigns accounting for antibody-dependent enhancement. *PLoS One*, 16(4):e0245417, April 2021.

[81] Kristan A Schneider, Gideon A Ngwa, Markus Schwehm, Linda Eichner, and Martin Eichner. The COVID-19 pandemic preparedness simulation tool: CovidSIM. *BMC Infect Dis*, 20(1):859, November 2020.

[82] Sue Denim. Code review of ferguson's model, https://dailysceptic.org/archive/code-review-of-fergusons-model, Jun 2022.

[83] Robert Hinch, William J M Probert, Anel Nurtay, Michelle Kendall, Chris Wymant, Matthew Hall, Katrina Lythgoe, Ana Bulas Cruz, Lele Zhao, Andrea Stewart, Luca Ferretti, Daniel Montero, James Warren, Nicole Mather, Matthew Abueg, Neo Wu, Olivier Legat, Katie Bentley, Thomas Mead, Kelvin Van-Vuuren, Dylan Feldner-Busztin, Tommaso Ristori, Anthony Finkelstein, David G Bonsall, Lucie Abeler-Dörner, and Christophe Fraser. OpenABM-Covid19-An agent-based model for non-

pharmaceutical interventions against COVID-19 including contact tracing. *PLoS Comput Biol*, 17(7):e1009146, July 2021.

[84] Cliff C Kerr, Robyn M Stuart, Dina Mistry, Romesh G Abeysuriya, Katherine Rosenfeld, Gregory R Hart, Rafael C Núñez, Jamie A Cohen, Prashanth Selvaraj, Brittany Hagedorn, Lauren George, Michał Jastrzebski, Amanda S Izzo, Greer Fowler, Anna Palmer, Dominic Delport, Nick Scott, Sherrie L Kelly, Caroline S Bennette, Bradley G Wagner, Stewart T Chang, Assaf P Oron, Edward A Wenger, Jasmina Panovska-Griffiths, Michael Famulare, and Daniel J Klein. Covasim: An agent-based model of COVID-19 dynamics and interventions. *PLoS Comput Biol*, 17(7):e1009149, July 2021.

[85] Eric Winsberg, Jason Brennan, and Chris W. Surprenant. How government leaders violated their epistemic duties during the sars-cov-2 crisis. *Kennedy Institute of Ethics Journal*, 30(3):215–242, 2020.

[86] P.S. Onge and G. Campan. *The Flawed Covid-19 Model That Locked Down Canada.* Health Policy Series. Montreal Economic Institute, 2020.

[87] Vishaal Ram and Laura P Schaposnik. A modified age-structured SIR model for COVID-19 type viruses. *Sci Rep*, 11(1):15194, July 2021.

[88] Qun Li, Xuhua Guan, Peng Wu, Xiaoye Wang, Lei Zhou, Yeqing Tong, Ruiqi Ren, Kathy S M Leung, Eric H Y Lau, Jessica Y Wong, Xuesen Xing, Nijuan Xiang, Yang Wu, Chao Li, Qi Chen, Dan Li, Tian Liu, Jing Zhao, Man Liu, Wenxiao Tu, Chuding Chen, Lianmei Jin, Rui Yang, Qi Wang, Suhua Zhou, Rui Wang, Hui Liu, Yinbo Luo, Yuan Liu, Ge Shao, Huan Li, Zhongfa Tao, Yang Yang, Zhiqiang Deng, Boxi Liu, Zhitao Ma, Yanping Zhang, Guoqing Shi, Tommy T Y Lam, Joseph T Wu, George F Gao, Benjamin J Cowling, Bo Yang, Gabriel M Leung, and Zijian Feng. Early transmission dynamics in wuhan, china, of novel Coronavirus-Infected pneumonia. *N Engl J Med*, 382(13):1199–1207, January 2020.

[89] Pranati Rakshit, Soumen Kumar, Samad Noeiaghdam, Unai Fernandez-Gamiz, Mohamed Altanji, and Shyam Sundar Santra. Modified SIR model for COVID-19 transmission dynamics: Simulation with case study of UK, US and india. *Results Phys*, 40:105855, July 2022.

[90] Joseph T Wu, Kathy Leung, and Gabriel M Leung. Nowcasting and forecasting the potential domestic and international spread of the 2019-nCoV outbreak originating in wuhan, china: a modelling study. *Lancet*, 395(10225):689–697, January 2020.

[91] Muhammad Taimoor, Sajid Ali, Ismail Shah, and Fred Roland Muwanika. COVID-19 pandemic data modeling in pakistan using Time-Series SIR. *Comput Math Methods Med*, 2022:6001876, June 2022.

[92] Mingwang Shen, Zhihang Peng, Yanni Xiao, and Lei Zhang. Modeling the epidemic trend of the 2019 novel coronavirus outbreak in china. *Innovation (Camb)*, 1(3):100048, September 2020.

[93] Tchavdar T Marinov and Rossitza S Marinova. Adaptive SIR model with vaccination: simultaneous identification of rates and functions illustrated with COVID-19. *Sci Rep*, 12(1):15688, September 2022.

[94] Kian Boon Law, Kalaiarasu M Peariasamy, Balvinder Singh Gill, Sarbhan Singh, Bala Murali Sundram, Kamesh Rajendran, Sarat Chandra Dass, Yi Lin Lee, Pik Pin Goh, Hishamshah Ibrahim, and Noor Hisham Abdullah. Tracking the early depleting transmission dynamics of COVID-19 with a time-varying SIR model. *Sci Rep*, 10(1):21721, December 2020.

[95] Ian Cooper, Argha Mondal, and Chris G Antonopoulos. A SIR model assumption for the spread of COVID-19 in different communities. *Chaos Solitons Fractals*, 139:110057, June 2020.

[96] Rishal Amar Singh, Rajnesh Lal, and Ramanuja Rao Kotti. Time-discrete SIR model for COVID-19 in fiji. *Epidemiol Infect*, 150:1–17, April 2022.

[97] Usafacts coronavirus tracker, May 2023.

[98] MTN News. Missoula county covid-19 update: 46 new, 256 active cases, Aug 2022.

[99] Vaughn Cooper. Massive numbers of new covid-19 infections, not vaccines, are the main driver of new coronavirus variants, Sep 2021.

[100] Louis Lucero II. Covid updates: New york reports 85,000 cases, a new daily record, 2022.

[101] Anonymous. Influenza in a boarding school. *British Medical Journal*, page 587, 1978.

[102] Nadia Yusuf and Lamia Saud Shesha. Economic role of population density during Pandemics-A comparative analysis of saudi arabia and china. *Int J Environ Res Public Health*, 18(8), April 2021.

[103] Andreas Eilersen and Kim Sneppen. SARS-CoV-2 superspreading in cities vs the countryside. *APMIS*, 129(7):401–407, February 2021.

[104] Mary Ann Branch, Thomas F. Coleman, and Yuying Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999.

[105] Nitya Kumar, Suha Quadri, Abdulla Ismaeel AlAwadhi, and Manaf AlQahtani. COVID-19 recovery patterns across alpha (b.1.1.7) and delta (b.1.617.2) variants of SARS-CoV-2. *Front Immunol*, 13:812606, February 2022.

[106] G. Strang. *Linear Algebra and Its Applications.* Thomson, Brooks/Cole, 2006.

[107] Elizabeth T. Chin, David Leidner, Lauren Lamson, Kimberley Lucas, David M. Studdert, Jeremy D. Goldhaber-Fiebert, Jason R. Andrews, and Joshua A. Salomon. Pro-

tection against omicron from vaccination and previous infection in a prison system. *New England Journal of Medicine*, 387(19):1770–1782, 2022.

[108] Dinah V Parums. Editorial: First approval of the Protein-Based adjuvanted nuvaxovid (NVX-CoV2373) novavax vaccine for SARS-CoV-2 could increase vaccine uptake and provide immune protection from viral variants. *Med Sci Monit*, 28:e936523, March 2022.

[109] Office of the Commissioner. Coronavirus (covid-19) update: Fda authorizes emergency use of novavax covid-19 vaccine, adjuvanted.

[110] Australian Government Department of Health and Aged Care. Is it true? how long does it take for the vaccines to work?, May 2023.

[111] Cdc covid data tracker, https://covid.cdc.gov/covid-data-tracker/variant-proportions.

[112] Nick Andrews, Julia Stowe, Freja Kirsebom, Samuel Toffa, Tim Rickeard, Eileen Gallagher, Charlotte Gower, Meaghan Kall, Natalie Groves, Anne-Marie O'Connell, David Simons, Paula B Blomquist, Asad Zaidi, Sophie Nash, Nurin Iwani Binti Abdul Aziz, Simon Thelwall, Gavin Dabrera, Richard Myers, Gayatri Amirthalingam, Saheer Gharbia, Jeffrey C Barrett, Richard Elson, Shamez N Ladhani, Neil Ferguson, Maria Zambon, Colin N J Campbell, Kevin Brown, Susan Hopkins, Meera Chand, Mary Ramsay, and Jamie Lopez Bernal. Covid-19 vaccine effectiveness against the omicron (b.1.1.529) variant. *N Engl J Med*, 386(16):1532–1546, March 2022.

[113] Carolina Ribeiro Xavier, Rafael Sachetto Oliveira, Vinícius da Fonseca Vieira, Marcelo Lobosco, and Rodrigo Weber Dos Santos. Characterisation of omicron variant during COVID-19 pandemic and the impact of vaccination, transmission rate, mortality, and reinfection in south africa, germany, and brazil. *BioTech (Basel)*, 11(2), April 2022.

[114] Nathalie Schwab, Ronny Nienhold, Maurice Henkel, Albert Baschong, Anne Graber, Angela Frank, Nadine Mensah, Jacqueline Koike, Claudia Hernach, Melanie Sachs,

Till Daun, Veronika Zsikla, Niels Willi, Tobias Junt, and Kirsten D Mertz. COVID-19 autopsies reveal underreporting of SARS-CoV-2 infection and scarcity of co-infections. *Front Med (Lausanne)*, 9:868954, April 2022.

[115] Alex. The most visited states in the united states, https://vividmaps.com/most-visited-us-states/, Oct 2021.

[116] Qian Wang, Tiantian Zhang, Huanhuan Zhu, Ying Wang, Xin Liu, Ge Bai, Ruiming Dai, Ping Zhou, and Li Luo. Characteristics of and public health emergency responses to COVID-19 and H1N1 outbreaks: A Case-Comparison study. *Int J Environ Res Public Health*, 17(12), June 2020.

[117] 2009 h1n1 pandemic (h1n1pdm09 virus), https://www.cdc.gov/flu/pandemic-resources/2009-h1n1-pandemic.html, Jun 2019.

[118] Who coronavirus (covid-19) dashboard, https://covid19.who.int/.