THE PATH TO MEASLES ELIMINATION IN THE UNITED STATES

THE PATH TO MEASLES ELIMINATION IN THE UNITED STATES

By ELIZABETH O'MEARA, H.B.Sc.

A Thesis Submitted to the School of Graduate Studies in Partial Fulfilment of the Requirements for

the Degree Master of Science

MASTER OF SCIENCE (2022) McMaster University

Hamilton, ON (Mathematics)

TITLE: The Path to Measles Elimination in the United States

AUTHOR: Elizabeth O'Meara, H.B.Sc. (McMaster University)

SUPERVISOR: Dr. David Earn

NUMBER OF PAGES: xiii, 321

Lay Abstract

This thesis demonstrates the use of a “canonical path”, as defined by Graham *et al.* [1], as a method of visually tracking disease elimination at a country level, focusing on measles elimination in the United States. Additionally, we determine what model structures are required to define a theoretical path using data on disease characteristics (*e.g.* vaccination coverage and natural history of infection), while demonstrating how the path changes when disease characteristics change. Our analysis suggests that using this method for other countries or eradicable diseases could provide useful insight of the successes and failures of elimination strategies. This has the potential to improve elimination and/or eradication programs in the future.

Abstract

The eradication of infectious diseases has been of key interest for many years. While the World Health Organization (WHO) and other health organizations typically track the progress of disease eradication based on whether regions are meeting their eradication targets, being able to quantify and/or visually track the eradication of a disease could prove beneficial. This thesis creates the "canonical path" to the elimination of measles in the United States (US), using similar methods as defined by Graham *et al.* [1]. We build on preliminary work conducted to fulfill the requirements of an Honours Bachelor of Science in Integrated Science at McMaster University, and the analysis conducted by Graham *et al.* [1], through the investigation of the sensitivity of the path to changes in its definition, as well how the path changes when we change the characteristics of the disease. This thesis demonstrates the ability to use a canonical path on a smaller, country-level scale, by using United States (US) state level data to create the US canonical path. We also determine the model structures necessary to simulate the canonical path, which suggests that the canonical path method is most useful for eradicable diseases for which we have ample knowledge of the disease, including the natural history of infection and vaccination. We also predict how the path is affected by the pattern of seasonality and by the natural history of infection. Overall, the analysis suggests that the more this method is implemented for other countries that have eliminated measles or for other diseases for which we have achieved elimination, we may gain insight of the successes and failures of elimination strategies. This knowledge could help the WHO and other organizations improve their disease elimination and eradication strategies in the future.

Acknowledgements

I wish to share my appreciation to my supervisor, Dr. David Earn, who provided guidance and support throughout the process of my thesis, even in light of the COVID-19 pandemic and while on sabbatical in British Columbia.

I would also like to thank my examination committee members Dr. Jonathan Dushoff and Dr. Benjamin Bolker for their comments and suggestions on my thesis.

Finally, I would like to thank the numerous members of the Mac-Theobio lab for providing their continued support. Special thanks to Mikael Jagan as his extensive knowledge and experience with `knitr` helped me put together a supplement that ensures reproducibility of my results.

Contents

List of Figures

List of Tables

Acronyms

2D	2-dimension
CDC	Centers for Disease Control and Prevention
CV	Coefficient of Variation
FOIA	Freedom of Information Act
IQR	Interquartile Range
MMR	Measles-Mumps-Rubella
MMWR	Morbidity and Mortality Weekly Report
PAHO	Pan American Health Organization
SEIR	Susceptible-Exposed-Infectious-Recovered
US	United States
WHO	World Health Organization

1 Introduction

With the emergence of the new pathogen SARS-CoV-2, and a 2 year long pandemic, many people want to know: when will COVID-19 be gone? Ridding the world of an infectious disease is known as eradication [2,3]. To date, the World Health Organization (WHO) has only declared two diseases eradicated: smallpox and rinderpest [3]. For diseases that we wish to eradicate, but have not yet achieved eradication, the WHO, and other organizations like the Pan American Health Organization (PAHO), focus on disease elimination. The key difference between these two goals is that eradication requires a global reduction of cases to 0, while the WHO defines (measles) elimination as "the absence of endemic[1] measles transmission in a defined geographical area (e.g. region or country) for at least 12 months in the presence of a surveillance system that has been verified to be performing well." [5] Measles is a perfect example of this; it has been considered eliminated in the United States as of 2000, however it is still considered endemic in several developing regions of Africa and Asia [6,7].

Graham *et al.* [1] presented a novel method of tracking the elimination measles, through the use of a "canonical path". In their paper, they use country-level data to define a global canonical path to the elimination of measles. The goal of this thesis is to evaluate the use of the canonical path to track the elimination of measles and other eradicable diseases within a single country. For our purposes we will focus on the United States, due to the accessibility of data. The canonical path will be described in detail in section 1.3.

1.1 Identifying Eradicable Diseases

As we are interested in tracking the elimination and/or eradication of infectious diseases, understanding what makes a disease eradicable can help us determine for which diseases a canonical path could be implemented. Currently, the WHO and the PAHO have identified 30+ diseases that are candidates for elimination or maintaining elimination status by 2030, including measles [8]. Although there are considerable economic and political constraints that affect whether disease eradication is feasible [9], a recent article [3] identified 4 key biological aspects that help determine if a disease is eradicable:

1. Is the disease easy to diagnose or recognize?
2. Is there a non-human reservoir and/or vector?

[1] "Endemic refers to the constant presence and/or usual prevalence of a disease or infectious agent in a population within a geographic area." [4]

3. Is the disease geographically restricted?
4. Is there a vaccine or other transmission-disrupting alternatives? [3]

During the smallpox eradication campaign, the specificity of smallpox symptoms made it very easy to diagnose and trace within communities [3]. Diseases for which diagnosis is more difficult, whether due to the nature of symptoms or the sophistication of testing requirements, make it much harder to track the disease epidemiology, making elimination more difficult. Additionally, if a disease can only infect humans, eradication becomes simpler as once eradicated within humans, it cannot persist in another population or reservoir. Furthermore, when we approach eradication, diseases that once affected the global population become increasingly geographically restricted [3]. While the geographical restriction of a disease can help the WHO and other organizations target regions still affected by the disease, regions that have achieved eradication will get used to the absence of the disease and may not longer see the benefit that eradication campaigns have. This could cause such regions to reduce their support to the global campaign, in turn negatively affecting other regions' eradication targets [3].

Finally, there must be some sort of transmission-disrupting mechanism in place. For smallpox and rinderpest, the transmission-disrupting mechanism used was a vaccine. However, the eradication campaign for Dracunculiasis (Guinea Worm Disease) has been successful in reducing case numbers without the use of a vaccine. Reduced prevalence was achieved by The Carter Center, in collaboration with UNICEF, through the distribution of water filtration systems to provide safe drinking water to affected communities. The campaign resulted in a reduction in incidence from approximately 3.5 million in 1986, to only 53 cases in 2019 [3, 10]. Thus, diseases do not necessarily require a vaccine to be eradicated, but they must have some efficient form of transmission-disrupting mechanism to be a candidate for eradication.

1.2 Measles Elimination Strategies

Once a disease has been deemed eradicable, the WHO and other health organizations create global strategies for the elimination and/or the eradication of the disease. Since shortly after the vaccine was invented in 1963, measles eradication has been of key interest to the WHO and the PAHO [11, 12]. In 2021, the WHO provided an updated program with the goal of measles and rubella elimination for the years 2021 to 2030 [13]. In this program, they outline the achievements made over the duration of the previous program (2012 to 2020). Over this period, 178 WHO member states had introduced a second dose of the measles-containing vaccine and global coverage had reached 71% in 2019. Additionally, in 2018, through 45 supplementary immunization activities across 37 countries, approximately 346 million individuals were vaccinated for measles. Furthermore, as of 2018, 82 countries have eliminated measles. However, in spite of the progress that was made, "several contextual changes and implementation challenges impeded progress and contributed to an increasing number of outbreaks and a resurgence in measles cases." [13] To improve measles elimination progress, the program has outlined several changes to the 2012 to 2020 plan. The changes include, but are not limited to, strengthening capacity for outbreak preparedness and response, strengthening cross-border monitoring, information sharing and collaboration, and improving surveillance.

1.3 A Novel Method of Tracking the Progress Made Towards Measles Elimination

While the WHO strategic plans outline specific goals for a given time-frame, tracking the progress made towards elimination is largely based on the world's progress through the current plan. For example, how many regions are meeting vaccination targets? Have we managed to improve our surveillance technologies? It would be helpful to be able to quantify and/or visualize our progress towards elimination based on the strategies in place, which is exactly what Graham *et al.* [1] accomplished. In their paper, they introduced a novel method that investigates the global path to the elimination of measles. They define a "canonical path", which each WHO region follows while progressing towards measles elimination [1].

The canonical path is defined as a path through **incidence space**, in which weighted mean incidence per 100,000 individuals is plotted against smoothed unweighted Coefficient of Variation (CV) of incidence. Graham *et al.* [1] use yearly measles incidence and population for each WHO region from 1980 to 2014, by country. They create the canonical path by finding the weighted mean incidence and CV for each country, and then find the mean point in incidence space in each year for both the WHO African and Americas regions. The mean position for each region is then plotted on the same figure. Since the African and Americas regions are at different stages in elimination progress, they join the paths at their intersection to define the global path that all countries follow while progressing towards measles elimination. The resulting path is shown in figure 1.

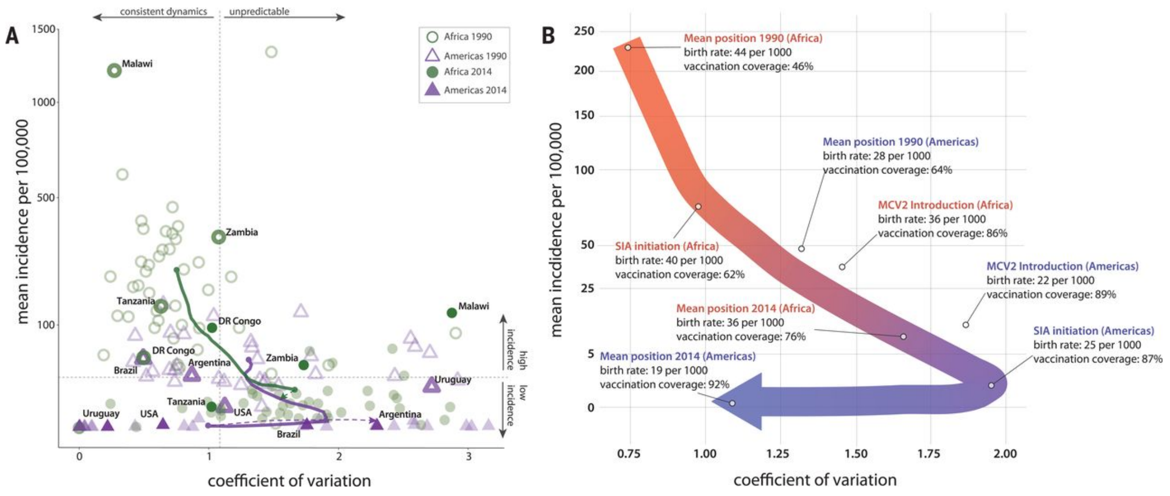


Figure 1: Figure reprinted from Graham *et al.* [1], representing the global canonical path to measles elimination. A: An incidence plane with the mean incidence per 100,000 for Africa (green) and the Americas (purple) on the y-axis and the CV on the x-axis. B: A more coherent version of the canonical path outlining key points in measles history [1].

With the global canonical path created, Graham *et al.* [1] conduct analysis on a country's progression along the path as well as speed of progression. In addition, they define a model that is capable of simulating country-level measles incidence, in order to determine how targeting different age-groups with supplemental immunization activities can change a country's progression along the path. They show that measuring a country's progression in this way can provide important insight on future dynamics, such as how measles dynamics change as we approach elimination, as well as each country's overall progress. They argue that as a public health tool, the canonical path analysis has the potential to help countries optimize their programs by evaluating each program's success based on changes in the canonical path. This has the potential to speed up the progression towards measles elimination worldwide.

1.4 Preliminary Work on the US Canonical Path

In a previous paper written to fulfill the requirements of an Honours Bachelor of Science in Integrated Science at McMaster University, preliminary analysis was conducted that demonstrated the ability to create a canonical path for the United States using state-level Incidence and Population data [14]. Unlike the path created by Graham *et al.* [1], this previous paper created a canonical path that not only shows the trajectory post-vaccination, but also prior to vaccination. However, the path was not able to show the trajectory in incidence space in the post-elimination era as data extended only to the year 2000. The paper also qualitatively analyzed the difference between the canonical path of the entire country and the individual state paths. We visually compared the similarities and differences, but did not conduct any form of quantitative analysis. The present paper builds on this preliminary work by addressing several research questions pertaining to the sensitivity of the canonical path to changes in path parameters and methodology. These research questions are outlined in the following section.

1.5 Research Objectives

The goal of this thesis is to build on the preliminary work, in order to better understand the canonical path to elimination. We begin by conducting a sensitivity analysis on the canonical path: we determine how changing parameters pertaining to the creation of the path affect the path and the inferences we make. Additionally, we aim to understand what aspects of a model are required to simulate the US canonical path. Finally, we investigate how the simulated canonical path depends on the properties of a disease, by considering how different seasonality and the natural history of infection affect the path. Addressing these questions has the potential to inform whether these methods could be used for other diseases deemed eradicable by the WHO.

2 Creating a Canonical Path for the US using Incidence Data

2.1 Incidence and Population Data

This study uses several different sources for both incidence and population data. A *.xlsx* file containing weekly measles incidence for each US state up to the year 2000 was obtained from Project Tycho [15]. From 2001 to 2016, annual reports were downloaded from Morbidity and Mortality Weekly Report (MMWR) as *.pdf* files [16, 17], from which we extracted the required data tables using the R function `tabulizer` [18]. The population data is also a combination of two data sources. The first is a data set containing US census counts by state from 1790 to 1990, which was obtained from the US Census Bureau [19]. As the US Census Bureau did not have census counts by state for 2000, 2010, or 2020, the second data set is comprised of intercensal population estimates by state from 2000 to 2010 and the census counts for 2000 and 2010 [20]. An in depth description of how the data was cleaned and combined can be found in section 7.2.

2.2 The US Canonical Path

2.2.1 Creation of a path for a Local Region

To create the path for a single region, we follow the methods of Graham *et al.* [1] to find the weighted mean incidence and the smoothed, unweighted CV. First, we calculate yearly measles incidence per 100,000 using the compiled incidence and population data. Next, we find the weighted mean incidence through the use of a Gaussian-weighted moving average. The peak of the Gaussian is two years before the current point in time, and the standard deviation is 3 years. We begin this process once we have 10 years of data, resulting in the first aggregated data point corresponding to the year 1921. Graham *et al.* [1] used a different method to smooth the CV: they first computed the normal mean and standard deviation of 10-year moving windows, and computed their ratio to find what they call "local CV ". Afterwards, they smooth the "local CV " using a Gaussian filter.

When Graham *et al.* [1] instead calculated CV as the ratio of *weighted* standard deviation and *weighted* mean incidence, they observed a sharp increase in CV when there was a period of time where cases dropped significantly (refer to the page 2 of the supplementary material in Graham *et al.* [1] for more details). This was not observed with their chosen method [1]. For consistency, we decided to follow their same methods for smoothing CV, although we investigate the sensitivity of the path to this choice in section 3.3.

2.2.2 Construction of the Canonical Path for the US as a whole

To construct the canonical path for the US as a whole, we begin by applying the algorithm as described in section 2.2.1 to each individual US state. We then find the mean point in incidence space of the US states each year and plot weighted mean incidence per 100,000 against the smoothed, unweighted CV to define the US canonical path. While Graham *et al.* [1] chose to weight each country equally when finding the mean point in incidence space, we conducted a sensitivity analysis to determine whether it would be better to weight each state based on their population size. We found that there is very little difference between the equally-weighted and population-weighted paths ($d = 0.124$, based on equation 1 below). Thus, we chose to leave the states equally weighted for consistency with Graham *et al.* [1]. Refer to section 7.4.1 for further detail.

Figure 2 shows the canonical path for the US as a whole. We plot incidence on a square-root scale to see finer detail near zero incidence. In the figure, we show a time progression by labeling points every 10 years, starting in 1922; in addition, above the canonical path we show the annual incidence time series. Around 1962, incidence begins lowering and the CV is increasing, suggesting that the path is beginning to progress towards elimination. This observation is exactly what we expect, as the vaccine for measles was invented in 1963 [6]. Additionally, from 1922 to 1942, the path indicates that incidence rose by a factor of 5. This is explained by the fact that in 1922, measles had only been nationally notifiable for 10 years. This could suggest that the cases did not *increase* by a factor of 5, but that measles was being reported less prior to 1942. To support this argument, one could compare births to reported cases over this time, as before vaccination almost everyone got measles [6]. However, as we do not have births data going back to 1922, this would need to done in future work.

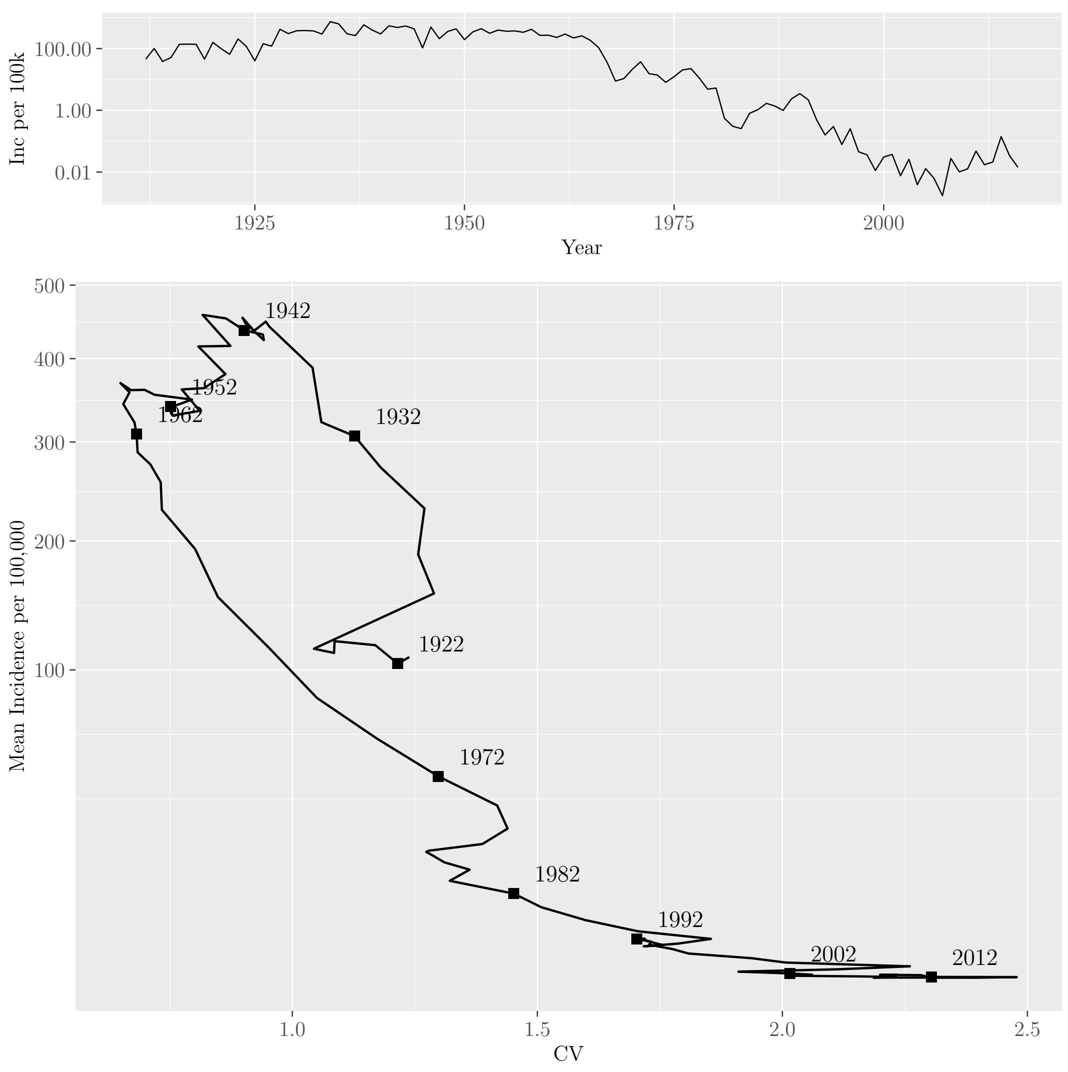


Figure 2: A canonical path of the US as a whole, with the time series plotted at the top. Starting in 1922, a point has been labeled on the path every 10 years to provide a sense of time progression along the path. On the y-axis, weighted mean incidence per 100,000 is plotted on a square-root scale to show finer detail near zero incidence. The x-axis is the smoothed unweighted CV.

2.3 Comparison of the US Canonical Path to that of Individual US States

Using the same methods to calculate the weighted mean incidence per 100,000 and the smoothed unweighted coefficient of variation, as previously discussed, we computed canonical paths for the individual US states. These individual state paths can then be compared to the path for the entirety of the US, in order to investigate to what degree states are deviating from the path derived for the whole country. We compare paths using the Euclidean distance between them. Equation 1 below is used to calculate the distance, d.

$$d = \sqrt{\frac{1}{N}\sum_{i=1}^{n}\left((\mathrm{inc}_{1,i} - \mathrm{inc}_{2,i})^2 + (\mathrm{cv}_{1,i} - \mathrm{cv}_{2,i})^2\right)} \tag{1}$$

inc_1 and cv_1 are the coordinates in incidence space of the first path, inc_2 and cv_2 are the coordinates in incidence space of the second path, and n is the number of points in the paths. This distance also works for $n = 1$, which represents the comparison of the two paths for a single year of data. We define the distance with $n = 1$ as the pointwise distance between two paths in a given year. We compute the pointwise distances in section 2.5.

If we were to evaluate the distance without applying any scaling, all of the weight would be placed on incidence, since it is on a much larger scale than CV . We therefore scale the data before computing the distance. Graham *et al.* [1] chose to take the log of incidence, and then scale both log incidence and CV to be between 0 and 1. If we were to use this method for our data, the scaling of incidence prior to vaccination and CV in the elimination era would be too strongly affected due to high levels of variation. To avoid this issue, we chose to instead scale to the 90th percentile of log incidence and CV. This method resulted in both axes being scaled consistently, without impacting the scale of incidence pre-vaccination and the scale of CV in the elimination era. We conducted a sensitivity analysis with respect to the percentile, comparing path generated with percentiles of 0.8 to 0.95. We found that the paths are not sensitive to the choice of percentile in this range (See section 7.3.4.1).

In initially comparing individual US states to the aggregate US path, we discovered that there were several states with paths that were impossible to interpret, e.g., Mississippi. In plotting each state's time series, we discovered that it was not until roughly 1960 that all states were sufficiently reporting measles. Thus, we chose to only compare each state to the US path for years after 1959. Using our distance measure (1) to compare the canonical path of each US state to the path for the US as a whole in this time period resulted in a median distance of 0.281 (Interquartile Range (IQR): 0.093). The state path that was furthest from the US as a whole was North Dakota ($d = 0.509$), whereas the closest was Virginia ($d = 0.183$). Figures 3 and 4 compare the canonical paths for Virginia and North Dakota with the aggregate US path. The paths of Virginia and North Dakota are coloured based on the fact that they are Democratic and Republican stronghold states respectively [21, 22]. A comparison of the paths of remaining states to the aggregate path can be found in section 7.3.3.1; each state is coloured in the same fashion to describe whether it is a Democratic or Republican stronghold, or a swing state [21–23].

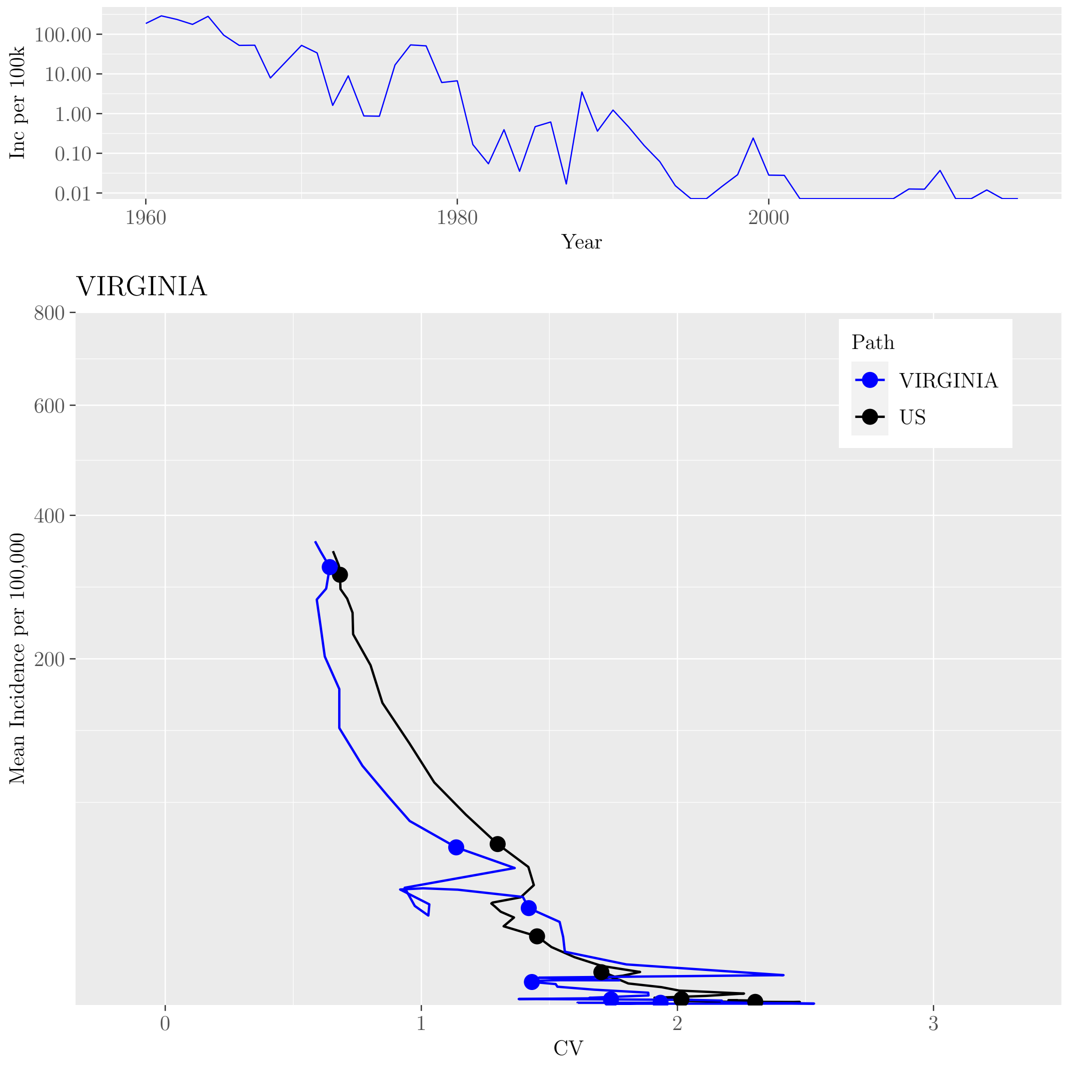


Figure 3: A comparison of the canonical path for the US (black) to that of Virginia (blue, democratic), with the time series of Virginia included at the top of the figure. On the y-axis, weighted mean incidence per 100,000 is plotted on a square-root scale to show finer detail near 0 incidence. The x-axis is the smoothed unweighted CV. Evaluating the distance between the paths identified Virginia as the path most similar to the aggregate US path with a distance of 0.183. This result is well represented in this plot as Virginia seems to follow a very similar trajectory to that of the aggregate US path.

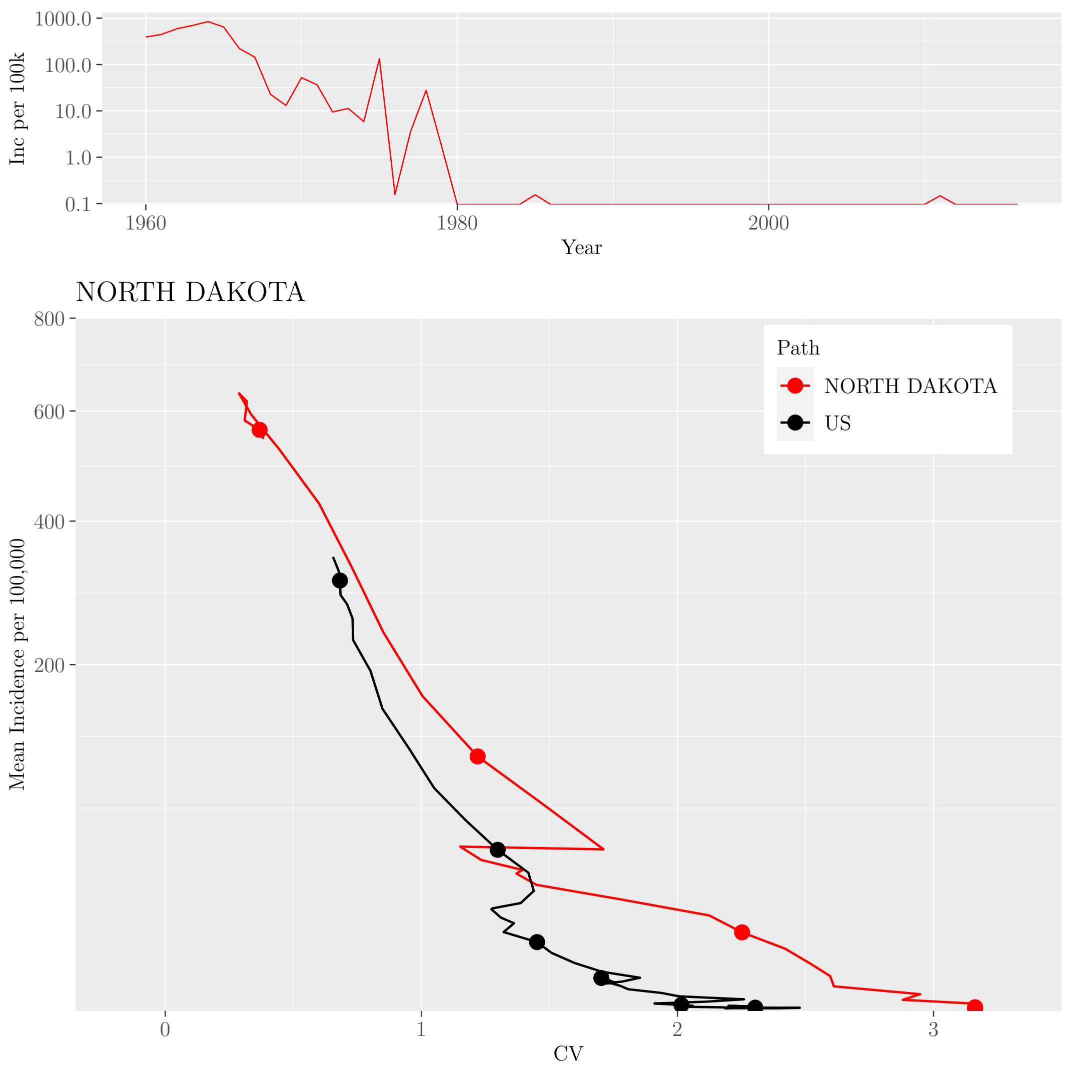


Figure 4: A comparison of the canonical path for the US (black) to that of North Dakota (red, republican), with the time series of North Dakota included at the top of the figure (*cf.* Figure 3). Evaluating the distance between the paths found that North Dakota is the most dissimilar path ($d = 0.509$).

2.4 Assigning Each US State its Closest Point on the US Path

To understand how the US states have progressed along the path over time, we chose to assign each US state its closest point on the US canonical path for each of the following years: 1962, 1972, . . . , 2012. As there are 51 states, and thus 51 points that will be labeled on the path for a given year, we create separate figures for Republican and Democratic stronghold states each year. We chose to split the data by political stronghold, as an article in the New York Times outlined how political stronghold is a very good predictor of COVID-19 vaccination [24]. Additionally, we split the plots into 2 separate grids in figures 5 and 6. In comparing the progression of Republican to Democratic stronghold states, we observe little difference in the speed of progression. This suggests that once state began reporting roughly uniformly, they progressed towards the elimination of measles at similar paces. We present a similar figure for Swing states in the supplement (Section 7.3.5).

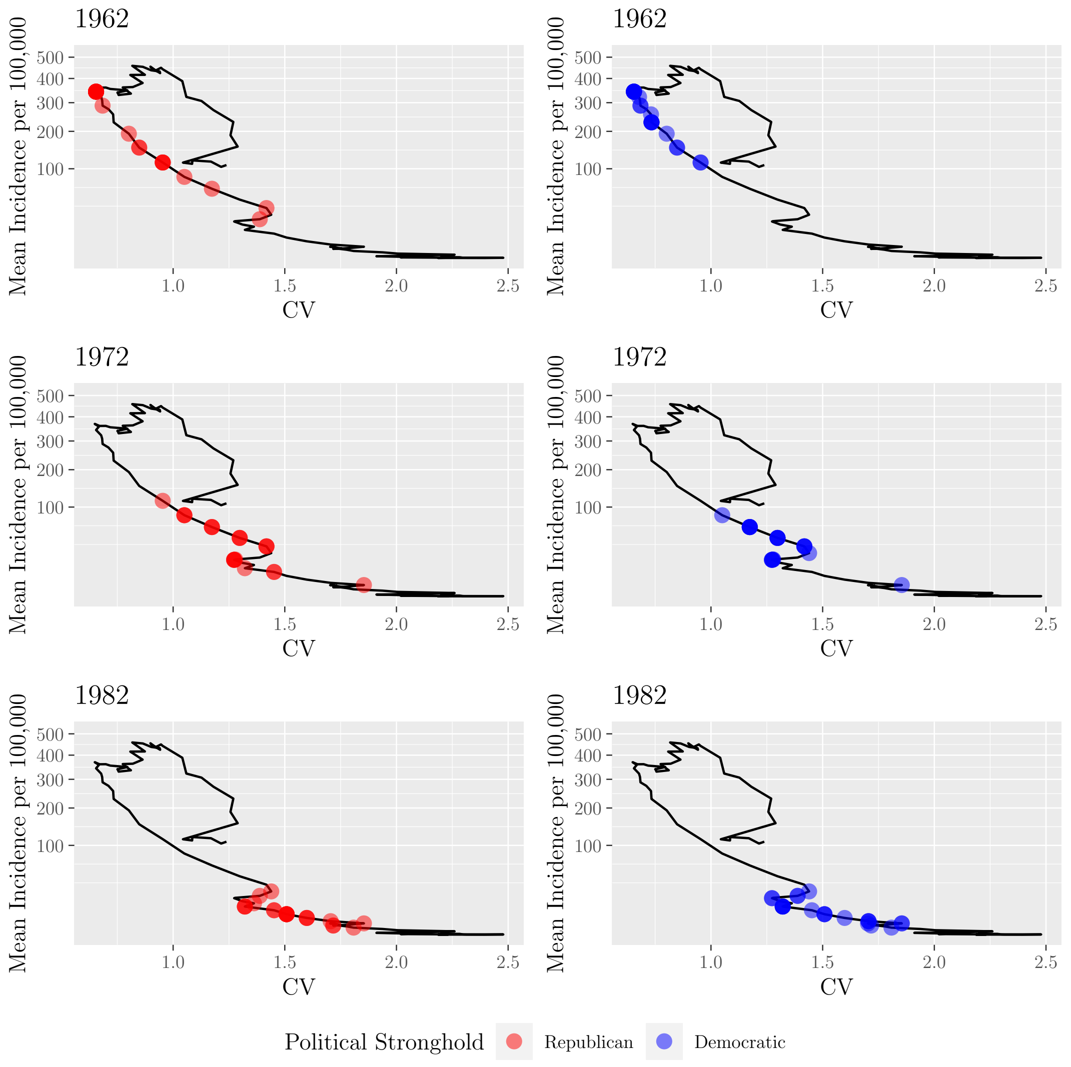


Figure 5: Demonstration of how US States progress along the US canonical path over time by presenting a grid with blocks showing the plots which correspond to the closest point in the following years: 1952, 1962, and 1972. We show two separate plots for Republican (red) and Democratic (blue) states to compare progression among these groups. The same plot created for the years 1982, 1992, and 2002, is presented in figure 6.

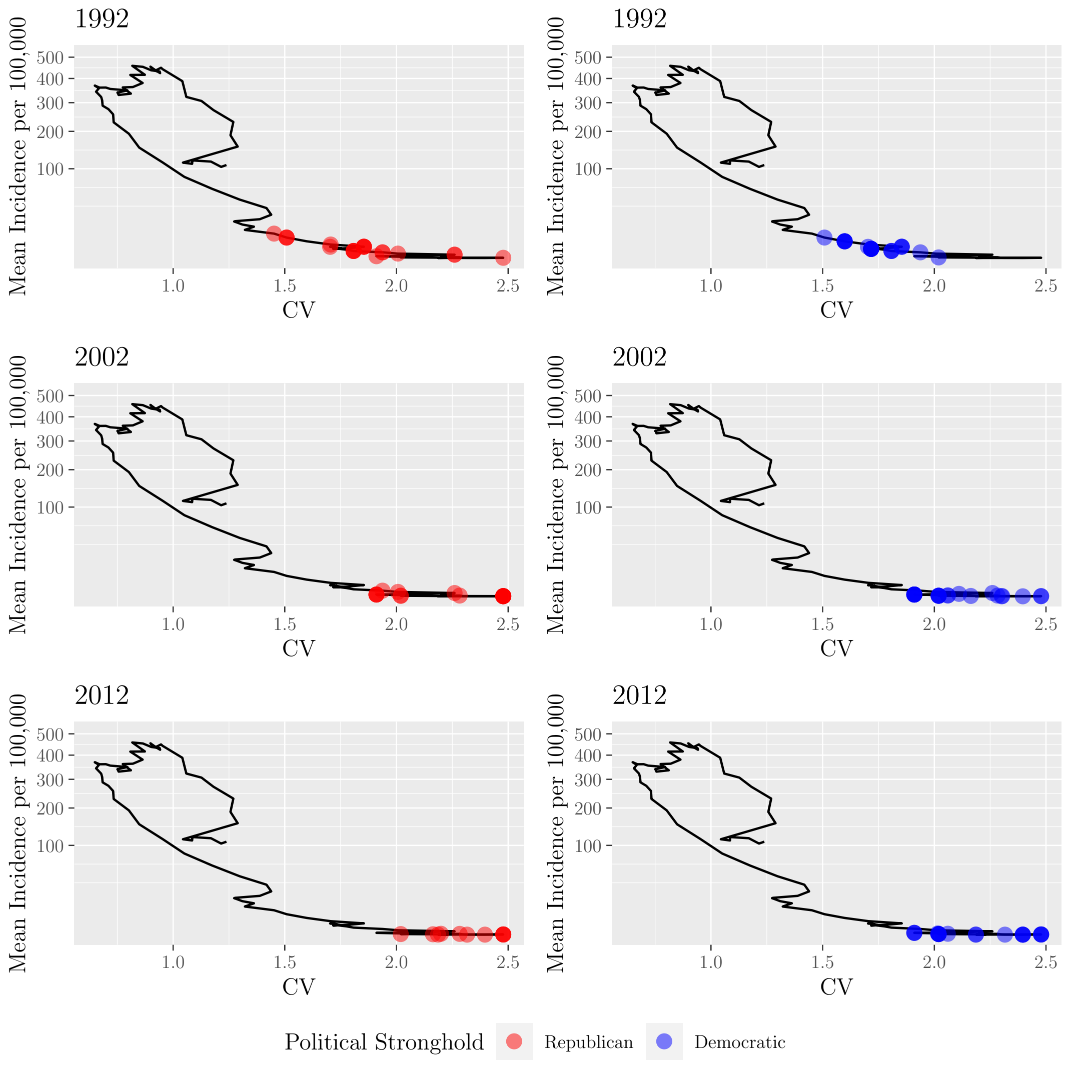


Figure 6: Demonstration of how US States progress along the US canonical path over time by presenting a grid with blocks showing the plots which correspond to the closest point in the following years: 1982, 1992, and 2002. We show two separate plots for Republican (red) and Democratic (blue) states to compare progression among these groups.

2.5 Pointwise Distances between States and the Aggregate US Path over time

To understand how the paths of individual states vary from the aggregate US path over time, we found the distance between the position of the US as a whole and each individual state in incidence space, each year from 1960 to 2016. Creating a box plot of this data shows an almost monotonic increase in variation over time and a seemingly systematic change in the median over time (see figure 7). The increase in variation is likely attributed to the fact that as we approach elimination, the paths of each state are noisier. This nature of the individual state paths also explains why there is an observed increase in the median distance as the country progresses towards elimination, since near elimination the variation in CV increases the dissimilarity among US states.

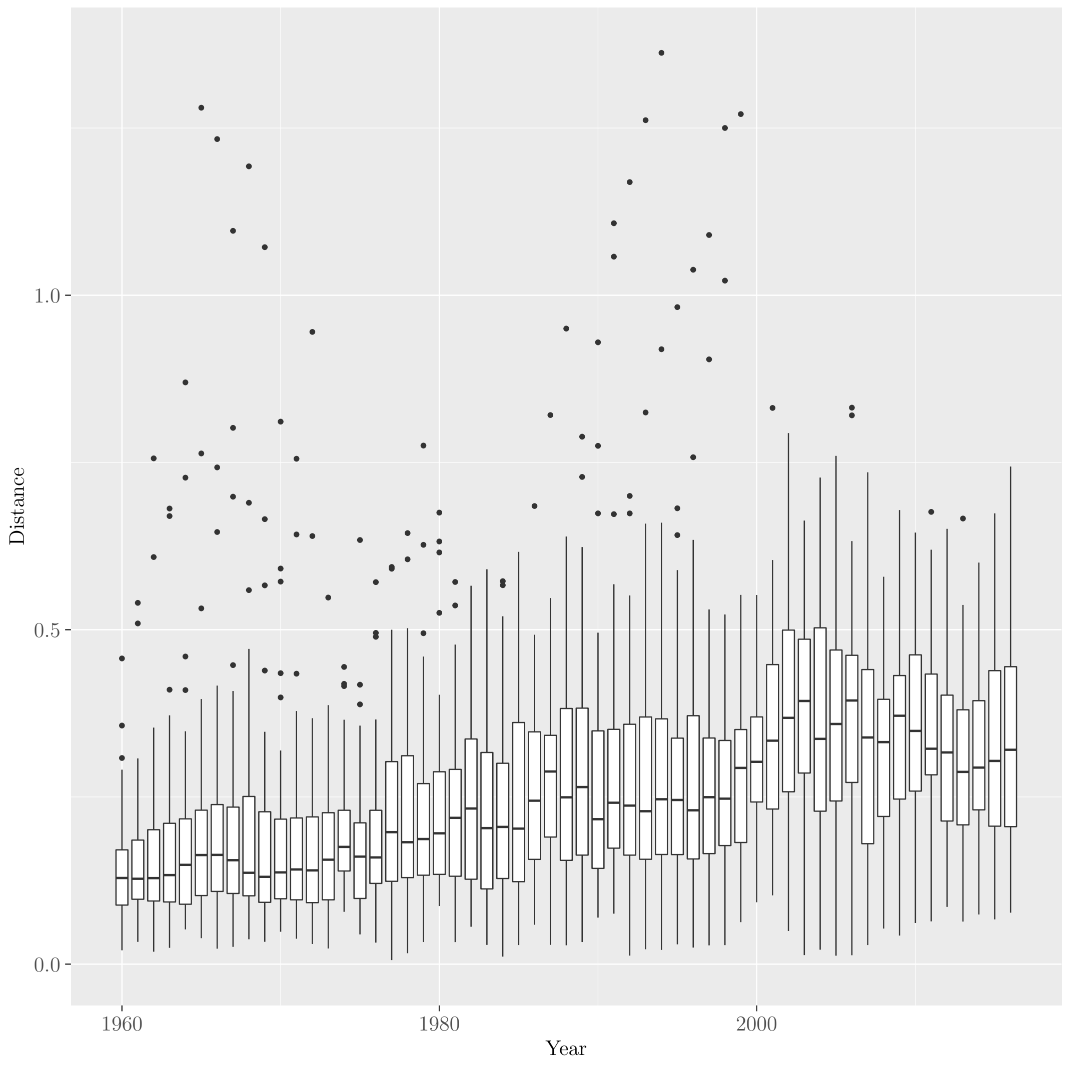


Figure 7: A box plot demonstrating how the pointwise distance (section 2.5 between each state's canonical path and the aggregate US path changes over time.

As shown in preliminary work [14] and in the previous sections, it is possible to construct a canonical path for an individual country using paths of individual states, in the same way that Graham *et al.* [1] created a path for a WHO region from paths of individual countries. We found that different states follow the aggregate path to differing degrees. However, in order to make inferences, for instance regarding how the structure of the path depends on characteristics of the disease, we must have a robust understanding of how sensitive the path is to our assumptions when creating the path, and also how dependent the path is on the characteristics of a disease. The following sections will explore these important questions.

3 Sensitivity Analysis of the Aggregate US Canonical Path

A key goal of canonical path analysis for measles is to investigate how different control strategies affect the path to elimination. In particular, this was investigated by Graham *et al.* [1], who looked at the effect of targeting different age cohorts with supplementary immunization activities. However, as previously mentioned, in order to use the canonical path for these types of inferences, we need to understand how sensitive the path is to the details of its construction. The following sections investigate how the values of several parameters affect the derived canonical path.

3.1 Method of Weighting Mean Incidence per 100,000

As defined by Graham *et al.* [1], the canonical path is generated by plotting points in *incidence space*. In incidence space, the x-axis is the smoothed, unweighted coefficient of variation, and the y-axis is the weighted mean incidence per 100,000. To calculate weighted incidence, Graham *et al.* [1] chose to use a Gaussian filter, putting the most weight on the mean incidence two years prior and weighting all years in the past less. While this is a reasonable choice, we are interested in determining whether or not we could use a simpler moving average and achieve the same result. Figure 8 shows the comparison of using a Gaussian filter *vs.* a uniformly weighted moving average to calculate the weighted mean. We include the time series of data used to create each of the paths at the top of the figure. Since the weighting does not change the time series, each path has an identical time series.

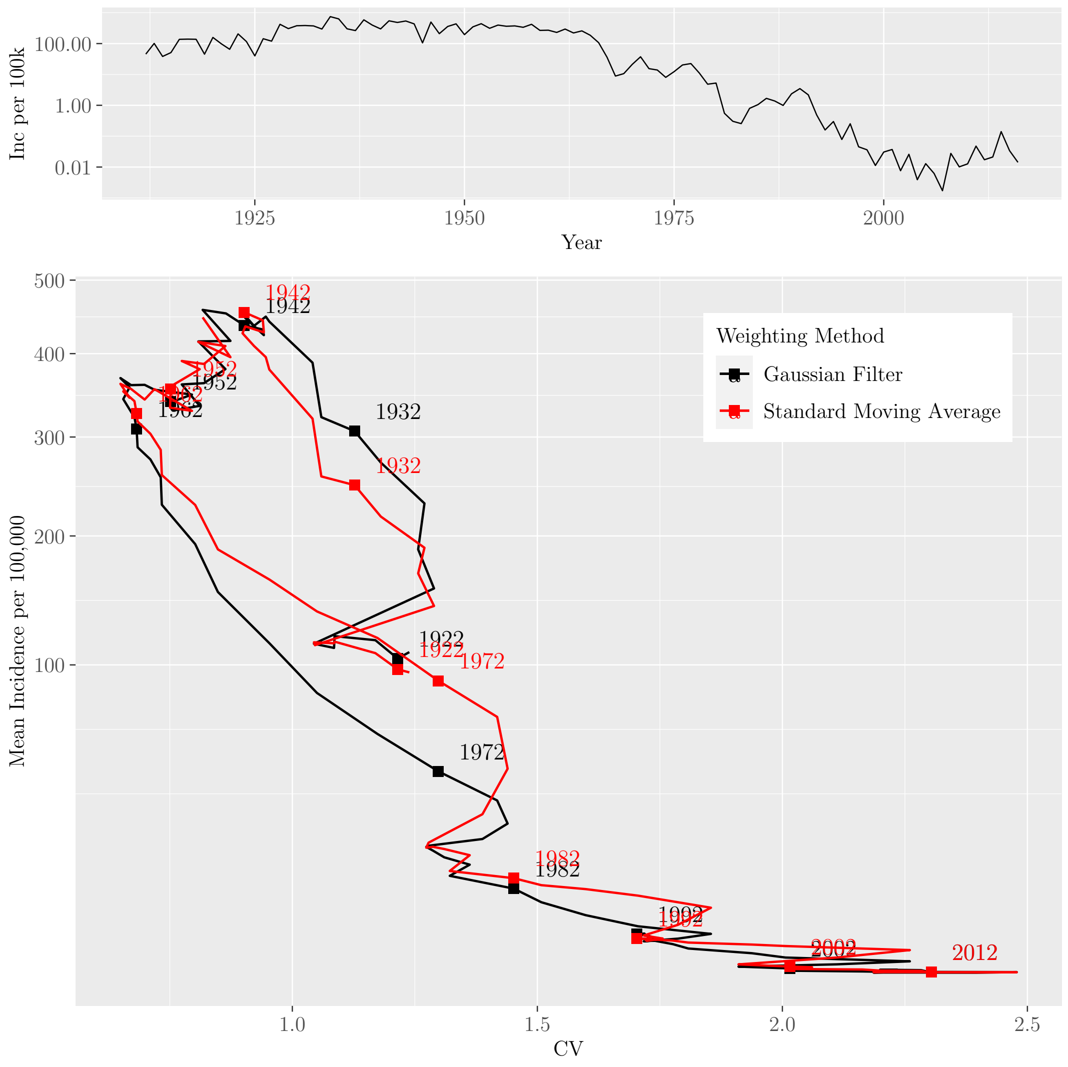


Figure 8: Comparison of canonical paths constructed with a Gaussian filter (black) *vs.* a standard moving average (red) in order to weight mean incidence. Points are labeled every 10 years starting in 1922 to have a sense of time progression along the path, as well as to aid in the comparison of the paths.

Figure 8 shows that the canonical paths are similar, regardless of which of the two ways the moving average is weighted (the distance (1) between the two paths is $d = 0.059$). Before 1942, mean incidence is typically higher when calculated using the Gaussian filter, whereas between 1962 and 1992 we see the opposite. The overall similarity of the two paths suggests that we could use the simpler standard moving average to calculate weighted mean incidence, without considerably affecting any inferences.

3.2 Changing the Shape of the Gaussian Filter

When using the Gaussian filter, we must specify its centre (c) relative to the current time, and its standard deviation (σ). Graham *et al.* [1] chose $c = -2\,\text{yr}$ and $\sigma = 3\,\text{yr}$. We investigated sensitivity of the canonical path to these parameters, considering $-5\,\text{yr} \leq c \leq 0$ and $1\,\text{yr} \leq \sigma \leq 10\,\text{yr}$. Figures 9 and 10 examine sensitivity to c and σ, respectively. In each figure we include the time series of data used to create each of the paths at the top of the figure. Since the weighting does not change the time series, each path has an identical time series.

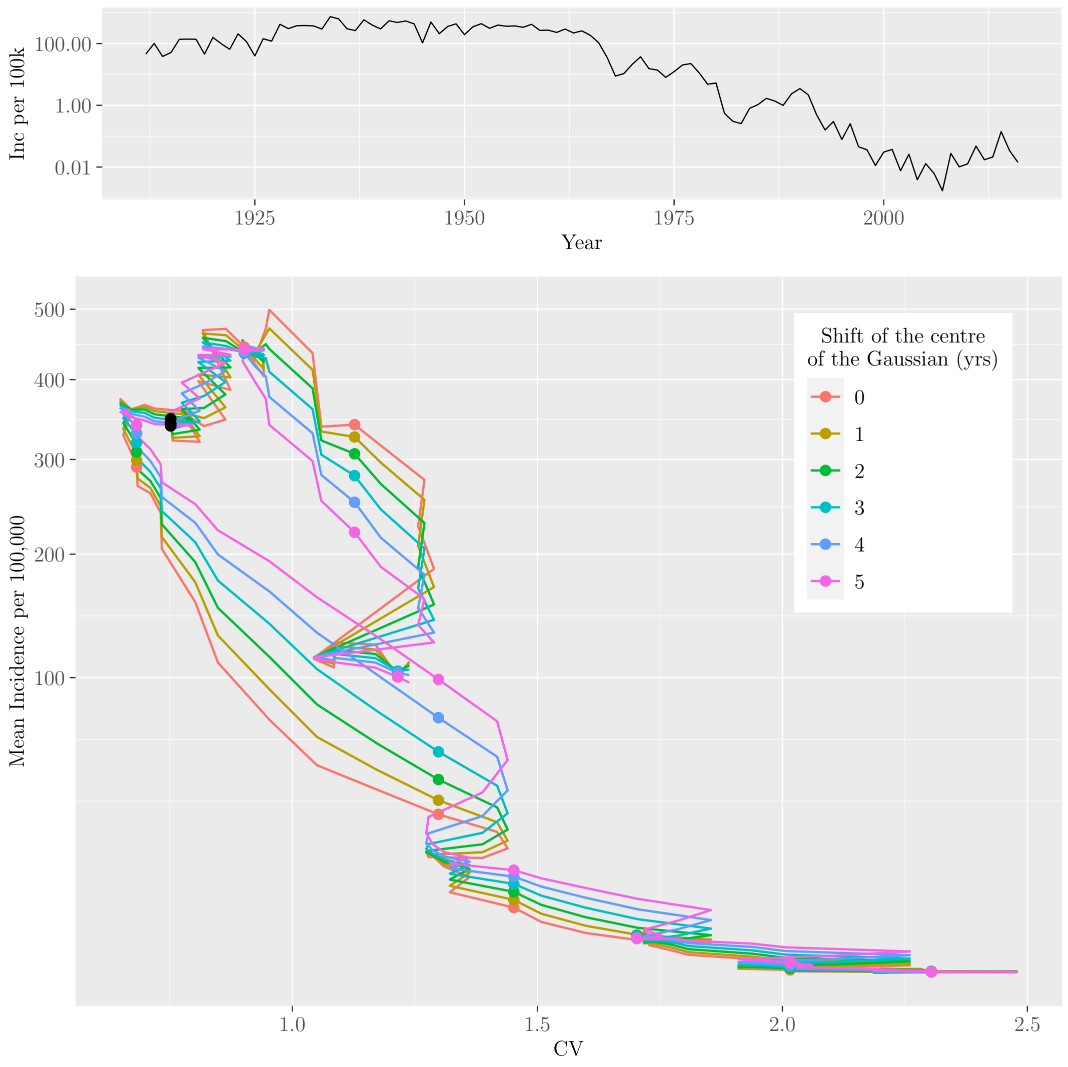


Figure 9: Comparison of the aggregate US canonical path created using values for the centre of the filter ranging from 0 to 5 years shifted from the current point in time. Points are highlighted with a large dot every 10 years starting in 1922, with the point corresponding to 1952 coloured black.

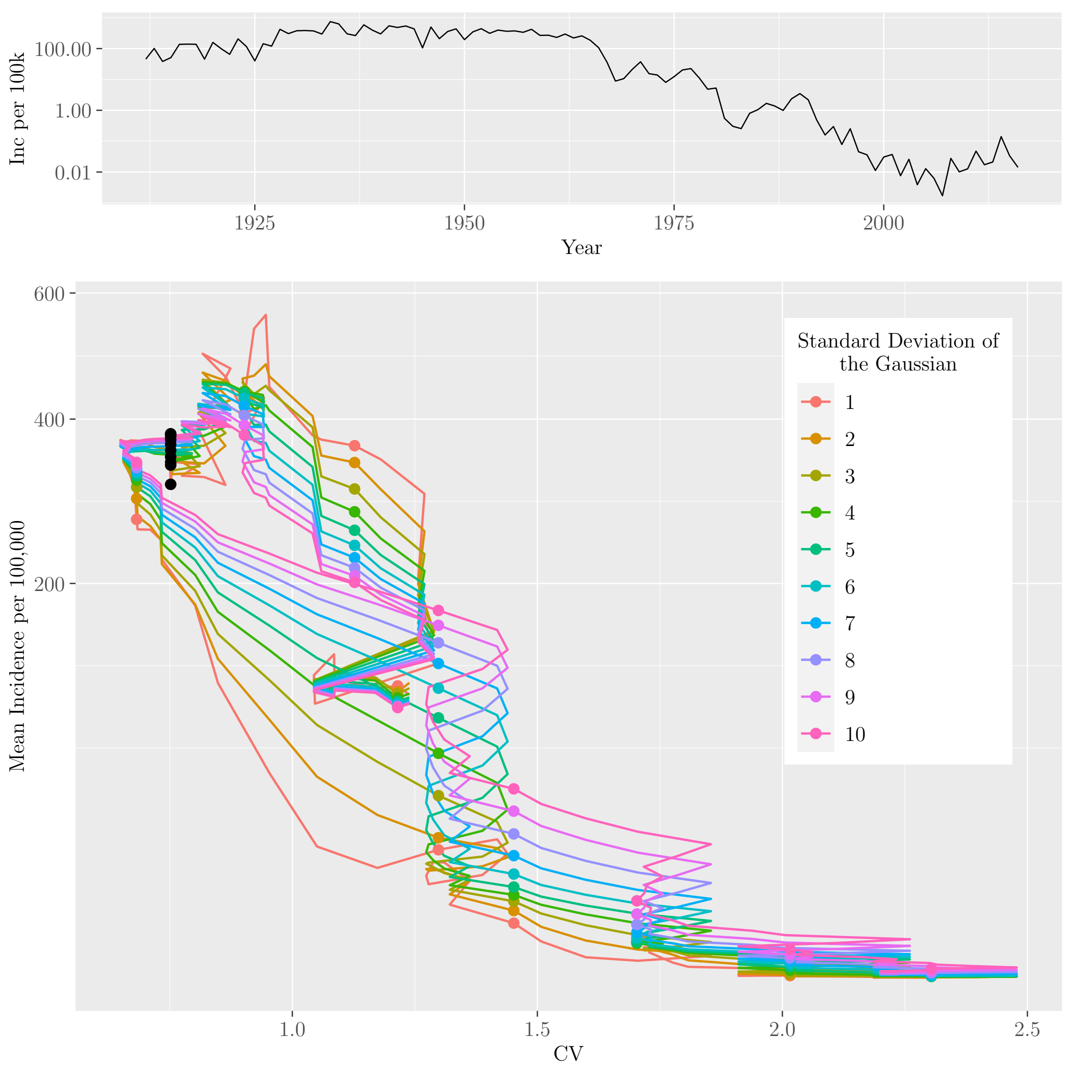


Figure 10: Comparison of the aggregate US canonical path created using values for the standard deviation of the Gaussian filter ranging from 1 to 10 years. Points are highlighted with a large dot every 10 years starting in 1922, with the point corresponding to 1952 coloured black.

In figures 9 and 10, increasing the number of years we shift the centre weights the past more heavily than the present, whereas increasing the standard deviation effectively increases the smoothing window. From 1921 to 1952, as the centre is shifted further into the past, when the number of cases is lower than of the current point in time, increasing the weight of the past reduces weighted incidence. Additionally, when we increase the smoothing window, we decrease the weight of the centre of the Gaussian and increase the weight of the tails. However, since we use a truncated Gaussian, with most of the left tail cut off, this has the most impact on the points in the future (the right tail). Thus, since the majority of cases in the future are less than that of the present, weighted incidence is reduced.

After 1952, shifting the centre further in the past results in increased weighted incidence, as cases in the past were higher than those of the present due to vaccination [6]. Furthermore, increasing the standard deviation also increases weighted incidence, since the left tail now contains a larger number of points with cases higher than that of the present. Thus, increasing the the weight of the tails increases weighted incidence.

From this analysis, although we do observe differences in the paths, they are easily explained by the history of measles. This suggests that the choice of the location of the centre or the standard deviation of the Gaussian filter will not have a substantial impact on the inferences we make.

3.3 Method of Weighting the Coefficient of Variation

Another choice made by Graham *et al.* [1] was to not weight the CV in the same way as used for the mean incidence. When they attempted to weight the CV in the same way, whenever there was a period of time where incidence was much lower than before (near elimination), there was a sharp increase in the weighted CV (refer to page 2 of the supplement from Graham *et al.* [1] for more detail). Since in this paper we are using finer scale data, we decided to see if this behaviour is also observed with the aggregate US path. Thus, in figure 11, we compare the path created using a Gaussian filter to weight both incidence and CV , to the path with CV weighted by first finding the 10-year average CV and then smoothing using the R `smth.gaussian()` function from the `smoother` package. Similar to figures 8 through 10, we include the time series of each path, which is identical, at the top of the figure.

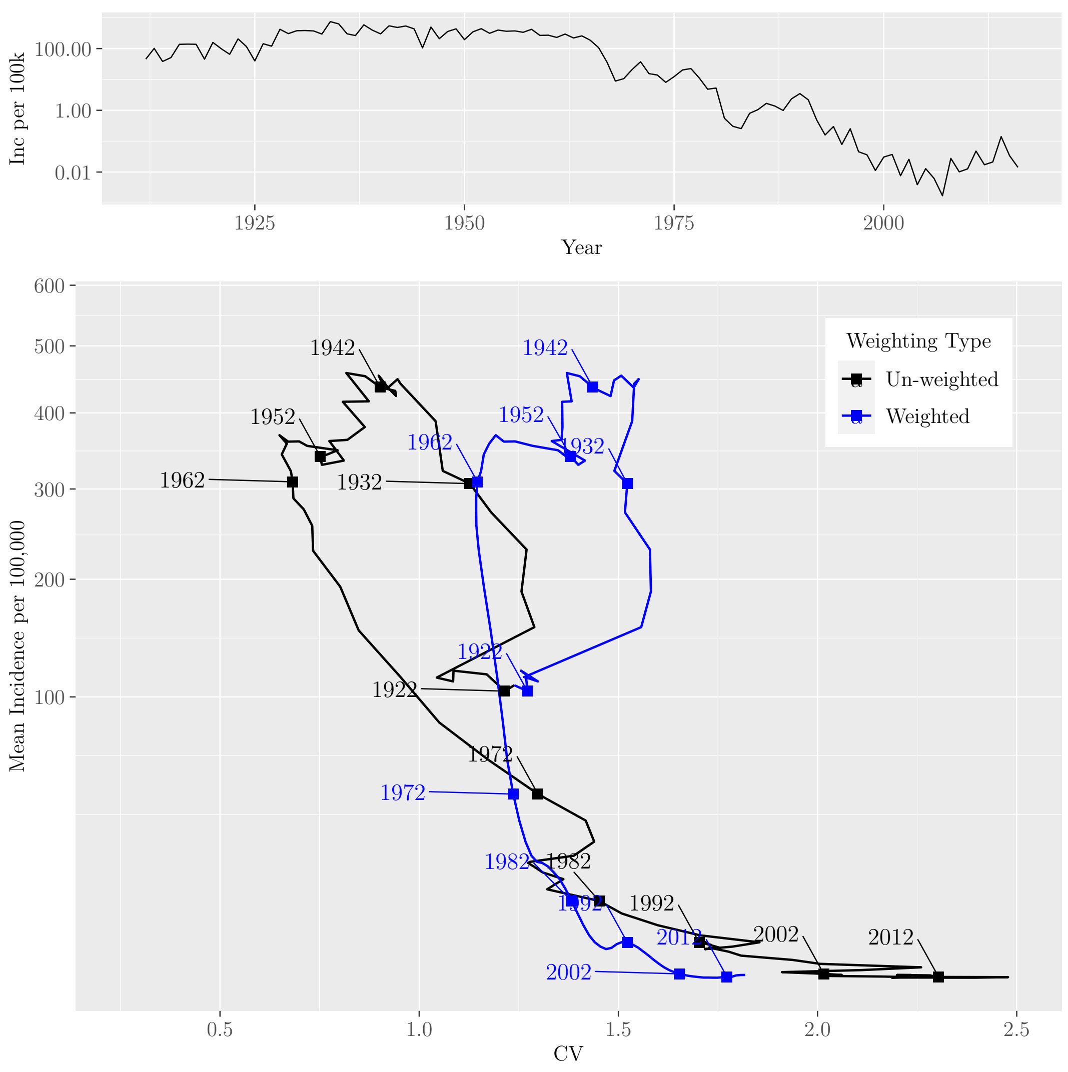


Figure 11: Comparison of the path created using the same scaling method as used for the mean incidence (blue), to the scaling method suggested by Graham *et al.* [1] that instead finds the CV every 10 years, and then smooths it using a Gaussian smoothing function in R (black) [1]. Points are plotted every 10 years starting in 1922.

In comparing the paths, there is very clearly a difference between the two paths. Using the defined distance (Equation 1) results in a distance of 0.188. In the path that uses a Gaussian filter to weight CV (blue), the weighting resulted in a much smoother path, with a smaller range of CV values. Interestingly, we do not observe any sharp increases in weighted CV as observed by Graham *et al.* [1]. Since we are creating the path on a finer scale, one would assume that we would observe more variation in the CV due to the greater amount of demographic stochasticity, but we are actually observing the opposite. Regardless, the results suggest that the better choice of weighting is to first calculate the 10-year average and then smooth, as the other method appears to reduce variation in the CV of incidence, which can get rid of any patterns in the US path.

Overall, the results of the sensitivity analysis suggest that the choices we make regarding the calculation of weighted incidence will not have a large impact on inferences. However, over-smoothing the CV hides actual variation in CV, compromising our ability to draw information from position on the canonical path.

4 Simulating the US Canonical Path

4.1 Building the Model

In order to fully understand any other factors that may influence the path, we must be able to simulate it, *i.e.*, we must be able to construct an equivalent path from simulated case report data (for which the generative mechanisms are exactly known). In their paper, Graham *et al.* [1] simulated the paths of individual countries in the WHO Americas and African regions using a discrete time age-structured model, which has a model matrix that defines transitions from all possible epidemiological stages[2] and age combinations to all other possible stages and age combinations, at each time step. The simulations used country-specific data for age distribution, vital dynamics, vaccination coverage, and other derived measles parameters such as rate of decline in maternal immunity, vaccine efficacy, basic reproductive number, and age-dependent contact rates [1]. As the model used by Graham *et al.* [1] was rather complex, one of the goals of the present paper is to determine what aspects of the model are truly required to simulate the path.

When creating our model, we decided to start with the base Susceptible-Exposed-Infectious-Recovered (SEIR) model with vital statistics and vaccination, and built it up until we were able to simulate the US path effectively. The base SEIR model equations are presented in equation 2.

[2]The stages included by Graham *et al.* [1] were: maternally immune, susceptible, infected, recovered, and vaccinated.

$$\begin{aligned}\frac{dS}{dt} &= \nu N(1-p) - \frac{\beta SI}{N} - \mu S\\ \frac{dE}{dt} &= \frac{\beta SI}{N} - \sigma E - \mu E\\ \frac{dI}{dt} &= \sigma E - \gamma I - \mu I\\ \frac{dR}{dt} &= \nu N p + \gamma I - \mu R\end{aligned} \tag{2}$$

In this equation, ν is the birth rate, N is the population size, p is the proportion of the population vaccinated, β is the transmission rate, μ is the death rate, $\frac{1}{\sigma}$ is the mean latent period, and $\frac{1}{\gamma}$ is the mean infectious period. Sections 4.1.1 through 4.1.4 outline the steps taken to build a model that accurately reproduced the US canonical path.

4.1.1 Time-varying β and vaccination

Since measles is seasonally forced, the first element added to the model was a time-varying transmission rate that uses cosine with a period of one year to represent the seasonality (equation 3) [25].

$$\beta = \beta_0(1 + \alpha\cos(2\pi t)) \tag{3}$$

$$\mathcal{R}_0 = \beta_0 \times \frac{\nu}{\mu} \times \frac{\sigma}{(\sigma+\mu)(\gamma+\mu)} \tag{4}$$

In equation 3, β_0 is the mean transmission rate calculated using the expression for the basic reproduction number, $\mathcal{R}_0$, (equation 4) and α is the amplitude of seasonal forcing. The value of α is chosen through optimization in section 4.2.

Since we are modeling the measles dynamics over the past 100 years, we also implemented time-varying vaccination. This is accomplished by creating a function that sets $p = 0$ before 1963, and linearly increases from 0 to the current estimated measles vaccination coverage in the US, approximately 90.7% [26].

4.1.2 Treating each simulation as a US state based on population

When we create the US canonical path, we are not using data for the entirety of the US; we use state level data and find the mean. To mimic this, we ran 51 simulations, each representing a US state based on the state's population in 1963. However, since most states have a population less than the critical community size for persistence in the model ($\sim$ 1 million [27]), we experienced extinction in the majority of simulations. To avoid fadeouts, we implemented immigration. This is accomplished by changing the $\frac{\beta SI}{N}$ term, to instead be $\frac{\beta S}{N}(I+q)$. In this expression, q models immigration as prevalence from imported infections. In order to best represent immigration, q needs to be a function of time, since we assume that as vaccination rates change, so will the number of infections resulting from immigration. We therefore took q to be a function of the equilibrium prevalence, which also changes over time due to vaccination. Thus, q becomes the equilibrium prevalence at each time point, multiplied an immigration factor, f (Equations 5 and 6). The value of f is chosen through optimization in section 4.2.

$$I_{\text{eqm}} = N \times \frac{\nu\sigma}{(\sigma+\mu)(\gamma+\mu)} \times \left((1-p) - \frac{1}{\mathcal{R}_0}\right) \tag{5}$$

$$q = I_{\text{eqm}} f \tag{6}$$

4.1.3 Using real data to improve the representation of vaccination and vital statistics in the US

Using the model to simulate each US state, solely by changing the population size in individual simulations, created a path that was not representative of the US canonical path, as it had a different shape and did not match up at all. To improve the model further, we updated the vaccination function, that we set in a naive way in section 4.1.1, to use US vaccination coverage data.

We collected cleaned Measles-Mumps-Rubella (MMR) vaccine coverage data for the entirety of the US from 1980 to 2020 from the WHO [26]. Additionally, we obtained cleaned MMR vaccine coverage data among 0–35 month olds for each US state from 2011 to 2017 from the National Center for Immunization and Respiratory Diseases [28]. We were able to find un-cleaned state-wise data from 1995 to 2010 and 2018 to 2019, however the time required to clean the data was not feasible for this paper [29]. This data was discussed in detail in section 7.2.5. Using the country-wide coverage from 1980, we updated the function to linearly increase to the coverage in 1980, and then moving forward, it either uses the coverage for the given time, or chooses the coverage value for the closest point in time. Although the function would be better with more data, this allowed for a more accurate representation of the vaccination time series within the simulations.

We further improved the model by combining births and deaths data by US state from 4 different sources into a single, working *.csv* file, which was then used to implement time varying birth and death rates into the model. This is represented by a function that either takes the given birth or death rate value at the time step or chooses the value based on the closest point in time. [30–33]. The vital statistics data is discussed in detail in section 7.2.4.

4.1.4 Accounting for observation error and preventing transient behaviour

Although the components added thus far resulted in a path that followed a very similar shape as the aggregate US path, its incidence was on a larger scale. This is due to the fact that the model simulates incidence, while the path with real data is reported cases. Thus, another component added to the model was to account for observation error.

Before vaccination, it was estimated that almost everyone ended up getting measles [6]. Thus, to estimate the under reporting rate, we compared data for births and reported cases in this era by plotting the ratio over time. However, we must divide reported cases in a given year by births. The year is shifted by the mean age at infection to approximate the year in which those with reported cases would have been born.[3] Since we do not know the exact mean age at infection for measles, we compare these ratios for different values of the mean age at infection in figure 12. From this figure, we estimate that approximately 10% of cases are reported. Additionally, we observe little variation across mean ages at infection.

[3]This is a crude estimate as we are assuming that everyone that gets measles from that birth cohort, gets measles at the mean age at infection. To improve this estimate, we could collect age-structured data and implement the methods used by Fine and Clarkson [34, 35].

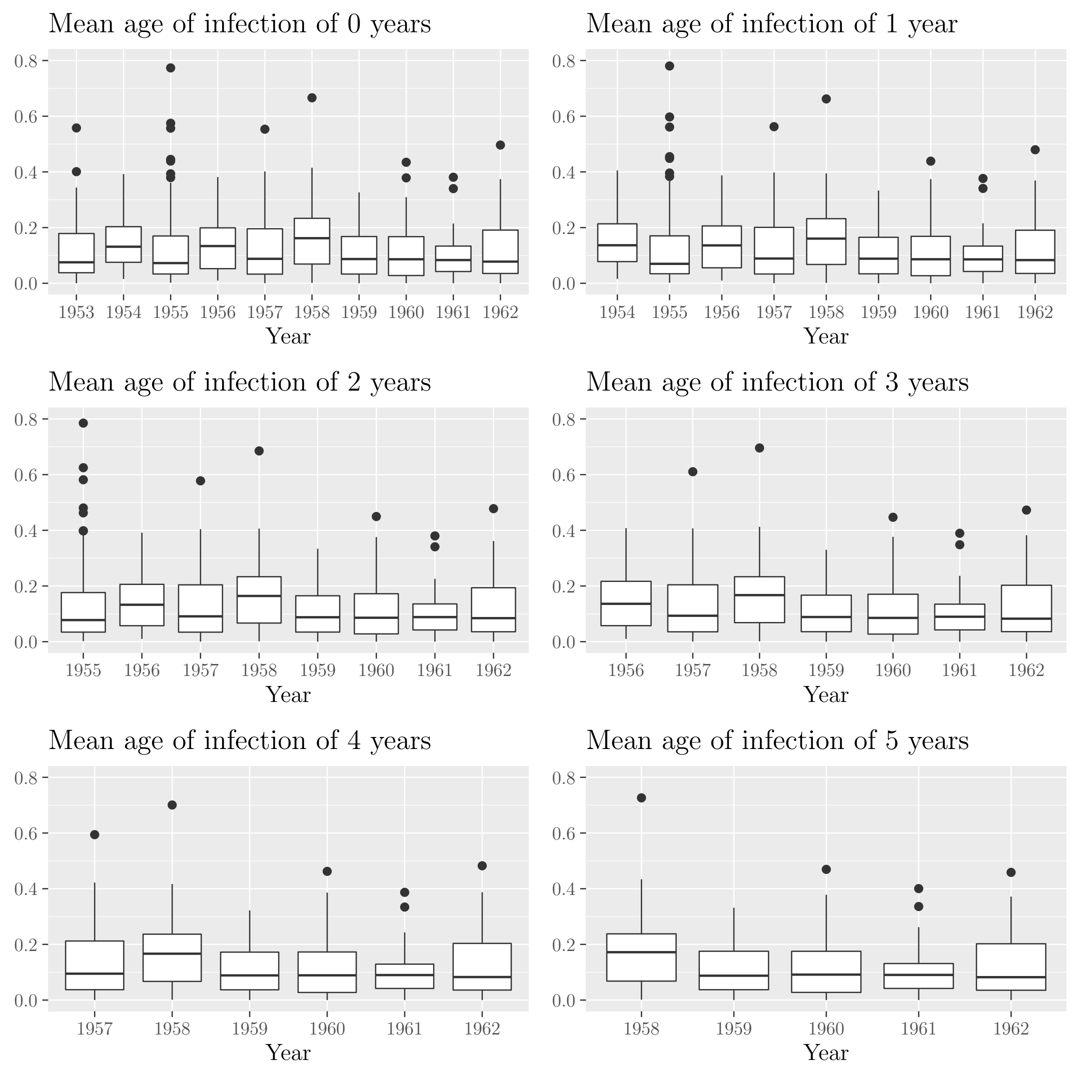


Figure 12: As an estimate of the proportion of cases reported, these box plots show how the ratio of reported cases in a given year to the births in the year shifted by the mean age at infection (y) changes over time (x) in the pre-vaccination era. Each plot represents different mean ages at infection. We define the proportion of un-reported cases as $1 - y$. This figure shows that, over time and across values of the mean age at infection, there is not much change in the median proportion reported. We estimate that approximately only 10% of measles cases are reported. We account for under reporting in the model using this estimated value of the proportion of cases reported with a binomial to sample our simulated incidence (Refer to section 7.5.1.5 for further detail).

To account for this level of under reporting, we sample the simulated incidence using a binomial, that requires the number of observations, the target number of reports (incidence), and the probability of a case being reported, 0.1. This resulted in a path that follows not only a similar trajectory, but is also very close in scale based on incidence.

The final addition to the model was simply to run for extra years at the initial conditions to ensure our model is converging to an attractor and not exhibiting transient behaviour. For our analysis we chose a length of the transient of 200 years. For more details on how we built the model, refer to section 7.5.

4.1.5 Model Comparisons

For the purposes of this paper, we found that we were able to construct a canonical path for the US using a simpler model that did not require age-structure or any additional epidemiological stages, as used in the model from Graham *et al.* [1]. However, we haven't implemented the model created by Graham *et al.* [1], so we cannot outline the difference the model makes for our data. This is something than can be done in future directions (Section 6.1). Additionally, although we were able to simulate the US path using a simpler model, we discovered that the knowledge of measles natural history of infection, vaccination timeline, and the availability of data, was crucial in order to simulate the path effectively. This suggests that this method is only useful for diseases in which we have extensive knowledge of the natural history of infection, vaccine efficacy, and level of waning immunity.

4.2 Optimization of Model Parameters

With a model that represents the aggregate US path reasonably well, the next step was to find values of certain parameters that do not have standard values, specifically, the immigration factor, f, and the amplitude of seasonal forcing, α. For α values ranging from 0 to 0.8 and f values of 0 and 10^{-9} to 0.01, we simulate the US canonical path for all 81 parameter combinations in the 2-dimension (2D) parameter space. To find the optimal parameter combination, we minimized the distance between each simulated path and the US canonical path using the distance defined in equation 1. A heat map of distance for each pairing of α and f is displayed in figure 13. In this figure grey blocks represent parameter combinations that resulted in extinction. We also create a figure comparing the canonical path of the US to each path created from the top 10 parameter pairings in figure 14 and plot each paths time series in figure 15.

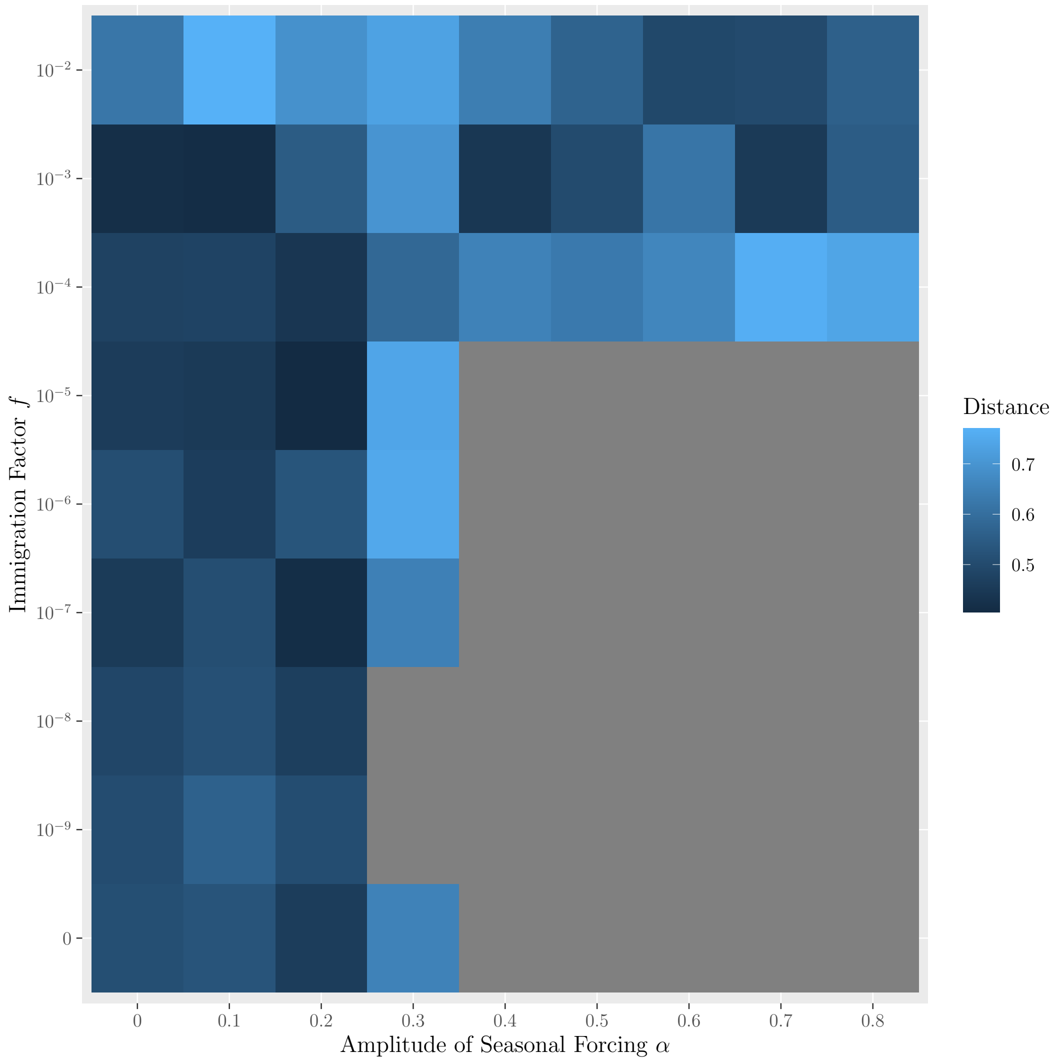


Figure 13: Heat map of distance with the immigration factor, f, on the y-axis and α on the x-axis. Each grid block is coloured a shade of blue based on the value of distance. Grey blocks represent parameter combinations that resulted in extinction. The minimum distance occurs when $\alpha = 0.2$ and $f = 10^{-5}$.

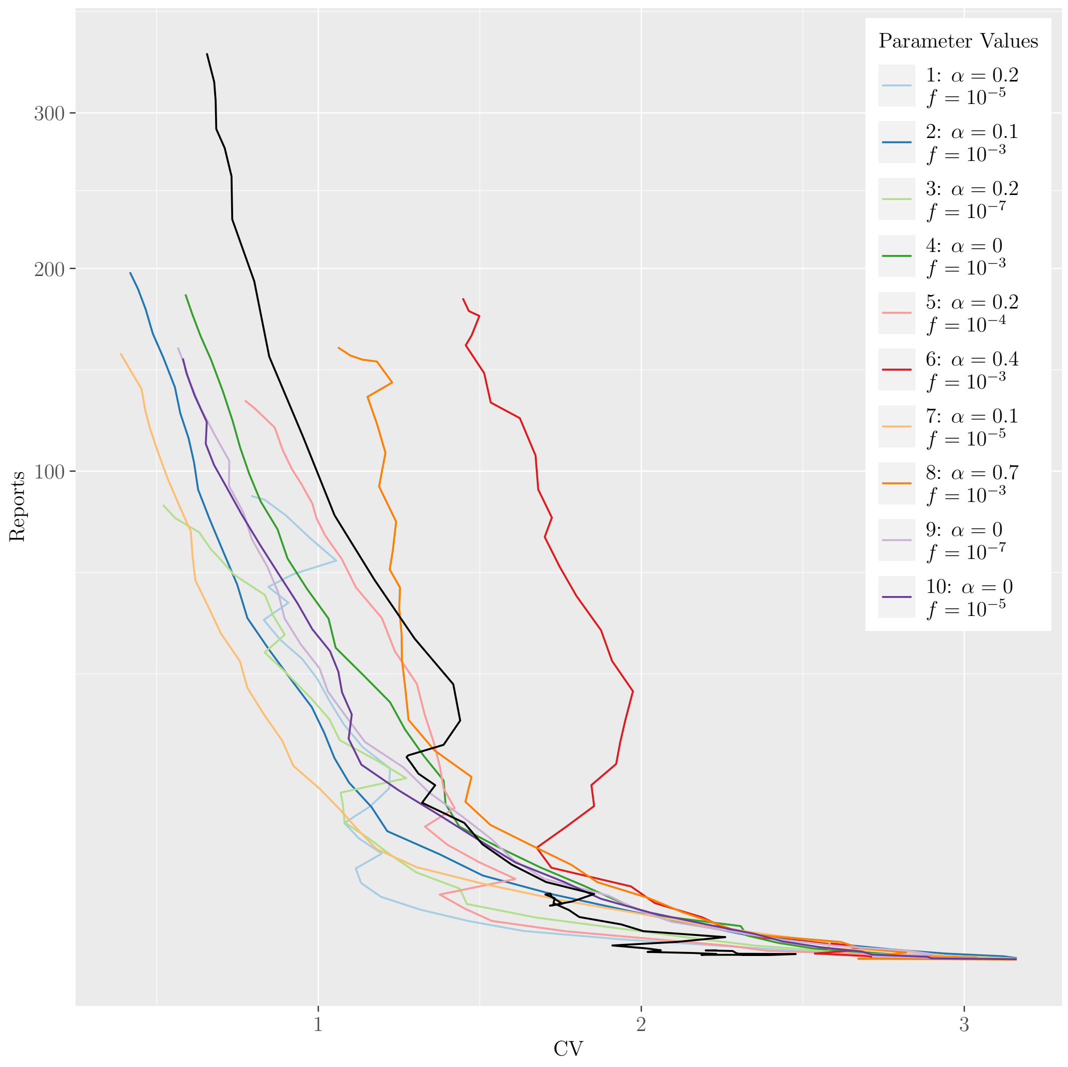


Figure 14: Paths corresponding to top 10 pairings of α and f. For the remaining analysis, we choose the following parameters: immigration factor, f, of 10^{-3} and α of 0.1 (table 1).

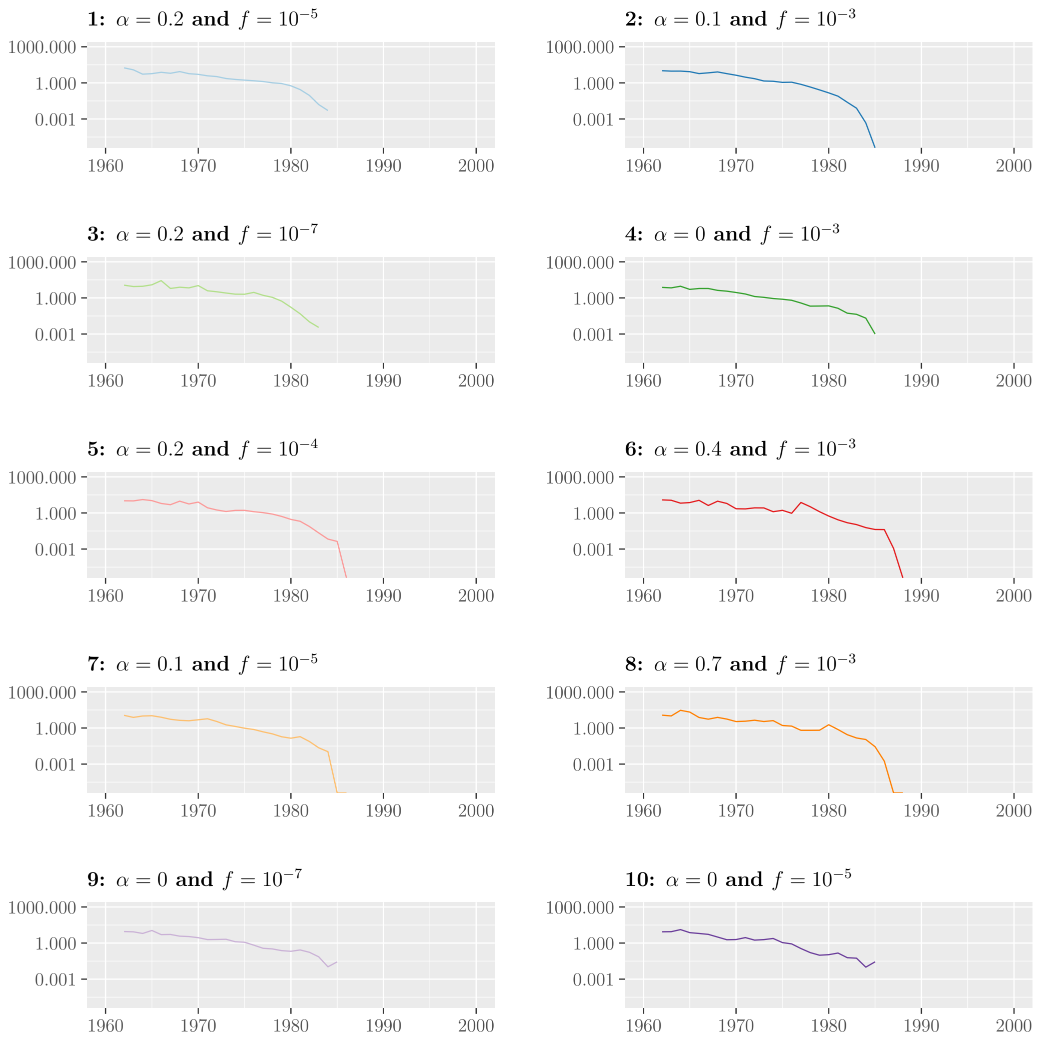


Figure 15: Time series corresponding to top 10 pairings of α and f. The colours of each time series correspond to the canonical paths in figure 14.

This process resulted in the optimal parameter pairing of: $\alpha = 0.2$ and $f = 10^{-5}$. However, in figure 13, it is evident that $f < 10^{-4}$ is not sufficient to prevent extinction for $\alpha > 0.2$. Since we wish to understand how the path changes for ranges of the amplitude of seasonal forcing, α, we must use $f > 10^{5}$. Thus, we choose to use the second best parameter pairing for our analysis: $\alpha = 0.1$ and $f = 10^{-3}$. The parameters used for the remaining analysis are presented in table 1.

Parameter List			
Parameter	**Value**	**Description**	**Source**
$\frac{1}{\gamma}$	5 days	mean infectious period	[36]
$\frac{1}{\sigma}$	8 days	mean latent period	[36]
$\mathcal{R}_0$	17	basic reproduction number	[36] [37]
α	0.1	amplitude of seasonal forcing	optimized (section 4.2)
initial run	200 years	transient period	chosen practically
f	10^{-3}	immigration factor	optimized (section 4.2)

Table 1: Values of model parameters used in simulations.

5 Dependence of the Simulated Path on Disease Characteristics

The creation of a model that well represents the US path opened up the potential to understand how the canonical path depends on properties of the disease and environmental conditions. Changing parameters such as the mean infectious period, $\frac{1}{\gamma}$, the mean latent period, $\frac{1}{\sigma}$, or the amplitude of seasonal forcing, α, allows us to explore what the path could look like for diseases with similar seasonality, but different natural history of infection, or in regions where the pattern of seasonal forcing may be different. The parameter values used in the simulations are either those obtained through optimization, values obtained from literature, or practical values (table 1).

5.1 Canonical Paths with Varying Mean Infectious and Latent Periods

As the mean infectious and latent periods vary across infectious diseases, understanding how the path changes based on these parameters could provide insight on what the path may look like for a disease with similar seasonality, but different history of infection. First, we simulated paths for the following mean infectious periods: 2, 4, 8, 16, and 32 days (figure 16).

Since we have fixed $\mathcal{R}_0$ to be 17, changing the mean infectious period changes the transmission rate. For mean infectious periods of 2, 4, and 8 days, we observe a similar trajectory to that of measles; the path is relatively smooth until we approach the elimination era. However, increasing the mean infectious period further results in a more jagged path. Additionally, we show the time series corresponding to each value of the mean infectious period in figure 17.

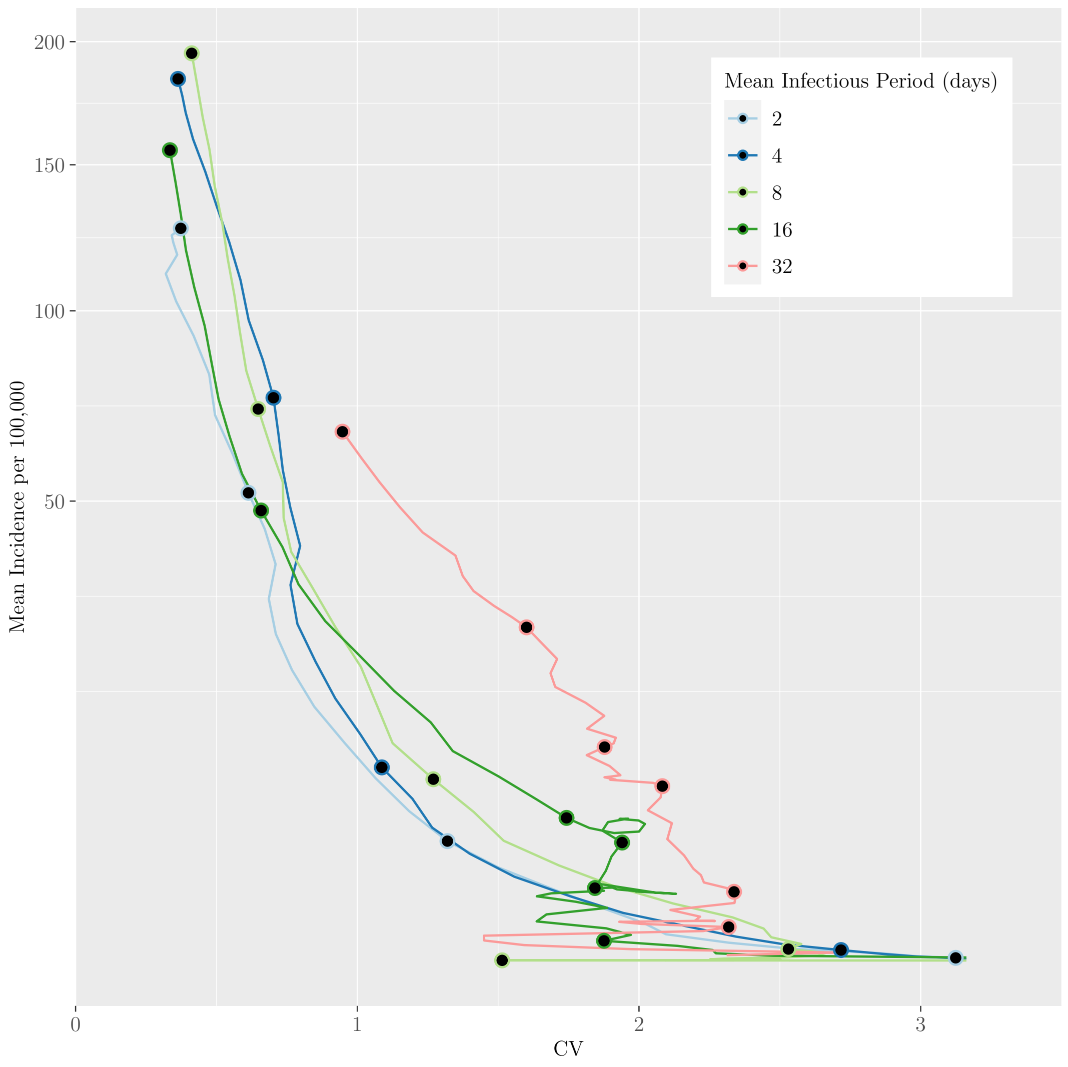


Figure 16: Results of simulating the US path for mean infectious period ranging from 2 to 32 days. The paths for mean infectious periods of 2, 4, and 8 days exhibit a similar trajectory of the path for measles (mean infectious period of 5 days [36]). However, increasing the mean infectious period further results in more jagged paths.

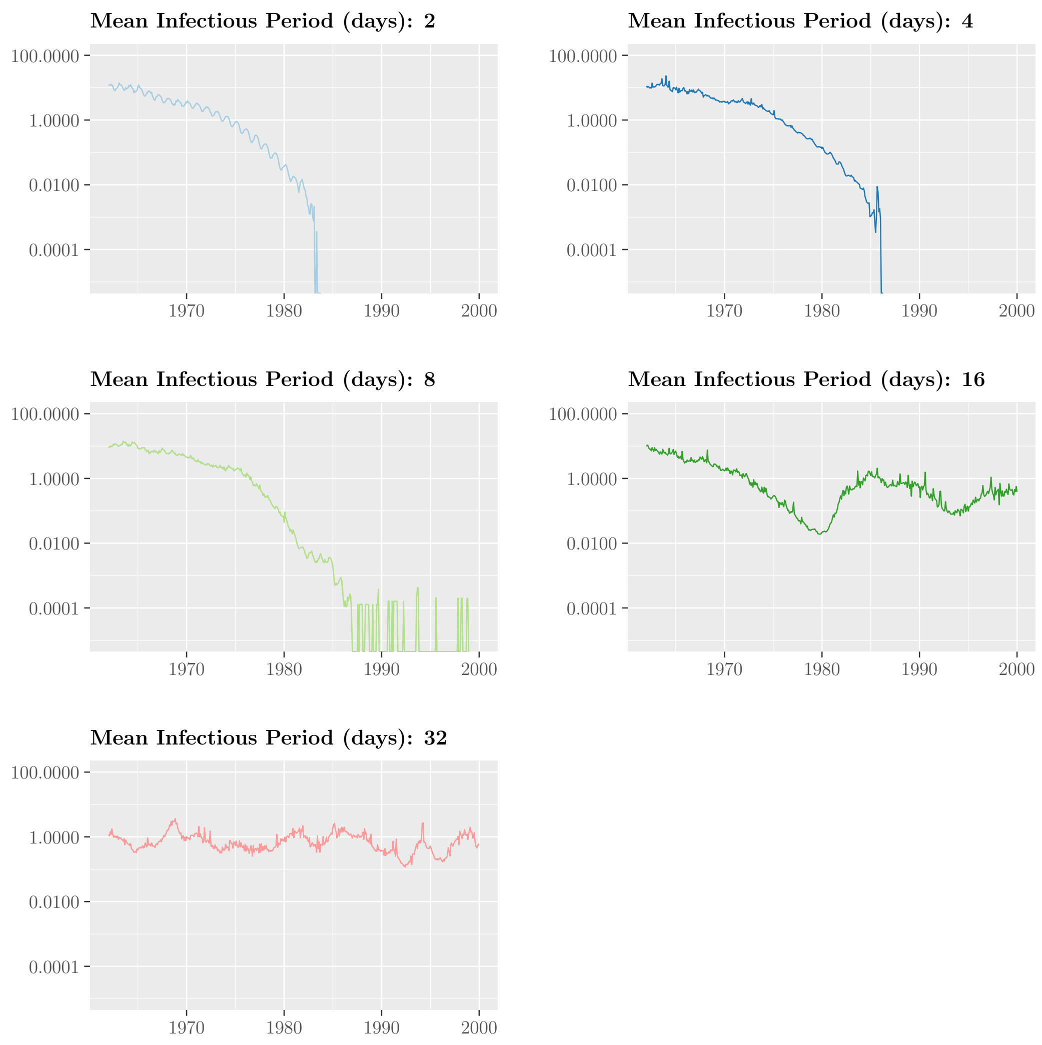


Figure 17: Time series corresponding to paths created using the following values of the mean infectious period: 2, 4, 8, . . . , 32. The colours of each time series correspond to the canonical paths in figure 16.

Next, we simulated paths for mean latent periods in the same range (figure 18). The time series of each path is presented in figure 19. When we compare the simulated paths for ranging mean latent period, we see more similarity among the paths. These results suggest that if an infectious disease, which only differs based on the natural history of infection, has a mean infectious or latent period only differing by a couple days, it is likely that its path to elimination will be similar. However, for a disease like whooping cough that has a similar $\mathcal{R}_0$ as measles but much longer infectious period [38], we would expect to see more jagged paths to elimination.

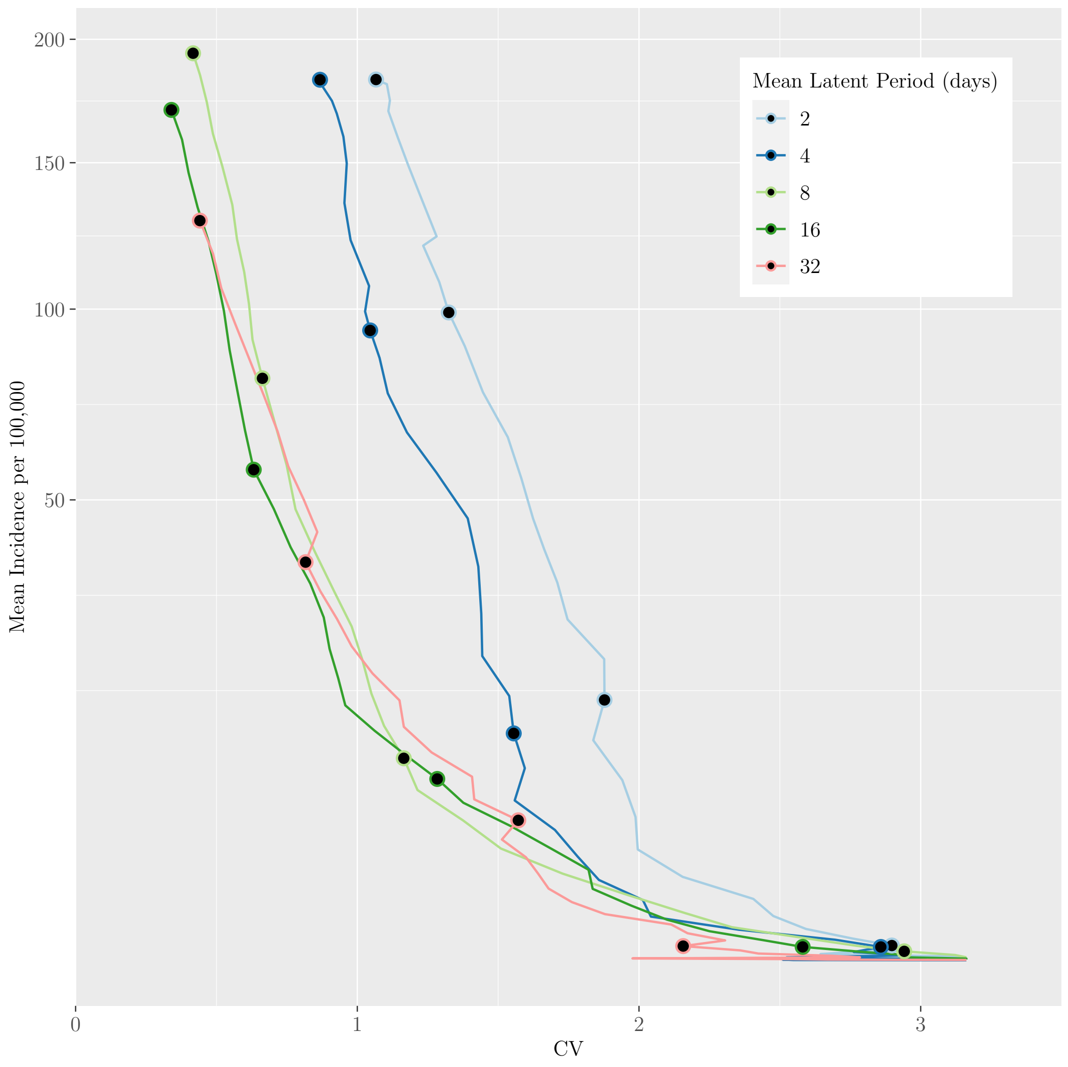


Figure 18: Results of simulating the US path for mean latent period ranging from 2 to 32 days. In comparing the simulations for different mean latent periods we see that the paths are very similar, with smooth trajectories to elimination.

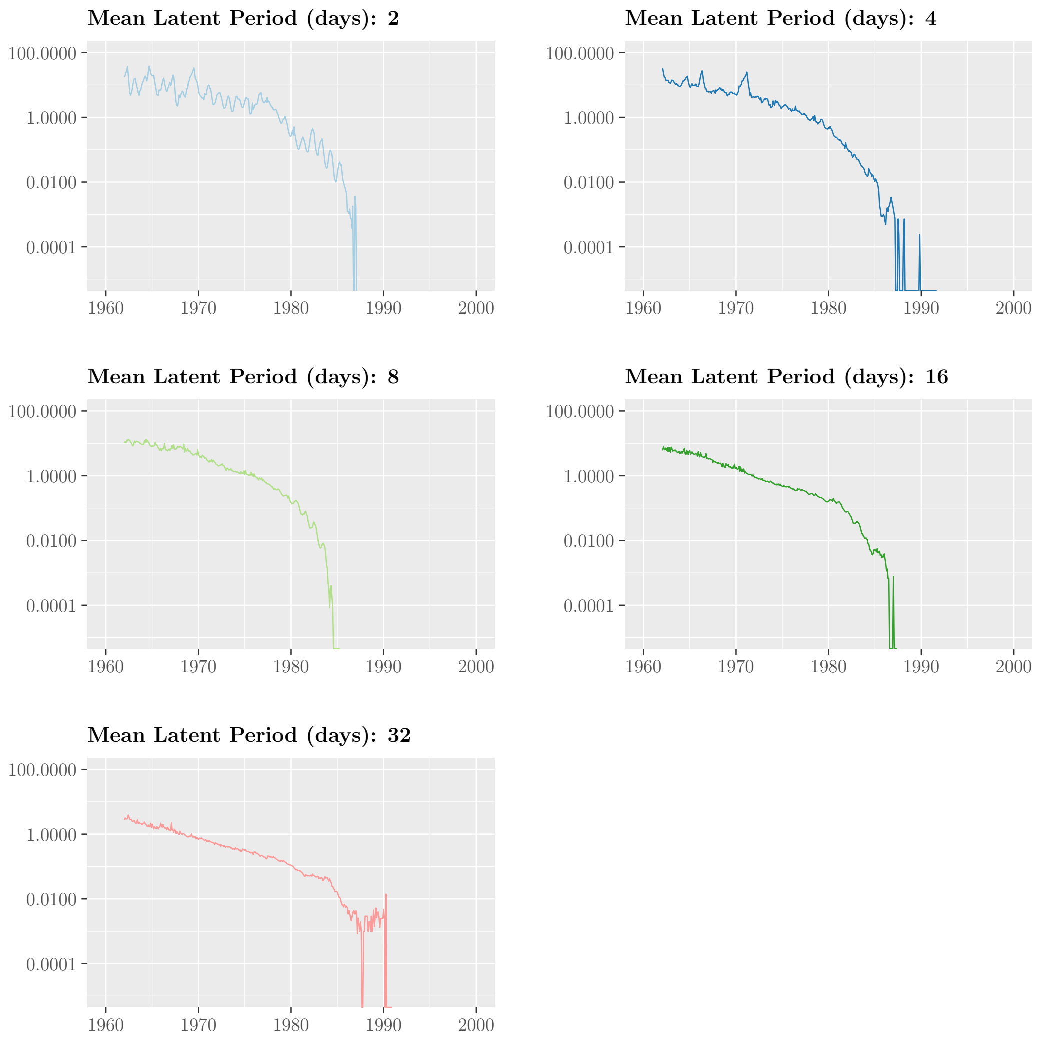


Figure 19: Time series corresponding to paths created using the following values of the mean latent period: 2, 4, 8, . . . , 32. The colours of each time series correspond to the canonical paths in figure 18.

5.2 Canonical Paths with Varying Seasonality

Finally, we wished to determine how changing the amplitude of seasonal forcing, α, affects the shape of the path. As the seasonal amplitude is increased from 0 to 1, the SEIR model displays a sequence of bifurcations and a wide range of behaviours, ranging from an annual cycle for α close to 0, to multiple co-existing stable cycles of various lengths, to chaos (e.g., Earn [36, Figure 11]). We capture most behaviours by considering $\alpha = 0, 0.1, 0.2, \ldots, 0.8$. Figure 20 shows the paths simulated for the specified values of α. To determine which behaviours are observed in our model with these values of α, we simulated the deterministic model for 2000 years from a single set of initial conditions to determine if the time series converged to stable cycle, or didn't and either experiences chaos or multiple co-existing cycles. To improve our ability to detect the presence of stable cycles, we would need to consider a grid of initial conditions. This could be conducted in future work. Details can be found in section 7.6, however the paths are labeled accordingly in figure 20. The time series of each path is presented in figure 21.

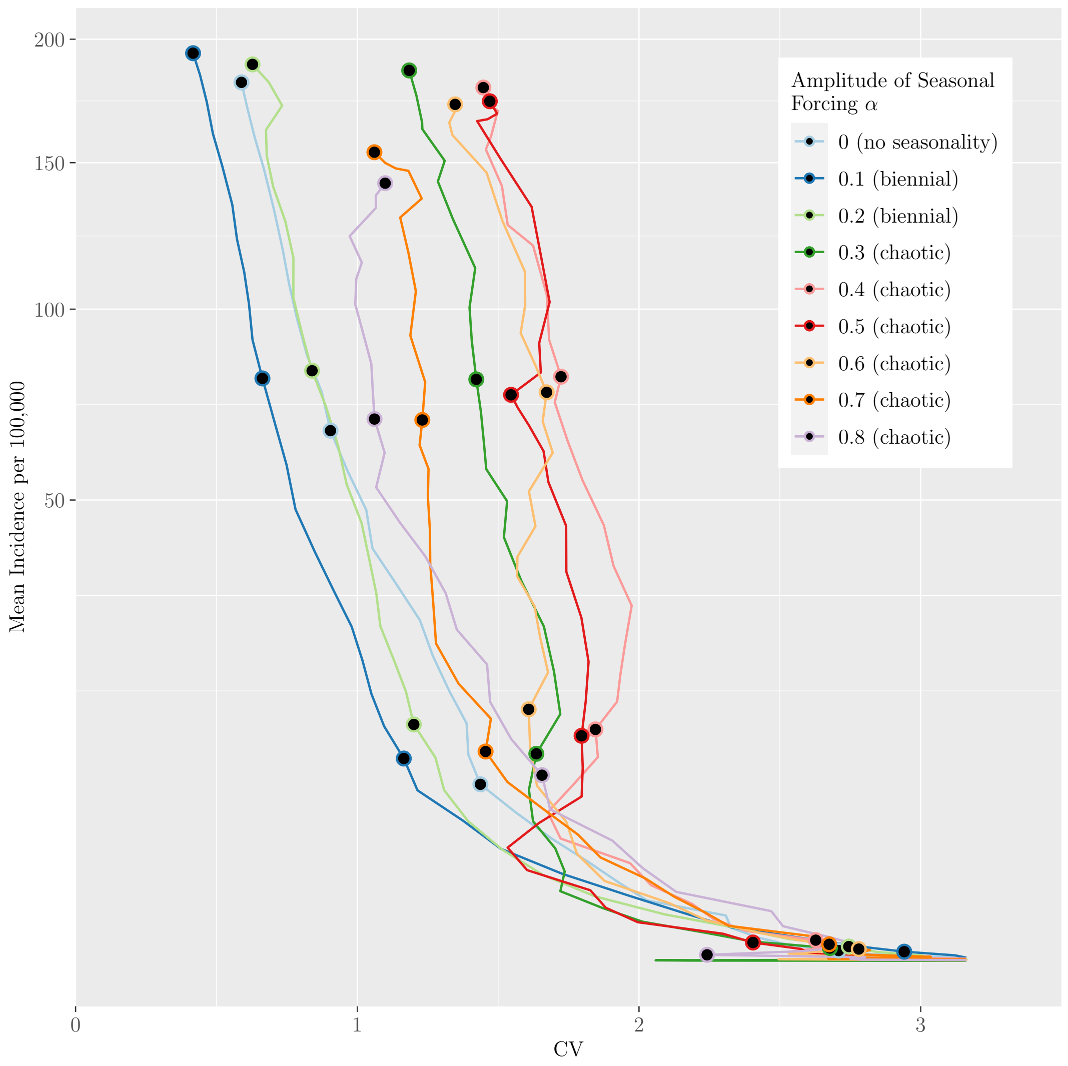


Figure 20: Canonical paths simulated for the following amplitudes of seasonal forcing, α: 0 to 0.8 in increments of 0.1. The legend describes both the value of α and the primary stable cycle or chaotic behaviour estimated from the deterministic models in section 7.6. This figure shows that paths created with α of 0 through 0.2, follow very similar trajectories, however increasing α further increases the CV of incidence.

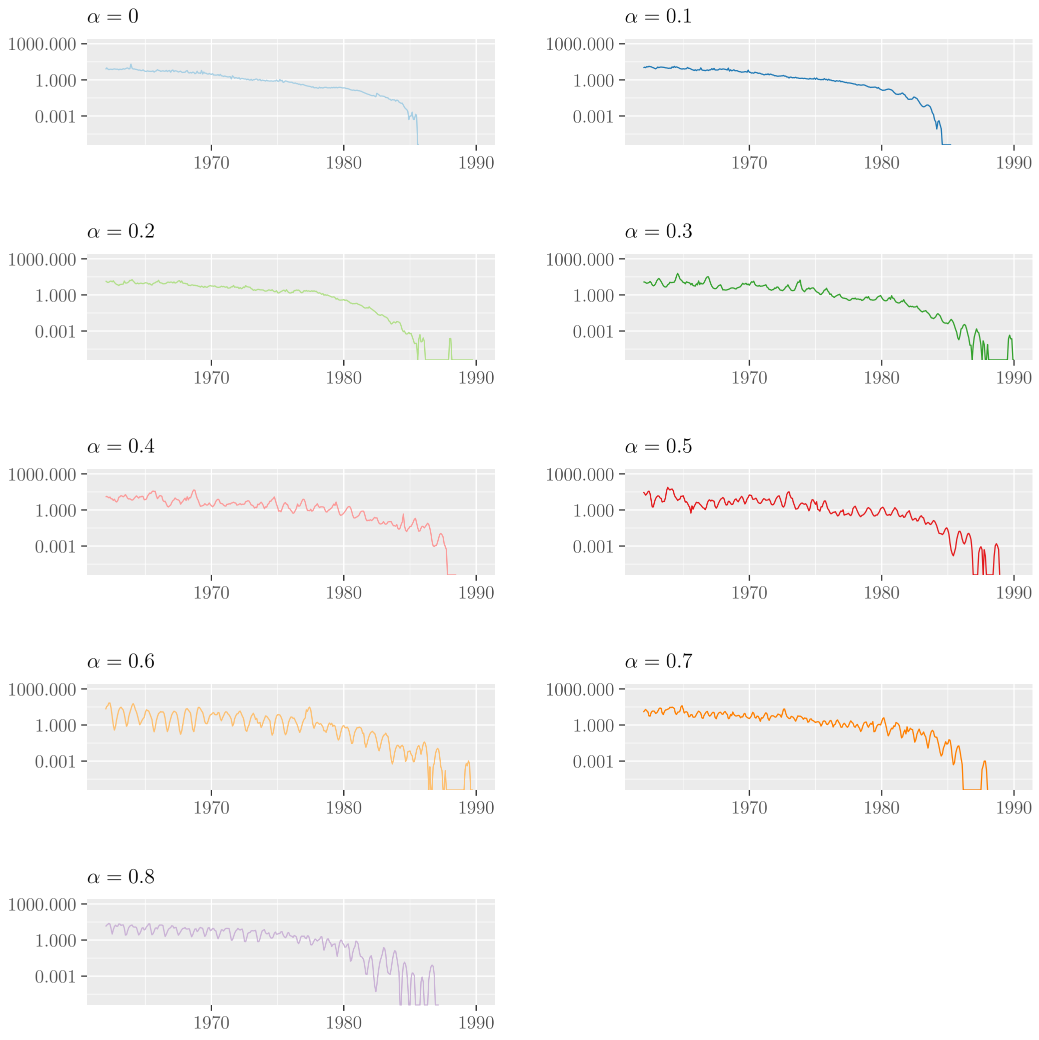


Figure 21: Time series corresponding to paths created using the following values of the α: 0 to 0.8 in increments of 0.1. The colours of each time series correspond to the canonical paths in figure 20.

The figure shows that for α values of 0 to 0.2, the trajectory is very similar, however increasing α further increases the CV. Interestingly, although the paths represent different levels of seasonality, changing the amplitude of seasonal forcing is only changing the CV of incidence. These results suggest that countries with either shorter stable cycles or chaotic dynamics follow a smoother trajectory, similar to that of the US path, although, countries with chaotic dynamics, like in Sub-Saharan Africa between 1986 and 2002 [39], will have increased CV of incidence due to the lack of a stable, periodic cycle.

6 Discussion

The goal of this thesis was to better understand the use of a canonical path to track the elimination of measles and its potential use for other eradicable diseases on a country-level scale by building on preliminary work conducted for an undergraduate thesis [14]. We began by creating the canonical path to the elimination of measles using the methods outlined by Graham *et al.* [1]. To understand to what degree individual states deviate from the path, we compared each state to the aggregate US path between 1960 and 2016 using the distance defined in equation 1. We identified Virginia and North Dakota as the states most and least similar to the US path respectively in section 2.3($d = 0.183$ and $d = 0.509$). Additionally, for the following years: 1962, 1972, . . . , 2012, we projected each state onto the aggregate path to understand whether states are progressing towards elimination at different speeds (Section 2.4). We compared the progression of Republican and Democratic stronghold states, and found that there was very little difference in progression across these groups (Section 2.5). Furthermore, in creating a box plot of the pointwise distance between each USstate and the aggregate path over time, we found that from 1960 to 2016, the median and variation of distance increased. This is exactly as expected, since as we approach elimination, there is increased variation in incidence, resulting in less similarity between the states and the aggregate path.

Although these results suggest that individual states progress towards the elimination of measles at similar speeds, this thesis demonstrated that we can successfully use the method developed by Graham *et al.* [1] on a country-level scale. Using this method for countries that have not yet eliminated measles, but may be close, has the potential to identify regions within the country that could be slowing the country's progression towards elimination. This information would allow the country to adjust their elimination strategy to target specific region(s), in turn improving their progression towards the elimination of measles.

In order to make inferences from the path, we conducted a sensitivity analysis to determine how sensitive the path is to changes in its definition. Specifically, we investigated how changing the weighting methods of incidence and CV, as well as the shape of the Gaussian filter used in weighting, changes the path. We discovered that we could use either a simple standard moving average or a linear Gaussian filter, as chosen by Graham *et al.* [1], to weight incidence without greatly affecting the shape of the path in section 3.1. Additionally, if we use the linear Gaussian filter to remain consistent with the methods defined by Graham *et al.* [1], changing the location of the centre or the standard deviation of the Gaussian filter has little impact on the shape of the path (Section 3.2). We do observe systematic changes in the level of weighted incidence, however this is explained by the fact that prior to vaccination, cases were lower than that of the present. Thus, weighting the past or the tails more heavily will increase weighted incidence. The opposite is observed after vaccination. These results suggest that the path and any inferences made are not sensitive to the choice of weighting method for incidence. Although, when we changed the weighting method for CV, we observed drastic changes in the path due to over-smoothing in section 3.3. Thus, even though our inferences will not be greatly impacted by changes in the weighting of incidence, we must choose a suitable weighting method for CV to prevent over-smoothing, which can affect inferences by hiding true variation in the path.

Another core element of this thesis was not only to simulate the aggregate US path, but also to understand which elements of a model are necessary to accurately predict a country's path. For example, do we need to implement complex components like age structure or additional epidemiological compartments, as implemented by Graham *et al.* [1]? Or is a basic SEIR model, (Equation 2), sufficient? We started with a base SEIR model, and built it up one component at a time until we achieved a model that accurately reproduced the aggregate US path. We found that our model needed to have the following time-varying structures to well represent the aggregate US path: transmission, vital dynamics, and vaccination. Additionally, we needed to incorporate a crude form of immigration to prevent extinction in states with a population below the critical community size for persistence in the model (~ 1 million). Furthermore, since the US path is constructed using reported case data rather than incidence, we implemented observation error to account for under-reporting of measles. Finally, we chose to run for extra time at the initial conditions to ensure that we converged to an attractor. (Section 4.1)

In qualitatively comparing our model to that used by Graham *et al.* [1], we found we were able to use a simpler model to simulate the UScanonical path. However, to determine which model provides better prediction on a country-level scale, further research that implements our model and the model used by Graham *et al.* [1] for the same country, *e.g.* needs to be conducted. Regardless of which model provides better prediction, an important take-away from the process of building up the model is that we must have a robust understanding of the characteristics of a disease. This suggests that this method is most useful for eradicable diseases for which we have data on the natural history of infection and vaccination. (Section 4.1.5)

The final component of this thesis aimed to predict what the path may look like for a disease with similar $\mathcal{R}_0$ of measles, but with different natural history of infection or seasonality. In simulating canonical paths with the following mean infectious periods, 2, 4, 8, . . . , 32 days, we discovered that the paths with mean infectious periods similar to that of measles experienced a smooth trajectory towards elimination. However, increasing the mean infectious period past 8 days results in the paths being more jagged. However, when considering mean latent periods in the same range, we observed little difference between the paths. The results of changing the natural history of infection suggest that if a disease has similar characteristics and natural history of infection varying only by a few days, it will experience a smooth path to elimination, like that of measles. However, for a disease like whooping cough, that has a similar $\mathcal{R}_0$ but much longer mean infectious period, the path will be more jagged (Section 5.1). Furthermore, when the amplitude of seasonal forcing is changed, and the dynamics move from short, stable cycles, to chaos, we observe that paths corresponding to chaos tend to have higher CV of incidence. This suggests that all countries will follow a similarly smooth path to elimination, but those with chaotic dynamics, like in Sub-Saharan Africa between 1986 and 2002 [39], will have increased CV of incidence (Section 5.2).

6.1 Future Directions and Limitations

The analysis conducted in this paper has provided important insight on the sensitivity of the canonical path and provides examples of how its trajectory changes when comparing ranges of key measles parameters. Although, there are limitations of this study. For example, the data used to represent time-varying vaccination assumes that the vaccination coverage is the same across all USstates. This could be ameliorated by cleaning the state-level vaccination data from 1995 to 2019 obtained from the National Immunization Surveys and collecting vaccine coverage data going back to the introduction of a measles vaccine[4] [29]. Additionally, we discovered that there are inconsistencies in the reporting of measles across the USstates before 1960. Although we were able to crudely estimate the approximate level of under-reporting to implement observation error in our model, we need to collect births data going back to the year in which measles became nationally notifiable and collect age-structured incidence data. This would allow us to define a time series of reporting rates across states, and correct for under-reporting when creating the canonical path for the USusing a more realistic method as used by Fine and Clarkson [34, 35]. Furthermore, our data collection was impacted by the COVID-19 pandemic. One of the initial goals of the paper was to compare the UScanonical path, to one created for Canada, in order to explore similarities and/or differences between countries. However, as the current focus is on COVID-19 research, collecting historical data for measles cases from each province became very time-consuming and got us no where. Collecting this data in order to compare the USand Canadian paths would be an interesting topic for future research.

One could also collect data for countries that exhibit different measles seasonality, to

[4]We reached out to the Centers for Disease Control and Prevention (CDC) Advisory Committee on Immunization Practices and they provided a link to their archives. We were not able to find data through that source, however they suggested that we submit an Freedom of Information Act (FOIA) request to get the data we need. Although this was not feasible for our report due to time constraints. [40, 41]

determine if this corroborates the findings that countries with different seasonality than that of measles have increased CV of incidence in their paths towards elimination. Another topic of future research is to create paths for other diseases that have been eliminated, i.e., smallpox, polio etc., and compare the paths to that of measles. Insights from the comparisons could inform whether this method of determining the path to elimination of a disease could be used for diseases we have yet to eliminate. The ability to use this method for a disease we wish to eliminate could allow countries to determine which elimination strategies would have the greatest impact on the trajectory of the path. We could also use the model defined by Graham *et al.* [1] that simulates the canonical path using a state-space compartmental model, to determine which model provides the best representation of the US canonical path. Finally, the canonical path created by Graham *et al.* [1] starts at small CV, increases in CV, and then returns to small CV as elimination is approached. Thus, the CV is not a one-to-one function of mean incidence and therefore provides information beyond what can be obtained from a mean incidence time series alone. In our case, if the shape of the path observed before vaccination is accurate, then the canonical path provides more information in our case as well, as it is clearly a 2D path. However, since we identified that the structure in this time period is affected by underreporting, we would need to correct for underreporting before comparing the different methods.

6.2 Conclusion

As the elimination of diseases is of key interest, whether regarding the COVID-19 pandemic, or other diseases deemed eradicable by the WHO and the PAHO [8], being able to track a country's progress towards the elimination of diseases could prove useful. While Graham *et al.* [1] demonstrated that the global canonical path analysis has the potential to help countries improve their measles elimination strategies, this paper not only showed that inferences are not sensitive to the definition of the path, but also that the path can be used on a smaller scale to compare the individual USstates to the US path as a whole. Using this method on a smaller scale could help countries determine which of the smaller regions are falling behind, and change elimination strategies accordingly. Our analysis also shows how the path changes based on aspects of measles, such as seasonality. Furthermore, the simulation analysis suggests that we can simulate a theoretical path for an emerging disease based on plausible vaccination strategies and known data regarding the natural history of infection. However, this is only possible for a disease with high vaccine efficacy and long-term immunity, otherwise it is unlikely to be eliminated. The more we use this method for diseases we wish to control or those already eliminated, we may learn more about strategies that are most useful in eliminating diseases, in turn improving elimination programs in the future.

References

[1] Graham M, Winter AK, Ferrari M, Grenfell B, Moss WJ, Azman AS, et al. Measles and the canonical path to elimination. Science. 2019;364(6440):584-7.

[2] Dowdle WR. The Principles of Disease Elimination and Eradication; 1999. https://www.cdc.gov/mmwr/preview/mmwrhtml/su48a7.htm?fbclid=IwAR0xiRW8KoTAQzNQPpSdXBhnASlYf3p49YfcKK2dmAt0817Hafy6pyEezIE.

[3] Corona A. Disease Eradication: What Does It Take to Wipe out a Disease?; 2020. https://asm.org/Articles/2020/March/Disease-Eradication-What-Does-It-Take-to-Wipe-out.

[4] Centers for Disease Control and Prevention. Lesson 1: Introduction to Epidemiology; 2012. https://www.cdc.gov/csels/dsepd/ss1978/lesson1/section11.html.

[5] Centers for Disease Control and Prevention. Measles Elimination; 2020. https://www.cdc.gov/measles/elimination.html.

[6] Centers for Disease Control and Prevention. History of Measles; 2020. https://www.cdc.gov/measles/about/history.html.

[7] The World Health Organization. Measles; 2019. https://www.who.int/news-room/fact-sheets/detail/measles.

[8] Pan American Health Organization. Elimination Initiative: Towards healthier generations free of diseases; 2019. https://www.paho.org/en/elimination-initiative-towards-healthier-generations-free-diseases.

[9] Roser M, Ochmann S, Behrens H, Ritchie H, Dadonaite B. Eradication of Diseases; 2014. https://ourworldindata.org/eradication-of-diseases.

[10] Centers for Disease Control and Prevention. CDC - Guinea Worm Disease - Eradication Program 2019; 2019. https://www.cdc.gov/parasites/guineaworm/gwep.html.

[11] Sencer DJ, Dull HB, Langmuir AD. Epidemiologic basis for eradication of measles in 1967. Public health reports. 1967;82(3):253.

[12] Moss WJ, Griffin DE. Global measles elimination. Nature Reviews Microbiology. 2006;4(12):900-8.

[13] Measles & Rubella Initiative. Measles & Rubella Strategic Framework 2021-2030; 2021. https://measlesrubellainitiative.org/measles-rubella-strategic-framework-2021-2030/.

[14] O'Meara E. Determining the Effectiveness of Using a "Canonical Path" to Understand the Factors that Lead to the Elimination of Measles in the United States; 2020.

[15] MIDAS Informatics Services Group. Project Tycho; 2019. https://www.tycho.pitt.edu/.

[16] Centers for Disease Control and Prevention. Morbidity and Mortality Weekly Reports; 2019. https://www.cdc.gov/mmwr/mmwr_nd/index.html.

[17] Centers for Disease Control and Prevention. National Notifiable Diseases Surveillance System, 2019 Annual Tables of Infectious Disease Data; 2021. https://wonder.cdc.gov/nndss/nndss_annual_tables_menu.asp.

[18] rOpenSci. Extract Tables from PDFs; 2021. https://github.com/ropensci/tabulizer.

[19] Forstall RL. Population of States and Counties of the United States: 1790 to 1990. Washington, DC: U.S. Bureau of the Census; 1996.

[20] United States Census Bureau. State Intercensal Tabels: 2000-2010; 2021. https://www.census.gov/data/tables/time-series/demo/popest/intercensal-2000-2010-state.html.

[21] WorldAtlas. States That Have Consistently Remained Democratic; 2020. https://www.worldatlas.com/articles/states-that-have-consistently-remained-democratic.html.

[22] WorldAtlas. List of Red States (Republican States); 2019. https://www.worldatlas.com/articles/states-that-have-voted-republican-in-the-most-consecutive-u-s-presidential-el
html.

[23] WorldAtlas. Which States Are Swing States?; 2018. https://www.worldatlas.com/articles/which-states-are-swing-states.html.

[24] Ivory D, Leatherby L, Gebeloff R. Least Vaccinated U.S. Counties Have Something in Common: Trump Voters. The New York Times. 2021 Apr. Available from: https://www.nytimes.com/interactive/2021/04/17/us/vaccine-hesitancy-politics.html.

[25] Papst I, Earn DJD. Invariant predictions of epidemic patterns from radically different forms of seasonal forcing. Journal of the Royal Society Interface. 2019;16(156):20190202.

[26] The Wold Health Organization. Measles Vaccination Coverage; 2021. https://immunizationdata.who.int/pages/coverage/MCV.html?CODE=USA+CAN&ANTIGEN=MCV1&YEAR=.

[27] Bartlett MS. The critical community size for measles in the United States. Journal of the Royal Statistical Society: Series A (General). 1960;123(1):37-44.

[28] National Center for Immunization and Respiratory Diseases. Vaccination Coverage among Young Children (0-35 Months); 2021. https://data.cdc.gov/Child-Vaccinations/Vaccination-Coverage-among-Young-Children-0-35-Mon/fhky-rtsk.

[29] Centers for Disease Control and Prevention. NIS-Child Data and Documentation for 2015 to Present; 2020. https://www.cdc.gov/vaccines/imz-managers/nis/datasets.html.

[30] Centers for Disease Control and Prevention. National Center for Health Statistics Mortality Data on CDC WONDER; 2020. https://wonder.cdc.gov/mortSQL.html.

[31] Centers for Disease Control and Prevention. Natality Information; 2021. https://wonder.cdc.gov/natality.html.

[32] Centers for Disease Control and Prevention. National Vital Statistics Reports; 2022. https://www.cdc.gov/nchs/products/nvsr.htm.

[33] Centers for Disease Control and Prevention. Vital Statistics of the United States; 2015. https://www.cdc.gov/nchs/products/vsus.htm.

[34] Fine PE, Clarkson JA. Measles in England and Wales—I: an analysis of factors underlying seasonal patterns. International journal of epidemiology. 1982;11(1):5-14.

[35] Fine PE, Clarkson JA. Measles in England and Wales—II: the impact of the measles vaccination programme on the distribution of immunity in the population. International Journal of Epidemiology. 1982;11(1):15-25.

[36] Earn DJD. Mathematical epidemiology of infectious diseases. In: Lewis MA, Chaplain MAJ, Keener JP, Maini PK, editors. Mathematical Biology. vol. 14 of IAS/Park City Mathematics Series. American Mathematical Society; 2009. p. 151-86. Available from: http://www.ams.org/books/pcms/014/.

[37] Guerra FM, Bolotin S, Lim G, Heffernan J, Deeks SL, Li Y, et al. The basic reproduction number ($\mathcal{R}_0$) of measles: a systematic review. The Lancet Infectious Diseases. 2017;17(12):e420–e428.

[38] New York State Department of Health. Pertussis or Whooping Cough Fact Sheet; 2016. https://www.health.ny.gov/publications/2171/.

[39] Ferrari MJ, Grais RF, Bharti N, Conlan AJ, Bjørnstad ON, Wolfson LJ, et al. The dynamics of measles in sub-Saharan Africa. Nature. 2008;451(7179):679-84.

[40] Centers for Disease Control and Prevention. Advisory Committee on Immunization Practices Archives; 2022. https://stacks.cdc.gov/cbrowse?pid=cdc%3A56588&parentId=cdc%3A56588.

[41] Centers for Disease Control and Prevention. CDC FOIA Public Access Link-Home; 2022. https://foia.cdc.gov/app/Home.aspx.

[42] Gavin B. What Is a DAT File (and How Do I Open One)?; 2018. https://www.howtogeek.com/363326/what-is-a-dat-file-and-how-do-i-open-one/.

7 Supplementary Materials

7.1 Introduction

This supplement is written in `knitr`, compiled using R version 4.0.3 (2020-10-10) and uses the following R package versions:

```
## tidyverse    readxl       date      stats   smoother calibrate
##     1.3.0     1.3.1     1.2-39      4.0.3        1.1     1.7.7
## RColorBrewer   tikzDevice        rlist    gridExtra  adaptivetau
##        1.1-2     0.12.3.1      0.4.6.2          2.3        2.2-3
##   deSolve      knitr tabulizer      shiny    miniUI   ggrepel
##      1.28       1.36     0.2.2      1.6.0   0.1.1.1     0.9.1
```

The goal of this supplement is to ensure that the results of this report are reproducible.

7.2 Data Collection and Cleaning

The analysis conducted in this report required the use of several different data sets regarding measles incidence, population, vital statistics, and vaccination coverage. Sections 7.2.1 through 7.2.5 outline how the data is cleaned and prepared for use.

7.2.1 Reading Incidence Data and Population into R

For measles incidence data up to 2001 and United States population data, we read the data into R using the `read_excel()` function in the code below. We save the generated data frames to a *.RData* file using `save()`. This allows us to use the data sets later on, without having to re-input the data into R.

```
library(readxl)
library(knitr)
pop.data <- read_excel('Good_data/Pop_data.xlsx',col_names =FALSE)
measles.data <- read_excel('Good_data/US.14189004.xlsx')
pop.data.2000.2010 <-
  read_excel(
    'Good_data/intercensal_pop_US_state_2000-2010_CensusBureau.xls'
    )
```

```
save(pop.data,measles.data,pop.data.2000.2010,
     file='Good_data/alldata.RData')
```

Since we have measles incidence data for the years 2000 to 2016 as MMWR reports in *.pdf* format, reading the data from these files is more complex. To extract tables from pages in *.pdf* files, we use either the `extract_tables()` or the `extract_areas()` functions from the `tabulizer` package. The difference between the two functions is that `extract_tables()` either guesses the location of a table on a specific set of pages or it allows you to provide values telling it where the table lies, while `extract_areas()` allows the user to highlight the specific area of the chosen pages that you wish to extract interactively using the shiny app that is opened when calling the function. Due to the nature of the `extract_area()` function, this code cannot be run in this document. Instead it is run in `pdftoexcel.R` and saves the data to `Good_data/measles.2000.2016.RData`. The cleaning of this data is discussed in section 7.2.3.2.

7.2.2 Population

As mentioned in section 2.1, we combine two separate data sets to obtain the annual population estimates of each US state from 1912 to 2016. We begin cleaning the data set containing annual population estimates for each US state from 2000 to 2010, `pop.data.2000.2010`, by removing the population estimates for the entirety of the US and the aggregate population estimates for the northeast, midwest, south, and west regions. Additionally, for ease, we name the column containing the US state names "State". We also remove the first two rows of the data set as they are irrelevant rows generated when reading in the file. Next, we fix the formatting of the US state names as each begins with a ".". This is accomplished using the `substr()` function where we pass the argument `start`$=2$ so that each entry in the "State" column is rewritten starting with the second character and ending with the last character, where the number of characters in the string is found using `nchar`.

```
library(tidyverse)

load("Good_data/alldata.RData")

## Remove rows corresponding to aggregate population
## values and irrelevant rows introduced due to
## formatting when loading the data
pop.data.2000.2010 <- pop.data.2000.2010[3:59, ]
colnames(pop.data.2000.2010) <- c("State", c(2:14))
```

```
remove <- c("United States", "Northeast", "Midwest", "South",
    "West")

pop.data.2000.2010 <- pop.data.2000.2010 %>%
    filter(!(State %in% remove))

## Fix the state name formatting by removing the first
## character (as is each begins with a '.')
for (i in 1:nrow(pop.data.2000.2010)) {
    pop.data.2000.2010$State[i] <- substr(pop.data.2000.2010$State[i],
        2, nchar(pop.data.2000.2010$State[i]))
}
```

With the "State" column properly formatted, we also need to get rid of the columns we do not need. For example, the data set has two entries for each of 2000 and 2010, where one is the census count as of April 1st and the other is an intercensal estimate as of July 1st. For consistency with the other intercensal estimates from 2001 to 2009, we choose to use the July 1st estimates instead of the April 1st counts. Thus, we name the columns corresponding to the years in which we want data with 2000 through 2010, and the others are left alone. This allows us to remove the columns that are not the "State" column or one corresponding to the years of data we need. Additionally, we use `na.omit()` to remove the first row that has `NA` entries due to formatting when reading the data into R. Additionally, we change the US state names to be all capital letters to match the other population data set using the `toupper()` function.

```
## Remove th columns containing data we do not need,
## e.g., in 2000 and 2010 there is a census count and
## a popualtion estimate. Remove the count for
## consistency
colnames(pop.data.2000.2010) <- c("State", 2, c(2000:2009),
    13, 2010)
pop.data.2000.2010 <- na.omit(pop.data.2000.2010)
index <- colnames(pop.data.2000.2010) %in% c("State", c(2000:2010))
pop.data.2000.2010 <- pop.data.2000.2010[, index]

## Change state names to uppercase for consistency
## with other data frames
pop.data.2000.2010$State <- toupper(pop.data.2000.2010$State)
```

Now that the population data from 2000 to 2010 is cleaned, we combine this with the census data from 1900 to 1990 (`pop.data`). First, we create an empty data frame that has the number of rows corresponding to the number of US states and columns for the years 1900 to 2010, where the row names are the US states and the column names are the years. Note that the census count data set has the columns organized based on descending years (1990 to 1890). For simplicity we arrange this data frame the same way, i.e., the columns are named 2010, 2000, 1990, ..., 1900. The population estimates for 2000 and 2010 are obtained from the intercensal estimates data set and entered in the data frame accordingly. The remaining data is provided by the census counts data set.

```
## Create an empty data frame with a row for each
## state and a column for each census year from 1900
## to 2010. Name the rows and columns accordingly.
nstates <- 51
year <- seq(2010, 1900, -10)

pop.data.all <- as.data.frame(matrix(NA, nrow = nstates,
    ncol = length(year)))
state.names <- unname(unlist(pop.data[(1:nstates) + 4, 1]))
state.names <- toupper(state.names)

colnames(pop.data.all) <- year
rownames(pop.data.all) <- state.names

## Assign the data for 2000 and 2010 to the
## corresponding columns, the remaining columns
## obtains the data from the data frame containing
## census counts from 1890 to 1990
pop.data.all[, "2010"] <- pop.data.2000.2010[, "2010"]
pop.data.all[, "2000"] <- pop.data.2000.2010[, "2000"]

pop.data.all[, 3:12] <- pop.data[5:55, 2:11]
```

Since, for the majority of the data, we only have census counts every 10 years, we can linearly interpolate in between census years to estimate the annual population by state. This is accomplished using `approx()` with the several arguments. The first arguments are the original time values (1900, 1910, ..., 2010) and the population in each state for each of those years. We also provide the function the number of total data points we want to end up with (`n`), i.e., 111 years so that we have annual population from 1900 to 2010. Finally we provide it the additional argument `rule`. If `rule` is 1 then the estimate for a time greater than 2010 or less than 1900 are returned as `NA`, if `rule` is 0 then the estimate is the closest extreme, i.e, either the population estimate for 1900 or 2010.

```r
## Interpolate the population between censuses using
## the 'approx' function. years_inter is the resulting
## years we wish to have after interpolation
years_inter <- c(1900:2010)

pop.estimates.inter <- as.data.frame(matrix(NA, nrow = nstates,
    ncol = length(years_inter)))

for (i in 1:nstates) {
    approx.out <- approx(year, pop.data.all[i, ], method = "linear",
        n = length(years_inter), rule = 1)
    pop.estimates.inter[i, ] <- approx.out$y
}
colnames(pop.estimates.inter) <- approx.out$x
rownames(pop.estimates.inter) <- rownames(pop.data.all)
```

Unfortunately, this function does not have the capability to extrapolate. So to obtain population estimates for 2011 to 2020, we use a linear model built on the data from 2001 to 2010, thus continuing at approximately the most recent observed slope. Before we use a linear model, we create a column titled "State". Its entries are the row names of the data set. Then we transform our data from wide-format to long-format using `pivot_longer`. This results in a data frame with the following columns: State, Year and Population. To be of the proper format for the `lm()` function, we change the structure of the "Year" column from character to numeric using `as.numeric()`. Finally, we use the `lm()` function to estimate the population in each year from 2011 to 2020 for each US state and sort the resulting data set by State. We save both this data set and one created by transforming it back to wide format using `pivot_wider()`. We also save the vector `state.names` for use in later code scripts.

```r
## Use pivot_longer to get the long version of the
## data with columns state, year and population
pop.estimates.inter$State <- rownames(pop.estimates.inter)

long_pop_estimates <- pivot_longer(pop.estimates.inter, cols = !State,
                                   names_to = "Year",
                                   values_to = "Population")

long_pop_estimates$Year <- as.numeric(long_pop_estimates$Year)

## Vector of years we need to extrapolate a population with lm
```

```r
years_new <- c(2011:2020)

## For each state I use lm to build a model of
## Population ~ year then use it to predict the missing years,
## then add it to the long population data frame
for (ii in state.names) {
  filter_pop <- long_pop_estimates %>%
    filter(State == ii & Year %in% c(2000:2010))

  my_model <- lm(Population ~ Year, data = filter_pop)
  pred_model <- predict(
    my_model,
    newdata = data.frame(Year = years_new)
    )

  new_data <- data.frame(
    State = rep(ii, length(years_new)),
    Year = years_new,
    Population = pred_model
  )

  long_pop_estimates <- rbind(long_pop_estimates, new_data)
}

## Sort the population dataframe by state
long_pop_estimates_final <- long_pop_estimates %>%
  arrange(State)

## Use pivot_wider to convert it back to wide format
pop.estimates <- pivot_wider(
  long_pop_estimates_final,
  names_from = "Year",
  values_from = "Population")

state.names <- pop.estimates$State
pop.estimates <- pop.estimates[,-1]
rownames(pop.estimates) <- state.names

pop.estimates <- as.data.frame(pop.estimates)

save(pop.estimates, nstates, state.names, long_pop_estimates_final,
     file = 'Good_data/pop.estimates.RData')
```

```r
## Vector of US states and years of pop data
state.names <- unique(long_pop_estimates$State)
years.pop <- unique(long_pop_estimates$Year)

## Create a dataframe of US population each year (not by state)
US_pop <- as.data.frame(matrix(ncol = 2, nrow = length(years.pop)))
colnames(US_pop) <- c("Year", "Population")
US_pop$Year <- years.pop


for (jj in years.pop) {
  filtered <- long_pop_estimates %>%
    filter(Year == jj)

  US_pop$Population[US_pop$Year == jj] <-
    sum(filtered$Population, na.rm = TRUE)

}

save(US_pop, file = "Good_data/US_population_data.RData")
```

7.2.3 Yearly Incidence per 100,000

7.2.3.1 Weekly Incidence per 100,000 from 1900 to 1999

With all of the required incidence data loaded and a cleaned data set containing annual population estimates by US state from 1900 to 2020, we begin creating a data set containing yearly incidence per 100,000 individuals. First, we create a data set containing weekly incidence per 100,000 using the data set containing incidence from 1900 to 2001, `measles.data`. As is, the data set contains weekly measles incidence for all US states, the entirety of the US and for US territories such as the Virgin Islands, Puerto Rico, *etc.* It also contains data for fatalities and cumulative counts. Thus, we filter the data set to only provide weekly measles counts for each of the US states, including District of Columbia. Next, we change the format of the period end date column to provide only the year, instead of the day, month and year. This will allow us to take the sum of all weeks in a year to find the aggregate measles counts in a given year in later code. For now, we create a column titled "per.100k". Its entry is the count for a given week and US state, multiplied by 100,000 and divided by the population for the year in which the week ends. This data set is saved for later use.

Code to follow.

```r
library(tidyverse)
library(date)
library(knitr)
load('Good_data/alldata.RData')
load('Good_data/pop.estimates.RData')

## Add column for cases per 100k
filtered<- measles.data %>%
  filter(
    !Admin1Name %in% c("AMERICAN SAMOA",
                       "GUAM",
                       "NORTHERN MARIANA ISLANDS",
                       "PUERTO RICO",
                       "VIRGIN ISLANDS, U.S.")) %>%
  filter(!PartOfCumulativeCountSeries) %>%
  filter(!Fatalities)

## Change the format of the date column
## from day, month, year to just year
filtered$PeriodEndDate <- format(as.Date(filtered$PeriodEndDate,
                                         format="%d/%m/%Y"),"%Y")

## Calculate incidence per 100k for
## all states and years and put them in the new column

filtered.per.100k <- filtered %>%
  mutate(per.100k = 0)

for (i in 1:length(filtered.per.100k$CountValue)) {
  pop <-
    pop.estimates[filtered.per.100k$Admin1Name[i],
                       filtered.per.100k$PeriodEndDate[i]]
  filtered.per.100k$per.100k[i] <-
    filtered.per.100k$CountValue[i]*100000/pop
}

save(filtered.per.100k, file='Good_data/incidence.per.100k.RData')
```

7.2.3.2 Yearly Incidence per 100,000 from 2000 to 2016

As mentioned in section 7.2.1, for the years 2000 to 2016, incidence data is obtained from annual MMWR reports. After reading in the data, the cleaning of the generated data set is extensive due to formatting issues when trying to pull a table from a *.pdf* file. The formatting issues include but are not limited to: columns being combined into a single column and unknown characters being inputted with strange formats. First, we format the output of the `extract_tables()` function to be a data frame. Next, if necessary, we split any combined columns into two using `separate()`.

With the columns split, we extract the desired indigenous measles data by US state. Note that some of the larger states are split in two, i.e., New York is split into Upstate NY and NY City, so there will be more than 51 rows. We also leave out rows containing aggregate counts for the entire US or other aggregate regions like the Pacific region. Unfortunately, the row selections are not uniform across all reports, so we had to go through each data frame to select the desired columns. Once the desired rows are selected, we use `as.numeric()` to ensure the entries are numeric values and not another structure. Additionally, we make the extracted column a data frame where the row names are the state names from the first column and the column name is the year. Finally, since we wish to combine all data from 2000 to 2016, we sort the data frame by state. However due to the differing structure of the names of states across years, for example sometimes you see District of Columbia and other times it is just D.C, we manually make adjustments to the arrangement of states after using the `arrange()` function to ensure each year has the same order of US states. After implementing this method for each year, we combine the data using the following code and save it as a *.RData* file. As the `extract_areas` function is interactive, the code cannot be run in this script. Thus, all of the code used to clean and combine each year of data can be found in `pdftoexcel.R`. The document is well commented to ensure easy reproduction.

With cleaned yearly measles incidence from 2000 to 2016, we calculate yearly incidence per 100,000 for each US state each year and save the result using the following code.

```r
library(tidyverse)
library(knitr)
## Load all required RData files
load('Good_data/alldata.RData')
load('Good_data/pop.estimates.RData')
load('Good_data/measles.2000.2016.RData')
load('Good_data/incidence.per.100k.RData')

## Define the number of states, a vector of state names
## and a vector of years
```

```
nstates <- 51
state.names <- rownames(measles_2000_2016)
years_2000_2016 <- seq(2000, 2016, 1)

## Create the empty data frame that will contain the
## incidence per 100k
measles_2000_2016_per_100k <- as.data.frame(
  matrix(
    NA,
    nrow = length(measles_2000_2016[,1]),
    ncol = length(measles_2000_2016[1,]))
  )
colnames(measles_2000_2016_per_100k) <- years_2000_2016
rownames(measles_2000_2016_per_100k) <- state.names

## For each state and each year, find the mean incidence per 100k
for (i in 1:nstates) {
  for (j in 1:length(years_2000_2016)) {
    measles_2000_2016_per_100k[i,j] <-
      measles_2000_2016[
        state.names[i],
        as.character(years_2000_2016[j])
        ]*100000/pop.estimates[
          state.names[i],
          as.character(years_2000_2016[j])
          ]
  }
}

save(measles_2000_2016_per_100k,
     file = 'Good_data/per.100k.2000.2016.RData')
```

7.2.3.3 Final Yearly Incidence per 100,000 from 1900 to 2016

The first step in creating the final yearly incidence per 100,000 data set is to create empty data sets that will contain our final yearly incidence per 100,000 and yearly incidence per 100,000 from 1900 to 1999. We also format the period end date as a numeric instead of character. Next, we find yearly measles incidence per 100,000 by state from 1900 to 2000. We then create our final data set by combining the two data sets. Note that the data set containing yearly measles incidence per 100,000 from

1900 to 2001 goes up to 2010, it just has no data for the years 2002 to 2010. We also set all NA values to 0. The final data set is saved to a *.RData* file.

```
library(tidyverse)

## Load all required RData files
load('Good_data/alldata.RData')
load('Good_data/incidence.per.100k.RData')
load('Good_data/measles.2000.2016.RData')
load('Good_data/per.100k.2000.2016.RData')
load('Good_data/pop.estimates.RData')

## Define two vectors of years, the first as the number
## of years in the first data set, the second
## containing the total years after combination
years_orig <- seq(1900, 2010, 1)
years_2016 <- seq(1900, 2016, 1)

## Create the necessary empty data frames
final.incidence.data <-
  as.data.frame(matrix(NA,
                       nrow = nstates,
                       ncol = length(years_2016)))
yearly.incidence.1900.2000 <-
  as.data.frame(matrix(NA,
                       nrow = nstates,
                       ncol = length(years_orig)))

## Make the year numeric instead of a character
filtered.per.100k$PeriodEndDate <- as.numeric(
  filtered.per.100k$PeriodEndDate)

## Since we already found weekly incidence per 100k,
## we aggregate over each year for each state
for (i in 1:length(years_orig)) {
  for (j in 1:nstates) {
    yearly.incidence.1900.2000[j,i] <-
      sum(ifelse(
        filtered.per.100k$Admin1Name == state.names[j] &
          filtered.per.100k$PeriodEndDate == years_orig[i],
        filtered.per.100k$per.100k, NA), na.rm = TRUE)
  }
}

colnames(yearly.incidence.1900.2000) <- years_orig
rownames(yearly.incidence.1900.2000) <- state.names

## Combine the data for 1900 to 2001
## to the .pdf data from 2002 to 2016
first_years <- as.character(c(1900:2001))
second_years <- as.character(c(2002:2016))

colnames(final.incidence.data) <- years_2016
rownames(final.incidence.data) <- state.names

final.incidence.data[,first_years] <-
  yearly.incidence.1900.2000[,first_years]
final.incidence.data[,second_years] <-
  measles_2000_2016_per_100k[,second_years]

final.incidence.data[is.na(final.incidence.data)] = 0

save(final.incidence.data,
     file = 'Good_data/yearly.data.per.100k.RData')
```

7.2.4 Vital Statistics

7.2.4.1 Mortality from 1968 to 2016

As mentioned in section 4.1.3, vital statistics data was collected from 4 different sources and combined into one *.csv* file. The first source provided 3 *.txt* files containing mortality data for the US by state from 1968 to 2016. Since they were provided as *.txt* files, we used the `read.delim()` function to read them into R. We provide the argument `header=TRUE` so that the function knows that the first row of the table is the column names. Next, since this file contains notes at the bottom that are not a part of the data, I wrote a function called `index_of_val` to determine the row in which the data ends and the notes begin. The function takes a vector (`v`) and some value (`c`), and returns the row index at which the value first occurs. This function is useful in our case as each of the files split the data and notes with "- - -". Thus, we use the function to find the row index for which each data frame has a "- - -" in the first column, and subsets the data frame to contain only the rows prior to that index.

We also define a vector of columns that we wish to eliminate as `drop`, and remove them from the data frame as well.

With these three data sets cleaned, we combine them to have one data set for mortality from 1968 to 2016 using the `rbind()` function. Once combined we remove rows with `NA` in the "Year" column as these represent the total deaths across all years in each state. We also order the data frame by the "State" column and make the entries be uppercase to match other data sets later on. Finally, since it is more useful to work with *.csv* files, we write this data frame into a *.csv* file.

```r
library(tidyverse)
library(knitr)
## Load the Mortality data
mortality_68_78 <- read.delim(
  "Good_data/mortality/mortality_US_1968-78_state_cdcwonder.txt",
  header = TRUE)
mortality_79_98 <- read.delim(
  "Good_data/mortality/mortality_US_1979-98_state_cdcwonder.txt",
  header = TRUE)
mortality_99_16 <- read.delim(
  "Good_data/mortality/mortality_US_1999-2016_state_cdcwonder.txt",
  header = TRUE)

## Function that takes a vector (v) and a
## value (c) and returns the index at which the value first occurs
index_of_val <- function(v, c) {
  for (i in 1:length(v)) {
    if (v[i] == c) {
      index = i
      stopifnot(v[index] == c)
      return(index)
    }
  }
}

## Vector of columns to eliminate
drop <- c("Notes", "Year.Code", "State.Code")

## Using my own function to determine
## where the data ends and notes begin (see function at top)
index_mort_68 <- index_of_val(mortality_68_78$Notes, "---")
index_mort_79 <- index_of_val(mortality_79_98$Notes, "---")
index_mort_99 <- index_of_val(mortality_99_16$Notes, "---")
```

```
## Clean up the data set by removing notes and un-necessary columns
mortality_68_78_new <-
  mortality_68_78[1:(index_mort_68 - 1),
                  !(names(mortality_68_78) %in% drop)]
mortality_79_98_new <-
  mortality_79_98[1:(index_mort_79 - 1),
                  !(names(mortality_79_98) %in% drop)]
mortality_99_16_new <-
  mortality_99_16[1:(index_mort_99 - 1),
                  !(names(mortality_99_16) %in% drop)]

## Combine the 3 datasets into one containing all data
mortality_68_16 <- rbind(mortality_68_78_new,
                         mortality_79_98_new, mortality_99_16_new)

## Remove rows with NA in the year since these
## represent the total over all years for each state,
## and order by state name
mortality_68_16_final <- mortality_68_16 %>%
  filter(!is.na(Year)) %>%
  arrange(State) %>%
  mutate(State = toupper(State))

write.csv(
  mortality_68_16_final,
  file = "Good_data/mortality_US_1968-2016_state_cdcwonder.csv")
```

7.2.4.2 Natality from 1995 to 2019

The second source of US vital statistics data provided 3 *.txt* files containing natality data for the US by state from 1995 to 2019. Using the same methods outlined in section 7.2.4.1, we clean and combine the files into one working file which we will write into a new *.csv* file.

```
## load the natality data
natality_95_02 <- read.delim(
  "Good_data/natality/natality_US_1995-2002_state_cdcwonder.txt",
  header = TRUE)
natality_03_06 <- read.delim(
```

```r
  "Good_data/natality/natality_US_2003-06_state_cdcwonder.txt",
  header = TRUE)
natality_07_19 <- read.delim(
  "Good_data/natality/natality_US_2007-19_state_cdcwonder.txt",
  header = TRUE)

## Use my own function to determine where
## the data ends and notes begin
## (see definition of index_of_val above)
index_nat_95 <- index_of_val(natality_95_02$Notes, "---")
index_nat_03 <- index_of_val(natality_03_06$Notes, "---")
index_nat_07 <- index_of_val(natality_07_19$Notes, "---")

## Clean up the data set by removing notes and un-necessary columns
natality_95_02_new <-
  natality_95_02[1:(index_nat_95 - 1),
                !(names(natality_95_02) %in% drop)]
natality_03_06_new <-
  natality_03_06[1:(index_nat_03 - 1),
                !(names(natality_03_06) %in% drop)]
natality_07_19_new <-
  natality_07_19[1:(index_nat_07 - 1),
                !(names(natality_07_19) %in% drop)]

## Combine the 3 datasets into one containing all data
natality_95_19 <- rbind(natality_95_02_new,
                        natality_03_06_new,
                        natality_07_19_new)

## Remove rows with NA in the year since
## these represent the total over all years for each state,
## and order by state name
natality_95_19_final <- natality_95_19 %>%
  filter(!is.na(Year)) %>%
  arrange(State) %>%
  mutate(State = toupper(State))

write.csv(
  natality_95_19_final,
  file = "Good_data/natality_US_1995-2019_state_cdcwonder.csv")
```

7.2.4.3 Creating a Single File Containing US Vital Statistics by State

As our goal was to use the US vital statistics data to improve our model that simulates the US canonical path (refer to section 7.5.1), we used the other two sources described in section 7.2.4.1 to have one data set containing births, deaths, population, birth rate and death rate by US state. However, since there were 61 *.pdf* files containing the missing mortality and natality data, using `tabulizer` to extract the tables from each PDF would be incredibly time consuming. Instead, we created a new *.csv* file, `Good_data/vital_statistics_US_1953-2019_state_NCHS.csv`, that would contain all of the births and deaths data from 1953 to 2019. We started by entering the deaths data for 1968 to 2016 and the births data from 1995 to 2019 using the *.csv* files created using the code in sections 7.2.4.1 and 7.2.4.2. Next, we entered the births and deaths data manually from the 61 individual *.pdf* files. Although time consuming, this resulted in one cohesive *.csv* file containing births and deaths from 1953 to 2019.

Since we will need birth and death rates for the model, we load the *.csv* file into R to create an *.RData* file containing, births, deaths, birth rate, death rate, and other related columns for use in later code. First, we read the file into R using `read.csv()`, with the argument `header=TRUE`. However, since we have a "Notes" column containing notes written while entering, we subset the data frame to only include the columns we need. When we read in the data file, the column names had obscure labels, where some used non-English characters. Thus, we change the column names to ones easy to type using a US-English keyboard. To match other data later on, we use `toupper()` to make the state names capitalized. Doing this now makes it simpler to find the population corresponding to each state and year, as the state names will be identical. We also create vectors containing the state names and the years of data for use in the code.

```r
library(tidyverse)

load("Good_data/pop.estimates.RData")

## Read the csv file
vital_stats <- read.csv(
  file = "Good_data/vital_statistics_US_1953-2019_state_NCHS.csv",
  header = TRUE)

## Set the first column containing years to "Year"
## due to formatting issue in column name (non-english character)
Year <- vital_stats[,1]

## Subset to take only the columns I need (year will be added after)
```

```r
vital_stats <- vital_stats[,c("State", "Births", "Deaths")]

## Change the column names
colnames(vital_stats) <- c("State", "Births", "Deaths")

## Re-add the year column with the proper column name
vital_stats$Year <- Year

## Make the state names uppercase for uniformity across data frames
vital_stats$State <- toupper(vital_stats$State)

state_names <- unique(long_pop_estimates_final$State)
years_final <- unique(long_pop_estimates_final$Year)
```

With the "State" column formatted properly, we create a new column, "Population", that contains the population for each state each year. Next, we reorder the columns in a sensible way and then use the `mutate()` function to create two new columns for the birth and death rates. At this point, the data set is ready for use in the model, however, we added a few more columns to rank states based on their population, birth rate and death rate. These were not necessary for the analysis in the report but is nice for summarizing the data. We then save the data set to a *.RData* file for use in later code.

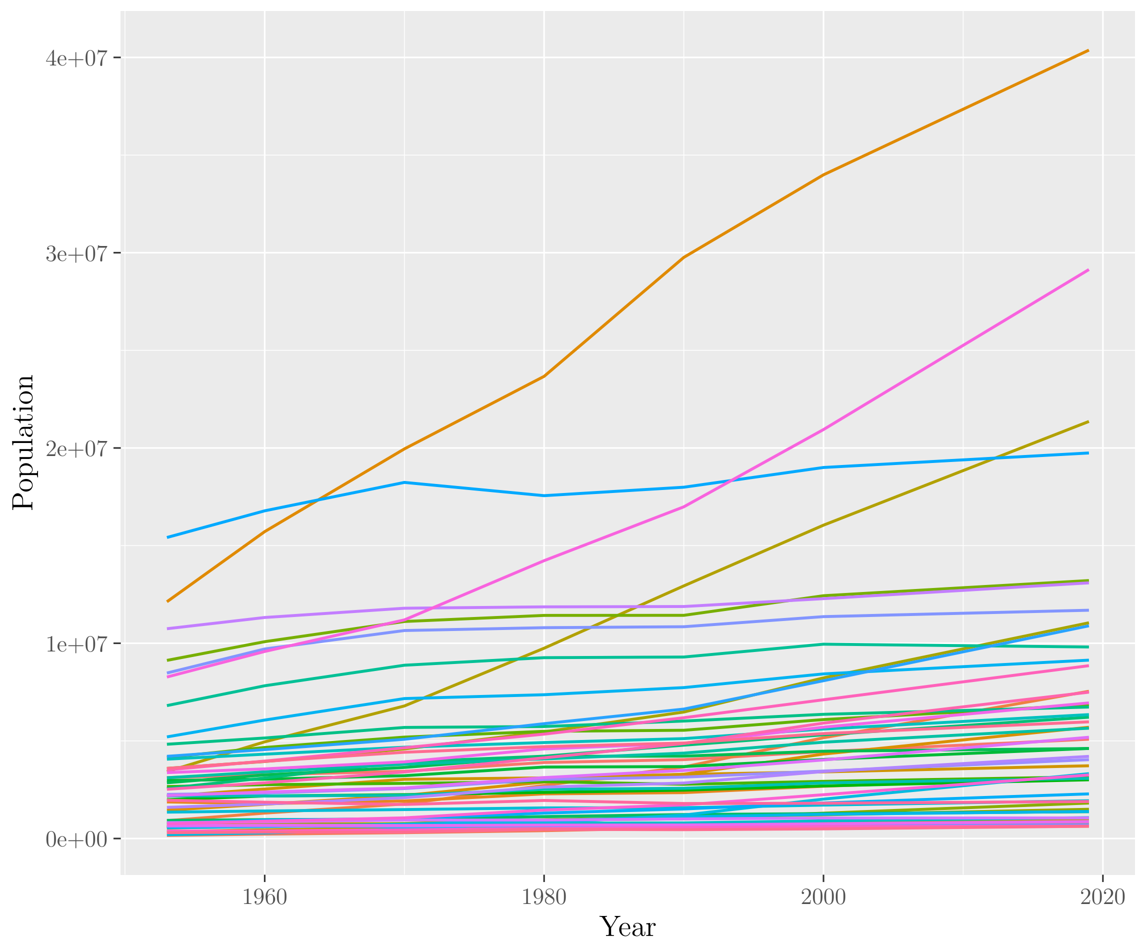
```r
## Add a population column
for (k in 1:nrow(vital_stats)) {
  for (sn in state_names) {
    for (yf in years_final) {
      if (vital_stats$State[k] == sn & vital_stats$Year[k] == yf) {
        vital_stats$Population[k] <-
          long_pop_estimates_final$Population[
            long_pop_estimates_final$State == sn &
              long_pop_estimates_final$Year == yf]
      }
    }
  }
}

## Reorder the columns
vital_stats <-
  vital_stats[,
              c("Year", "State", "Population", "Births", "Deaths")]
```

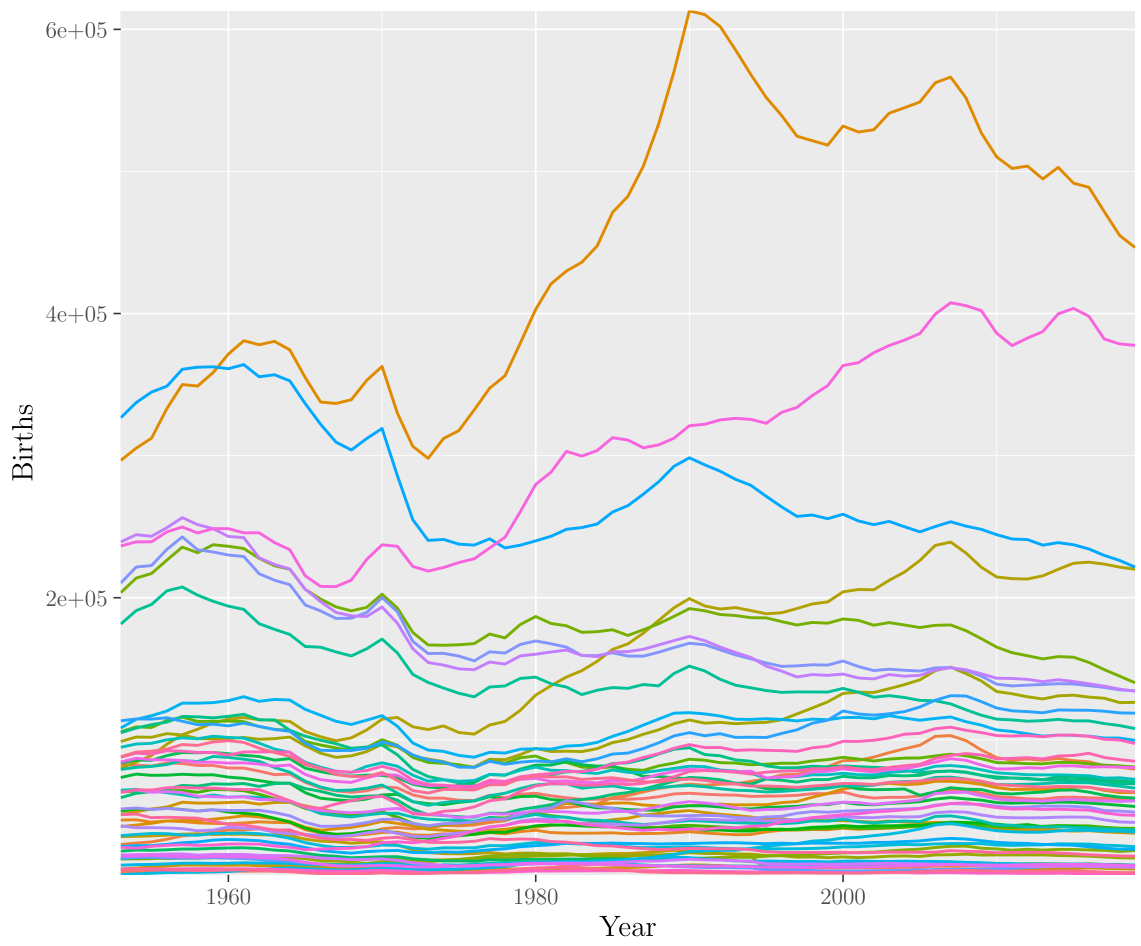
```
## Create Birth Rate and Death rate columns
vital_stats <- vital_stats %>%
  mutate(`Birth Rate` = Births/Population) %>%
  mutate(`Death Rate` = Deaths/Population)

## Create columns for rankings
vital_stats$`Pop Rank` <-
  rep(0, length(vital_stats$Population))
vital_stats$`Birth Rate Rank` <-
  rep(0, length(vital_stats$`Birth Rate`))
vital_stats$`Death Rate Rank` <-
  rep(0, length(vital_stats$`Death Rate`))

years <- unique(vital_stats$Year)

## Sort by the column we want to rank by in a decreasing order,
## then assign a number from 1:51 according to that order.
## Then reorder back the same way and add the column
## into the original data frame

### Population ranking
for (ii in years) {
  vital_stats_filtered <- vital_stats %>%
    filter(Year == ii) %>%
    arrange(desc(Population))

  vital_stats_filtered$`Pop Rank` <- 1:51

  vital_stats_normal <- vital_stats_filtered %>%
    arrange(State)

  vital_stats[vital_stats$Year == ii,] <- vital_stats_normal
}

### Birth Rate Ranking
for (ii in years) {
  vital_stats_filtered <- vital_stats %>%
    filter(Year == ii) %>%
    arrange(desc(`Birth Rate`))

  vital_stats_filtered$`Birth Rate Rank` <- 1:51
  vital_stats_normal <- vital_stats_filtered %>%
    arrange(State)

  vital_stats[vital_stats$Year == ii,] <- vital_stats_normal
}

### Death Rate Ranking
for (ii in years) {
  vital_stats_filtered <- vital_stats %>%
    filter(Year == ii) %>%
    arrange(desc(`Death Rate`))

  vital_stats_filtered$`Death Rate Rank` <- 1:51

  vital_stats_normal <- vital_stats_filtered %>%
    arrange(State)

  vital_stats[vital_stats$Year == ii,] <- vital_stats_normal

save(vital_stats, file = "data/vital_stats_data.RData")
}
```

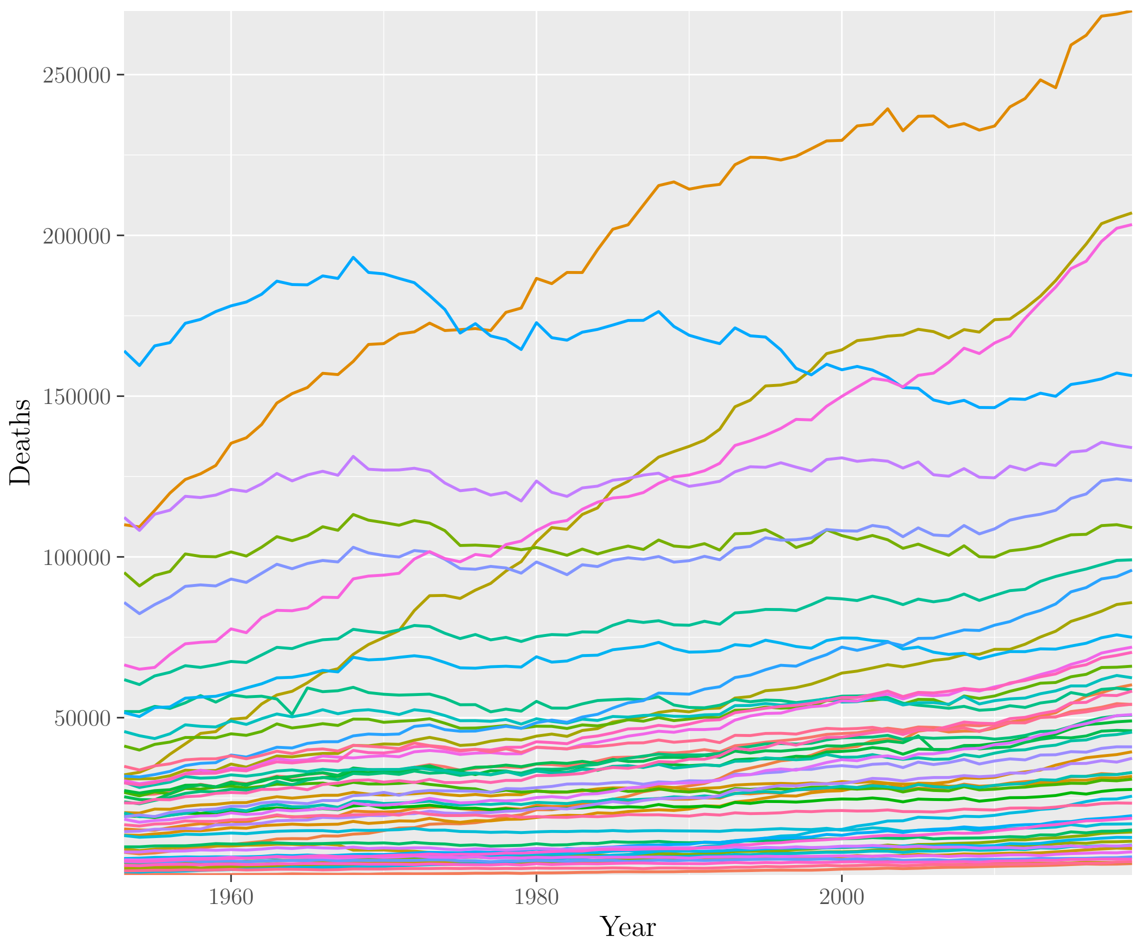
7.2.4.4 Visualizing Vital Statistics

The following figures show population, births, deaths, birth rate, and death rate by state over time.

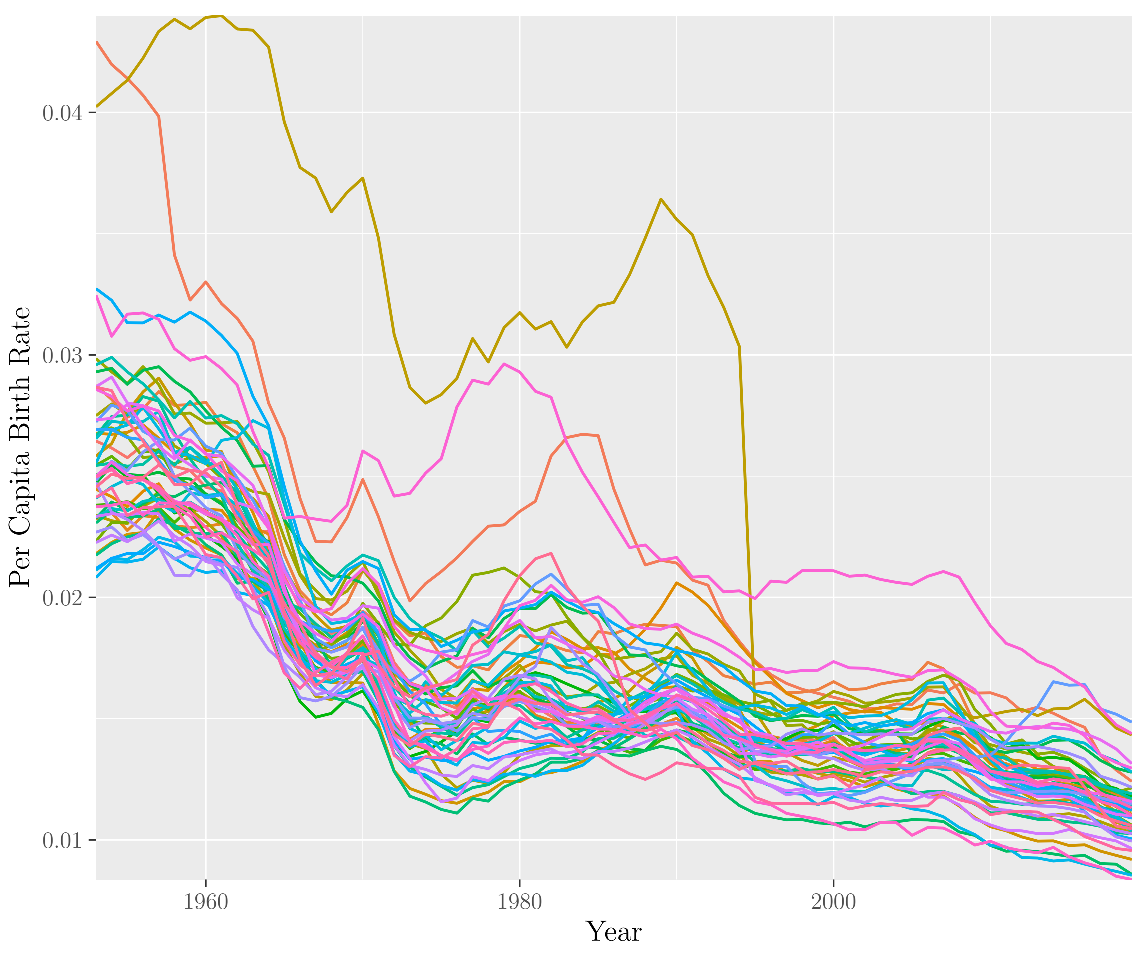
```
## Plot population
pop_plot <- ggplot() +
  geom_line(data = vital_stats,
            aes(Year, Population, col = State),
            lty = 1, lwd = 1) +
  ylab("Population") +
  theme(legend.position="none")

## Plot births
birth_plot <- ggplot() +
  geom_line(data = vital_stats,
            aes(Year, Births, col = State),
            lty = 1, lwd = 1) +
  ylab("Births") +
  theme(legend.position="none") +
  coord_cartesian(expand = FALSE)

## Plot deaths
death_plot <- ggplot() +
  geom_line(data = vital_stats,
            aes(Year, Deaths, col = State),
            lty = 1, lwd = 1) +
  ylab("Deaths") +
  theme(legend.position="none") +
  coord_cartesian(expand = FALSE)

## Plot per capita birth rates
birth_rate_plot <- ggplot() +
  geom_line(data = vital_stats,
            aes(Year, `Birth Rate`, col = State),
            lty = 1, lwd = 1) +
  ylab("Per Capita Birth Rate") +
  theme(legend.position="none") +
  coord_cartesian(expand = FALSE)

## Plot per capita death rates
death_rate_plot <- ggplot() +
  geom_line(data = vital_stats,
            aes(Year, `Death Rate`, col = State),
            lty = 1, lwd = 1) +
  ylab("Per Capita Death Rate") +
  theme(legend.position="none") +
  coord_cartesian(expand = FALSE)
```

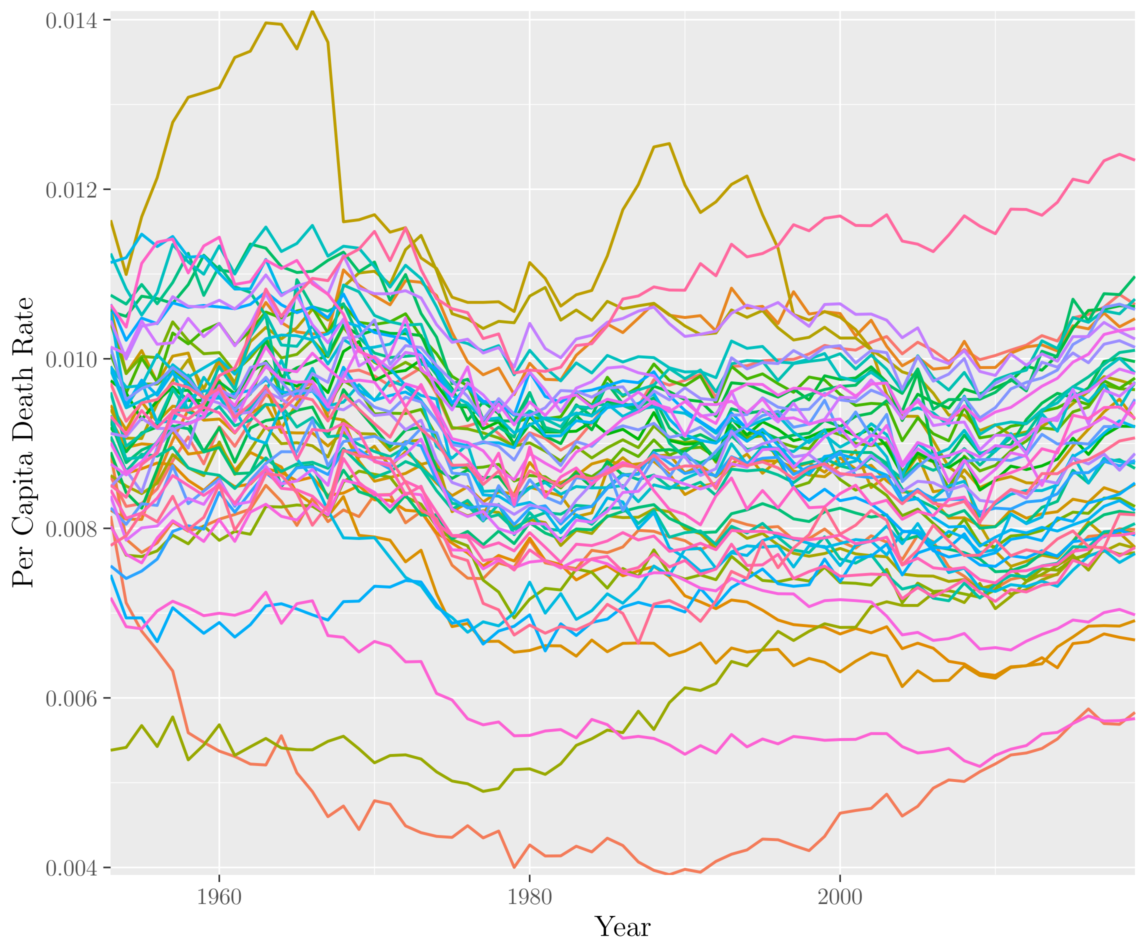

4e+07
3e+07
2e+07
1e+07
0e+00
Population
1960
1980
2000
2020
Year


6e+05
4e+05
2e+05
Births
1960
1980
2000
Year



Deaths
250000
200000
150000
100000
50000
1960
1980
2000
Year


0.04
0.03
0.02
0.01
Per Capita Birth Rate
1960
1980
2000
Year



7.2.5 Vaccination Coverage

In section 4.1.3 we describe the vaccination coverage data we were able to obtain that came from 3 different sources. The first contained MMR vaccine coverage data among 0 to 35 month olds in the United States for each US state from 2011 to 2017 as a *.csv* file. We attempted to find the data that was used to create this file to try to find more years of data, which resulted in the third source of data that provided uncleaned individual-level vaccine data from 1995 to 2010 and 2018 to 2020. Most of the files are provided as *.DAT* files[5] and some years are accompanied by either SAS or R input statements or both. However, the input statements were substantial and un-commented. Thus, cleaning this data would be incredibly time-consuming and not in the scope of this project. From here we decided to look for other sources of

[5] *.DAT* files are used to store data and are written based on the program that created them. They can be composed of plain text or binary. Thus, knowing what type of data it contains is crucial [42].

MMR coverage data and were able to find coverage estimates for the entirety of the US from 1980 to 2020 as a *.xlsx* file. Unfortunately this data set was only for the US as a whole, not by state. Thus, we decided to compare the coverage estimates for the entirety of the US to the data split by US state from 2011 to 2017.

First we loaded in each data file. Next, we set all of the character columns of interest to be uppercase for simplicity. Additionally, we filter the data set for each individual state to only provide estimates of MMR coverage in each of the US states, while getting rid of areas that are not of interest. We also re-format the Birth year/cohort column to be numeric instead of a character. For the data set containing coverage for the whole US, we create two different data frames. One containing WHO estimates and the other containing Official Coverage Values. This data set also contained coverage data for Canada, so we filter out the Canadian coverage as well as rows containing `NA`, and then order the data frame by the "YEAR" column. The vaccination data for each US state from 2011 to 2017 is split into different age categories: 13, 19, 24, and 35 months. The MMR coverage for each state and age category is an estimate based on a sample of the population. In order to find the MMR coverage for all age groups within a state, we must find the weighted mean coverage, calculated using the given sample sizes. First, we create a column containing the coverage multiplied by the sample size. Next, we create a data frame that contains the total sample size for each US state each year for all age groups. Finally we create a data frame that contains the weighted mean MMR coverage in each state each year across all age groups by dividing each entry in the "Weighted Coverage" column and then taking the sum of all age groups. We also add a column representing the US states. Additionally, we transform the data frame from wide-format to long format for use in later code and add a column for the sample size.

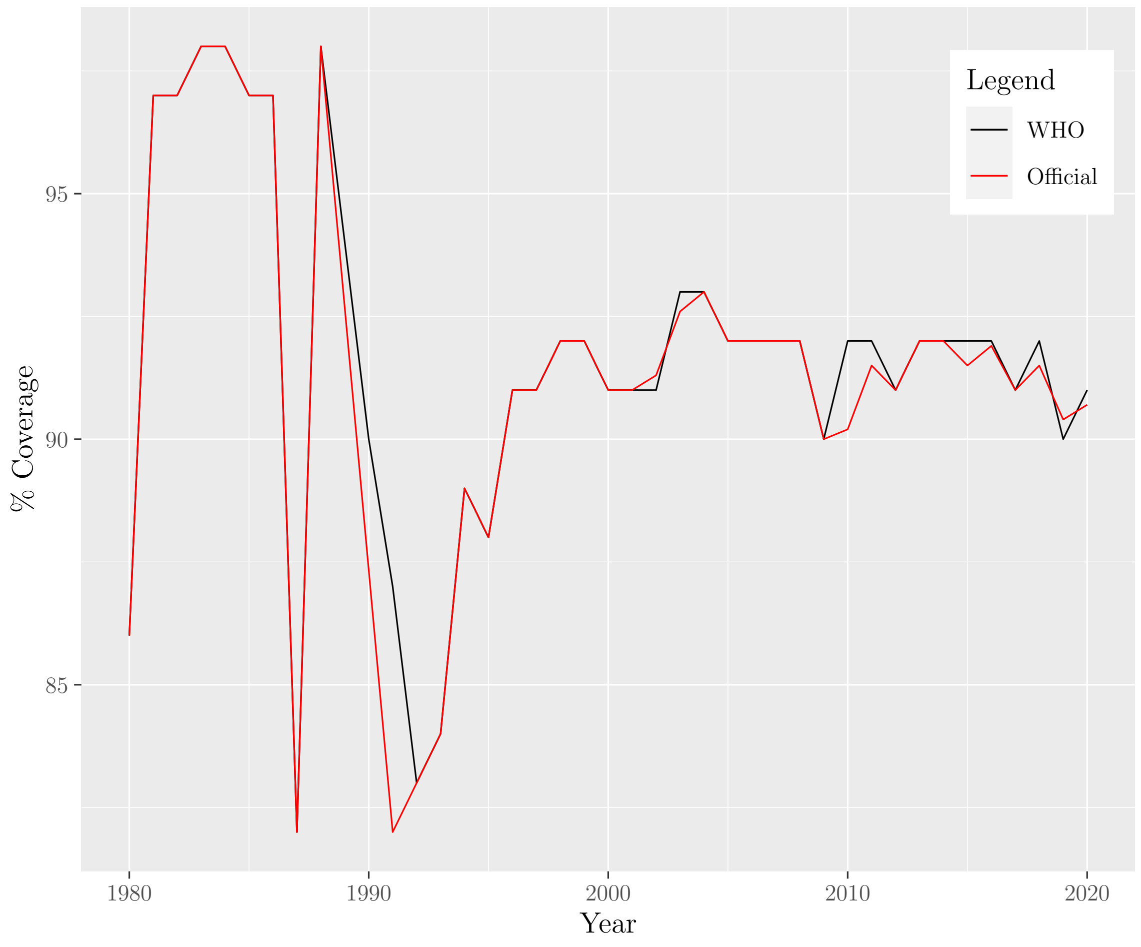
```r
library(readxl)

## Read in the files
vaccination <- read.csv(
  file = "Good_data/vaccine_coverage_US_2011-17_state_vaxxview.csv",
  header = TRUE)
vaccination_1980_2020 <- read_excel(
  "Good_data/vaccine_coverage_US_CAN_1980-2020_WHO.xlsx")

## Make character vectors all uppercase for simplicity
vaccination$Vaccine <- toupper(vaccination$Vaccine)
vaccination$Geography <- toupper(vaccination$Geography)

vaccination_1980_2020$NAME <- toupper(vaccination_1980_2020$NAME)

library(tidyverse)
```

```
## Filter out unwanted elements of the data
## (!grepl('-', Geography) gets rid of strings with a hyphen )
vaccination_mmr <- vaccination %>%
  filter(Vaccine == "MMR" &
           Geography.Type == "States/Local Areas" &
           !grepl('-', Geography) &
           Geography != "GUAM" &
           !grepl('-', Birth.Year.Birth.Cohort))
vaccination_mmr$Birth.Year.Birth.Cohort <-
  as.numeric(vaccination_mmr$Birth.Year.Birth.Cohort)

### WHO Estimates
vaccination_US_1980_2020_WUENIC <-vaccination_1980_2020 %>%
  filter(CODE == "USA",
         COVERAGE != is.na(COVERAGE),
         COVERAGE_CATEGORY == "WUENIC")

vaccination_US_1980_2020_WUENIC <-
  vaccination_US_1980_2020_WUENIC[
    order(vaccination_US_1980_2020_WUENIC$YEAR),]

### Official Coverage
vaccination_US_1980_2020_OFFICIAL <- vaccination_1980_2020 %>%
  filter(CODE == "USA",
         COVERAGE != is.na(COVERAGE),
         COVERAGE_CATEGORY == "OFFICIAL")

vaccination_US_1980_2020_OFFICIAL <-
  vaccination_US_1980_2020_OFFICIAL[
    order(vaccination_US_1980_2020_OFFICIAL$YEAR),]

vaccination_mmr_weighted <- vaccination_mmr %>%
  mutate(`Weighted Coverage` = `Estimate....`*`Sample.Size`)

## Create vectors for state names and years,
## as well as the number of states and years
states <- unique(vaccination_mmr$Geography)
n_states <- length(states)
years <- unique(vaccination_mmr$Birth.Year.Birth.Cohort)
n_years <- length(years)

## Create the empty data frame that will contain the data
```

```r
sample_sizes <- as.data.frame(matrix(nrow = n_states*n_years, ncol = 3))
colnames(sample_sizes) <- c("State", "Year", "Sample Size")

## Set the State column to be the state vector repeated
## for each year of data
sample_sizes$State <- rep(states, n_years)

## Create a vector that has the years vector repeated for
## each state and then ordering it by the year, this way we can
## put it in the data frame so that each state is
## accompanied by each year in the data frame
years_ordered <- sort(rep(years, n_states))
## Assigning the sorted repeating vector of years to the Year column
sample_sizes$Year <- years_ordered

## Filter the state-wise MMR data frame for each state
## each year and finding the sum of the sample sizes and
## assign this to the corresponding state and year
## in the sample sizes data frame
for (ii in states) {
  for (jj in years) {
    filtered <- vaccination_mmr %>%
      filter(Geography == ii & `Birth.Year.Birth.Cohort` == jj)

    sample_sizes$`Sample Size`[sample_sizes$State == ii &
                               sample_sizes$Year == jj] <-
      sum(filtered$`Sample.Size`)
  }
}

## Create a data frame that will contain the
## mean vaccine coverage across age groups
vaccination_mmr_final <-
  as.data.frame(matrix(nrow = n_states, ncol = n_years))

for (i in 1:n_years) {
  for (j in 1:n_states) {
    ## Filter for each year and state
    filtered_vacc <-
      vaccination_mmr_weighted %>%
      filter(Birth.Year.Birth.Cohort == years[i]
             & Geography == states[j])
    vaccination_mmr_final[j, i] <-
      sum(
        filtered_vacc$`Weighted Coverage`/sample_sizes$`Sample Size`[
          sample_sizes$State == states[j] & sample_sizes$Year == years[i]
          ]
        )

  }
}
colnames(vaccination_mmr_final) <- as.character(years)

## Add a column that contains the US States
vaccination_mmr_final$state <- states

## Create a long version of the data set with
## columns for state, year and coverage
vaccination_final <-
  pivot_longer(vaccination_mmr_final,
               cols = !state,
               names_to = "year",
               values_to = "coverage")
vaccination_final$year <- as.numeric(vaccination_final$year)

## Check that the state and year columns match
setequal(vaccination_final$state, sample_sizes$State) &
  setequal(vaccination_final$year, sample_sizes$Year)

## Add a sample size column
vaccination_final$`Sample Size` <- sample_sizes$`Sample Size`
```

After creating the data sets, we created a couple visuals to determine which sources to use in the model. First we create a plot of US wide MMR coverage data, comparing the WHO estimates to the Official Coverage.

```
## Set the line colours manually
colours <- c("WHO" = "black", "Official" = "red")

## Compare the sources of vaccine coverage for the US
US_vaccine_compare_source <- ggplot() +
  geom_line(data = vaccination_US_1980_2020_WUENIC,
            aes(YEAR, COVERAGE, color = "WHO")) +
  geom_line(data = vaccination_US_1980_2020_OFFICIAL,
            aes(YEAR, COVERAGE, color = "Official")) +
  labs(x = "Year",
       y = "$\\%$ Coverage",
       color = "Legend") +
  scale_colour_manual(values = colours) +
  theme( ## Change the position of the legend to
    ## be on the plot instead of beside it
    legend.position = c(0.98, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6)
  )
US_vaccine_compare_source
```



After creating this figure, we decided to move forward with the Official Coverage data as there was very little difference. The next plot we created was comparing the US-wide data (black) to the state-wise data (red).

```r
## Set the line colours manually
colours <- c("US" = "black", "States" = "red")

## Compare vaccine coverage by state to aggregate vaccine coverage
## of the US
vaccine_coverage_plot <- ggplot() +
  geom_line(data = vaccination_final,
            aes(year, coverage, group = state, color = "States"),
            lty = 1) +
  geom_line(data = vaccination_US_1980_2020_OFFICIAL,
            aes(YEAR, COVERAGE, color = "US")) +
  geom_point(data = vaccination_final,
             aes(year, coverage, group = state,
                 color = "States", size = `Sample Size`)) +
  labs(x = "Year",
       y = "\\% Coverage",
       color = "Data Type") +
  scale_x_continuous(limits = c(2010,2018), expand = c(0,0))+
  scale_colour_manual(values = colours)
vaccine_coverage_plot
```

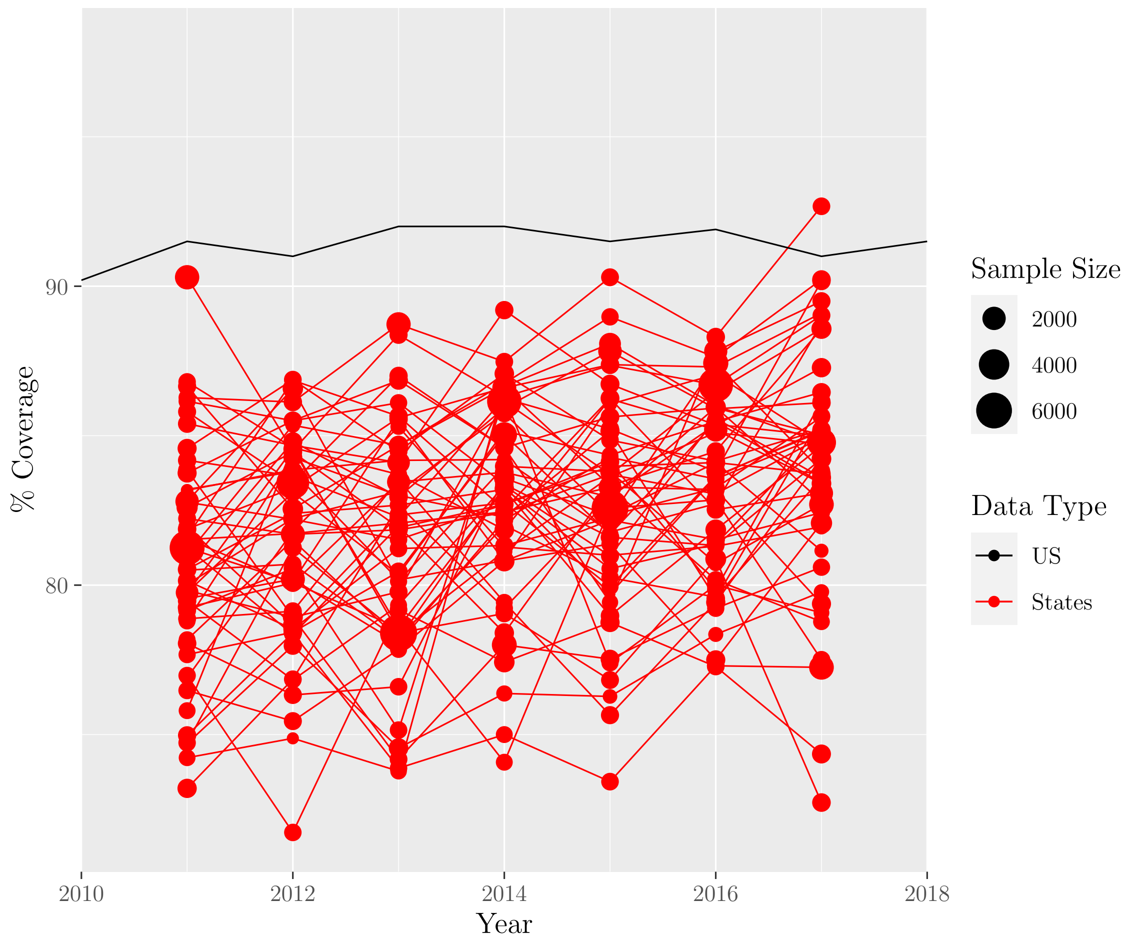


Figure 22: Comparison of state-level and aggregate US measles vaccine coverage. The reason these data do not match up is due to the fact that the US wide data is the percent of children that received their first dose at the recommended interval (12 months) or later, while the state-wise data only considers those that received their doses at 13, 19, 24, 35 months .

Due to the differences in the definition of coverage between the two sources, the MMR coverage estimates do not line up. Thus, we could not use a combination of these data sets in our simulations, so we decided to use the source that provides the most data, i.e., the data for the entirety of the US. Finally, we save the chosen data set to an *.RData* file for use in other code scripts.

```
vaccination_data <- vaccination_US_1980_2020_OFFICIAL

save(vaccination_data, file = "data/vaccination_data.RData")
```

7.3 Creating the Canonical Path

7.3.1 Calculating Weighted Mean Incidence and CV

In order to create the canonical path, we must calculate the weighted mean incidence and un-weighted smooth CV for for each US state, and then find their mean point in incidence space each year to create the US canonical path. Since we will be using the same method of calculating weighted incidence and smooth un-weighted CV in other code, we create a function called `cv_inc_fun()`. This allows us to find weighted incidence and smoothed CV uniformly across all code files without duplicating code. The function takes the following arguments: `data`, `state_names`, `all_years`, `gaussian_window`, `sd`, `mean_loc`, `year_labels`, `find_CI`, `find_states`, `inc_weighting`, `cv_weighting`, and an optional argument `state_weights`. `data` is the data containing incidence per 100,000 individuals for each state and each year, where each row is a state and each column is a year. `state_names` is the vector of state names included in the data. `all_years` is a vector containing the years that will be used to find weighted mean incidence and smoothed un-weighted CV. `gaussian_window` is the number of years to wait until the first data point is calculated, 10 by default. `sd` is the standard deviation of the Gaussian filter, 3 by default. `mean_loc` is how many years prior to the year in question does the center of the Gaussian filter lie, 2 by default. `find_CI` is a logical argument that tells the function if it should calculate the confidence intervals on weighted incidence or not, `FALSE` by default. `find_states` is a logical argument that tells the function if it should save the weighted incidence and smoothed un-weighted CV by state as well as the aggregate values, `FALSE` by default. `inc_weighting`, either "gaussian" or "moving", tells the function which type of weighting to use for incidence, "gaussian" by default. `cv_weighting`, either "smoothed" or "gaussian", tells the function which type of weighting to use for CV, "smoothed" by default. Finally, `state_weights` is a data frame containing the weights for each state in each year that will be used when finding the mean point in incidence space, `NULL` by default.

Within the function, we begin by requiring the `smoother()` package, for the `smth.gaussian()` function. Next we define variables representing the number of states and the final years of data that will be shown in the plot. If the function is not provided a data frame of weights, the function creates a data frame containing equally-weighted weights. Additionally, we re-name the data frame containing incidence to be more descriptive. Next, we use the pre-defined function, `gaussian()`, that calculates Gaussian values for a given mean location and standard deviation. This function is found in `functions.R`, however the code is displayed below.

```
### Function that applies a Gaussian transform to a
### vector x, given the mean and standard deviation
```

```
gaussian <- function(x, mean, sd) {

  gaussian_value <-
    (1/(sd*sqrt(2*pi))*exp(-((x-mean)^2)/(2*sd^2)))
  return(gaussian_value)

}
```

Using this function, the Gaussian weights for each calculation of the weighted mean are found. Next, if `inc_weighting`="gaussian" the function finds the sum of the Gaussian weights and then divides each weight by the sum to get the final Gaussian weights that will be used for weighted incidence. With the weights calculated, the function multiplies the year in question and all years prior by their corresponding weights and take their sum to find the weighted mean incidence for the given year. if `inc_weighting`="moving" it will use a moving average to calculate weighted incidence.

Next, CV is calculated. If `cv_weighting`="smoothed", we use the same method as Graham *et al.*, such that we use a moving CV that is calculated for each 10 year time shift, i.e., for 1921, it finds the CV of the incidence (un-weighted) from 1912 to 1921. For 1922, the CV of 1913 to 1922 and so on. Finally, it smooths the un-weighted CV using the `smth.gaussian()` function. It provides `smth.gaussian()` the following arguments: `tails`=`TRUE` (tells the function to include the tails), `window`=`smooth_window`= 1 (tells it to smooth the entire input), and `alpha`= $3 * 3$ (determines the breadth of the Gaussian window, chosen as the variance). If `cv_weighting`="gaussian", the function will the Gaussian filter that is used to weight incidence to weight CV.

With the weighted incidence and CV calculated for each US state, we find the mean weighted incidence and CV across all US states to create the US canonical path, using the weights either provided to the function, or generated within. We conducted a sensitivity analysis to determine whether each state should be weighted equally, or if we should weight each state based on population size (Refer to section 7.4.1).

We also find the standard deviation of incidence so that if `find_CI`=`TRUE`, we can calculate confidence intervals. Finally, the function creates the data frames that contain Incidence, CV, Year, and the confidence interval values (if desired). To provide a sense of time progression along the plot, the function also creates a data frame containing only the values based on the `year_labels` argument.

If `find_states`=`TRUE`, the function will output data frames containing weighted incidence, smoothed un-weighted CV and filtered weighted incidence and smoothed un-weighted CV (filtered based on the `year_labels` argument), along with data frames containing this data for the US as a whole. The function also outputs the `final_data` vector for use in other code. This function can be found in `functions.R`, however the code is displayed below.

```
### Function that takes several arguments required to create the
### data used for the canonical path, and finds weighted mean
### incidence and smooth unweighted cv. These are then outputted
### into a data frame that will be used for plotting.
### It also creates a data frame containing points to plot on the path.
### Outputs the two data frames and a vector of all years plotted
cv_inc_fun <- function(data, state_names, all_years,
                       inc_window=10, sd=3, mean_loc=2,
                       year_labels, find_CI=FALSE,
                       find_states=FALSE,
                       inc_weighting = "gaussian",
                       cv_weighting = "smoothed",
                       state_weights = NULL) {
  require(smoother)
  final_incidence_data <- data
  ## Number of states
  nstates <- length(state_names)
  ## The resulting years of data
  final_data <- seq(all_years[1] + inc_window - 1,
                    all_years[length(all_years)], 1)

  ## Define the empty data frame for the Gaussian weights.
  ## Each row is the weights for a given year
  gaussian_weights <-
    as.data.frame(matrix(
      NA,
      ncol = length(all_years),
      nrow = length(final_data)))

  for (i in 1:length(final_data)) {
    ## Vector of 1 to the length of the number of years
    ## we need to weight
    x <- c(1:(i+inc_window - 1))
    ## Location of the mean (2 years before the year)
    mean <- (i+inc_window - 1)-mean_loc
    gaussian_vec <- gaussian(x, mean, sd)
    gaussian_weights[i,(1:(i+inc_window - 1))] <- gaussian_vec
  }
```

```r
sum_gaussian_weights <- vector(length= length(final_data))
## Find the sum of the Gaussian weights for each year
for (i in 1:length(final_data)) {
  sum_gaussian_weights[i] <- sum(gaussian_weights[i,],
                                 na.rm = TRUE)
}

final_gaussian_weights <- as.data.frame(
  matrix(NA,
         nrow = length(final_data),
         ncol = length(all_years)))
## Divide the weights for each year by the sum of the
## weights in that year
for (i in 1:length(final_data)) {
  final_gaussian_weights[i,(1:(i+inc_window - 1))] <-
    gaussian_weights[i,(1:(i+inc_window - 1))] / sum_gaussian_weights[i]
}

## Create a vector that will contain the weighted
## mean incidence for a given state
weighted_mean_incidence <- vector(length = length(final_data))
## Create a data frame that will contain the weighted
## mean incidence for all states (each row is a different state)
weighted_mean_incidence_all <-
  as.data.frame(matrix(
    NA,
    nrow = nstates,
    ncol = length(final_data)))

## In case our data has more years than we wish to plot,
## we find the column corresponding to the first year of data
## to use as an index when calculating weighted incidence
starting_column <-
  which(colnames(final_incidence_data) == as.character(all_years[1]))

stopifnot(inc_weighting %in% c("gaussian", "moving"))

if (inc_weighting == "gaussian") {
  for (i in 1:nstates) {
    for (j in 1:length(final_data)) {
      ## Find the weighted mean incidence for a given state (each row).
      inc <- final_incidence_data[i,
            starting_column:(j+(starting_column + inc_window - 2))]
      weights <- final_gaussian_weights[j,(1:(j+inc_window - 1))]
      weighted_mean_incidence[j] <-
        sum(inc*weights , na.rm = TRUE)

      weighted_mean_incidence_all[i,j] <- weighted_mean_incidence[j]

    }
  }
} else {
  for (i in 1:nstates) {
    for (j in 1:length(final_data)) {
      ## Find the weighted mean incidence for a given state (each row).
      inc <-
        final_incidence_data[
          i,
          (starting_column -1 + j):(j+(starting_column + inc_window - 2))]

      weighted_mean_incidence[j] <-
        sum(inc, na.rm = TRUE)/inc_window

      weighted_mean_incidence_all[i,j] <- weighted_mean_incidence[j]

    }
  }
}

## Set the row names as the state names and
## the column names as the years (1921 to 2016)
colnames(weighted_mean_incidence_all) <- final_data
rownames(weighted_mean_incidence_all) <- state_names

## Create empty vectors and data frames to contain the CV data
unweighted_cv <- vector(length = length(final_data))
unweighted_cv_all <-
  as.data.frame(matrix(
    NA,
    nrow = nstates,
    ncol = length(final_data)))

stopifnot(cv_weighting %in% c("gaussian", "smoothed"))
if (cv_weighting == "smoothed") {
  for (i in 1:nstates) {
    for (j in 1:length(final_data)) {
      ## CV = standard deviation/mean
      inc <-
        final_incidence_data[
          i,
          (j+starting_column -1):(j+(starting_column -1 + inc_window - 1))]
      sd_inc <- sd(as.numeric(inc), na.rm = TRUE)
      mean_inc <- mean(as.numeric(inc), trim = 0, na.rm = TRUE)
      unweighted_cv[j] <- sd_inc / mean_inc

      unweighted_cv_all[i,j] <- unweighted_cv[j]

    }
  }

  ## Set the row names as the state names
  ## and the column names as the years
  colnames(unweighted_cv_all) <- final_data
  rownames(unweighted_cv_all) <- state_names

  ## Smooth the CV using smth.gaussian()
  smooth_unweighted_cv_all <-
    as.data.frame(matrix(
      NA,
      nrow = nstates,
      ncol = length(final_data)))

  for (i in 1:nstates) {
    smooth_unweighted_cv_all[i,] <- smth.gaussian(
      as.numeric(unweighted_cv_all[i,]),
      tails = TRUE,
      window = 1,
      alpha = 3 * 3)
  }

  ## Set the row names as the state names
  ## and the column names as the years
  colnames(smooth_unweighted_cv_all) <- final_data
  rownames(smooth_unweighted_cv_all) <- state_names
} else {
```

```
  weighted_cv <- vector(length = length(final_data))

  weighted_cv_all <-
    as.data.frame(matrix(
      NA,
      nrow = nstates,
      ncol = length(final_data)))

  for (i in 1:nstates) {
    for (j in 1:length(final_data)) {
      inc <-
        final_incidence_data[
          i,
          (starting_column):(j+(starting_column + inc_window - 2))]
      sd_inc <- sd(as.numeric(inc), na.rm = TRUE)
      mean_inc <- mean(as.numeric(inc), trim = 0, na.rm = TRUE)
      weights <- final_gaussian_weights[j,(1:(j+inc_window - 1))]
      weighted_cv[j] <- sum((sd_inc/mean_inc)*weights)

      weighted_cv_all[i,j] <- weighted_cv[j]

    }
  }

  colnames(weighted_cv_all) <- final_data
  rownames(weighted_cv_all) <- state_names

  smooth_unweighted_cv_all <- weighted_cv_all

  colnames(smooth_unweighted_cv_all) <- final_data
  rownames(smooth_unweighted_cv_all) <- state_names
}


## Find mean incidence and CV and standard deviation of incidence
  mean_cv <-
  vector(length = length(smooth_unweighted_cv_all[1,]))
mean_incidence <-
  vector(length = length(weighted_mean_incidence_all[1,]))
sd_incidence <-
  vector(length = length(weighted_mean_incidence_all[1,]))
if (is.null(state_weights)) {
  for (i in 1:length(mean_cv)) {
    mean_cv[i] <-
    mean(smooth_unweighted_cv_all[,i],
        na.rm = TRUE)
    mean_incidence[i] <-
      mean(weighted_mean_incidence_all[,i])
    sd_incidence[i] <-
      sd(weighted_mean_incidence_all[,i])
  }
    } else {
  for (i in 1:length(mean_cv)) {
  mean_cv[i] <-
    sum(state_weights[,i]*smooth_unweighted_cv_all[,i],
        na.rm = TRUE)
  mean_incidence[i] <-
    sum(state_weights[,i]*weighted_mean_incidence_all[,i])
  sd_incidence[i] <-
    sd(weighted_mean_incidence_all[,i])
}
}

if (find_CI) {
  ## Find the 95% incidence confidence intervals
  CI_incidence <- vector(length = length(mean_incidence))

  for (i in 1:length(mean_cv)) {
    CI_incidence[i] <- qnorm(0.95) * (sd_incidence[i]/nstates)
  }

  upper_CI <- vector(length = length(mean_incidence))
  lower_CI <- vector(length = length(mean_incidence))


  upper_CI <- mean_incidence + CI_incidence
  lower_CI <- mean_incidence - CI_incidence

  ## Data frame containing canonical path data
  data_us <- data.frame(
    CV = as.vector(mean_cv),
    Incidence = as.vector(mean_incidence),
    Upper = upper_CI,
    Lower = lower_CI,
    year = as.character(final_data)
  )
} else {
  ## Data frame containing canonical path data
  data_us <- data.frame(
    CV = as.vector(mean_cv),
    Incidence = as.vector(mean_incidence),
    year = as.character(final_data)
  )
}

## Filter the data frame to include only data
## based on year_labels argument
point_mean_us <- data_us %>%
  filter(year %in% as.character(year_labels))

if (find_states) {
  ## Empty vector for max incidence
  max_incidence_per_state <- vector(length = nstates)

  ## Create CV and Incidence data frames where each
  ## column is a state and each row is a year
  CV_states <-
    as.data.frame(matrix(
      nrow = length(final_data),
      ncol = nstates))
  colnames(CV_states) <- state_names
  CV_states$Year <- final_data

  Incidence_states <-
    as.data.frame(matrix(
      nrow = length(final_data),
      ncol = nstates))
  colnames(Incidence_states) <- state_names
  Incidence_states$Year <- final_data

  ## Create a data frame that will contain
  ## incidence and CV based on year_labels
  Incidence_states_per_10y <- Incidence_states %>%
    filter(Year %in% year_labels)
  CV_states_per_10y <- CV_states %>%
    filter(Year %in% year_labels)

  ## Transpose CV and incidence and adding a Year column
  all_state_cv <- as.data.frame(t(smooth_unweighted_cv_all))
  all_state_cv$Year <- final_data
  all_state_inc <- as.data.frame(t(weighted_mean_incidence_all))
  all_state_inc$Year <- final_data

  ## Create data frames to contain the points every
  ## 10 years starting in 1922
  all_state_cv_per_10y <- all_state_cv %>%
    filter(Year %in% year_labels)

  all_state_inc_per_10y <- all_state_inc %>%
    filter(Year %in% year_labels)

  CV_states <- all_state_cv
  Incidence_states <- all_state_inc

  CV_states_per_10y <- all_state_cv_per_10y
  Incidence_states_per_10y <- all_state_inc_per_10y

  return(list(data_us = data_us, point_mean_us = point_mean_us,
              final_data = final_data,
              CV_states = CV_states,
              Incidence_states = Incidence_states,
              CV_states_per_10y = CV_states_per_10y,
              Incidence_states_per_10y = Incidence_states_per_10y))

} else {
  return(list(data_us = data_us,
              point_mean_us = point_mean_us,
              final_data = final_data))
}
}
```

7.3.2 Plotting the US canonical path

To generate the necessary data required to create the canonical paths, we start by loading all required libraries, load pre-created data files, and source the `functions.R` file.

```
library(stats)
library(smoother) #smth.gaussian()
library(tidyverse)
library(RColorBrewer)
library(knitr)
library(ggrepel)
library(gridExtra)
source('functions.R')
load('Good_data/yearly.data.per.100k.RData')
load("Good_data/pop.estimates.RData")
load("data/presentation_logical.RData")
```

Next, we define the arguments required by the `cv_inc_fun()`.

```
## Vector of state names
state_names <- rownames(final.incidence.data)
nstates <- length(state_names)
## Vector of the years of data we wish to use in
## calculations (1912 as this is when measles became
## nationally notifiable)
usable_data <- seq(1912, 2016, 1)
## How many years to wait until the first data point
## is calculated (same as Graham et al.)
inc_window <- 10
## Standard deviation of the Gaussian filter (same as
## Graham et al.)
sd <- 3
## Location of the center of the Gaussian filter (2
## years prior, same as Graham et al.)
mean_loc <- 2
## Years for which we want to plot a point and Year
## label along the path
year_labels <- seq(1922, 2012, 10)
## Logical values to tell the function whether we want
## state path data and or confidence intervals
find_states <- TRUE
find_CI <- TRUE
## Use gaussian for incidence and smoother for CV
inc_weighting <- "gaussian"
cv_weighting <- "smoothed"
```

Finally, we call the function, without `state_weights` so that the states are equally weighted and save the output for use in plotting.

```r
## Call the cv_inc_fun
cv_inc_results <- cv_inc_fun(data = final.incidence.data,
                             state_names = state_names,
                             all_years = usable_data,
                             inc_window = inc_window,
                             sd = sd, mean_loc = mean_loc,
                             year_labels = year_labels,
                             find_CI = find_CI,
                             find_states = find_states,
                             inc_weighting = inc_weighting,
                             cv_weighting = cv_weighting)

## Save results for use in plotting
data_us <- cv_inc_results$data_us
point_mean_us <- cv_inc_results$point_mean_us
final_data <- cv_inc_results$final_data
CV_states <- cv_inc_results$CV_states
Incidence_states <- cv_inc_results$Incidence_states
CV_states_per_10y <- cv_inc_results$CV_states_per_10y
Incidence_states_per_10y <- cv_inc_results$Incidence_states_per_10y
```

Before creating the plot included in figure 2, we created plots with un-scaled incidence, square-root incidence and log incidence (square-root is used in the final plot). In these plots we also include confidence intervals.

```r
## Canonical path, with y on a normal scale
path_unscaled <- ggplot() +
  geom_path(data = data_us, aes(CV, Incidence),
            colour = 'black', lty = 1, lwd = 1) +
  geom_errorbar(data = data_us,
                aes(CV, Incidence, ymin = Lower, ymax = Upper),
                colour = "black", width = 0.1, inherit.aes = FALSE) +
  scale_y_continuous("Mean Incidence per 100,000",
                     limits = c(min(data_us$Incidence),
                                max(data_us$Incidence))) +
  scale_x_continuous("CV", limits = c(0,3.5)) +
  ggtitle("Un-scaled Incidence")
path_unscaled
```

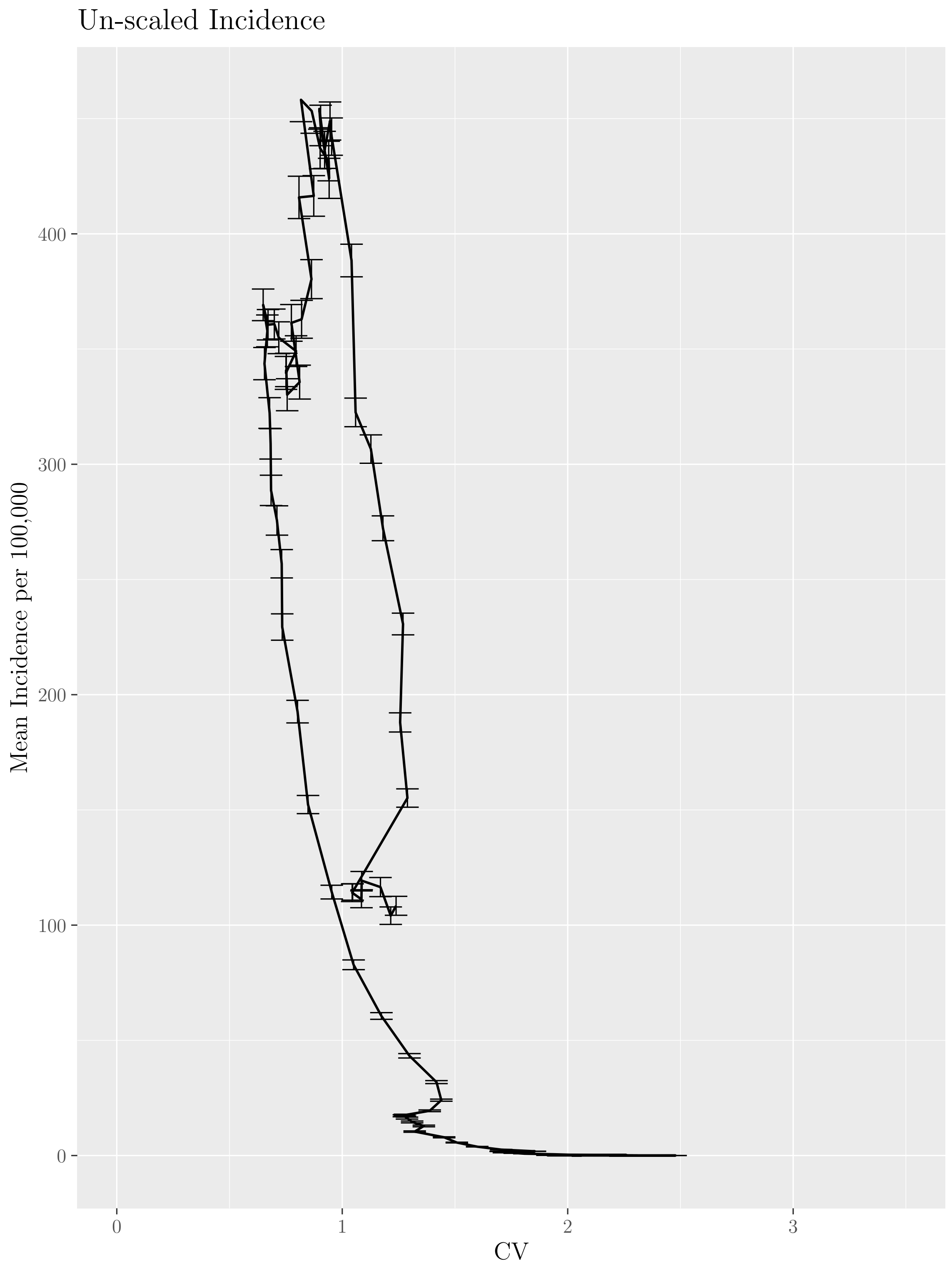


Figure 23: The US canonical path with Incidence un-scaled.

```
## Canonical path, with y on a square root scale
path_sqrt <- ggplot()+
  geom_path(data = data_us, aes(CV, Incidence),
            colour = 'black', lty = 1, lwd = 1) +
  geom_errorbar(data = data_us, aes(CV, Incidence,
                                    ymin = Lower,
                                    ymax = Upper),
                colour = "black", width = 0.1,
                inherit.aes = FALSE) +
  scale_y_sqrt("Mean Incidence per 100,000",
               limits = c(min(data_us$Incidence),
                          max(data_us$Incidence))) +
  scale_x_continuous("CV", limits = c(0,3.5)) +
  ggtitle("Square-root Incidence")
path_sqrt
```

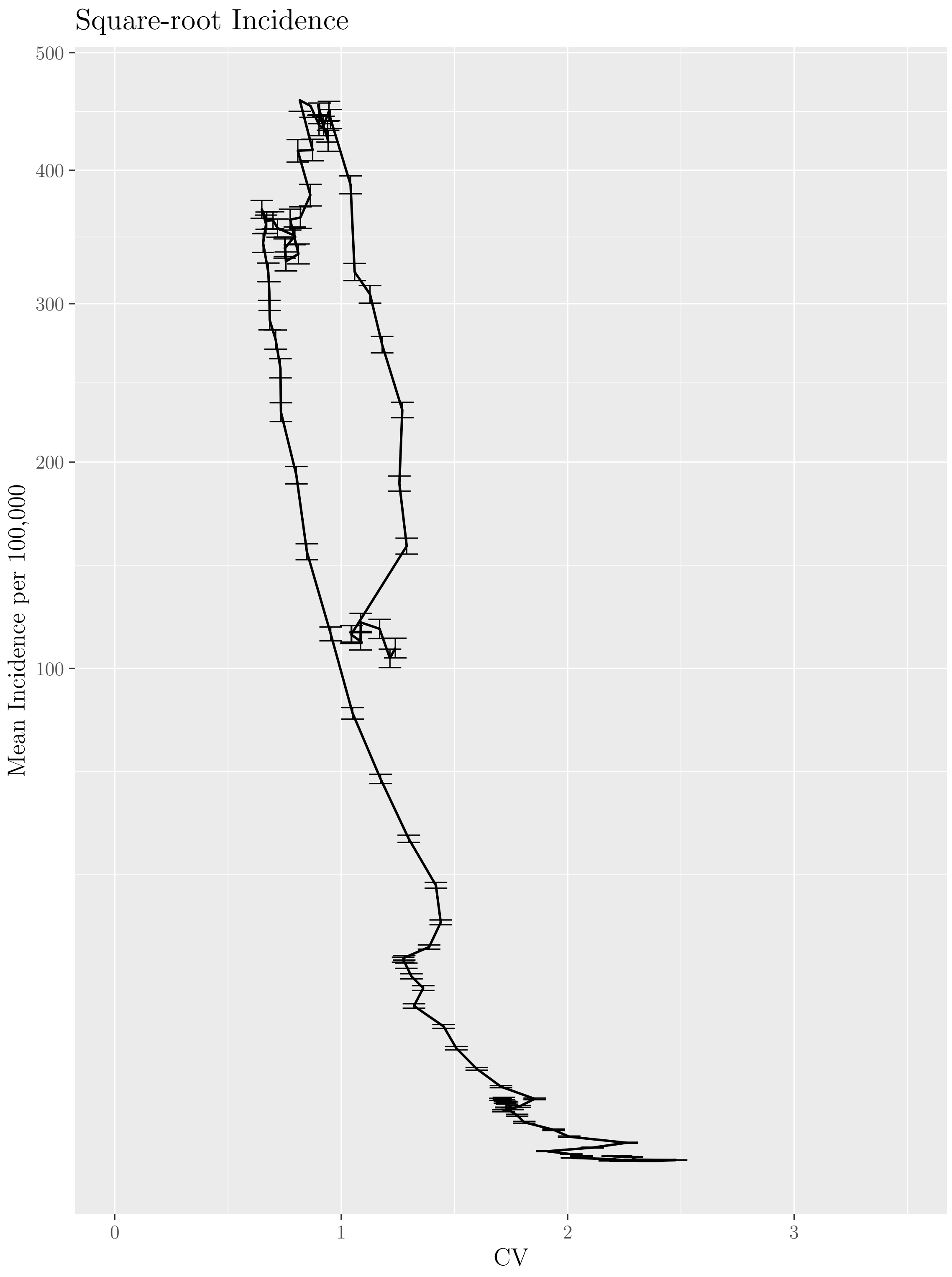


Figure 24: The US canonical path with square-root Incidence.

```
## Canonical path, with y on a log scale
path_log <- ggplot()+
  geom_path(data = data_us, aes(CV, Incidence),
            colour = 'black', lty = 1, lwd = 1) +
  geom_errorbar(data = data_us, aes(CV, Incidence,
                                    ymin = Lower,
                                    ymax = Upper),
                colour = "black", width = 0.1,
                inherit.aes = FALSE) +
  scale_y_log10("Mean Incidence per 100,000",
                limits = c(min(log10(data_us$Incidence)),
                           10)) +
  scale_x_continuous("CV", limits = c(1,3)) +
  ggtitle("Log Incidence")
path_log
```

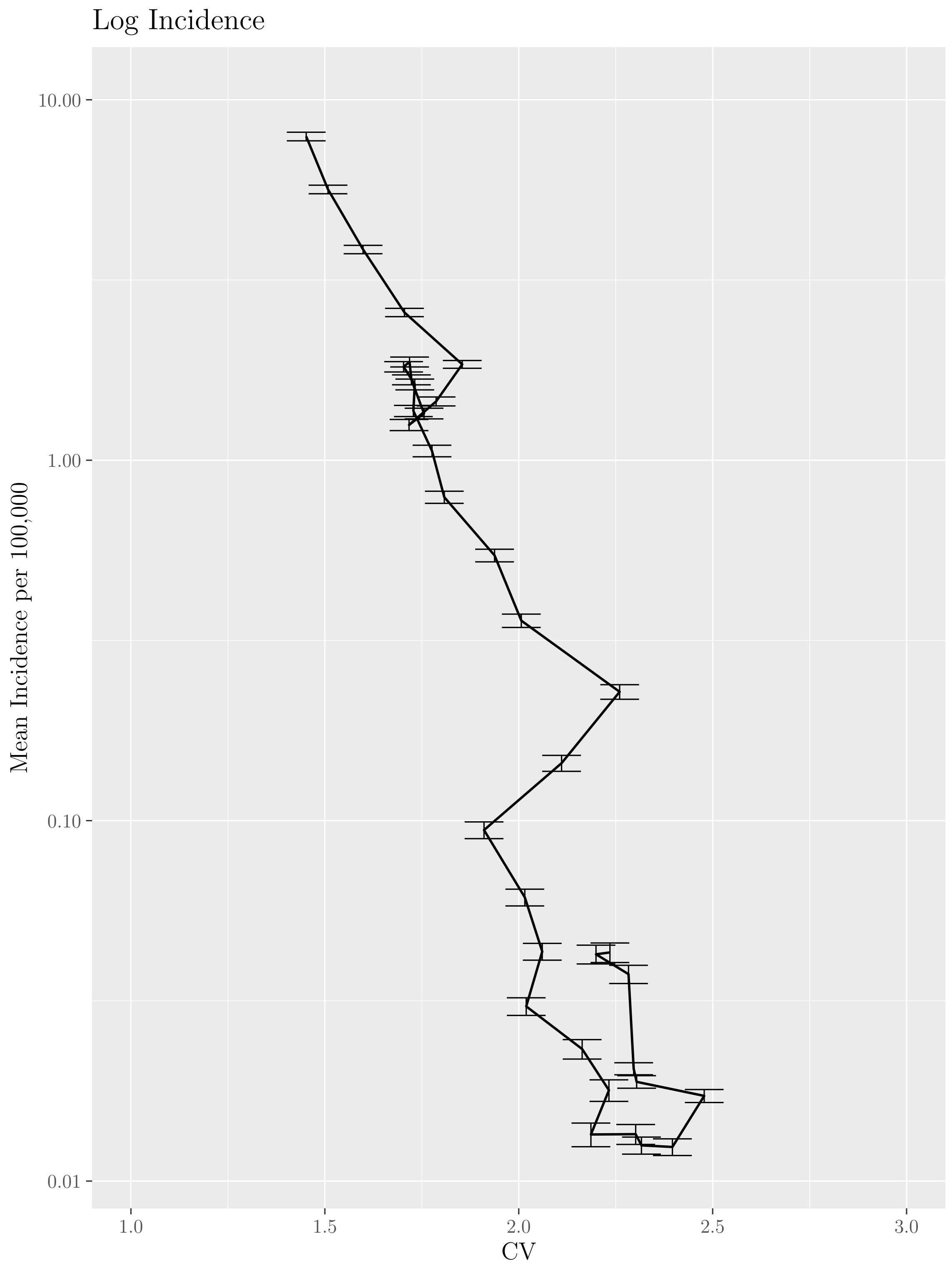


Figure 25: The US canonical path with log Incidence.

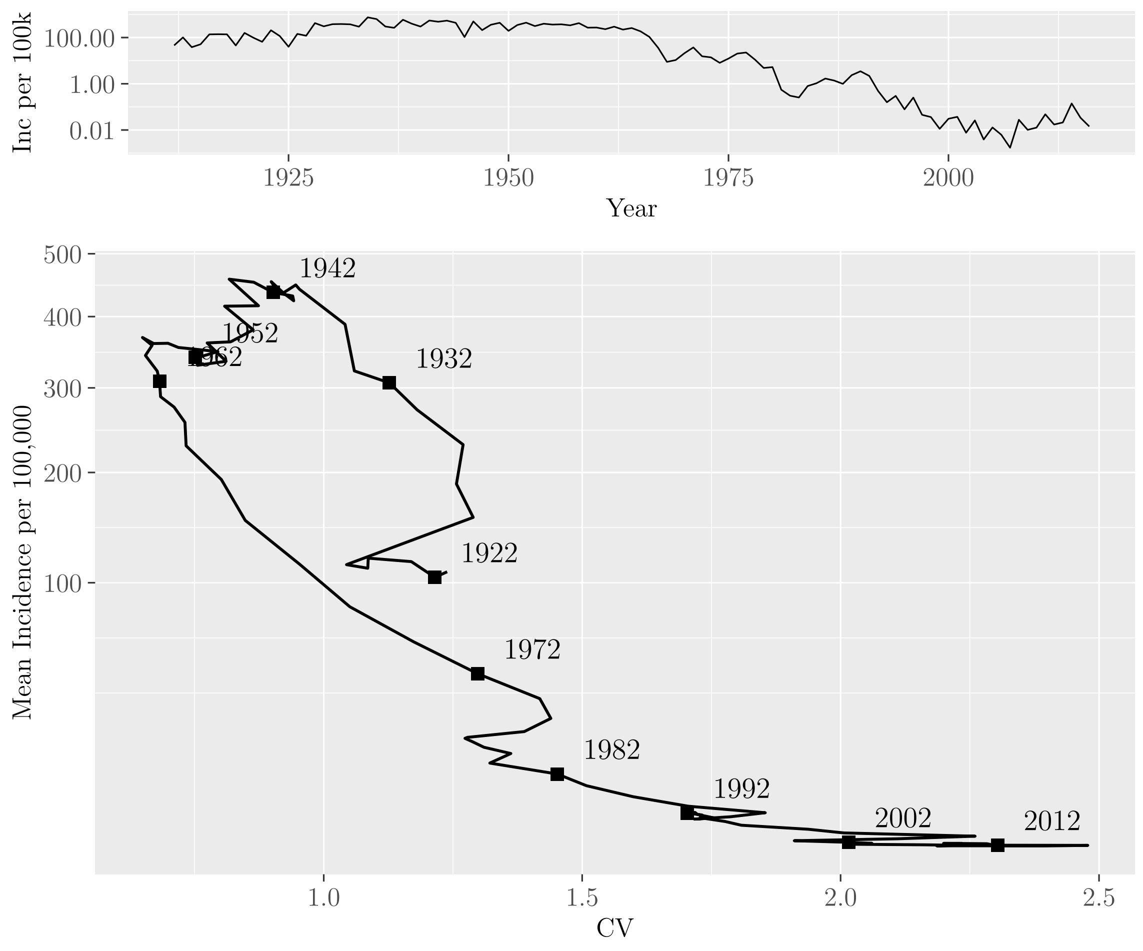
From these plots, we chose to use a square-root scale as it shows more behaviour near zero incidence, while preserving the shape of the path.

Additionally, using the following code, we plot the time series of incidence per 100,000 for the United States. We generate this data by taking the mean across all US states each year.

```
## Create a long format data frame containing
## incidence per 100k of each state and the US since
## 1912
incidence_data <- as.data.frame(t(final.incidence.data))
incidence_data$US <- colMeans(final.incidence.data)
incidence_data$Year <- as.numeric(rownames(incidence_data))

long_incidence_data <- pivot_longer(incidence_data, cols = !Year,
                                    names_to = "Region",
                                    values_to = "Incidence")

long_incidence_data <- long_incidence_data %>%
  filter(Year >= 1912)

## Create a data frame containing only the US
us_only_inc <- long_incidence_data %>%
  filter(Region == "US")

us_ts <- ggplot() +
  geom_path(data = us_only_inc, aes(Year, Incidence)) +
  scale_y_log10("Inc per 100k",
                labels =
                  function(x) format(x, scientific = FALSE)) +
  theme(
    legend.text = element_text(size = text_size),
    legend.title = element_text(size = text_size),
    axis.text = element_text(size = text_size),
    axis.title = element_text(size = text_size)
  )
```

The following code creates the figure shown in figure 2.

```
us_grobs <- list(us_ts, true_path)
us_lay <- rbind(1,2,2,2)
grid.arrange(grobs = us_grobs, layout_matrix = us_lay)
```



7.3.3 US State Path Comparisons

7.3.3.1 Plotting the Canoncial Path Comparisons

To create the plots comparing each state path to the aggregate US path, we begin by defining which political stronghold each state represents. This allows us to colour each state path based on political stronghold: republican, democratic, or swing state. The data was obtained from WorldAtlas [21–23].

```
## Create vectors defining the type of political stronghold of each state
## Data retrieved from the following sites
## https://www.worldatlas.com/articles/states-that-have-voted-republican-in-the-m
## https://www.worldatlas.com/articles/states-that-have-consistently-remained-dem
## https://www.worldatlas.com/articles/which-states-are-swing-states.html
```

```r
red_states <- c("alabama", "alaska", "arizona", "arkansas", "georgia",
                "idaho", "kansas", "kentucky", "louisiana",
                "mississippi","missouri", "montana", "nebraska",
                "north dakota","oklahoma", "south carolina",
                "south dakota","tennessee", "texas", "utah",
                "west virginia","wyoming")
red_states <- toupper(red_states)

swing_states <- c("colorado", "florida", "iowa", "michigan",
                  "minnesota")
swing_states <- toupper(swing_states)

blue_states <-
  state_names[!(state_names %in% c(red_states, swing_states))]
```

Next, we create several lists that will contain data used to generate the state comparison plots as well as the plots themselves. We also define a vector containing the years we wish to label. Additionally, we add a Path column to the US data frames, while filtering them to only include the necessary columns. Finally, we create a data frame that only contains data for the years we wish to label.

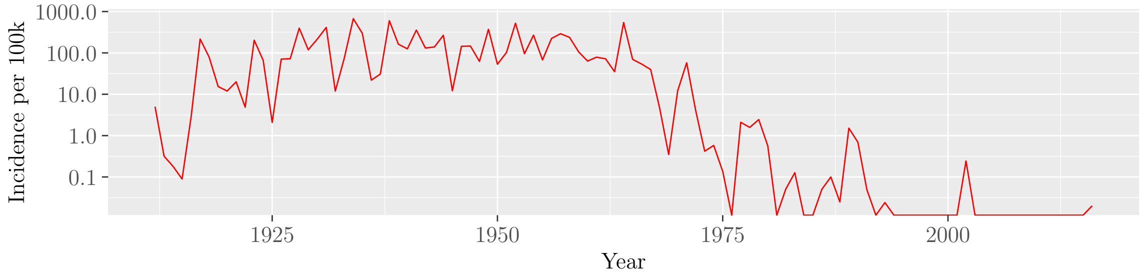
Once the US data is prepared, we run a for loop that creates data frames containing the Incidence and CV of a single state, using the output of the `cv_inc_fun()`, while defining max incidence of that state, and save the generated data frames to items of a list. We then combine the US and state data frames for use in plotting. We also create a named vector that tells `ggplot` which colour to use for each path based on whether it is of the entire US or a single state. We save the named vector to a list for each state to use in later code. The next argument of the for loop generates a plot comparing the US canonical path to that of an individual state and saves it to a list. Lastly, the for loop creates a plot of the time series of the corresponding state and saves it to a list. The names of all lists are set equal to the vector of state names.

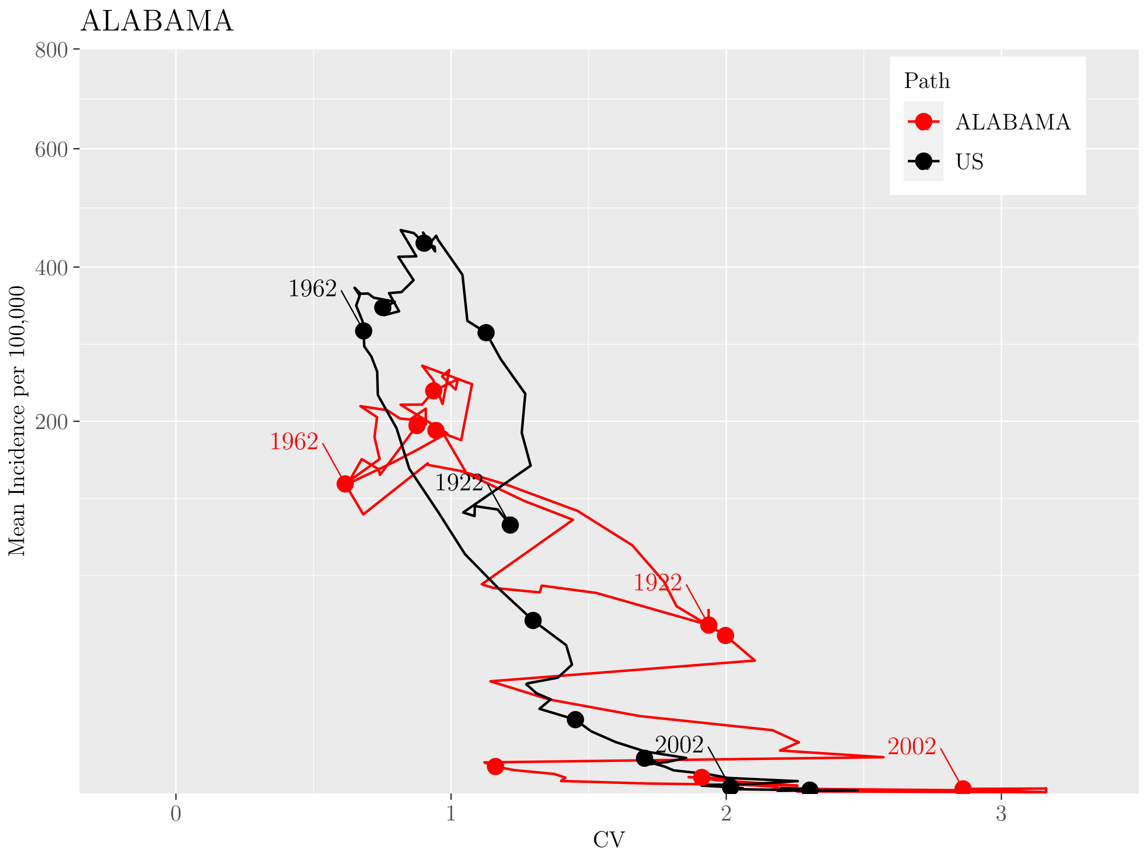
```r
## Empty list that will contain the paths
state_paths <- list()
state_data_list <- list()
state_point_list <- list()
state_text_list <- list()
state_ts_list <- list()
colour_each_state <- list()
max_incidence_per_state <- vector(length = nstates)

## Vector of years to label
```

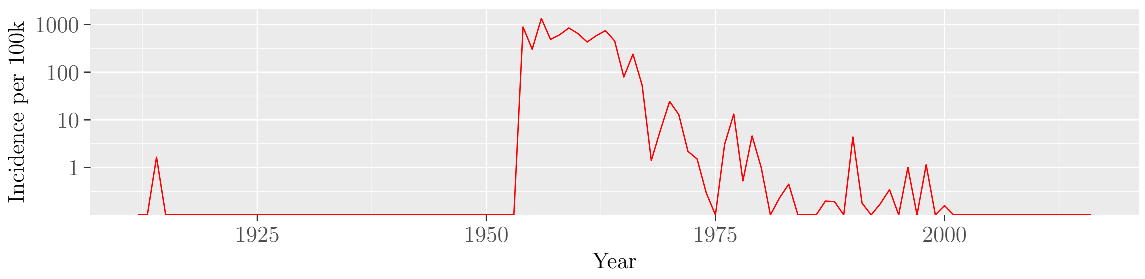
```r
label_years <- c("1922", "1962", "2002")

## Create a Path column that identifies
## that it is the US path
data_us$Path <- "US"
point_mean_us$Path <- "US"

## Filter the data to only containing the
## necessary columns and rename the columns
data_us_filter <-
  data_us[colnames(data_us) %in% c("CV", "Incidence",
                                   "year", "Path")]
colnames(data_us_filter) <-
  c("CV", "Incidence", "Year", "Path")
point_mean_us_filter <-
  point_mean_us[colnames(point_mean_us) %in% c("CV",
                                               "Incidence",
                                               "year",
                                               "Path")]
colnames(point_mean_us_filter) <-
  c("CV", "Incidence", "Year", "Path")

## Create a data frame filtered for the years we wish to label
us_text_data <- point_mean_us_filter %>%
    filter(Year %in% label_years)

for (i in 1:nstates) {
  ## Define the current state
  state <- state_names[i]

  ## Create the data frames
  ## that contain the CV and incidence for a single state
  state_data <- data.frame(
    CV = CV_states[,i] ,
    Incidence = Incidence_states[,i],
    Year = Incidence_states$Year,
    Path = state
  )

  point_state_data <- data.frame(
    CV = CV_states_per_10y[,i],
    Incidence = Incidence_states_per_10y[,i],
    Year = Incidence_states_per_10y$Year,
    Path = state
  )

  state_text_data <- point_state_data %>%
    filter(Year %in% label_years)

  ## Find the max incidence for each state
  max_incidence_per_state[i] <-
    max(state_data$Incidence, na.rm = TRUE)

  ## Save data frames for later
  state_data_list[[i]] <- state_data
  state_point_list[[i]] <- point_state_data
  state_text_list[[i]] <- state_text_data



  ## Create long data frames containing the US and state path data
  all_data <- rbind(data_us_filter, state_data)
  all_point_data <- rbind(point_mean_us_filter, point_state_data)
  all_text_data <- rbind(us_text_data, state_text_data)

  ## Determine the colour of the state path
  ## based on the type of political stronghold
  if (state %in% red_states) {
    state_col <- "red"
  } else if (state %in% blue_states) {
    state_col <- "blue"
  } else {
    state_col <- "purple4"
  }

  ## Manually choose the colours of the state and US path
  colours <- c(state_col, "black")
  names(colours) <- c(state, "US")

  colour_each_state[[i]] <- colours

  ## Create the plot that plots both the state and US canonical path
  state_plot <- ggplot() +
    geom_path(data = all_data, aes(CV, Incidence, colour = Path),
              lty = 1, lwd = 1) +
    geom_point(data = all_point_data,
               aes(CV, Incidence, colour = Path),
               size = 3) +
    geom_text_repel(data = all_text_data,
                    aes(label = Year, '', colour = Path),
            x = all_text_data$CV,
            y = sqrt(all_text_data$Incidence),
            nudge_x = 1,
            nudge_y = 1,
            hjust = -0.45,
            vjust = -0.75
            ) +
    scale_y_sqrt("Mean Incidence per 100,000",
                 limits = c(min(all_data$Incidence, na.rm = TRUE),
                            800)) +
    scale_x_continuous("CV",
                       limits = c(-.35, 3.5)) +
    scale_colour_manual(values = colours) +
    coord_cartesian(expand = c(0,0)) +
    labs(colour="Path") +
    theme(legend.position = c(.95, .99),
          legend.justification = c("right", "top"),
          legend.box.just = "right",
          legend.margin = margin(6, 6, 6, 6),
          legend.text = element_text(size = text_size),
          legend.title = element_text(size = text_size),
          axis.text = element_text(size = text_size),
          axis.title = element_text(size = text_size)
    ) +
    ggtitle(state)

  state_inc_data <- long_incidence_data %>%
    filter(Region == state)

  state_inc_data$Path <- state

  state_ts <- ggplot() +
    geom_path(data = state_inc_data,
              aes(Year, Incidence, col = Path)) +
    scale_colour_manual(values = colours) +
    scale_y_log10(labels =
                    function(x) format(x, scientific = FALSE)) +
    ylab("Incidence per 100k") +
    theme(
      legend.position = "none",
      legend.text = element_text(size = text_size),
      legend.title = element_text(size = text_size),
      axis.text = element_text(size = text_size),
      axis.title = element_text(size = text_size)
    )

  ## Assign the plots to an entry in a list to use in other code
  state_paths[[i]] <- state_plot
  state_ts_list[[i]] <- state_ts
}

## Assign the state names to be the names of the list
names(state_paths) <- state_names
names(state_ts_list) <- state_names
names(state_data_list) <- state_names
names(state_point_list) <- state_names
names(state_text_list) <- state_names
names(colour_each_state) <- state_names
```

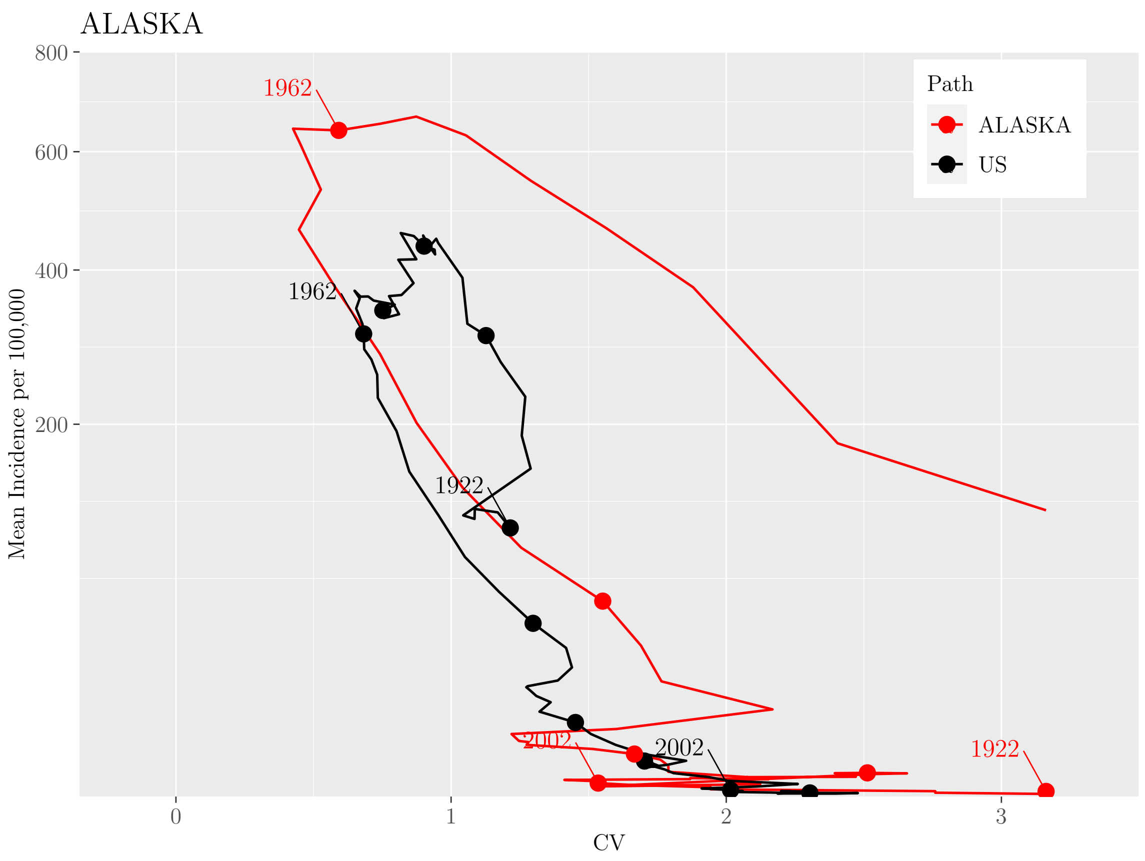
The following are the plots comparing each of the US states' canonical path to that of the entire US. First we define several arguments required for the `grid.arrange()` function: a list of `ggplot` items that contain the canonical path comparisons and the layout matrices that tell `grid.arrange` how to arrange the plots. Finally, we call `grid.arrange()` to create the plot grids.

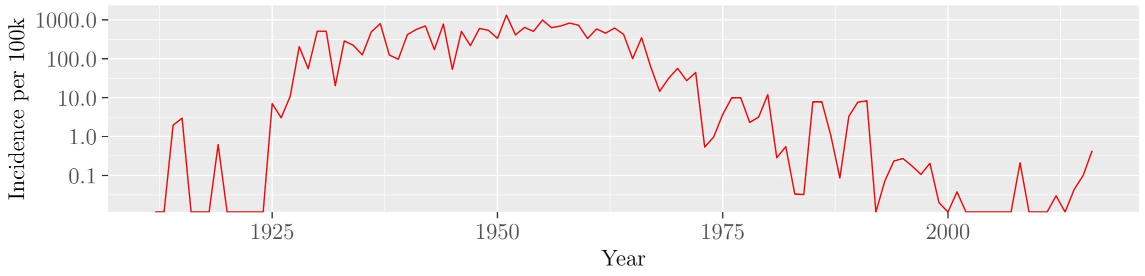
```
for (i in 1:nstates) {
  grobs_state <- list(state_ts_list[[i]], state_paths[[i]])
  lay_state <- rbind(1, 2, 2, 2)
  grid.arrange(grobs = grobs_state, layout_matrix = lay_state)
}
```

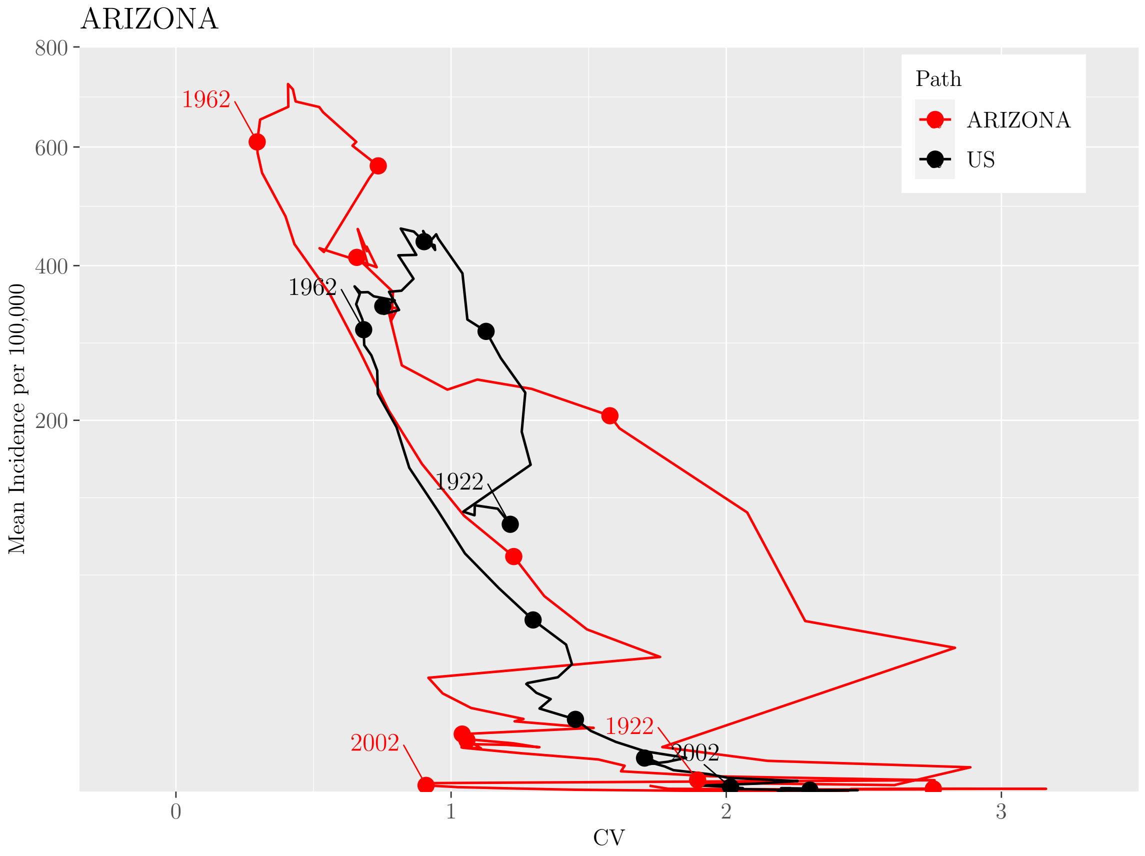

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

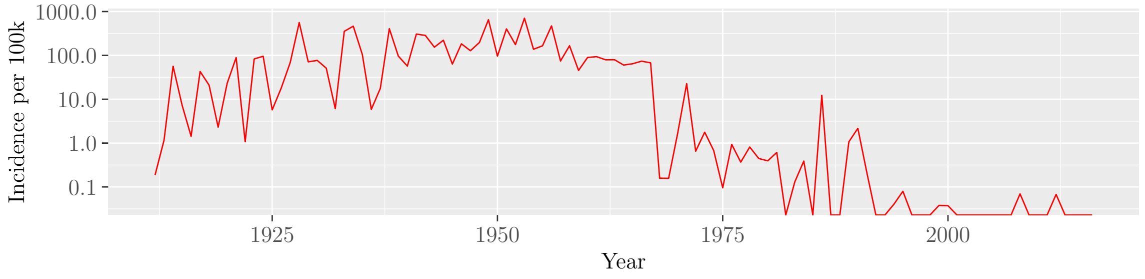

ALABAMA
Mean Incidence per 100,000
800
600
400
200
CV
0
1
2
3
Path
ALABAMA
US
1962
1922
2002
1962
1922
2002

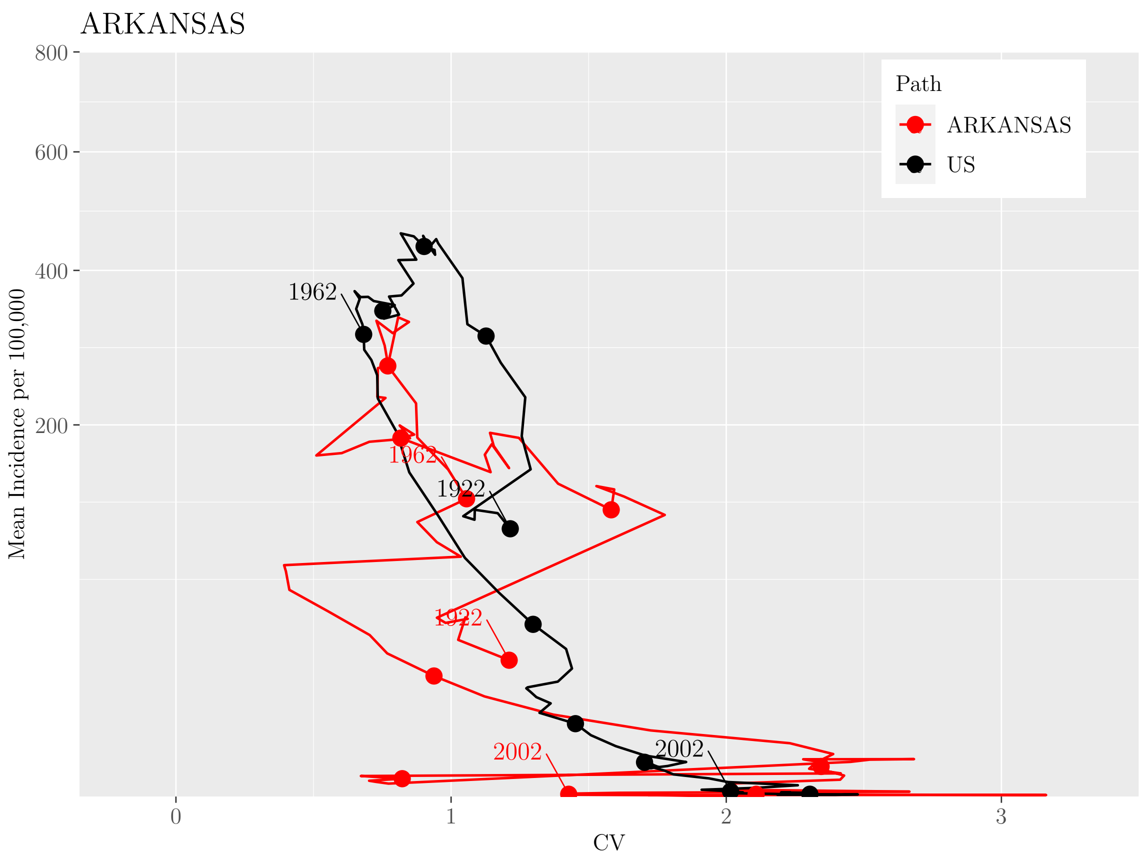

Incidence per 100k
1000
100
10
1
1925
1950
1975
2000
Year

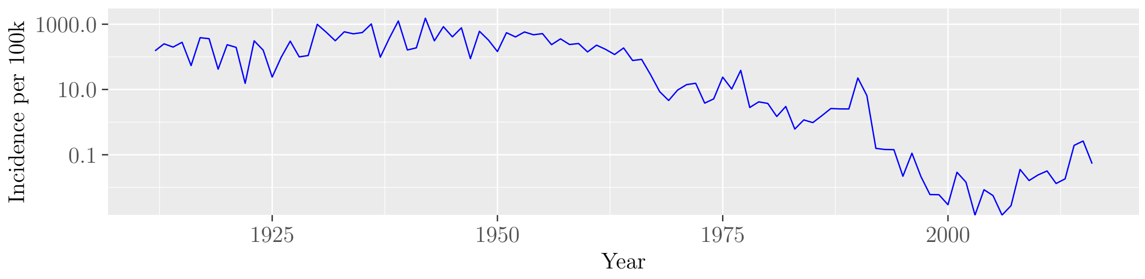

ALASKA
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
1962
1962
1922
1922
2002
2002
Path
ALASKA
US

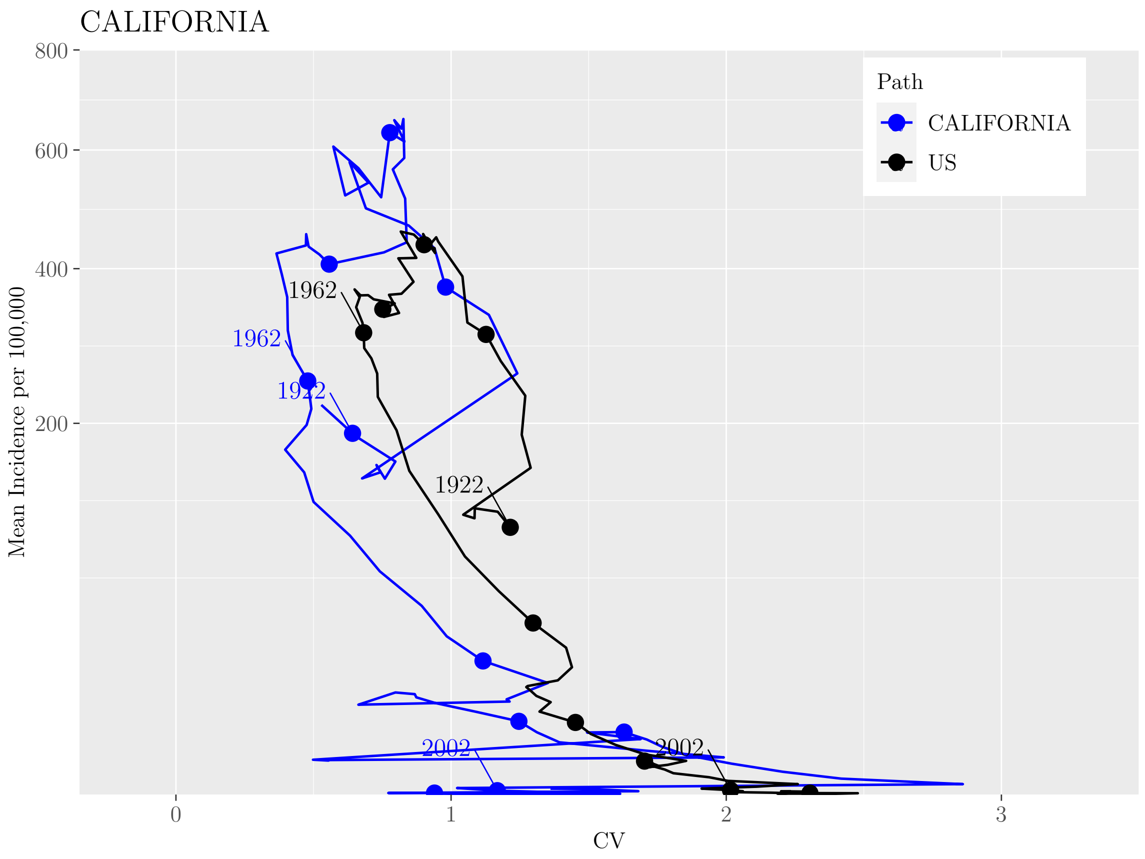

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

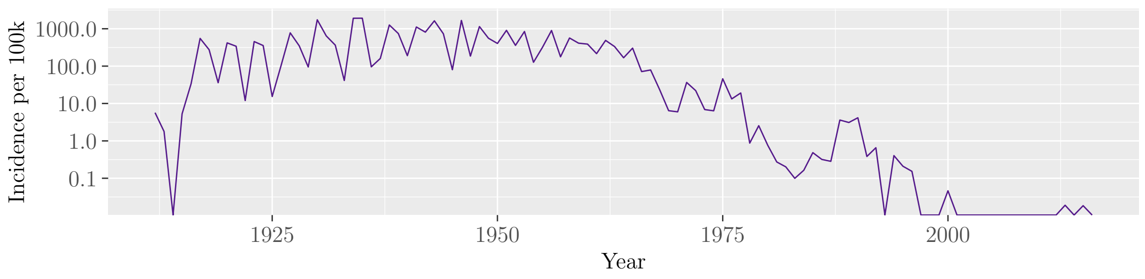

ARIZONA
800
600
400
200
Mean Incidence per 100,000
1962
1962
1922
1922
2002
2002
0
1
2
3
CV
Path
ARIZONA
US

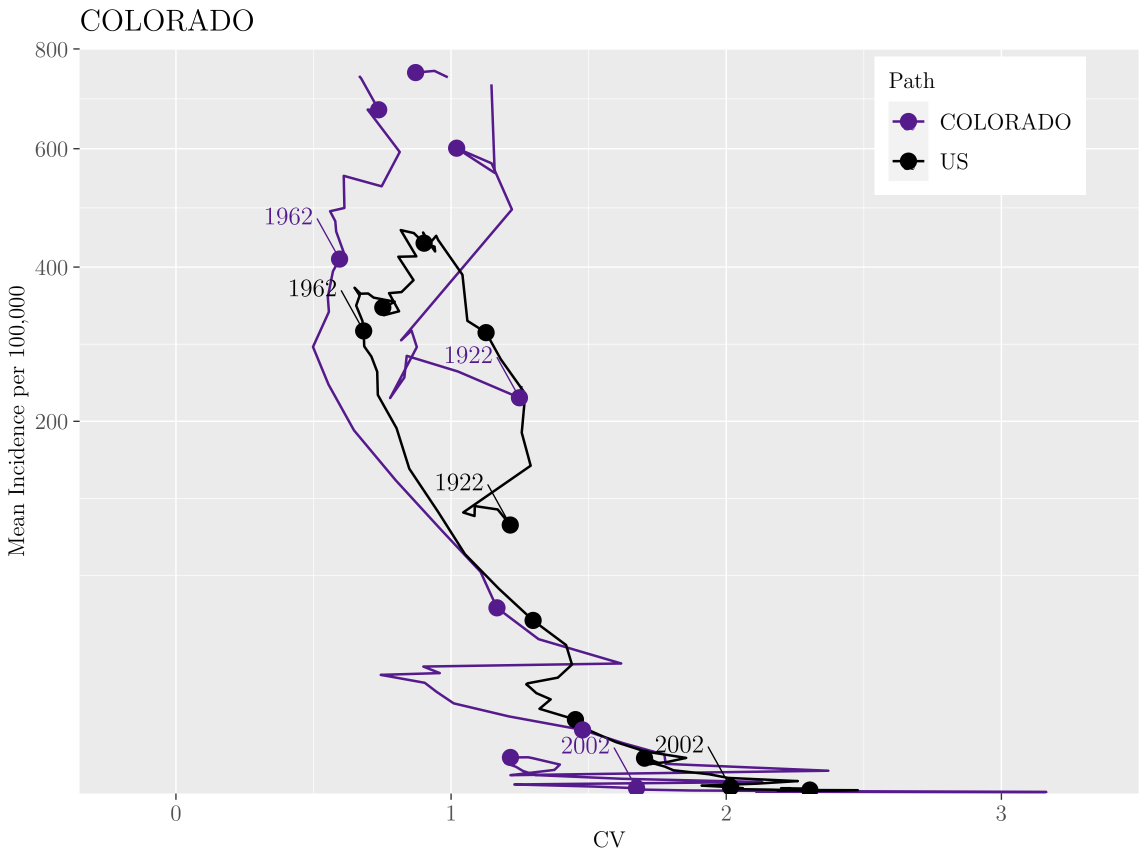

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

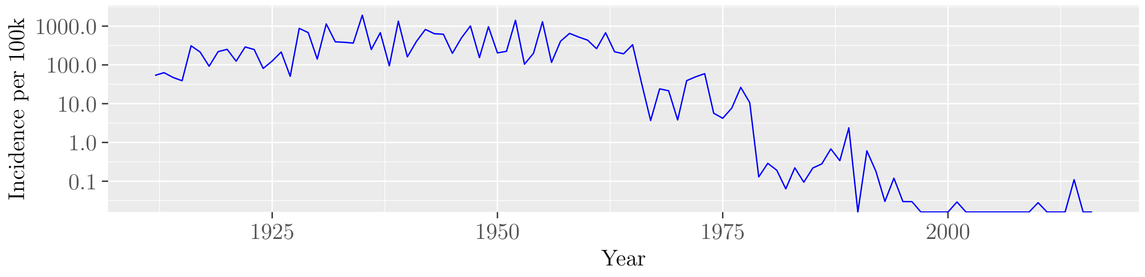

ARKANSAS
800
600
400
200
Mean Incidence per 100,000
Path
ARKANSAS
US
1962
1922
2002
1962
1922
2002
0
1
2
3
CV

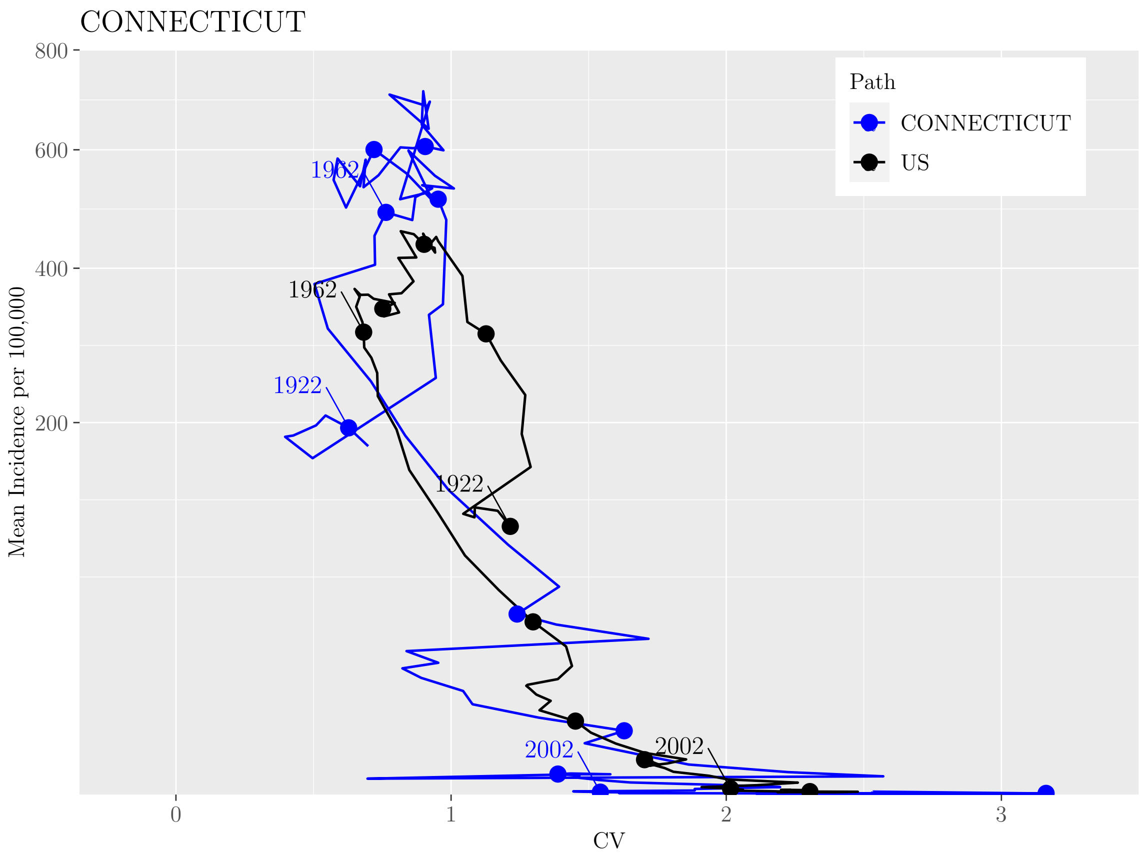

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

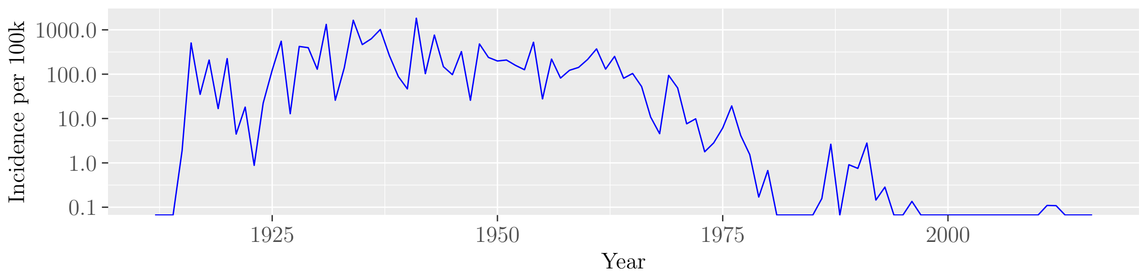

CALIFORNIA
800
600
400
200
Mean Incidence per 100,000
Path
CALIFORNIA
US
1962
1962
1922
1922
2002
2002
0
1
2
3
CV

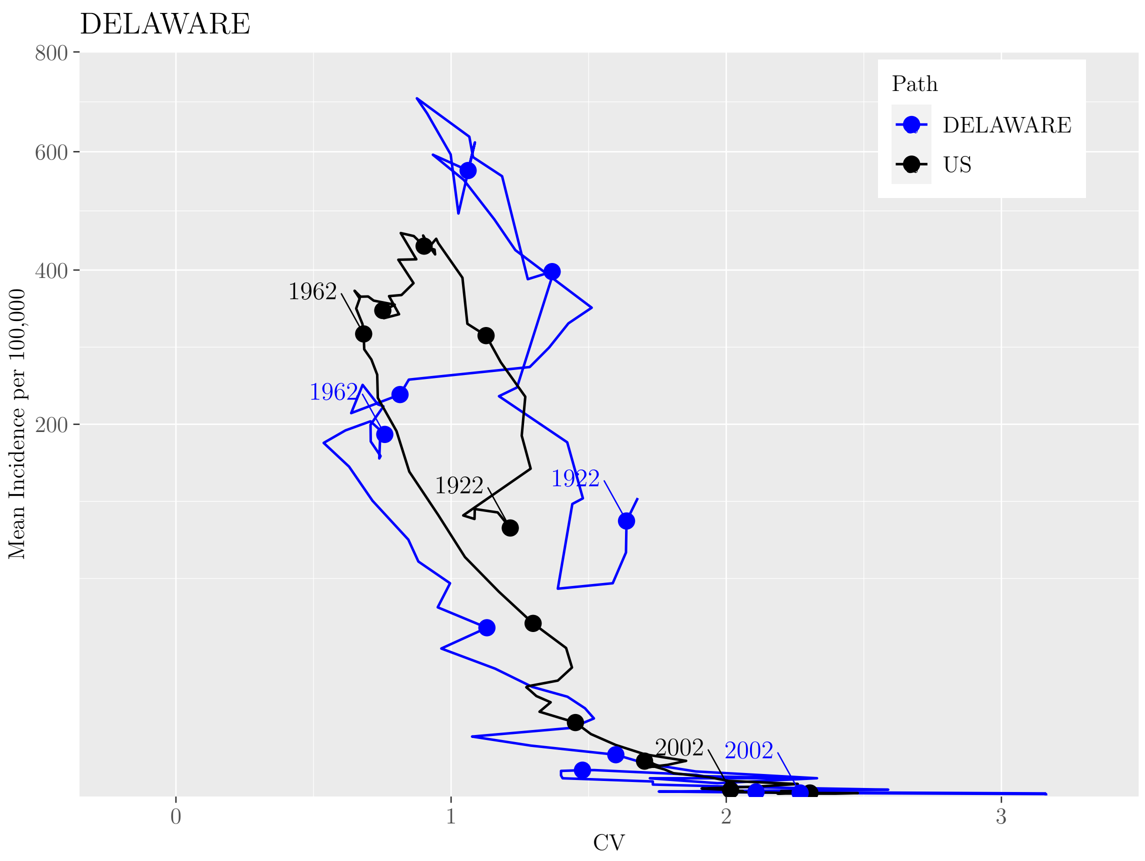

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

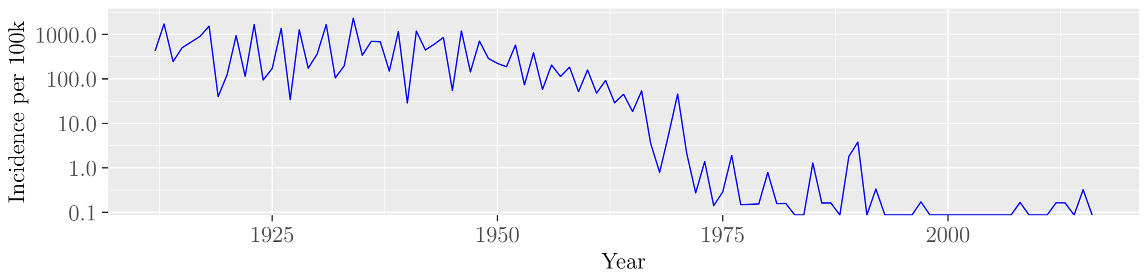

COLORADO
800
600
400
200
Mean Incidence per 100,000
1962
1962
1922
1922
2002
2002
0
1
2
3
CV
Path
COLORADO
US

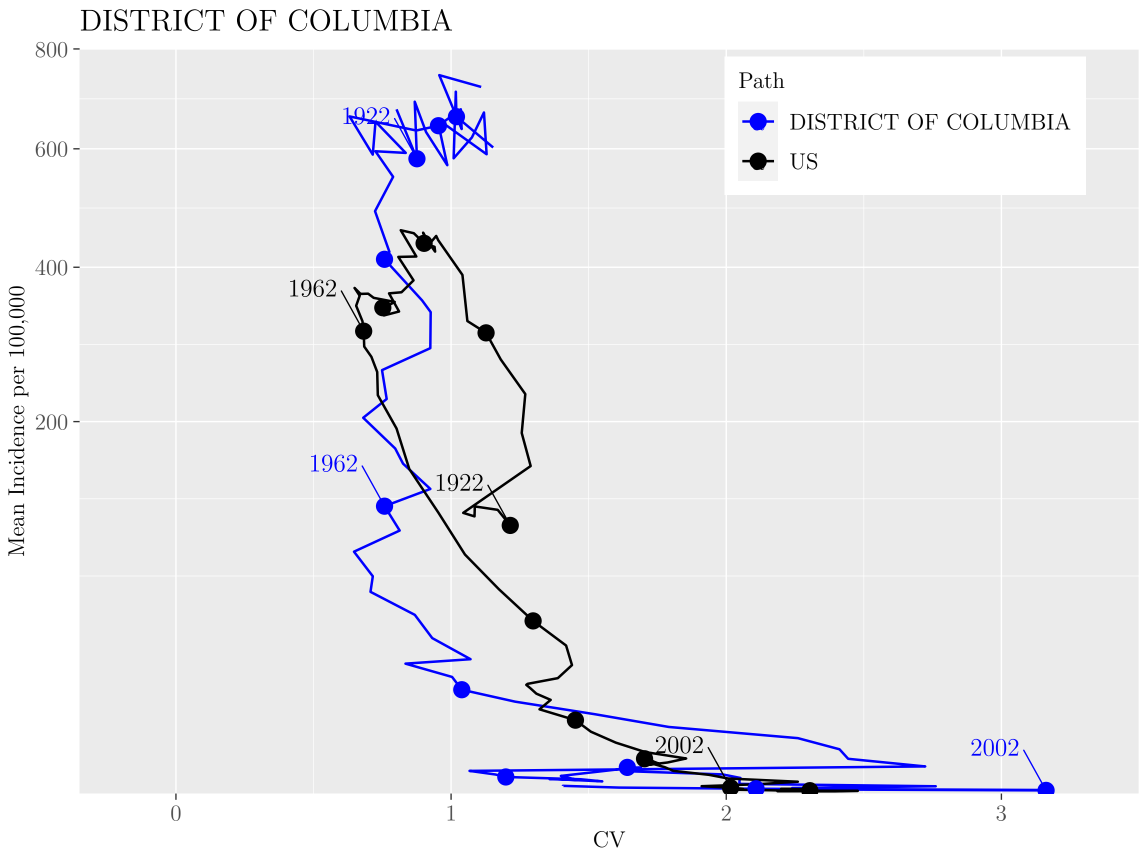

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

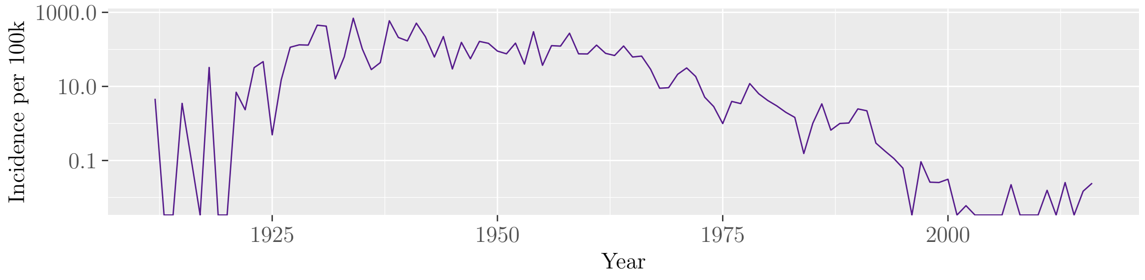

CONNECTICUT
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
CONNECTICUT
US
1962
1922
2002
1962
1922
2002

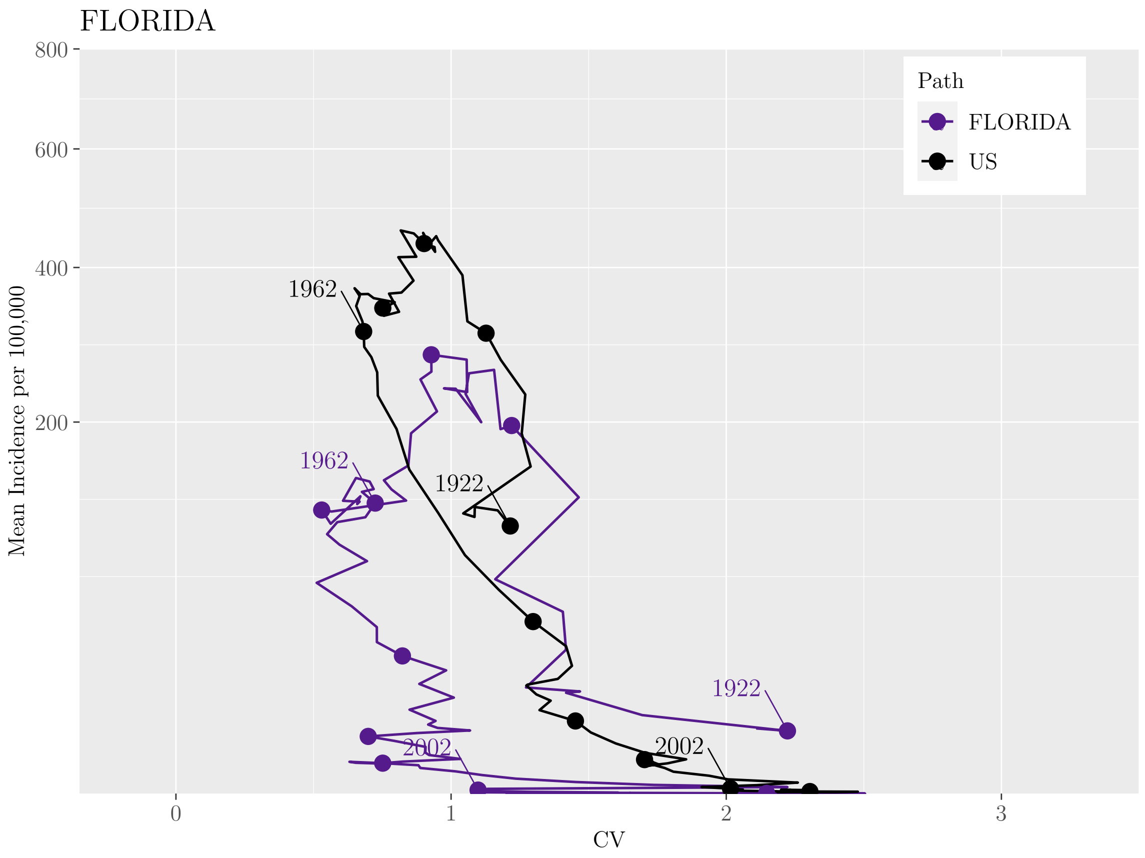

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

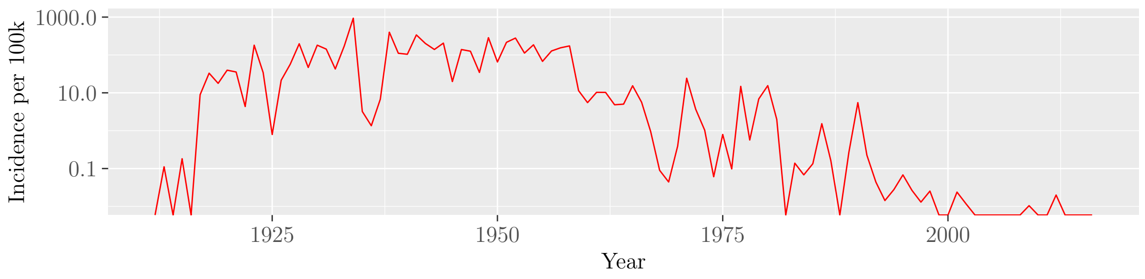

DELAWARE
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
DELAWARE
US
1962
1962
1922
1922
2002
2002

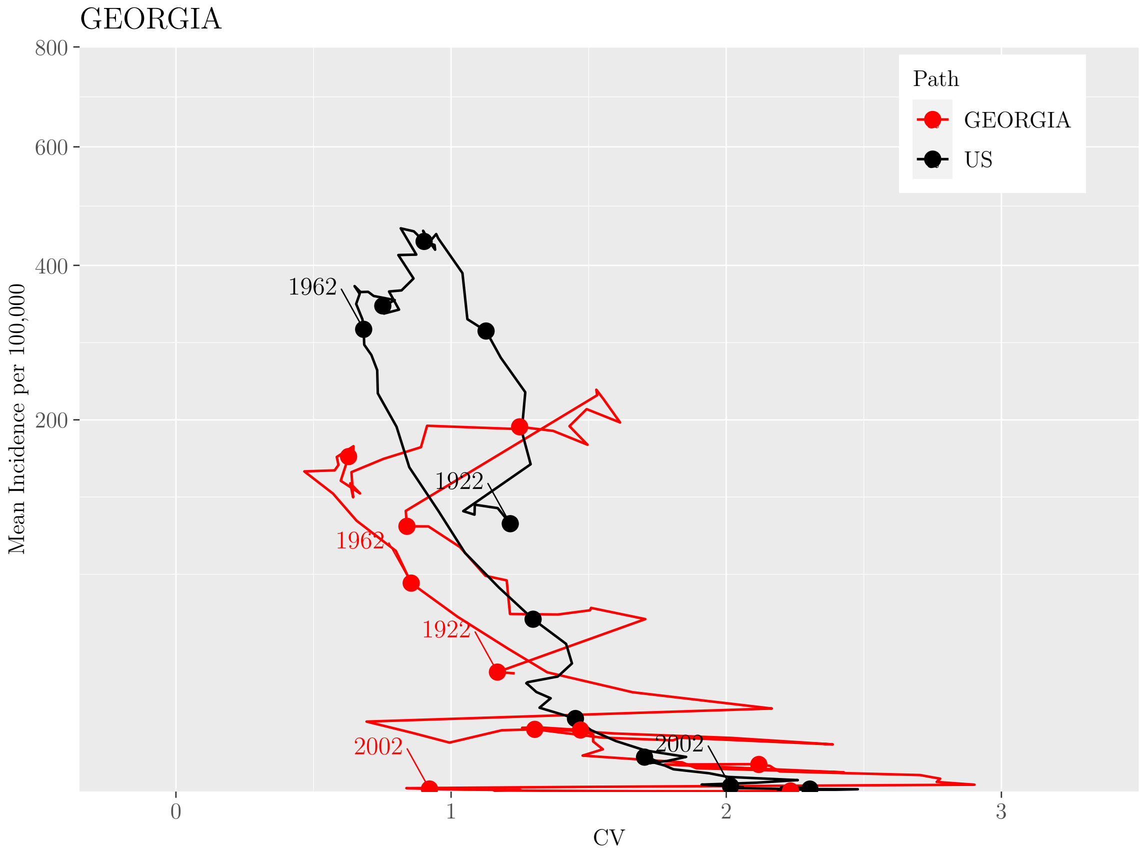

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

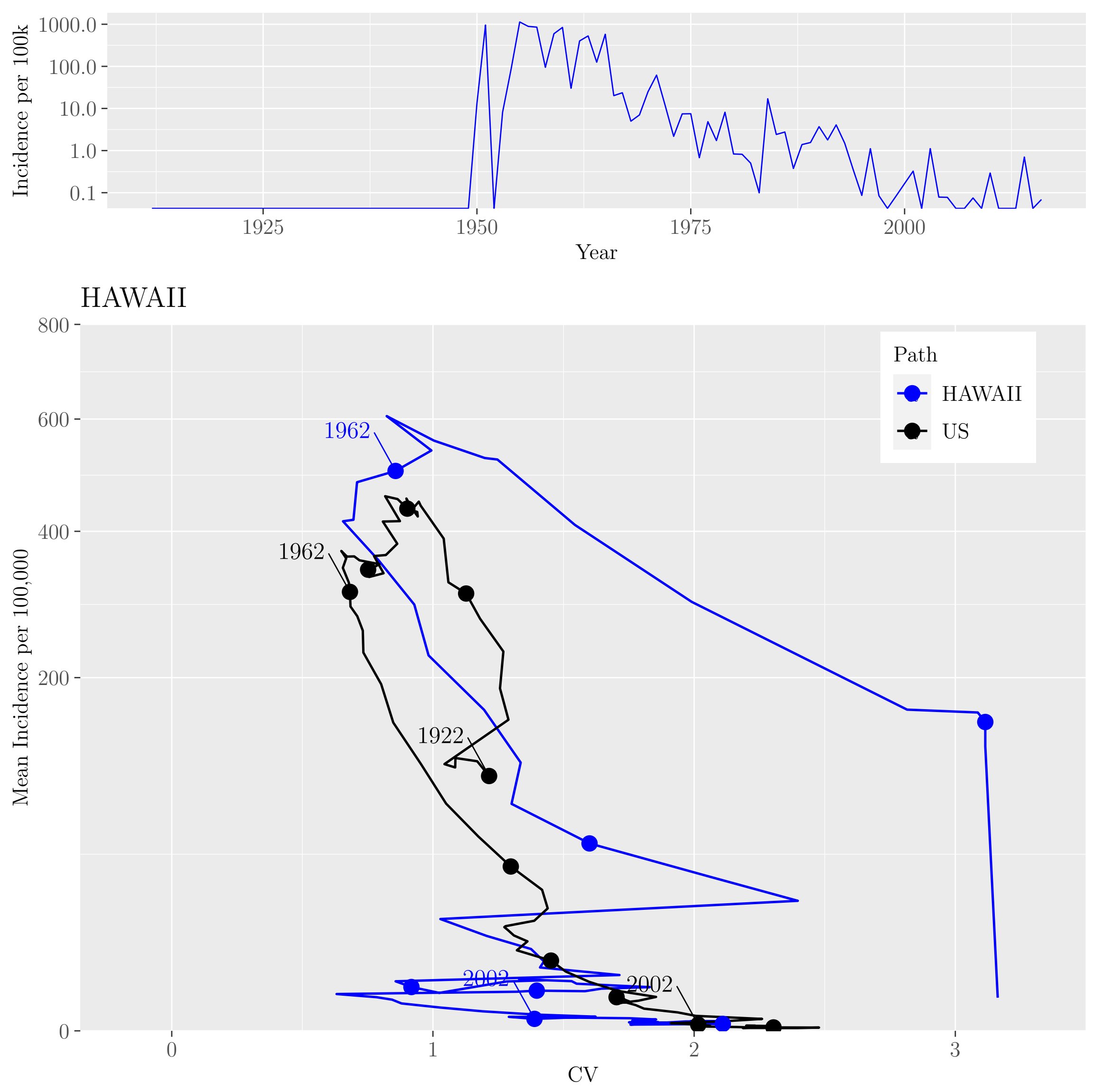

DISTRICT OF COLUMBIA
Mean Incidence per 100,000
800
600
400
200
CV
0
1
2
3
Path
DISTRICT OF COLUMBIA
US
1922
1962
2002
1962
1922
2002

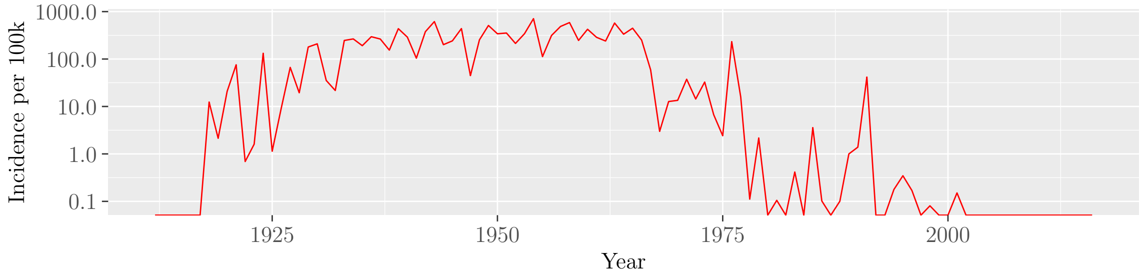

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

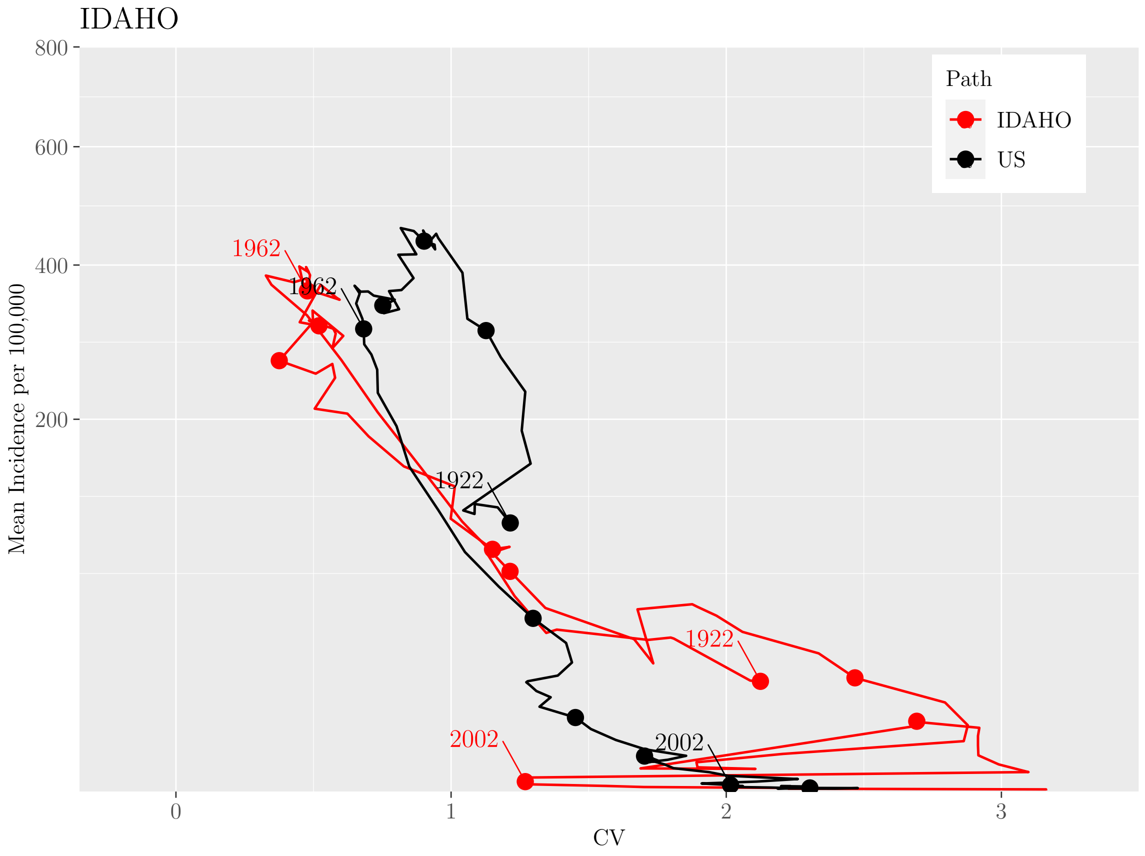

FLORIDA
800
600
400
200
Mean Incidence per 100,000
Path
FLORIDA
US
1962
1962
1922
1922
2002
2002
0
1
2
3
CV

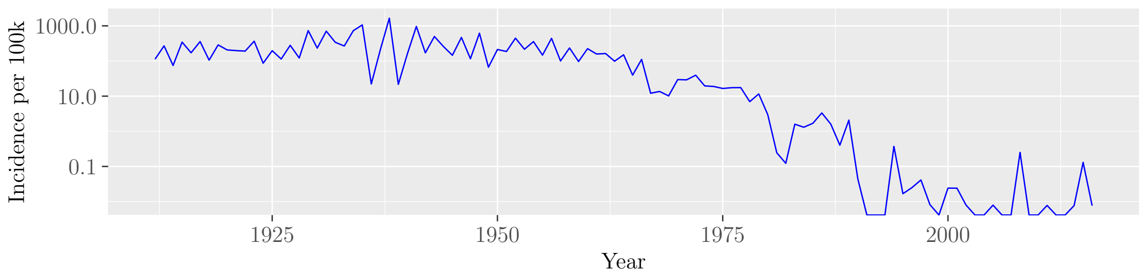

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

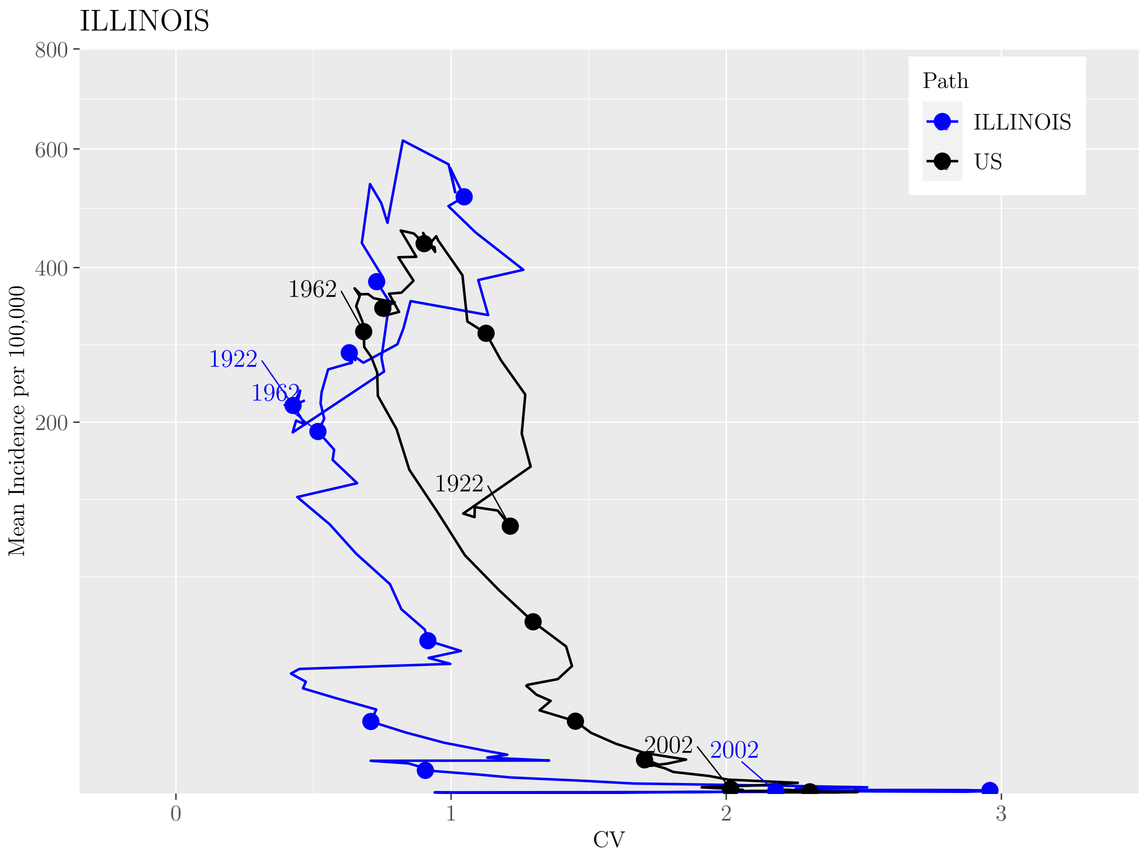

GEORGIA
800
600
400
200
Mean Incidence per 100,000
Path
GEORGIA
US
1962
1922
1962
1922
2002
2002
0
1
2
3
CV

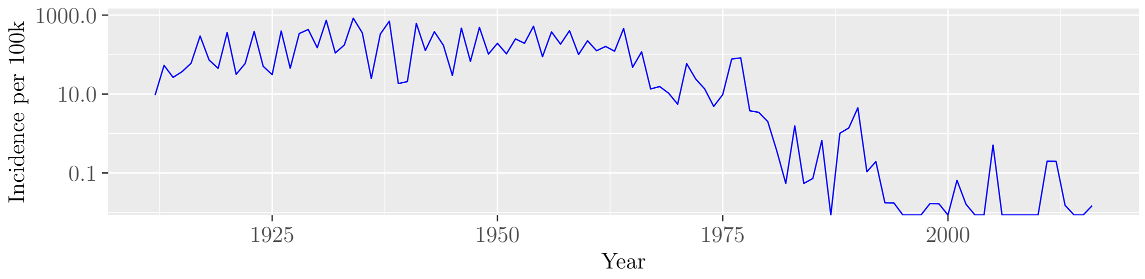

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year
HAWAII
Mean Incidence per 100,000
800
600
400
200
0
CV
0
1
2
3
Path
HAWAII
US
1962
1962
1922
2002
2002

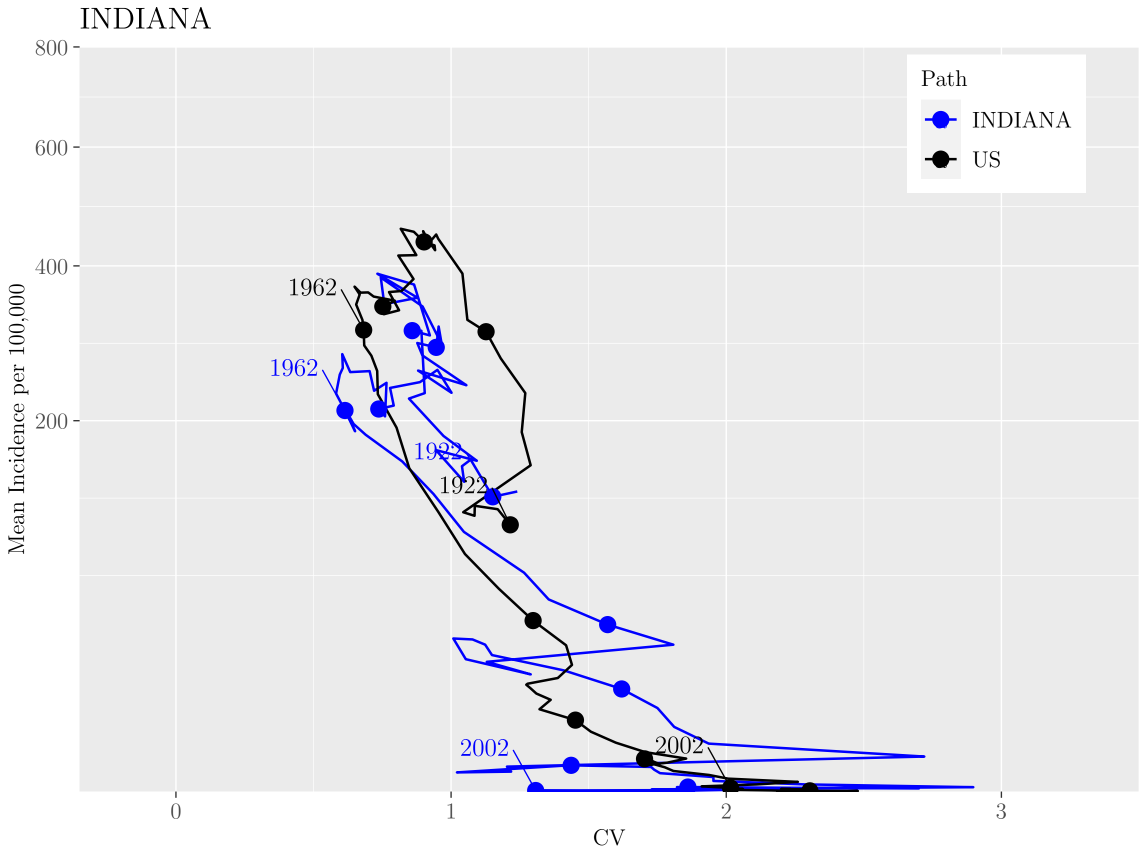

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

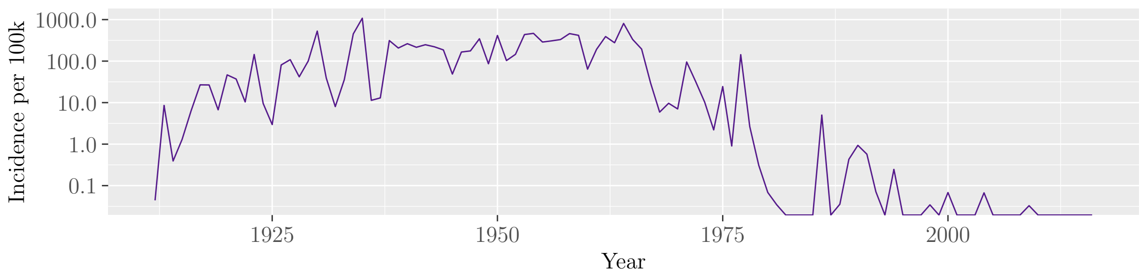

IDAHO
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
IDAHO
US
1962
1962
1922
1922
2002
2002

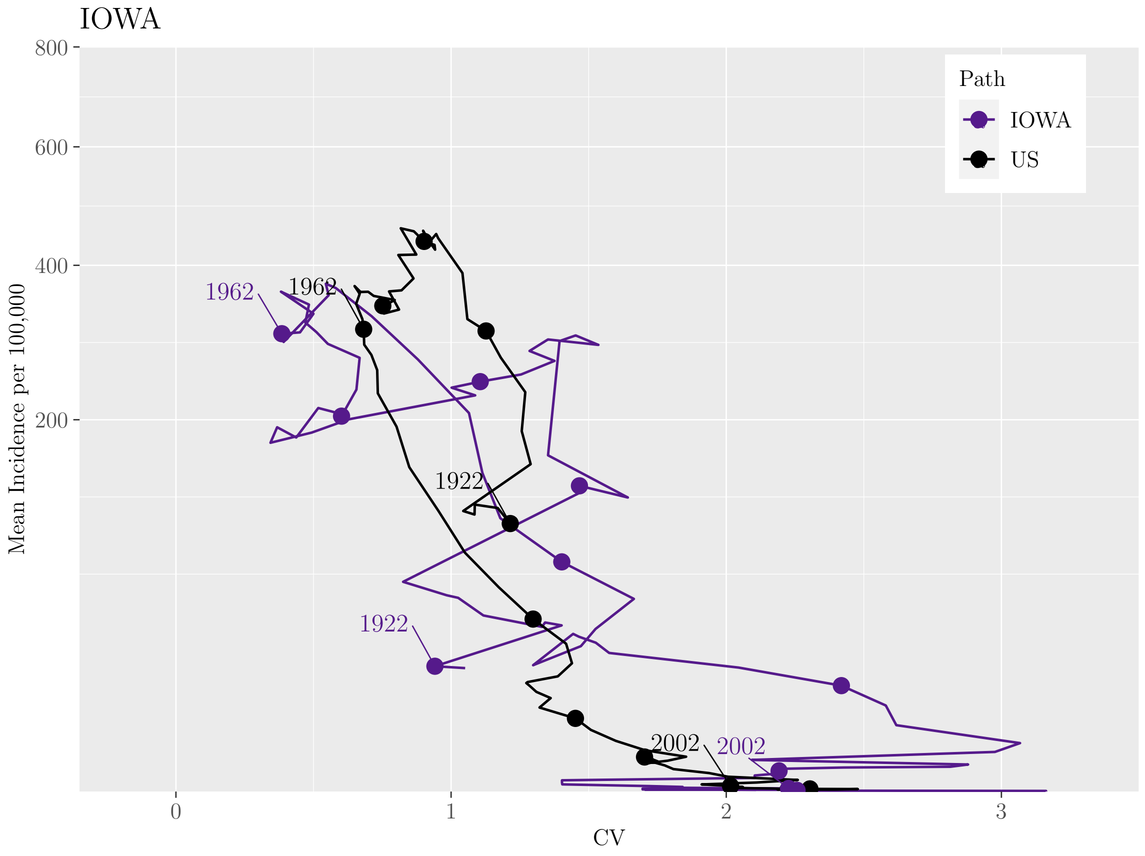

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

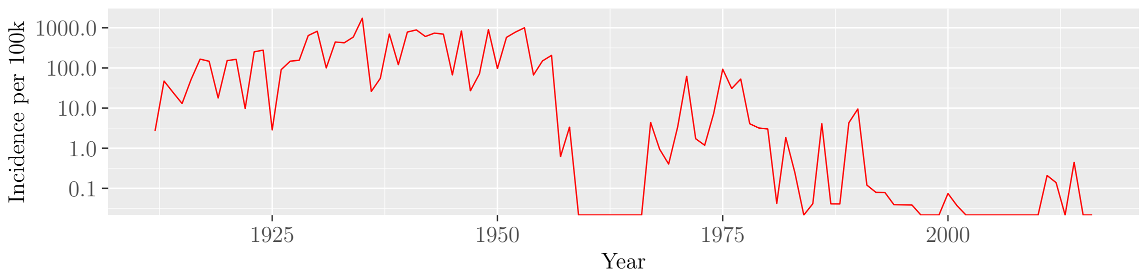

ILLINOIS
Mean Incidence per 100,000
800
600
400
200
CV
0
1
2
3
Path
ILLINOIS
US
1962
1922
1962
1922
2002
2002

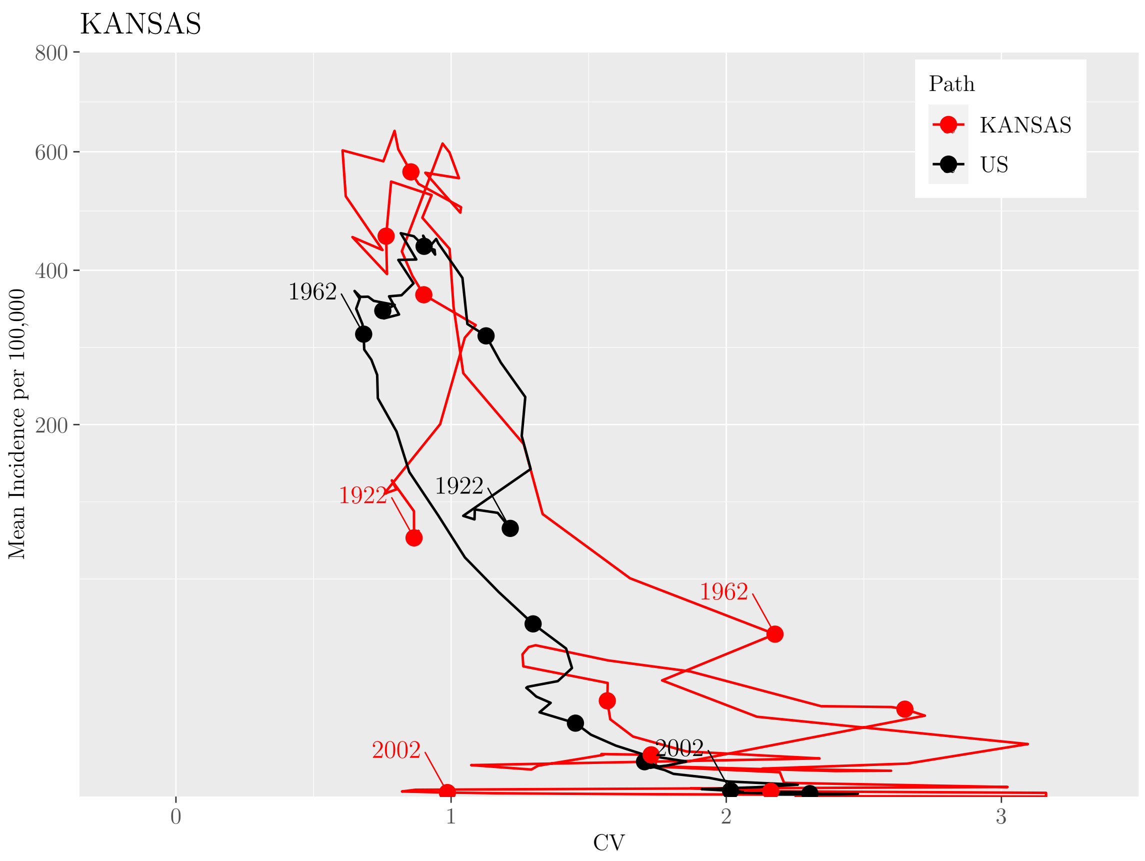

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

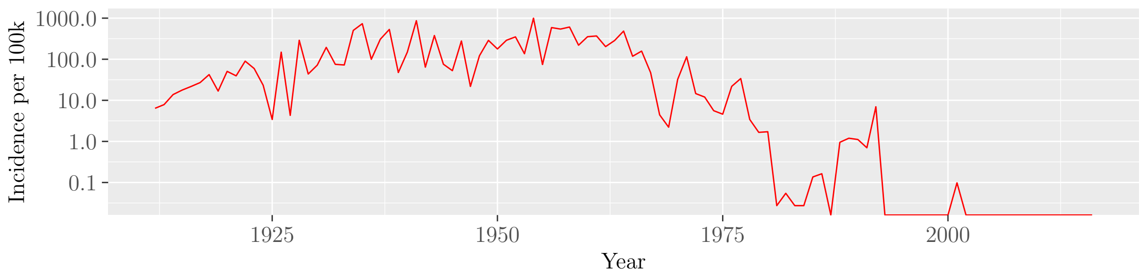

INDIANA
Path
INDIANA
US
800
600
400
200
Mean Incidence per 100,000
1962
1962
1922
1922
2002
2002
0
1
2
3
CV

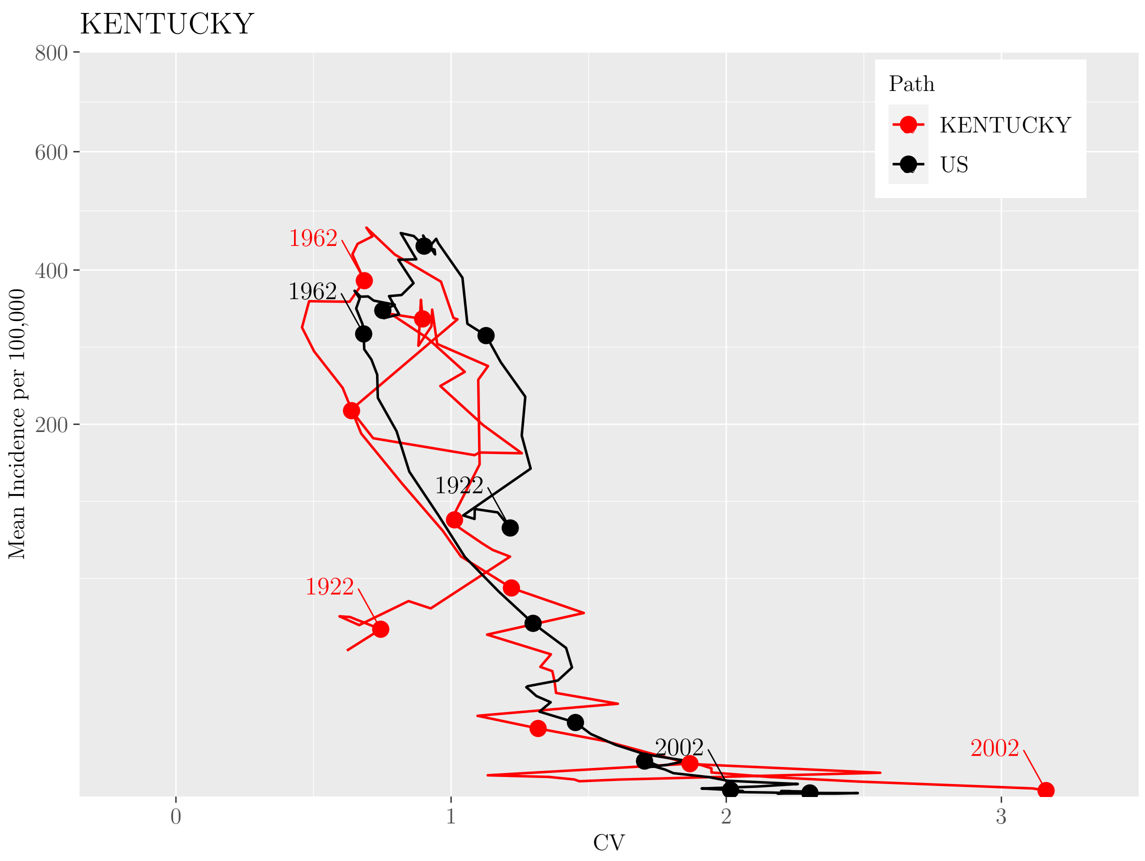

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

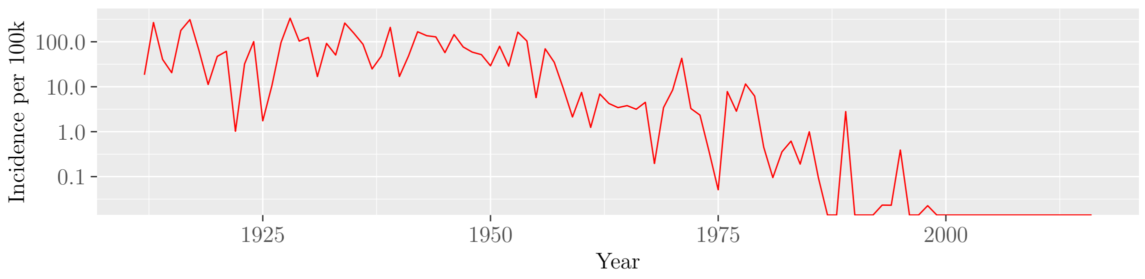

IOWA
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
IOWA
US
1962
1962
1922
1922
2002
2002

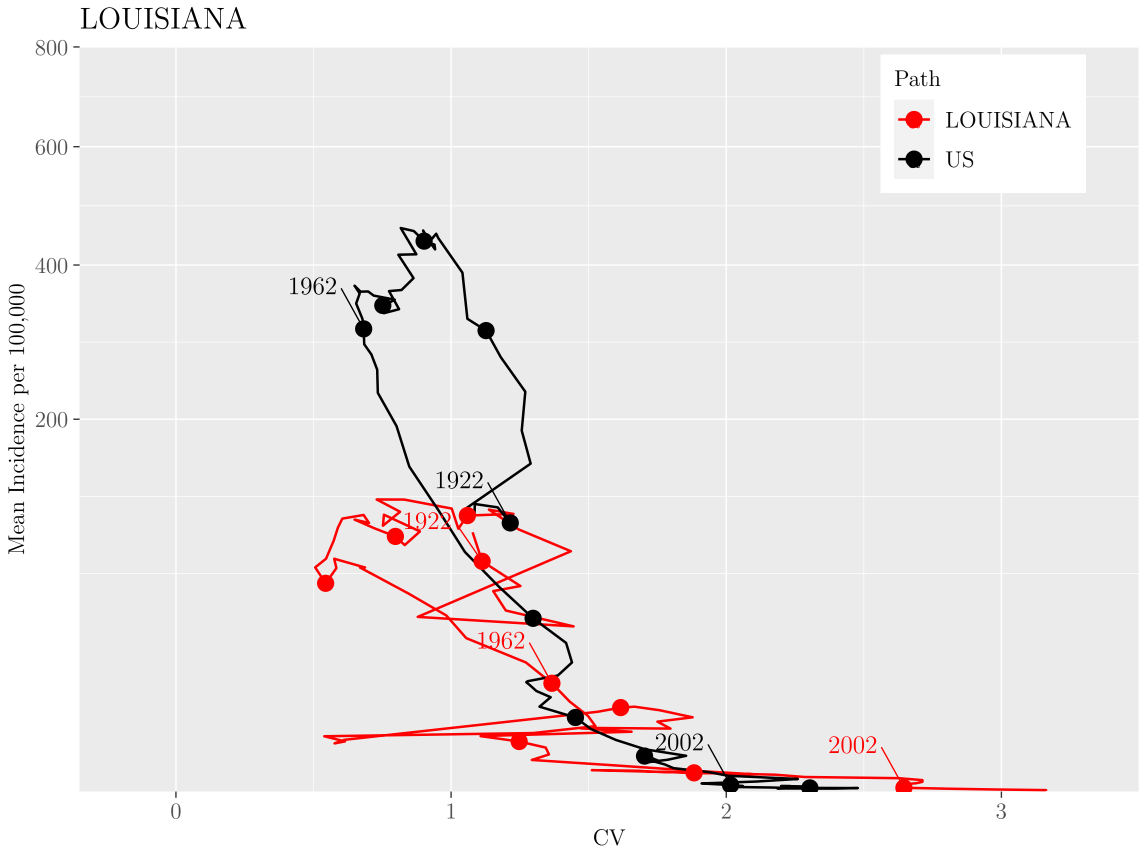

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

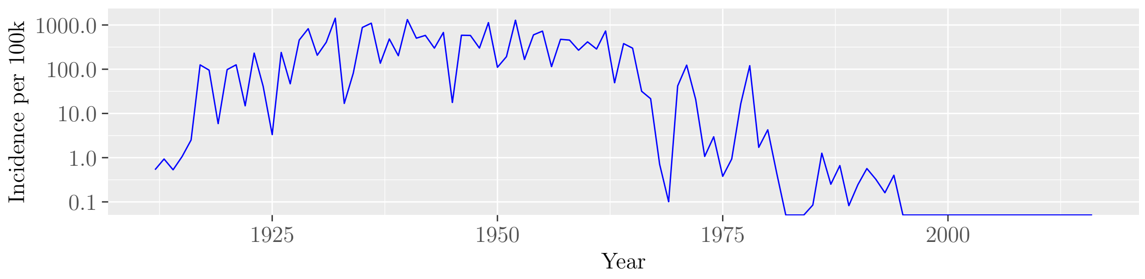

KANSAS
800
600
400
200
Mean Incidence per 100,000
Path
KANSAS
US
1962
1922
1922
1962
2002
2002
0
1
2
3
CV

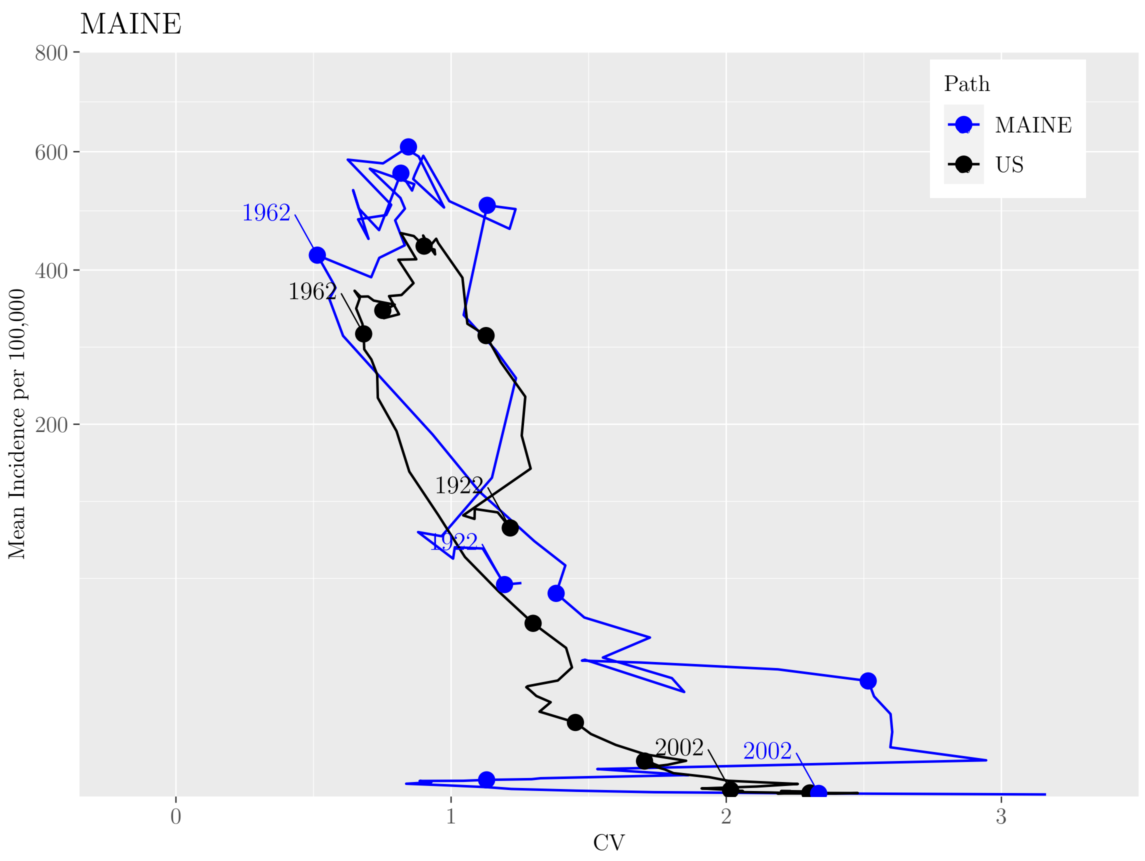

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

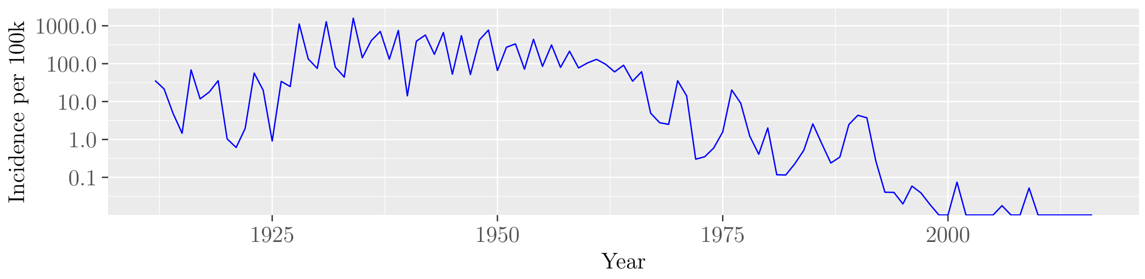

KENTUCKY
Mean Incidence per 100,000
800
600
400
200
CV
0
1
2
3
Path
KENTUCKY
US
1962
1962
1922
1922
2002
2002

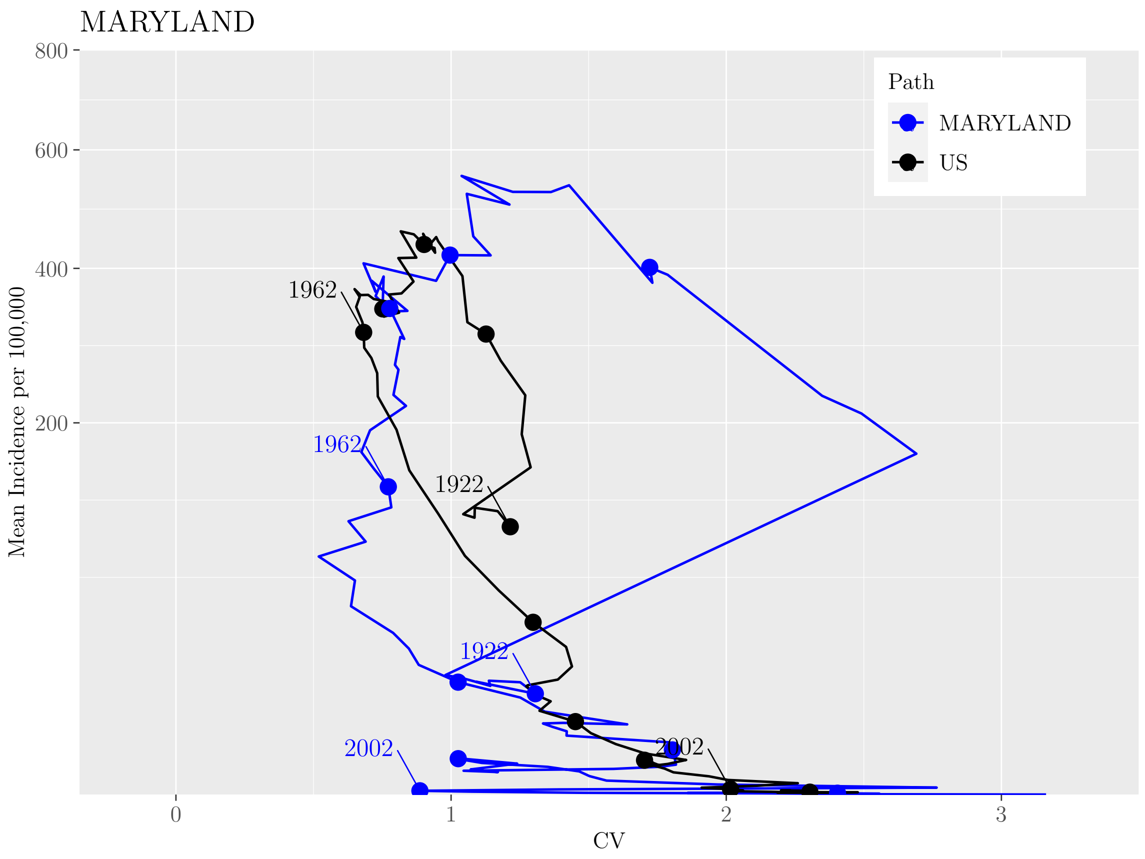

Incidence per 100k
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

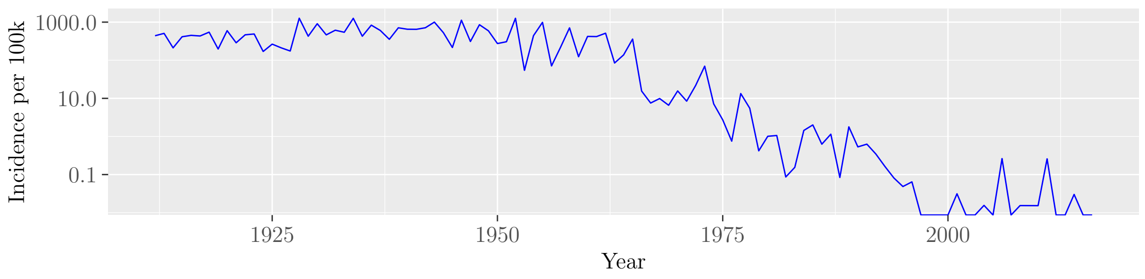

LOUISIANA
Mean Incidence per 100,000
800
600
400
200
CV
0
1
2
3
Path
LOUISIANA
US
1962
1922
1922
1962
2002
2002

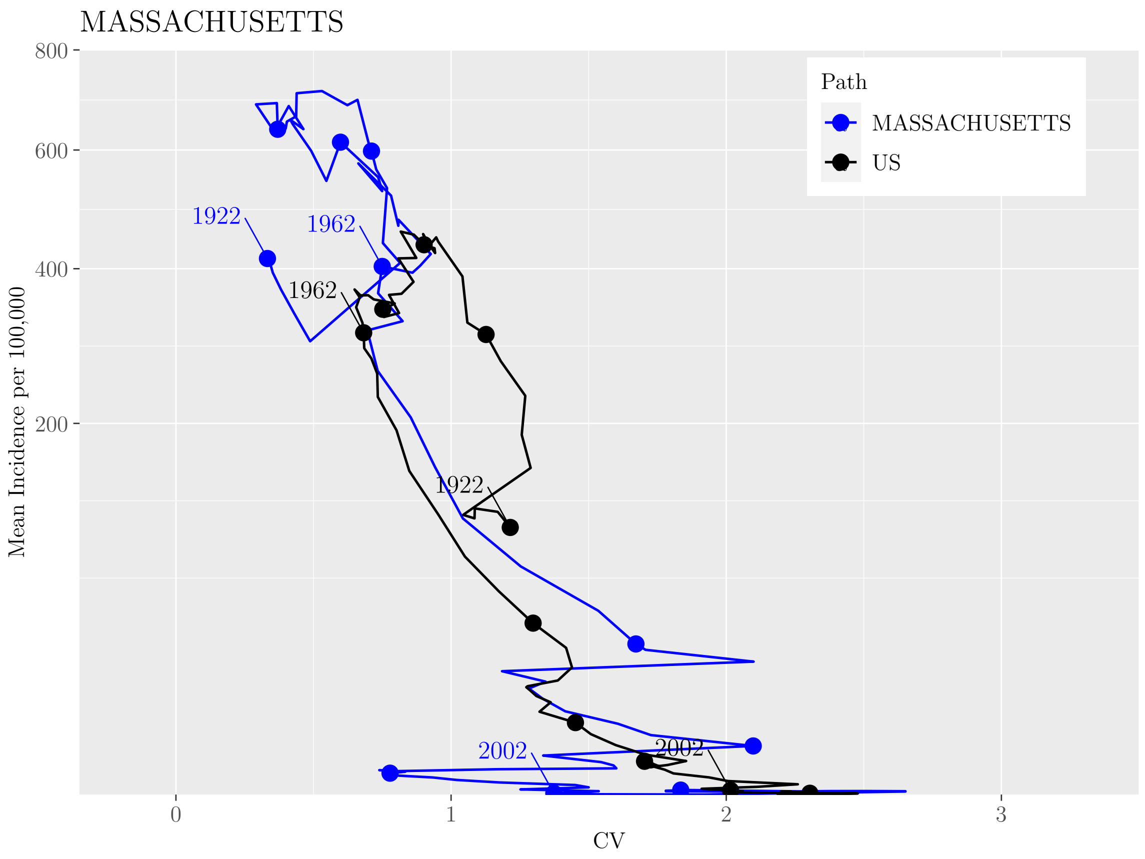

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

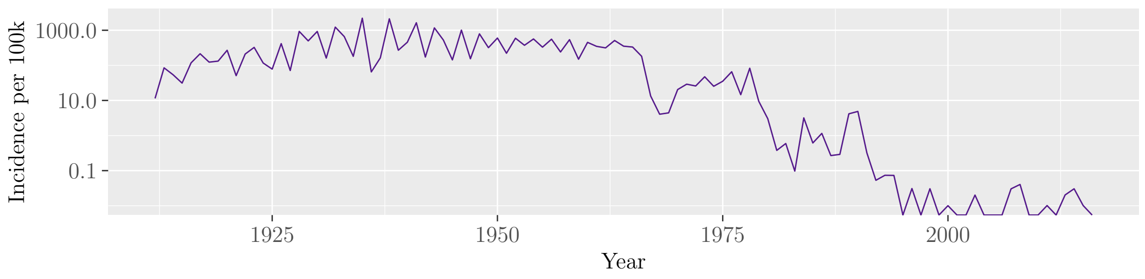

MAINE
Path
MAINE
US
800
600
400
200
Mean Incidence per 100,000
1962
1962
1922
1922
2002
2002
0
1
2
3
CV

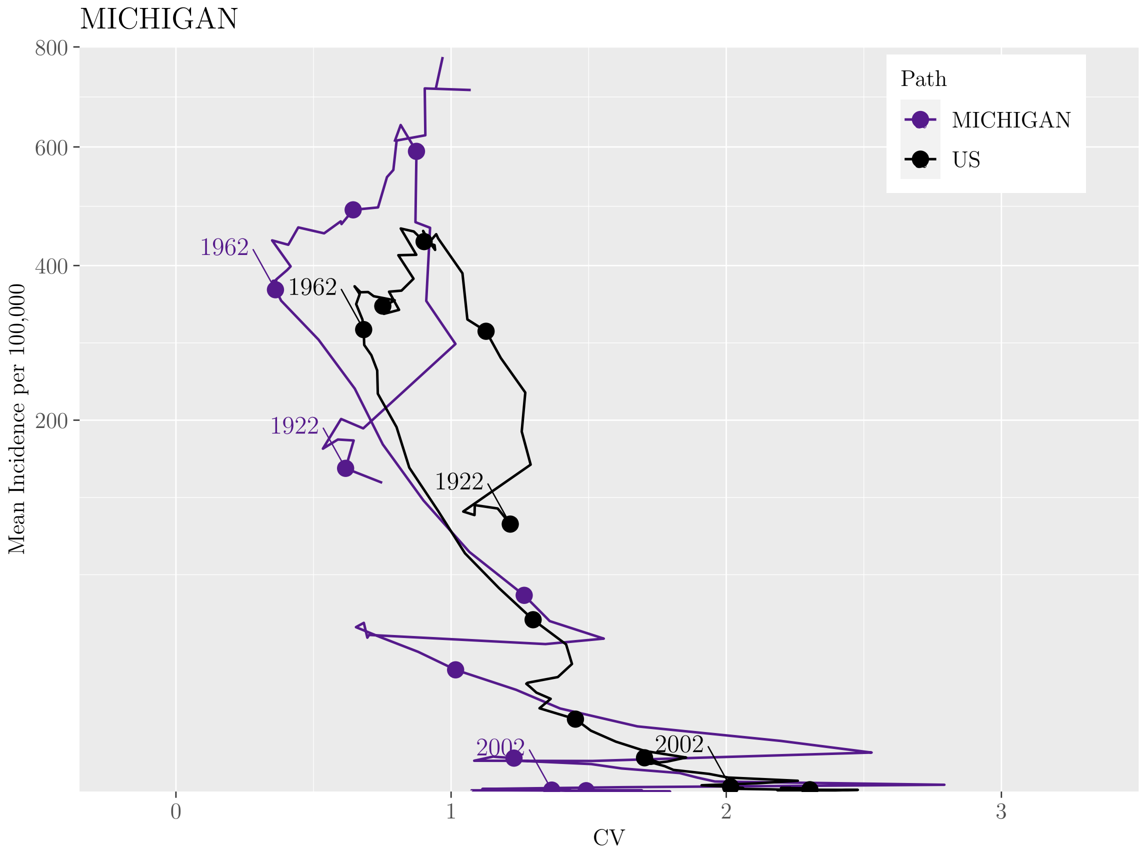

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

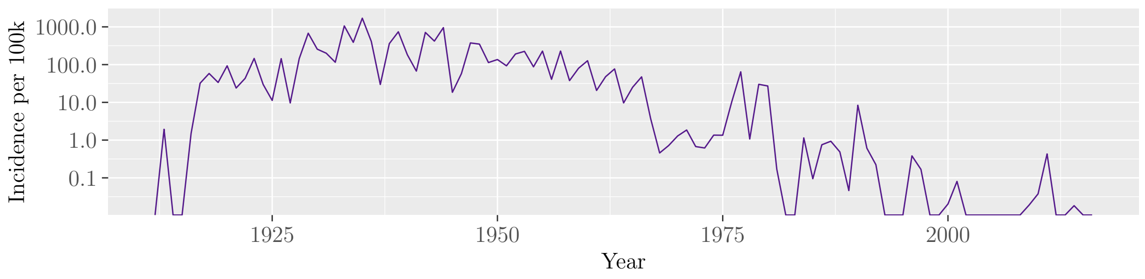

MARYLAND
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
1962
1962
1922
1922
2002
2002
Path
MARYLAND
US

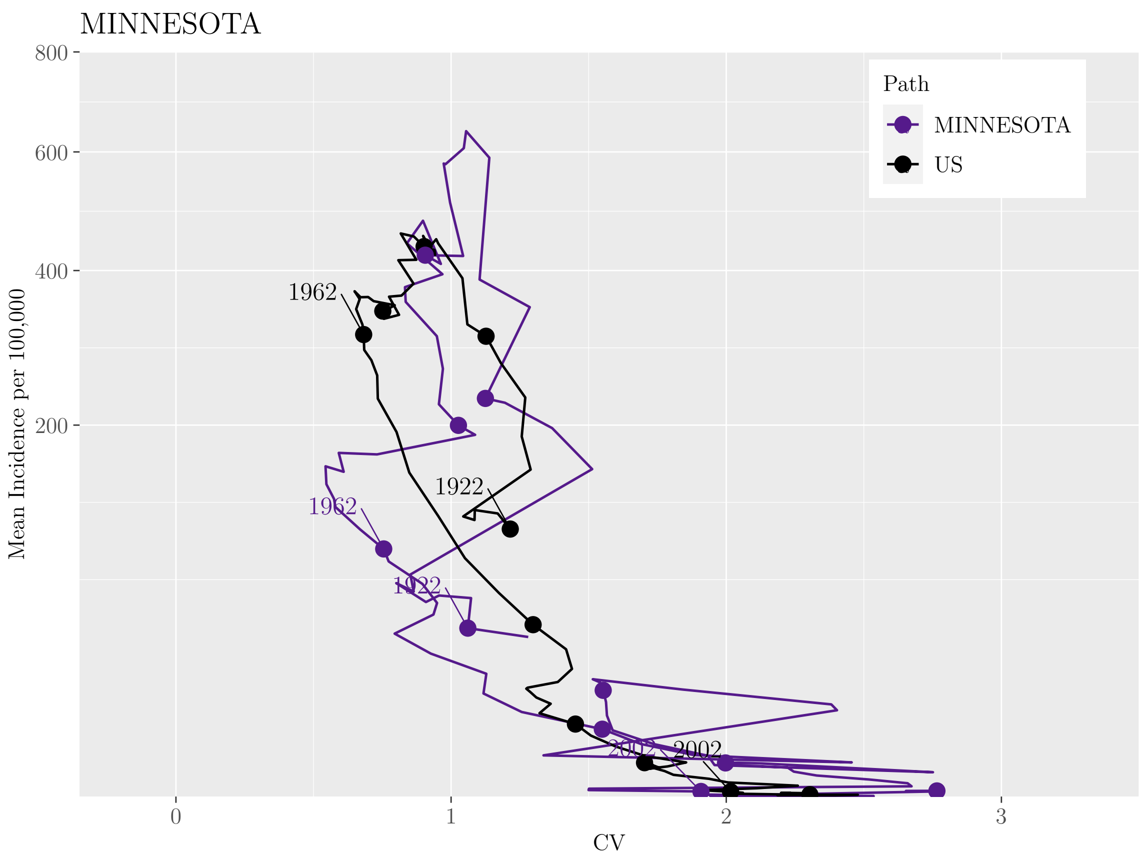

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

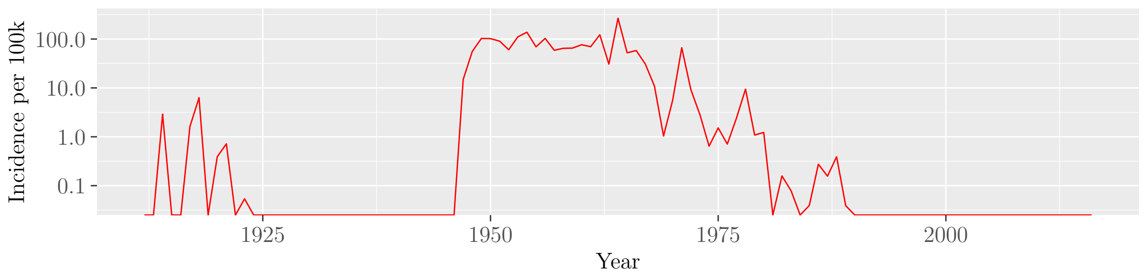

MASSACHUSETTS
800
600
400
200
Mean Incidence per 100,000
1922
1962
1962
1922
2002
2002
0
1
2
3
CV
Path
MASSACHUSETTS
US

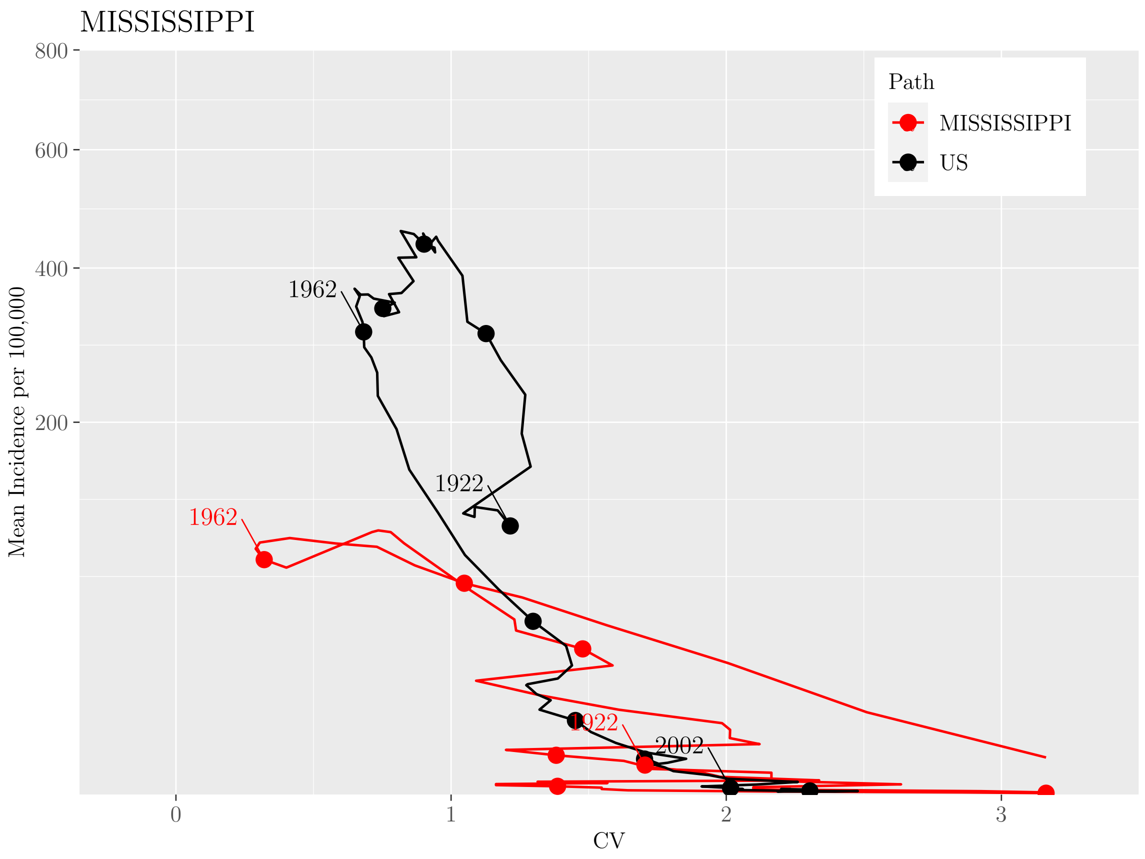

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

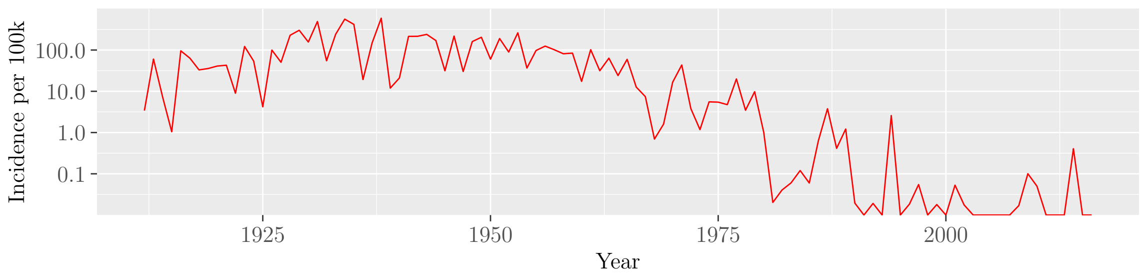

MICHIGAN
Path
MICHIGAN
US
800
600
400
200
Mean Incidence per 100,000
1962
1962
1922
1922
2002
2002
0
1
2
3
CV

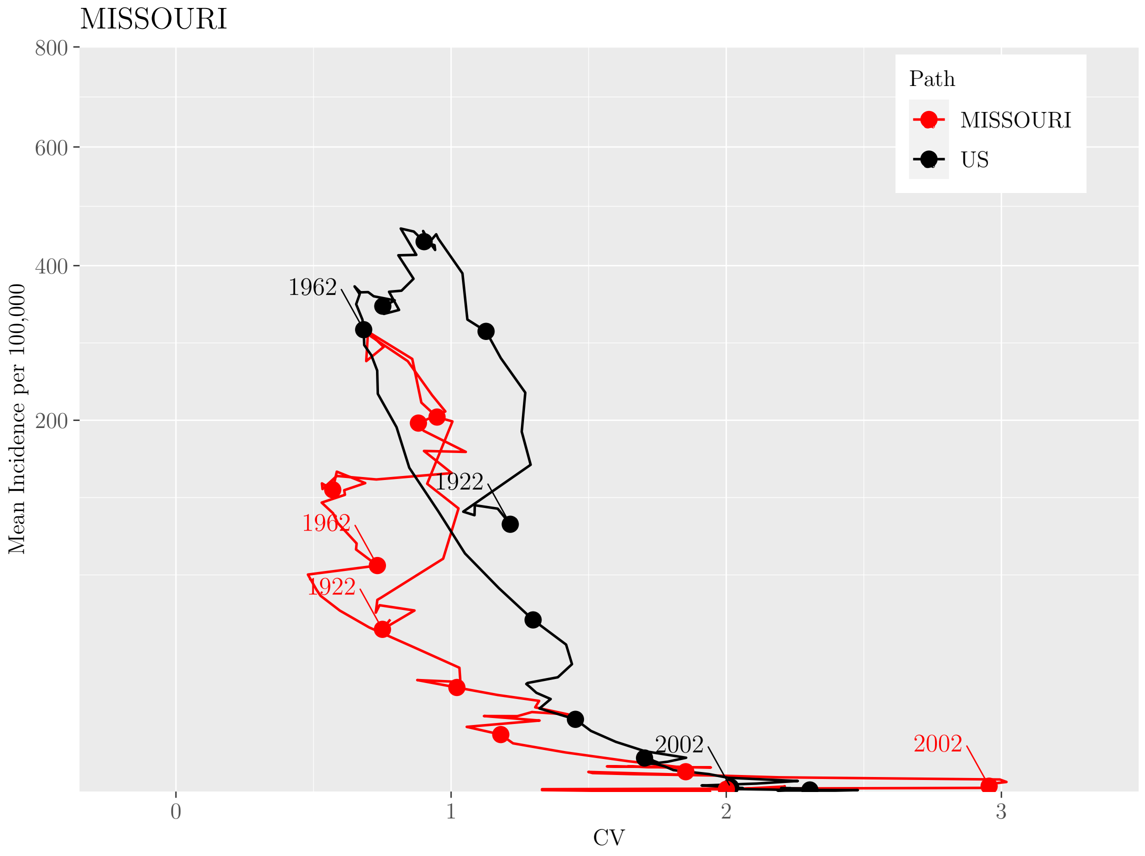

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

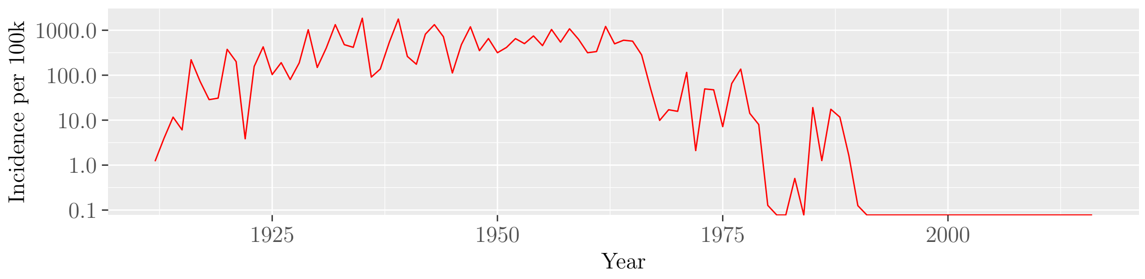

MINNESOTA
Mean Incidence per 100,000
800
600
400
200
Path
MINNESOTA
US
1962
1922
1962
1922
2002
2002
0
1
2
3
CV

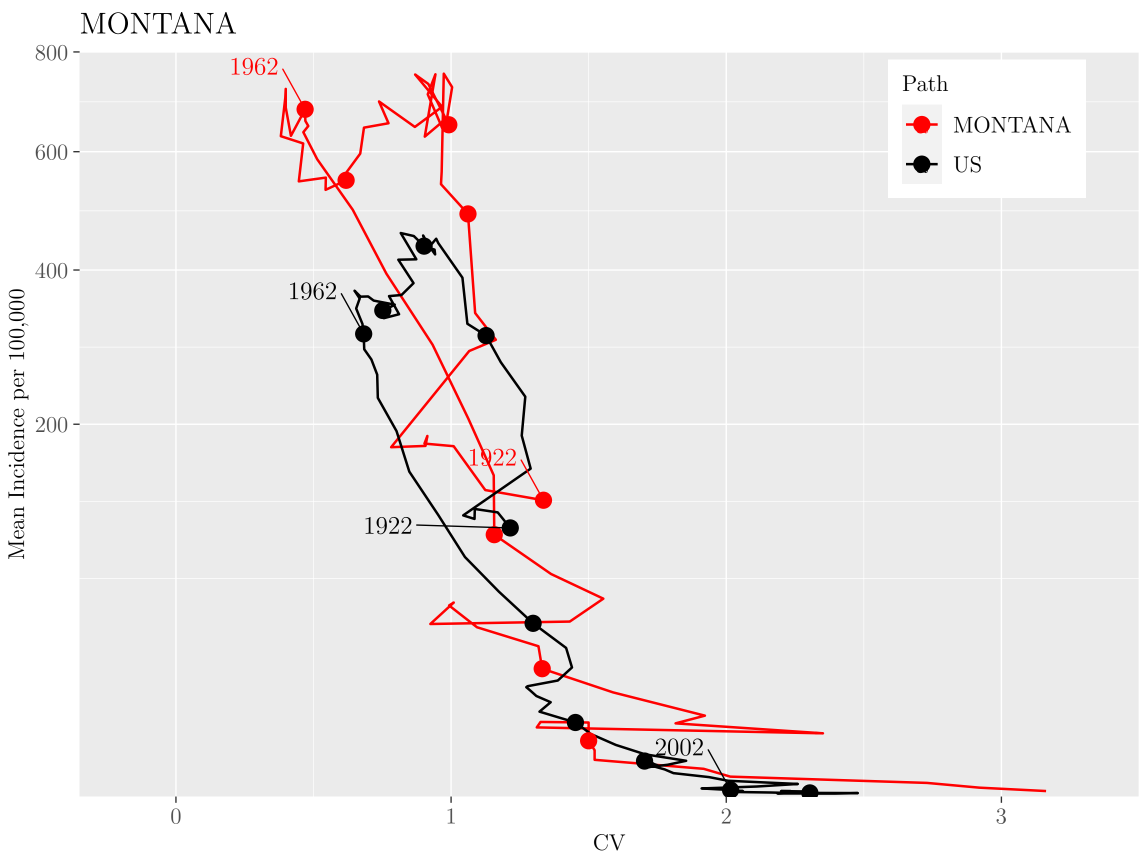

Incidence per 100k
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

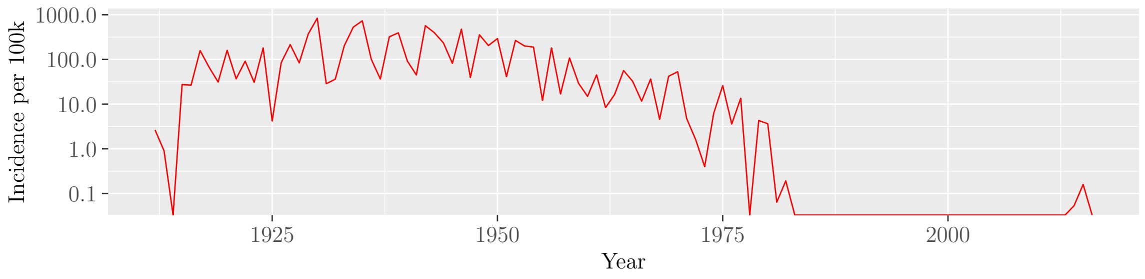

MISSISSIPPI
Mean Incidence per 100,000
800
600
400
200
CV
0
1
2
3
Path
MISSISSIPPI
US
1962
1922
1962
1922
2002

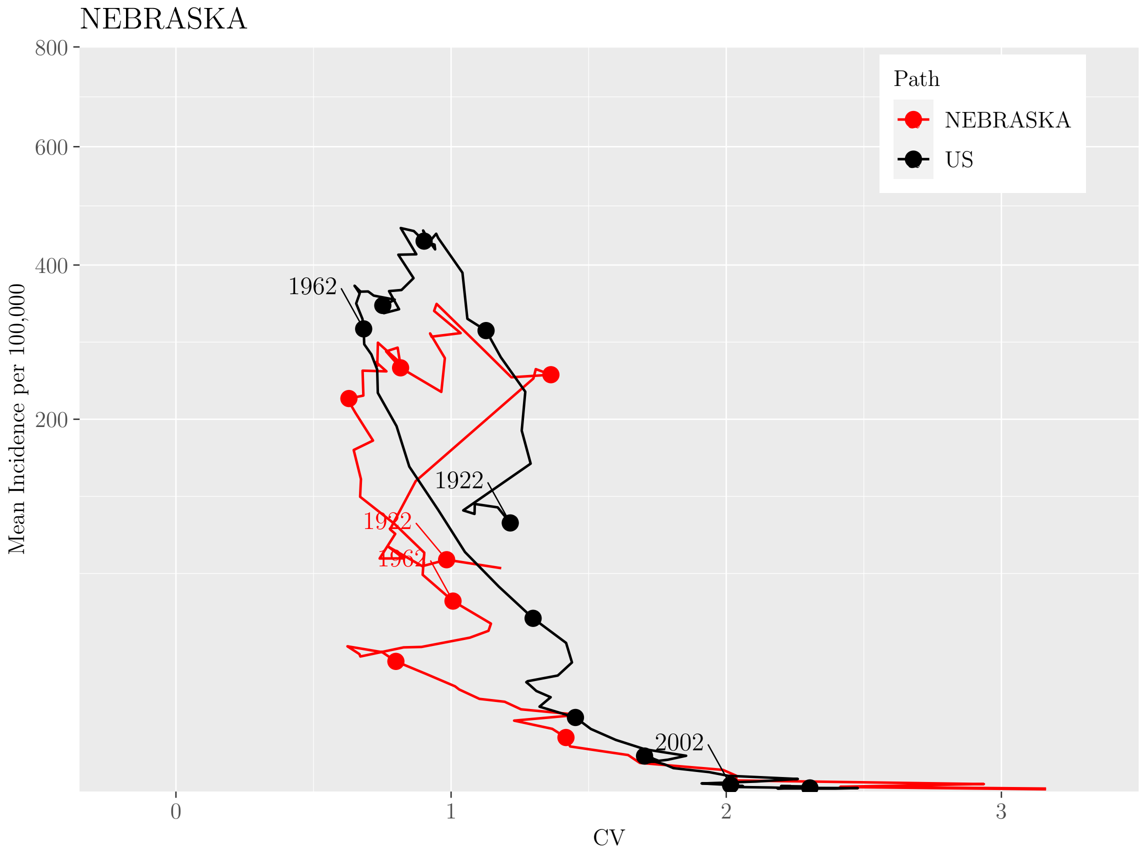

Incidence per 100k
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

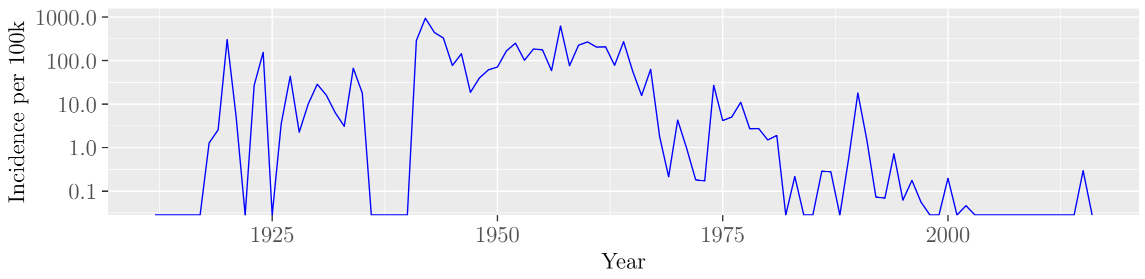

MISSOURI
800
600
400
200
Mean Incidence per 100,000
Path
MISSOURI
US
1962
1922
1962
1922
2002
2002
0
1
2
3
CV

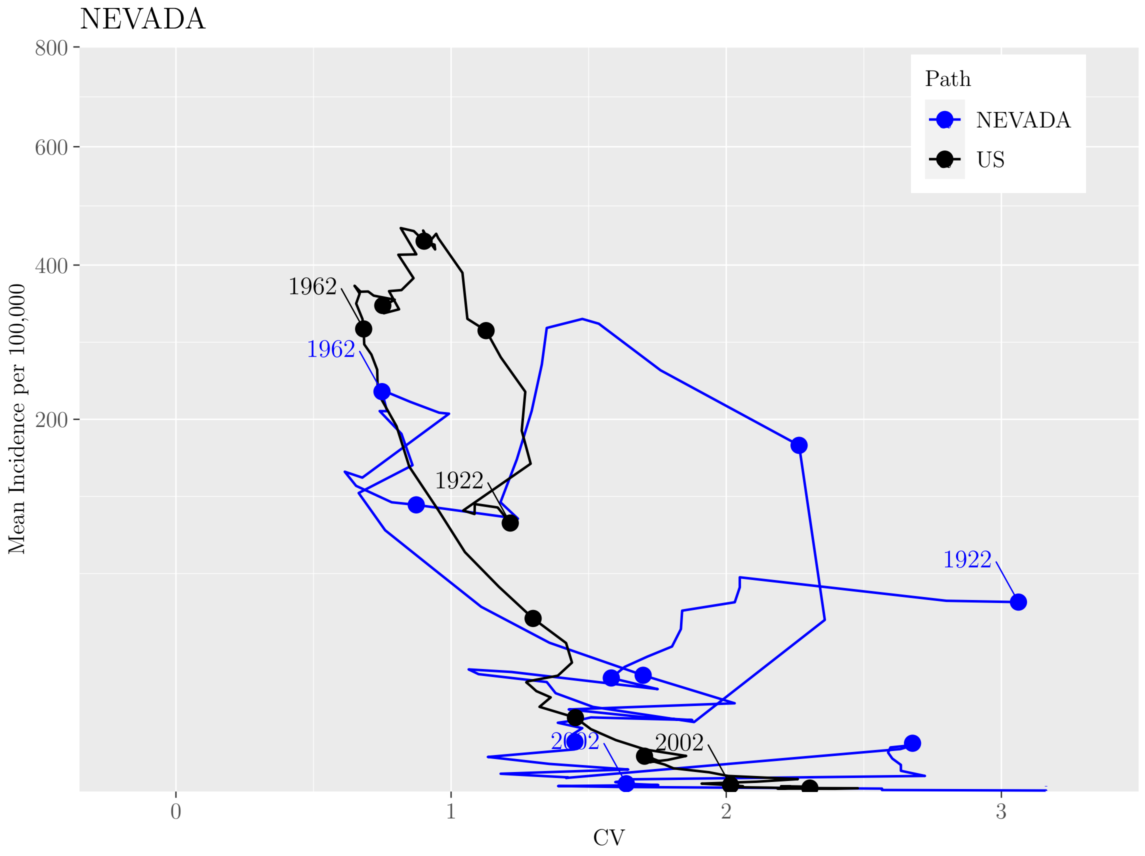

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

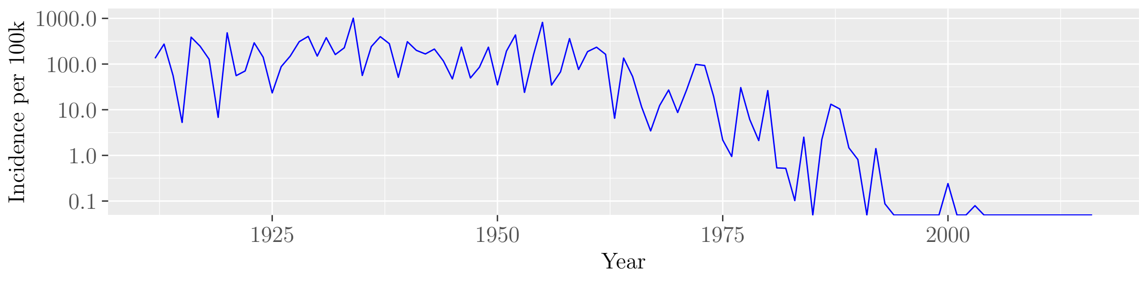

MONTANA
800
600
400
200
Mean Incidence per 100,000
1962
1962
1922
1922
2002
0
1
2
3
CV
Path
MONTANA
US


1000.0
100.0
10.0
1.0
0.1
Incidence per 100k
1925
1950
1975
2000
Year

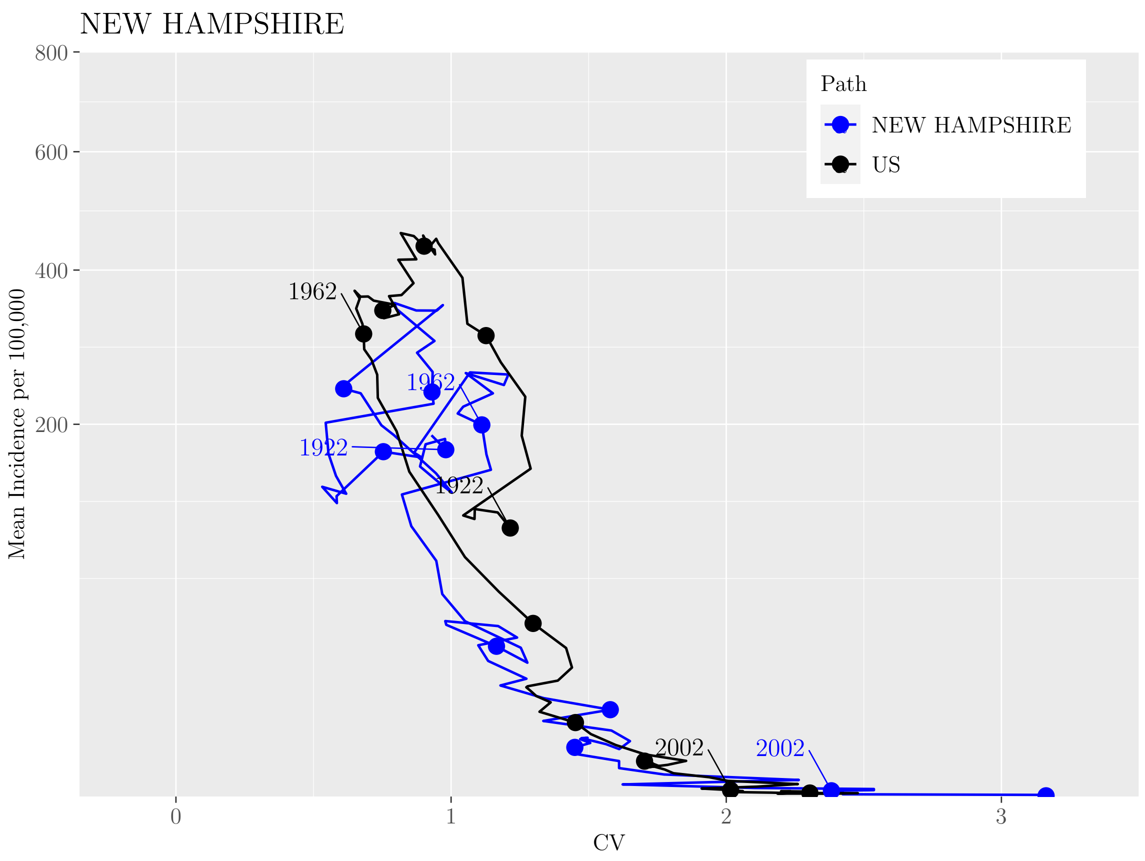

NEBRASKA
800
600
400
200
Mean Incidence per 100,000
Path
NEBRASKA
US
1962
1922
1922
1962
2002
0
1
2
3
CV


Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

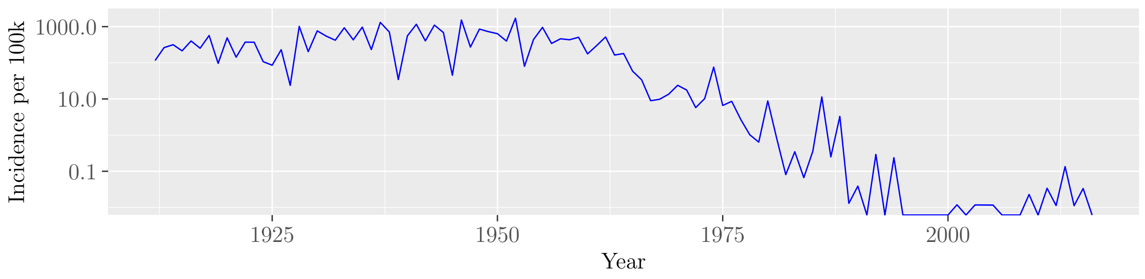

NEVADA
Mean Incidence per 100,000
800
600
400
200
CV
0
1
2
3
Path
NEVADA
US
1962
1962
1922
1922
2002
2002

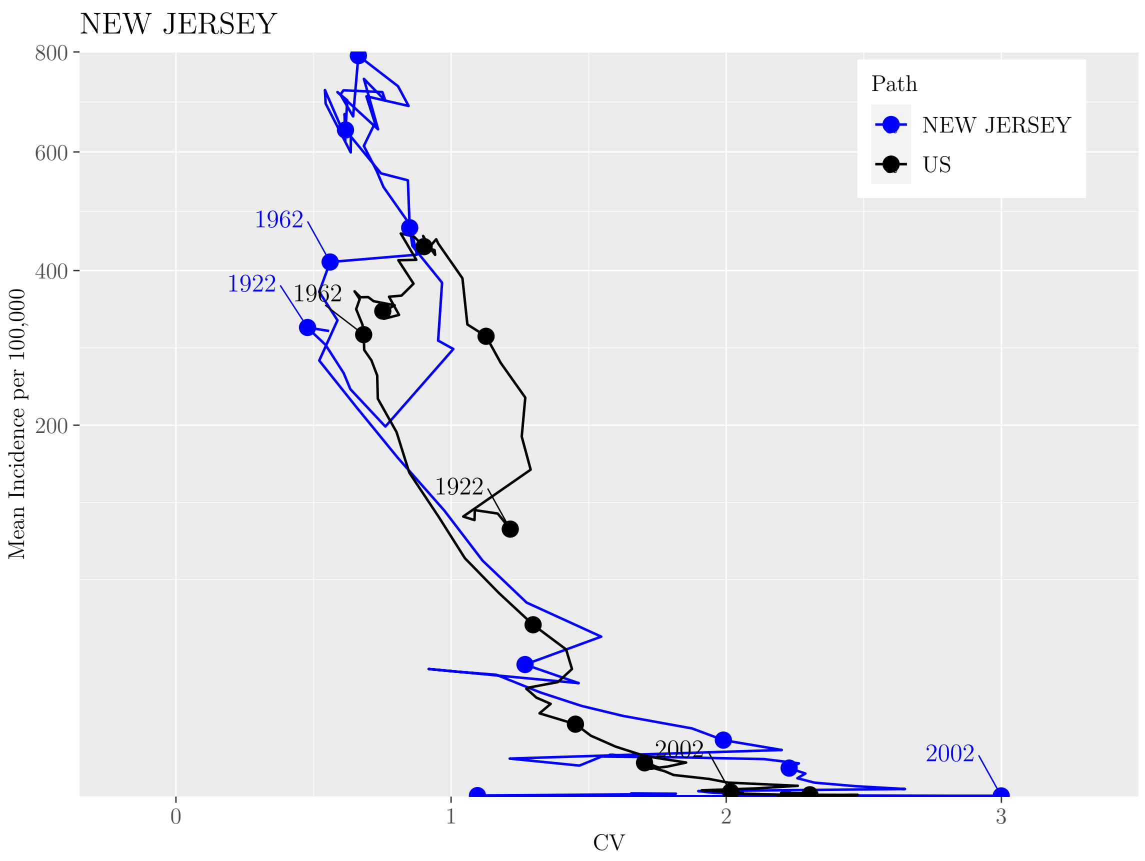


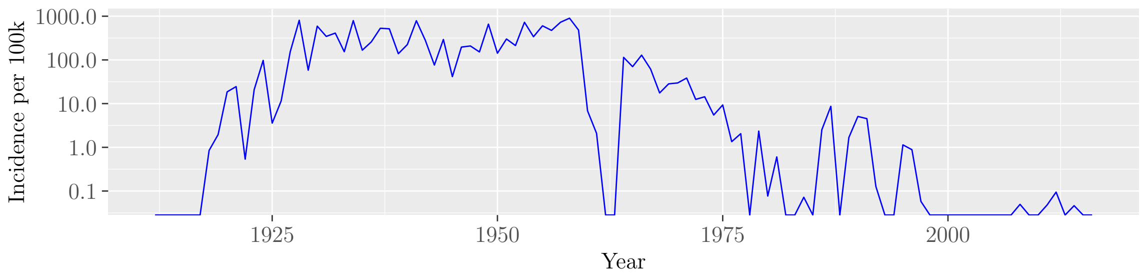
Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

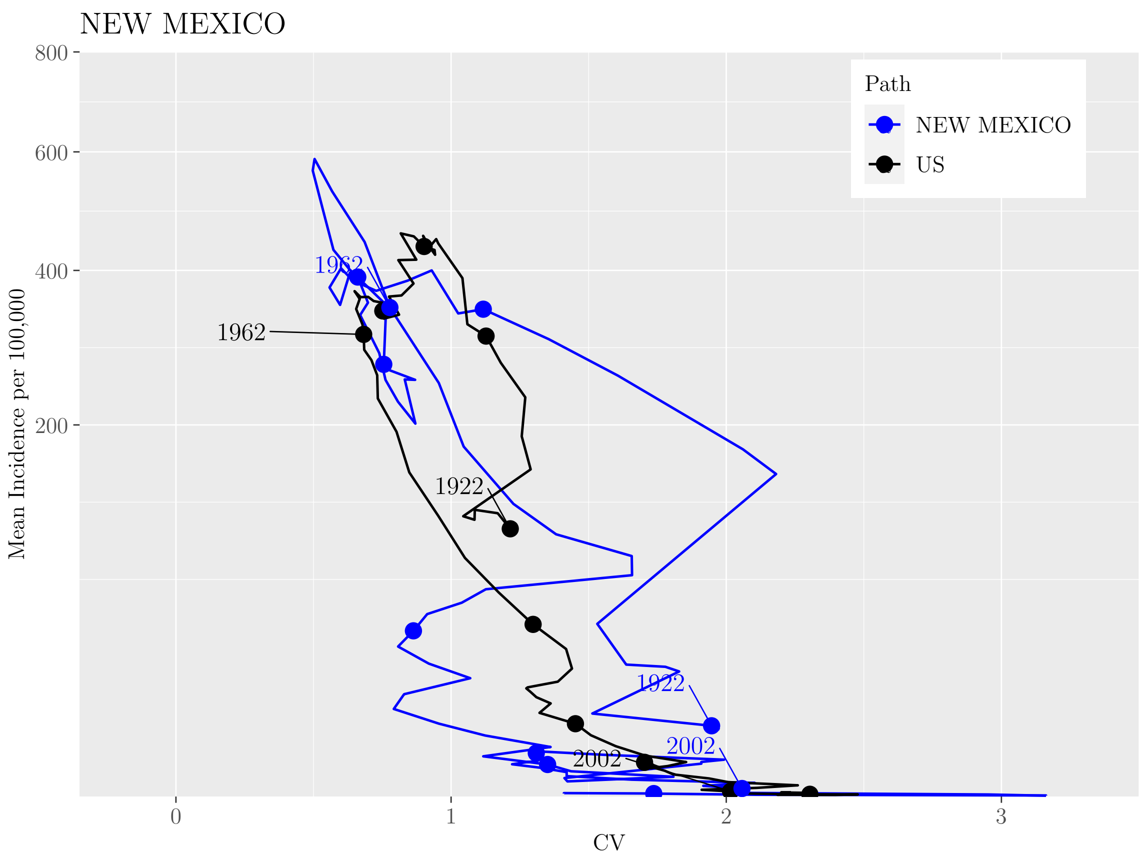


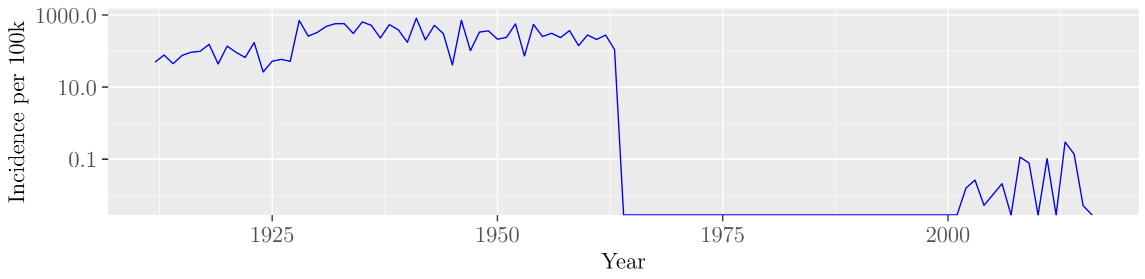
NEW HAMPSHIRE
Mean Incidence per 100,000
800
600
400
200
CV
0
1
2
3
Path
NEW HAMPSHIRE
US
1962
1962
1922
1922
2002
2002

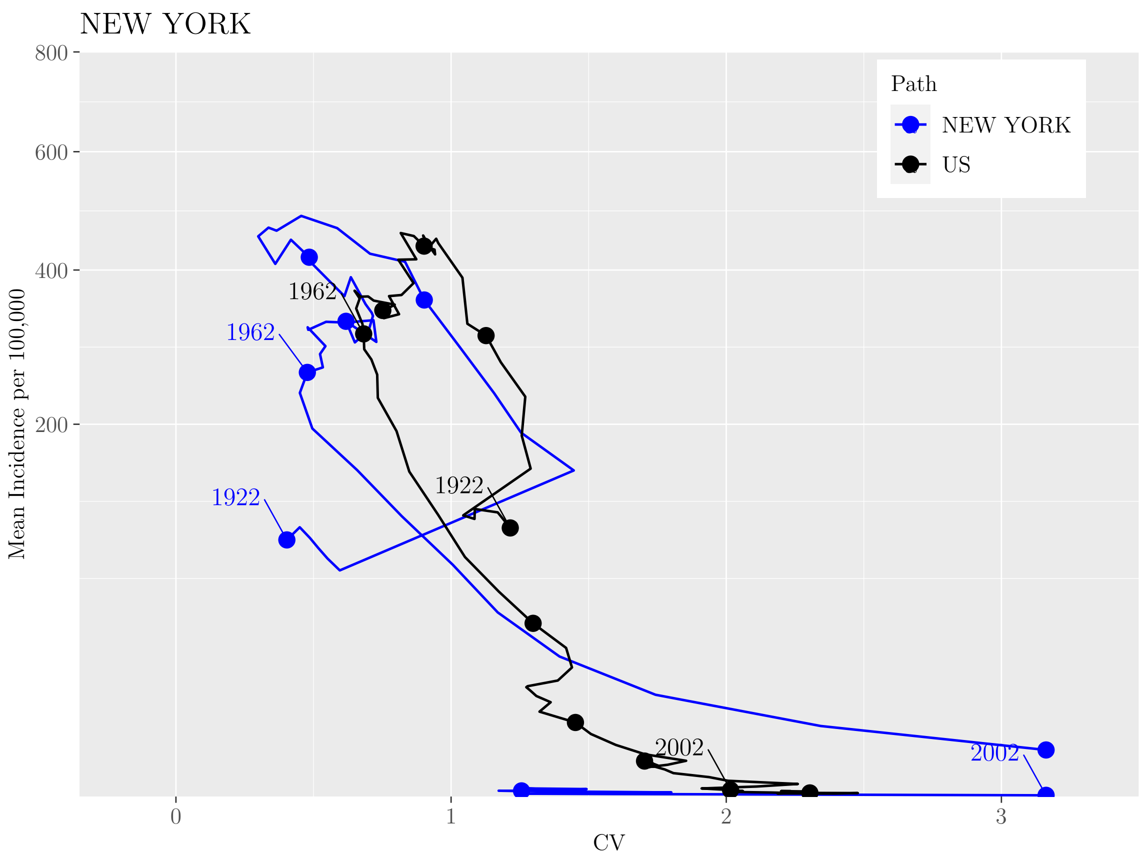

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

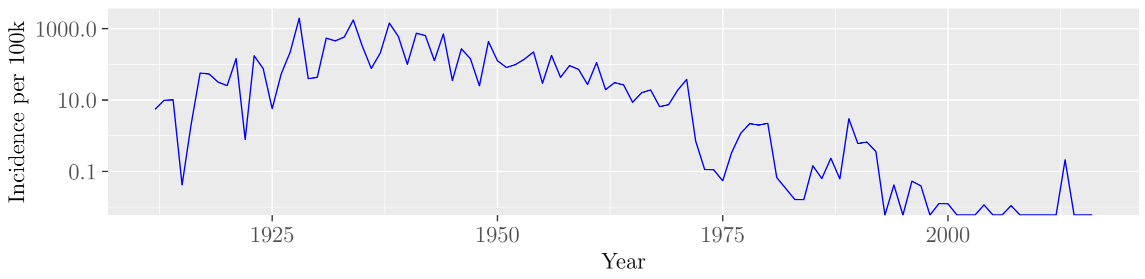

NEW JERSEY
Mean Incidence per 100,000
800
600
400
200
1962
1922
1962
1922
2002
2002
Path
NEW JERSEY
US
0
1
2
3
CV

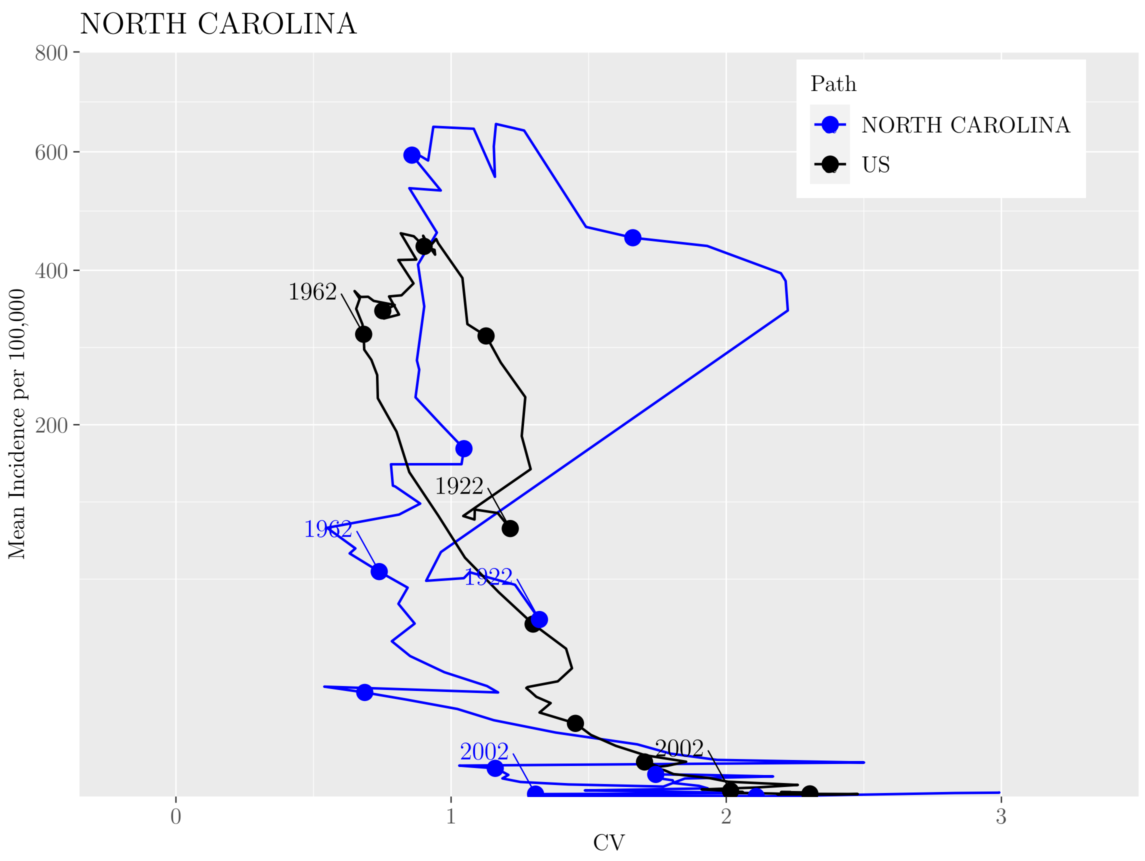

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

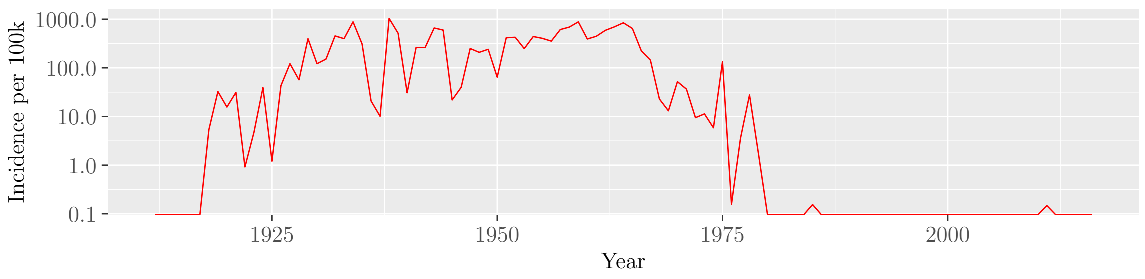

NEW MEXICO
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
NEW MEXICO
US
1962
1962
1922
1922
2002
2002

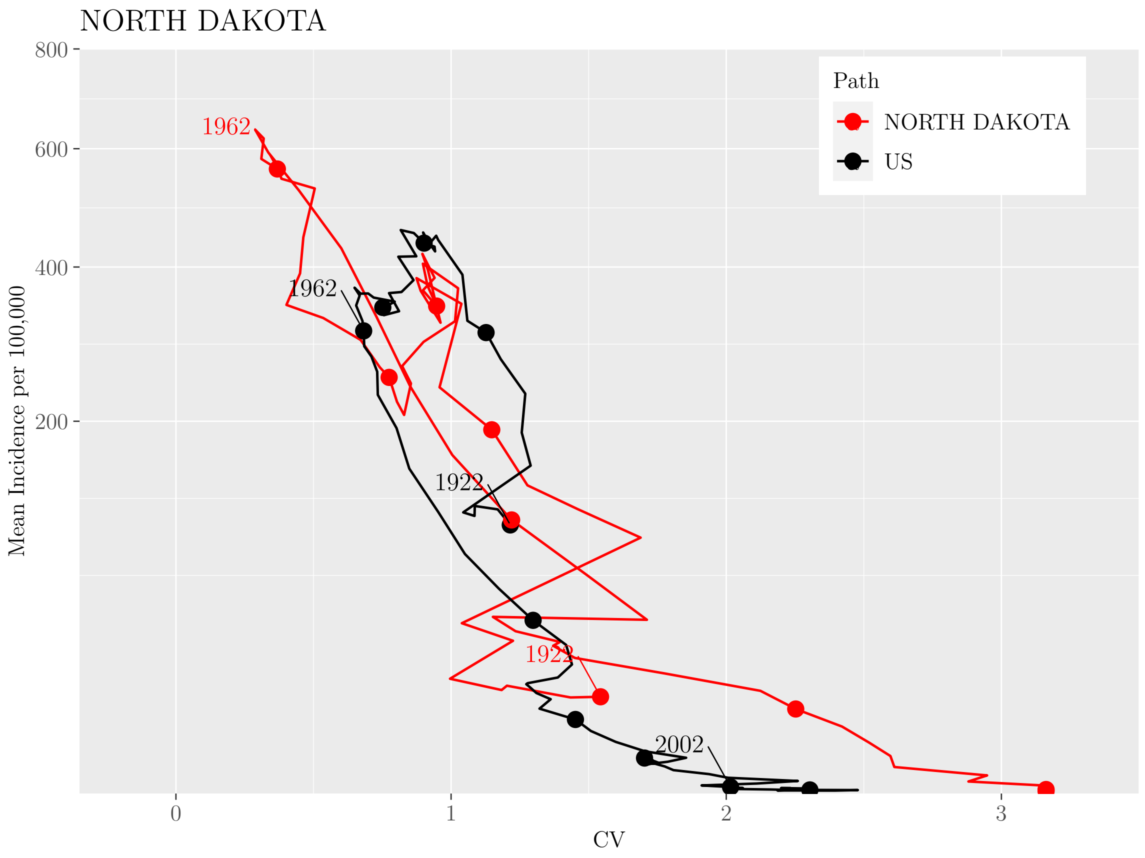

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

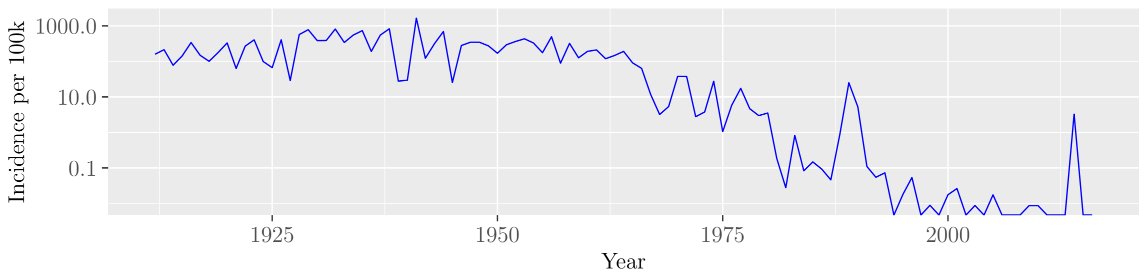

NEW YORK
Mean Incidence per 100,000
800
600
400
200
Path
NEW YORK
US
1962
1962
1922
1922
2002
2002
0
1
2
3
CV

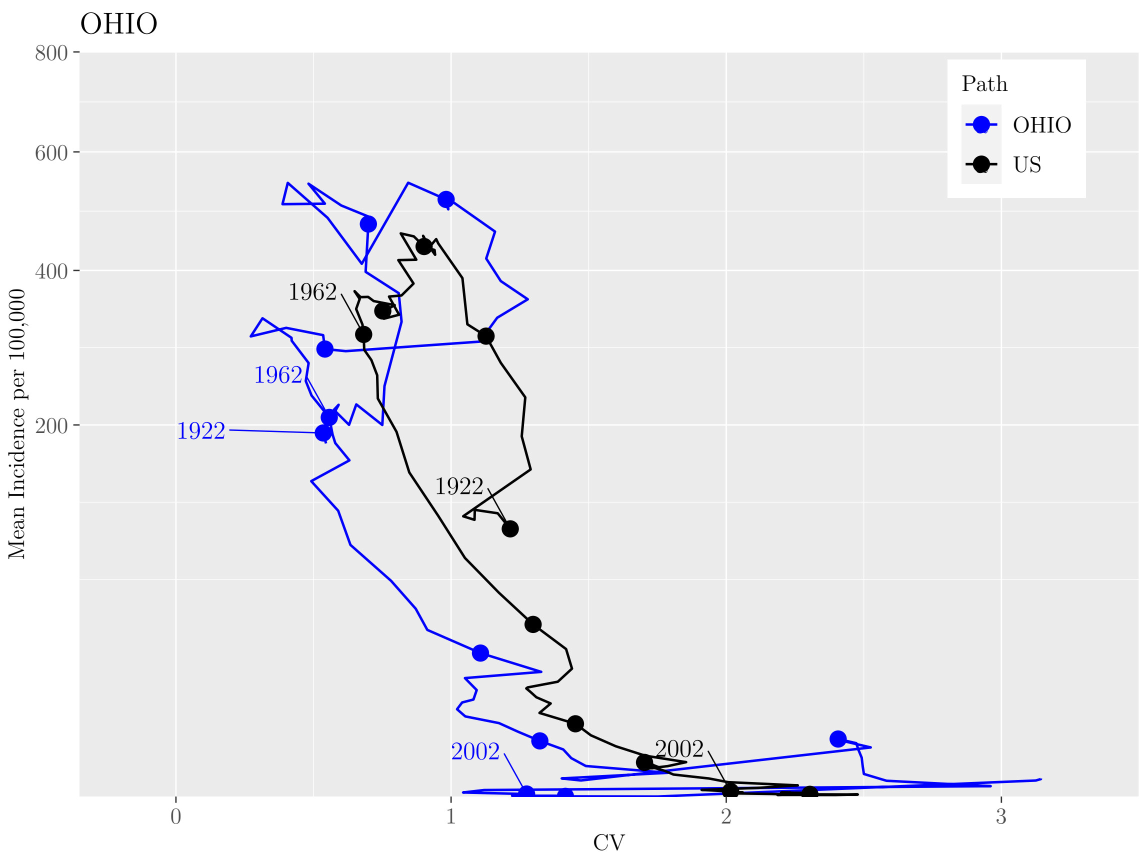

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

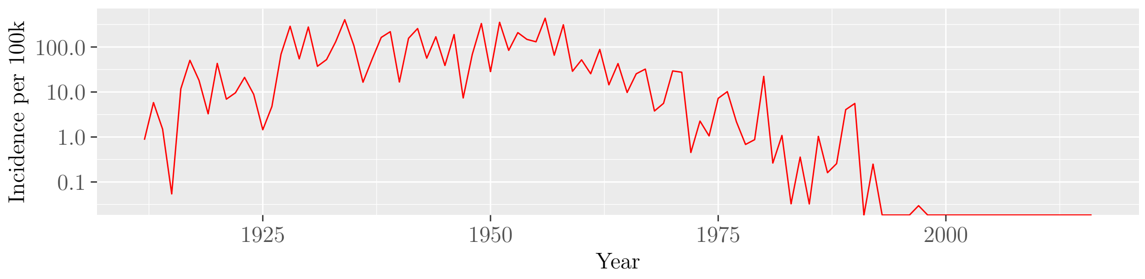

NORTH CAROLINA
800
600
400
200
Mean Incidence per 100,000
Path
NORTH CAROLINA
US
1962
1922
1962
1922
2002
2002
0
1
2
3
CV

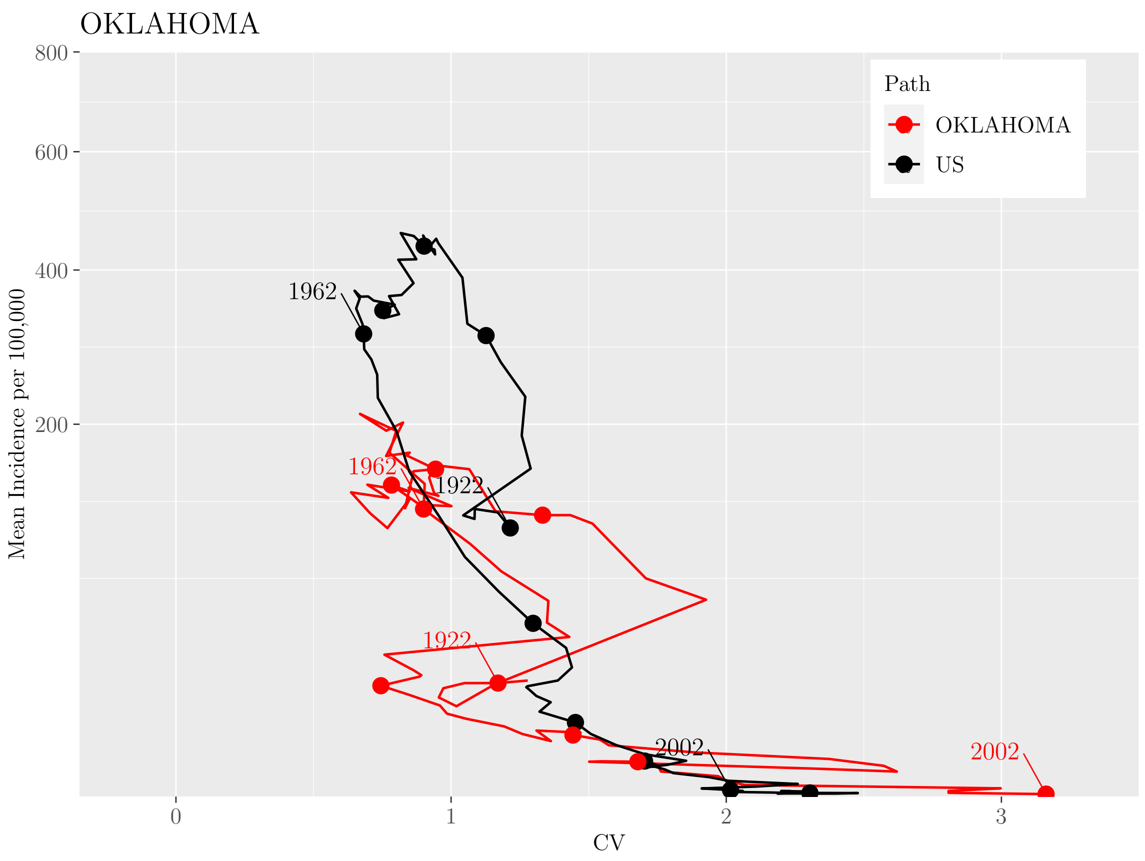

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

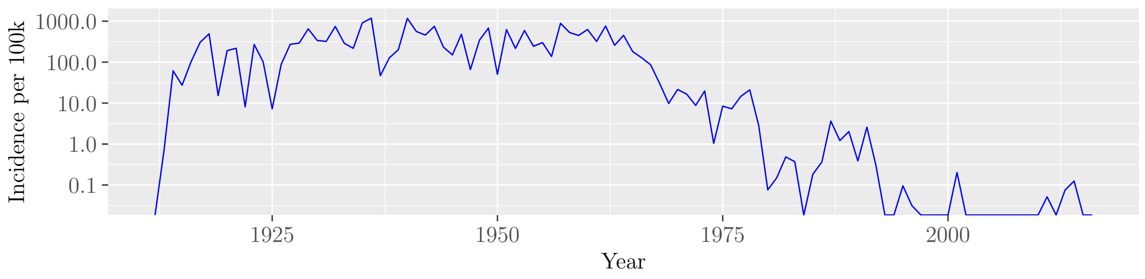

NORTH DAKOTA
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
NORTH DAKOTA
US
1962
1962
1922
1922
2002

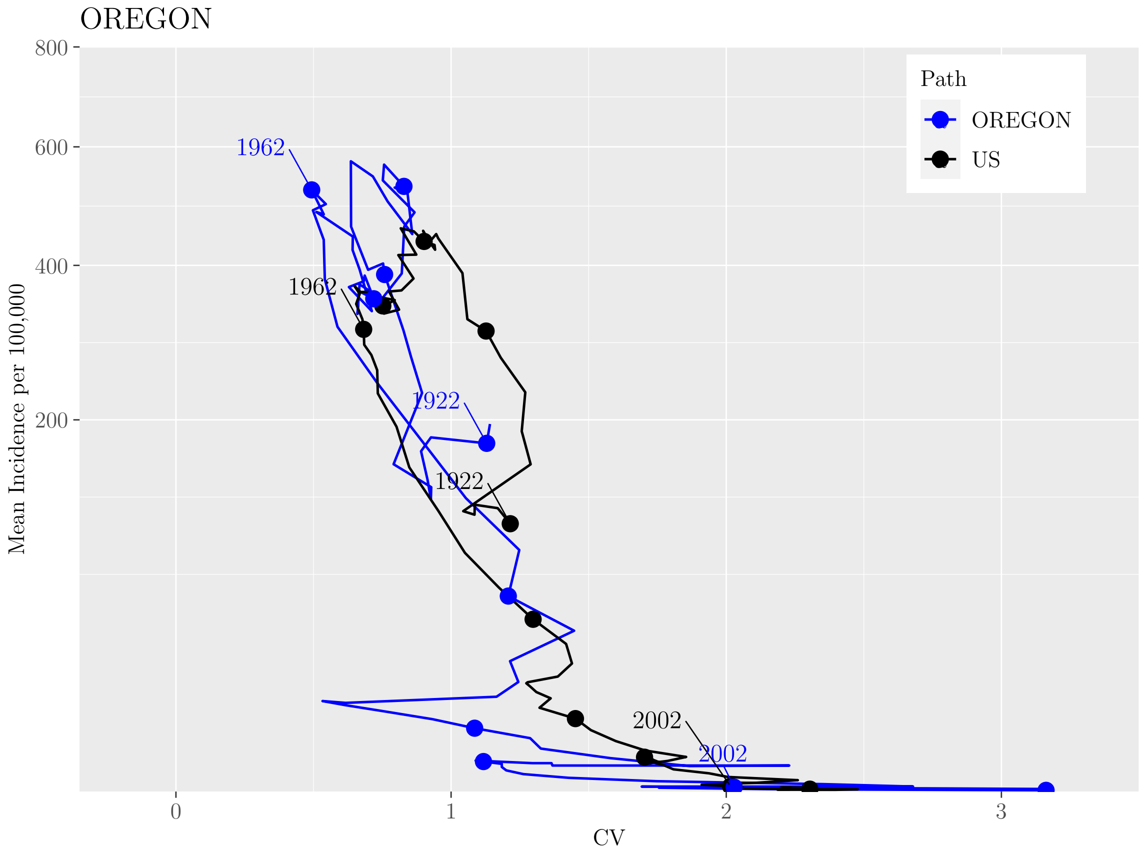

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

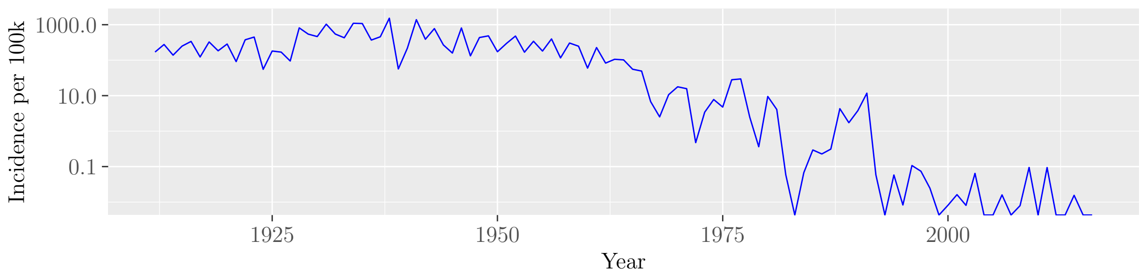

OHIO
Path
OHIO
US
800
600
400
200
Mean Incidence per 100,000
1962
1962
1922
1922
2002
2002
0
1
2
3
CV


Incidence per 100k
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

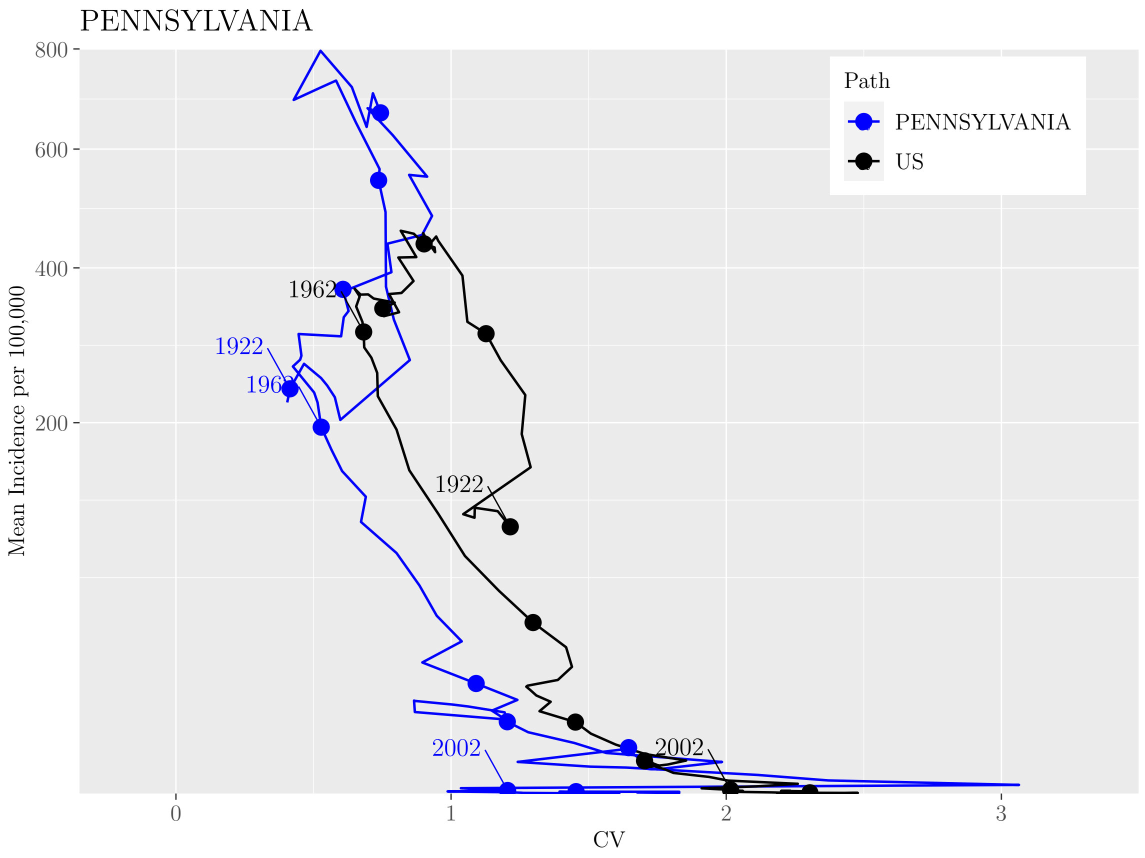

OKLAHOMA
800
600
400
200
Mean Incidence per 100,000
Path
OKLAHOMA
US
1962
1962
1922
1922
2002
2002
0
1
2
3
CV


Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

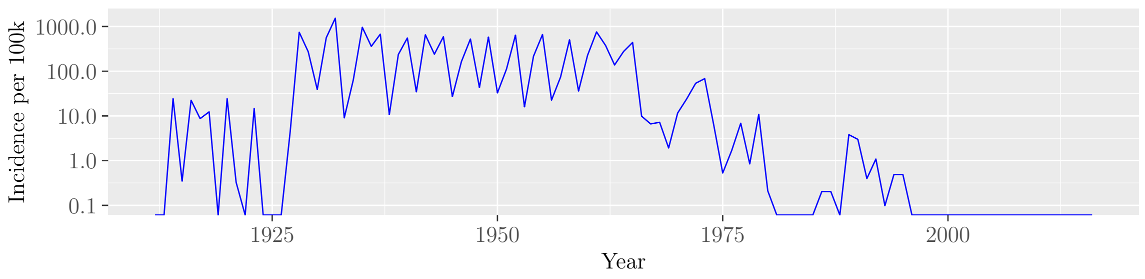

OREGON
800
600
400
200
Mean Incidence per 100,000
1962
1962
1922
1922
2002
2002
0
1
2
3
CV
Path
OREGON
US

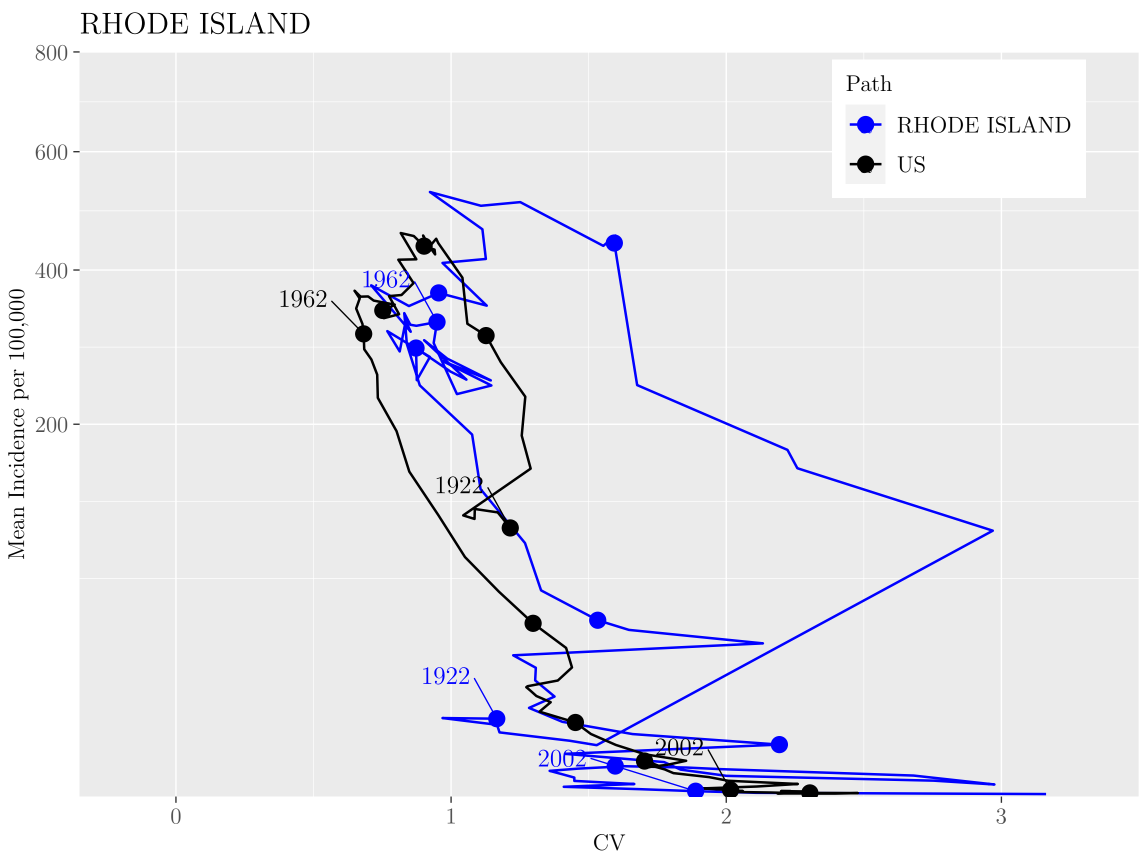


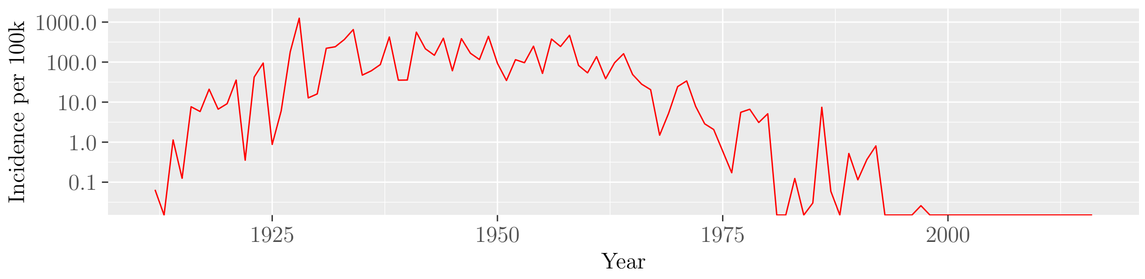
Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

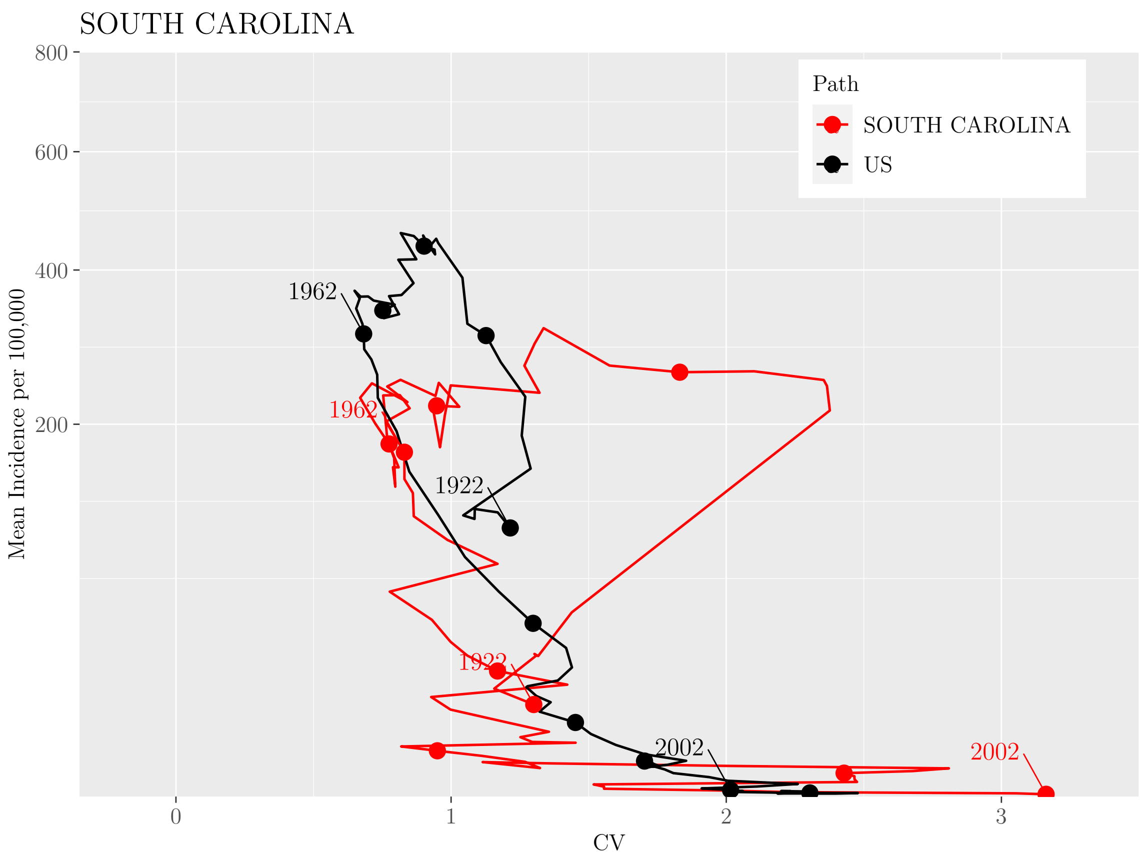


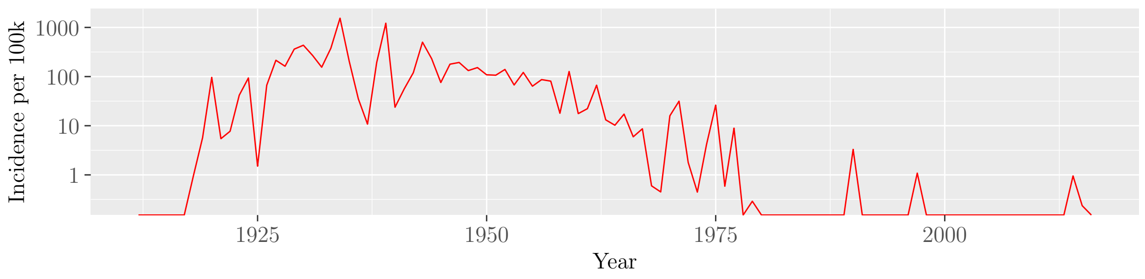
PENNSYLVANIA
Mean Incidence per 100,000
800
600
400
200
CV
0
1
2
3
Path
PENNSYLVANIA
US
1962
1922
1962
1922
2002
2002

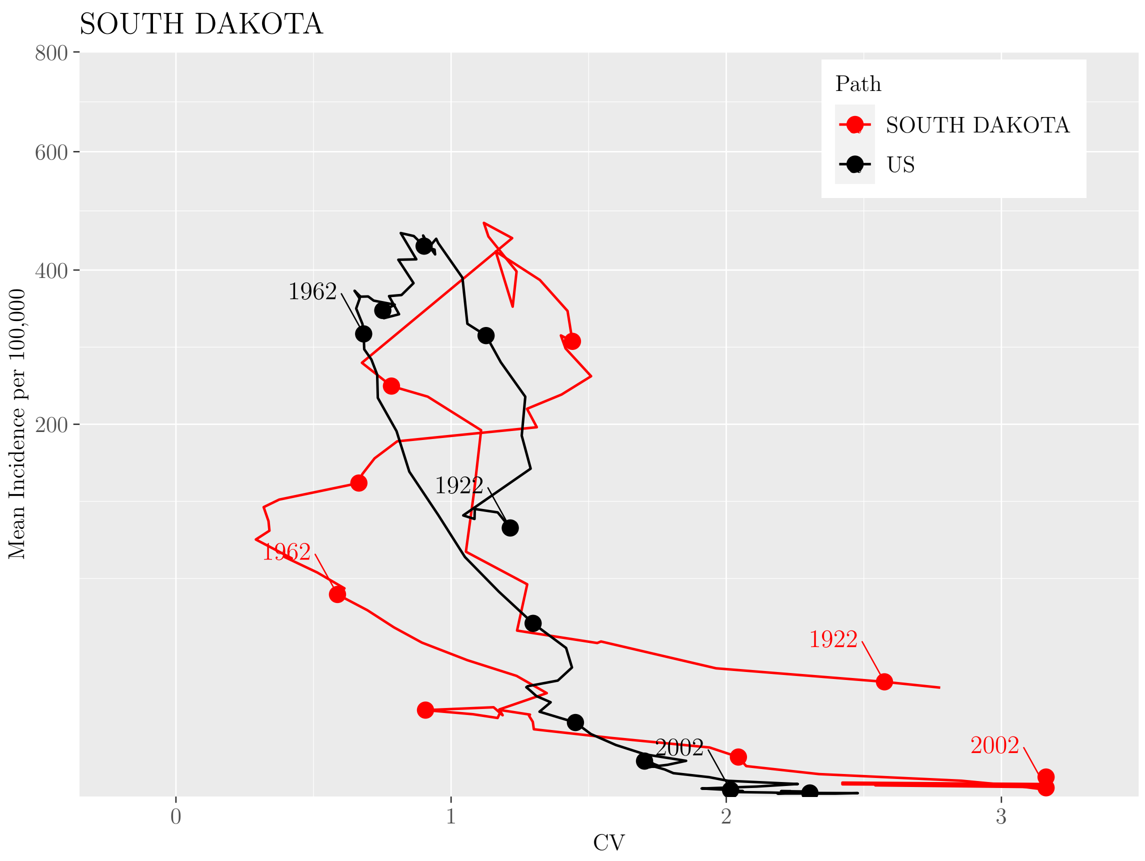

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

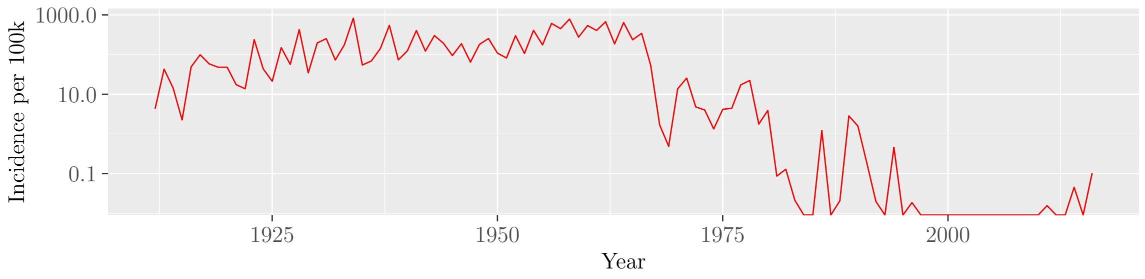

RHODE ISLAND
800
600
400
200
Mean Incidence per 100,000
Path
RHODE ISLAND
US
1962
1962
1922
1922
2002
2002
0
1
2
3
CV

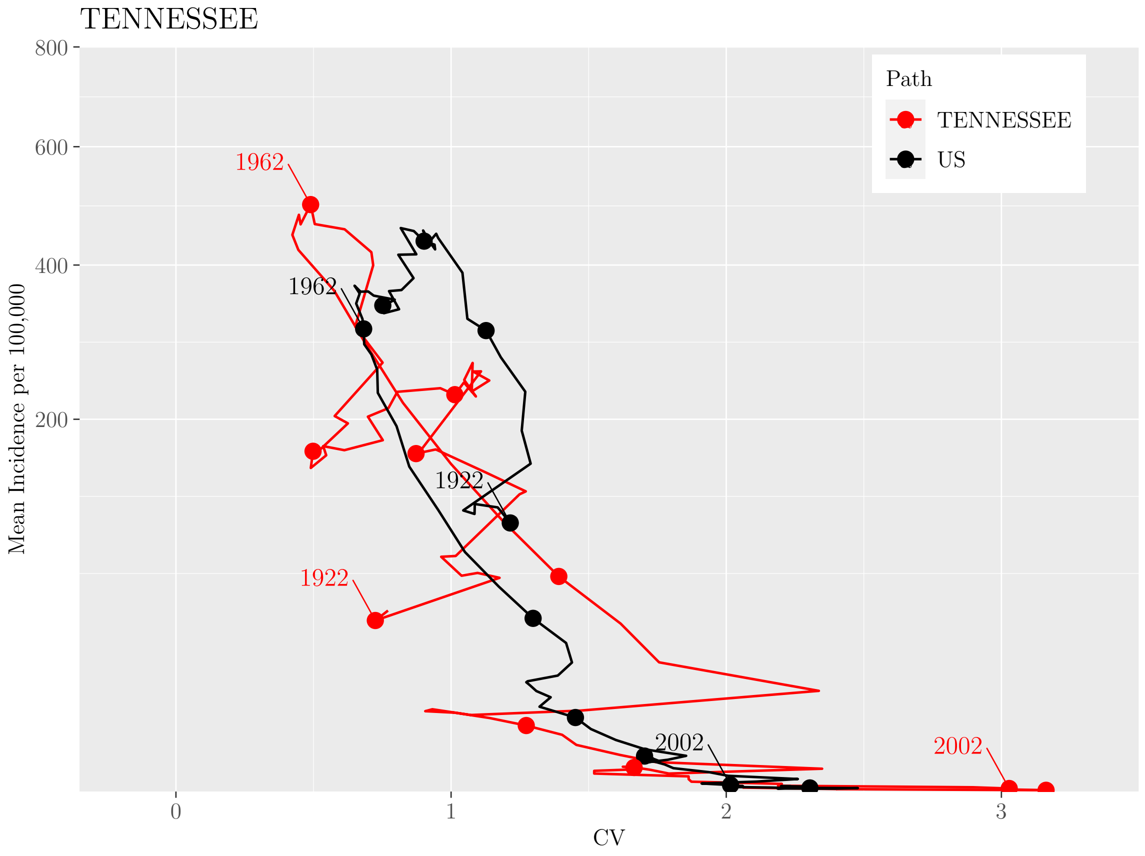

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

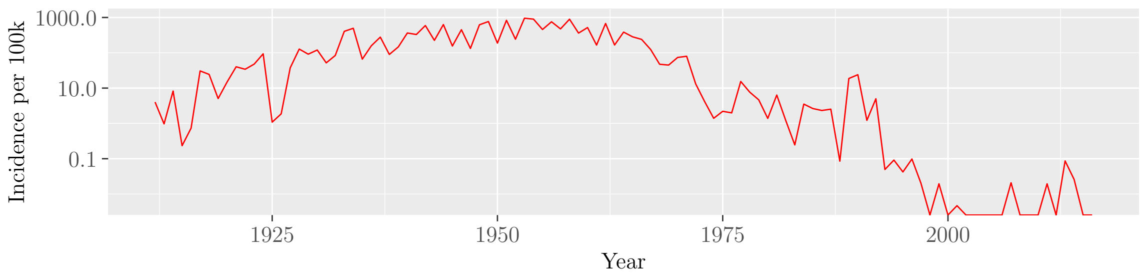

SOUTH CAROLINA
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
SOUTH CAROLINA
US
1962
1922
2002
1962
1922
2002

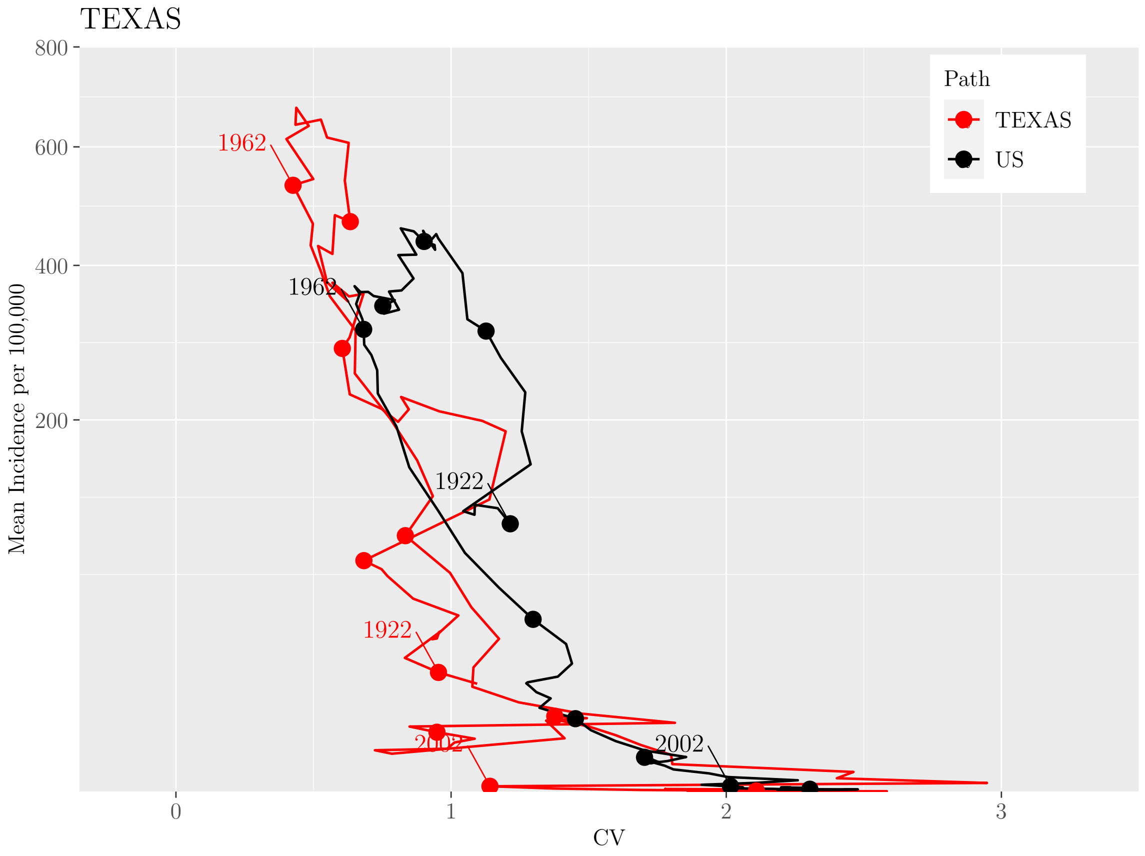

Incidence per 100k
1000
100
10
1
1925
1950
1975
2000
Year

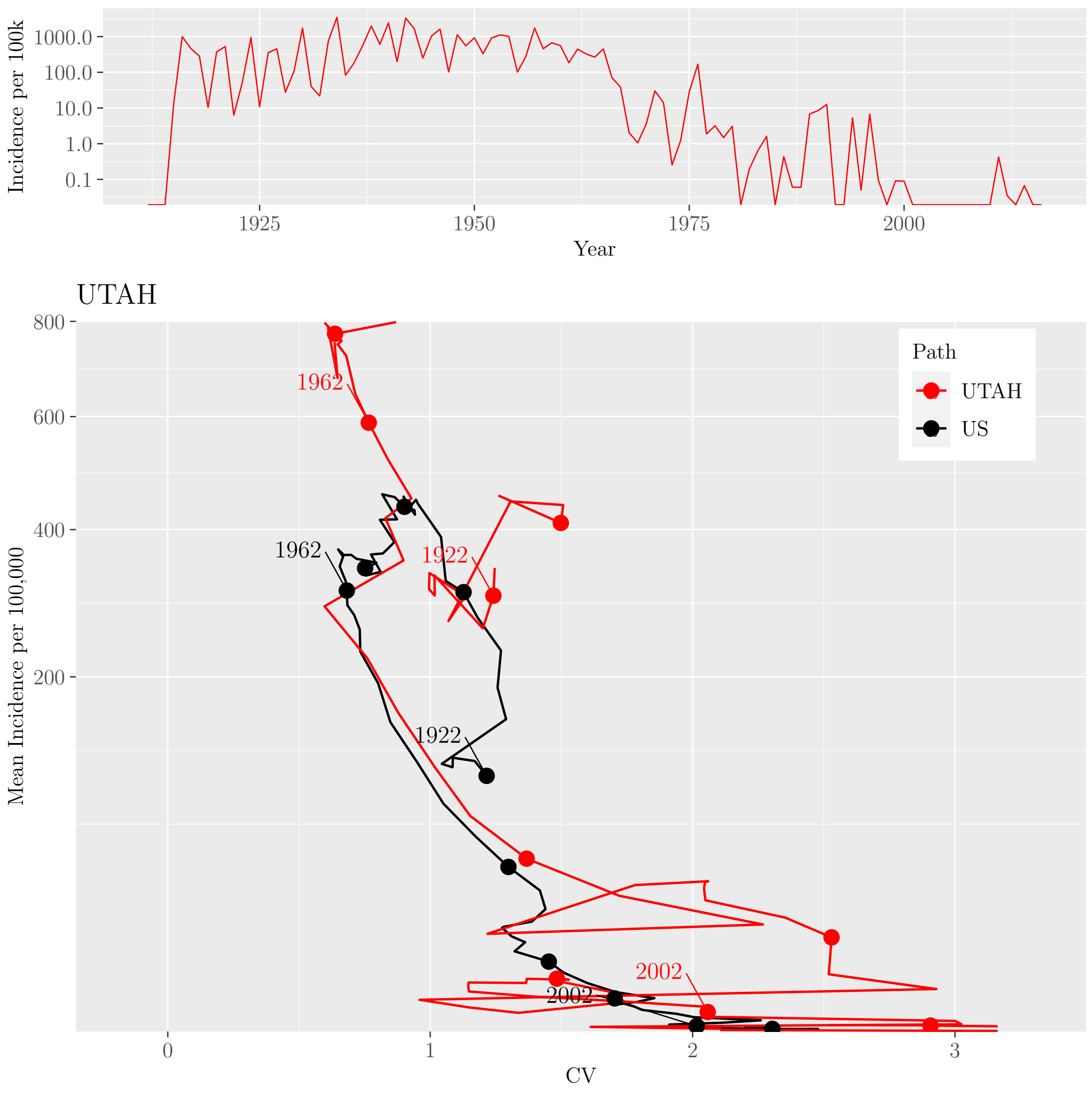

SOUTH DAKOTA
Path
SOUTH DAKOTA
US
800
600
400
200
Mean Incidence per 100,000
1962
1922
1962
1922
2002
2002
0
1
2
3
CV

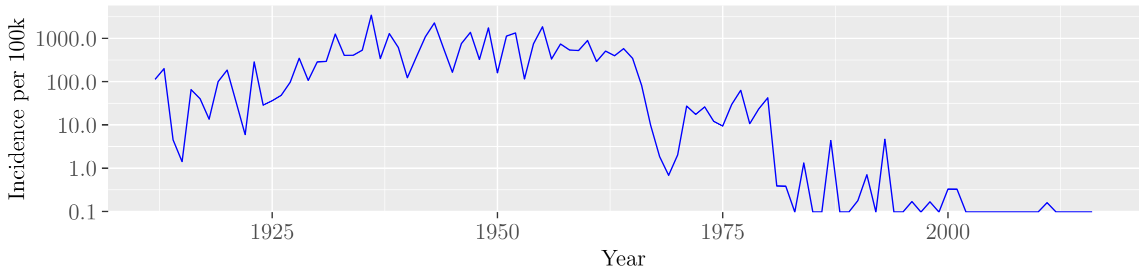

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

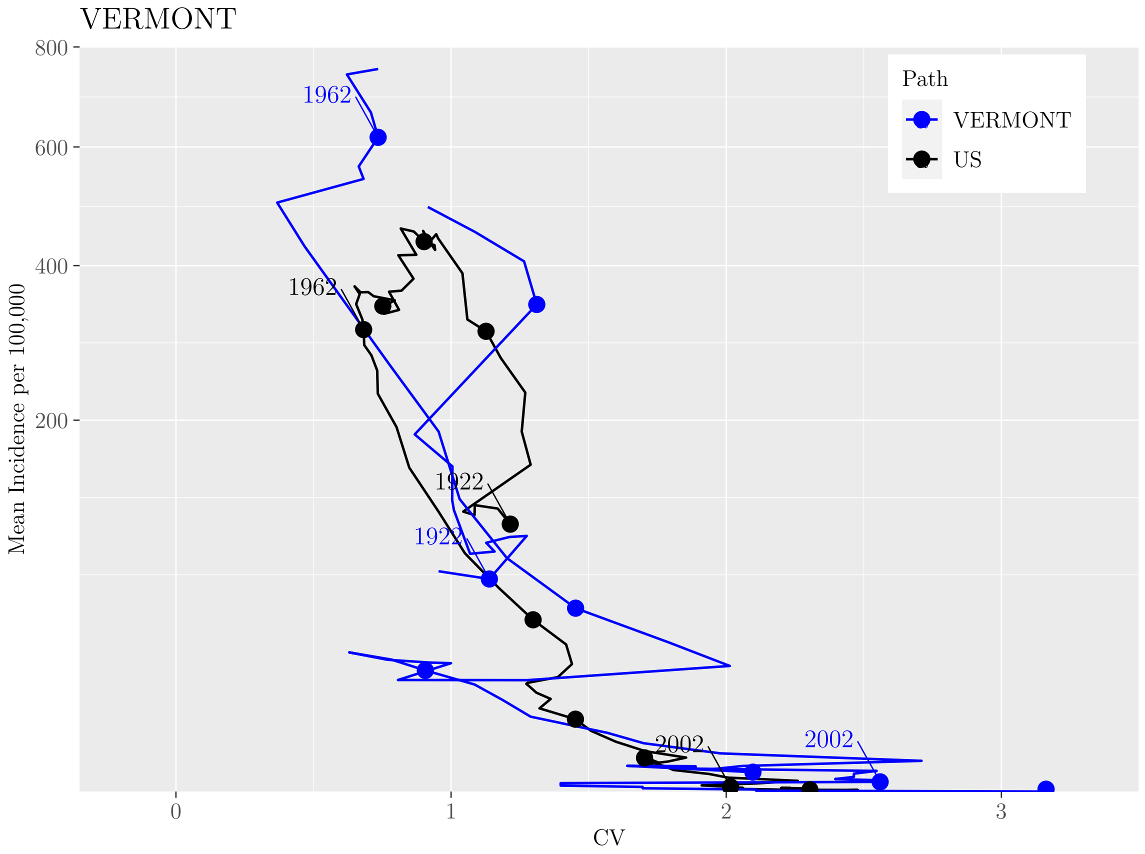

TENNESSEE
Mean Incidence per 100,000
800
600
400
200
CV
0
1
2
3
Path
TENNESSEE
US
1962
1922
2002
1962
1922
2002

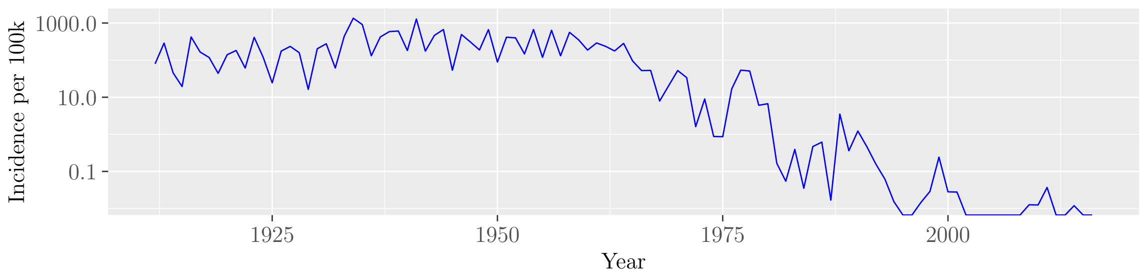

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

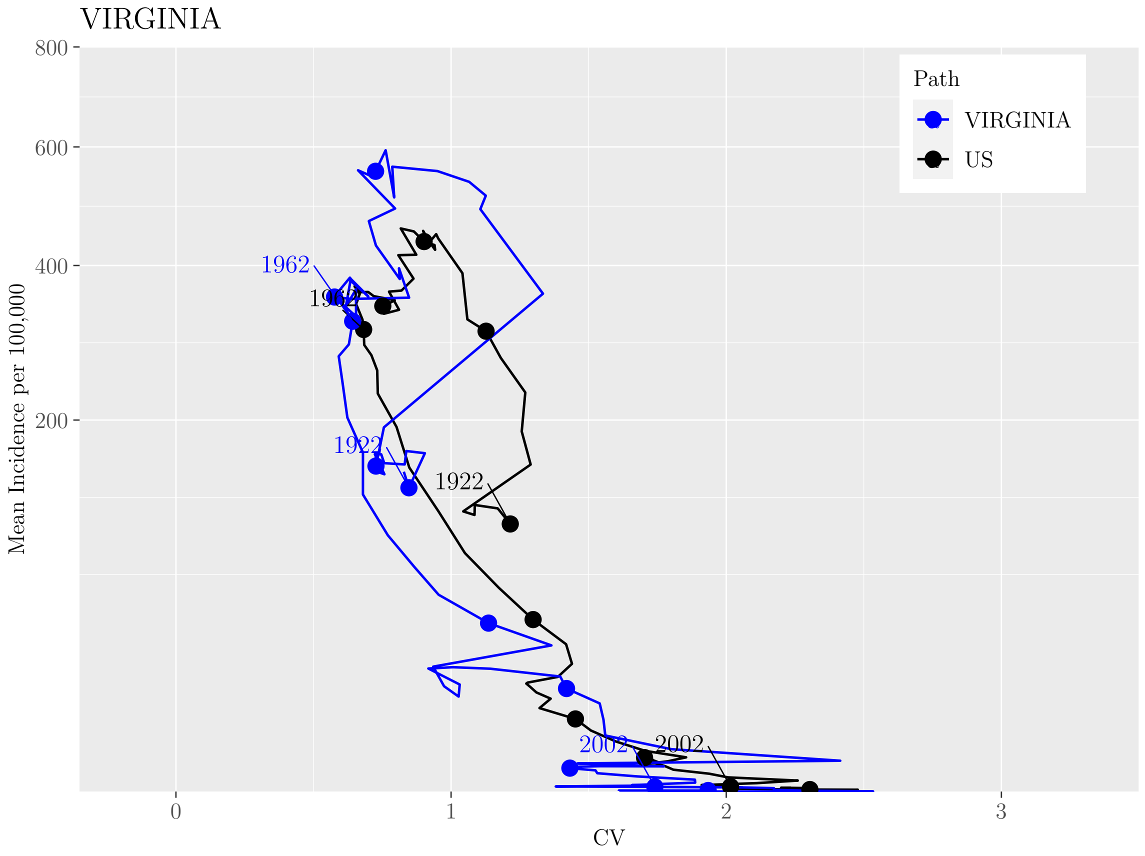

TEXAS
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
1962
1922
2002
1962
1922
2002
Path
TEXAS
US

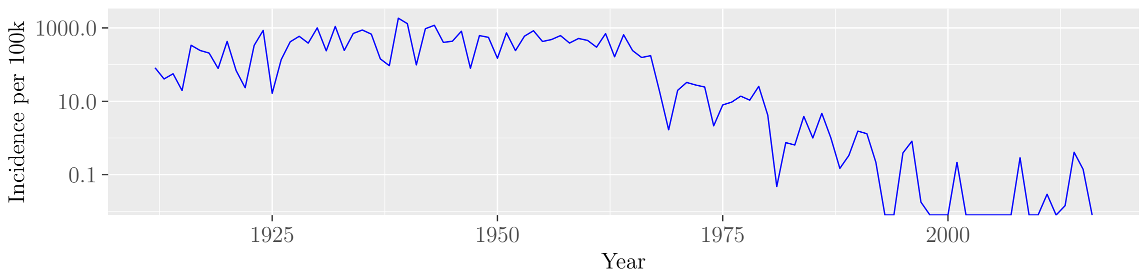

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year
UTAH
Mean Incidence per 100,000
800
600
400
200
CV
0
1
2
3
1962
1922
2002
Path
UTAH
US

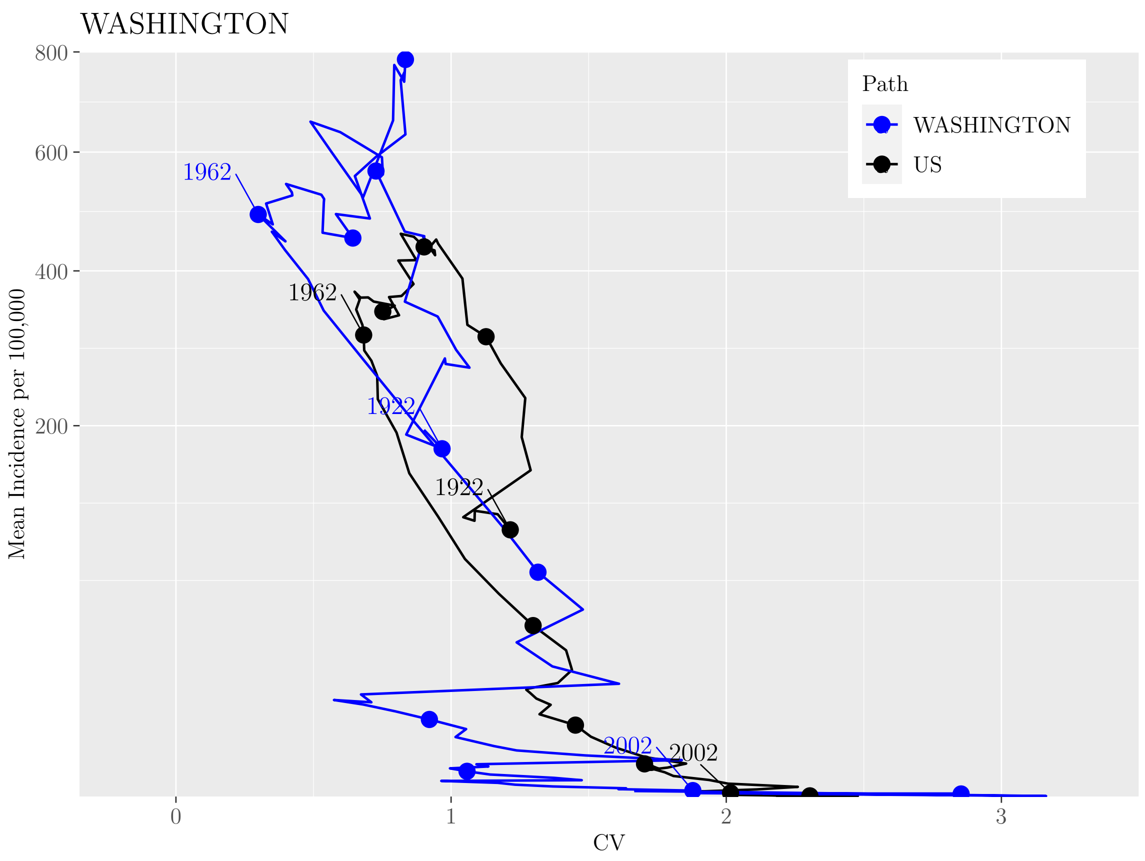

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

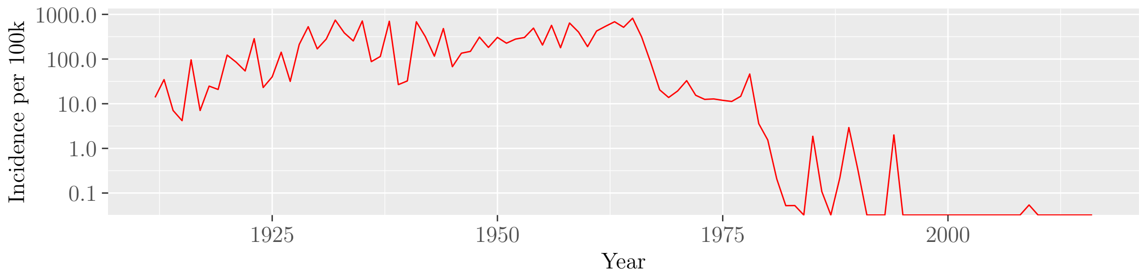

VERMONT
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
1962
1962
1922
1922
2002
2002
Path
VERMONT
US

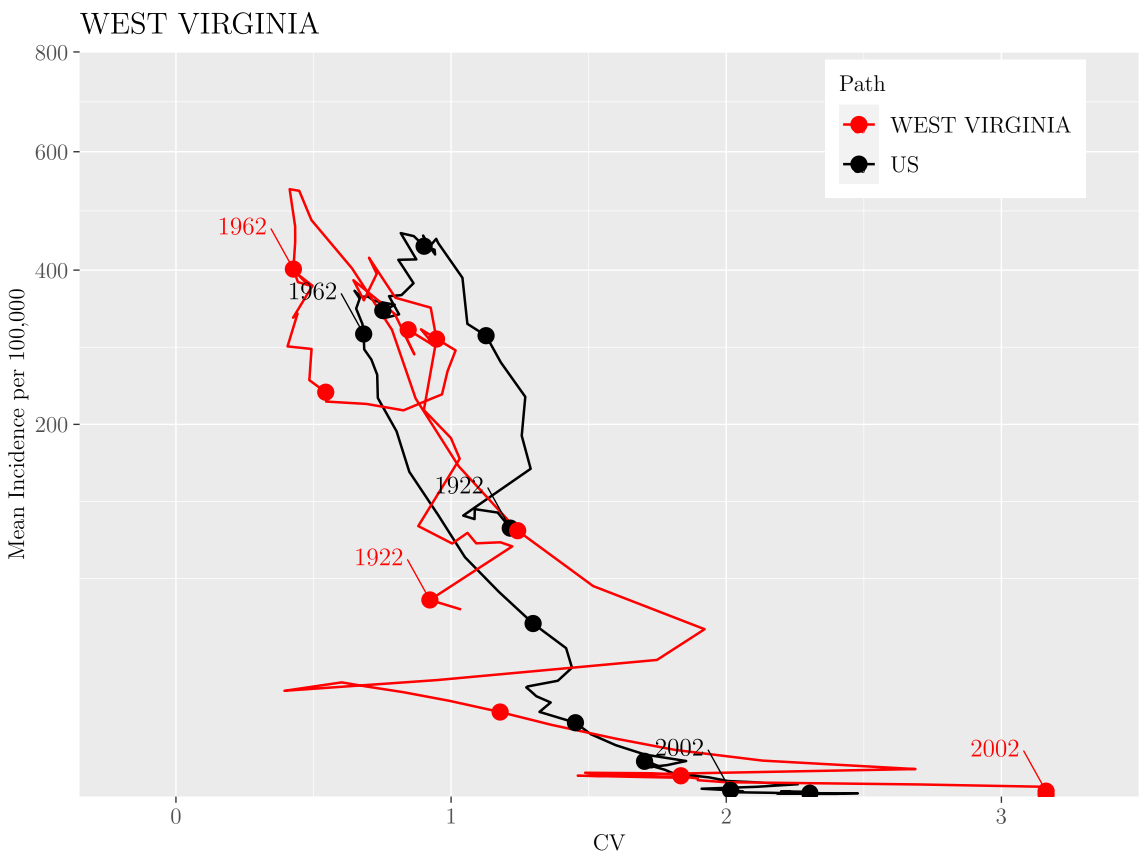

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

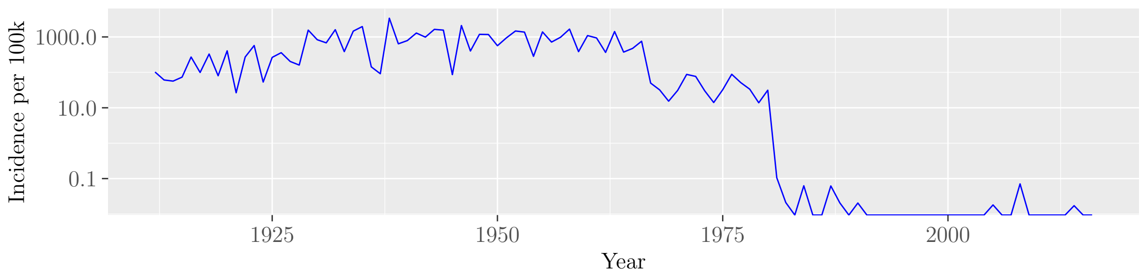

VIRGINIA
Mean Incidence per 100,000
800
600
400
200
CV
0
1
2
3
1962
1962
1922
1922
2002
2002
Path
VIRGINIA
US

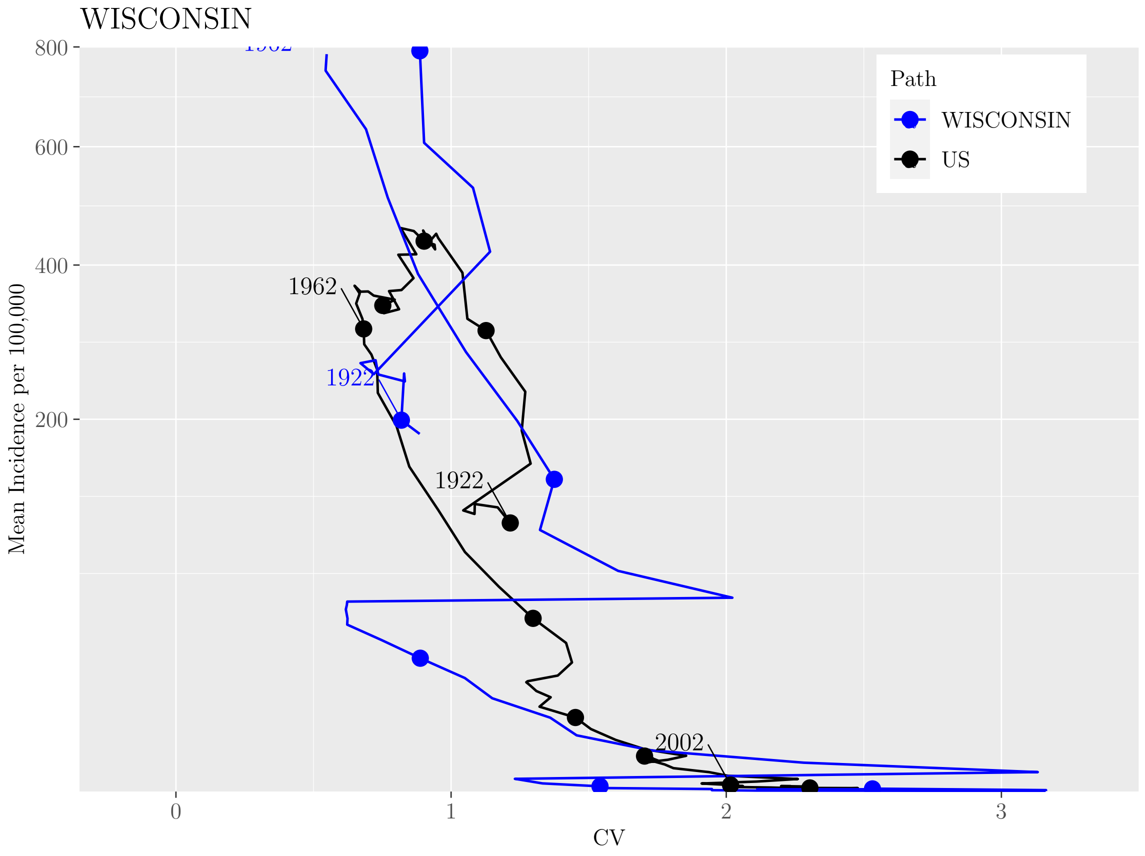

Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year

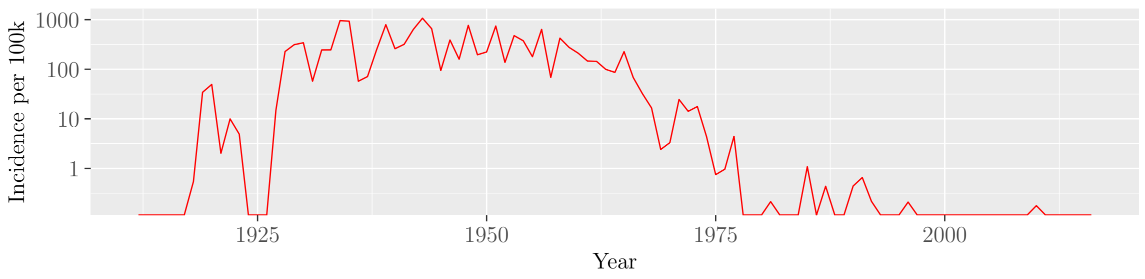

WASHINGTON
800
600
400
200
Mean Incidence per 100,000
1962
1962
1922
1922
2002
2002
0
1
2
3
CV
Path
WASHINGTON
US

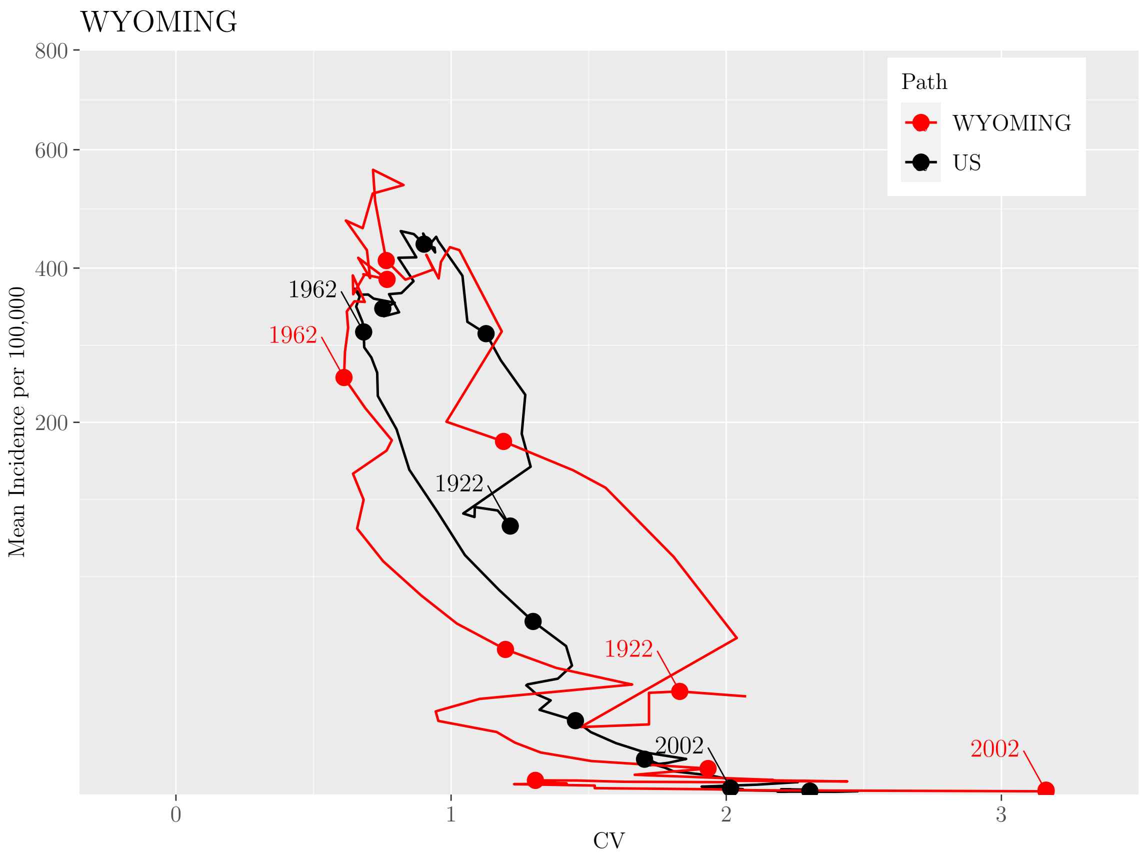

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year


WEST VIRGINIA
Path
WEST VIRGINIA
US
800
600
400
200
Mean Incidence per 100,000
1962
1962
1922
1922
2002
2002
0
1
2
3
CV


Incidence per 100k
1000.0
10.0
0.1
1925
1950
1975
2000
Year


WISCONSIN
800
600
400
200
Mean Incidence per 100,000
1962
1922
1962
1922
2002
Path
WISCONSIN
US
0
1
2
3
CV

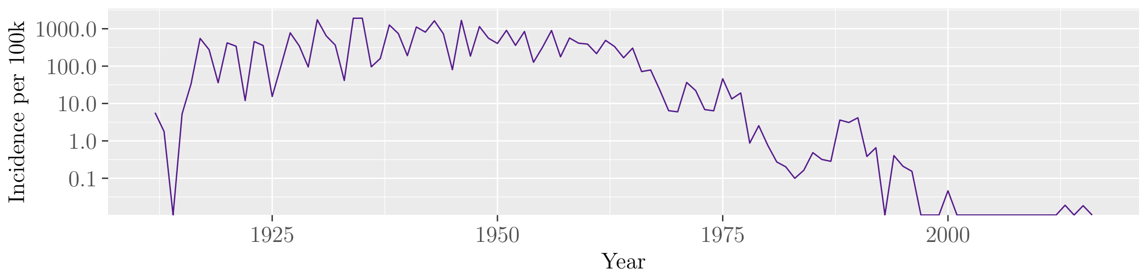


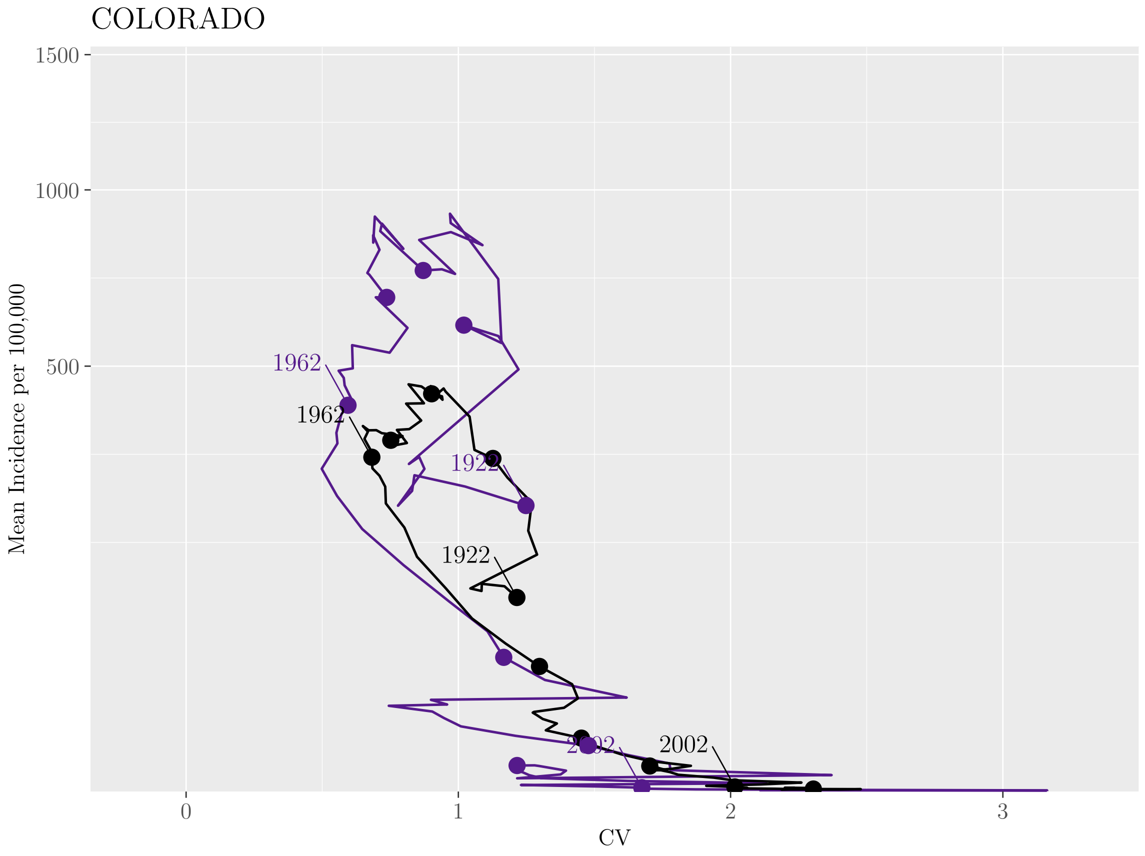


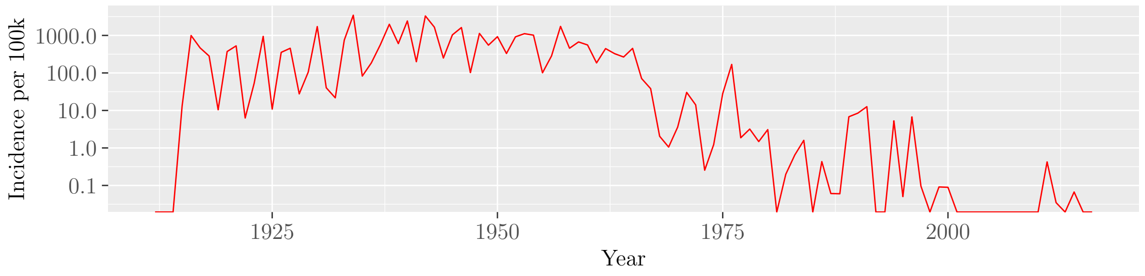
Additionally, since Colorado, Utah, Vermont, and Wisconsin had a higher maximum number reported cases than most states, we re-create the grids for these states, increasing the y-axis limit.

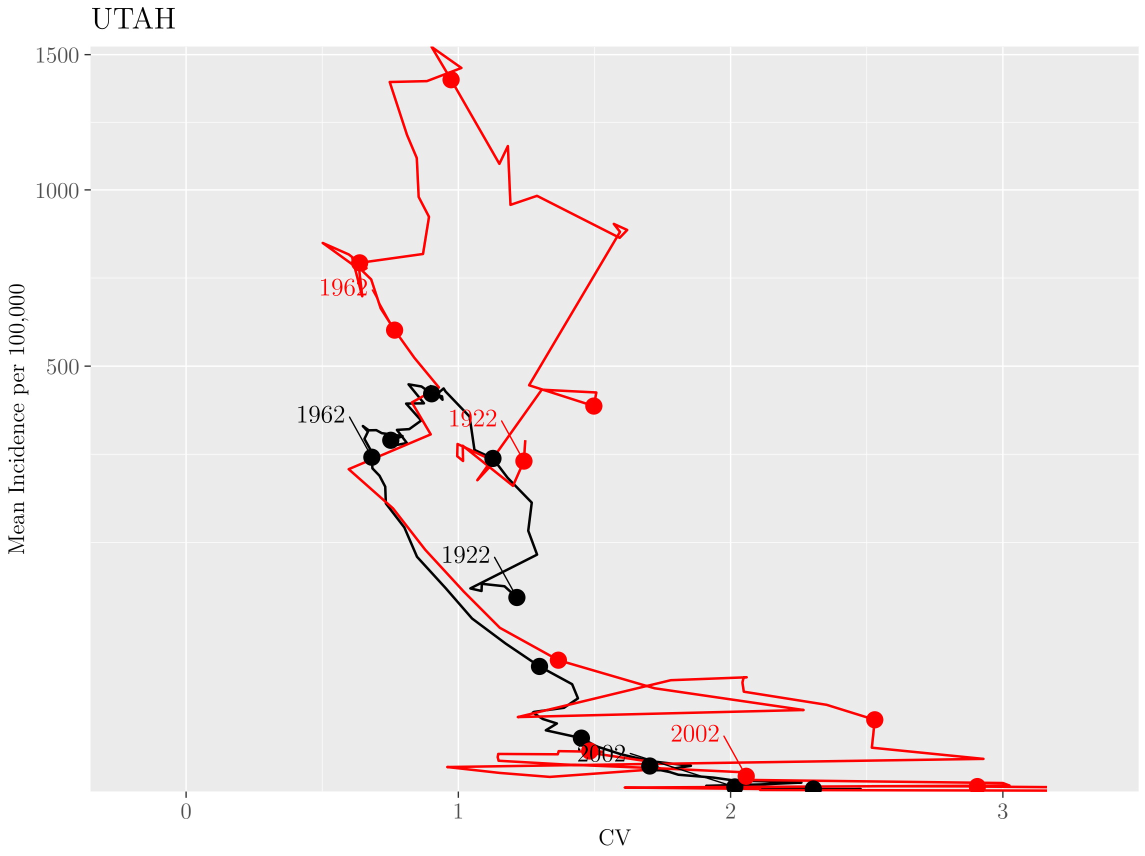
```
## Extract only Colorado, Utah, Vermont and Wisconsin as
## they are cut off
above_800 <- c("COLORADO", "UTAH", "VERMONT", "WISCONSIN")

state_above_800 <- state_paths[names(state_paths) %in% above_800]
state_ts_above_800 <- state_ts_list[names(state_paths) %in% above_800]
```

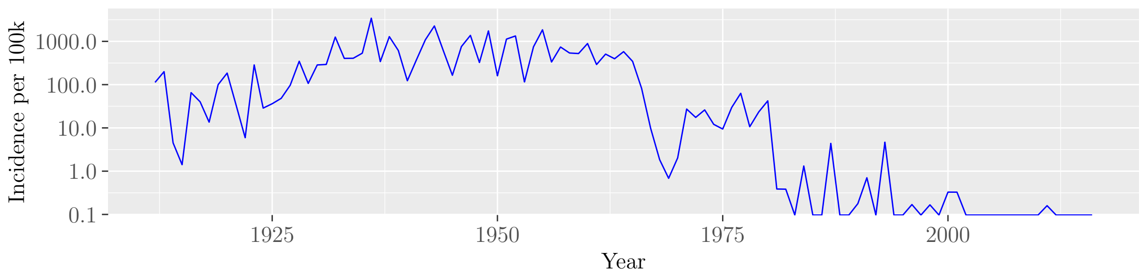
```r
## Get rid of legends and make y limit the max value in
## Utah to see the plots fully
for (i in 1:length(state_above_800)) {
  state_above_800[[i]] <- state_above_800[[i]] +
    scale_y_sqrt("Mean Incidence per 100,000",
                 limits = c(min(c(min(state_data$Incidence,
                                      na.rm = TRUE),
                                  min(data_us$Incidence,
                                      na.rm = TRUE))),
                            max(Incidence_states$UTAH))) +
    theme(legend.position = "none")

  grobs_state <- list(state_ts_above_800[[i]], state_above_800[[i]])
  lay_state <- rbind(1, 2, 2, 2)
  grid.arrange(grobs = grobs_state, layout_matrix = lay_state)
}
```

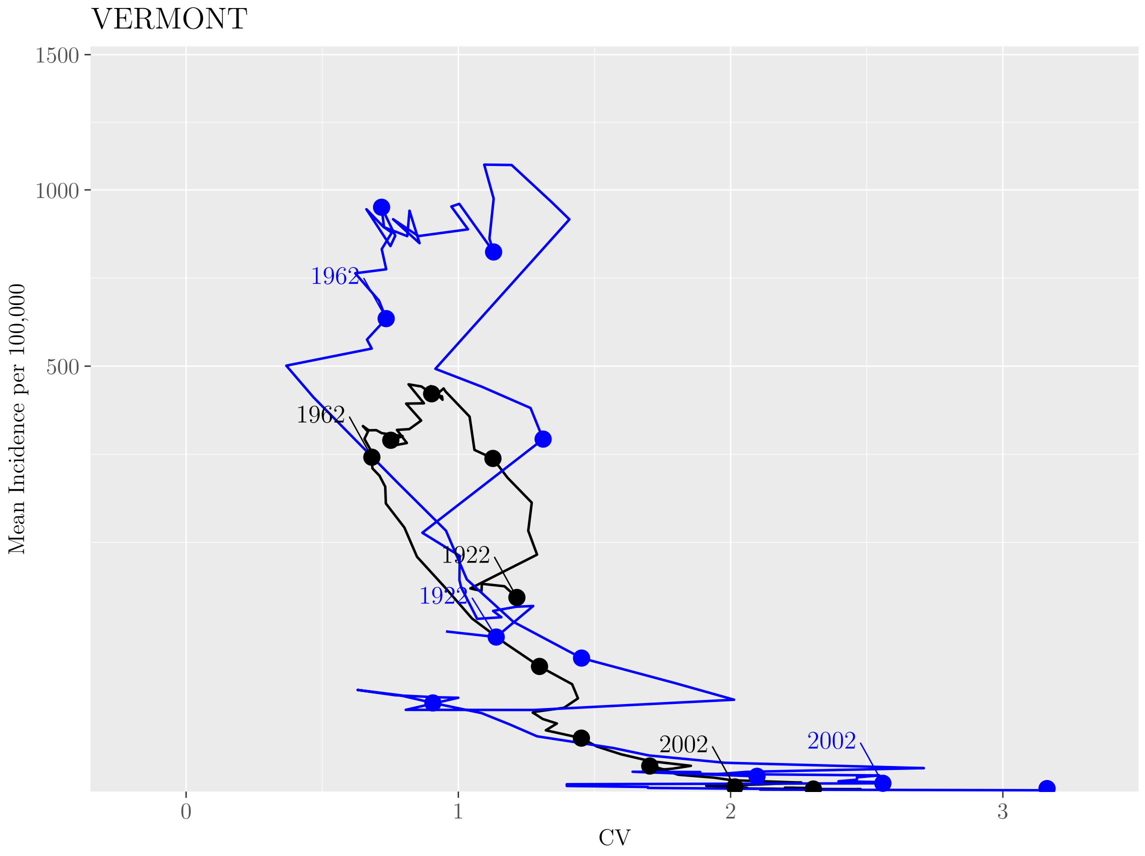

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year

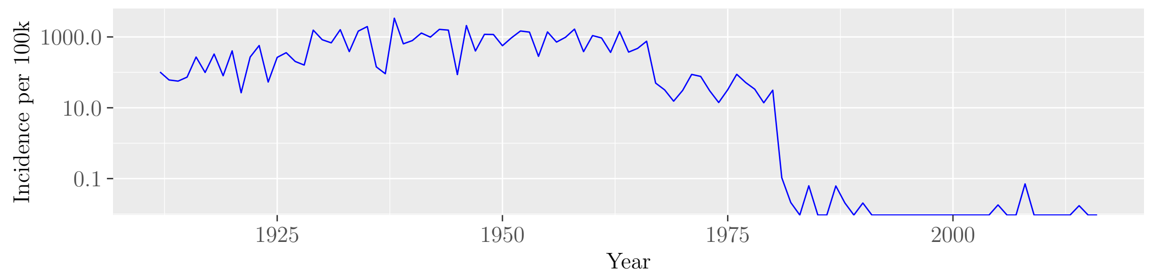

COLORADO
Mean Incidence per 100,000
1500
1000
500
1962
1962
1922
1922
2002
2002
0
1
2
3
CV

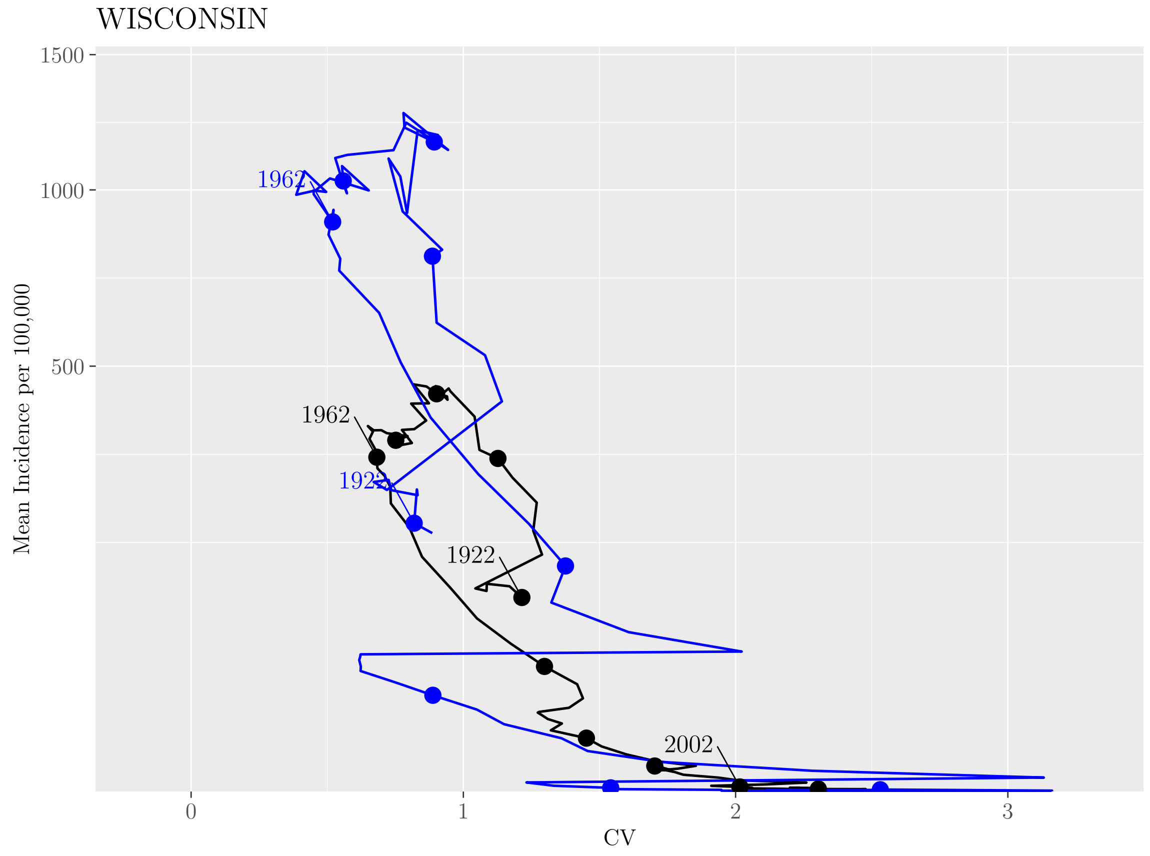

Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year


UTAH
1500
1000
500
Mean Incidence per 100,000
1962
1962
1922
1922
2002
2002
0
1
2
3
CV


Incidence per 100k
1000.0
100.0
10.0
1.0
0.1
1925
1950
1975
2000
Year


VERMONT
1500
1000
500
Mean Incidence per 100,000
1962
1962
1922
1922
2002
2002
0
1
2
3
CV





Through the inspection of the time series of each US state, we discovered that it wasn't until roughly 1959 that all states began reporting measles uniformly. Thus, conducting the comparison before this date may skew the results due to increased variation. We re-create the US canonical path, greying-out the path before 1960 in the following figure.

```r
## Create a column that colours the path based on whether
## the reporting was varied or uniform across
## states for that year

data_us$Reporting <- 0
```

```r
point_mean_us$Reporting <- 0

unif_year_val <- 1959

for (i in 1:nrow(data_us)) {
  if (as.numeric(data_us$year[i]) <= unif_year_val) {
    data_us$Reporting[i] <- "Varied"
  } else {
    data_us$Reporting[i] <- "Uniform"
  }
}

for (i in 1:nrow(point_mean_us)) {
  if (as.numeric(point_mean_us$year[i]) <= unif_year_val) {
    point_mean_us$Reporting[i] <- "Varied"
  } else {
    point_mean_us$Reporting[i] <- "Uniform"
  }
}

path_colours <- c(Varied = "darkgrey", Uniform = "black")

true_path_greyed <- ggplot()+
  geom_path(data = data_us,
            aes(CV, Incidence, col = Reporting),
            lty = 1, lwd = 1) +
  geom_point(data = point_mean_us,
             aes(CV, Incidence, col = Reporting),
             pch = 15, size = 2) +
  geom_text(data = point_mean_us,
            aes(label = year , col = Reporting, ''),
            position = position_nudge(x = -1, y = -1),
            hjust = -0.45,
            vjust = -0.75,
            x = point_mean_us$CV,
            y = sqrt(point_mean_us$Incidence)) +
  scale_y_sqrt("Mean Incidence per 100,000",
               limits = c(min(data_us$Incidence),
                          max(data_us$Incidence))) +
  scale_x_continuous("CV", limits = c(min(data_us$CV),
                                      max(data_us$CV))) +
  scale_colour_manual(values = path_colours) +
  labs(colour = "Reporting")
true_path_greyed
```

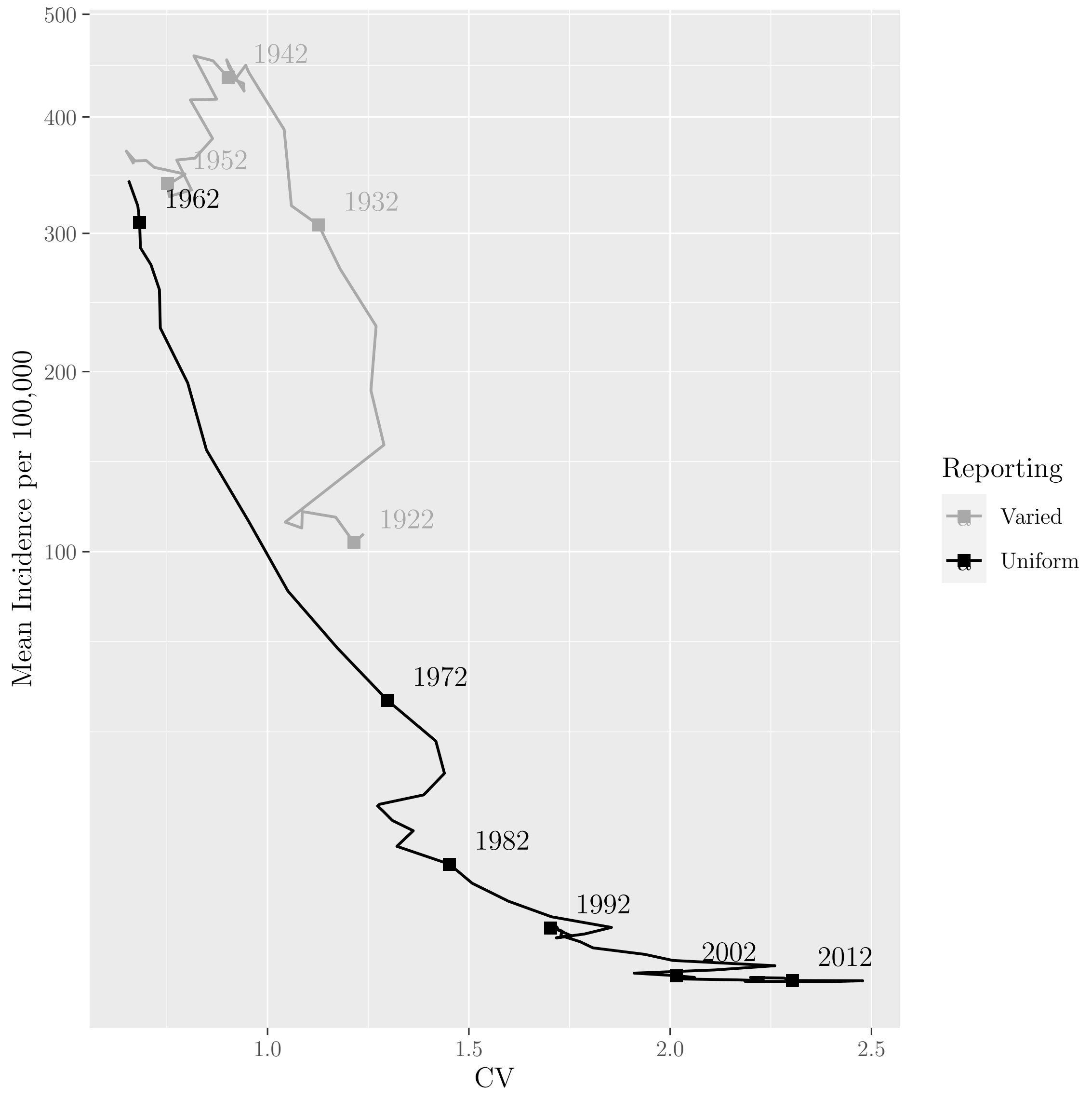


Figure 26: US canonical path with the path greyed out before 1960.

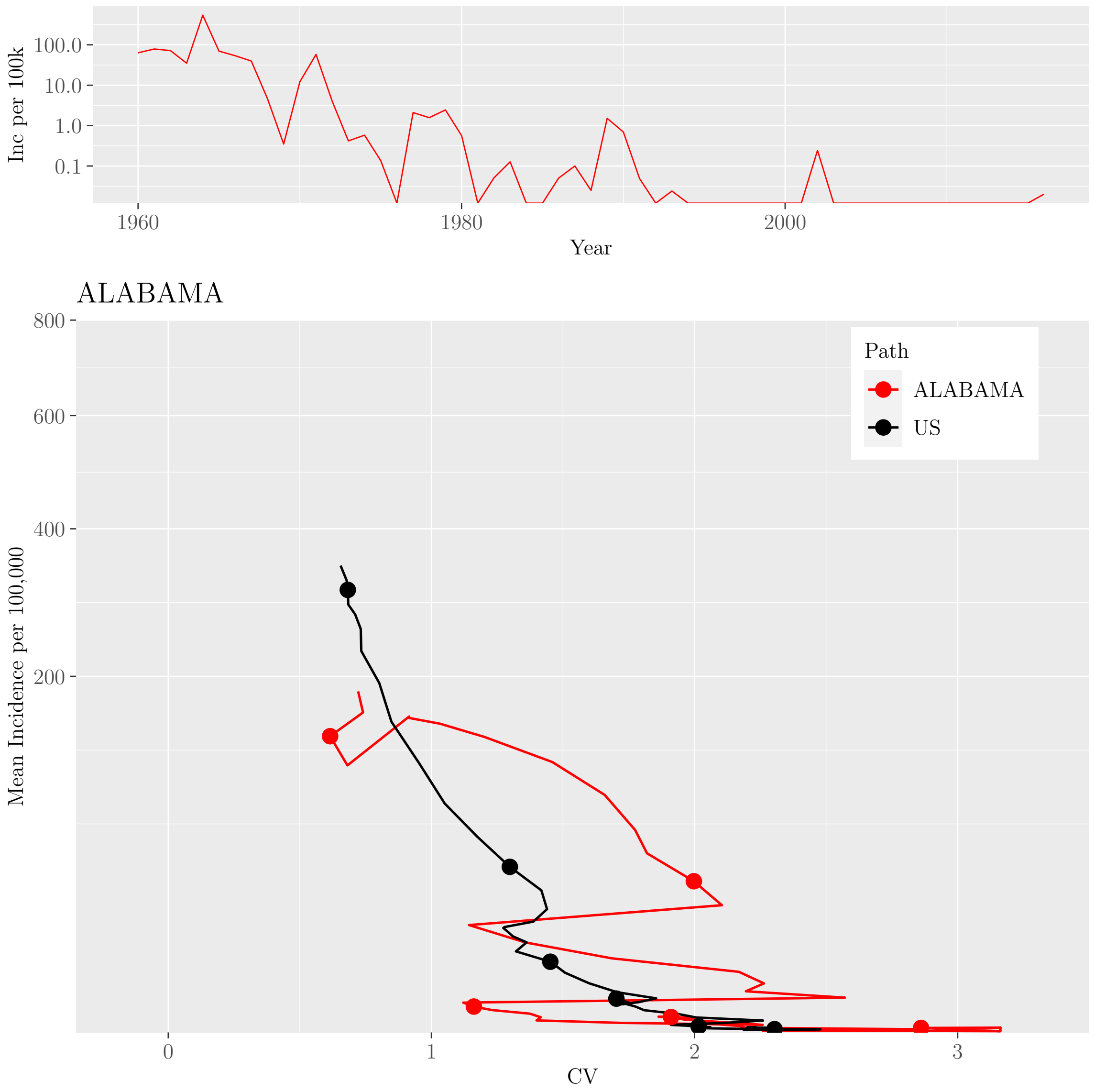
We now re-create the state comparison plots and the state time series, for only 1960 to 2016 using the following code.

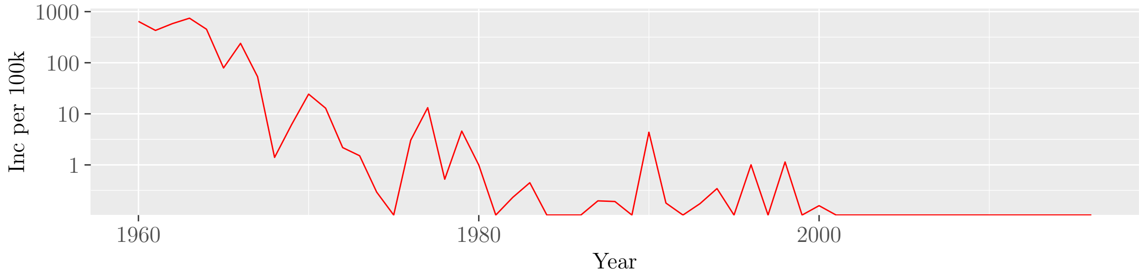
```r
data_us_unif <- data_us_filter %>%
  mutate(Year = as.numeric(data_us_filter$Year)) %>%
  filter(Year > unif_year_val)

unif_years <- as.numeric(data_us_unif$Year)

point_mean_us_unif <- point_mean_us_filter %>%
  mutate(Year = as.numeric(point_mean_us_filter$Year)) %>%
  filter(Year > unif_year_val)

point_unif_years <- as.numeric(point_mean_us_unif$Year)

unif_paths <- list()
unif_state_data <- list()
unif_point_state <- list()

for (i in 1:nstates) {
  state <- state_names[i]

  state_unif <- state_data_list[[state]] %>%
    filter(Year %in% unif_years)

  unif_state_data[[i]] <- state_unif

  point_state_unif <- state_point_list[[state]] %>%
    filter(Year %in% point_unif_years)

  unif_point_state[[i]] <- point_state_unif


  ## Find the max incidence for each state
  max_incidence_per_state[i] <-
    max(unif_state_data$Incidence, na.rm = TRUE)

  ## Create long data frames containing the US and state path data
  all_data_unif <- rbind(data_us_unif, state_unif)
  all_point_data_unif <- rbind(point_mean_us_unif, point_state_unif)

  ## Create the plot that plots both the state and US canonical path
  state_plot <- ggplot() +
    geom_path(data = all_data_unif, aes(CV, Incidence, colour = Path),
              lty = 1, lwd = 1) +
    geom_point(data = all_point_data_unif,
               aes(CV, Incidence, colour = Path),
               size = 3) +
    scale_y_sqrt("Mean Incidence per 100,000",
                 limits = c(min(all_data$Incidence, na.rm = TRUE),
                            800)) +
    scale_x_continuous("CV",
                       limits = c(-.35, 3.5)) +
    scale_colour_manual(values = colour_each_state[[i]]) +
    coord_cartesian(expand = c(0,0)) +
    labs(colour="Path") +
    theme(legend.position = c(.95, .99),
          legend.justification = c("right", "top"),
          legend.box.just = "right",
          legend.margin = margin(6, 6, 6, 6),
          legend.text = element_text(size = text_size),
          legend.title = element_text(size = text_size),
          axis.text = element_text(size = text_size),
          axis.title = element_text(size = text_size)
    ) +
    ggtitle(state)

  unif_paths[[i]] <- state_plot
}

names(unif_paths) <- state_names
names(unif_state_data) <- state_names
names(unif_point_state) <- state_names
```

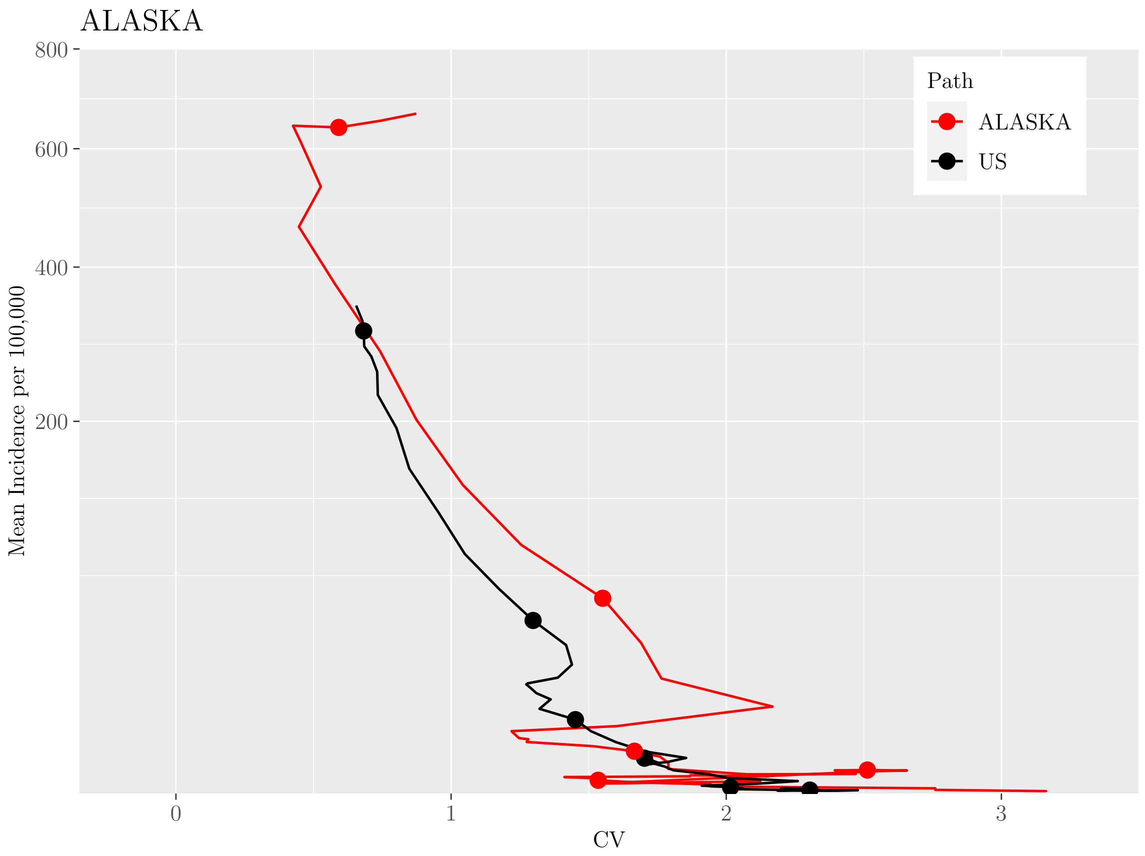
```
us_only_inc_unif <- us_only_inc %>%
  filter(Year > unif_year_val)

us_only_inc_unif$Path <- "US"

us_ts_plot_unif <- ggplot() +
  geom_path(data = us_only_inc_unif,
            aes(Year, Incidence, col = Path)) +
  scale_y_log10("Inc per 100k",
                labels =
                  function(x) format(x, scientific = FALSE)) +
  scale_colour_manual(values = colour_each_state[[1]]) +
```

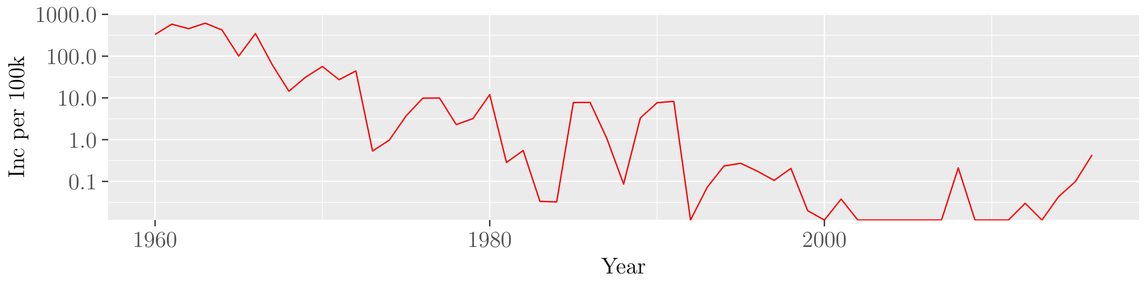
```r
  theme(
    legend.position = "none",
    legend.text = element_text(size = text_size),
    legend.title = element_text(size = text_size),
    axis.text = element_text(size = text_size),
    axis.title = element_text(size = text_size)
  )

state_ts_unif <- list()
state_ts_unif_plot <- list()

for (i in 1:nstates) {
  state <- state_names[i]

  state_inc_data <- long_incidence_data %>%
    filter(Region == state & Year > unif_year_val) %>%
    mutate(Path = state)

  state_ts_unif[[i]] <- state_inc_data

  state_ts_unif_plot[[i]] <- ggplot() +
    geom_path(data = state_inc_data,
              aes(Year, Incidence, col = Path)) +
    scale_y_log10("Inc per 100k",
                  labels =
                    function(x) format(x, scientific = FALSE)) +
    scale_colour_manual(values = colour_each_state[[i]]) +
    theme(
      legend.position = "none",
      legend.text = element_text(size = text_size),
      legend.title = element_text(size = text_size),
      axis.text = element_text(size = text_size),
      axis.title = element_text(size = text_size)
    )
}

names(state_ts_unif) <- state_names
names(state_ts_unif_plot) <- state_names
```

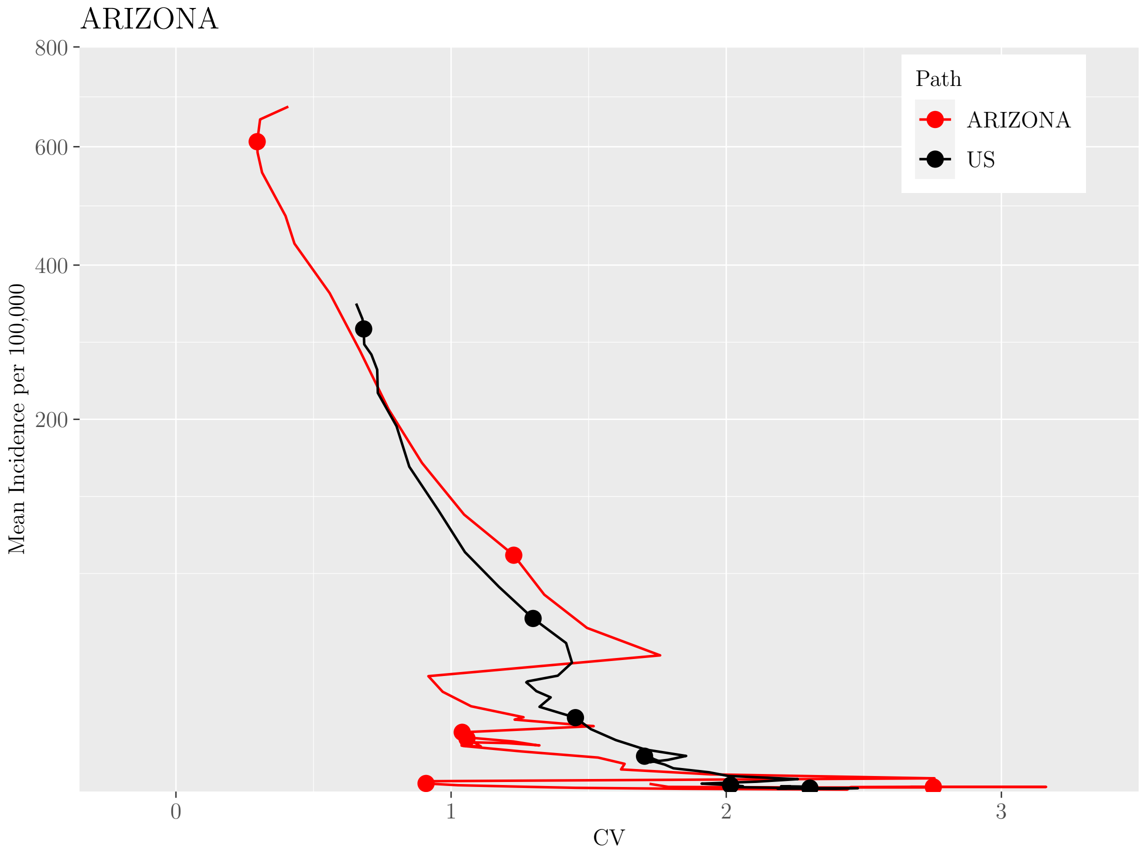
```
for (i in 1:nstates) {
  final_grobs <- list(state_ts_unif_plot[[i]], unif_paths[[i]])

  lay_final <- rbind(1, 3, 3, 3)

  grid.arrange(grobs = final_grobs, layout_matrix = lay_final)
}
```

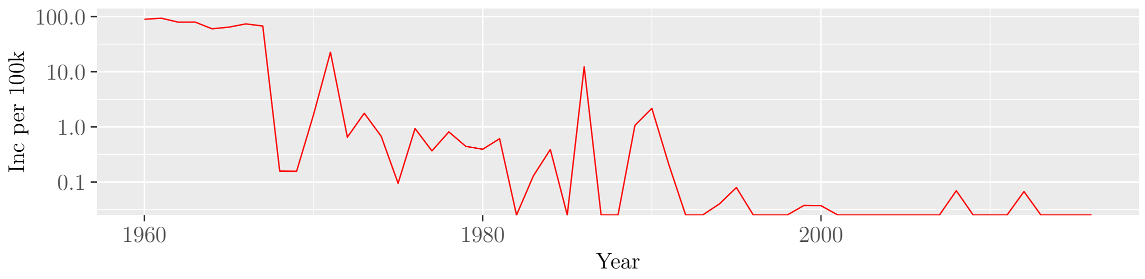



1000
100
10
1
Inc per 100k
1960
1980
2000
Year

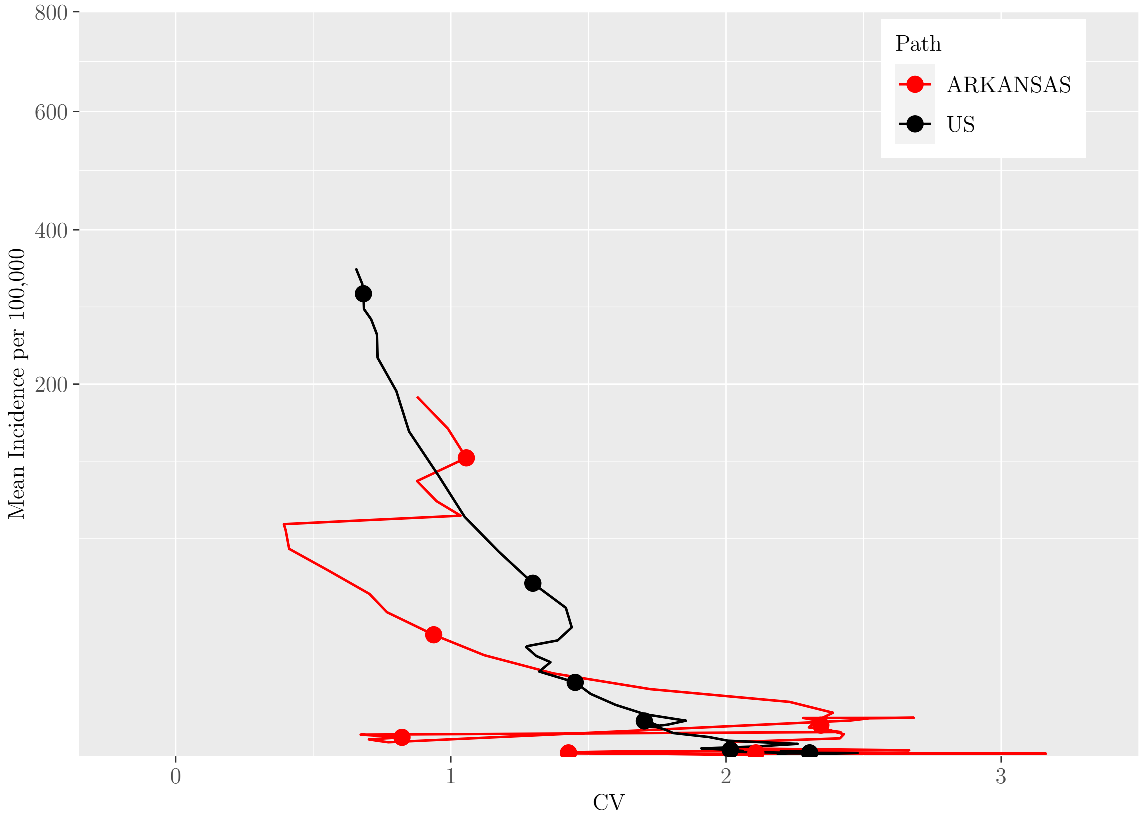

ALASKA
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
ALASKA
US

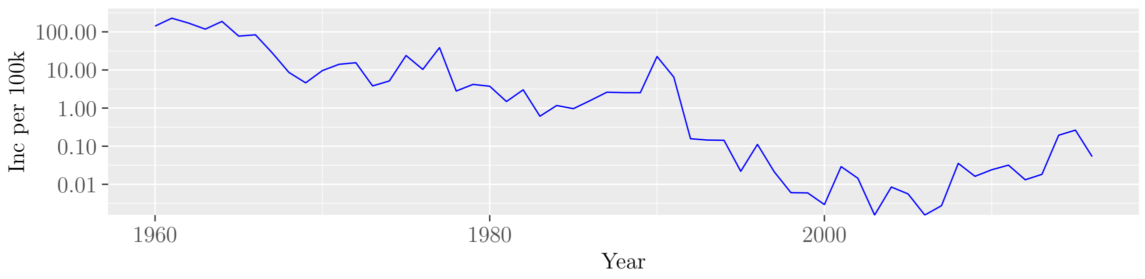

1000.0
100.0
10.0
1.0
0.1
Inc per 100k
1960
1980
2000
Year

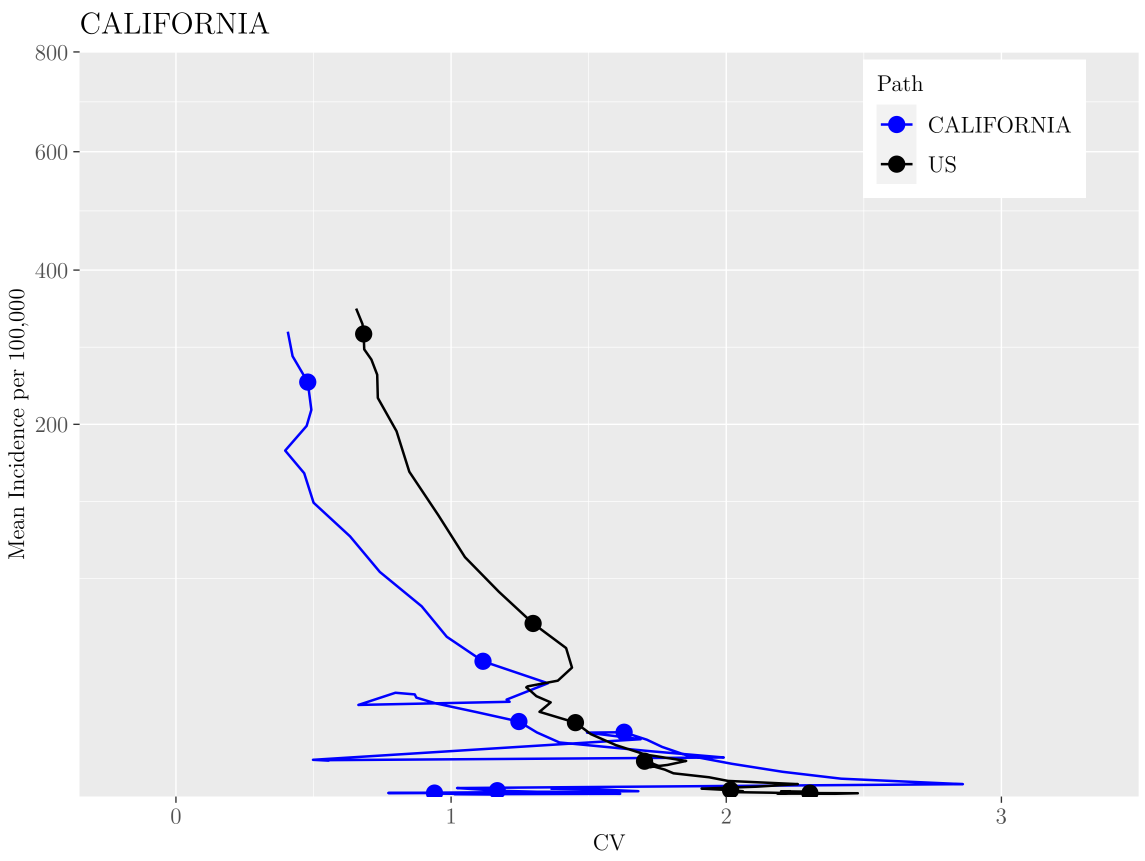

ARIZONA
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
ARIZONA
US

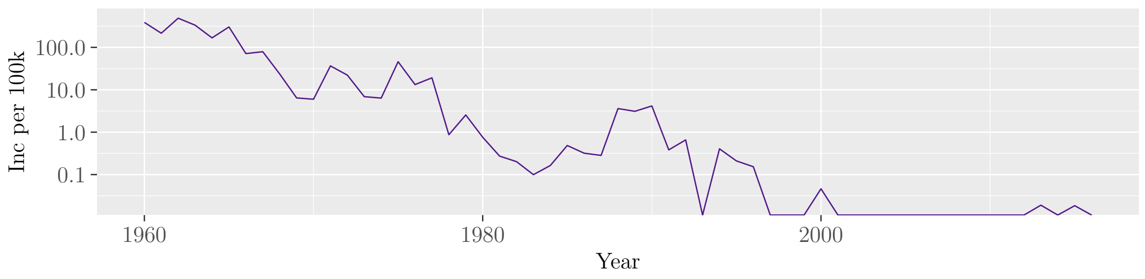

Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

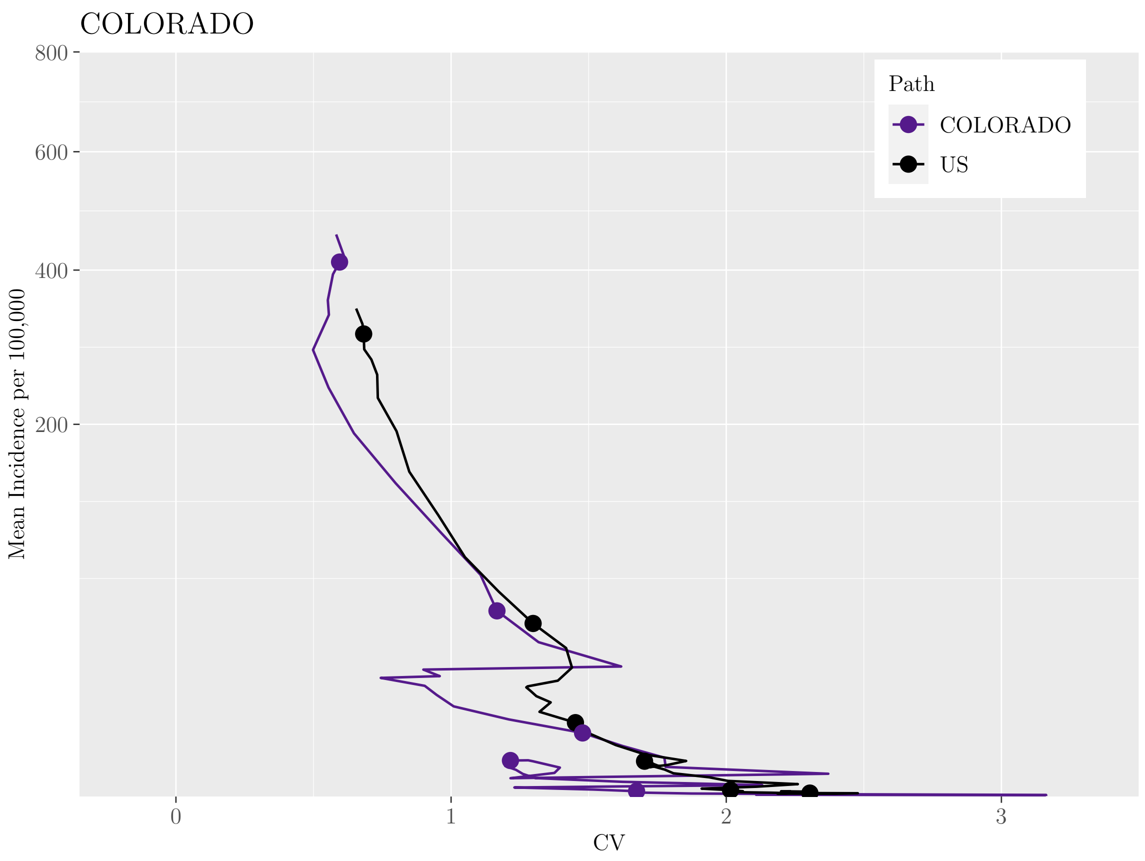
ARKANSAS

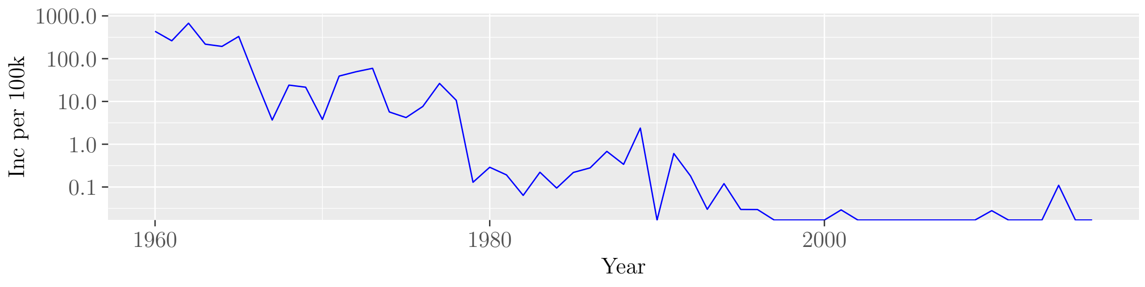

800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
ARKANSAS
US

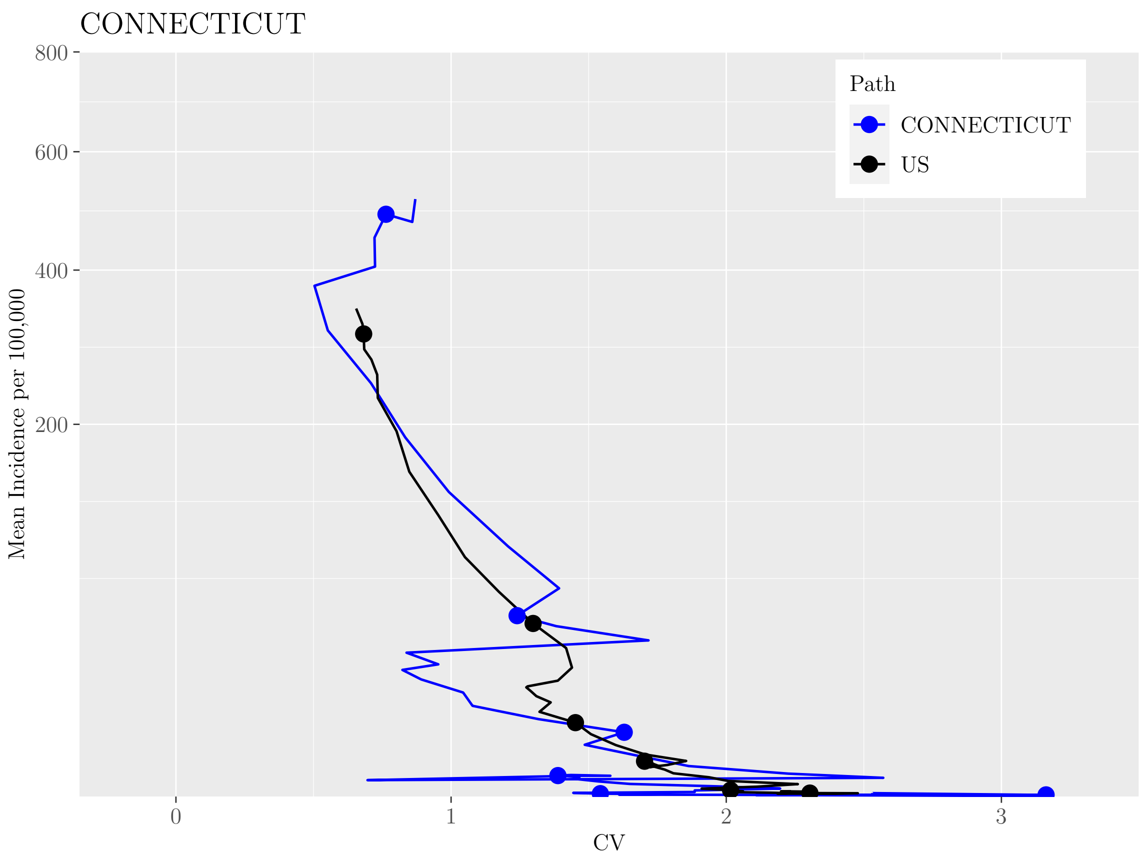

100.00
10.00
1.00
0.10
0.01
Inc per 100k
1960
1980
2000
Year

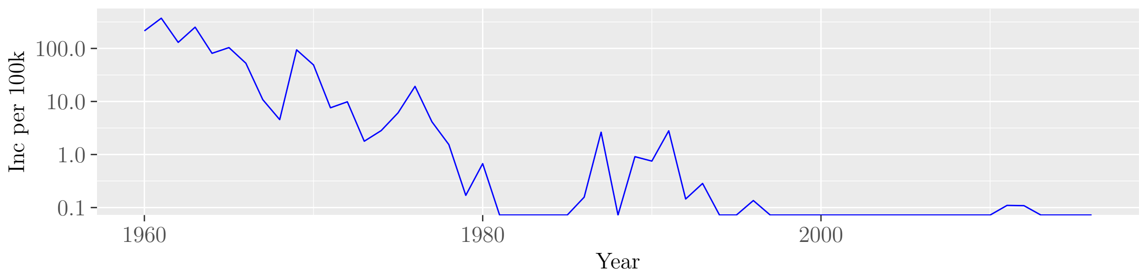

CALIFORNIA
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
CALIFORNIA
US

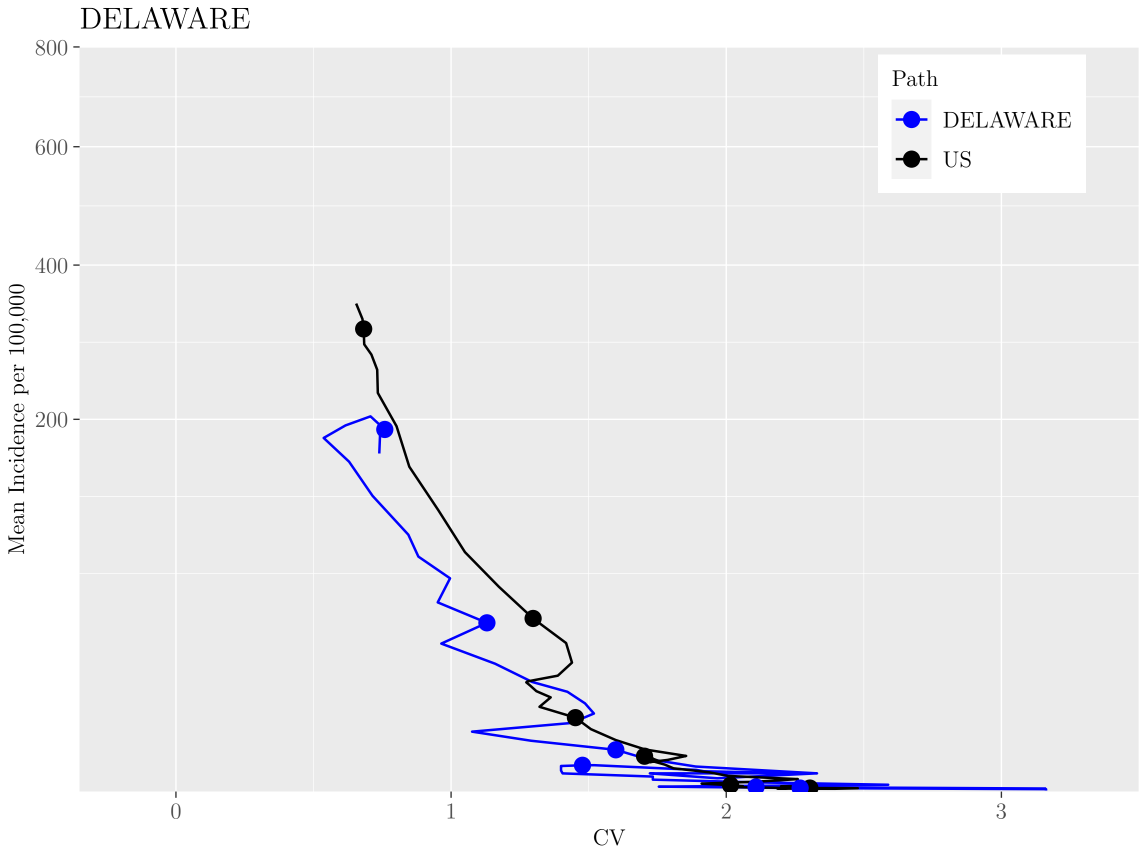

Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

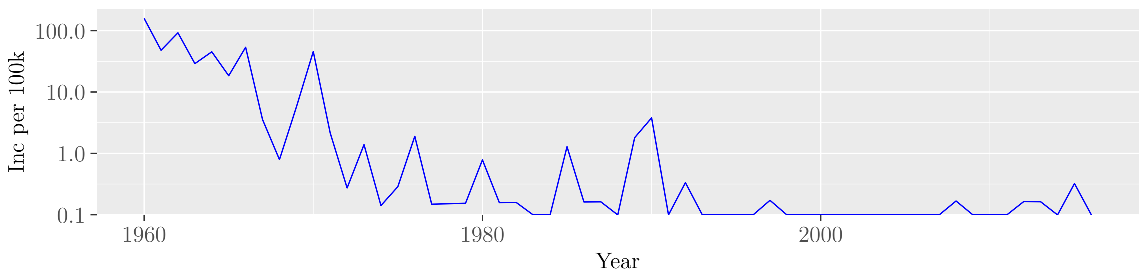

COLORADO
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
COLORADO
US

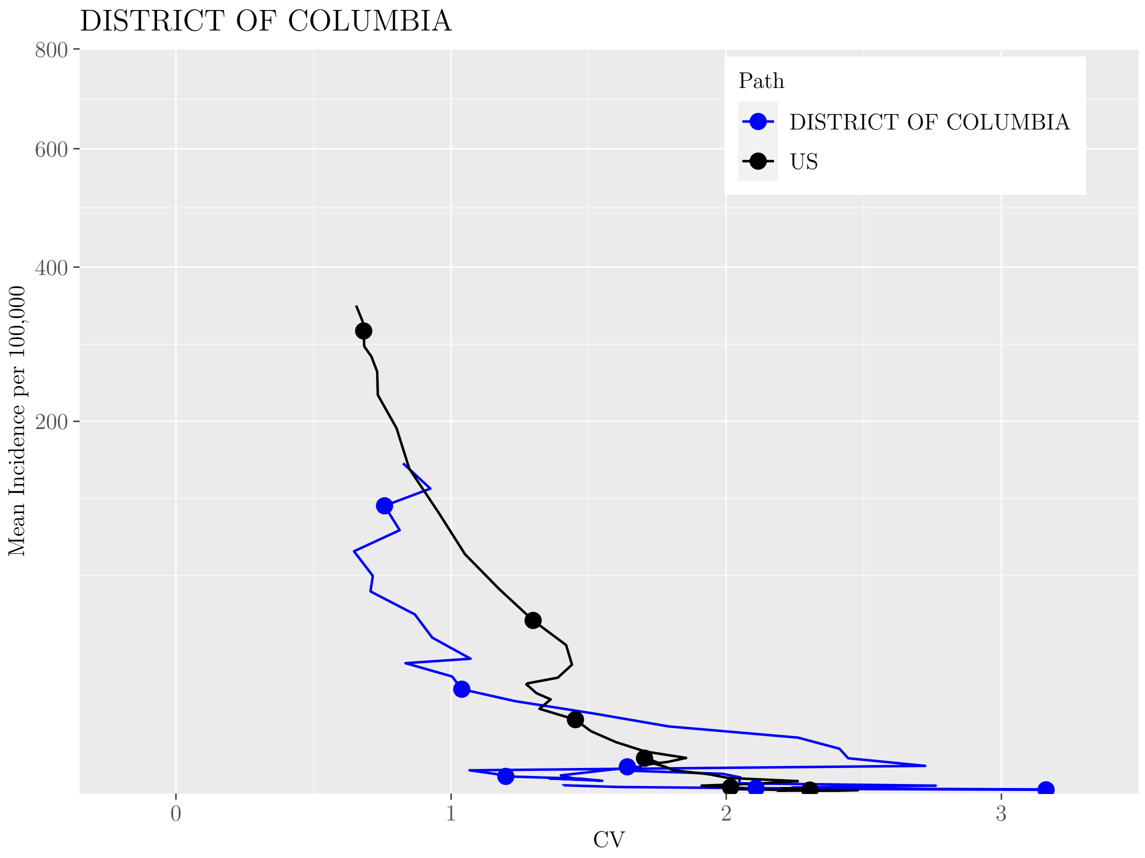

1000.0
100.0
10.0
1.0
0.1
Inc per 100k
1960
1980
2000
Year

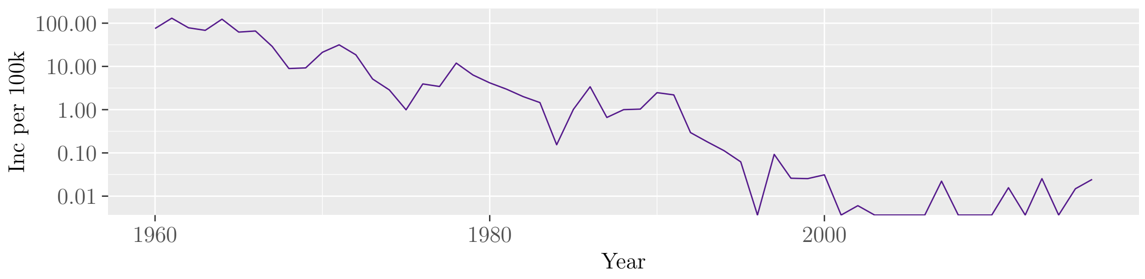

CONNECTICUT
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
CONNECTICUT
US

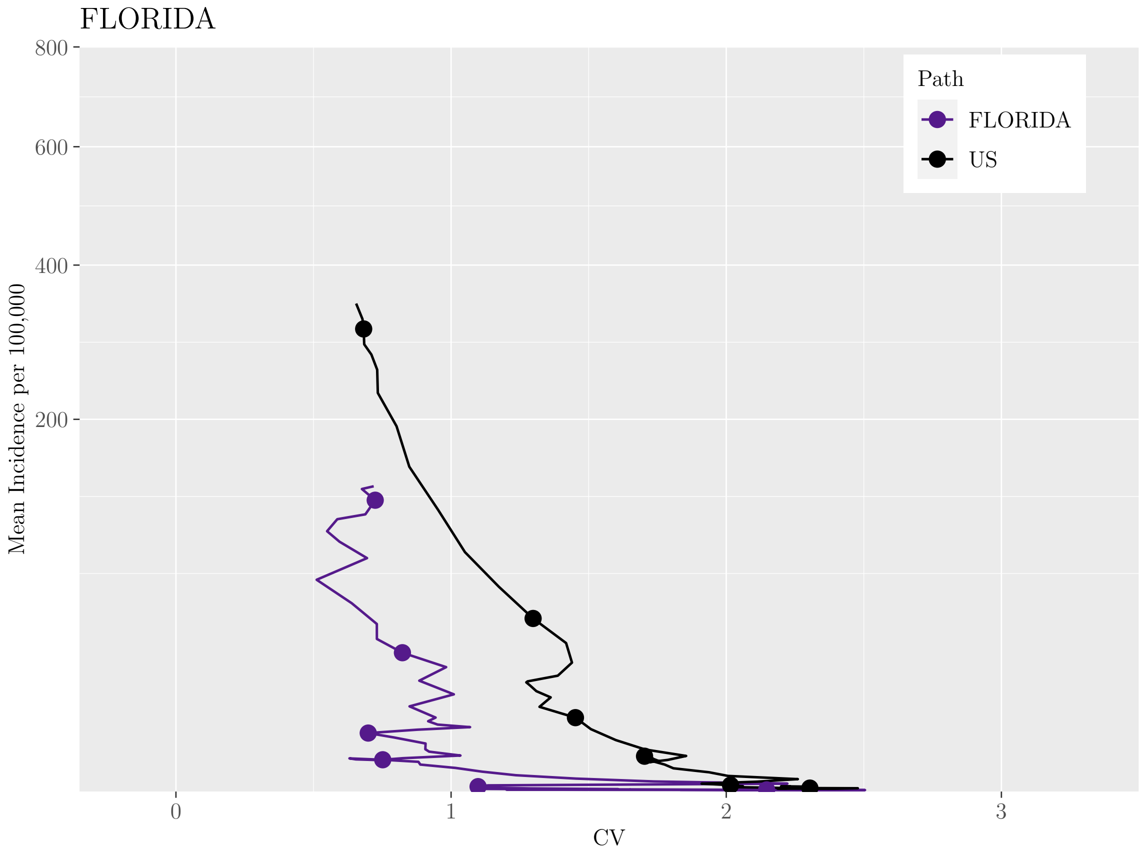

Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

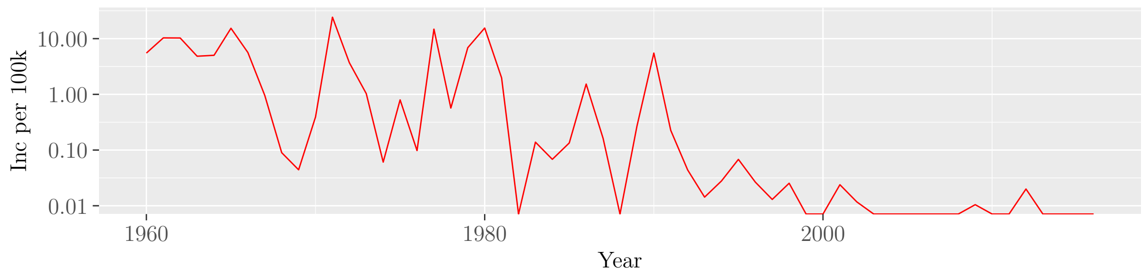

DELAWARE
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
DELAWARE
US

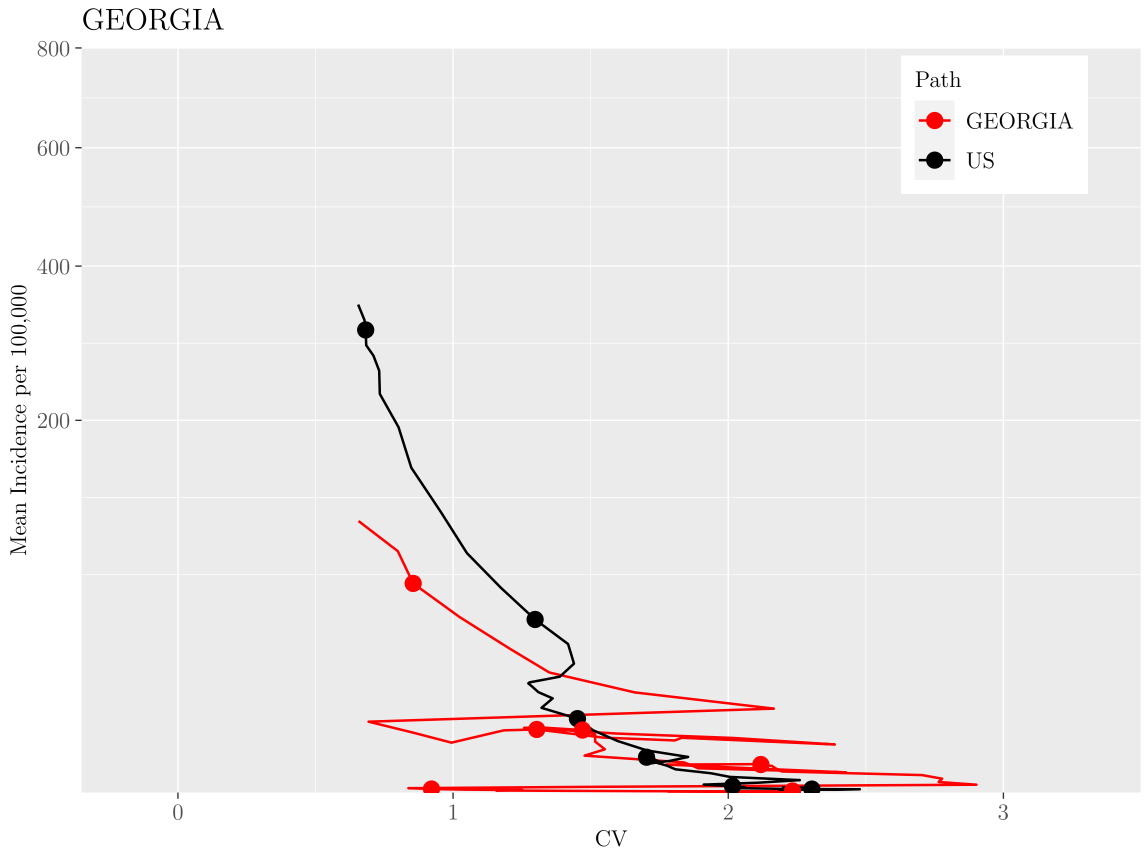

Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

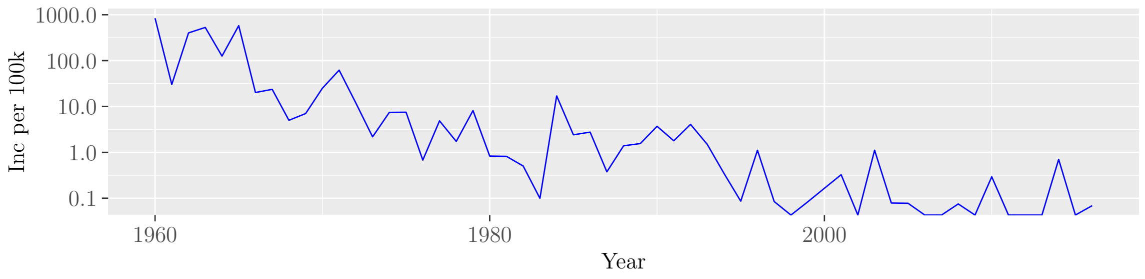

DISTRICT OF COLUMBIA
Path
DISTRICT OF COLUMBIA
US
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV

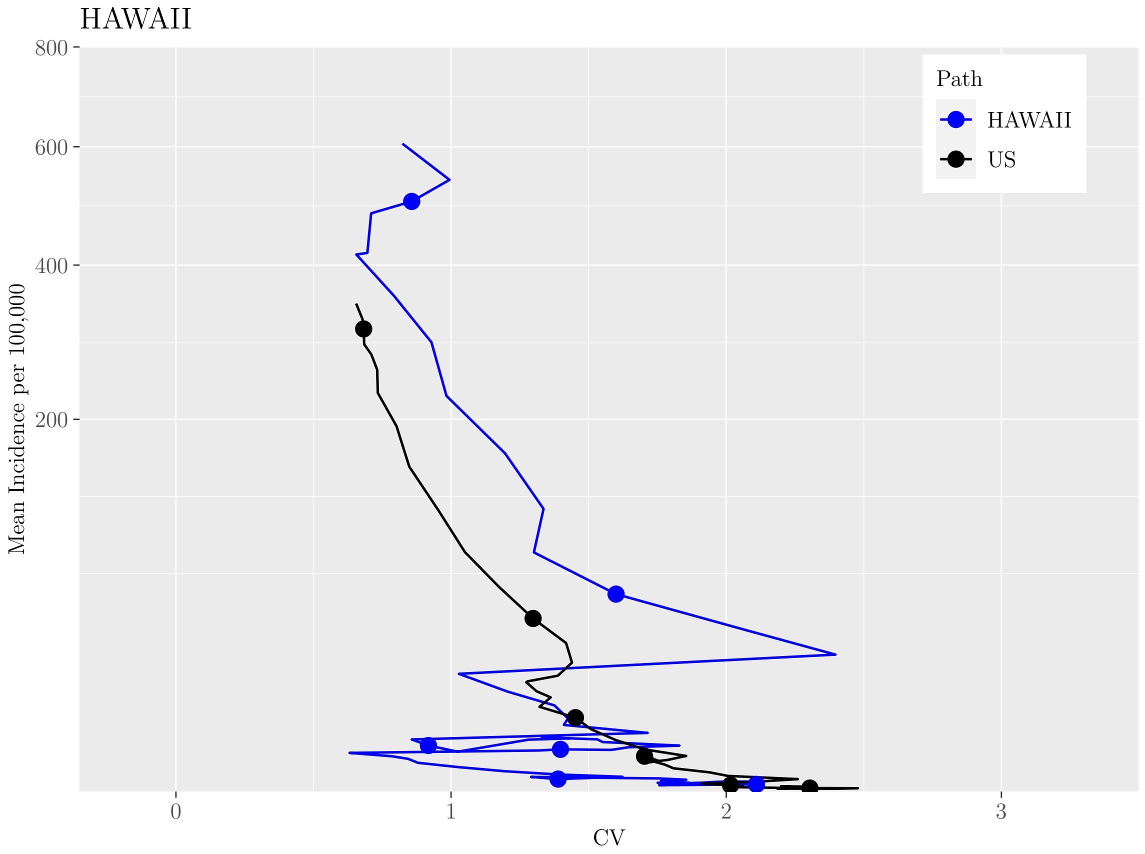

Inc per 100k
100.00
10.00
1.00
0.10
0.01
1960
1980
2000
Year

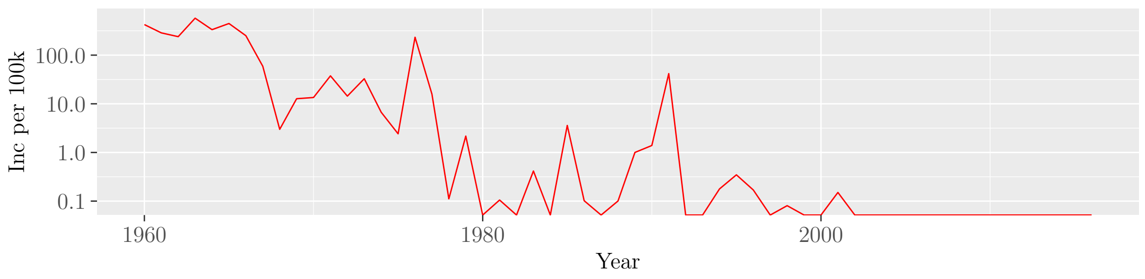

FLORIDA
800
600
400
200
Mean Incidence per 100,000
Path
FLORIDA
US
0
1
2
3
CV

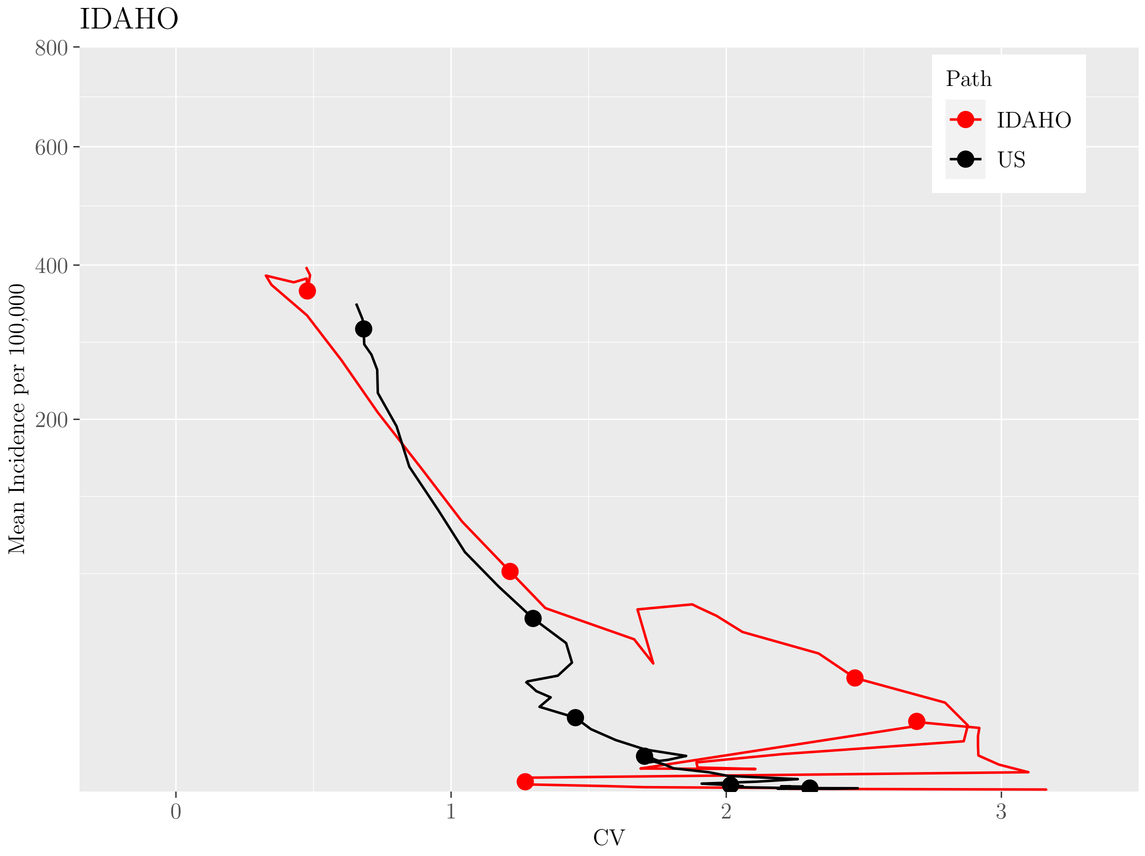

Inc per 100k
10.00
1.00
0.10
0.01
1960
1980
2000
Year

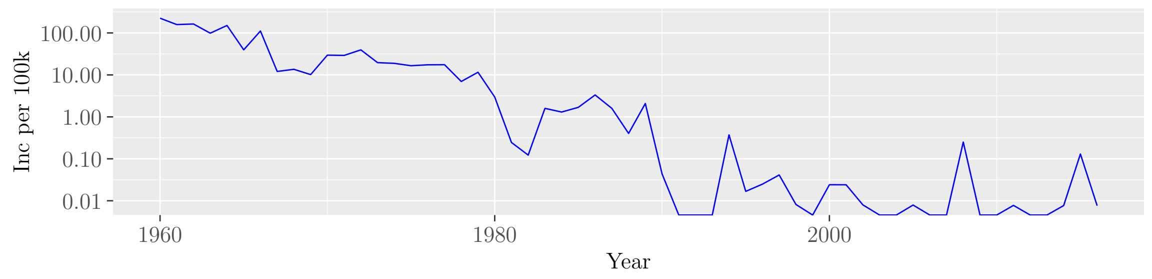

GEORGIA
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
GEORGIA
US


1000.0
100.0
10.0
1.0
0.1
Inc per 100k
1960
1980
2000
Year

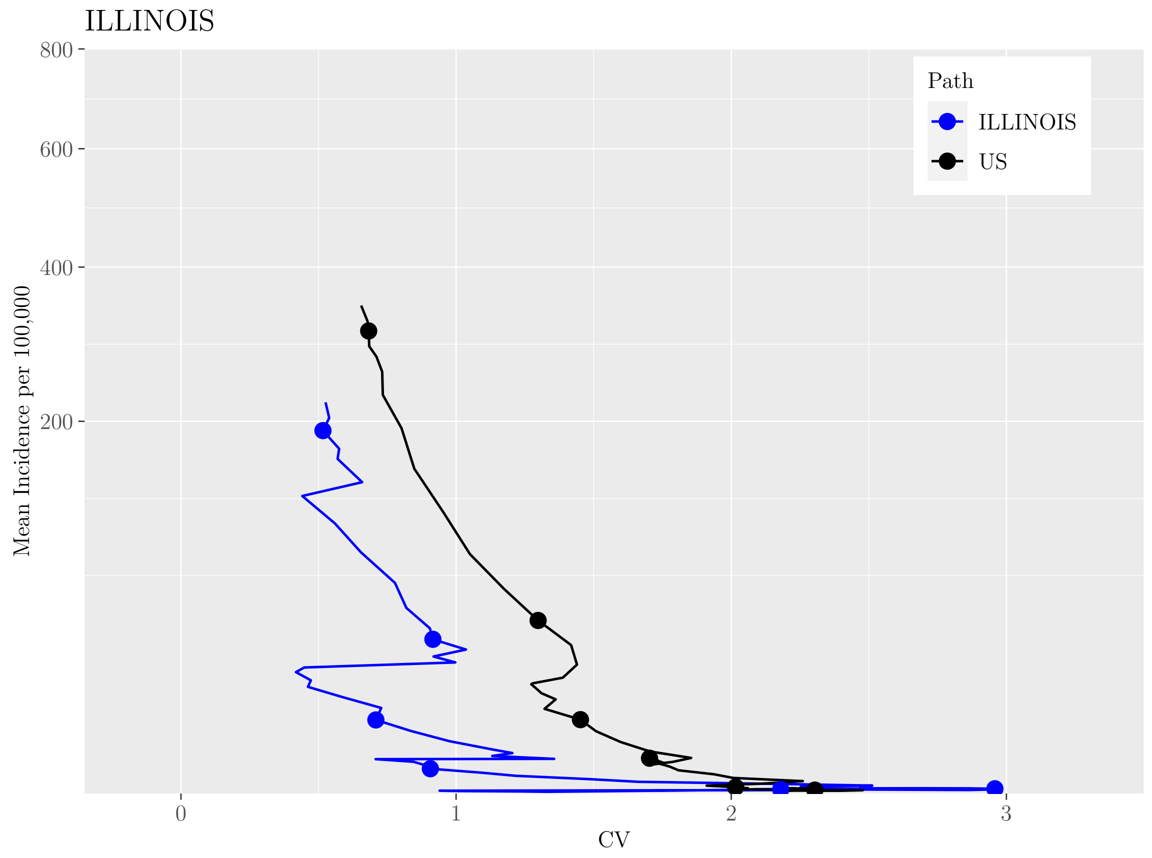

HAWAII
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
HAWAII
US


Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

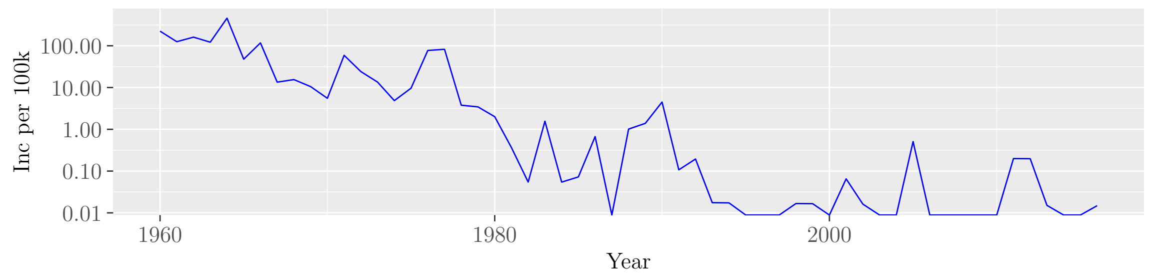

IDAHO
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
IDAHO
US

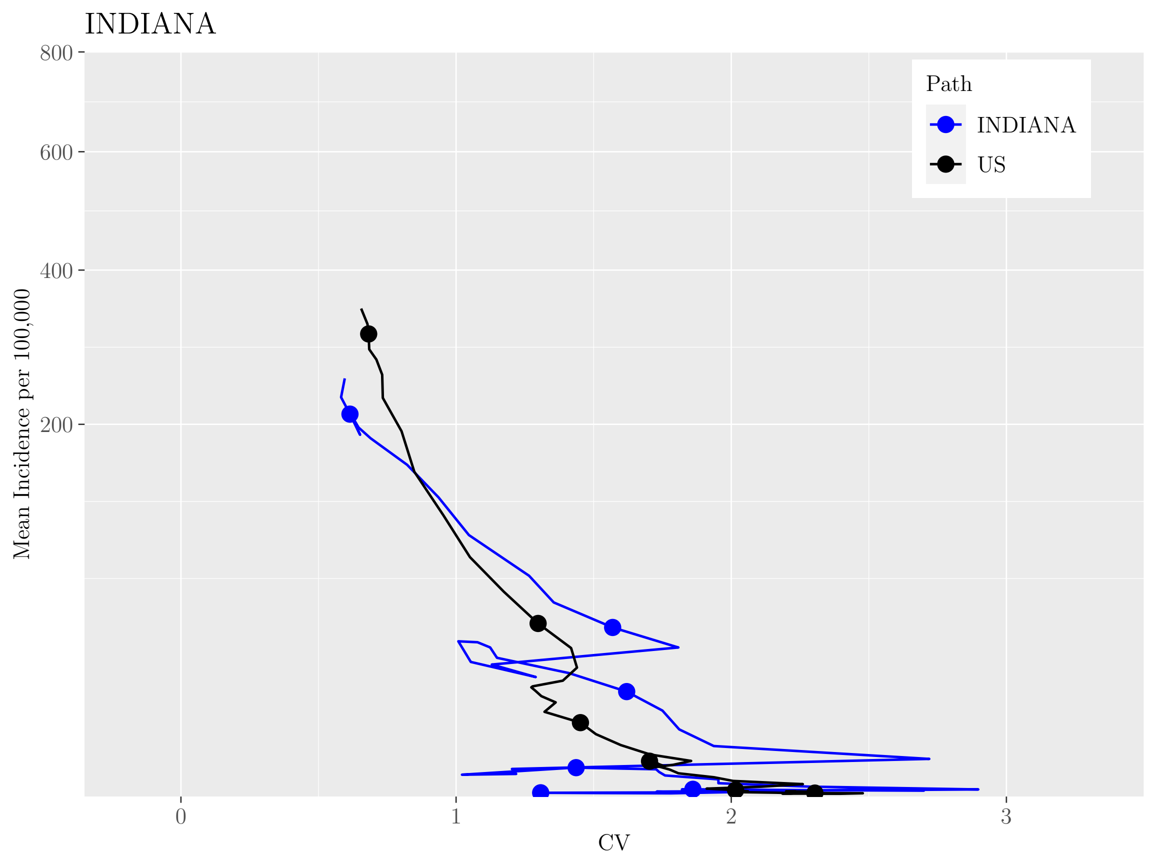


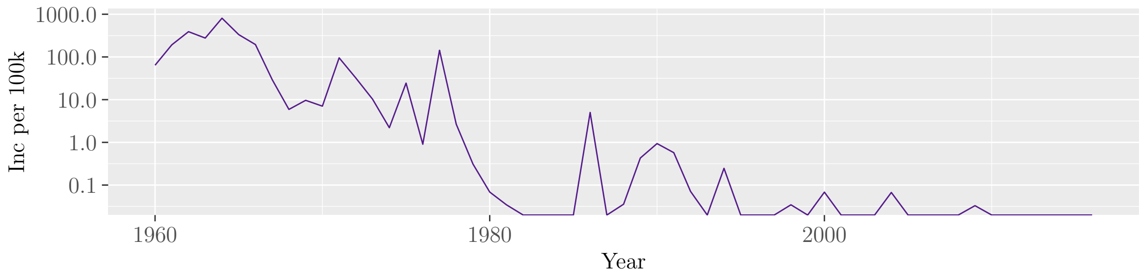
Inc per 100k
100.00
10.00
1.00
0.10
0.01
1960
1980
2000
Year

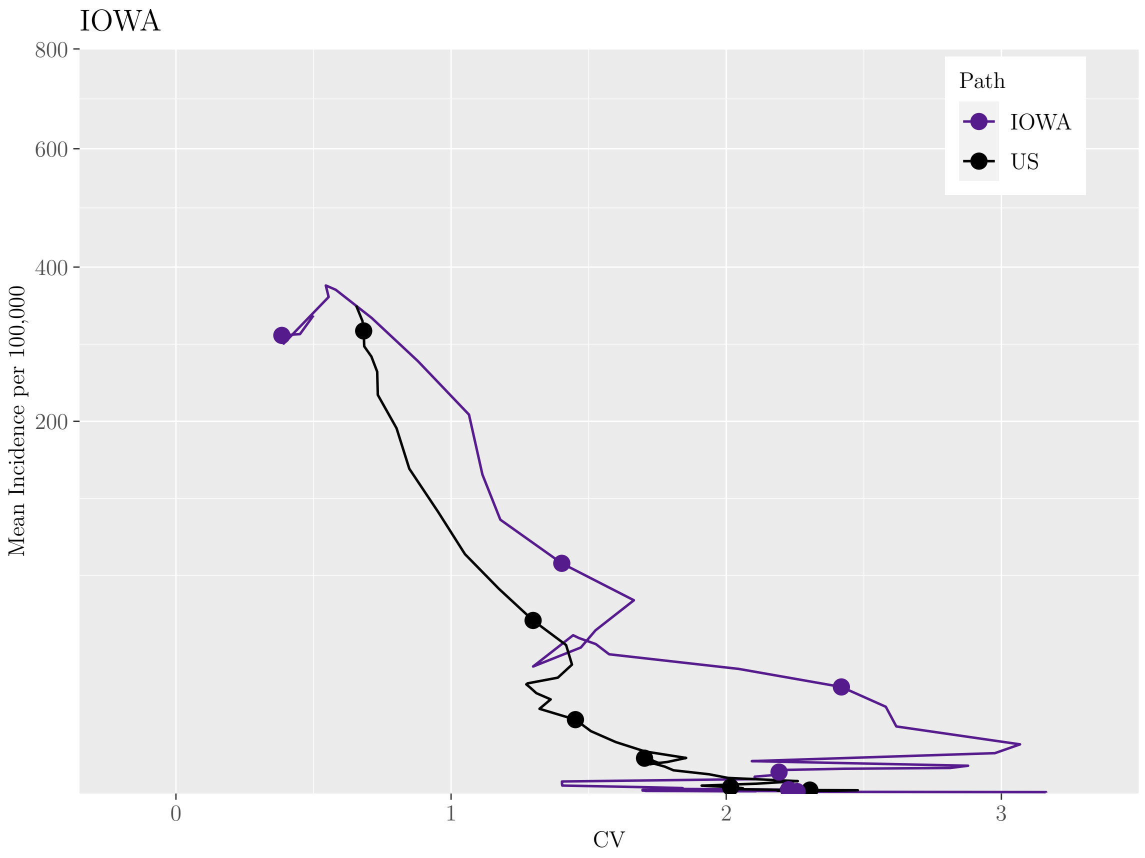


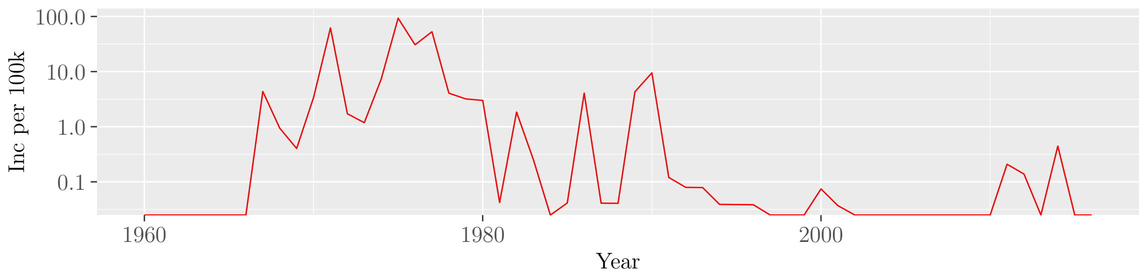
ILLINOIS
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
ILLINOIS
US

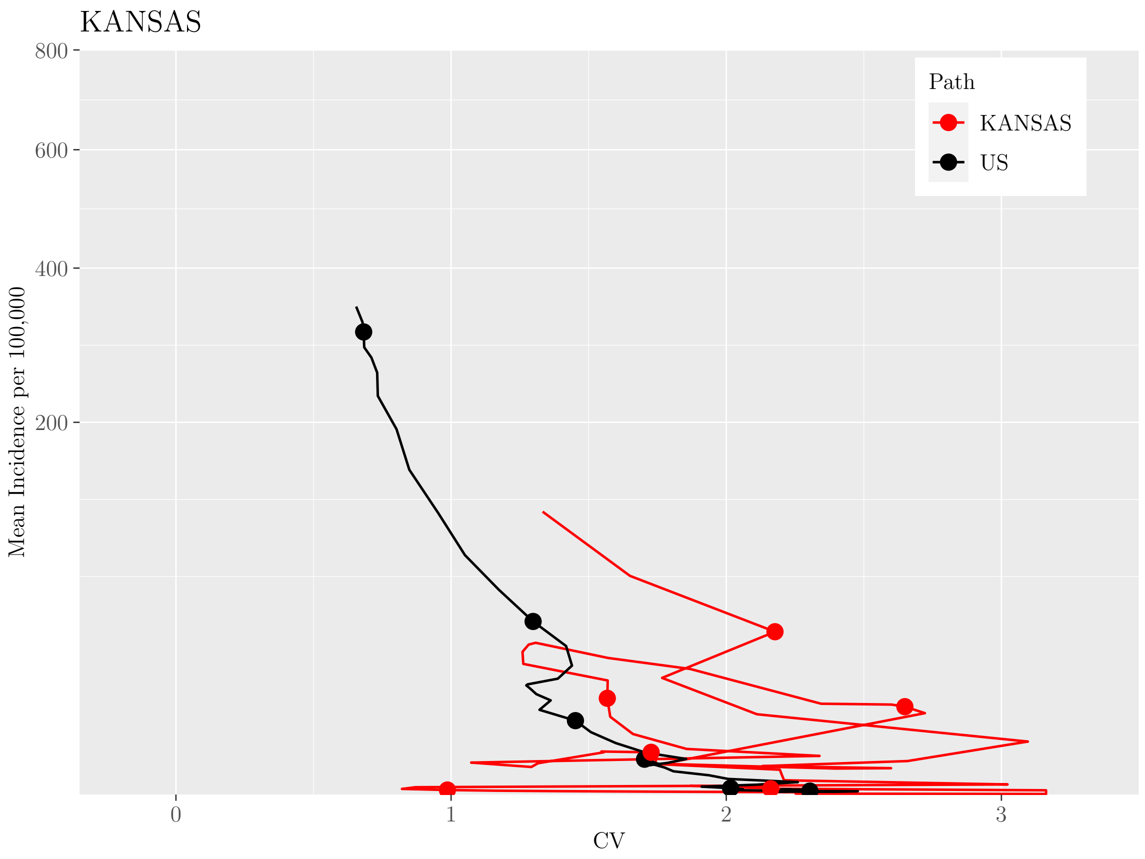

Inc per 100k
100.00
10.00
1.00
0.10
0.01
1960
1980
2000
Year

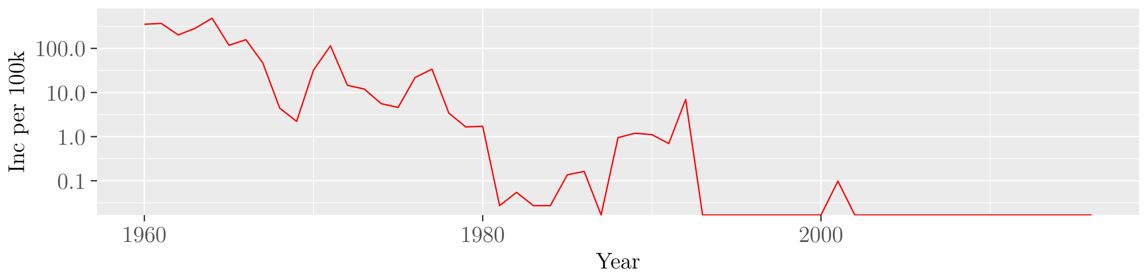

INDIANA
Mean Incidence per 100,000
800
600
400
200
CV
0
1
2
3
Path
INDIANA
US

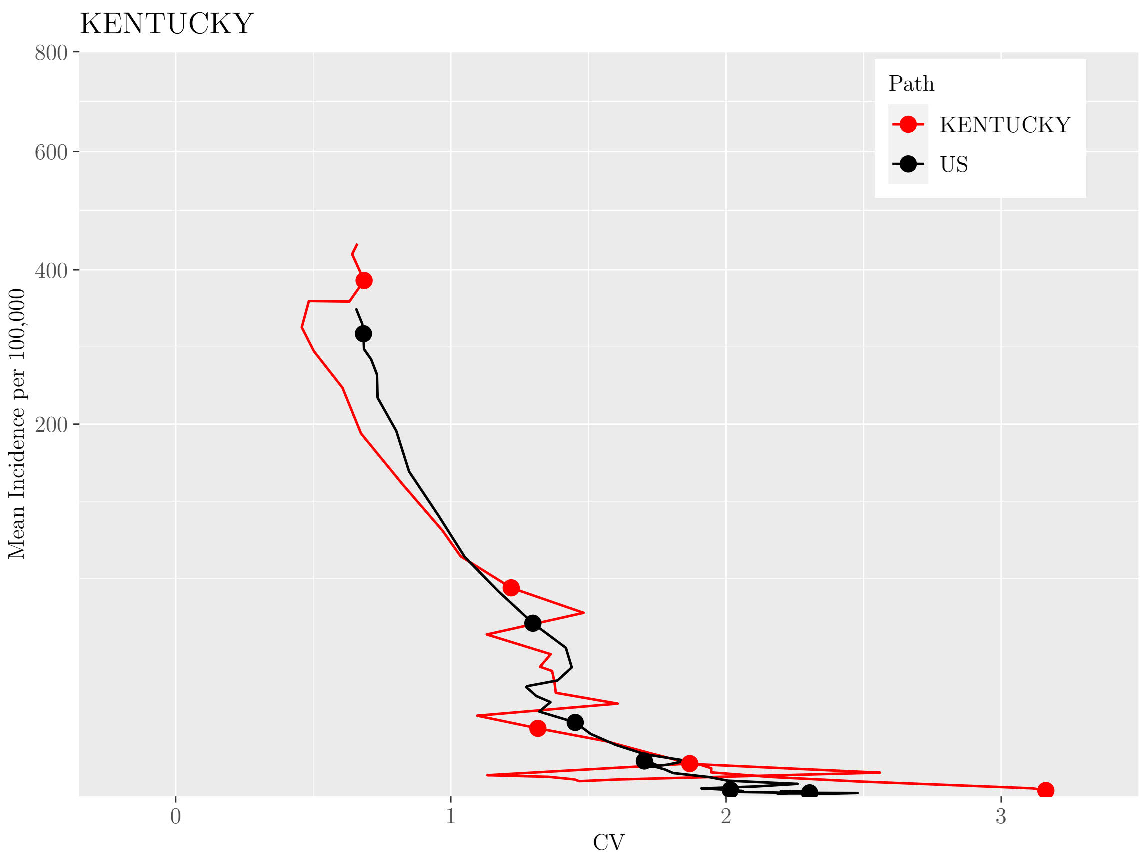

Inc per 100k
1000.0
100.0
10.0
1.0
0.1
1960
1980
2000
Year

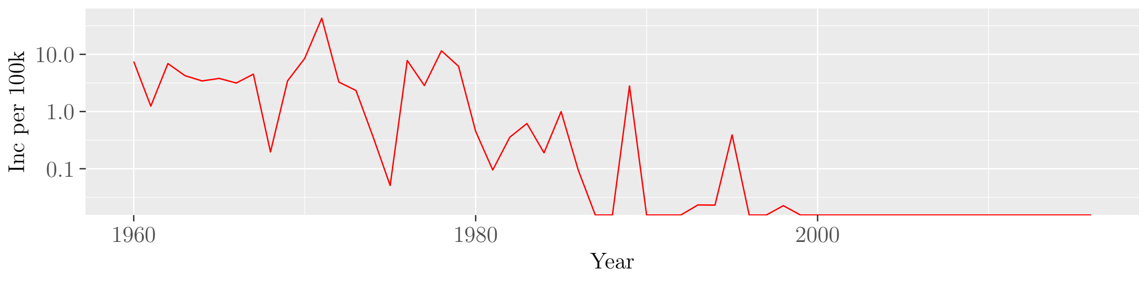

IOWA
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
IOWA
US


Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

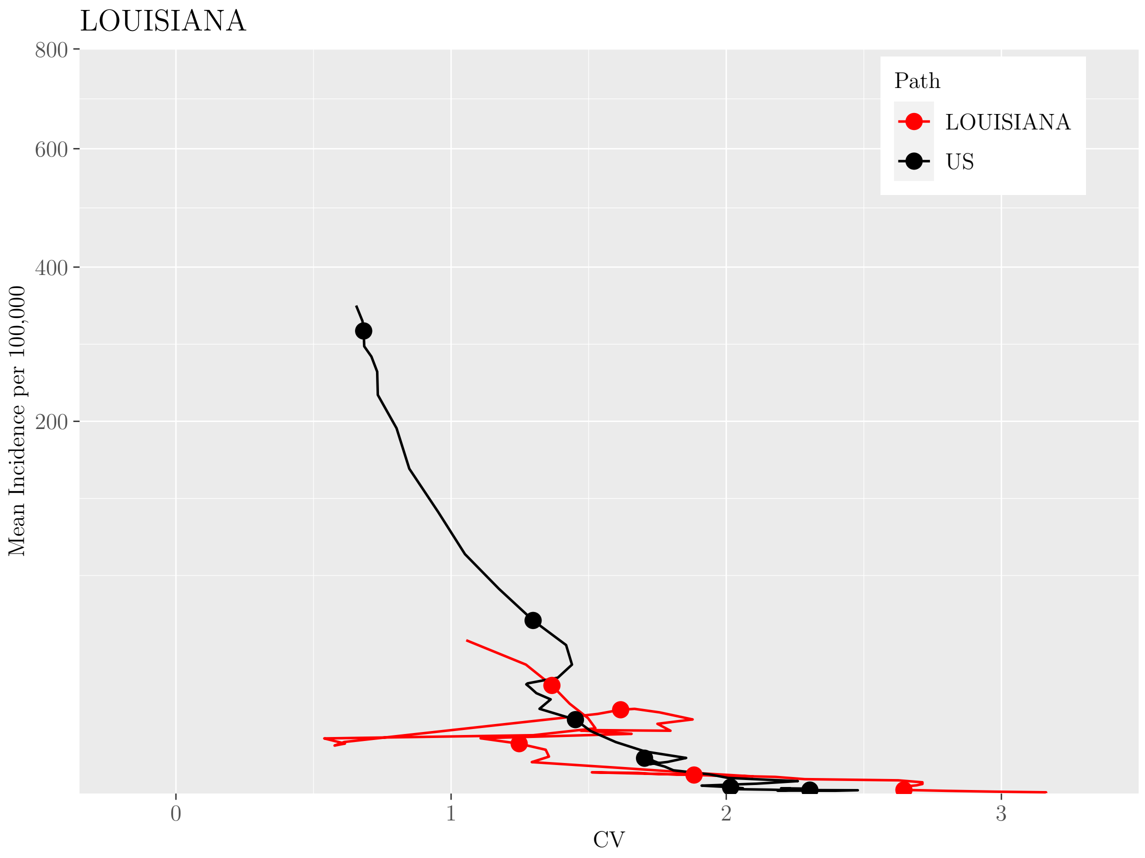

KANSAS
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
KANSAS
US


Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

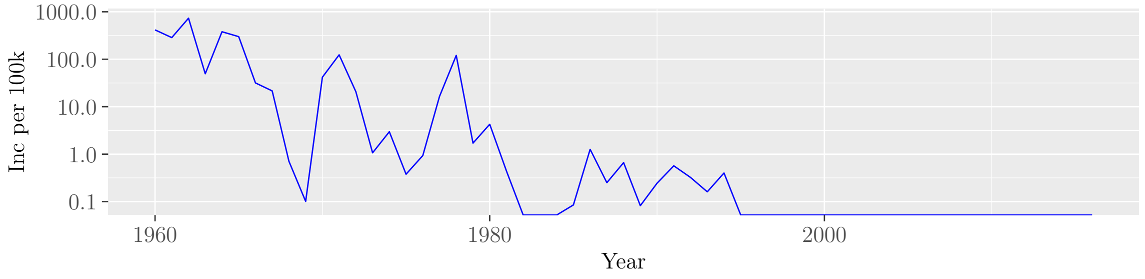

KENTUCKY
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
KENTUCKY
US

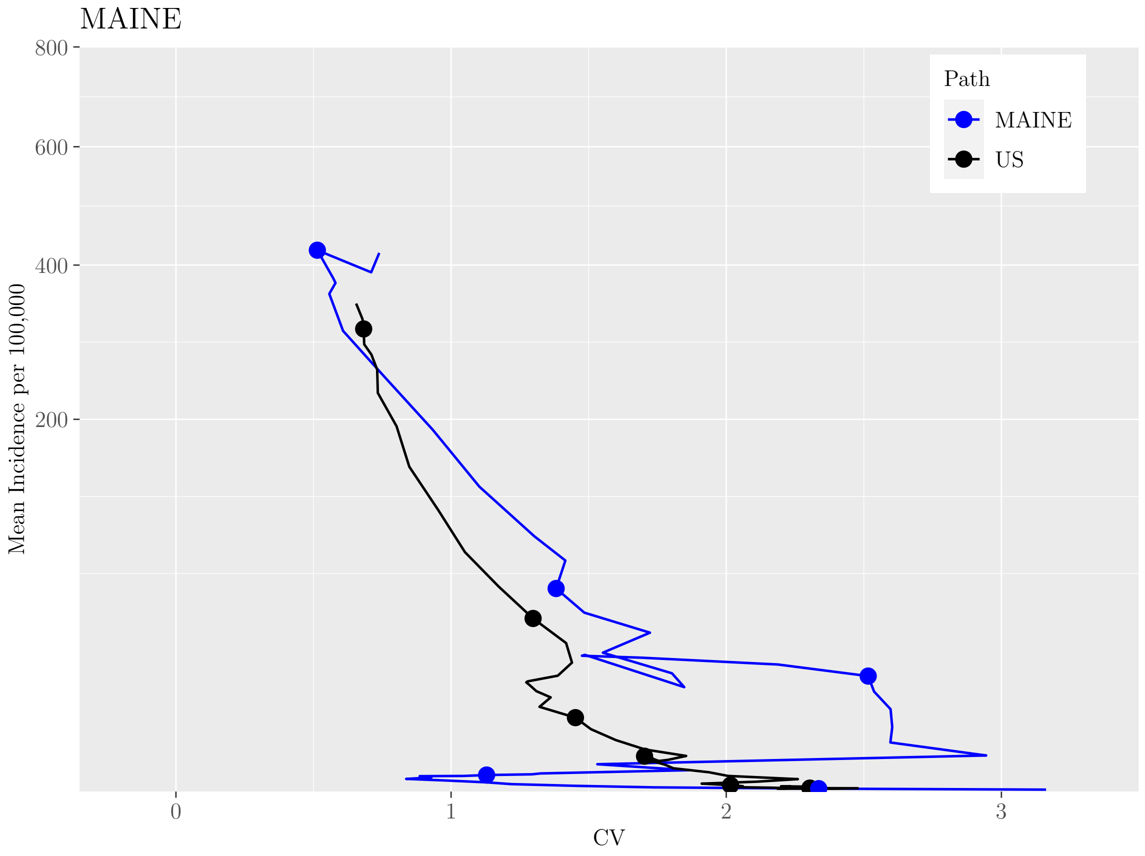


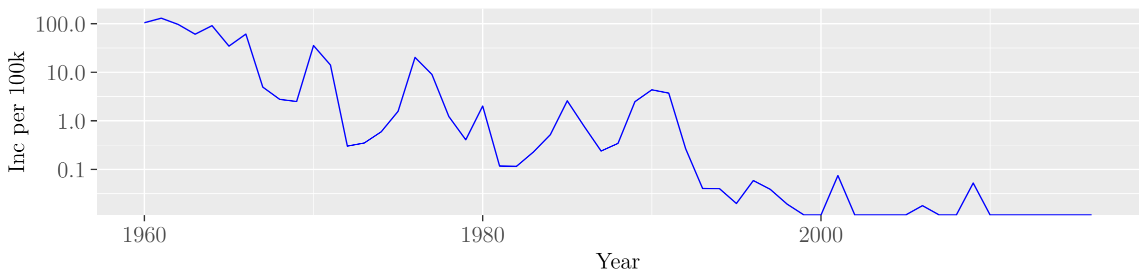
Inc per 100k
10.0
1.0
0.1
1960
1980
2000
Year

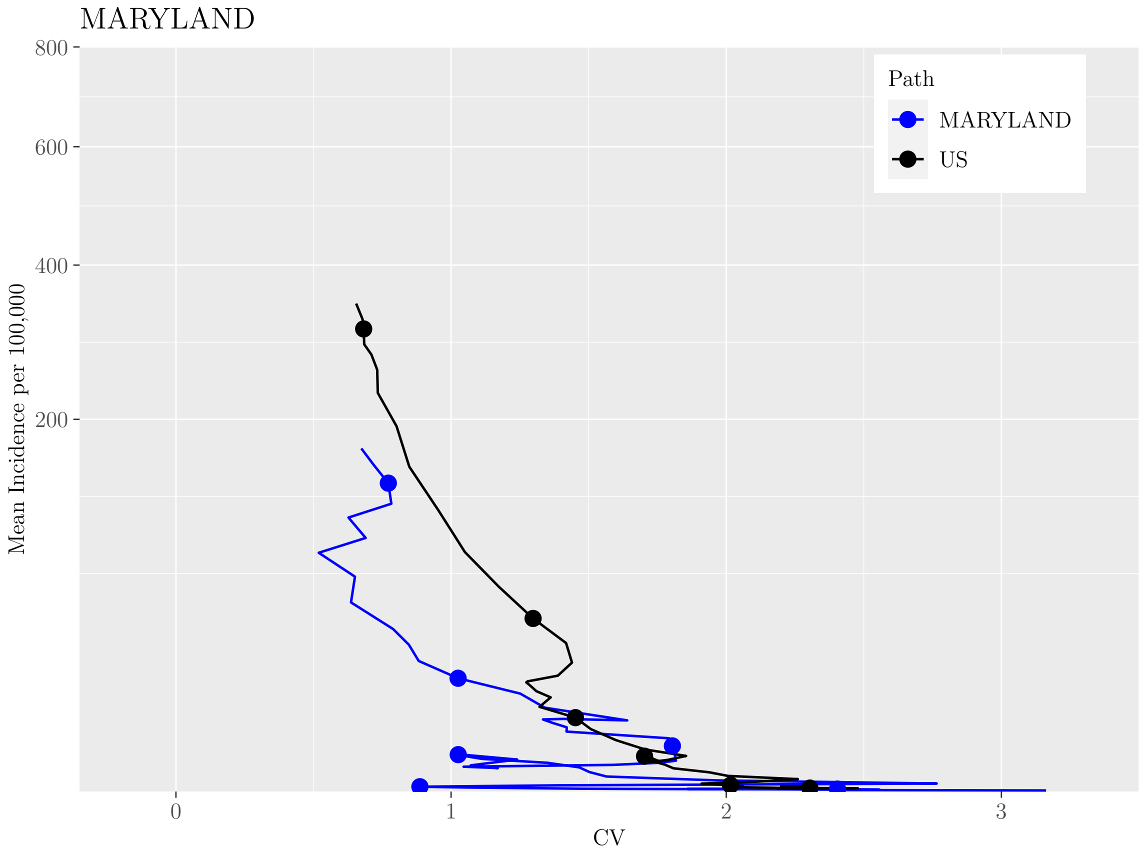


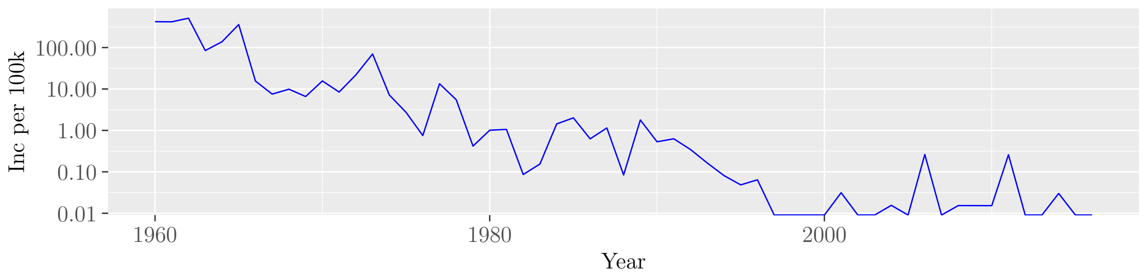
LOUISIANA
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
LOUISIANA
US

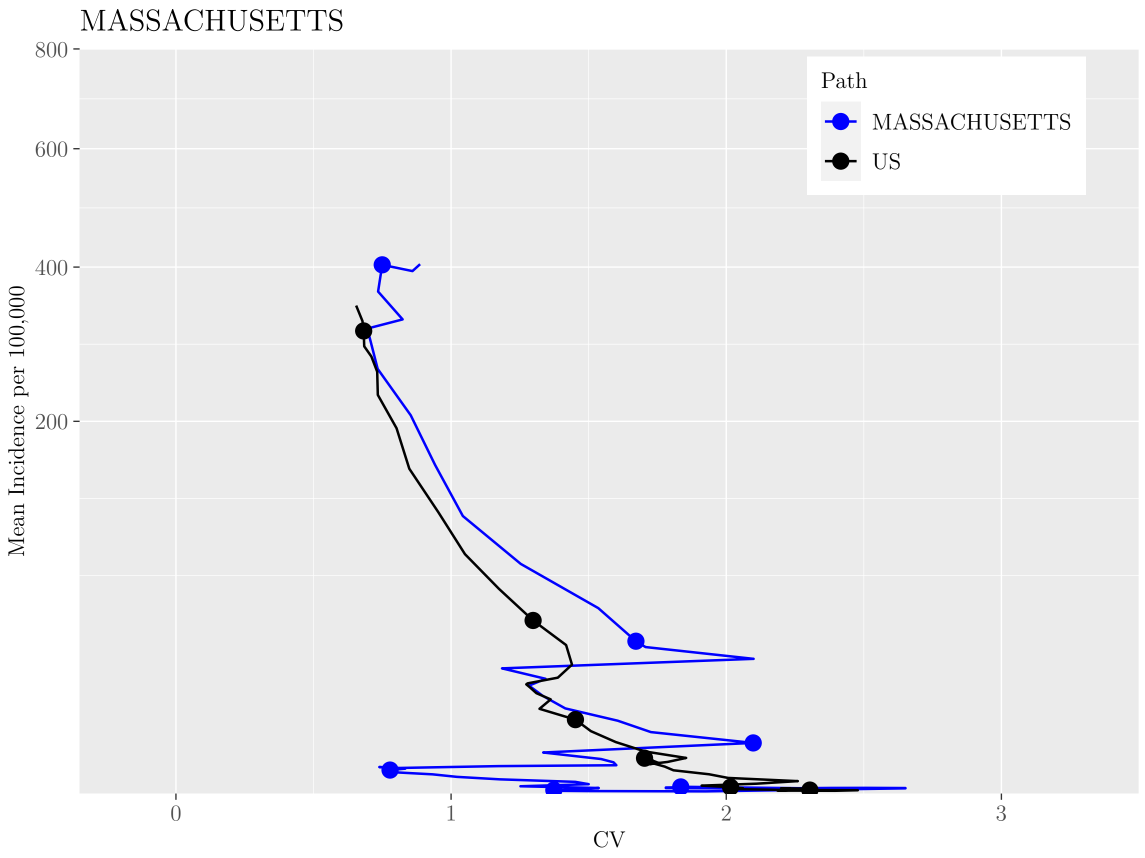

Inc per 100k
1000.0
100.0
10.0
1.0
0.1
1960
1980
2000
Year

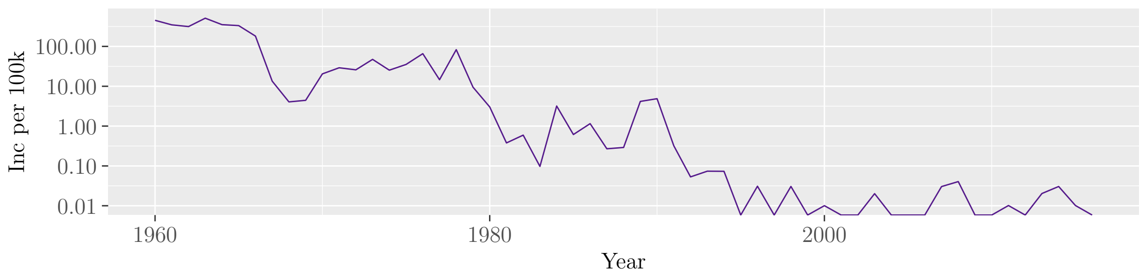

MAINE
Path
MAINE
US
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV

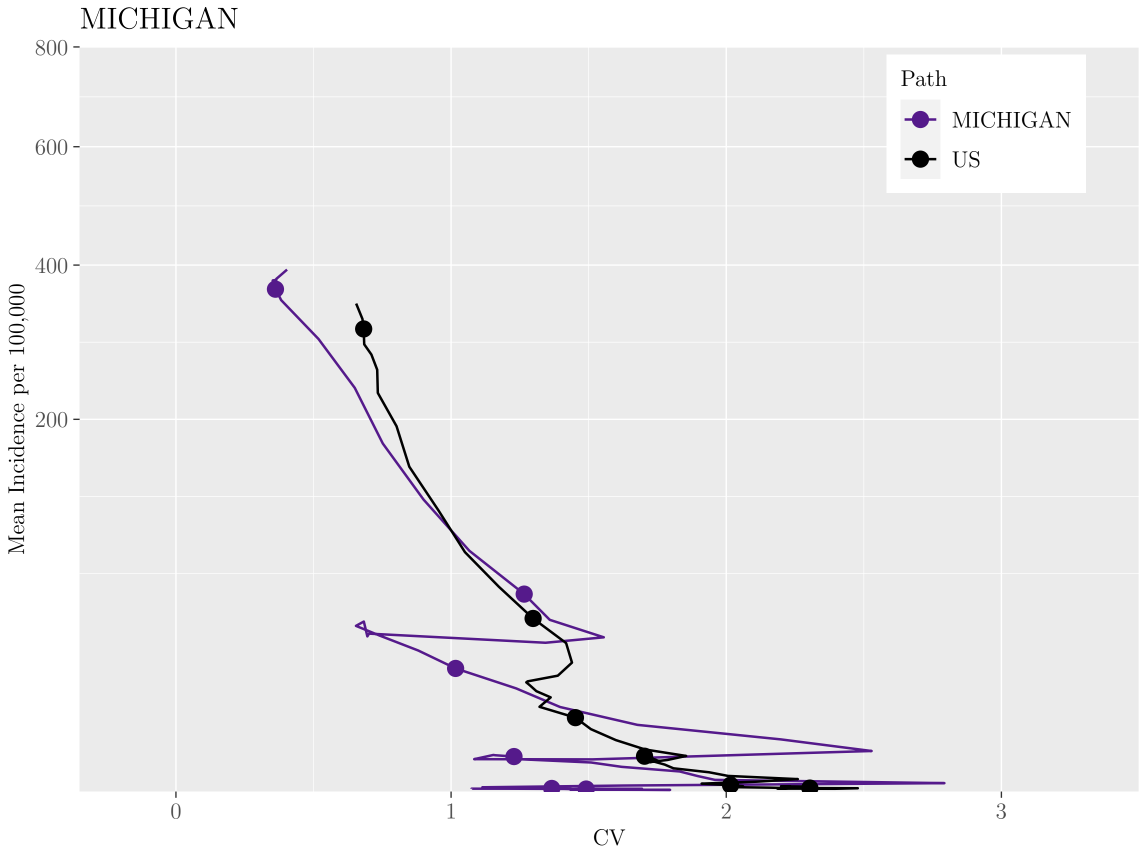

100.0
10.0
1.0
0.1
Inc per 100k
1960
1980
2000
Year

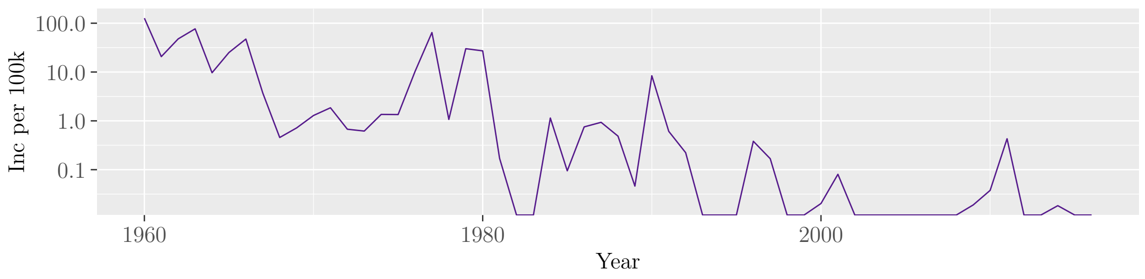

MARYLAND
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
MARYLAND
US

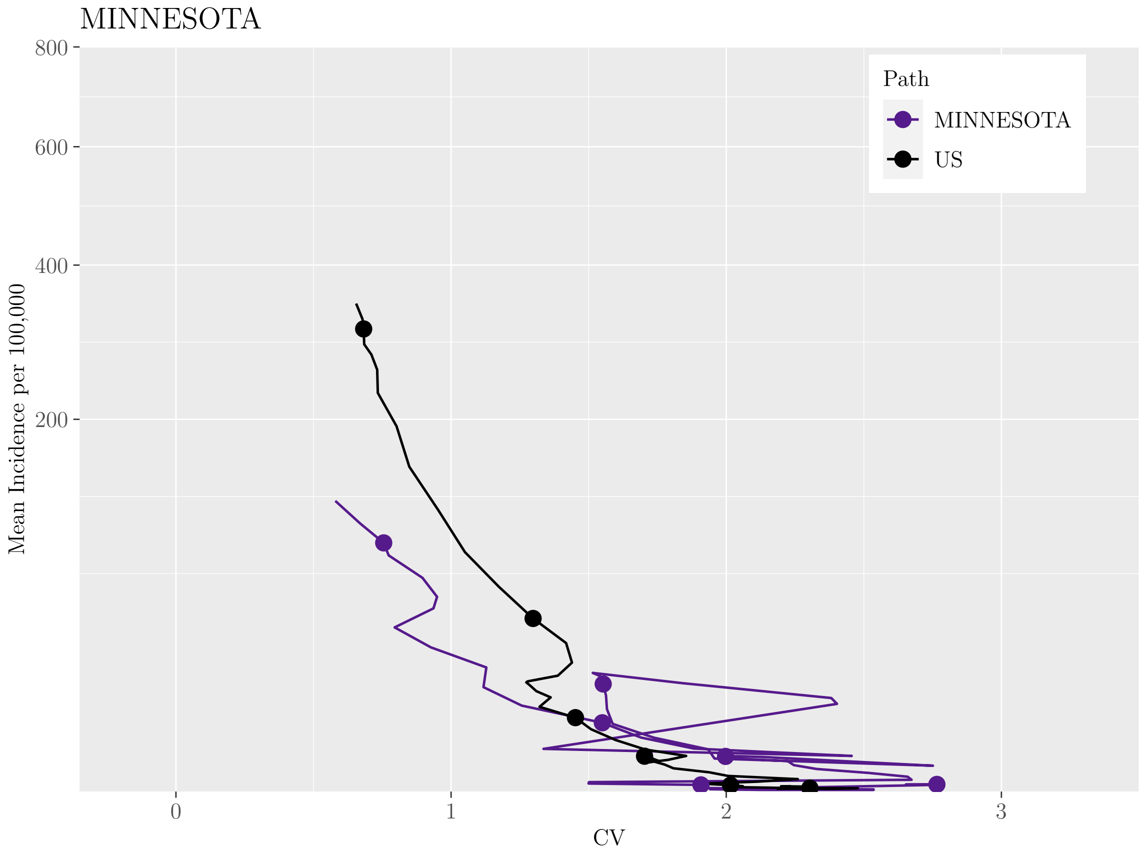

Inc per 100k
100.00
10.00
1.00
0.10
0.01
1960
1980
2000
Year

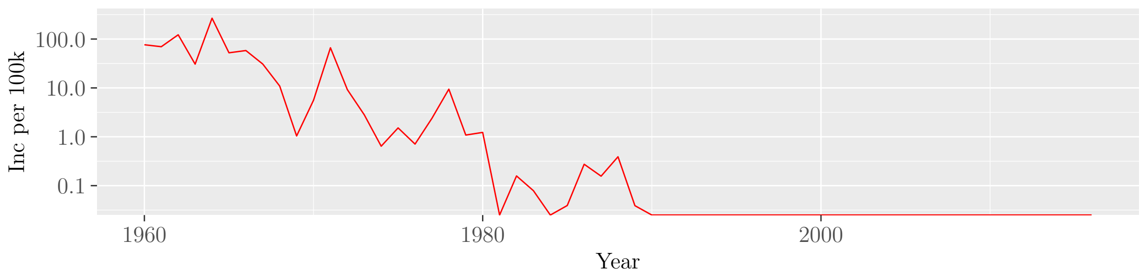

MASSACHUSETTS
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
MASSACHUSETTS
US

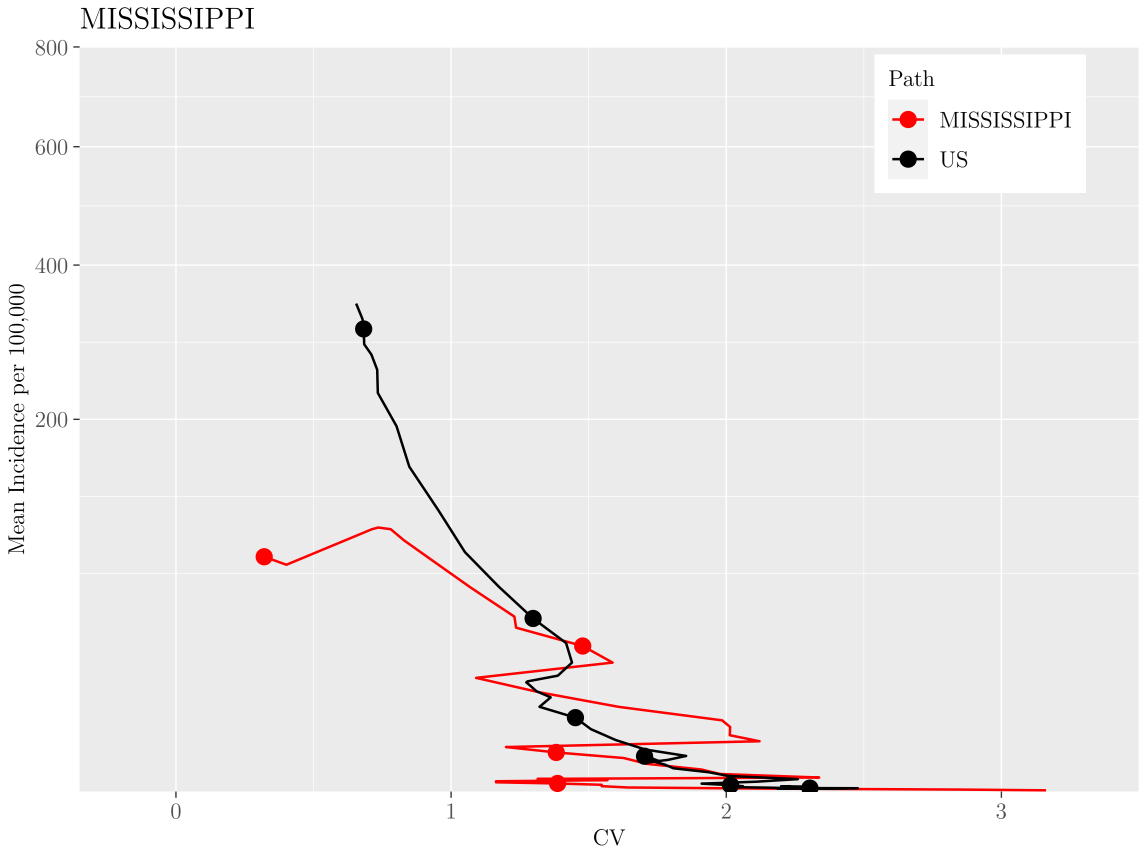

Inc per 100k
100.00
10.00
1.00
0.10
0.01
1960
1980
2000
Year

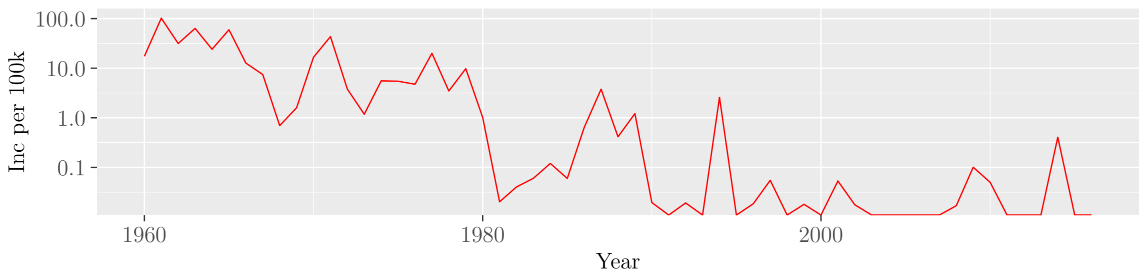

MICHIGAN
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
MICHIGAN
US

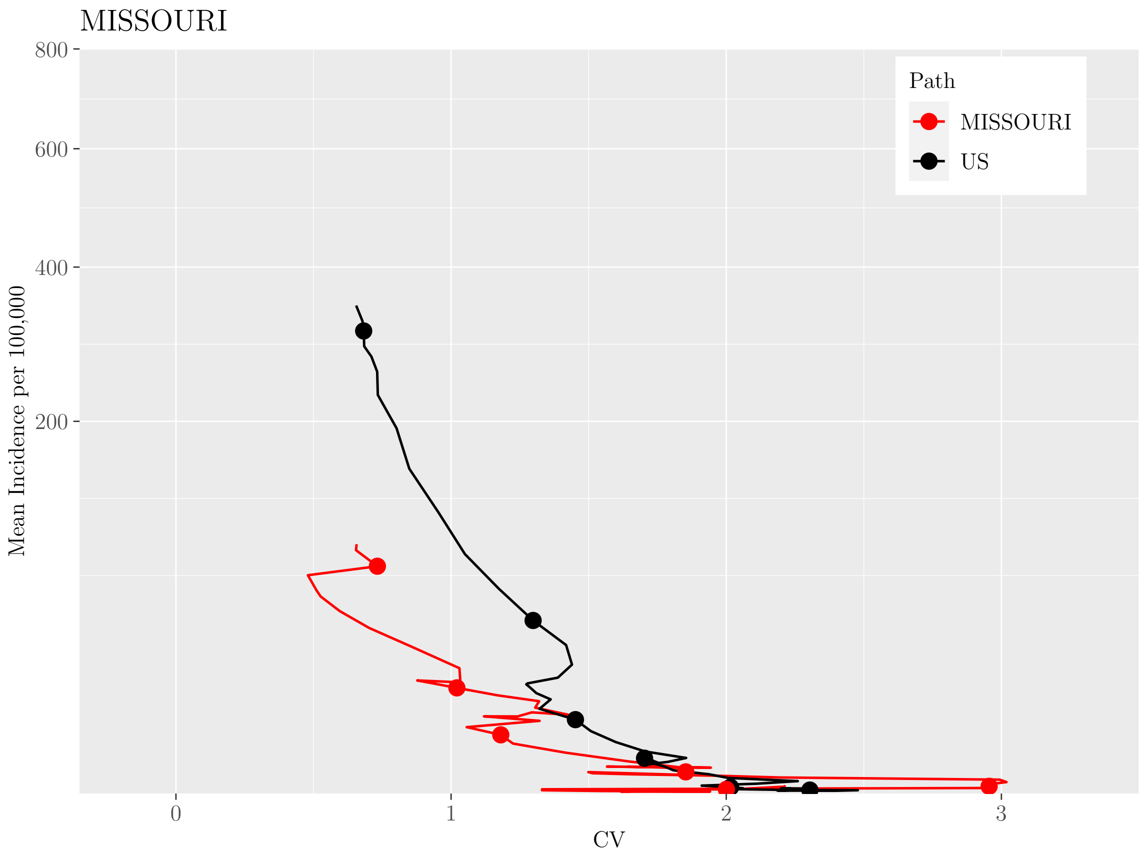

Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

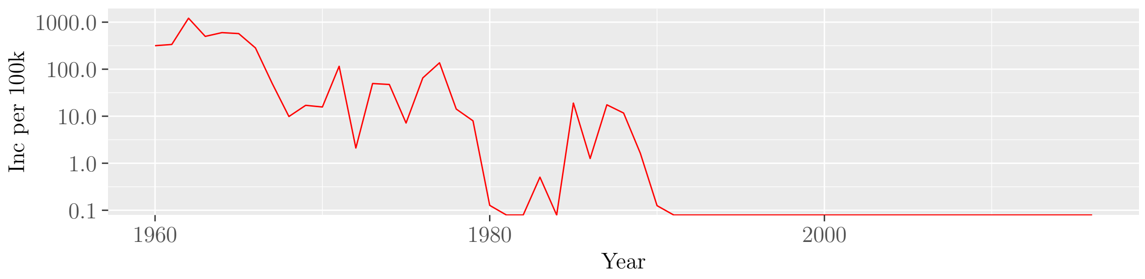

MINNESOTA
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
MINNESOTA
US

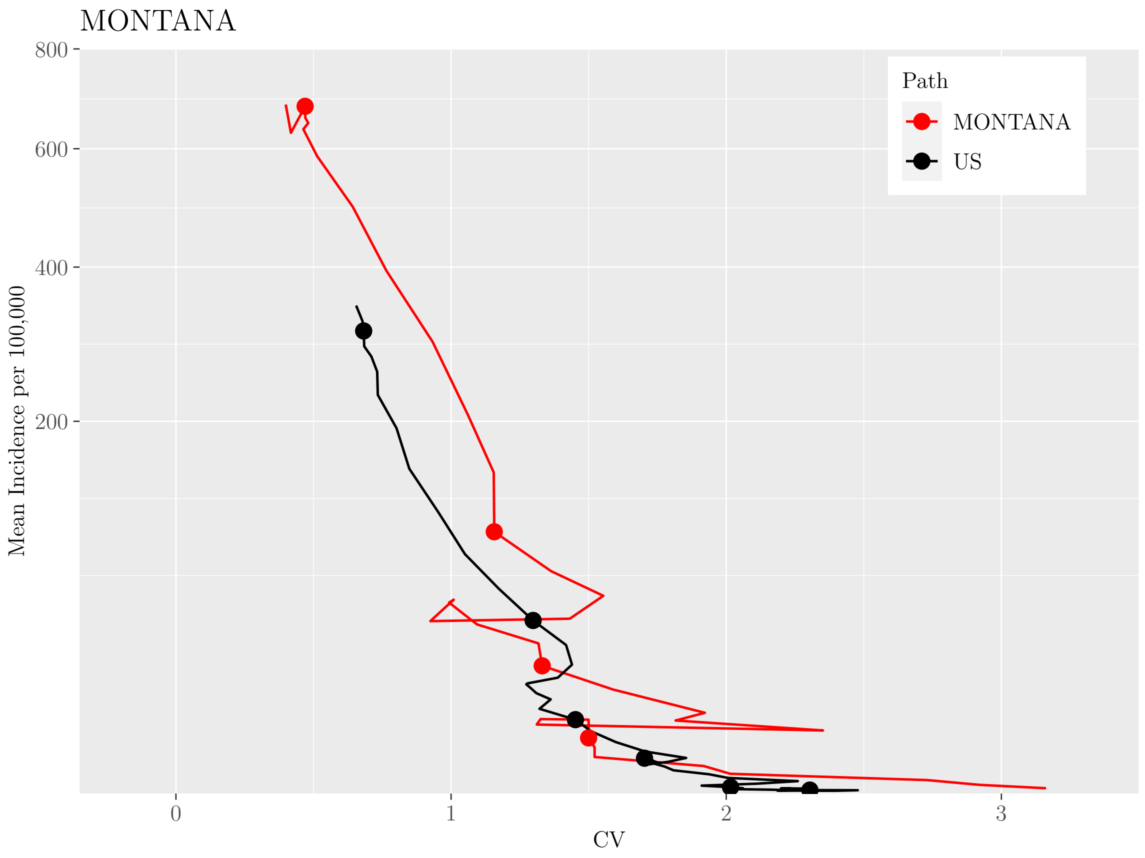

Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

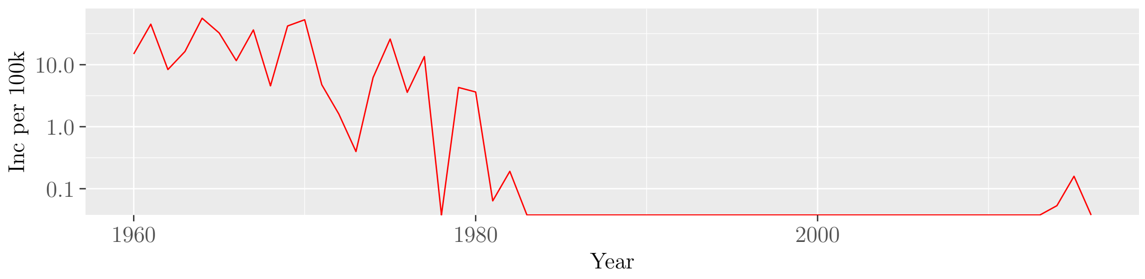

MISSISSIPPI
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
MISSISSIPPI
US

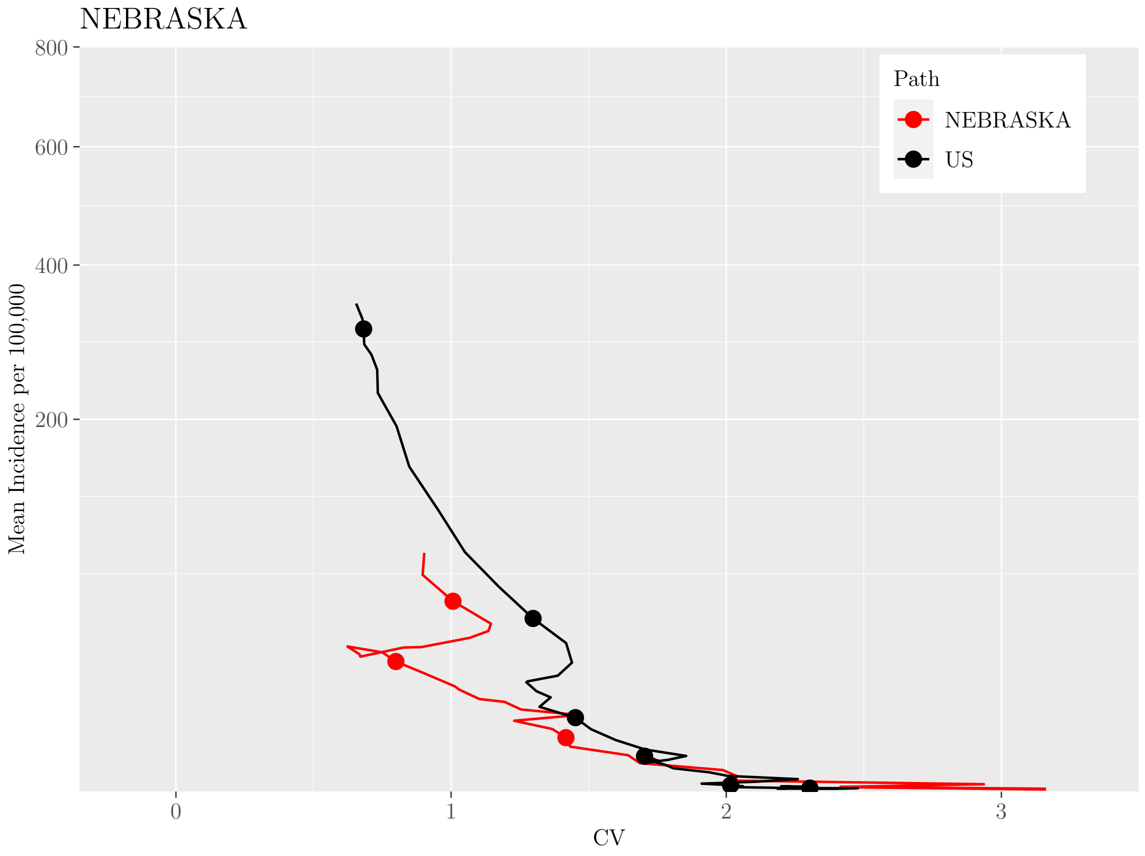

Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

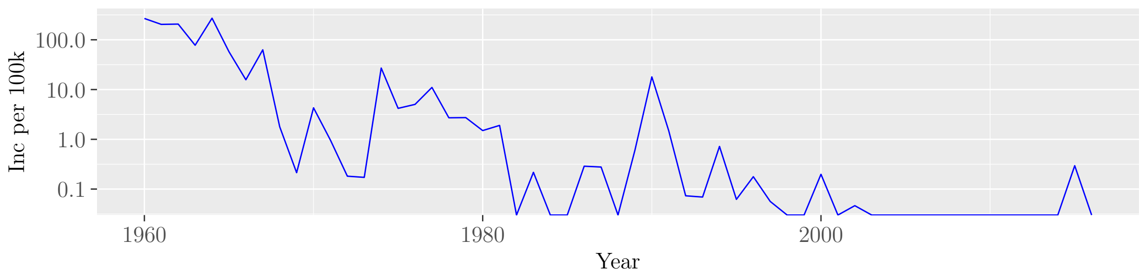

MISSOURI
Mean Incidence per 100,000
800
600
400
200
CV
0
1
2
3
Path
MISSOURI
US

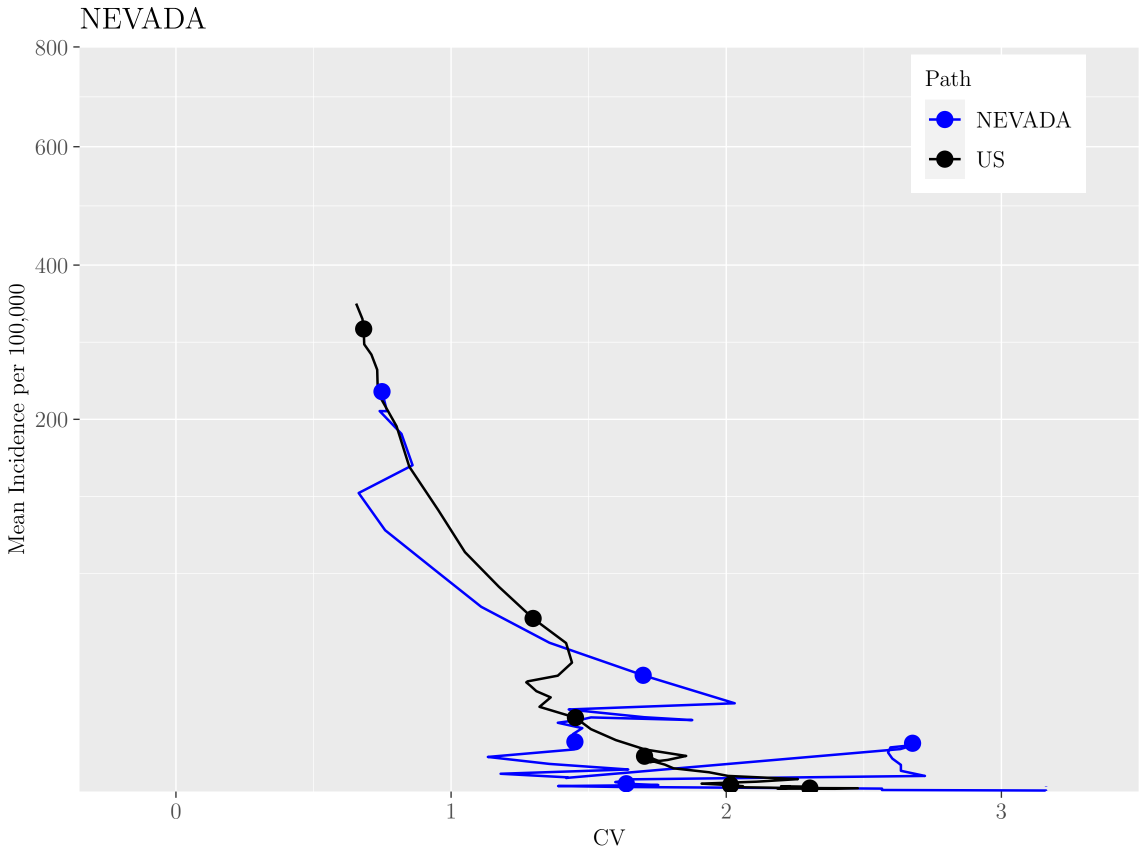

Inc per 100k
1000.0
100.0
10.0
1.0
0.1
1960
1980
2000
Year

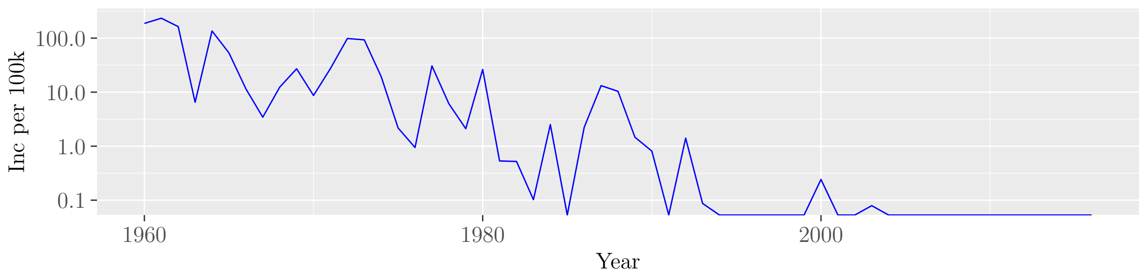

MONTANA
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
MONTANA
US

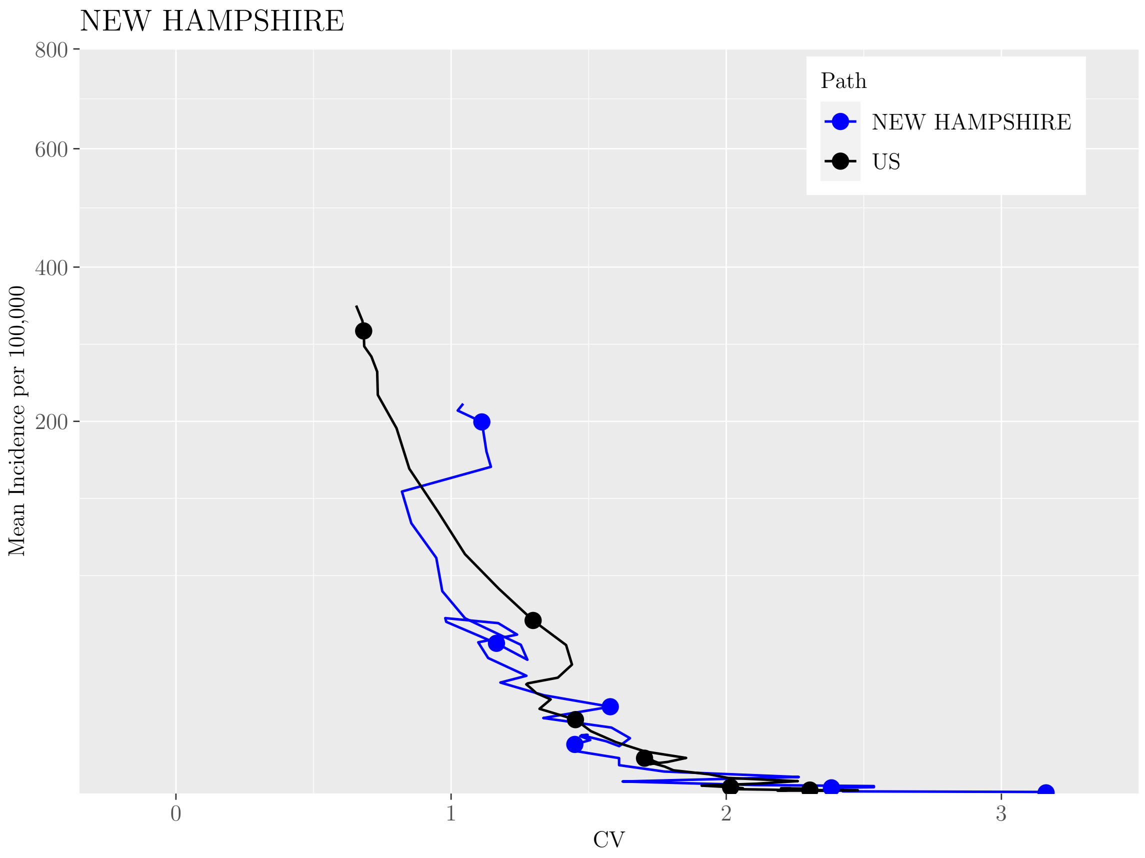

Inc per 100k
10.0
1.0
0.1
1960
1980
2000
Year

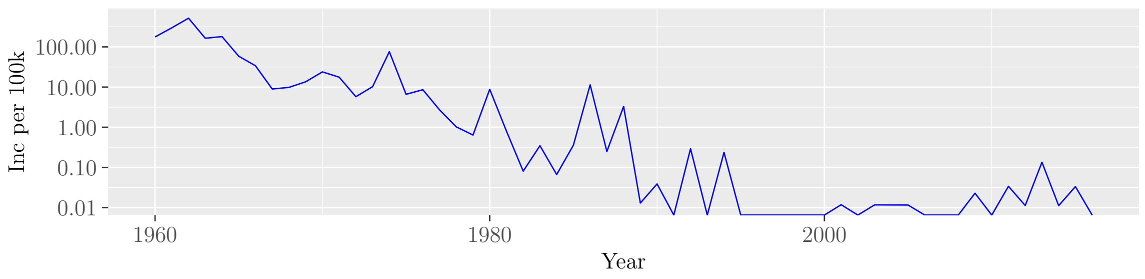

NEBRASKA
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
NEBRASKA
US

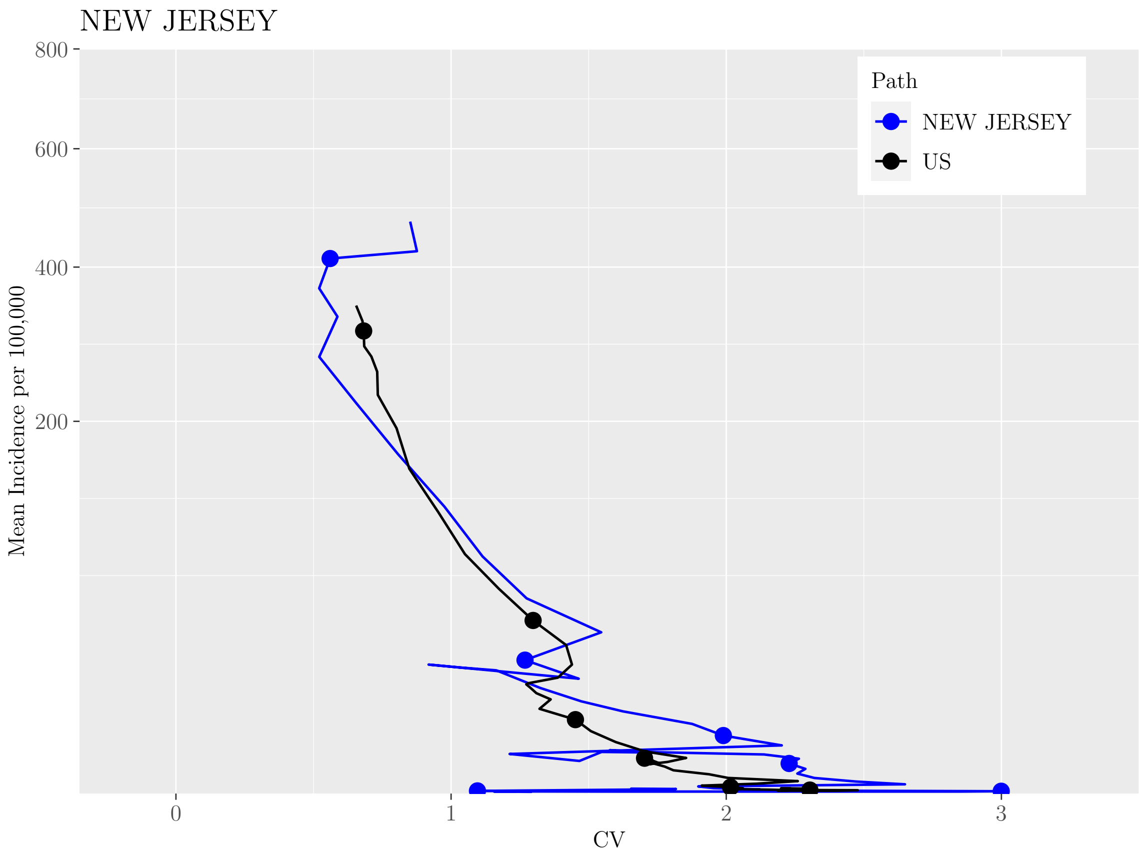

Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

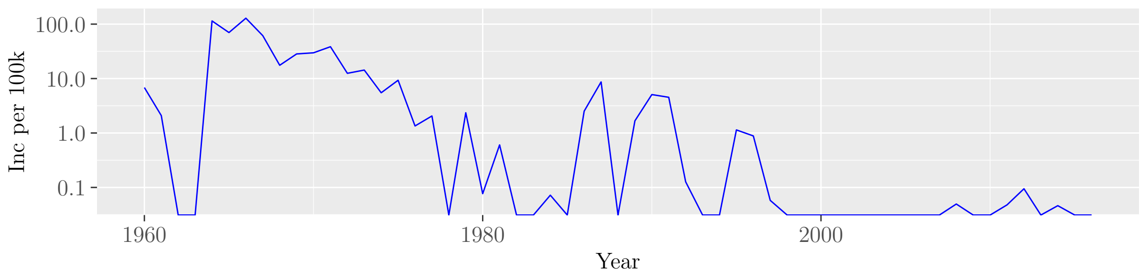

NEVADA
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
NEVADA
US

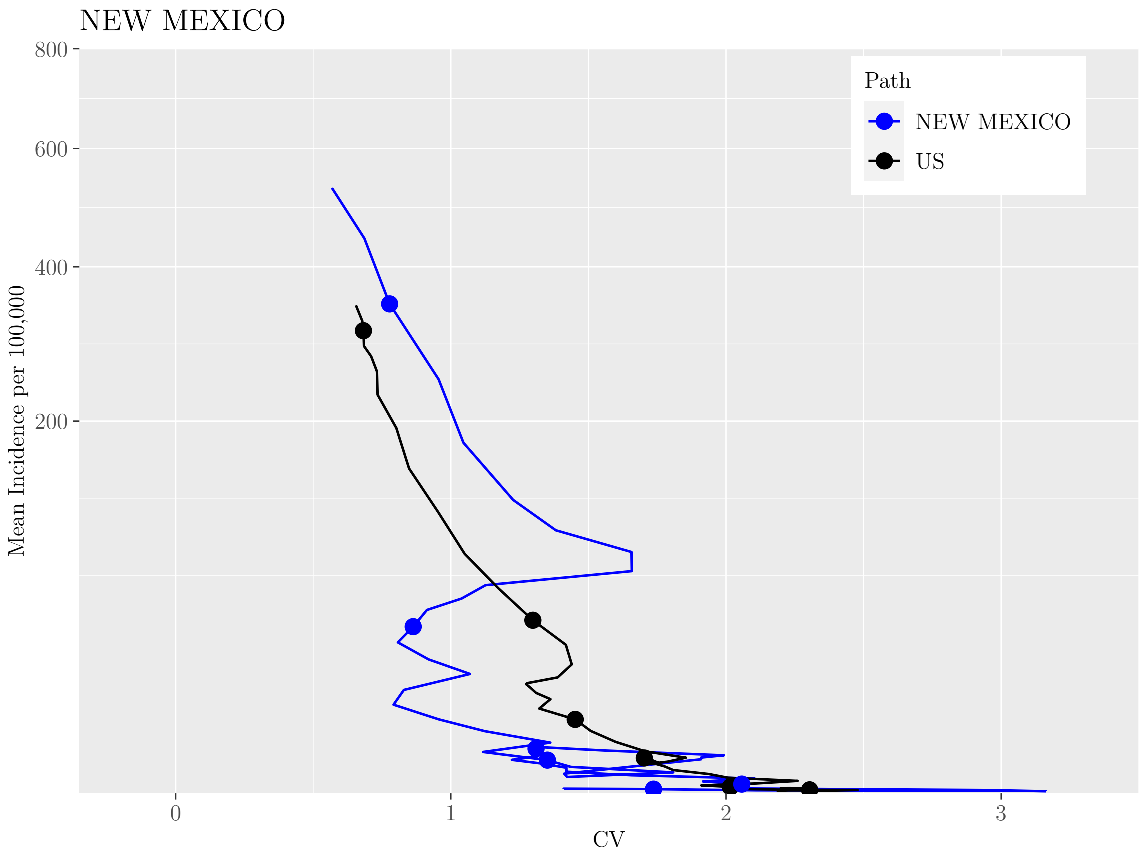

Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

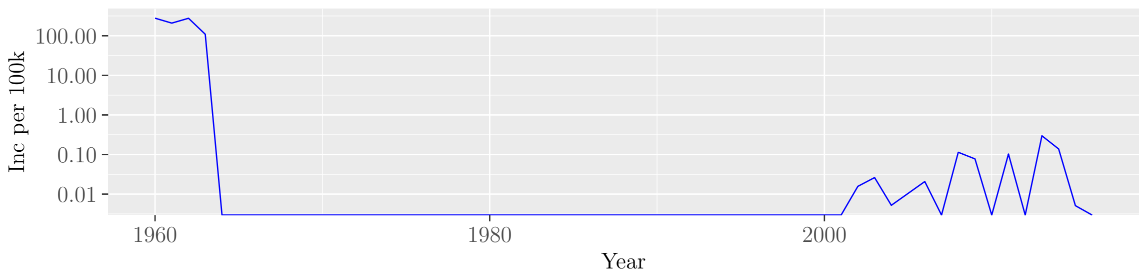

NEW HAMPSHIRE
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
NEW HAMPSHIRE
US

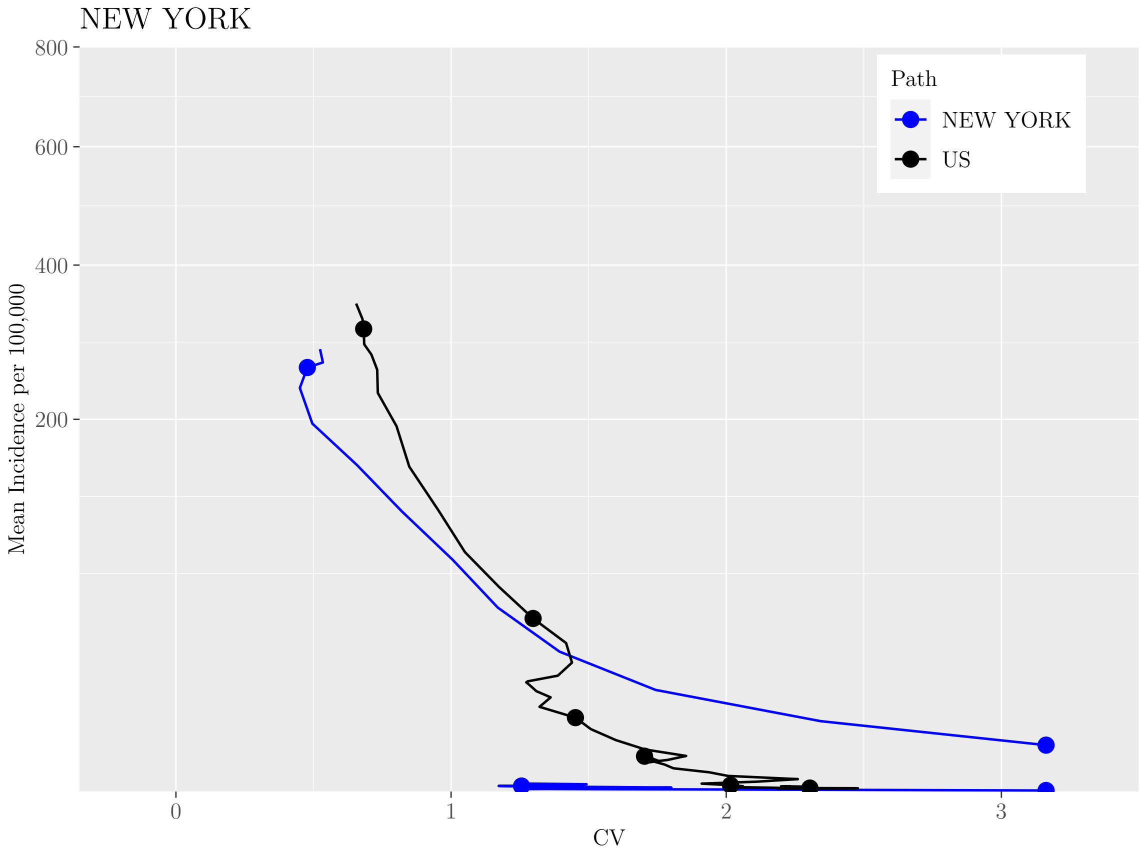

Inc per 100k
100.00
10.00
1.00
0.10
0.01
1960
1980
2000
Year

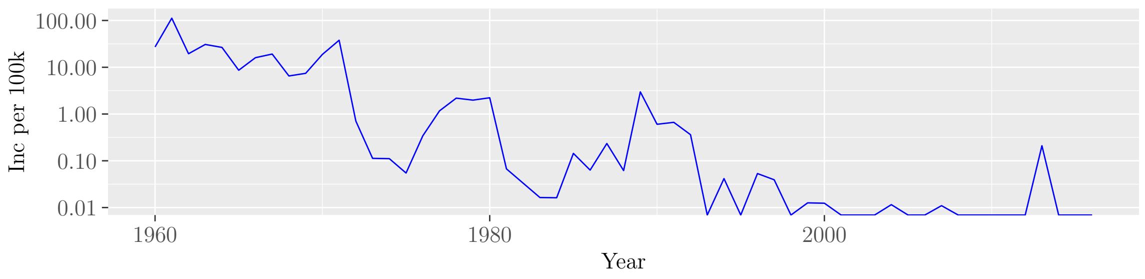

NEW JERSEY
800
600
400
200
Mean Incidence per 100,000
Path
NEW JERSEY
US
0
1
2
3
CV

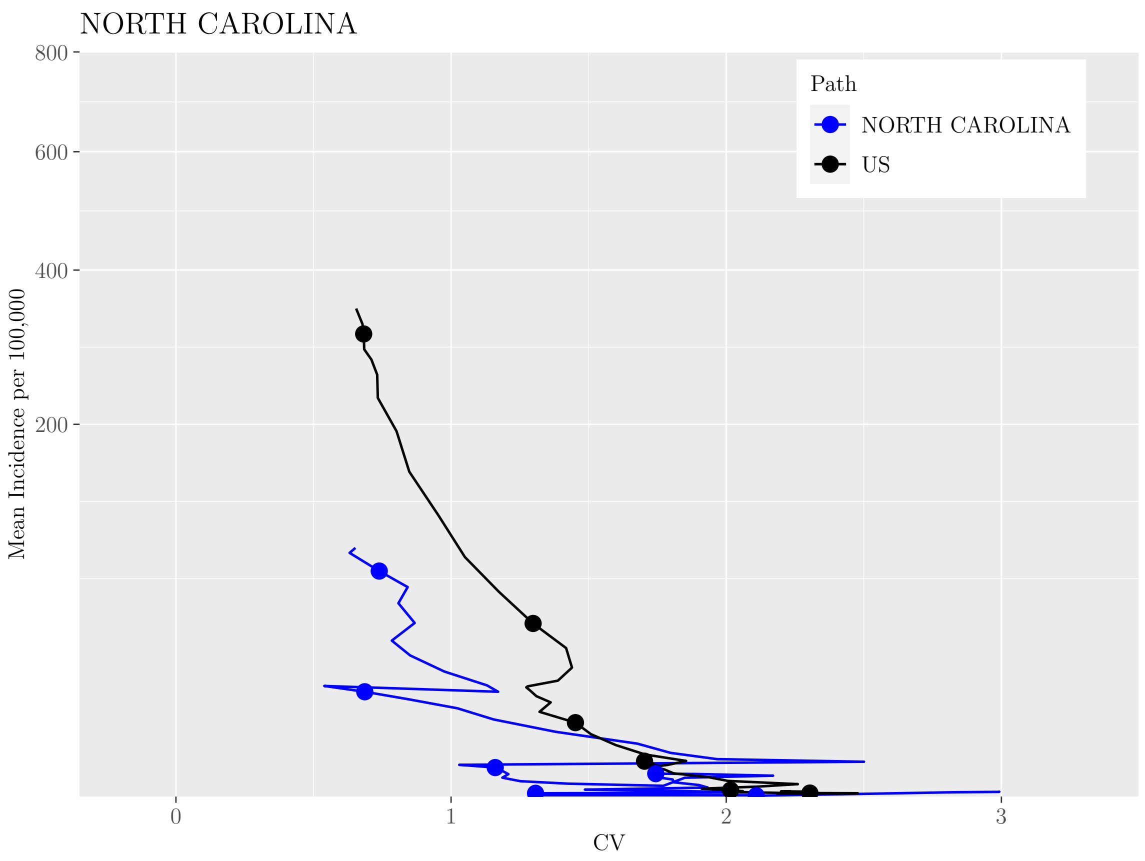

Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

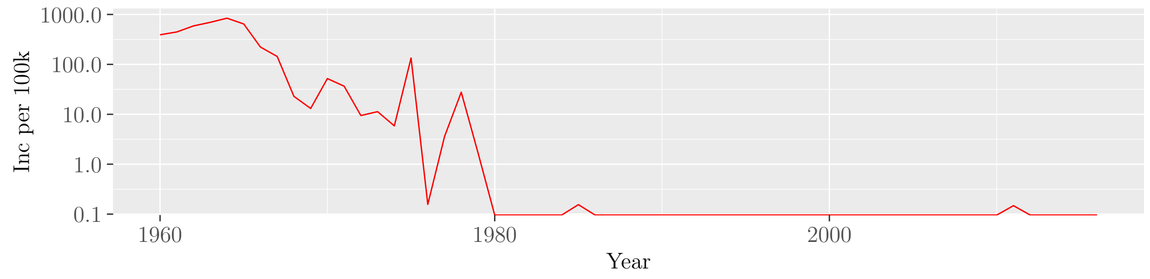

NEW MEXICO
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
NEW MEXICO
US


Inc per 100k
100.00
10.00
1.00
0.10
0.01
1960
1980
2000
Year

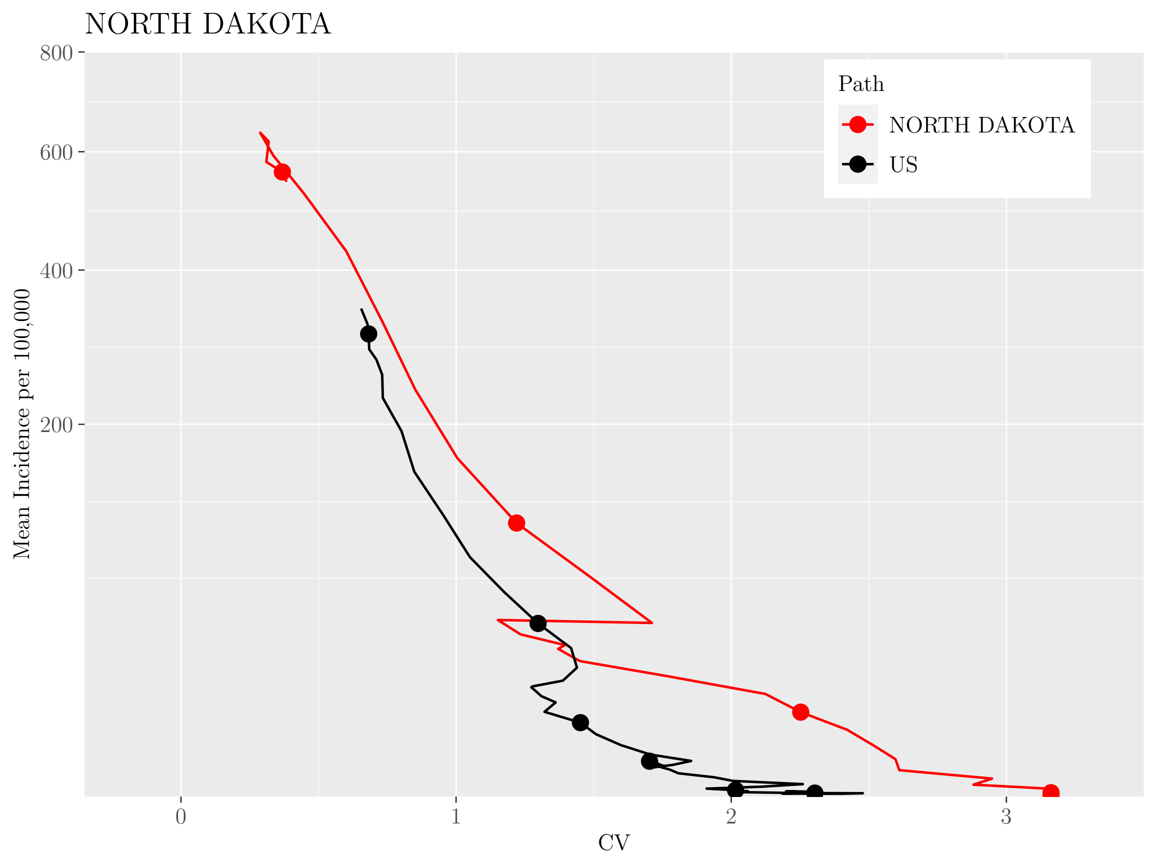

NEW YORK
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
NEW YORK
US


100.00
10.00
1.00
0.10
0.01
Inc per 100k
1960
1980
2000
Year

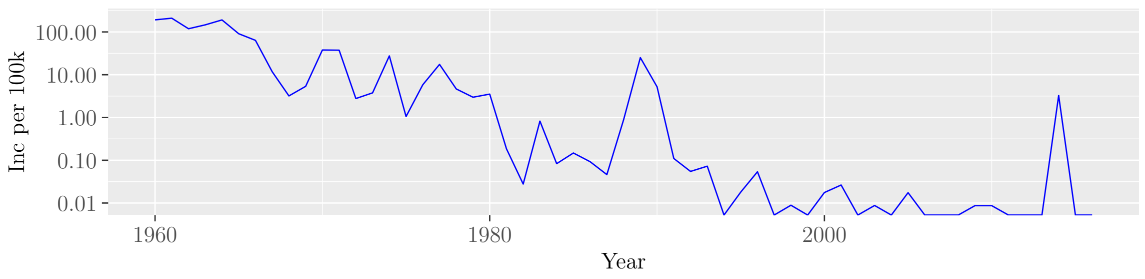

NORTH CAROLINA
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
NORTH CAROLINA
US



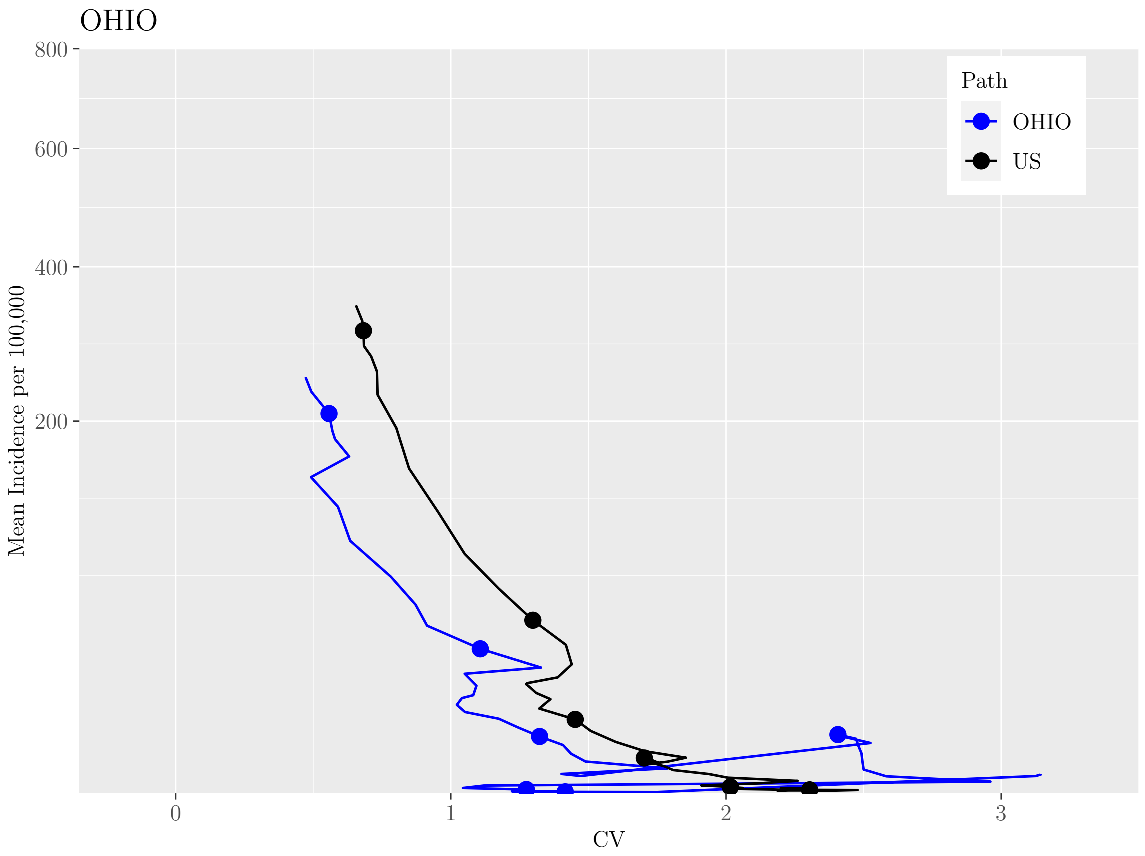
Inc per 100k
1000.0
100.0
10.0
1.0
0.1
1960
1980
2000
Year



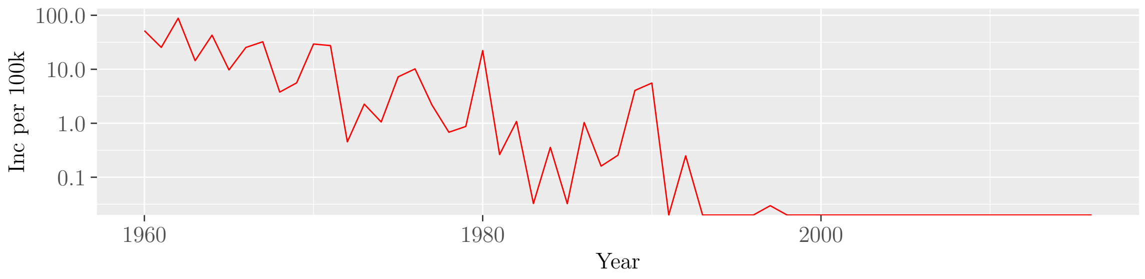
NORTH DAKOTA
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
NORTH DAKOTA
US



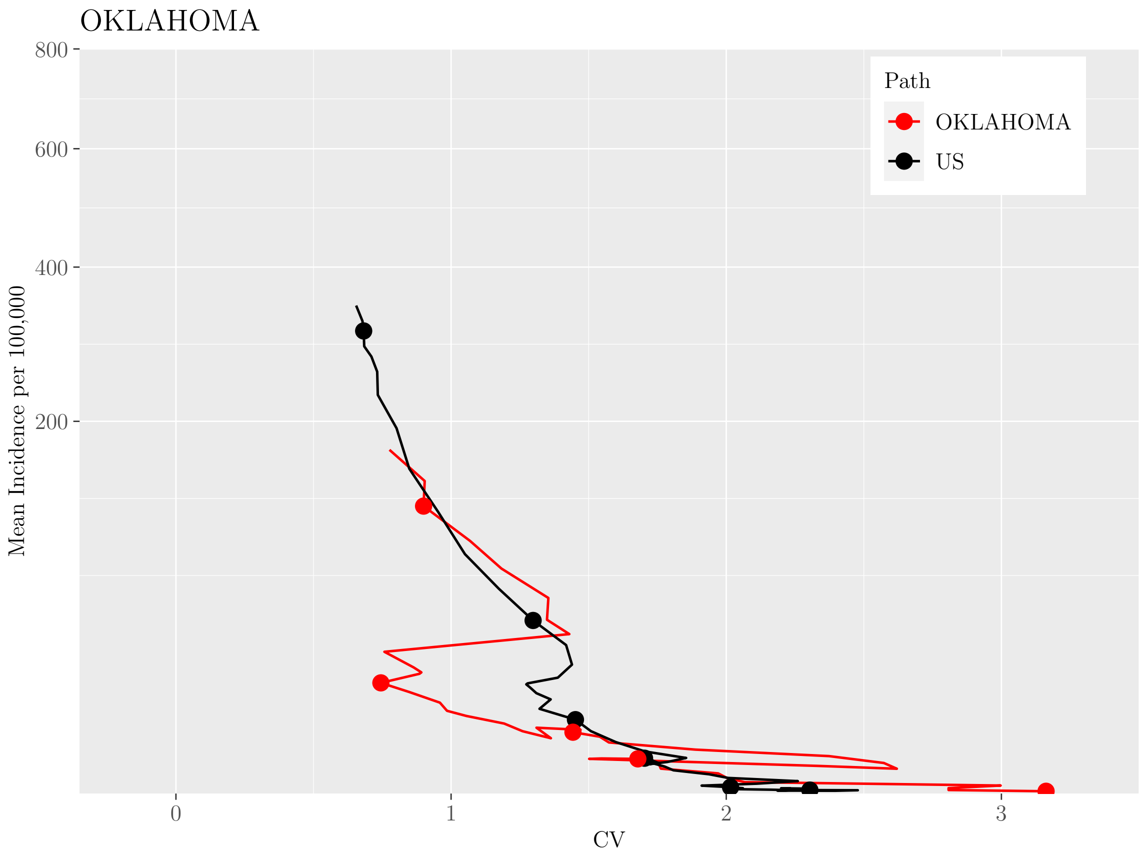
Inc per 100k
100.00
10.00
1.00
0.10
0.01
1960
1980
2000
Year



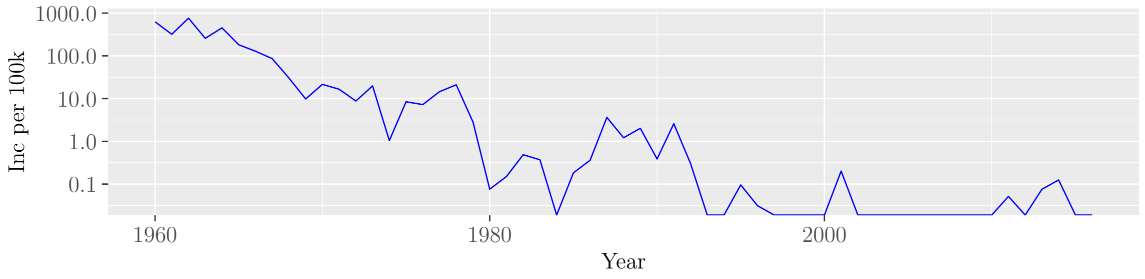
OHIO
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
OHIO
US

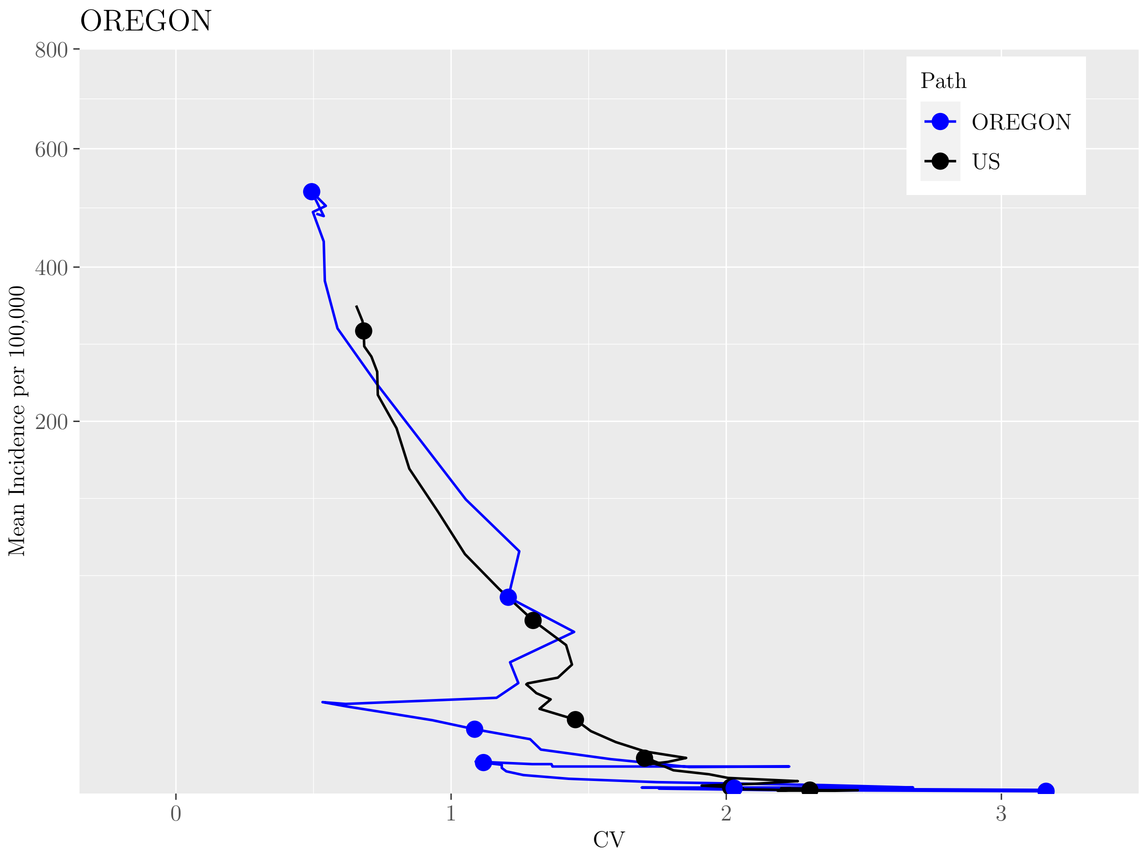


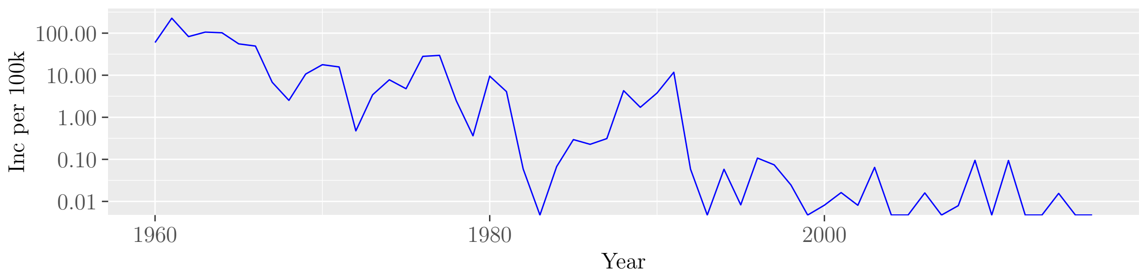
Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

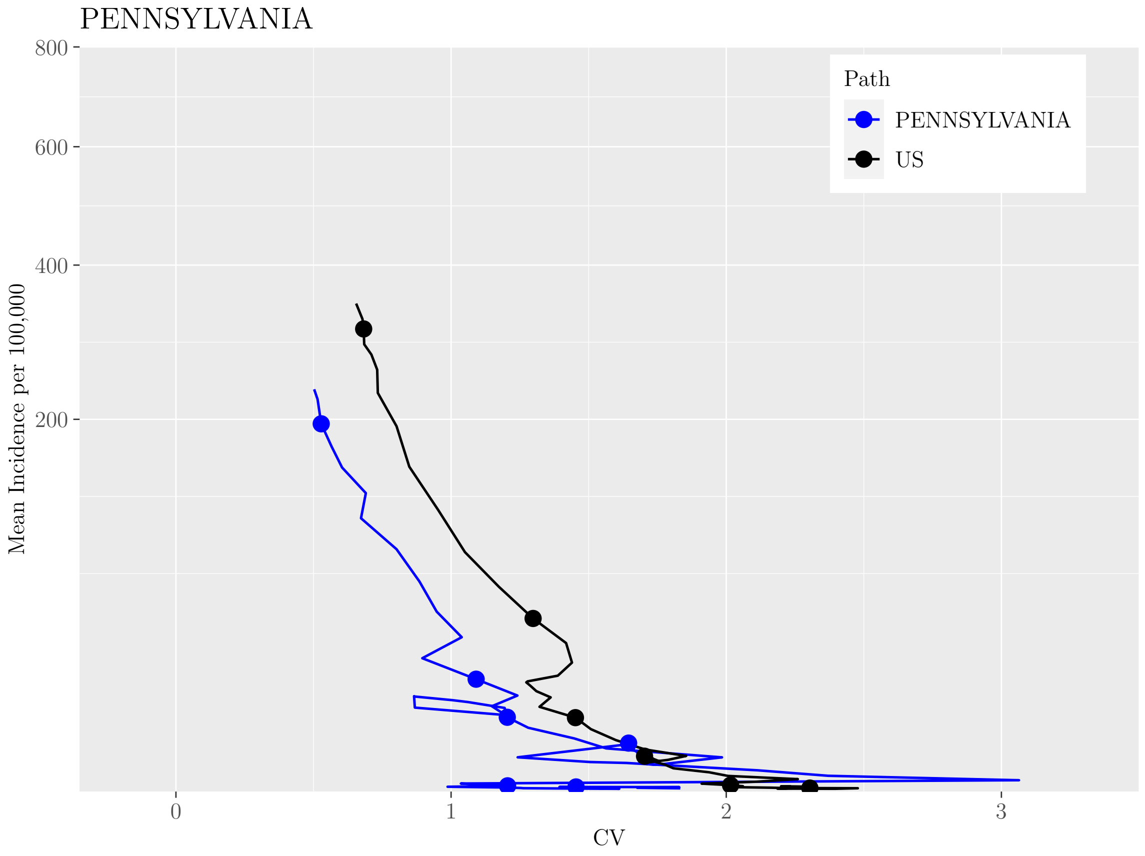


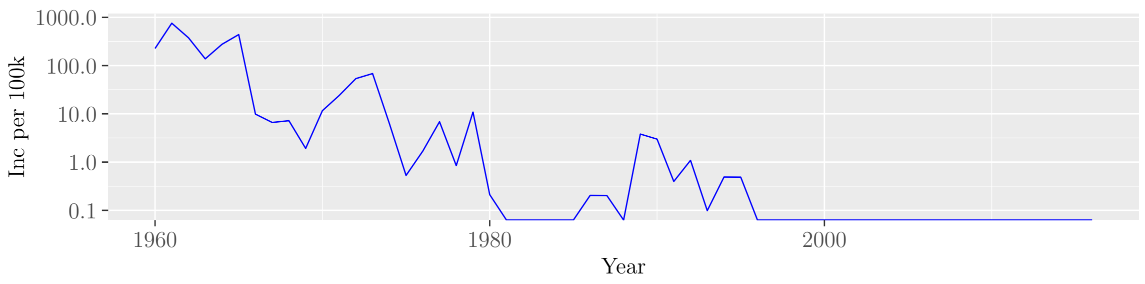
OKLAHOMA
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
OKLAHOMA
US


1000.0
100.0
10.0
1.0
0.1
Inc per 100k
1960
1980
2000
Year

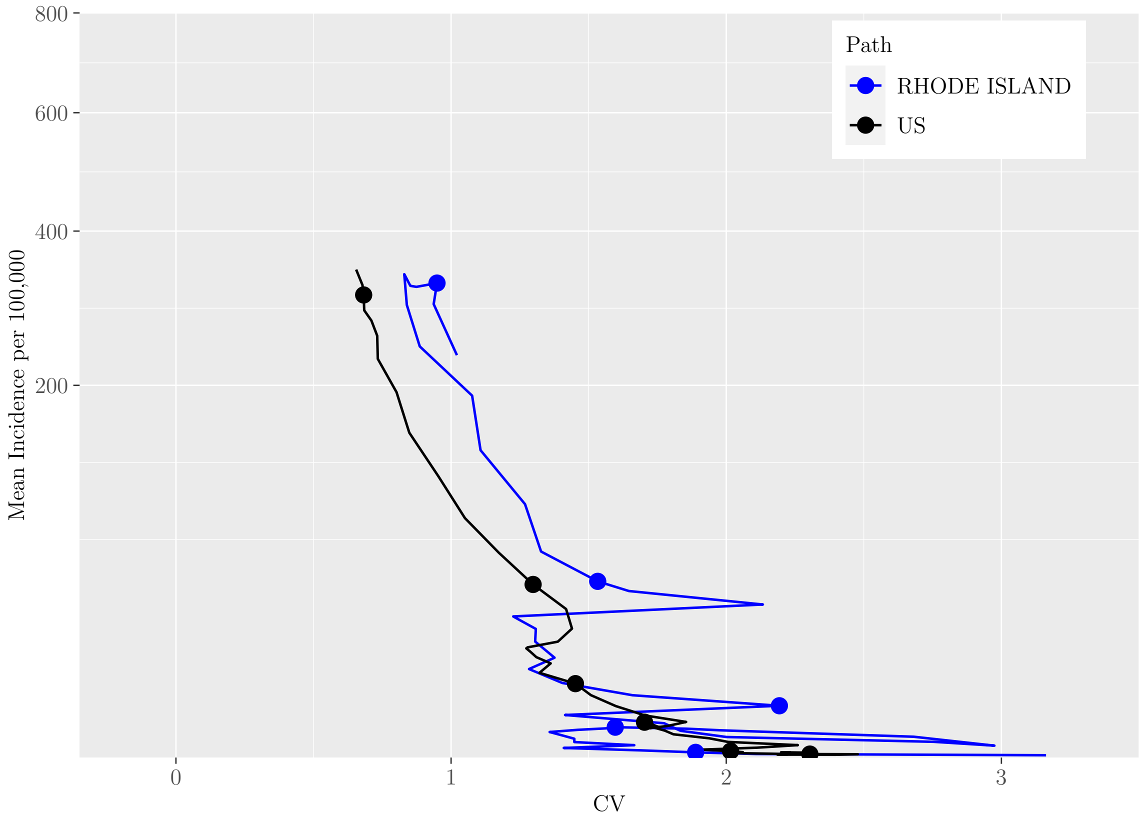

OREGON
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
OREGON
US

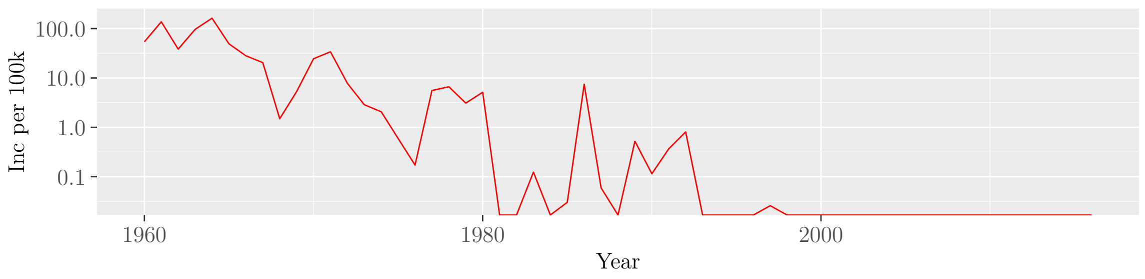

Inc per 100k
100.00
10.00
1.00
0.10
0.01
1960
1980
2000
Year

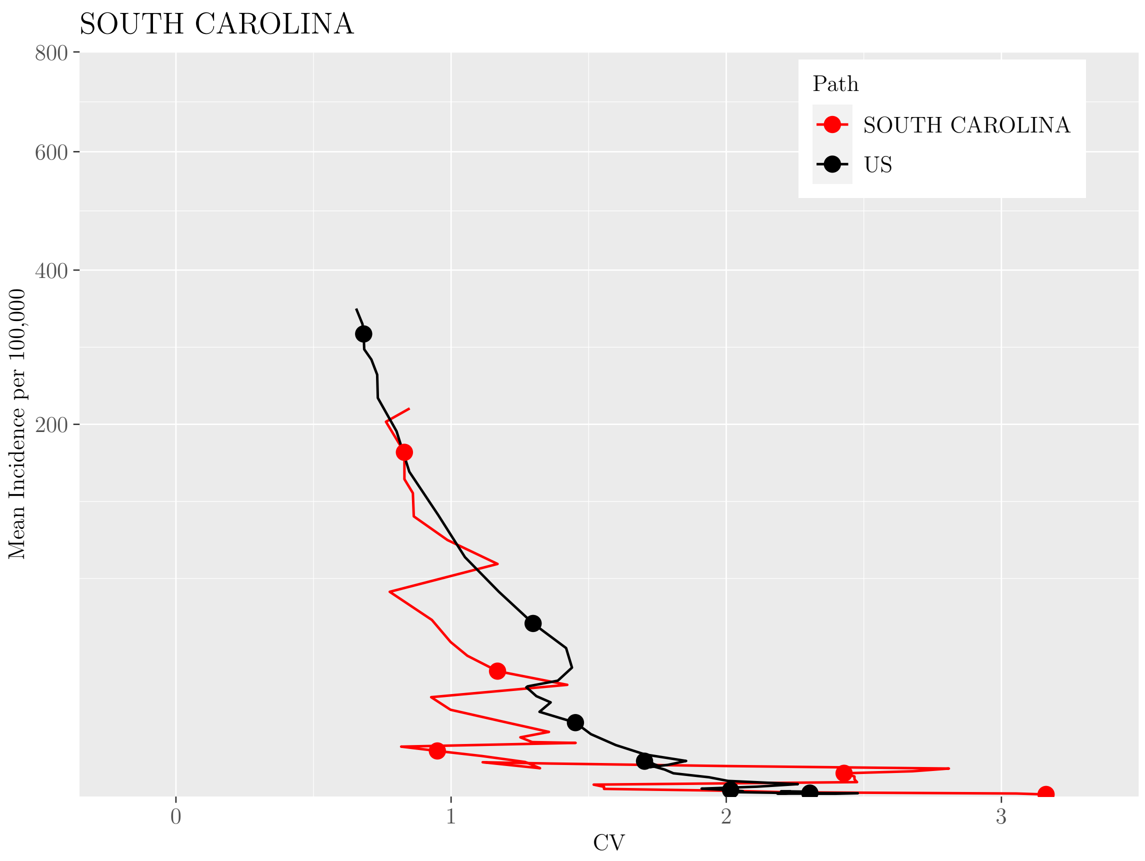

PENNSYLVANIA
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
PENNSYLVANIA
US

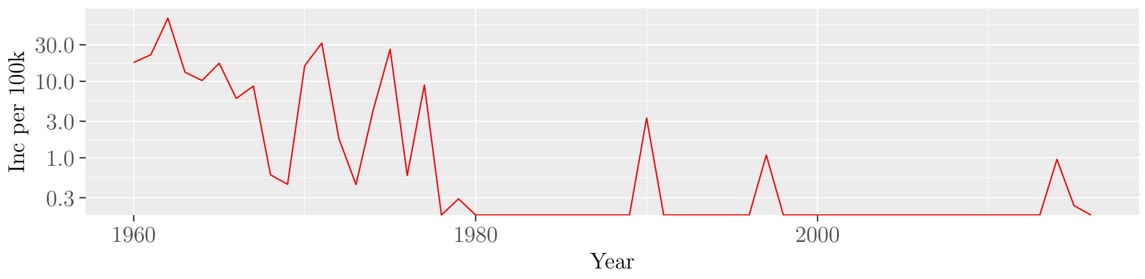


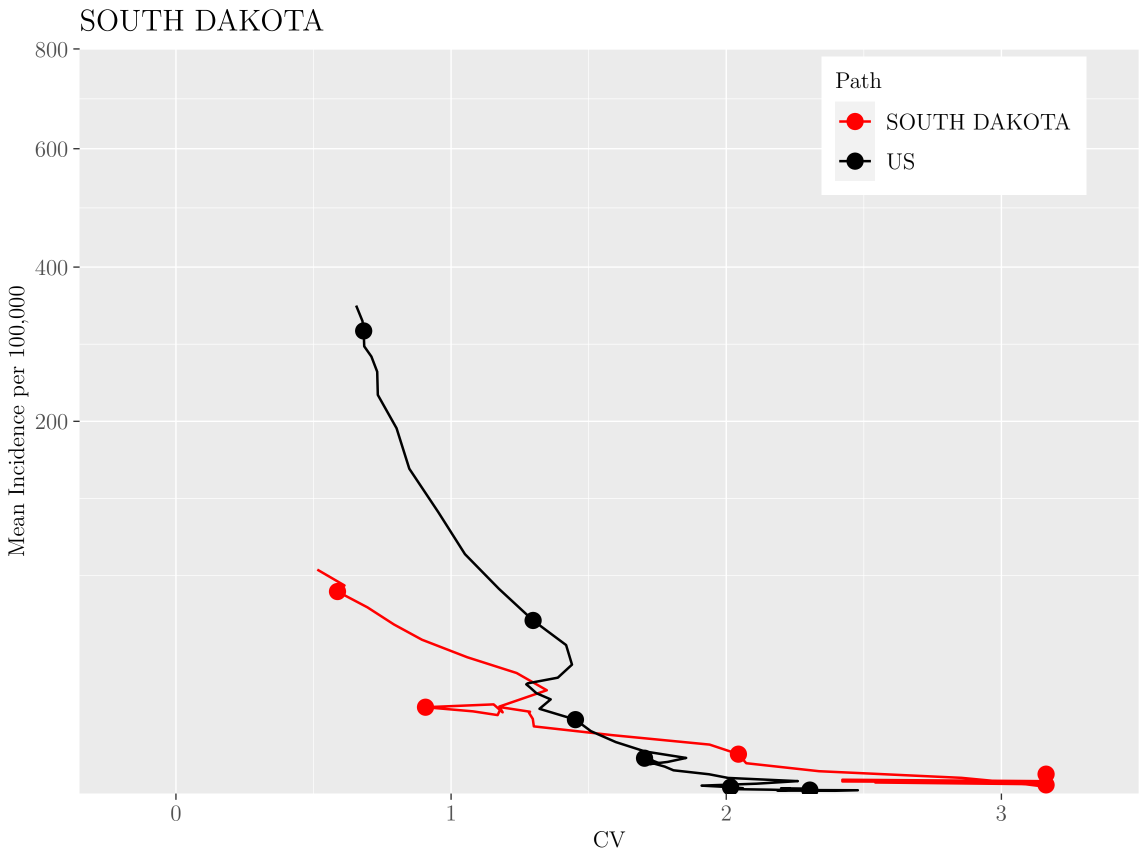
RHODE ISLAND

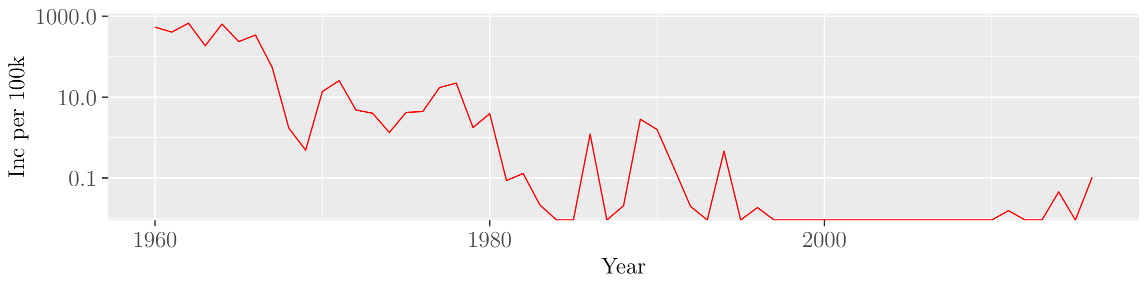


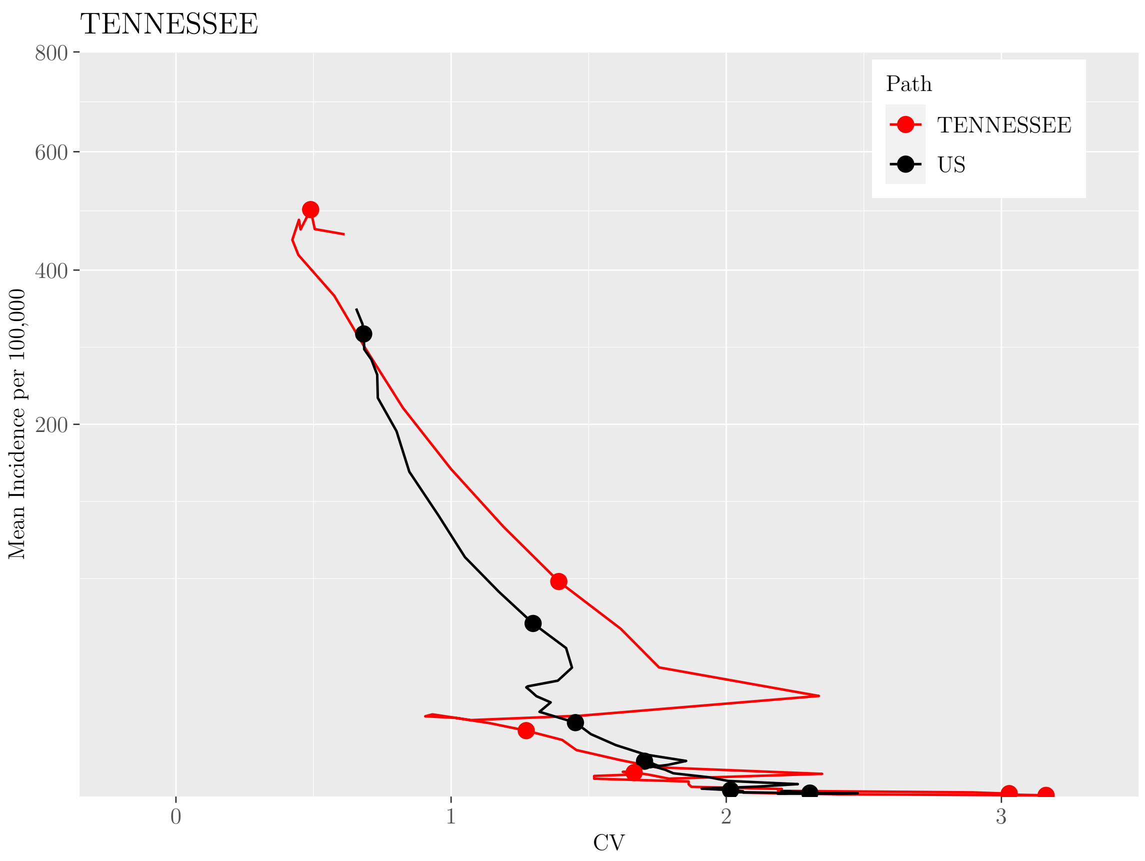

Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

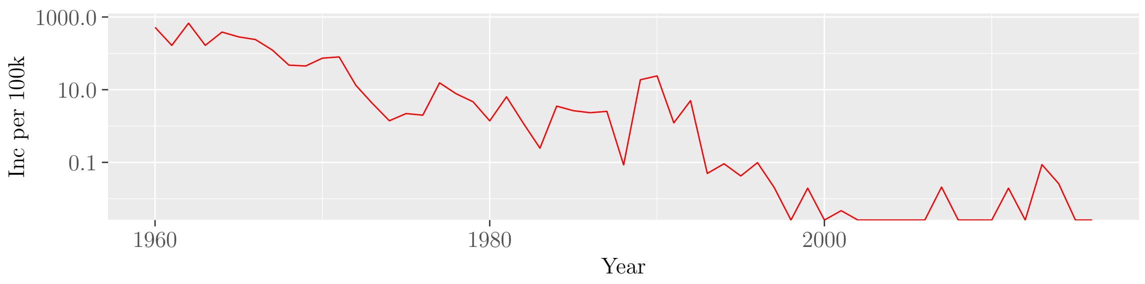

SOUTH CAROLINA
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
SOUTH CAROLINA
US

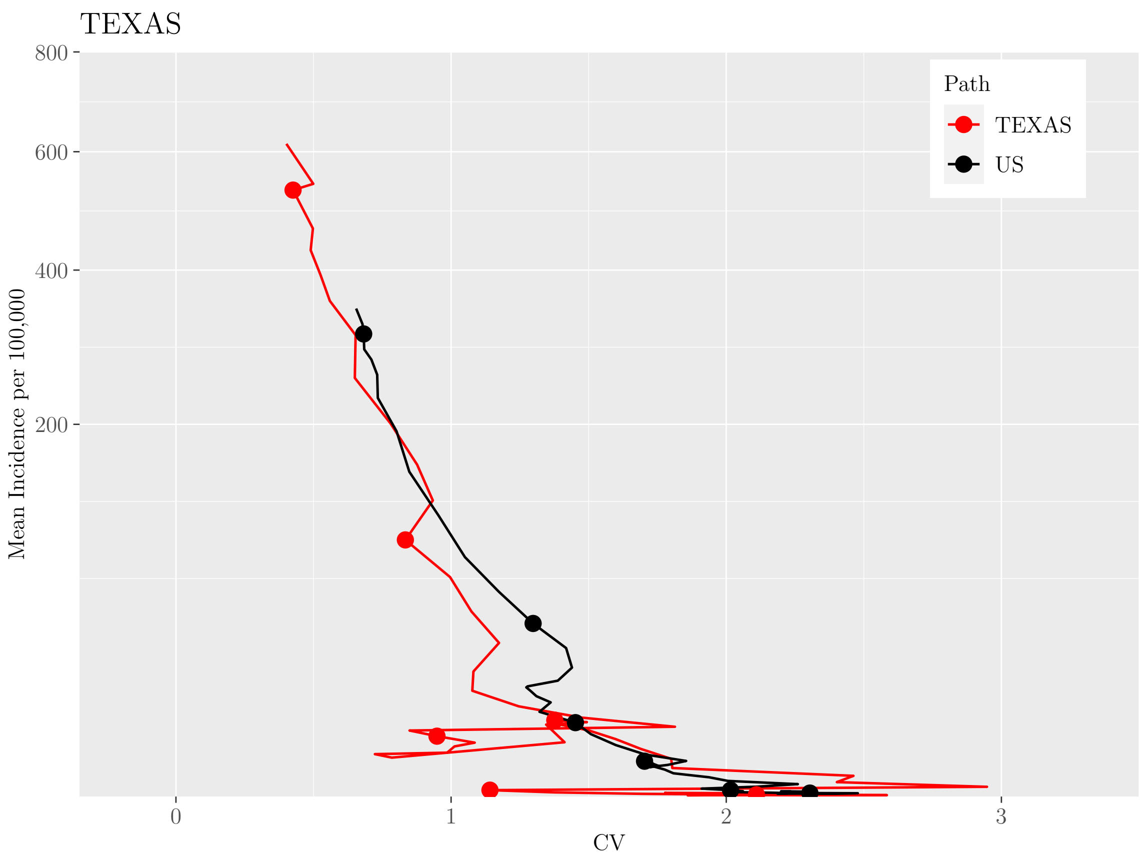

Inc per 100k
30.0
10.0
3.0
1.0
0.3
1960
1980
2000
Year

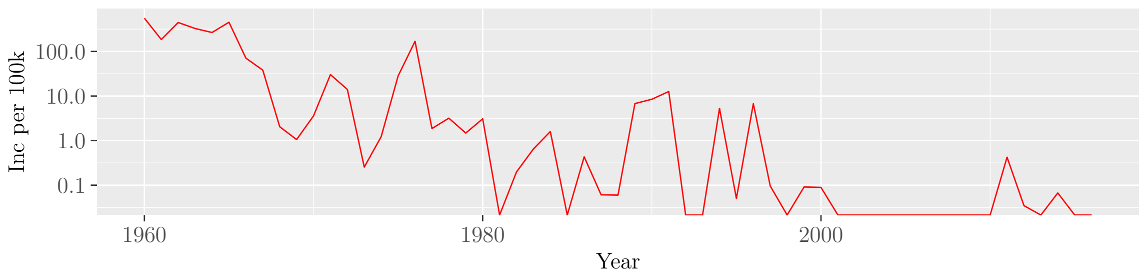

SOUTH DAKOTA
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
SOUTH DAKOTA
US

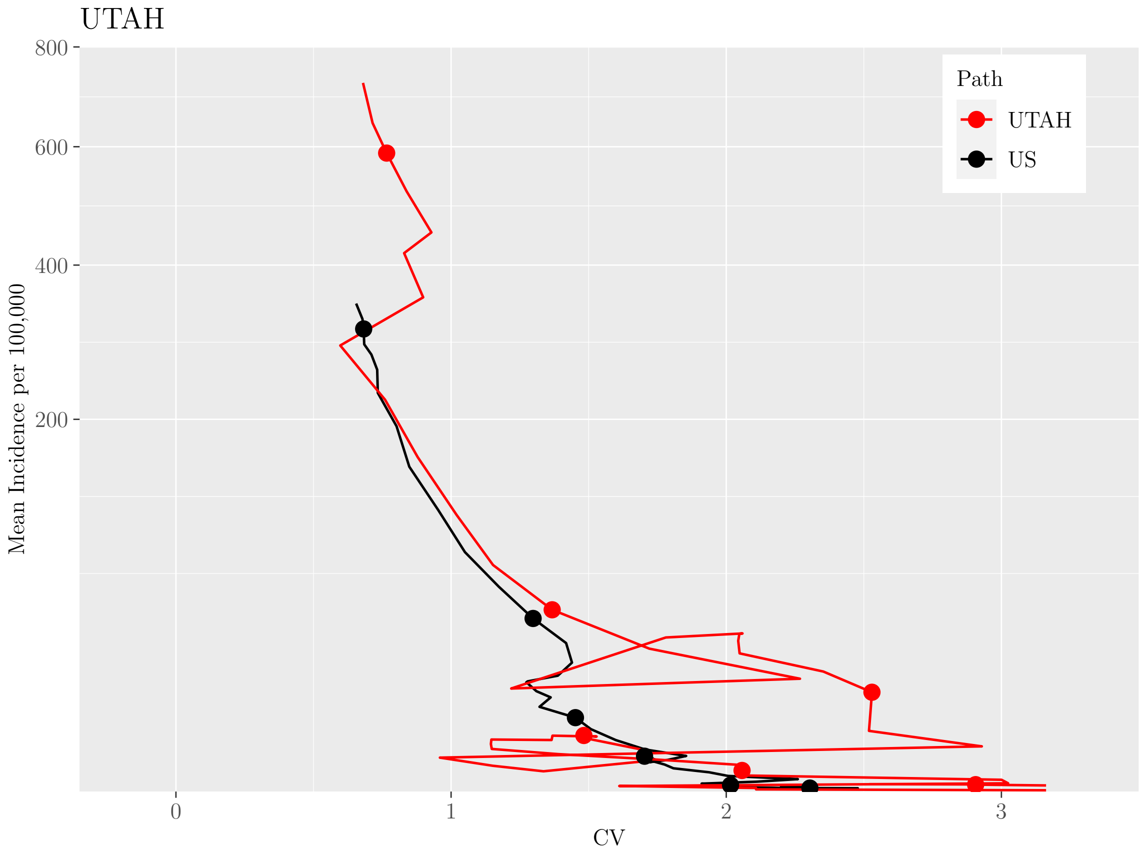

1000.0
10.0
0.1
Inc per 100k
1960
1980
2000
Year

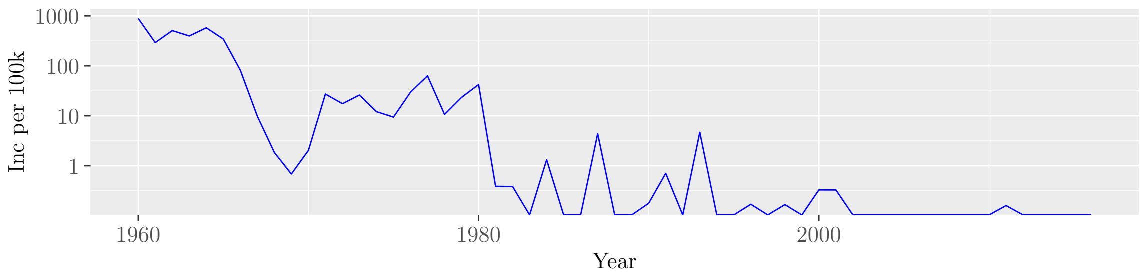

TENNESSEE
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
TENNESSEE
US

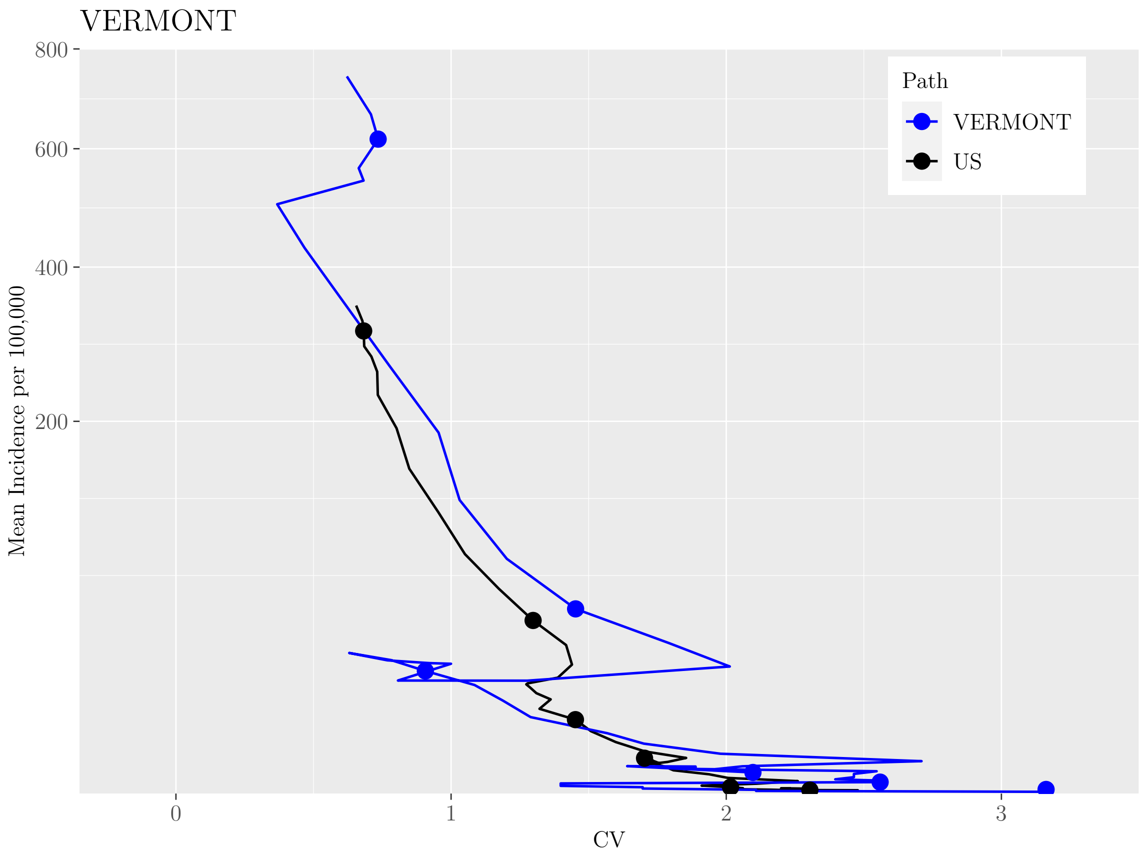

1000.0
10.0
0.1
Inc per 100k
1960
1980
2000
Year

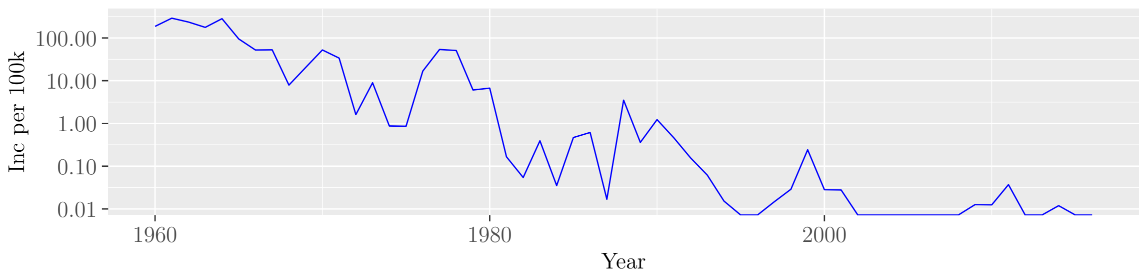

TEXAS
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
TEXAS
US


Inc per 100k
100.0
10.0
1.0
0.1
1960
1980
2000
Year

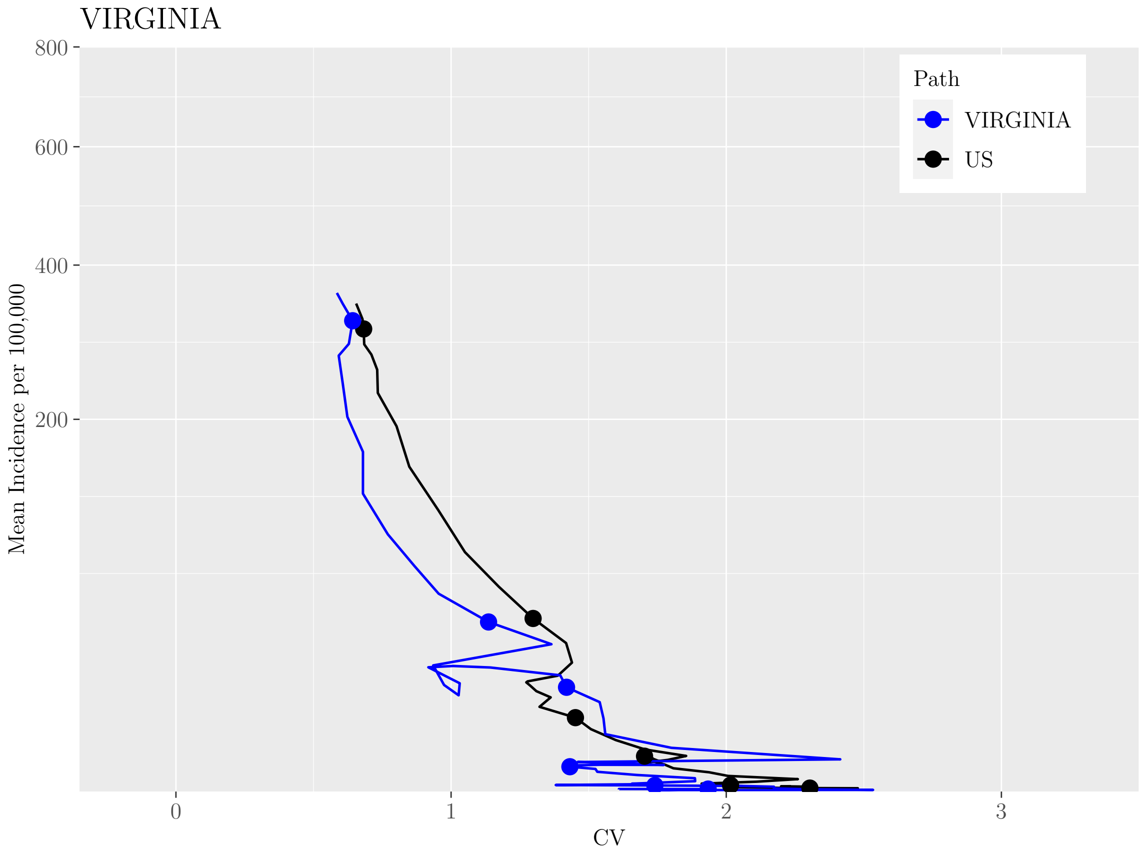

UTAH
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
UTAH
US


Inc per 100k
1000
100
10
1
1960
1980
2000
Year

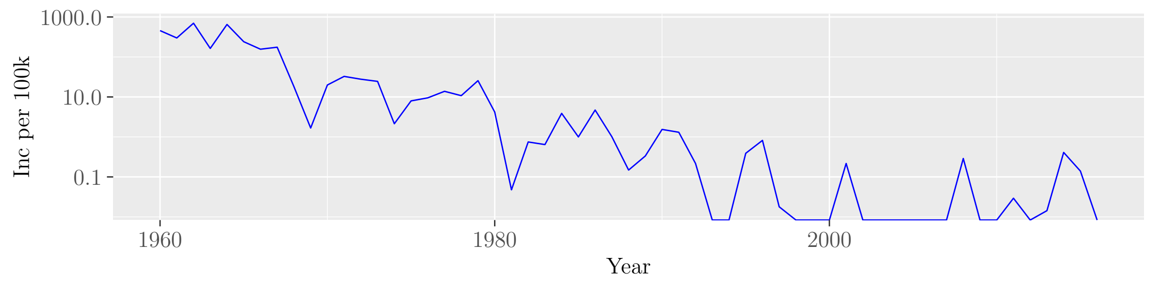

VERMONT
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
VERMONT
US

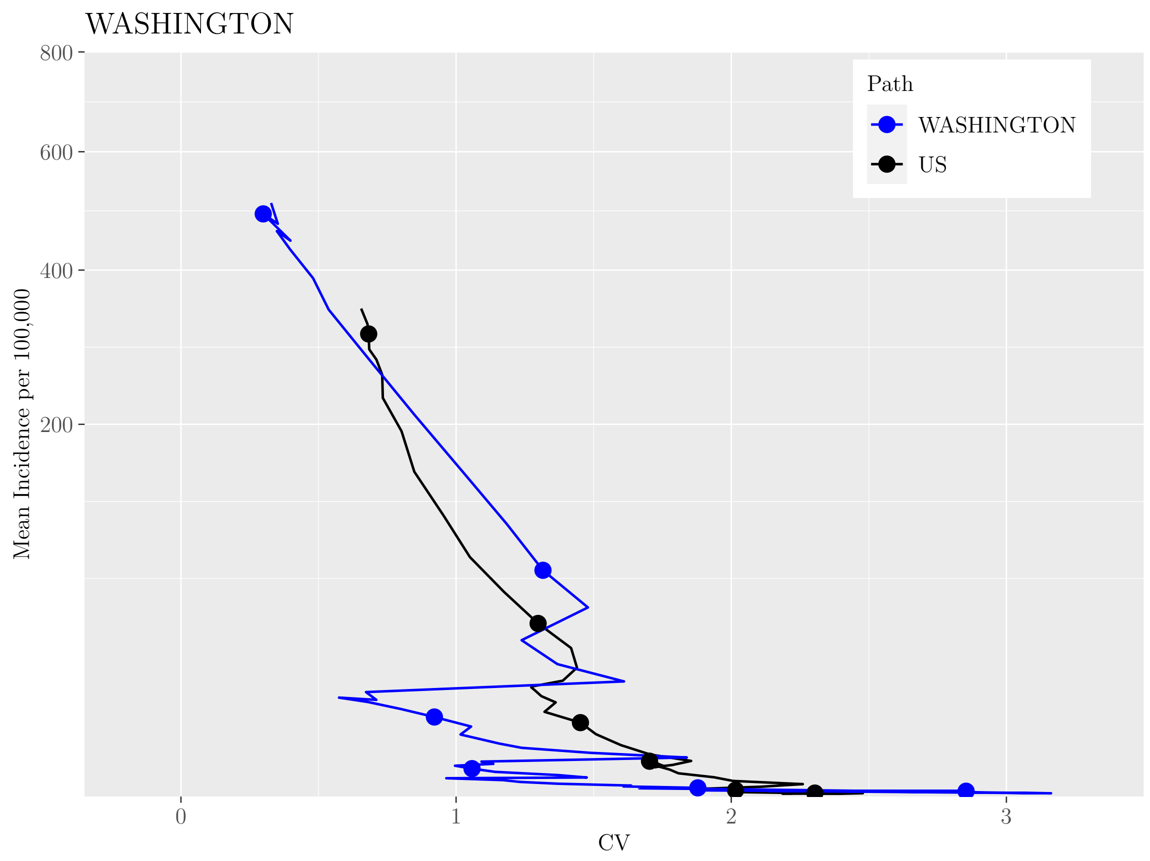


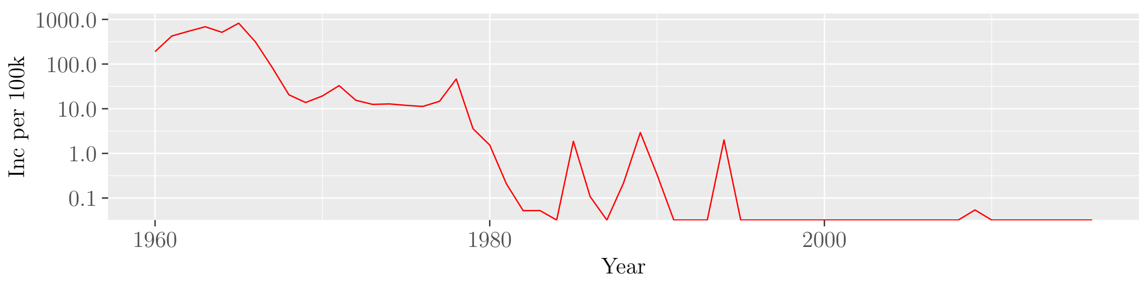
Inc per 100k
100.00
10.00
1.00
0.10
0.01
1960
1980
2000
Year

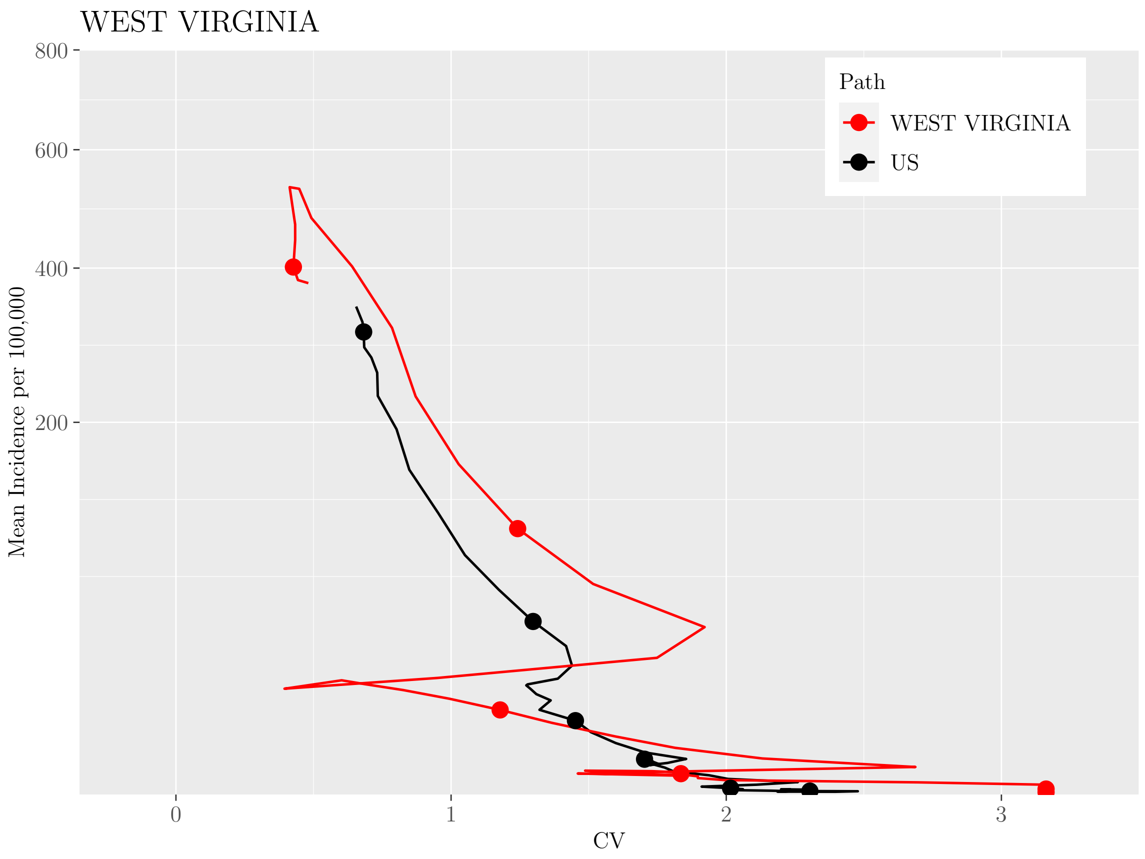


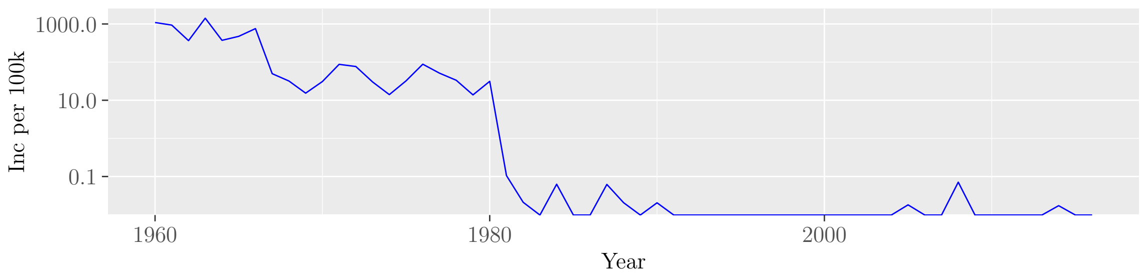
VIRGINIA
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
VIRGINIA
US

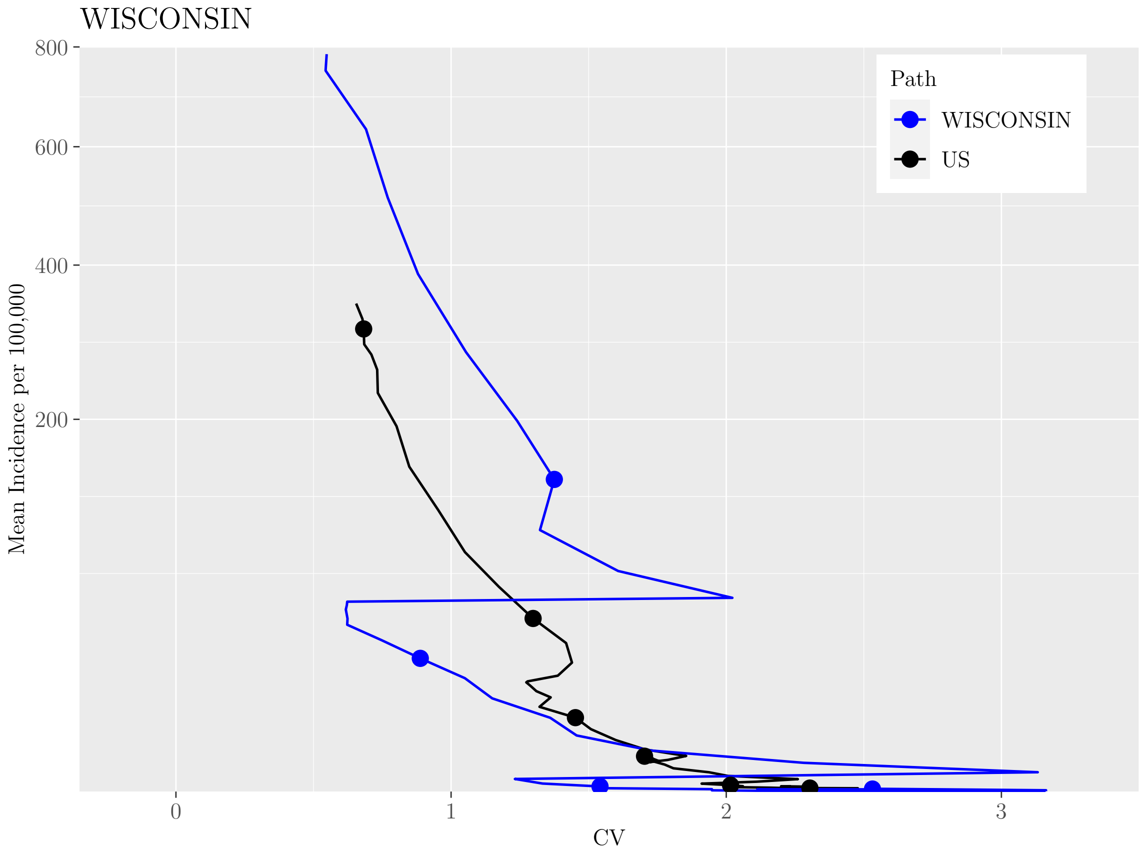

1000.0
10.0
0.1
Inc per 100k
1960
1980
2000
Year

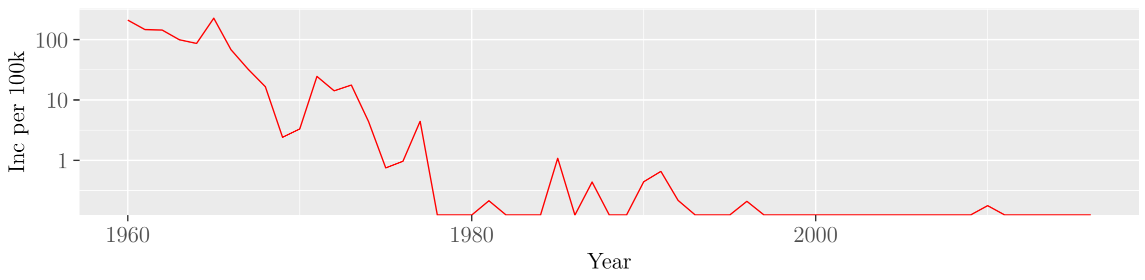

WASHINGTON
800
600
400
200
Mean Incidence per 100,000
0
1
2
3
CV
Path
WASHINGTON
US

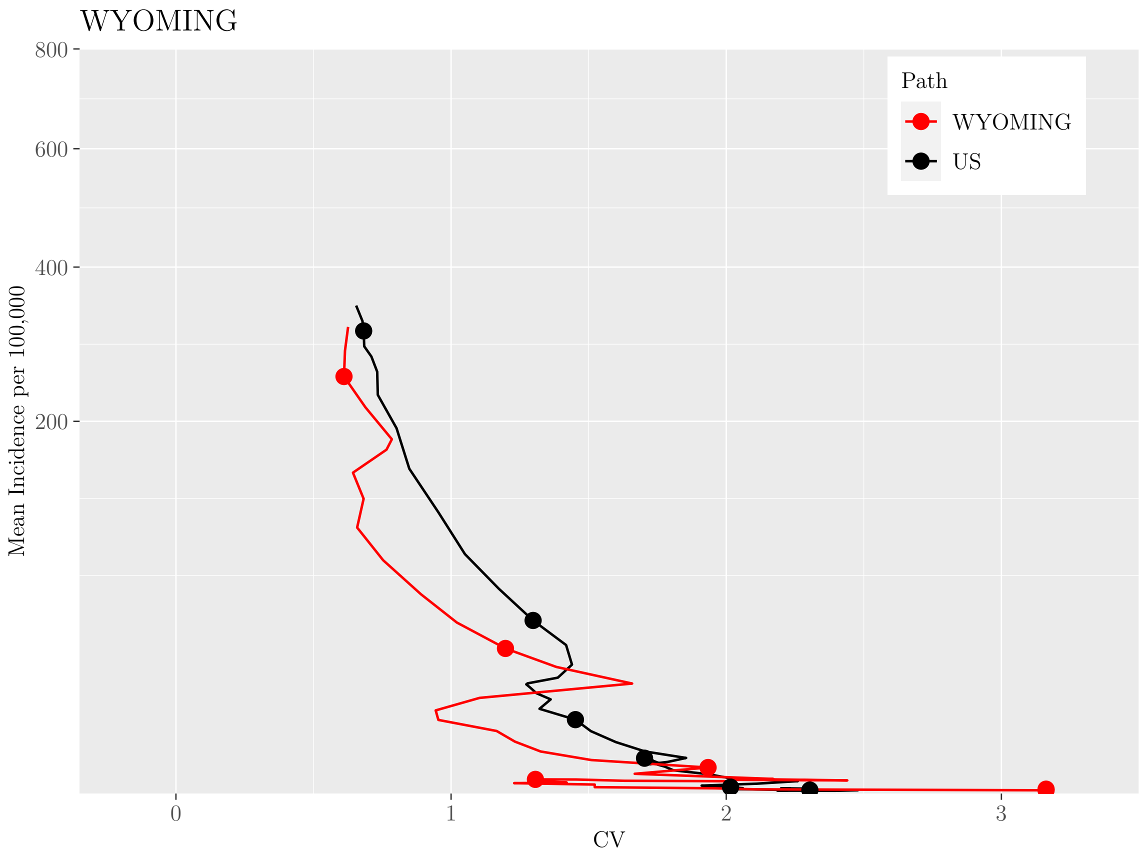

Inc per 100k
1000.0
100.0
10.0
1.0
0.1
1960
1980
2000
Year


WEST VIRGINIA
Mean Incidence per 100,000
800
600
400
200
0
1
2
3
CV
Path
WEST VIRGINIA
US


Inc per 100k
1000.0
10.0
0.1
1960
1980
2000
Year


WISCONSIN
800
600
400
200
Mean Incidence per 100,000
Path
WISCONSIN
US
0
1
2
3
CV





The results from sections 7.3.2 and 7.3.3 are then saved to a *.RData* file for use in other code.

```
file_name <- paste(paste("Good_data/true_path",
                         inc_window, sd, sep = "_"),
                   ".RData", sep = "")

save(true_path, data_us, final_data, point_mean_us, CV_states, Incidence_states, s
```

7.3.4 Analyzing the Distance Between State Paths and the US Path

7.3.4.1 Scaling and Distance Function Definitions

To determine how much each state varies from the US canonical path we find the distance between each state's path and that of the whole US using the distance as defined in equation 1. To find the distance, we created two different functions. The first function, `scale_cases_percentile()`, is used to scale incidence and CV so that the log of incidence and CV at the specified percentile is normalized to 1. We chose this method of scaling as opposed to having all of incidence and CV be between 0 and 1, as having incidence be between 0 and 1 would introduce noise before vaccination due to increased levels of variation.

```r
## Create a function that scales data so that the
## specified percentile value is 1
## Takes two arguements: data frame containing Year and
## either a column titled Incidence or Reports
## Returns two arguements: data frame that contains the
## scaled incidence/reports and the scaling value
## that all incidence values were divided by

scale_cases_percentile <- function(data, type, percentile,
                                   use_log) {
  ## Ensure that the inputs are of the proper format
  stopifnot(type %in% c("Incidence", "Reports") |
              is.numeric(percentile) |
              percentile < 1 | is.logical(use_log))
  require(dplyr)

  ## Take the log of incidence first if specified
  if(use_log) {
    data[,type] <- log(data[,type])
  }

  ## Filter out any CV with NA values
  data <- data %>%
    filter(!is.na(CV))

  ## Find the value of Incidence and CV at the
  ## specified percentile
  scale_value_case <- quantile(data[,type],
                               percentile, na.rm = TRUE)
  scale_value_cv <- quantile(data[,"CV"],
                             percentile, na.rm = TRUE)

  ## Divide the Incidence and CV by the percentile value
  scaled_data <- data
  scaled_data$CV <- data$CV / scale_value_cv
  scaled_data[, type] <- data[, type] / scale_value_case

  ## Return the scaled data AND the values used to scale the
  ## data for incidence and CV
  return(list(scaled_data = scaled_data,
              scale_value_case = scale_value_case,
              scale_value_cv = scale_value_cv))
}
```

The function takes the arguments: `data`, `type`, `percentile`, and `use_log`. `data` is the data we want to scale, `type` is either "Incidence" or "Reports" depending on how the column is labeled in the data frame, `percentile` is a value between 0 and 1 telling the function which percentile to use, e.g., the 90th percentile would be 0.9, and `use_log` is logical and tells the function whether it should take the log of incidence prior to scaling. The function outputs the scaled data as well as the values used to scale incidence and CV. This functionality allows us to use the scaling values on data sets we wish to compare to the scaled data.

The second function, `distance_btw_canon_path()`, finds the distance between canonical paths using the Euclidean distance as defined in equation 1. The function takes the following arguments: `orig_data`, `compare_data`, `type_orig`, `type_compare`, and `data_years`. `orig_data` is the data frame containing the data that was scaled based on its own specified percentile, `compare_data` is a list containing all data frames you wish to compare with `orig_data`, such that they are all scaled based on the specified percentile of `orig_data`. `type_orig` and `type_compare` tell the function whether incidence is labeled as "Incidence" or "Reports". Finally, `data_years` is a vector containing the years for which data should be compared.

First, the function determines whether the function arguments are of the correct format. Next, it creates an empty vector of length of the `compare_data` list that will contain the distance measure for each comparison. The function filters the data so that we are only comparing the years of data specified by `data_years`, and makes the "Year" column numeric. If there is no data to compare, the function sets the distance to `NA`. Otherwise, it finds the mean of squared differences and outputs it into the vector. The function outputs the vector containing the distance values, named using the names of `compare_data`.

```
### A function that provides a metric for sensitivity by
### finding the euclidean distance between paths
distance_btw_canon_path <- function(orig_data, compare_data,
                                    type_orig,
                                    type_compare, data_years) {
  ## Check that the inputs are the correct format
  stopifnot(type_orig %in% c("Incidence", "Reports"),
            type_compare %in% c("Incidence", "Reports"),
            is.list(compare_data))
  require(dplyr)
  ## Create the empty vector
  sensitivity <- vector(length = length(compare_data))

 ## Filter the year column to only include the specified years
  for (i in 1:length(compare_data)) {
    orig_data_filtered <- orig_data %>%
      filter(Year %in% data_years & !is.na(CV))
    compare_data_filtered <- compare_data[[i]] %>%
      filter(Year %in% data_years & !is.na(CV))

    if (nrow(compare_data_filtered) == 0) {
      sensitivity[i] <- NA
    } else {
      ## If the years still are not the same,
      ## it filters the data frame further based on
      ## which data frame has more arguments
      if (!identical(orig_data_filtered$Year,
                     compare_data_filtered$Year)) {
        if (orig_data_filtered$Year[length(orig_data_filtered$Year)] >
            compare_data_filtered$Year[length(compare_data_filtered$Year)] |
            length(orig_data_filtered$Year) >
            length(compare_data_filtered$Year)) {
          orig_data_filtered <- orig_data_filtered %>%
            filter(Year %in% compare_data_filtered$Year)
        } else {
          compare_data_filtered <- compare_data_filtered %>%
            filter(Year %in% orig_data_filtered$Year)
        }
      }

      ## Make the Year column numeric
      orig_data_filtered$Year <-
```

```
        as.numeric(orig_data_filtered$Year)
      compare_data_filtered$Year <-
        as.numeric(compare_data_filtered$Year)

      ## Check that the year columns are the same
      stopifnot(identical(orig_data_filtered$Year,
                          compare_data_filtered$Year))

      sq_diff_inc <-
        (orig_data_filtered[,type_orig] -
           compare_data_filtered[,type_compare])^2
      sq_diff_cv <-
        (orig_data_filtered[,"CV"] - compare_data_filtered[,"CV"])^2

      ## Find the mean of squared differences and put it in the vector
      sensitivity[i] <-
        sqrt(mean((sq_diff_cv + sq_diff_inc)))
    }

  #   }
  }
  ## Name the vector using the names of the data list
  names(sensitivity) <- names(compare_data)

  ## Return the list of distances
  return(sensitivity = sensitivity)
}
```

To choose the value of the percentile, we conducted a sensitivity analysis to determine whether the choice of the percentile changes the time-progression of the path. We begin by loading data for our path created using real data, source `functions.R`, and load the necessary libraries.

```
library(tidyverse)
library(knitr)
source("functions.R")
load("Good_data/true_path_10_3.RData")
```

Next, we define the arguments required to use the scaling function.

```r
## Path data using real US data
real_data <- data_us
## Vector of percentiles we wish to use
percentiles <- seq(0.8, 0.95, 0.05)
## Empty list that will house scaled data and scaling factors
scaled_real_data_percentile <- list()
use_log <- TRUE
```

We then run a for loop through each value of the percentile and create a single data frame containing the scaled data for each percentile.

```r
for (i in 1:length(percentiles)) {
  ## Scaling the real data using scale_percentile
  scale_data <-
    scale_cases_percentile(real_data,
                           "Incidence",
                           percentiles[i],
                           use_log)
  scaled_real_data_percentile[[i]] <- scale_data$scaled_data
}

## Set the names of the lists to be the percentiles
names(scaled_real_data_percentile) <- percentiles

## Create a single data frame for the individual data
## with a Percentile column
all_percentile_data <-
  bind_rows(scaled_real_data_percentile,
            .id = "Percentile")
```

Finally, we create a figure comparing the scaled paths for each percentile. In this plot we observe no difference in the shape of the path. The only difference across the paths is that as you increase the percentile, the path shrinks. This suggests that the choice of the percentile in this range does not matter. Thus, we choose to use a percentile of 0.9.

```r
scaled_canonical_path_real_percentile_vary <- ggplot() +
  geom_path(data = all_percentile_data,
            aes(CV, Incidence,
                colour = Percentile),
            lty = 1, lwd = 1) +
  scale_y_sqrt("Mean Incidence per 100,000") +
  scale_x_continuous("CV", limits = c(0,1.25),
                     expand = c(0,0))
scaled_canonical_path_real_percentile_vary
```

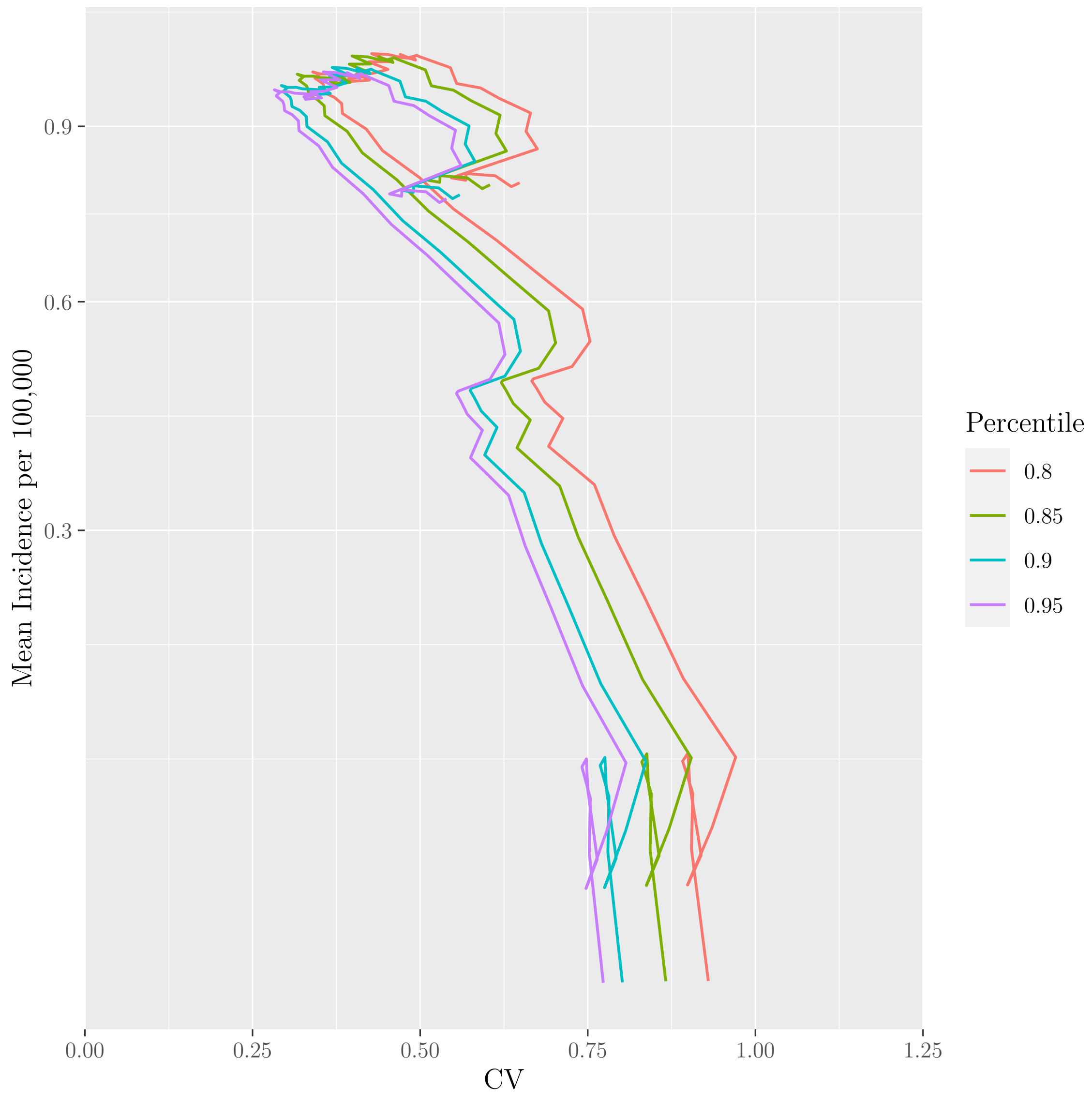


Figure 27: Demonstration of how changing the percentile used in scaling changes the path.

Additionally, we save the plot to a *.RData* file.

```
save(scaled_canonical_path_real_percentile_vary,
     file = "data/scaling_plots.RData")
```

7.3.4.2 Evaluating the Distance Between Each State and the Aggregate US Path

Using the functions as defined, we find the distance between each state's canonical path and the US canonical path from 1960 to 2016. We begin by loading `tidyverse`, sourcing the `functions.R` file and loading the data frames for the canonical paths.

```
library(tidyverse)
library(knitr)
library(gridExtra)
library(ggpubr)
source("functions.R")
load("Good_data/true_path_10_3.RData")
load('Good_data/yearly.data.per.100k.RData')
load("data/presentation_logical.RData")
```

Next, we rename the data frame containing the US canonical path data and add a column corresponding to the year. We set `data_years` to be all years from 1960 to 2016 and set `use_log`=`TRUE`. Using these arguments, we call the `scale_cases_percentile()` function to scale the data using the 90th percentile.

```
## Filter the data to only contain the years in which
## states reported uniformly Additionally rename the US
## path data and add Year
real_data <- data_us %>%
  mutate(Year = as.numeric(year)) %>%
  filter(Year > 1959)

## Define function arguments
data_years <- real_data$Year
use_log <- TRUE
percentile <- 0.9

## Call the function and set variables to each
## component of the function output for use later on
scale_real_data_values <-
```

```
  scale_cases_percentile(real_data,
                         "Incidence",
                         percentile,
                         use_log)

scale_factor_reports_state <-
  scale_real_data_values$scale_value_case

scale_factor_cv_state <-
  scale_real_data_values$scale_value_cv

scale_real_data <-
  scale_real_data_values$scaled_data
```

Now that the US path data is scaled, we prepare the individual state path data for scaling. First we transform the data from wide-format to long-format and make sure that the "Year" and "State" columns are identical in the Incidence and CV data frames. Additionally, we bind the columns of the CV data frame to the Incidence column from the incidence data frame using `cbind()` to create one data frame.

```
## Convert to long format
long_CV_state <-
  pivot_longer(CV_states, !Year,
               names_to = "State",
               values_to = "CV")
long_Incidence_state <-
  pivot_longer(Incidence_states, !Year,
               names_to = "State",
               values_to = "Incidence")

## Check that the state and year columns are the same
setequal(long_CV_state$Year,
         long_Incidence_state$Year) &
  setequal(long_CV_state$State, long_Incidence_state$State)

## Combine the CV data frame to the incidence column
## to make one cohesive data frame,
## naming the columns accordingly
state_data <-
  cbind(long_CV_state, long_Incidence_state$Incidence)
colnames(state_data) <-
  c(colnames(long_CV_state), "Incidence")
```

With a single data frame, we divide CV by the defined CV scaling factor and divide the log of incidence by the defined incidence scaling factor, resulting in scaled data.

```
## Scale log incidence and CV by dividing by the pre-defined
## scaling factors
scaled_state_data <- state_data

scaled_state_data$CV <- state_data$CV/scale_factor_cv_state
scaled_state_data$Incidence <-
  log(state_data$Incidence)/scale_factor_reports_state
```

Since we must have a list containing the data frame of each state, we use the following code to filter the data for each state and enter it into a list.

```
## Create the empty list
scaled_state_data_list <- list()

states <- unique(scaled_state_data$State)

## Filter the data frame for each state and making it an entry in the list
for (i in 1:length(states)) {
  filtered <- scaled_state_data %>%
    filter(State == states[i])

  scaled_state_data_list[[i]] <- filtered
}

## Name the data frames using the names of the states
names(scaled_state_data_list) <- states
```

Finally, we call the `distance_btw_canon_path()` function to find the distance between each US state and the aggregate US path between 1960 and 2016.

```
## Call the defined distance function
distance_states <-
  distance_btw_canon_path(scale_real_data,
                          scaled_state_data_list,
                          "Incidence", "Incidence",
                          data_years)
```

```r
## Make the output containing the distances
## a data frame (so that we can visualize the results)
states_data <- as.data.frame(distance_states)
colnames(states_data) <- "Distance"
```

The following plot shows a histogram of the distances between each state and the aggregate US path.

```r
ggplot() + geom_histogram(data=states_data, aes(x=Distance))
```

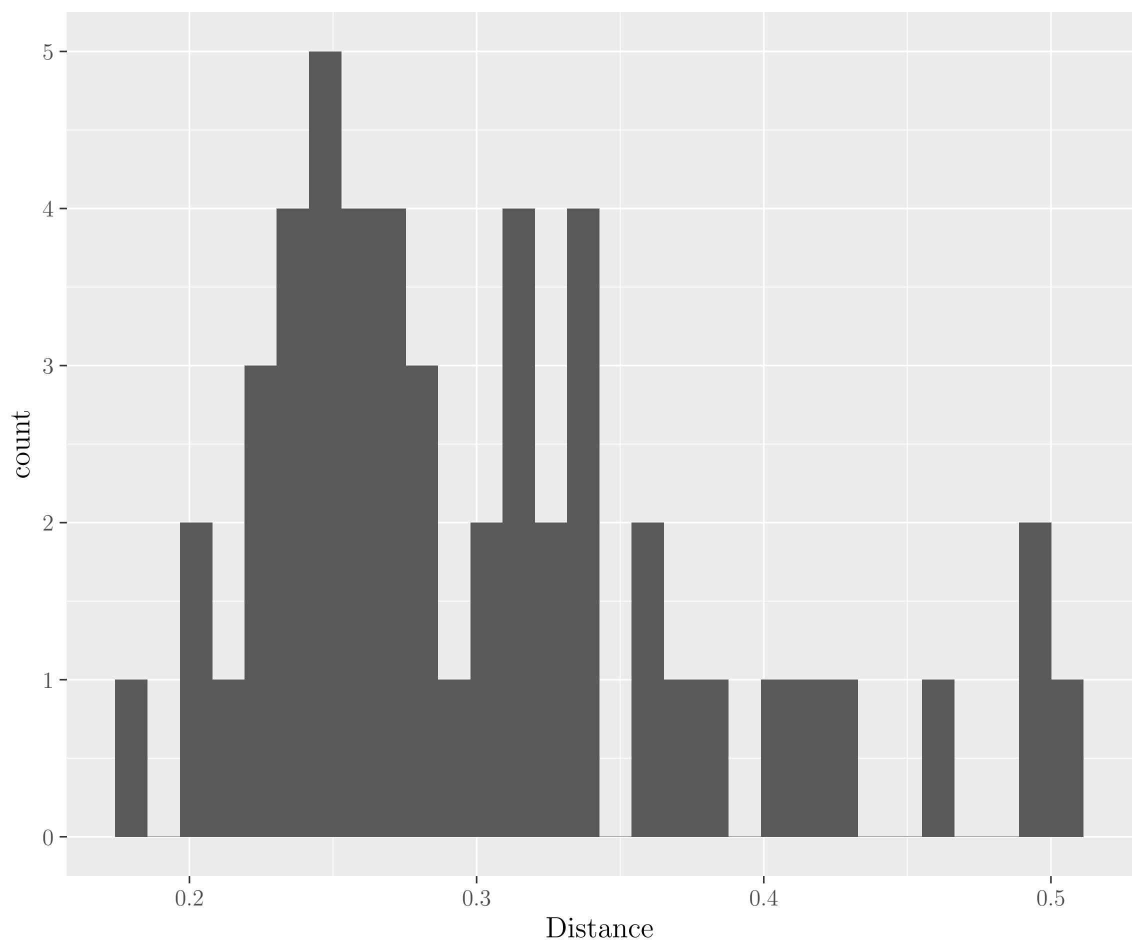


Figure 28: Histogram of the distances between US states and the aggregate US path. The smallest distance corresponds to Virginia and the largest distance corresponds to North Dakota.

We use the `summary()` function to obtain summary statistics pertaining to the median and IQR.

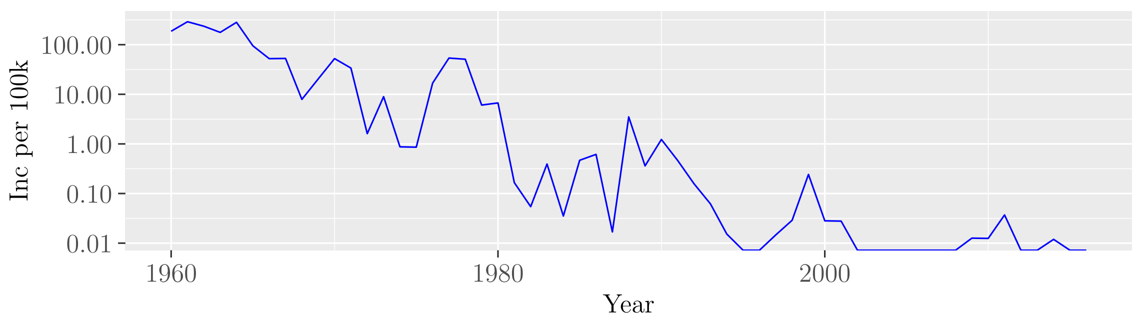
```
summary(distance_states)


##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1829  0.2487  0.2809  0.3034  0.3414  0.5089
```

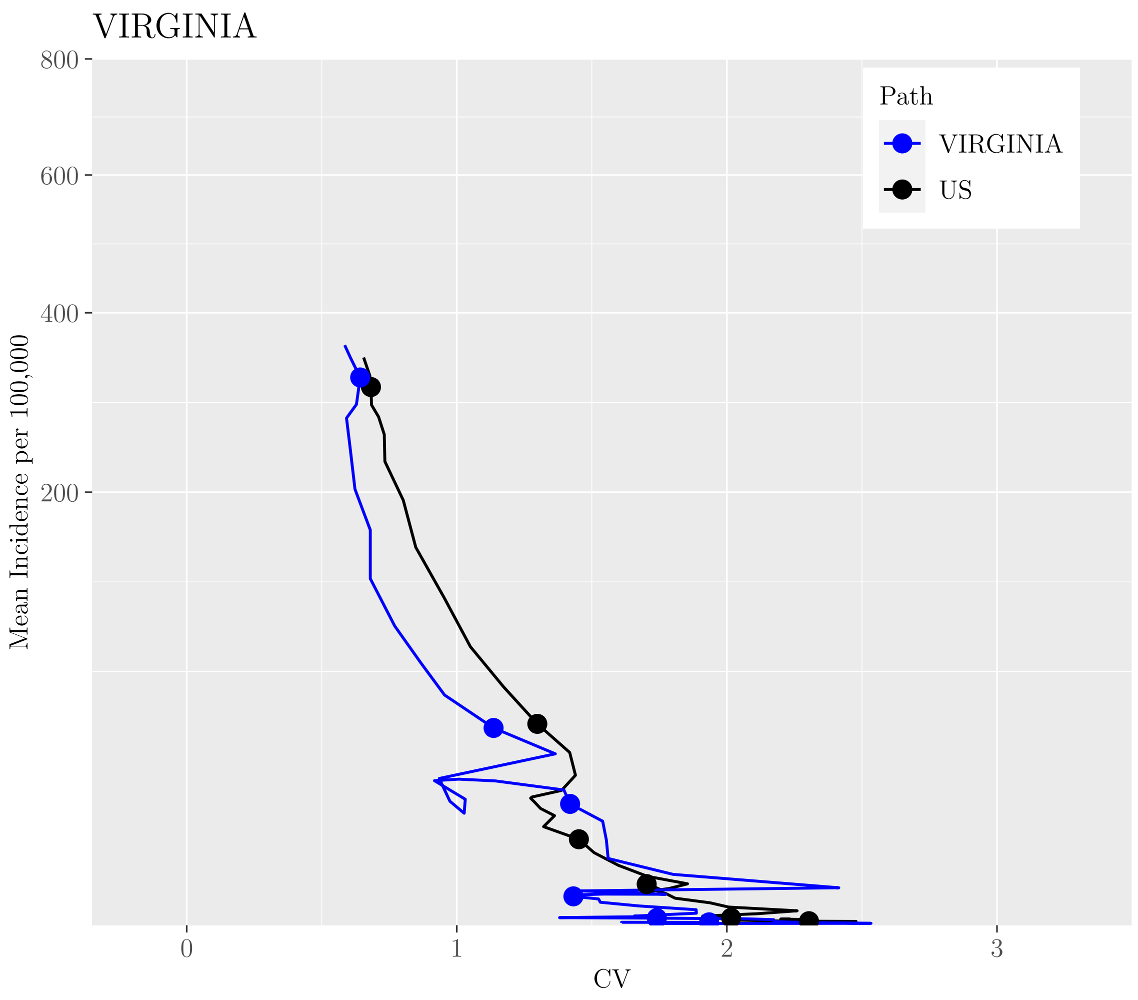
We can also find the states with the maximum and minimum distance using the following code.

```
min_loc <- which.min(distance_states)
max_loc <- which.max(distance_states)
min_distance_state <-
  distance_states[min_loc]
min_distance_state
max_distance_state <-
  distance_states[max_loc]
max_distance_state
```

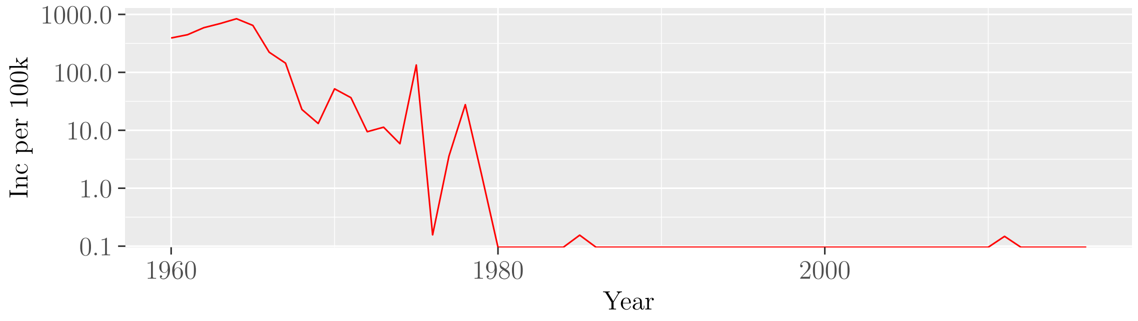
Next, we create plots that compare the states corresponding to the minimum and maximum distances to the US canonical path, with their time series at the top. The following figures show the comparison plots for the minimum and maximum distances respectively, while including a time series plot at the top, as shown in figures 3 and 4.

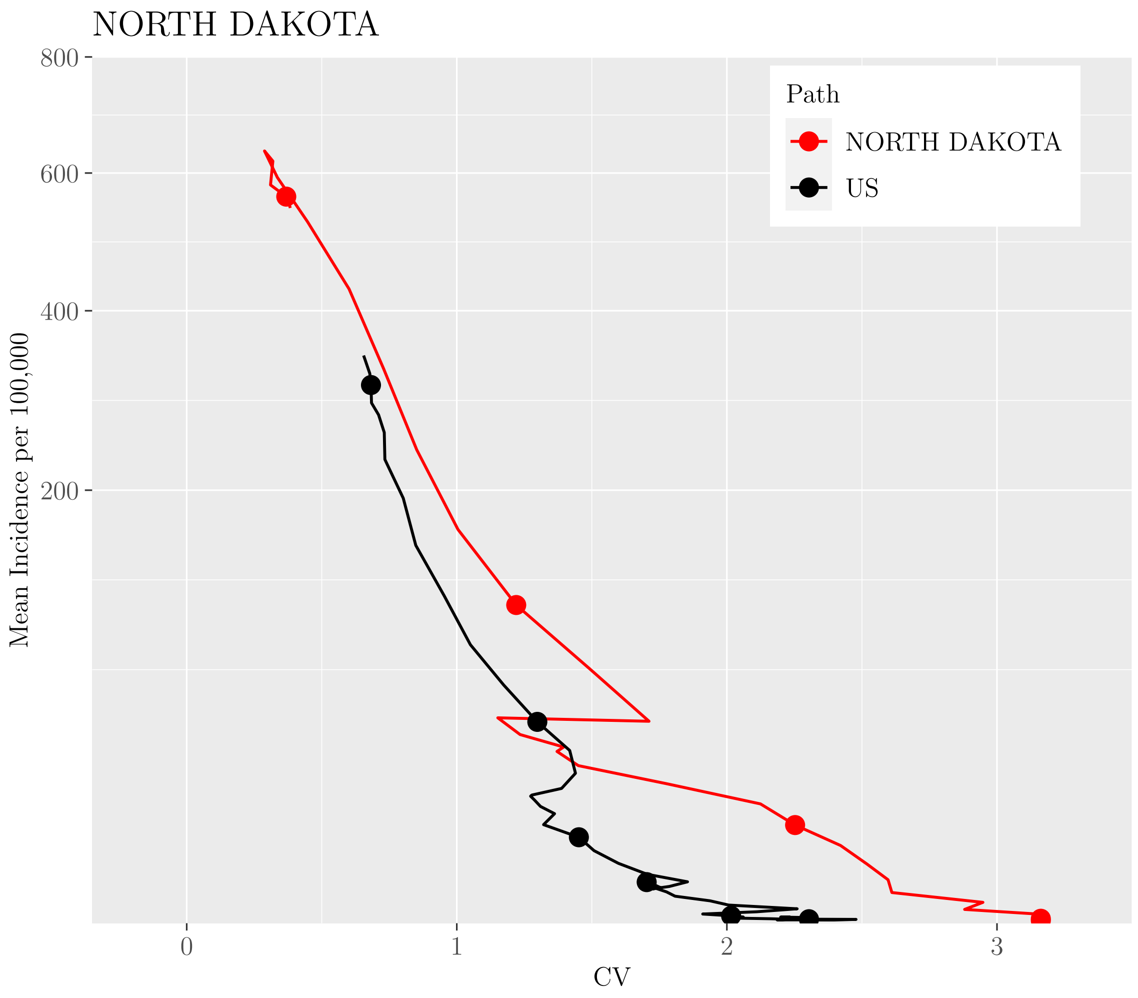
```
grobs_state_min_unif <- list(state_ts_unif_plot[[min_loc]],
                             unif_paths[[min_loc]])
lay_state_min_unif <- rbind(1, 2, 2, 2)
grid.arrange(grobs = grobs_state_min_unif,
             layout_matrix = lay_state_min_unif)
```





```r
grobs_state_max_unif <- list(state_ts_unif_plot[[max_loc]],
                             unif_paths[[max_loc]])
lay_state_max_unif <- rbind(1, 2, 2, 2)
grid.arrange(grobs = grobs_state_max_unif,
             layout_matrix = lay_state_max_unif)
```





7.3.5 Assigning Each US State its Closest Point on the US Path

To see how US states progress along the path over time, we created a figure that plots the closest point on the canonical path for each US state for the following years: 1962, 1972, ..., 2012. We begin by creating empty lists to contain the data and plots, as well as define the vector of years we wish to plot. Additionally, we define vectors that provide the colours of the points of each US state, based on whether the state is a Republican or Democratic Stronghold, or a swing state.

```
## Vector of years to plot closest point
closest_years <- seq(1962, 2012, 10)
## Empty lists to contain closest point data and paths
closest_points_list <-
  as.list(vector(length = length(closest_years)))
closest_points_paths_red <-
  as.list(vector(length = length(closest_years)))
closest_points_paths_blue <-
  as.list(vector(length = length(closest_years)))
closest_points_paths_swing <-
  as.list(vector(length = length(closest_years)))
names(closest_points_list) <- closest_years
names(closest_points_paths_red) <- closest_years
names(closest_points_paths_blue) <- closest_years
names(closest_points_paths_swing) <- closest_years

## Create named vectors that will be used to
## identify the colours of the plotted points
colours_cp <- c("red", "blue", "purple4")
names(colours_cp) <- c("Republican", "Democratic", "Swing")
```

To find the closest point on the path for each state in a given year, we create a function called `closest_point`. The function takes the following arguments: `full_path`, `compare`, `type_full`, `type_compare`, and `year`. `full_path` is a data frame containing the weighted incidence, CV, and years of the US canonical path. `compare` is a list containing the data frames containing the weighted incidence, CV, and years of each US state. In these data frames, the years are numeric. `type_full` and `type_compare` tell the function if the weighted incidence is labelled "Incidence" or "Reports". `year` is the year (numeric) for which we wish to find the closest point of each state on the US path. The function finds the euclidean distance between each state in a given year and all points in the US path. It finds the point corresponding to the minimum distance, and saves the data in the corresponding row of the US path data frame.

```
## Finds the closest point on the path for all states in a given year
closest_point <- function(full_path, compare,
                          type_full, type_compare, year) {
  stopifnot(type_full %in% c("Incidence", "Reports"),
            type_compare %in% c("Incidence",
                                "Reports"),
            is.list(compare))
  require(dplyr)
  closest_point_list <- compare
  full_path_distance <- full_path
  ## For each state, filter the data to only include
  ## a point for the given year
  for (i in 1:length(compare)) {
    filtered <- compare[[i]] %>%
      filter(Year == year)

    ## Find euclidean distance between the state in a
    ## given year to each year of data in the original path
    full_path_distance[,type_full] <-
      (full_path[,type_full] - filtered[,type_compare])^2
    full_path_distance$CV <-
      (full_path$CV - filtered$CV)^2
    full_path_distance$euc_dist <-
      sqrt(full_path_distance[,type_full] +
             full_path_distance$CV)

    closest_point_list[[i]] <-
      full_path[full_path$Year == full_path_distance$Year[
        which.min(full_path_distance$euc_dist)],]
  }
  names(closest_point_list) <- names(compare)

  return(closest_point = closest_point_list)
}
```

With the function created, we run a for loop across the years we wish to create a plot of, that creates a plot of the US canonical path with the closest points of each US state plotted. Within the for loop, we begin by calling the `closest_point` function. We then change the format of the output, resulting in a single data frame containing the closest point of each state in the given year. Next, we create a "Stronghold" column that identifies whether the state is a Republican or Democratic stronghold, or a swing state. Finally, we create separate plots for each type of Stronghold and assign them as an item in a list. Additionally, we save the data frame containing the closest point data to a separate list.

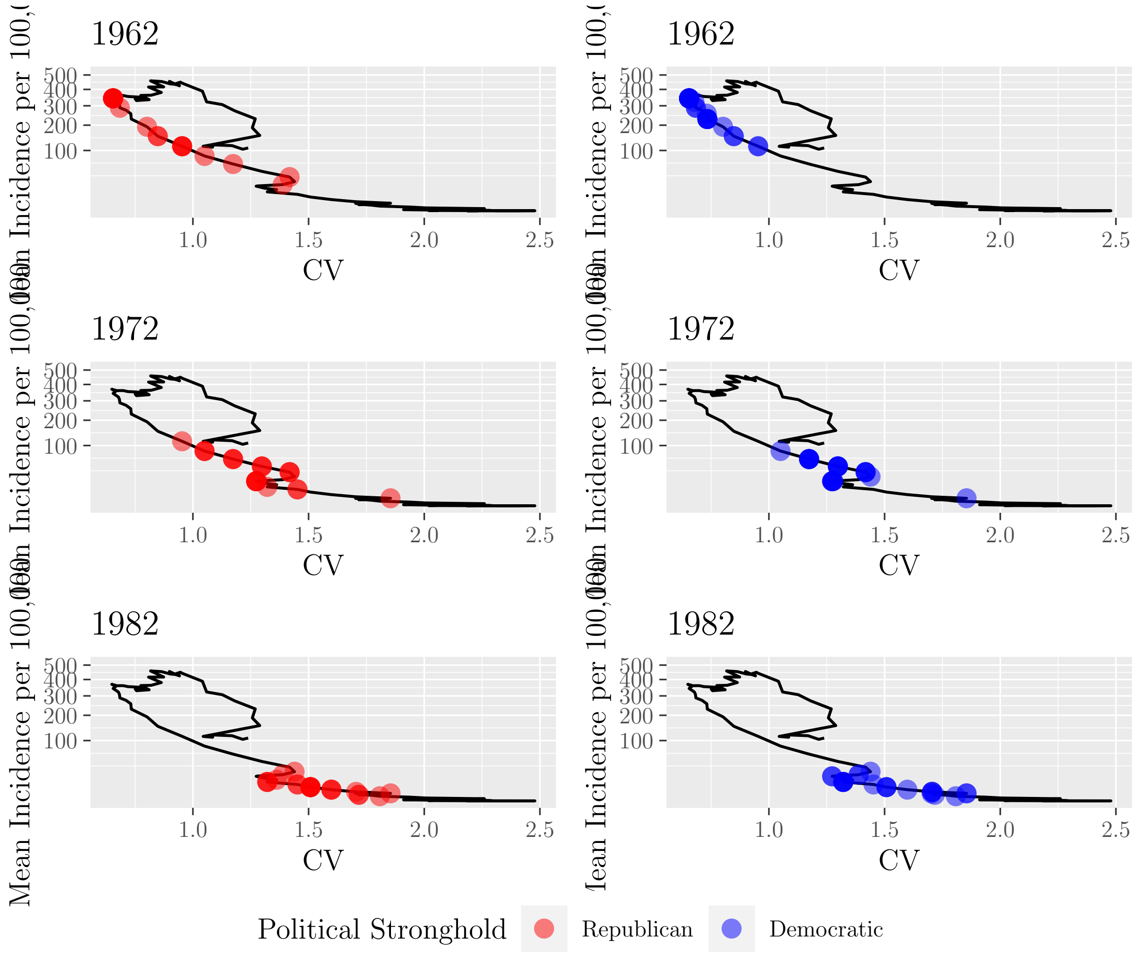
```
## For each year, plot the closest point of each state
## that year on the US path
for (yr in closest_years) {
  ## Call closest_point function
  closest_result <- closest_point(scale_real_data,
                                  scaled_state_data_list,
                                  "Incidence", "Incidence",
                                  yr)

  ## Create long data frame
  closest_result_long <- bind_rows(closest_result, .id = "State")

  ## Add a label column to identify each point and the year
  ## corresponding to the closest point on the path
  closest_result_long <- closest_result_long %>%
    mutate(Label = paste(State, paste("(", Year, ")",
                                      sep = ""),
                         sep = " "))

  ## Find regular weighted incidence (was log), un-scaled
  closest_result_long$Incidence <-
    exp(closest_result_long$Incidence*scale_factor_reports_state)
  ## Find un-scaled CV
  closest_result_long$CV <-
    closest_result_long$CV*scale_factor_cv_state

  ## Filter the data frame to contain only the columns we want
  col_keep <- c("State", "CV", "Incidence", "Year", "Label")
  col_index <- which(colnames(closest_result_long) %in% col_keep)

  closest_result_long <- closest_result_long[,col_index]

  ## Create a column for political stronghold
  ## (using vectors defined in canonical_path.Rnw)
  for (i in 1:nrow(closest_result_long)) {
    if (closest_result_long$State[i] %in% red_states) {
      closest_result_long$Stronghold[i] <- "Republican"
    } else if (closest_result_long$State[i] %in% blue_states) {
      closest_result_long$Stronghold[i] <- "Democratic"
    } else {
      closest_result_long$Stronghold[i] <- "Swing"
    }
```

```
}

## Current year index
yr_index <- which(closest_years == yr)

closest_result_long_red <- closest_result_long %>%
  filter(Stronghold == "Republican")
closest_result_long_blue <- closest_result_long %>%
  filter(Stronghold == "Democratic")
closest_result_long_swing <- closest_result_long %>%
  filter(Stronghold == "Swing")

## Create plot
closest_points_paths_red[[yr_index]] <- ggplot() +
  geom_path(data = data_us, aes(CV, Incidence),
            colour = 'black', lty = 1, lwd = 1) +
  geom_point(data = closest_result_long_red,
             aes(CV, Incidence, colour = Stronghold),
             alpha = 0.5, size = 3) +
  scale_colour_manual(values = colours_cp[c("Republican",
                                            "Democratic")]) +
  scale_y_sqrt("Mean Incidence per 100,000",
             limits = c(min(data_us$Incidence),
                        510)) +
  scale_x_continuous("CV",
                     limits = c(min(data_us$CV),
                                max(data_us$CV))) +
  labs(colour = "Political Stronghold") +
  ggtitle(paste(as.character(yr), sep = " "))

closest_points_paths_blue[[yr_index]] <- ggplot() +
  geom_path(data = data_us, aes(CV, Incidence),
            colour = 'black', lty = 1, lwd = 1) +
  geom_point(data = closest_result_long_blue,
             aes(CV, Incidence, colour = Stronghold),
             alpha = 0.5, size = 3) +
  scale_colour_manual(values = colours_cp[c("Republican",
                                            "Democratic")]) +
  scale_y_sqrt("Mean Incidence per 100,000",
             limits = c(min(data_us$Incidence),
                        510)) +
  scale_x_continuous("CV",
```

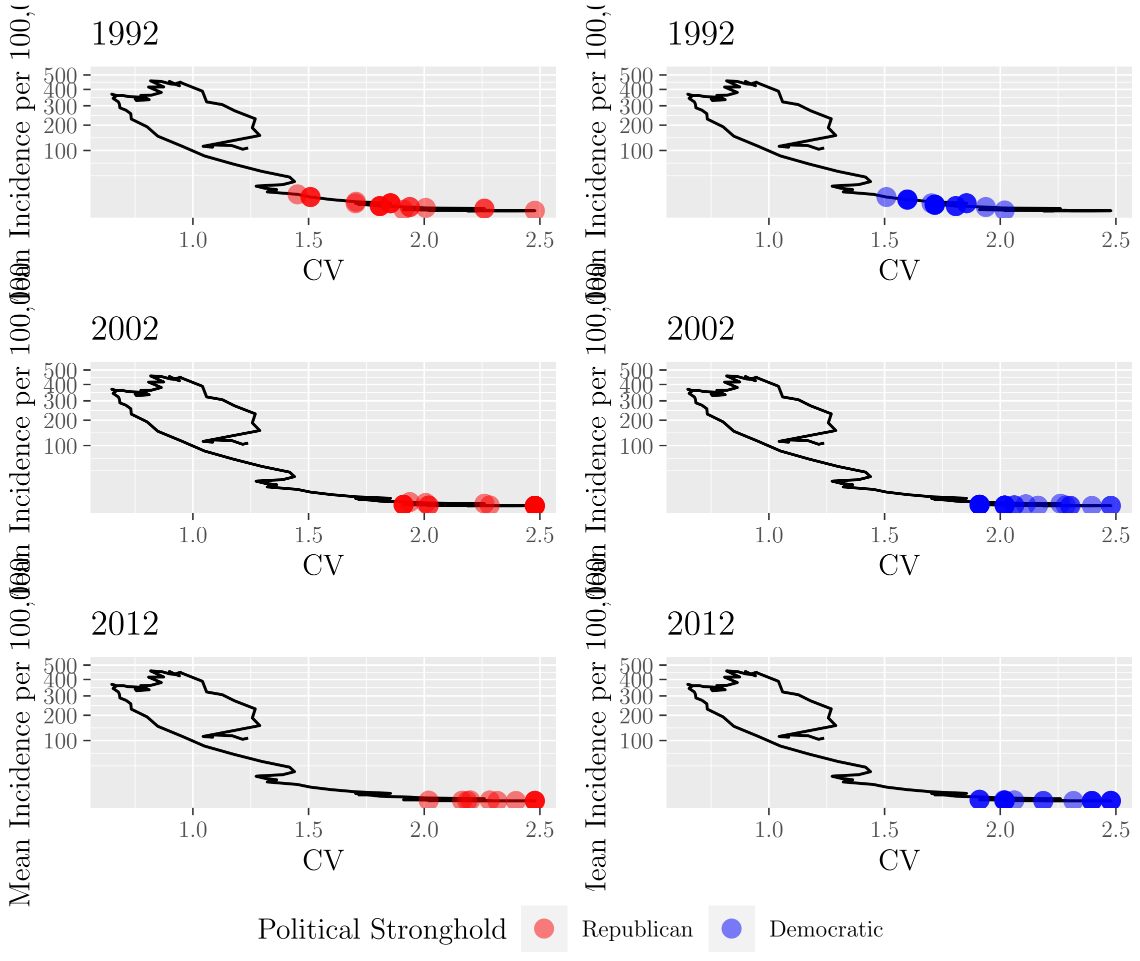
```r
                        limits = c(min(data_us$CV),
                                   max(data_us$CV))) +
    labs(colour = "Political Stronghold") +
    ggtitle(paste(as.character(yr), sep = " "))

  closest_points_paths_swing[[yr_index]] <- ggplot() +
    geom_path(data = data_us, aes(CV, Incidence),
              colour = 'black', lty = 1, lwd = 1) +
    geom_point(data = closest_result_long_swing,
               aes(CV, Incidence, colour = Stronghold),
               alpha = 0.5, size = 3) +
    scale_colour_manual(values = colours_cp[c("Swing")]) +
    scale_y_sqrt("Mean Incidence per 100,000",
               limits = c(min(data_us$Incidence),
                          510)) +
    scale_x_continuous("CV",
                        limits = c(min(data_us$CV),
                                   max(data_us$CV))) +
    labs(colour = "Political Stronghold") +
    ggtitle(paste(as.character(yr), sep = " "))

  ## Save data to a list
  closest_points_list[[yr_index]] <- closest_result_long
}
red_blue_paths <- append(closest_points_paths_red,
                         closest_points_paths_blue)
```

We generate the figure presented in figures 5 and 6 using the following code.

```r
## Create a grid with a single common legend
ggarrange(plotlist = red_blue_paths[c(1, 7, 2, 8, 3, 9)],
          ncol=2, nrow=3, common.legend = TRUE,
          legend="bottom")
```



```r
## Create a grid with a single common legend
ggarrange(plotlist = red_blue_paths[c(4, 10, 5, 11, 6, 12)],
          ncol=2, nrow=3, common.legend = TRUE,
          legend="bottom")
```



We additionally create a similar figure, but for only swing states using the following code.

```r
## Create a grid with a single common legend
ggarrange(plotlist = closest_points_paths_swing,
          ncol=2, nrow=3, common.legend = TRUE,
          legend="bottom")
```

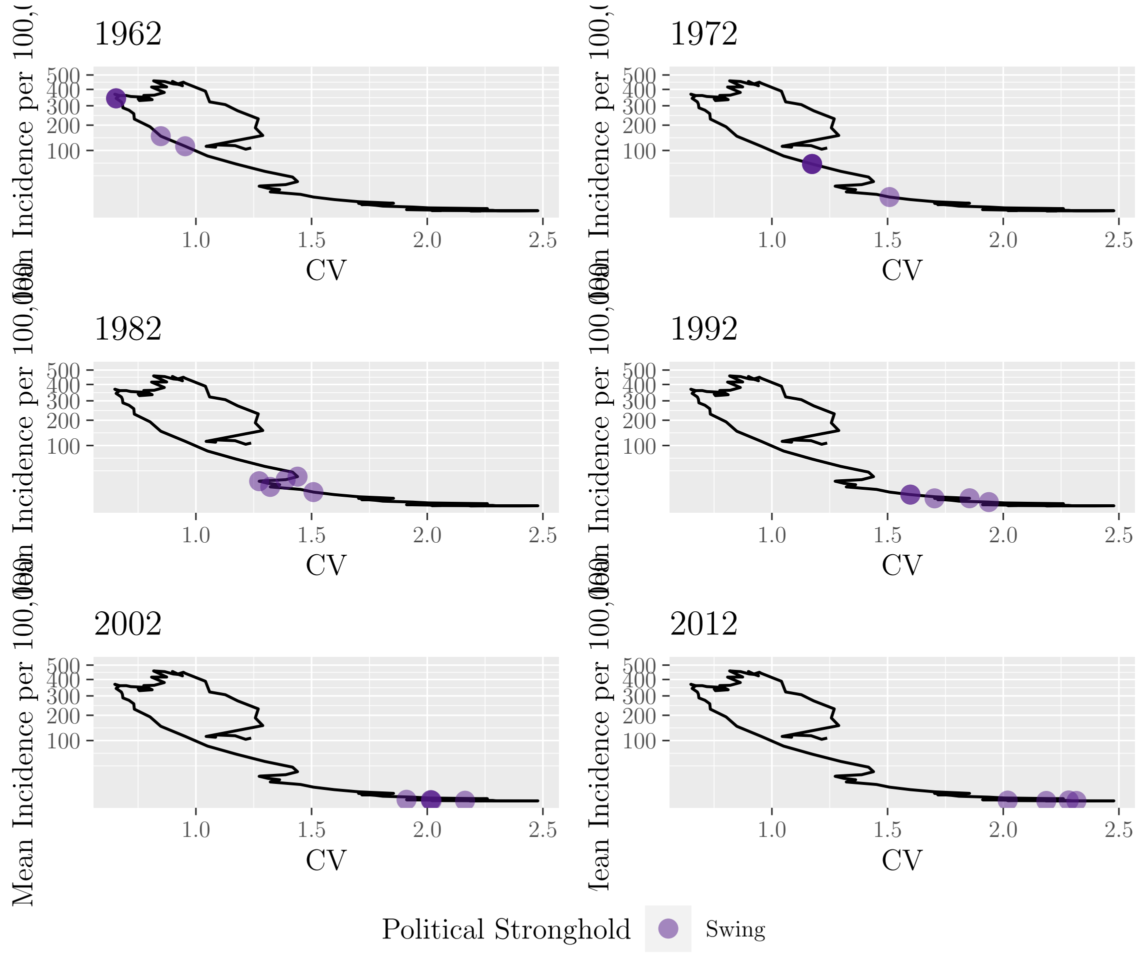


Figure 29: A grid that demonstrates the progression swing states along the path from 1962 to 2012.

7.3.6 Box plot over Time of Pointwise Distances between States and the Aggregate US Path

To find the pointwise distance between the states and the US path from 1960 to 2016, we begin by defining the data frame that will contain the following data: Year, State, and Distance.

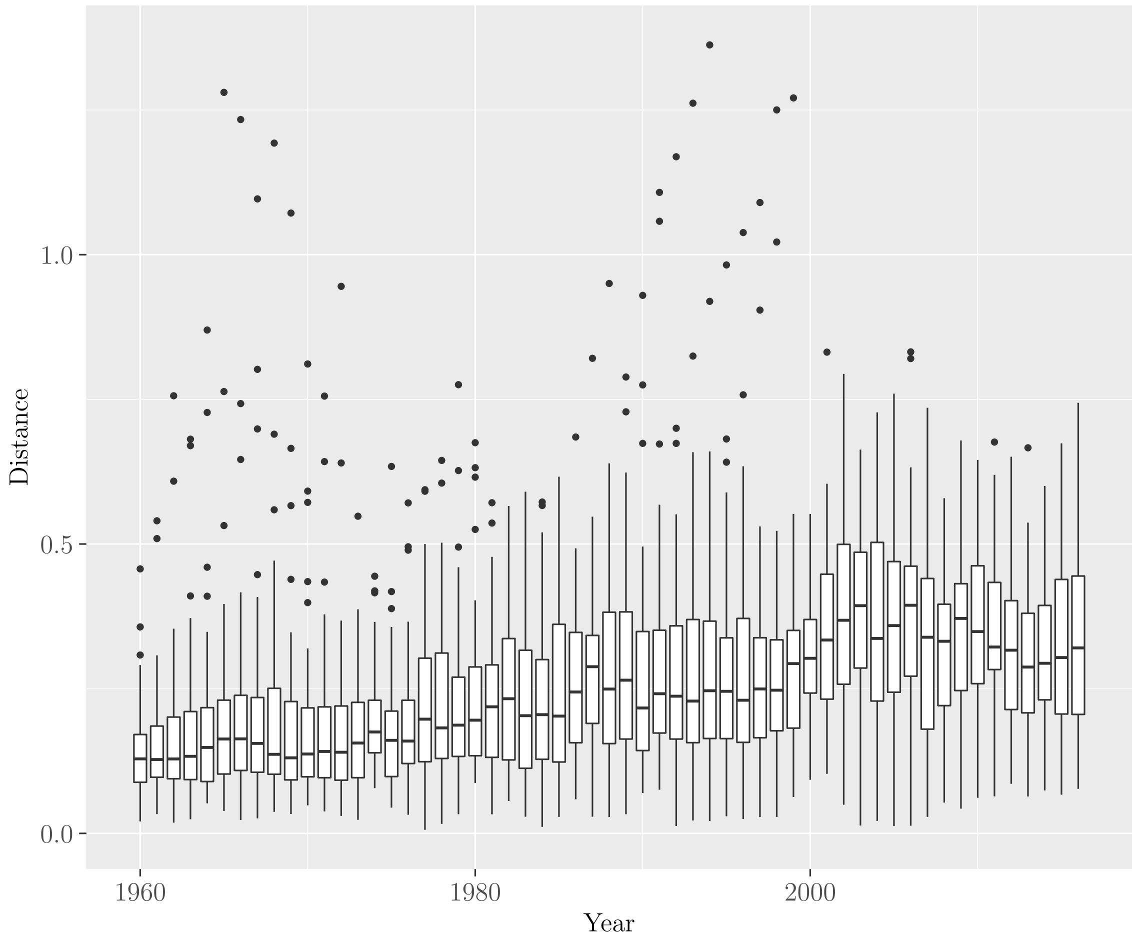
```
## Create the empty data frame
pointwise_distance_results <- data.frame(
  Year = rep(0, length(data_years)*length(states)),
  State = rep(0, length(data_years)*length(states)),
  Distance = rep(0, length(data_years)*length(states))
)

## Set Year equal to 1921 to 2016 repeated for the number of states
pointwise_distance_results$Year <- rep(data_years, length(states))

## Order the data frame by Year
pointwise_distance_results <-
  pointwise_distance_results[order(pointwise_distance_results$Year),]

## Set State equal to the state names repeated for the number of years
pointwise_distance_results$State <- rep(states, length(data_years))
```

Next, we use our defined `distance_btw_canon_path()` function to find the distance between each state and the aggregate path for each individual year using the following code. The value for each year and each state is entered into the corresponding row in the data frame.

```
for (yr in 1:length(data_years)) {
  ## Distance function like before, but our data_years argument
  ## is a single year
  distances <-
    distance_btw_canon_path(scale_real_data,
                            scaled_state_data_list,
                            "Incidence",
                            "Incidence", c(data_years[yr]))
  pointwise_distance_results$Distance[
    which(pointwise_distance_results$Year == data_years[yr])] <-
    as.vector(distances)
}
```

The following is the figure presented in figure 7.

```
pointwise_dist_year <- ggplot() +
  geom_boxplot(data = pointwise_distance_results,
               aes(x= Year, y = Distance, group = Year),
               outlier.size = 0.75) +
  theme(
    legend.text = element_text(size = text_size),
    legend.title = element_text(size = text_size),
    axis.text = element_text(size = text_size),
    axis.title = element_text(size = text_size)
  )
pointwise_dist_year
```



These results are then saved to a *.RData* file for use later on.

```
save(state_paths, min_distance_state,
     max_distance_state,
     distance_states, states_data,
     pointwise_distance_results,
     pointwise_dist_year, red_blue_paths,
     closest_points_paths_blue,
     closest_points_paths_red,
     closest_points_paths_swing,
     closest_points_list,
     scale_real_data,
     scaled_state_data_list,
     scale_factor_cv_state,
     scale_factor_reports_state,
     file = "data/dist_btw_states.RData")
```

7.4 Sensitivity Analysis of the US Canonical Path

The following sections show how the sensitivity analysis is conducted in R while creating figures 8 through 11.

7.4.1 Use of an Un-weighted vs. a Weighted Mean to find the Mean Point in Incidence Space

To determine whether we should weight each state equally, or based on population size when finding the mean point in incidence space, we compare the path generated with equal weighting in section 7.3.2 to one created by weighting each US state based on their population size each year.

We begin by loading in necessary libraries, sourcing `functions.R`, and loading required *.RData* files.

```
library(stats)
library(smoother) #smth.gaussian()
library(tidyverse)
library(RColorBrewer)
library(knitr)
library(ggrepel)
source('functions.R')
load('Good_data/yearly.data.per.100k.RData')
load("Good_data/pop.estimates.RData")
```

Next, we define the arguments required by `cv_inc_fun()` and create a data frame that contains the population weights for each state in each year.

```
## Vector of state names
state_names <- rownames(final.incidence.data)
```

```r
nstates <- length(state_names)
## Vector of the years of data we wish to use in
## calculations (1912 as this is when measles became
## nationally notifiable)
usable_data <- seq(1912, 2016, 1)
## How many years to wait until the first data point
## is calculated (same as Graham et al.)
inc_window <- 10
## Standard deviation of the Gaussian filter (same as
## Graham et al.)
sd <- 3
## Location of the center of the Gaussian filter (2
## years prior, same as Graham et al.)
mean_loc <- 2
## Years for which we want to plot a point and Year
## label along the path
year_labels <- seq(1922, 2012, 10)
## Logical values to tell the function whether we want
## state path data and or confidence intervals
find_states <- TRUE
find_CI <- TRUE
## Use gaussian for incidence and smoother for CV
inc_weighting <- "gaussian"
cv_weighting <- "smoothed"
## Set state weights using population
pop_data <- pop.estimates[, colnames(pop.estimates) %in%
    as.character(usable_data)]

state_weights <- pop_data
for (i in 1:ncol(state_weights)) {
    state_weights[, i] <- pop_data[, i]/sum(pop_data[, i])
}
```

Once the arguments are defined, we call the function and create variables to store the output of the function. These will be used for plotting.

```r
## Call the cv_inc_fun
cv_inc_results <-
  cv_inc_fun(data = final.incidence.data,
             state_names = state_names,
             all_years = usable_data,
             inc_window = inc_window,
             sd = sd,
             mean_loc = mean_loc,
             year_labels = year_labels,
             find_CI = find_CI,
             find_states = find_states,
             inc_weighting = inc_weighting,
             cv_weighting = cv_weighting,
             state_weights = state_weights)

## Save results for use in plotting
data_us_pop_weight <- cv_inc_results$data_us
point_mean_us_pop_weight <- cv_inc_results$point_mean_us
final_data <- cv_inc_results$final_data
CV_states <- cv_inc_results$CV_states
Incidence_states <- cv_inc_results$Incidence_states
CV_states_per_10y <- cv_inc_results$CV_states_per_10y
Incidence_states_per_10y <- cv_inc_results$Incidence_states_per_10y
```

Finally, we create a figure comparing the original path to that created with the states weighted based on population.

```
load("Good_data/true_path_10_3.RData")

comp_true_paths <- true_path +
  geom_path(data = data_us_pop_weight, aes(CV, Incidence),
            colour = 'blue', lty = 1, lwd = 1) +
  geom_point(data = point_mean_us_pop_weight, aes(CV, Incidence),
             colour = 'blue', pch = 15, size = 2) +
  scale_y_sqrt("Mean Incidence per 100,000",
               limits = c(min(data_us_pop_weight$Incidence),
                          max(data_us_pop_weight$Incidence))) +
  scale_x_continuous("CV", limits = c(min(data_us_pop_weight$CV),
                                      max(data_us_pop_weight$CV)))
comp_true_paths
```

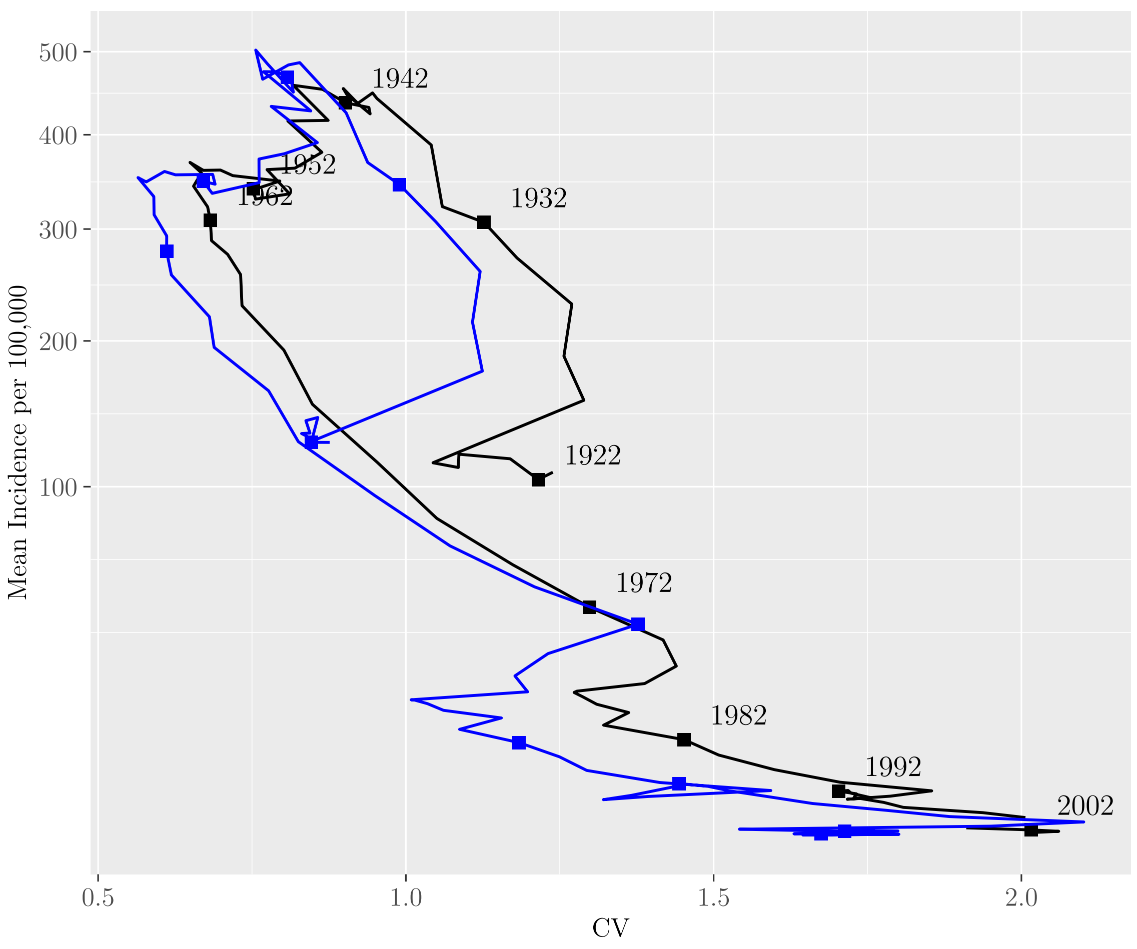


Figure 30: Comparison of using equally weighted states (black) to states weighed by population (blue) to find the mean point in incidence space. From this plot we find that there is very little difference between the two paths (distance: 0.124).

We find the distance between the two paths to quantify the similarity using the following code.

```
## Scale original path data
scaled_orig <- scale_cases_percentile(data_us, "Incidence",
                                      percentile = 0.9,
                                      use_log = TRUE)
scaled_data_us <- scaled_orig$scaled_data
scale_factor_cv <- scaled_orig$scale_value_cv
scale_factor_inc <- scaled_orig$scale_value_case
```

```
## Scale path data where the states are weighted by population
scaled_data_us_pop_weight <- data_us_pop_weight
scaled_data_us_pop_weight$CV <-
  data_us_pop_weight$CV/scale_factor_cv
scaled_data_us_pop_weight$Incidence <-
  log(data_us_pop_weight$Incidence)/scale_factor_inc

colnames(scaled_data_us) <- c("CV", "Incidence", "Upper",
                              "Lower", "Year",
                              "Path", "Reporting")
colnames(scaled_data_us_pop_weight) <- c("CV", "Incidence",
                                         "Upper", "Lower",
                                         "Year")


distance <-
  distance_btw_canon_path(scaled_data_us,
                          list(scaled_data_us_pop_weight),
                          "Incidence", "Incidence",
                          final_data)
distance


## [1] 0.1244686
```

Additionally, we save the results to a *.RData* file.

```
file_name <- paste(paste("Good_data/true_path_pop_weight",
                         inc_window, sd, sep = "_"),
                   ".RData", sep = "")

save(true_path_pop_weight, data_us_pop_weight, final_data, point_mean_us_pop_weigh
```

7.4.2 Method of Weighting Mean Incidence per 100,000

One of the components of the sensitivity analysis was to determine whether we could use a standard moving average as opposed to a Gaussian filter, as chosen by Graham *et al.*, to calculate weighted mean incidence. To determine this, we use the function defined in section 7.3.1 and set `inc_weighting`="moving". First, we load required libraries, source `functions.R` and load pre-created data frames.

```
library(stats)
library(smoother)
library(calibrate)
library(tidyverse)
library(knitr)
source("functions.R")
load("Good_data/yearly.data.per.100k.RData")
load("Good_data/alldata.RData")
load("Good_data/true_path_10_3.RData")
load("data/presentation_logical.RData")
```

Next, we define the arguments required by the `cv_inc_fun()`.

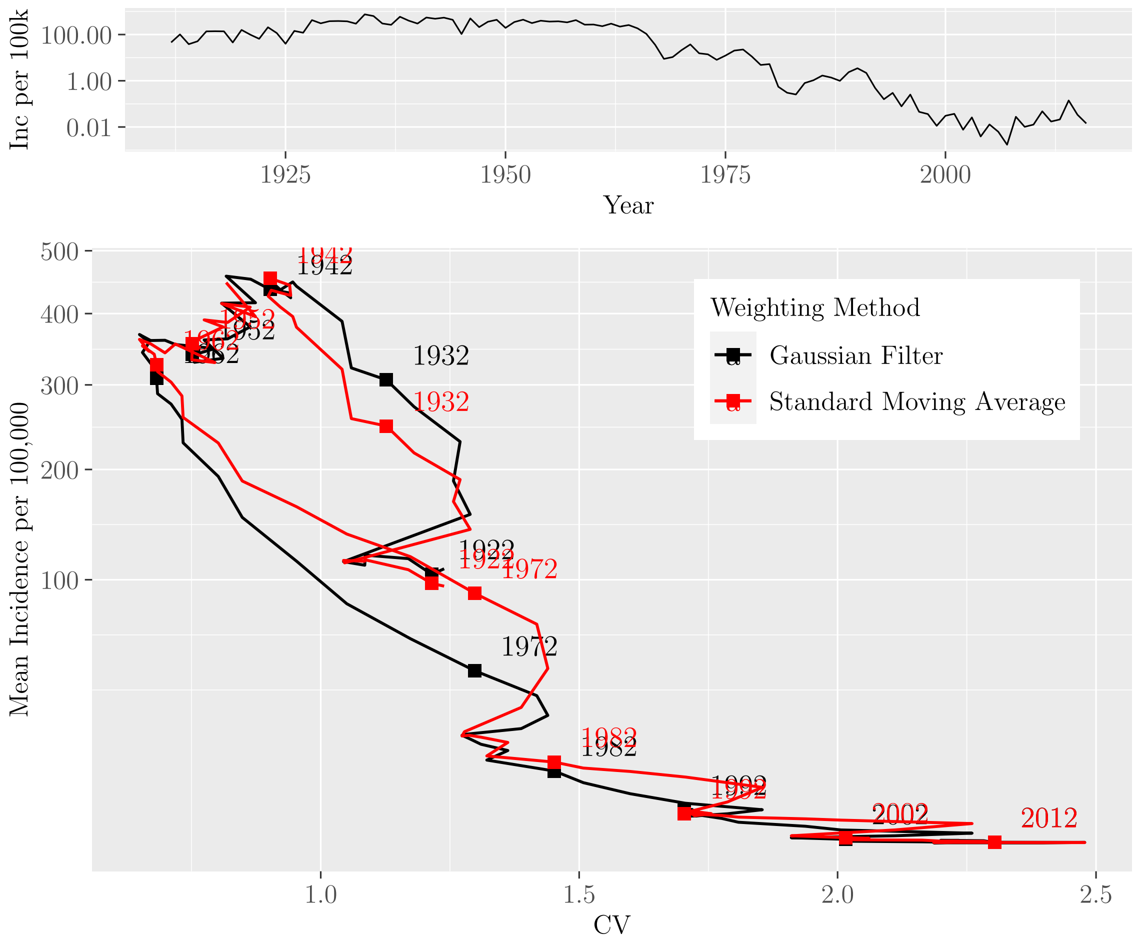
```
## Vector of state names
state_names <- rownames(final.incidence.data)
nstates <- length(state_names)
## Vector of the years of data we wish to use in
## calculations (1912 as this is when measles became
## nationally notifiable)
usable_data <- seq(1912, 2016, 1)
## How many years to wait until the first data point
## is calculated (same as Graham et al.)
inc_window <- 10
## Standard deviation of the Gaussian filter (same as
## Graham et al.)
sd <- 3
## Location of the center of the Gaussian filter (2
## years prior, same as Graham et al.)
mean_loc <- 2
## Years for which we want to plot a point and Year
## label along the path
year_labels <- seq(1922, 2012, 10)
## Logical values to tell the function whether we want
## state path data and or confidence intervals
find_states <- FALSE
find_CI <- FALSE
## Use moving average for incidence and smoother for
## CV
inc_weighting <- "moving"
cv_weighting <- "smoothed"
```

Finally, we call the function and save the output to create plots.

```
## Call the cv_inc_fun
cv_inc_results <- cv_inc_fun(data = final.incidence.data,
                             state_names = state_names,
                             all_years = usable_data,
                             inc_window = inc_window,
                             sd = sd,
                             mean_loc = mean_loc,
                             year_labels = year_labels,
                             find_CI = find_CI,
                             find_states = find_states,
                             inc_weighting = inc_weighting,
                             cv_weighting = cv_weighting)

## Save results for use in plotting
data_us_moving <- cv_inc_results$data_us
point_mean_us_moving <- cv_inc_results$point_mean_us
final_data <- cv_inc_results$final_data
```

Using the created data frames and those that contain the data for the path generated using a Gaussian filter for incidence, we create the following figure (Figure 8).

```
## Set colours manually
colours <- c("Gaussian Filter" = "black",
             "Standard Moving Average" = "red")

path_compare <- ggplot() +
  geom_path(data = data_us,
            aes(CV, Incidence, colour = "Gaussian Filter"),
            lty = 1, lwd = 1) +
  geom_point(data = point_mean_us,
             aes(CV, Incidence, colour = "Gaussian Filter"),
             pch = 15, size = 2) +
  geom_text(data = point_mean_us,
            aes(label = year,'', colour = "Gaussian Filter"),
            position = position_nudge(x = -1, y = -1),
            hjust = -0.45, vjust = -0.75,
            x = point_mean_us$CV,
            y = sqrt(point_mean_us$Incidence)) +
  geom_path(data = data_us_moving,
            aes(CV, Incidence, colour = "Standard Moving Average") ,
            lty = 1, lwd = 1) +
  geom_point(data = point_mean_us_moving,
             aes(CV, Incidence, colour = "Standard Moving Average"),
             pch = 15, size = 2) +
  geom_text(data = point_mean_us_moving,
            aes(label = year,'', colour = "Standard Moving Average"),
            position = position_nudge(x = -1, y = -1),
            hjust = -0.45, vjust = -0.75,
            x = point_mean_us_moving$CV,
            y = sqrt(point_mean_us_moving$Incidence)) +
  scale_y_sqrt("Mean Incidence per 100,000",
               limits = c(min(data_us$Incidence),
                          max(data_us$Incidence))) +
  scale_x_continuous("CV",
                     limits = c(min(data_us$CV),
                                max(data_us$CV))) +
  scale_color_manual(values = colours) +
  labs(colour = "Weighting Method") +
  theme(legend.position = c(.95, .95),
        legend.justification = c("right", "top"),
        legend.box.just = "right",
        legend.margin = margin(6, 6, 6, 6),
        legend.text = element_text(size = text_size),
        legend.title = element_text(size = text_size),
        axis.text = element_text(size = text_size),
        axis.title = element_text(size = text_size)
        )

ma_grobs <- list(us_ts, path_compare)
ma_lay <- rbind(1,2,2,2)
grid.arrange(grobs = ma_grobs, layout_matrix = ma_lay)
```



The results are saved to a *.RData* file in the following code.

```
file_name <- paste(paste("Good_data/true_path_moving_avg",
                         inc_window, sep = "_"),
                   ".RData", sep = "")

save(path_compare, data_us_moving,
     data_us, point_mean_us,
     point_mean_us_moving, final_data,
     file = file_name)
```

7.4.3 Changing the Shape of the Gaussian Filter

Another component of the sensitivity analysis investigated how changing the shape of the Gaussian filter, that is used to calculate weighted incidence, changes the path. We began by changing the location of its center. First, we load required libraries, source `functions.R` and load pre-created data frames.

```r
library(stats)
library(smoother)
library(calibrate)
library(tidyverse)
library(rlist)
library(knitr)
source('functions.R')
load('Good_data/yearly.data.per.100k.RData')
load('Good_data/alldata.RData')
load("data/presentation_logical.RData")
```

Next, we define the arguments required by the `cv_inc_fun()`.

```r
## Vector of state names
state_names <- rownames(final.incidence.data)
nstates <- length(state_names)
## Vector of the years of data we wish to use in
## calculations (1912 as this is when measles became
## nationally notifiable)
usable_data <- seq(1912, 2016, 1)
## How many years to wait until the first data point
## is calculated (same as Graham et al.)
inc_window <- 10
## Standard deviation of the Gaussian filter (same as
## Graham et al.)
sd <- 3
## Range of Location of the center of the Gaussian
## filter
mean_loc <- c(0:5)
## Years for which we want to plot a point and Year
## label along the path
year_labels <- seq(1922, 2012, 10)
## Logical values to tell the function whether we want
## state path data and or confidence intervals
```

```
find_states <- FALSE
find_CI <- FALSE
## Use gaussian for incidence and smoother for CV
inc_weighting <- "gaussian"
cv_weighting <- "smoothed"
```

Finally, we call the function for each value of the location of the center and save the output to create plots.

```
## Create empty lists that will contain the output of the cv_inc_fun()
## function for each mean location value
us_dataframes <- as.list(vector(length = length(mean_loc)))
point_us_dataframes <- as.list(vector(length = length(mean_loc)))
names(us_dataframes) <- as.character(mean_loc)
names(point_us_dataframes) <- as.character(mean_loc)

for (mean_loc_val in mean_loc) {
  ## Call the cv_inc_fun
  cv_inc_results <-
    cv_inc_fun(data = final.incidence.data,
               state_names = state_names,
               all_years = usable_data,
               inc_window = inc_window,
               sd = sd,
               mean_loc = mean_loc_val,
               year_labels = year_labels,
               find_CI = find_CI,
               find_states = find_states,
               inc_weighting = inc_weighting,
               cv_weighting = cv_weighting)

  ## Save results for use in plotting
  mean_ind <- which(mean_loc == mean_loc_val)
  us_dataframes[[mean_ind]] <-
    cv_inc_results$data_us
  point_us_dataframes[[mean_ind]] <-
    cv_inc_results$point_mean_us

  us_dataframes[[mean_ind]]$Mean_location <-
    factor(mean_loc_val, levels = as.character(mean_loc_val))
  point_us_dataframes[[mean_ind]]$Mean_location <-
```

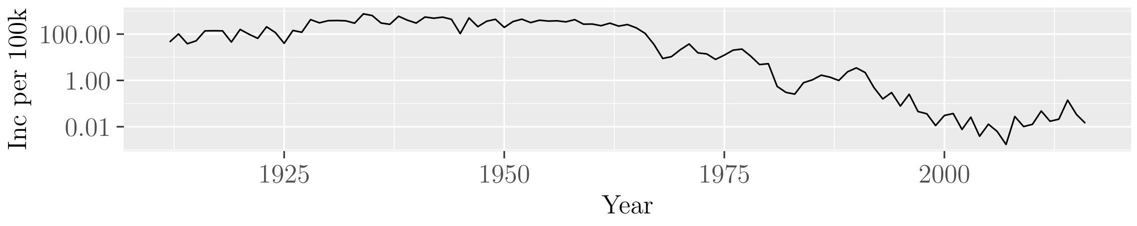
```
    factor(mean_loc_val, levels = as.character(mean_loc_val))
  final_data <- cv_inc_results$final_data
}
```

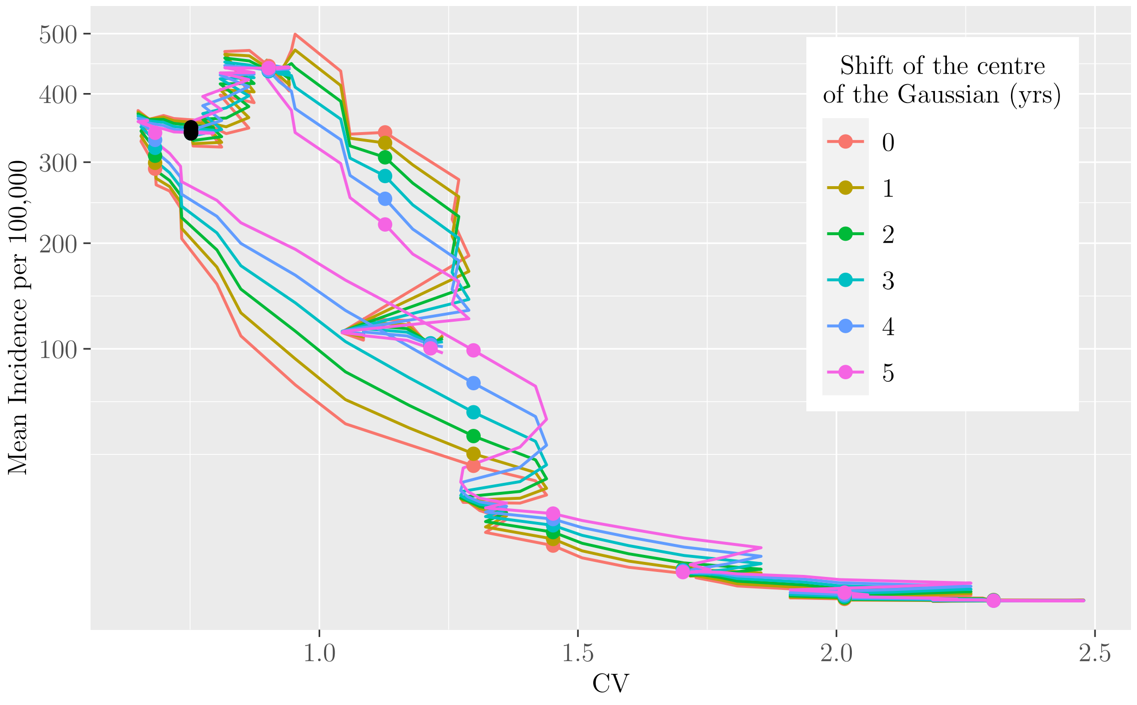
Since the function has created a list containing the data frames for each center, we bind each list element by row to get a single data frame with data for each value of the center.

```
## Bind the list items by row to ave one data frame that contains
## data for each mean location value
us_data <- list.rbind(us_dataframes)
point_us_data <- list.rbind(point_us_dataframes)
only_1952 <- point_us_data %>% filter(year == "1952")
```

Next, we create the figure shown in figure 9.

```
compare_mean_location <- ggplot()+
  geom_path(data = us_data,
            aes(CV, Incidence, colour = Mean_location),
            lty = 1, lwd = 1) +
  geom_point(data = point_us_data,
             aes(CV, Incidence, colour = Mean_location),
             size = 2) +
  geom_point(data = only_1952,
             aes(CV, Incidence),
             size = 2,
             colour = "black") +
  scale_y_sqrt("Mean Incidence per 100,000",
               limits = c(min(us_data$Incidence),
                          max(us_data$Incidence))) +
  scale_x_continuous("CV",
                     limits = c(min(us_data$CV),
                                max(us_data$CV))) +
  labs(colour = "Shift of the centre \n of the Gaussian (yrs)") +
  guides(colour = guide_legend(title.hjust = 0.5)) +
  theme(
        legend.position = c(.95, .95),
        legend.justification = c("right", "top"),
        legend.box.just = "center",
        legend.margin = margin(6, 6, 6, 6),
        legend.title.align = 0.5,
        legend.direction = "vertical",
        legend.text = element_text(size = text_size),
        legend.title = element_text(size = text_size),
        axis.text = element_text(size = text_size),
        axis.title = element_text(size = text_size)
  )

load('Good_data/true_path_10_3.RData')
mean_loc_grobs <- list(us_ts, compare_mean_location)
mean_loc_lay <- rbind(1,2,2,2)
grid.arrange(grobs = mean_loc_grobs, layout_matrix = mean_loc_lay)
```





The results are saved to a *.RData* file in the following code.

```r
save(compare_mean_location, us_data, us_dataframes,
     file = "data/compare_mean_location_real.RData")
```

We also investigated how changing the standard deviation of the Gaussian filter changes the trajectory of the path. We use the same methods as implemented when changing the location of the center. First, we load required libraries, source `functions.R` and load pre-created data frames.

```r
library(stats)
library(smoother)
library(calibrate)
library(tidyverse)
library(rlist)
library(knitr)
source('functions.R')
load('Good_data/yearly.data.per.100k.RData')
load('Good_data/alldata.RData')
load("data/presentation_logical.RData")
```

Next, we define the arguments required by the `cv_inc_fun()`.

```r
## Vector of state names
state_names <- rownames(final.incidence.data)
nstates <- length(state_names)
## Vector of the years of data we wish to use in
## calculations (1912 as this is when measles became
## nationally notifiable)
usable_data <- seq(1912, 2016, 1)
## How many years to wait until the first data point
## is calculated (same as Graham et al.)
inc_window <- 10
## Standard deviation of the Gaussian filter (same as
## Graham et al.)
sd <- c(1:10)
## Location of the center of the Gaussian filter (2
## years prior, same as Graham et al.)
mean_loc <- 2
## Years for which we want to plot a point and Year
## label along the path
year_labels <- seq(1922, 2012, 10)
## Logical values to tell the function whether we want
## state path data and or confidence intervals
find_states <- FALSE
find_CI <- FALSE
## Use gaussian for incidence and smoother for CV
inc_weighting <- "gaussian"
cv_weighting <- "smoothed"
```

Finally, we call the function for each value of the standard deviation and save the output to create plots.

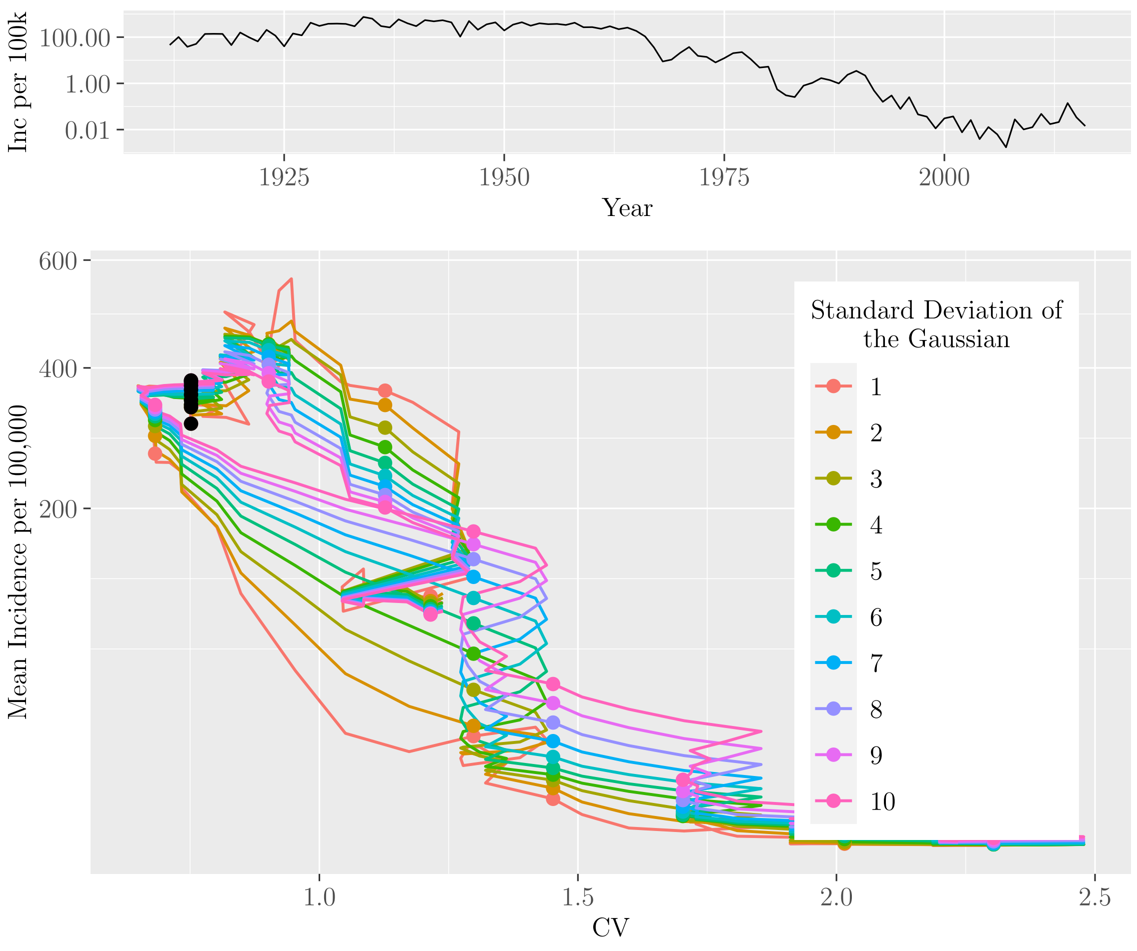
```r
## Create empty lists that will contain the output of the
## cv_inc_fun() function for each sd value
us_dataframes <- as.list(vector(length = length(sd)))
point_us_dataframes <- as.list(vector(length = length(sd)))
names(us_dataframes) <- as.character(sd)
names(point_us_dataframes) <- as.character(sd)

for (sd_val in sd) {
  ## Call the cv_inc_fun
  cv_inc_results <- cv_inc_fun(data = final.incidence.data,
                               state_names = state_names,
                               all_years = usable_data,
                               inc_window = inc_window,
                               sd = sd_val, mean_loc = mean_loc,
                               year_labels = year_labels,
                               find_CI = find_CI,
                               find_states = find_states,
                               inc_weighting = inc_weighting,
                               cv_weighting = cv_weighting)

  ## Save results for use in plotting
  us_dataframes[[which(sd == sd_val)]] <-
    cv_inc_results$data_us
  point_us_dataframes[[which(sd == sd_val)]] <-
    cv_inc_results$point_mean_us

  us_dataframes[[which(sd == sd_val)]]$Standard_deviation <-
    factor(sd_val, levels = as.character(sd_val))
  point_us_dataframes[[which(sd == sd_val)]]$Standard_deviation <-
    factor(sd_val, levels = as.character(sd_val))
  final_data <- cv_inc_results$final_data
}
```

Since the function has created a list containing the data frames for each standard deviation, we bind each list element by row to get a single data frame with data for each standard deviation value.

```
## Bind the list items by row to save one data frame that
## contains data for each standard deviation value
us_data_sd <- list.rbind(us_dataframes)
point_us_data_sd <- list.rbind(point_us_dataframes)
only_1952 <- point_us_data_sd %>% filter(year == "1952")
```

Next, we create the figure shown in figure 10.

```
compare_standard_deviation <- ggplot()+
  geom_path(data = us_data_sd,
            aes(CV, Incidence, colour = Standard_deviation),
            lty = 1, lwd = 1) +
  geom_point(data = point_us_data_sd,
             aes(CV, Incidence, colour = Standard_deviation),
             size = 2) +
  geom_point(data = only_1952,
             aes(CV, Incidence),
             size = 2,
             colour = "black") +
  scale_y_sqrt("Mean Incidence per 100,000",
               limits = c(min(us_data_sd$Incidence),
                          max(us_data_sd$Incidence))) +
  scale_x_continuous("CV",
                     limits = c(min(us_data_sd$CV),
                                max(us_data_sd$CV))) +
  labs(colour = "Standard Deviation of \n the Gaussian") +
  guides(colour = guide_legend(title.hjust = 0.5)) +
  theme(
        legend.position = c(.95, .95),
        legend.justification = c("right", "top"),
        legend.box.just = "center",
        legend.margin = margin(6, 6, 6, 6),
        legend.title.align = 0.5,
        legend.direction = "vertical",
        legend.text = element_text(size = text_size),
        legend.title = element_text(size = text_size),
        axis.text = element_text(size = text_size),
        axis.title = element_text(size = text_size)
  )
load('Good_data/true_path_10_3.RData')

sd_grobs <- list(us_ts, compare_standard_deviation)
sd_lay <- rbind(1,2,2,2)
grid.arrange(grobs = sd_grobs, layout_matrix = sd_lay)
```



The results are saved to a *.RData* file in the following code.

```r
save(compare_standard_deviation, us_data_sd, us_dataframes,
     point_us_data_sd,
     file = "data/compare_standard_deviation_real.RData")
```

7.4.4 Method of Weighting the Coefficient of Variation

The last component of the sensitivity analysis investigated how the choice of weighting for CV changes the canonical path. First, we load required libraries, source `functions.R` and load pre-created data frames.

```r
library(stats)
library(smoother)
library(calibrate)
library(tidyverse)
library(knitr)
library(ggrepel)
library(gridExtra)
source("functions.R")

load('Good_data/yearly.data.per.100k.RData')
load('Good_data/alldata.RData')
load('Good_data/true_path_10_3.RData')
load("data/presentation_logical.RData")
```

Next, we define the arguments required by the `cv_inc_fun()`. We set `cv_weighting` to `"gaussian"`, so that the function knows to use a Gaussian filter to find the CV.

```r
## Vector of state names
state_names <- rownames(final.incidence.data)
nstates <- length(state_names)
## Vector of the years of data we wish to use in
## calculations (1912 as this is when measles became
## nationally notifiable)
usable_data <- seq(1912, 2016, 1)
## How many years to wait until the first data point
## is calculated (same as Graham et al.)
inc_window <- 10
## Standard deviation of the Gaussian filter (same as
## Graham et al.)
```

```
sd <- 3
## Location of the center of the Gaussian filter (2
## years prior, same as Graham et al.)
mean_loc <- 2
## Years for which we want to plot a point and Year
## label along the path
year_labels <- seq(1922, 2012, 10)
## Logical values to tell the function whether we want
## state path data and or confidence intervals
find_states <- FALSE
find_CI <- FALSE
## Use moving average for incidence and smoother for
## CV
inc_weighting <- "gaussian"
cv_weighting <- "gaussian"
```

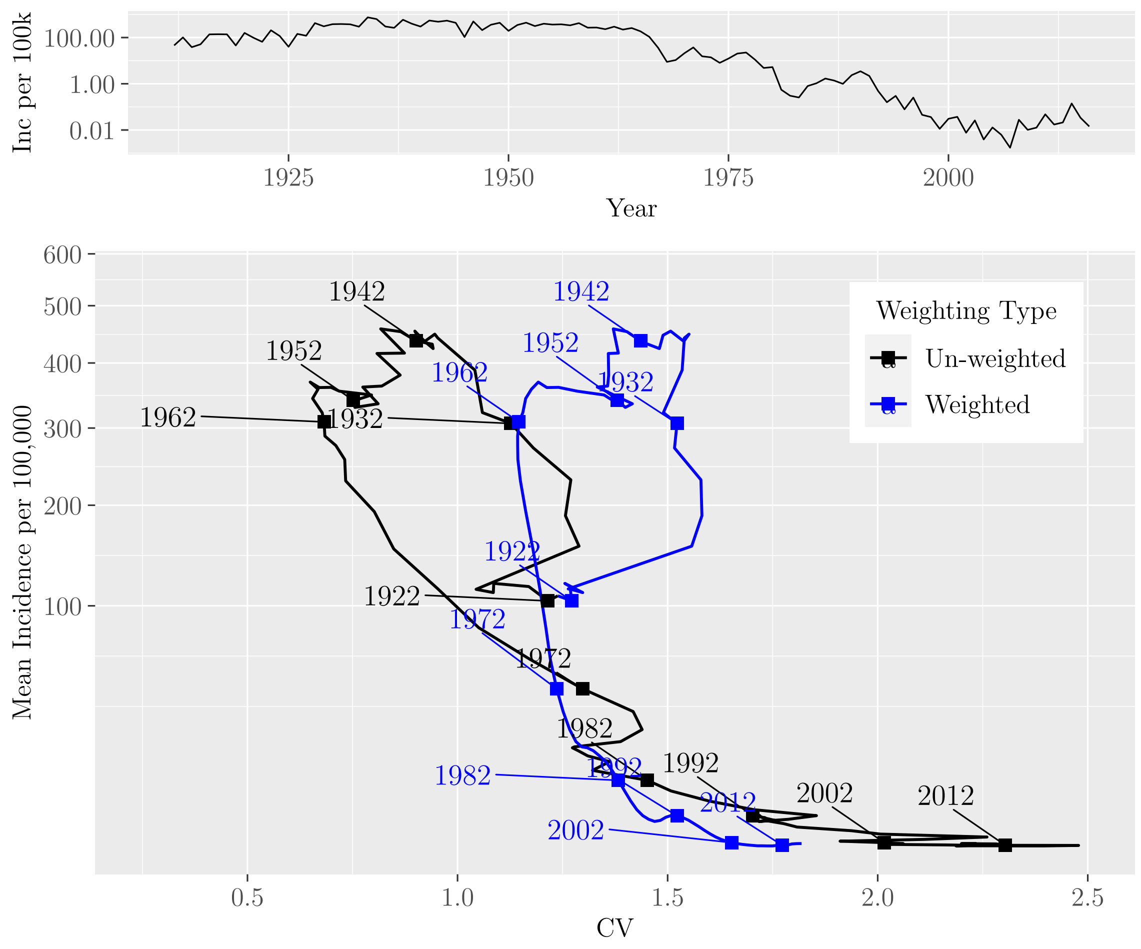
Finally, we call the function and save the output to create the plot.

```
## Call the cv_inc_fun
cv_inc_results <- cv_inc_fun(data = final.incidence.data,
                             state_names = state_names,
                             all_years = usable_data,
                             inc_window = inc_window,
                             sd = sd, mean_loc = mean_loc,
                             year_labels = year_labels,
                             find_CI = find_CI,
                             find_states = find_states,
                             inc_weighting = inc_weighting,
                             cv_weighting = cv_weighting)

## Save results for use in plotting
data_us_gaussian <- cv_inc_results$data_us
point_mean_us_gaussian <- cv_inc_results$point_mean_us
final_data <- cv_inc_results$final_data

## Create a Path column that identifies
## that it is the original path
data_us$Path <- "Un-weighted"
point_mean_us$Path <- "Un-weighted"
data_us_gaussian$Path <- "Weighted"
point_mean_us_gaussian$Path <- "Weighted"
```

```
## Filter the data to only containing the
## necessary columns and rename the columns
data_us_filter <-
  data_us[colnames(data_us) %in% c("CV", "Incidence",
                                   "year", "Path")]
point_mean_us_filter <-
  point_mean_us[colnames(point_mean_us) %in% c("CV",
                                               "Incidence",
                                               "year",
                                               "Path")]

all_data <- rbind(data_us_filter, data_us_gaussian)
all_point_data <- rbind(point_mean_us_filter, point_mean_us_gaussian)
```

Next, we create the figure shown in figure 11.

```
## Set colours manually
colours <- c("black", "blue")
names(colours) <- c(unique(data_us$Path),
                    unique(data_us_gaussian$Path))

compare_cv_weighting <- ggplot() +
  geom_path(data = all_data,
            aes(CV, Incidence, colour = Path),
            lty = 1, lwd = 1) +
  geom_point(data = all_point_data,
             aes(CV, Incidence, colour = Path),
             pch = 15, size = 2) +
  geom_text_repel(data = all_point_data,
            aes(label = year, '', colour = Path),
            position = position_nudge(x = -1, y = -1),
            hjust = -0.45, vjust = -0.75,
            x = all_point_data$CV,
            y = sqrt(all_point_data$Incidence)) +
  scale_y_sqrt("Mean Incidence per 100,000",
               limits = c(min(all_data$Incidence),
                          550)) +
  scale_x_continuous("CV", limits = c(0.25,
                                       2.5))+
  scale_colour_manual(values = colours) +
  labs(colour = "Weighting Type") +
  theme(
    legend.position = c(.95, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "center",
    legend.margin = margin(6, 6, 6, 6),
    legend.title.align = 0.5,
    legend.direction = "vertical",
    legend.text = element_text(size = text_size),
    legend.title = element_text(size = text_size),
    axis.text = element_text(size = text_size),
    axis.title = element_text(size = text_size)
  )

comp_cv_grobs <- list(us_ts, compare_cv_weighting)
comp_cv_lay <- rbind(1,2,2,2)
grid.arrange(grobs = comp_cv_grobs, layout_matrix = comp_cv_lay)
```



The results are saved to a *.RData* file in the following code.

```
file_name <- paste(paste("Good_data/true_path",
                         inc_window, sd, "gaussian_test",
                         sep = "_"), ".RData", sep = "")

save(data_us_gaussian, final_data, point_mean_us_gaussian,
     true_path, data_us, point_mean_us,
     compare_cv_weighting, file = file_name)
```

7.4.5 Distance Between the Original Canonical Path and those Created Through Sensitivity Analysis

In sections 7.4.2 and 7.4.4, we compared the original aggregate US path to the path generated using a different weighting method for either incidence or CV. We can use the distance function, as described in section 7.3.4.1, to quantify how far apart the paths with different weighting methods are from the original aggregate US path. First, we load in the required libraries, source `functions.R`, and load the necessary *.RData* files.

```r
library(tidyverse)
library(rlist)
library(knitr)
source("functions.R")
load("Good_data/true_path_10_3.RData")
load("Good_data/true_path_moving_avg_10.RData")
load("Good_data/true_path_10_3_gaussian_test.RData")
```

Next, since we are comparing each of the paths generated with difference weighting methods to the original path, we use the scaling function, as described in section 7.3.4.1, to scale the original path data and generate the scaling factors for use on the other data.

```r
## Prepare the original canonical path for scaling
real_data <- data_us
use_log <- TRUE
real_data$Year <- real_data$year

## Column names need to be of the form "CV",
## "Incidence", "Reports" and "Year"
real_data <- real_data[, c("CV", "Incidence", "Year")]

## Scale the data using the 90th percentile
## using self defined function
scale_real_data_values <-
  scale_cases_percentile(real_data, "Incidence",
                         0.9, use_log)

## Extract the scaling factors and scaled data
scale_factor_reports <- scale_real_data_values$scale_value_case
```

```
scale_factor_cv <- scale_real_data_values$scale_value_cv

scale_real_data <- scale_real_data_values$scaled_data
```

With the original data scaled, and scaling factors created, we first scale the data where incidence was calculated using a standard moving average and find its distance from the original path.

```
## Prepare the moving average canonical path for scaling
moving_average_data <- data_us_moving
colnames(moving_average_data) <- c("CV", "Incidence", "Year")

scaled_moving_average_data <- moving_average_data

## Scale the moving average path data
## using the factors used by the original path
scaled_moving_average_data$Incidence <-
  log(moving_average_data$Incidence)/scale_factor_reports
scaled_moving_average_data$CV <-
  moving_average_data$CV/scale_factor_cv

## Turn the scaled data into a list for
## the distance_btw_canon_path function
scaled_moving_average_data_list <-
  list(scaled_moving_average_data)

## Find the distance between the original
## path and the moving average path
dist_moving_average <-
  distance_btw_canon_path(scale_real_data,
                          scaled_moving_average_data_list,
                          "Incidence",
                          "Incidence",
                          final_data)

dist_moving_average


## [1] 0.05852666
```

Finally, we scale the data where CV is calculated using a Gaussian filter and find its distance from the original path.

```r
## Prepare the gaussian CV canonical path for scaling
colnames(data_us_gaussian) <- c("CV", "Incidence", "Year", "Path")

scale_gaussian_cv_data <- data_us_gaussian

## Scale the gaussian CV path data using the
## factors used by the original path
scale_gaussian_cv_data$Incidence <-
  log(scale_gaussian_cv_data$Incidence)/scale_factor_reports
scale_gaussian_cv_data$CV <-
  scale_gaussian_cv_data$CV/scale_factor_cv

## Turn the scaled data into a list for the
## distance_btw_canon_path function
scale_gaussian_cv_list <- list(scale_gaussian_cv_data)

## Find the distance between the original
## path and the gaussian CV path
dist_gaussian_cv <-
  distance_btw_canon_path(scale_real_data,
                          scale_gaussian_cv_list,
                          "Incidence",
                          "Incidence",
                          data_years = final_data)

dist_gaussian_cv

## [1] 0.188065
```

7.5 Simulating the Canonical Path

7.5.1 Building the Model to Simulate the US Path

After completing the sensitivity analysis, the next goal was to determine what model structures were required to simulate the aggregate US canonical path. In their paper, Graham *et al.* [1] chose to use a discrete-time age-structured model to simulate incidence. This model uses a model matrix at each time step to define a transition from every epidemiological stage-age combination to the other possible epidemiological stage-age combination. The epidemiological stages used in this model include:

maternally immune, susceptible, infected, recovered, and vaccinated. They used reported data for age distribution, vital dynamics, and vaccination coverage data, and measles epidemiologic parameters (e.g., basic reproduction number and age-contact rates).

To determine which of the model components are necessary to simulate the aggregate US path, we started with the base SEIR model, as described in section 4.1. We built the model up, adding a single new structure at a time, until we achieved a model capable of simulating the aggregate US path. We ended up with a model that had the following time-varying parameters: transmission (β), vaccination coverage (p), and vital dynamics (μ and ν). We also found that the base SEIR compartments (susceptible, exposed, infected, and recovered) were sufficient for our purposes. Additionally, to mimic the way we create the aggregate US path, i.e., taking the mean across each US state, we decided to treat each state as its own simulation based on population and vital statistics data. Finally, since the population of some states were below the extinction threshold of 1 million, we needed to implement immigration into the model. The following sections will describe the functions used to implement these components into the model, and explain the code used to simulate the model.

7.5.1.1 Time-varying Transmission Rate β

We implement time-varying transmission by assuming that measles seasonality is sinusoidally forced with a peak in January and a period of one year. Thus, we defined a function that calculates the transmission rate using a cosine function in the following code.

```
### Time-varying transmission rate beta
beta_fun <- function(t, alpha, beta0, one_year) {
  beta_val <- beta0*(1 + alpha*cos(2*pi*t/one_year))
  return(
    beta_val
  )
}
```

In `beta_fun`, `alpha` is the amplitude of seasonal forcing that is between 0 and 1, and `beta0` is found using the expression for the basic reproduction number $\mathcal{R}_0$in the following code.

```
### beta0 function calculated using the expression for R0
beta0_fun <- function(params) {
  with(as.list(params), {
    beta0 <- R0*mu*(sigma + mu)*(gamma + mu)/(nu*sigma)
    return(
      beta0
    )
  })
}
```

In `beta0_fun`, `R0` is the basic reproduction number, `mu` is the death rate, `nu` is the birth rate, `sigma` is the reciprocal of the mean latent period, and `gamma` is the reciprocal of the mean infectious period.

7.5.1.2 Time-varying Vital Statistics

To best represent the US as a whole, we used state level data for births and deaths, as described in section 7.2.4, to implement time-varying vital statistics into the model. We created a function called, `birth_death_rate_fun` that takes the following arguments: `t`, `vital_stats`, and `init_run`. `t` is the time for which we wish to estimate the birth and death rate, `vital_stats` is a data frame containing birth and death rate data for a particular US state with the corresponding columns labeled "Birth Rate" and "Death Rate", and `init_run` is the number of years we are running from the initial conditions. The function outputs the birth and death rate as a list for the given time based on several if statements. If `t` is during the extra run time, the function returns the birth and death rate values corresponding to the first data point. If `t` is not during the extra run time, then the function outputs either the values at that time (if the data is available) or it outputs the values at the closest point in time for which we have data. The function is displayed below.

```
### Vital stats function
birth_death_rate_fun <- function(t, vital_stats, init_run) {
  ## If t is in the extra run time from the initial conditions,
  ## return the rates at the earliest data point
  if (t < init_run + 1) {
    nu_estim <- vital_stats$`Birth Rate`[vital_stats$Time == init_run + 1]
    mu_estim <- vital_stats$`Death Rate`[vital_stats$Time == init_run + 1]
  } else {
    ## If t is not in the extra run time, either return the rates
    ## at the given time (if the data exists) OR return the rates
    ## at the closest time for which data exists
    if (t %in% vital_stats$Time) {
      nu_estim <- vital_stats$`Birth Rate`[vital_stats$Time == t]
      mu_estim <- vital_stats$`Death Rate`[vital_stats$Time == t]
    } else {
      t_round <- vital_stats$Time[which.min(abs(vital_stats$Time-t))]
      nu_estim <- vital_stats$`Birth Rate`[vital_stats$Time == t_round]
      mu_estim <- vital_stats$`Death Rate`[vital_stats$Time == t_round]
    }
  }
  nu_estim <- unname(nu_estim)
  mu_estim <- unname(mu_estim)
  return(
    list(nu = nu_estim, mu = mu_estim)
  )
}
```

7.5.1.3 Time-varying Vaccination Coverage

We implement time-varying vaccination coverage in a similar way as with time-varying vital statistics. We defined a function called `vaccination_fun_test` that takes the following arguments: `t`, `t_final`, `t_first_vacc`, `vaccine_data`, `start_year`, `end_year`, and `vacc_year`. `t` is the time for which we wish to estimate the vaccination coverage, `t_final` is the final time of the simulation for which we need vaccination coverage estimates, `t_first_vacc` is the time at which vaccination begins in the model, `vaccine_data` is a data frame containing vaccination coverage for the entirety of the US as a percent, with column names of the form "YEAR" and "COVERAGE", `start_year` is a numeric value stating the year in which the simulation begins (after running for extra time), `end_year` is the year in which the simulation ends, and `vacc_year` is the year of the first vaccination (i.e., earliest year we could have data on vaccination coverage).

First, the function defines the ranges of years in which we have vaccination coverage data. Next, it determines how many years there are between the year when vaccination begins and the year when the vaccination data begins. The function then adds two new columns, one that provides the time in a way that the `adaptivetau` function can interpret, e.g., instead of the time being in years, it is the number of years elapsed. The second is a column containing the proportion of the population vaccinated. If the "COVERAGE" column is already a proportion, the "Proportion" column is equal to the "COVERAGE" column. If the "COVERAGE" column is a percent, the column is converted to a proportion. Finally, we estimate the vaccination coverage at a given time based on an if-statement. If the time is before the time of the first vaccine, the estimated proportion is 0. If the time is after the first vaccine was invented, but before we have data, the vaccine proportion linearly increases from 0 to the value of the vaccinated proportion at the first data point. Finally, if the time is within the range for which we have data, the function either outputs the the value at that time (if the data is available) or it outputs the values at the closest point in time for which we have data. The function is displayed below.

```r
### Time-varying proportion of the population vaccinated p
vaccination_fun_data <- function(t, t_final, t_first_vacc,
                                 vaccine_data, start_year=1953,
                                 end_year=2019,
                                 vacc_year=1963) {
  start_data_year <- vaccine_data$YEAR[1]
  end_data_year <- vaccine_data$YEAR[length(vaccine_data$YEAR)]

  ## Find the number of years we need to add to t_first_vacc
  ## in order to define time in a way that
  ## adaptivetau can understand
  if (t_final > (end_year - start_year + 1)) {
    add <- abs(start_data_year - vacc_year) + 1
  } else {
    if (start_year == 1963) {
      add <- abs(start_data_year - vacc_year) + 2
    } else{
      add <- abs(start_data_year - vacc_year) + 1
    }

  }

  vaccine_data$Time <- c((t_first_vacc + add):t_final)

  if (all(vaccination_data$COVERAGE >= 0) &
      all(vaccination_data$COVERAGE < 1)) {
    vaccine_data$Proportion <- vaccine_data$COVERAGE
  } else {
   vaccine_data$Proportion <- vaccine_data$COVERAGE/100
  }
  if (t >= 0 & t < t_first_vacc) {
    vacc <- 0
  } else if (t >= t_first_vacc & t < (t_first_vacc + add)) {
    ## Linearly increases to specified final
    ## vaccination coverage
    vacc <- (vaccine_data$Proportion[1]/(add))*t -
```

```
      vaccine_data$Proportion[1]*t_first_vacc/(add)
  } else {
    if (t %in% vaccine_data$Time) {
      vacc <- vaccine_data$Proportion[vaccine_data$Time == t]
    } else {
      t_round <- vaccine_data$Time[which.min(abs(vaccine_data$Time-t))]
      vacc <- vaccine_data$Proportion[vaccine_data$Time == t_round]
    }

  }
  if (vacc > 1 | vacc < 0) {
    stop("Vaccination proportion of", vacc, "at time", t)
  } else {
    return(vacc)
  }
}
```

7.5.1.4 Immigration

To implement immigration, we decided to change the $\frac{\beta SI}{N}$ term instead of adding spatial structure to the model. This choice was in tune with the notion of building model complexity as necessary, while reducing the computing time of the model. We change the term, so that in a population with no infecteds, there is some immigration from outside the model. The new term has the following form: $\frac{\beta S}{N}(I+q)$, where q is the number of infected immigrants. Since as vaccination coverage increases, we will observe less cases due to immigration, we chose to make q a function of the equilibrium prevalence (See equations 5 and 6).

Thus, we first created `eq_prev_fun_data` which calculates the equilibrium points of the model with time-varying vaccination, vital statistics and transmission. The function takes the following arguments: `t`, `params` and `N`. `t` is the time at which we want to calculate the equilibria, `params` is the list of parameters used in the `adaptivetau` function and `N` is the population size at the given time. Using the arguments, the function finds p, μ, ν, and β for the given time using the functions as described in sections 7.5.1.1 to 7.5.1.3. Using these values and the other necessary parameters from `params` the function calculates and outputs the equilibria as a list.

```
## Endemic equilibrium with time varying vaccination (from data),
## time varying beta and time varying vital stats (from data)
eq_prev_fun_data <- function(t, params, N) {
```

```r
  p <- vaccination_fun_data(t, params$t_final,
                            params$t_first_vacc,
                            params$vaccine_data,
                            params$start_year,
                            params$end_year)
  beta <- beta_fun(t, params$alpha, params$beta0,
                   params$one_year)
  nu <- birth_death_rate_fun(t, params$vital_stats,
                             params$init_run)$nu
  mu <- birth_death_rate_fun(t, params$vital_stats,
                             params$init_run)$mu

  with(as.list(params), {
    zeta <- sigma/(gamma + mu)
    R0 <- ((beta0*nu)/((sigma + mu)*mu))*zeta
    pcrit <- (1 - 1/R0)
    if (p < pcrit) {
      S_eqm <- round((N/R0)*(nu/mu))
      E_eqm <- round(N*(nu/(sigma + mu))*((1 - p) - 1/R0))
      I_eqm <- round(E_eqm*zeta)
      R_eqm <- N - S_eqm - E_eqm - I_eqm
      if (E_eqm < 0 || I_eqm < 0) {
        cat("S_eqm =", S_eqm, ", I_eqm =", I_eqm,
            ", E_eqm =", E_eqm, ", R_eqm =", R_eqm,
            ", N =", N, "\n")
        cat("beta =", beta, ", p =", p, ", pcrit = ",
            pcrit, "\n")
        stop("E_eqm < 0 or I_eqm < 0")
      }
    } else if (p >= pcrit) {
      S_eqm <- round(N*(1-p)*(nu/mu))
      E_eqm <- 0
      I_eqm <- 0
      R_eqm <- N - S_eqm - E_eqm - I_eqm
    }
    return(
      list(S_eqm = S_eqm, E_eqm = E_eqm, I_eqm = I_eqm,
           R_eqm = R_eqm)
    )
  })
}
```

With a function that can calculate the equilibrium prevalence at each time, we set q equal to the maximum of the equilibrium prevalence multiplied by some small immigration factor and the small immigration factor alone (i.e., one immigrant multiplied by the immigration factor). As the choice of immigration factor is arbitrary, we optimize this value in section 7.5.3.

7.5.1.5 Estimating the Level of Under-reporting

Since as is, our model simulates incidence, we must implement observation error to account for the fact that not every measles case is reported. The way in which we implement observation error will be discussed in section 7.5.2.2, however for now we estimate the probability of a case being reported. To accomplish this, we compare births shifted by the mean age at infection to reported cases, prior to vaccination, since almost everyone got measles before the vaccine was infected. We begin by loading in necessary *.RData* files and libraries.

```r
load('data/vital_stats_data.RData')
load('Good_data/yearly.data.per.100k.RData')
load("data/vaccination_data.RData")

library(tidyverse)
library(gridExtra)
library(ggplot2)
library(knitr)
```

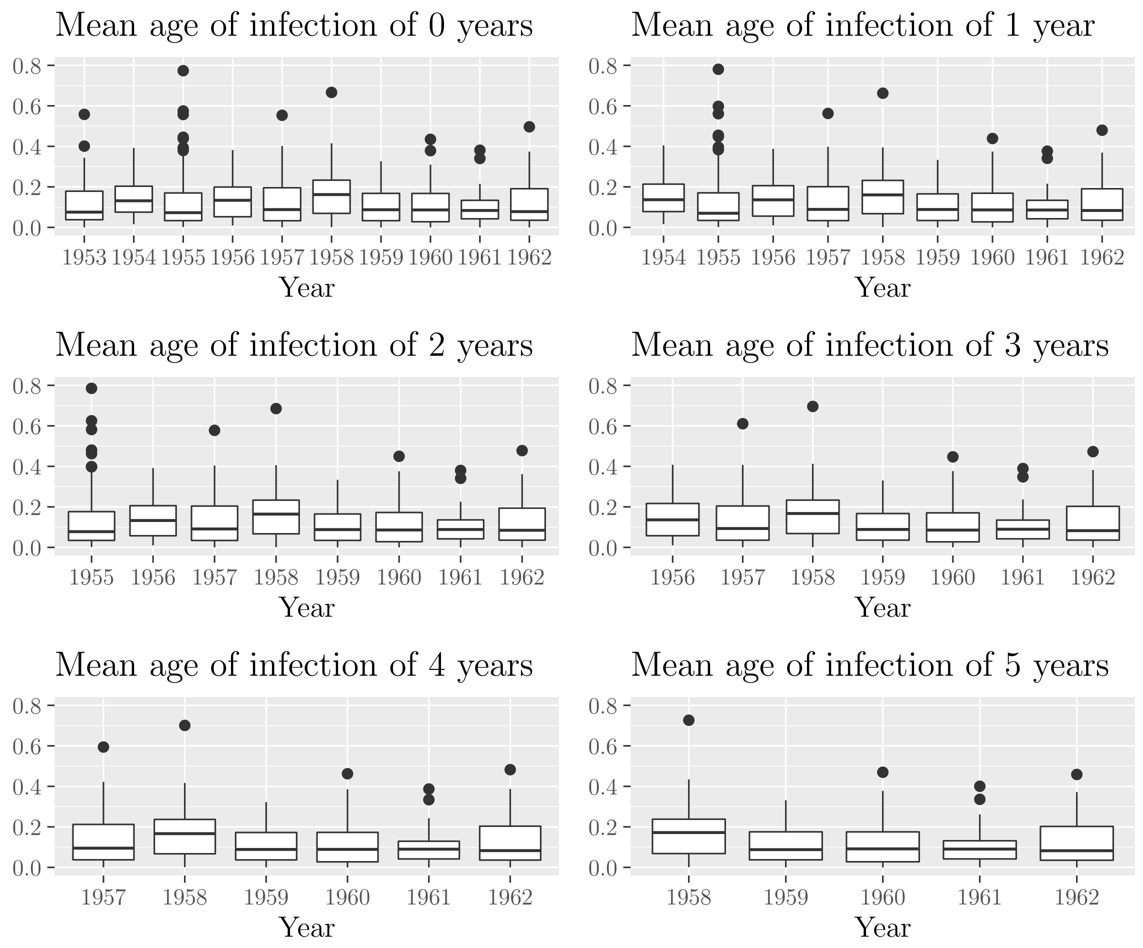
Then, for mean age at infection ranging from 0 to 5 years, we find the ratio of reported cases to births in each state each year. We use this data to create a box-plot of the ratio of reported cases to births over time for each value of the mean age at infection.

```r
## Add a column containing US states
final.incidence.data$State <- rownames(final.incidence.data)

## Convert to long format
final_incidence_data_long <-
  pivot_longer(final.incidence.data,
               cols = !State,
               names_to = "Year",
               values_to = "Incidence_per_100k")

## Vector of varying age of infections that we can shift by
age_of_infection_shift <- c(0:5)
```

```r
vaccination_data$YEAR <- as.numeric(vaccination_data$YEAR)

## Empty list to contain under-reporting plots
plot_list <- as.list(vector(length = length(age_of_infection_shift)))

for(i in 1:length(age_of_infection_shift)) {
  ## Years to consider for incidence based on mean age at infection
  no_vax_years_incidence <-
    c((1953 + age_of_infection_shift[i]):1962)

  ## Filter incidence and population based on no_vax_years_incidence
  final_incidence_data_long_no_vax <- final_incidence_data_long %>%
    filter(Year %in% no_vax_years_incidence)
  vital_stats_no_vax_pop <- vital_stats %>%
    filter(Year %in% no_vax_years_incidence)

  final_incidence_data_long_no_vax <-
    final_incidence_data_long_no_vax[order(
      final_incidence_data_long_no_vax$Year,
      decreasing = FALSE),]

  ## Years to consider for births based on mean age at infection
  no_vax_years_births <-
    c(1953:(1962 - age_of_infection_shift[i]))

  ## Filter vital stats based on no_vax_years_births
  vital_stats_no_vax_births <- vital_stats %>%
    filter(Year %in% no_vax_years_births)

  ## Check that the mean difference is the mean age at infection
  mean_diff_years <-
    mean(no_vax_years_incidence - no_vax_years_births)

  ## Create a data frame containing all data required
  ## to estimate under-reporting
  if(setequal(vital_stats_no_vax_pop$Year,
              final_incidence_data_long_no_vax$Year)
     & setequal(vital_stats_no_vax_pop$State,
                final_incidence_data_long_no_vax$State)
     & mean_diff_years == age_of_infection_shift[i]) {
    reporting_state <- data.frame(
      Year = final_incidence_data_long_no_vax$Year,
      State = final_incidence_data_long_no_vax$State,
      Population = vital_stats_no_vax_pop$Population,
      Births_shift_by_age_infec = vital_stats_no_vax_births$Births,
      Incidence_per_100k =
        final_incidence_data_long_no_vax$Incidence_per_100k
    )
  } else {
    stop("Years", setequal(vital_stats_no_vax_pop$Year,
                           final_incidence_data_long_no_vax$Year),
         "States",
         setequal(vital_stats_no_vax_pop$State,
                  final_incidence_data_long_no_vax$State),
         "Diff years", mean_diff_years)
  }

  ## Add an incidence column that is pure incidence instead of per 100k
  reporting_state <- reporting_state %>%
    mutate(Incidence = Incidence_per_100k*Population/100000)
  ## Add a column for the proportion reported estimate
  reporting_state <- reporting_state %>%
    mutate(reporting = Incidence/Births_shift_by_age_infec)

  ## Create a boxplot of the ratio of reported cases to births over time
  if (age_of_infection_shift[i] == 1) {
    reporting_boxplot_year_shift <- ggplot() +
      geom_boxplot(data = reporting_state,
                   aes(as.character(Year), reporting)) +
      xlab("Year") +
      ylab("Ratio of Reports to Births") +
      ylim(c(0, 0.8)) +
      ggtitle(paste("Mean age of infection of ",
                    age_of_infection_shift[i], " year", sep = "")) +
      theme(axis.title.y = element_blank())
  } else {
    reporting_boxplot_year_shift <- ggplot() +
      geom_boxplot(data = reporting_state,
                   aes(as.character(Year), reporting)) +
      xlab("Year") +
      ylab("Ratio of Reports to Births") +
      ylim(c(0, 0.8)) +
      ggtitle(paste("Mean age of infection of ",
                         age_of_infection_shift[i], " years", sep = "")) +
        theme(axis.title.y = element_blank())
  }

  ## Output the plot to a list
  plot_list[[i]] <- reporting_boxplot_year_shift
}
```

Next, we create the plot displayed in figure 12.

```
layout_matrix <- rbind(c(1,2),
                       c(3,4),
                       c(5,6))

grid.arrange(grobs = plot_list, layout_matrix = layout_matrix)
```



The results are then saved to a *.RData* file for use later on.

```
save(layout_matrix, plot_list, file = "data/underreporting.RData")
```

7.5.2 Creating Parameter Files and Functions Used for Simulation

7.5.2.1 Setting Default Model Arguments

Since we will be simulating model in several different contexts, we begin by creating a file that houses default values of all parameters used in the model. The default parameters are displayed in the code below.

```
## Determines time unit (1 = years, 12 = months, 365 = days...)
one_year <- 1
## Number of years run from the initial conditions without vaccination
init_run <- 0
## Start and end years of the simulation
start_year <- 1953
end_year <- 2019
## Mean infectious and latent periods
t_latent <- 8/365*one_year # 8 days latent
t_infec <- 5/365*one_year # 5 days infectious
gamma <- 1/t_infec
sigma <- 1/t_latent
## Amplitude of seasonal forcing
alpha <- 0.1
## Basic Reproduction Number
R0 <- 17
## Time between saved data
clock_tick <- one_year/12
## Immigration factor
immigration_factor <- 0
## Determine whether immigration is used or not (logical)
use_immigration <- FALSE
## Probability of a case being reported
probability_of_report <- 0.1
## List of the transitions used in the adaptivetau function
transitions_SEIR <- list(
  c(S = +1), #unvaccinated birth
  c(S = -1), #death of susceptible
  c(S = -1, E = +1, C = +1), #exposed
  c(E = -1), #death of exposed
  c(E = -1, I = +1), #become infectious
  c(I = -1), #death of infected
  c(I = -1, R = +1), #recovery
  c(R = +1), #vaccinated birth
  c(R = -1) #death of a recovered
)
```

Since the simulations take at least 30 minutes to run, they must be conducted in separate R scripts. Thus, we use the following code to create an R document containing the default model arguments that can be sourced from other R scripts.

```
library(knitr)
xfun::Rscript_call(purl,
                   list(input='model_parameter_inputs_new.Rnw',
                        output='model_parameter_inputs_new.R'))
```

7.5.2.2 Creating the Function that Simulates Yearly Incidence per 100,000

Once the model arguments are loaded we can simulate the model. We created the function `path_model()` that simulates the model and outputs data frames containing yearly and monthly incidence per 100,000, as well as yearly and monthly reports per 100,000. `path_model` takes the following arguments: `pop`, `vit_stats`, `vacc_data`, `US_states`, `rate_fun`, `transitions`, `start_year`, `end_year`, `use_immigration`, `immigration_factor`, `init_run`, `one_year`, `gamma`, `sigma`, `alpha`, `R0`, `clock_tick`, and `probability_of_report`. `pop` and `vit_stats` are data frames containing yearly population estimates and birth and death rates by state. `vacc_data` is a data frame containing yearly vaccination coverage estimates for the US as a whole. `US_states` is a vector containing the US state names, as they appear in `pop` and `vit_stats`. `rate_fun` is a function required by `adaptivetau` that returns the rates for each transition in the defined list of transitions. `transitions` is a list containing all possible epidemiological transitions, e.g., become infected, death of susceptible, recovery, etc. `start_year` and `end_year` are the start and end years of the simulation. `use_immigration` is a logical value that tells the function whether or not to include immigration. `immigration_factor` is a small factor between 0 and 1 that tells the function how much immigration to include. `init_run` is the number of years to run for from the initial conditions before the simulation begins in order to avoid transient behaviour. `one_year` tells the function how many time steps there are in a year, e.g., 1 means the time step is a year while 12 means that the time step is a month. `gamma` and `sigma` are the reciprocals of the mean infectious and latent periods respectively. `alpha` is the amplitude of seasonal forcing. `R0` is the basic reproduction number. `clock_tick` tells the function how often we wish to save data, e.g., if the time step is a year, `clock_tick`$= \frac{1}{12}$ saves the data monthly. Finally, `probability_of_report` is the probability that a measles case will be reported.

The function begins by filtering the vital statistics data to contain on the years we wish to simulate. Next, the function determines the value of `immigration_factor` based on `use_immigration`. Additionally, the function finds several other arguments pertaining to the model including the final time of the simulation, the time at which the first vaccination occurs, the number of clock ticks that will be saved, and the population of each US state at the given start year. Next, it creates empty data frames that will contain the simulated cumulative incidence for each US state and the population of each state over time.

Since we are simulating data for each US, the function then runs a for loop. Within the for loop, first, the data frame containing vital statistics is filtered to include the data for a single US state. Next, it sets the initial values of the birth and death rates to be the value at the earliest year of data. Additionally, it adds a time column to the vital statistics data frame. With all of the necessary parameters defined, the parameters are combined into a list for use the in rate function. Using the list of the parameters and the vector of initial population values, the initial conditions of the model are set equal to the equilibrium point at $t = 0$. The `adaptivetau` function is then run. To save data at specific times we use the following `insert_clock_ticks` function created by Dr. David Earn. The last argument of the for loop outputs the saved values of time, cumulative incidence and population into the corresponding data frames.

```
### Function to save the data at specific time intervals
insert_clock_ticks <- function(x, clockTick, time.var="t" ) {
  clockTick <- clockTick
  xt <- x[,time.var] # it is *much* faster to deal with a vector
  ####################################
  ## FIX: don't really need this bit:
  mean.dt <- mean(diff(xt))
  ## 0.000478 = mean(dt) from diagnostic
  ## histogram (tau_hist)
  events.per.clockTick <- round(clockTick / mean.dt)
  message("insert_clock_ticks:\n",
          "\tclockTick: ", clockTick, "\n",
          "\tmean.dt: ", mean.dt, "\n",
          "\tleaps.per.clockTick: ", events.per.clockTick, "\n"
  )
  ####################################
  w <- which(diff(floor(xt/clockTick))==1) ## +1 ? before/after?
  tickvec <- rep(NA_real_,length(xt))
  tickvec[w] <- xt[w]
  x[,"clock"] <- tickvec
  return(x)
}
```

After the for loop is complete, the function calculates incidence from cumulative incidence and outputs it into a new data frame. Since the model simulates incidence, not reported cases, the function implements a binomial to sample incidence with a probability of case being reported set equal to `probability_of_report`. Next, a new column is added to each data frame to contain the mean across each US state. Finally, the function calculates yearly incidence and reports per 100,000, outputs them to data frames and returns the data frames in a list. See the function below.

```
path_model <- function(pop, vit_stats, vacc_data, US_states,
                       rate_fun, transitions, start_year,
                       end_year, use_immigration,
                       immigration_factor,
                       init_run, one_year, gamma,
                       sigma, alpha, R0, clock_tick,
                       probability_of_report) {

  ## Number of years we are simulating
  data_years <- end_year - start_year +1

  ## Filter the vital stats data to only contain data
  ## for the years we wish to simulate
  vital_stats <- vital_stats %>%
    filter(Year %in% c(start_year:end_year))

  ## Choose the immigration factor based on the value
  ## of use_immigration
  if (use_immigration) {
    immigration_factor <- immigration_factor
  } else {
    immigration_factor <- 0
```

```
}

## Final time of simulation
t_final <- (init_run + data_years)*one_year

## Time at which vaccination begins, counts the number
## of years until 1963 (when vaccine was invented)
if (init_run == 0) {
  if (start_year == 1963) {
    t_first_vacc <- 0
  } else {
    t_first_vacc <- 1963 - start_year + 1
  }
} else {
  t_first_vacc <-
    (init_run + 1963 - start_year + 1)*one_year
}

## Number of clock ticks that will be saved
n_ticks <- length(seq(clock_tick,
                      t_final,
                      clock_tick))

## Get the population for each state
## from pop in the given starting year
pop_states <-
  round(pop$Population[pop$Year == start_year])

## Create the data frames that will contain the
## cumulative incidence and population
SEIR_cuminc_sims <-
  as.data.frame(matrix(nrow = n_ticks,
                       ncol = length(pop_states) + 1))
colnames(SEIR_cuminc_sims) <- c("time", US_states)
SEIR_pop_sims <- SEIR_cuminc_sims

## Run the simulation for each state
for (s in 1:length(US_states)) {
  ## Filter the vital stats to only contain data
  ## for the desired state
  vital_stats_filtered <- vital_stats %>%
  filter(State == US_states[s])
```

```
## Set the initial values of mu and nu to the
## values in start_year
nu_init <- vital_stats_filtered$`Birth Rate`[1]
mu_init <- vital_stats_filtered$`Death Rate`[1]

## Create a time vector in vital stats that
## depends on init_run
if (init_run == 0) {
  vital_stats_filtered$Time <-
    c(1:(length(unique(vital_stats$Year))))
} else {
  vital_stats_filtered$Time <-
    c((init_run + 1):t_final)
}

## List of the parameters used in the
## adaptivetau function
params_SEIR <- list(
  vital_stats = vital_stats_filtered,
  init_run = init_run,
  t_first_vacc = t_first_vacc,
  nu_init = nu_init,
  mu_init = mu_init,
  t_final = t_final,
  N = pop_states[s],
  gamma = gamma,
  sigma = sigma,
  alpha = alpha,
  beta0 = beta0_fun(c(nu = nu_init,
                      mu = mu_init,
                      gamma, sigma, R0)),
  one_year = one_year,
  immigration_factor = immigration_factor,
  vaccine_data = vaccination_data,
  start_year = start_year,
  end_year = end_year
)

## Use the eq_prev_fun to set the initial conditions
## to the endemic equilibrium point
EE_time_var <-
```

```
    eq_prev_fun(t=0, params = params_SEIR,
                N = pop_states[s])

  S <- EE_time_var$S_eqm
  E <- EE_time_var$E_eqm
  I <- EE_time_var$I_eqm
  R <- EE_time_var$R_eqm
  C <- 0

  init_values_SEIR <- c(S = S, E = E,
                        I = I, R = R, C = 0)

  ## Simulate the model using ssa.adaptivetau
  set.seed(36)
  SEIR_sims <- ssa.adaptivetau(init_values_SEIR,
                               transitions, rate_fun,
                               params_SEIR, tf=t_final)

  SEIR_results <- as.data.frame(SEIR_sims)

  ## Use the insert_clock_ticks function David Earn
  ## created to save the states at chosen time interval
  fixed_time_SEIR_results <-
    insert_clock_ticks(SEIR_results,
                       clockTick = clock_tick,
                       time.var = "time")

  fixed <- fixed_time_SEIR_results %>%
    filter(!is.na(clock))
  fixed_N <- fixed %>%
    mutate(N = S + E + I + R)

  ## Output the saved values of time,
  ## C and N into the corresponding data frames
  SEIR_cuminc_sims[1:length(fixed$clock),"time"] <- fixed$clock
  SEIR_cuminc_sims[1:length(fixed$clock),s + 1] <- fixed$C
  SEIR_pop_sims[1:length(fixed$clock),"time"] <- fixed$clock
  SEIR_pop_sims[1:length(fixed$clock),s + 1] <- fixed_N$N
}

## Remove the time column from the cumulative incidence,
## and create a data frame the same size that will contain incidence
```

```r
cuminc_sims <- SEIR_cuminc_sims[,-1]
inc_sims <- cuminc_sims

## Calculate incidence
for (i in 1:ncol(cuminc_sims)) {
  inc_sims[, i] <- c(NA, diff(cuminc_sims[ ,i]))
}

## Define a new data frame that contains incidence
SEIR_inc_sims <- SEIR_cuminc_sims
SEIR_inc_sims[,-1] <- inc_sims


## Create a data frame that will contain reported
## cases (our model simulates incidence)
reports <- SEIR_inc_sims[,-1]

## Sample incidence using a negative binomial,
## where the probability of a case being reported is 10%
for (i in 1:ncol(reports)) {
  for (j in 1:nrow(reports)) {
    set.seed(400)
    reports[j,i] <-
      rbinom(n = 1,
              size = SEIR_inc_sims[j, (i + 1)],
              prob = probability_of_report)
  }
}

## Add columns that will contain the mean
SEIR_inc_sims$mean <- 0
SEIR_pop_sims$mean <- 0
reports$mean <- 0

## Find the mean each year using rowMeans
for (i in 1:(nrow(SEIR_inc_sims)-1)) {
  SEIR_inc_sims$mean[i+1] <- rowMeans(SEIR_inc_sims[i+1,-1],
                                    na.rm = TRUE)
  SEIR_pop_sims$mean[i+1] <- rowMeans(SEIR_pop_sims[i+1,-1],
                                    na.rm = TRUE)
  reports$mean[i+1] <- rowMeans(reports[i+1,], na.rm = TRUE)
}
```

```r
## Calculate the incidence per 100k individuals using the
## saved population size at each time
inc_per_100k <- SEIR_inc_sims[,-1]
reports_per_100k <- reports

for (i in 1:ncol(inc_per_100k)) {
  inc_per_100k[,i] <- SEIR_inc_sims[,i+1]*100000/SEIR_pop_sims[,i+1]
  reports_per_100k[,i] <- reports[,i]*100000/SEIR_pop_sims[,i+1]
}

### Aggregate the incidence per 100k to yearly
## (current interval chosen by clock_tick)
yearly_inc_per_100k <-
  as.data.frame(matrix(ncol = ncol(inc_per_100k),
                       nrow = t_final))
yearly_reports_per_100k <-
  as.data.frame(matrix(ncol = ncol(reports_per_100k),
                       nrow = t_final))

for (j in 1:ncol(yearly_inc_per_100k)) {
  yearly_inc_per_100k[1, j] <-
    sum(inc_per_100k[1:11, j],
        na.rm = TRUE)
  yearly_reports_per_100k[1, j] <-
    sum(reports_per_100k[1:11, j],
        na.rm = TRUE)
  for (i in 1:(t_final-1)) {
    yearly_inc_per_100k[i+1, j] <-
      sum(inc_per_100k[(12*i):(12*i+11), j],
          na.rm = TRUE)
    yearly_reports_per_100k[i + 1, j] <-
      sum(reports_per_100k[(12*i):(12*i+11), j],
          na.rm = TRUE)
  }
}

## Set the column names to be the state names and then mean
colnames(yearly_inc_per_100k) <- colnames(SEIR_inc_sims[,-1])
colnames(yearly_reports_per_100k) <- colnames(reports)
colnames(inc_per_100k) <- colnames(SEIR_inc_sims[,-1])
colnames(reports_per_100k) <- colnames(SEIR_inc_sims[,-1])
```

```
  yearly_inc_states <- yearly_inc_per_100k
  yearly_reports_states <- yearly_reports_per_100k

  return(list(yearly_inc_states = yearly_inc_states,
              yearly_reports_states = yearly_reports_states,
              monthly_inc_states = inc_per_100k,
              monthly_reports_states = reports_per_100k))
}
```

7.5.3 Optimization of Model Parameters

While most of the model parameters can be obtained through literature, we decided to optimize the following parameters: amplitude of seasonal forcing, α and the immigration factor. For α values ranging from 0 to 0.8 and immigration factor values ranging from 0 to 0.01 by factors of 10, we simulate the aggregate US path for all 81 parameter combinations in the 2D parameter space. For each parameter combination, we use the `path_model` function to simulate yearly reports per 100,000, `cv_inc_fun` to create the data frames containing weighted Incidence and CV, and then use `scale_cases_percentile` and `dist_btw_canon_path` to find the distance between the simulated path and the path created using real world incidence data. The distances are then outputted in a data frame. Refer to `optimizing_parameters_new.R` if you wish to recreate the results, as the code takes approximately 1 day to run.

7.5.3.1 Optimization Result Summary

To summarize the optimization results, we begin by loading all required *.RData* files and libraries.

```
load("Good_data/optimization_results.RData")
library(tidyverse)
library(gridExtra)
library(grid)
library(ggplot2)
library(lattice)
library(kableExtra)
library(knitr)
```

Next, we output a table containing the parameter values corresponding to the minimum distance.

```r
distance_results <- distance_results %>%
  filter(Immigration_factor != 0.1)
## Location of minimum distance
min_loc <- which.min(distance_results$Distance)

## Table containing parameter values corresponding to the minimum distance
kable(distance_results[min_loc,])
```

	Init_run	Distance	Immigration_factor	Alpha
48	200	0.405692	1e-05	0.2

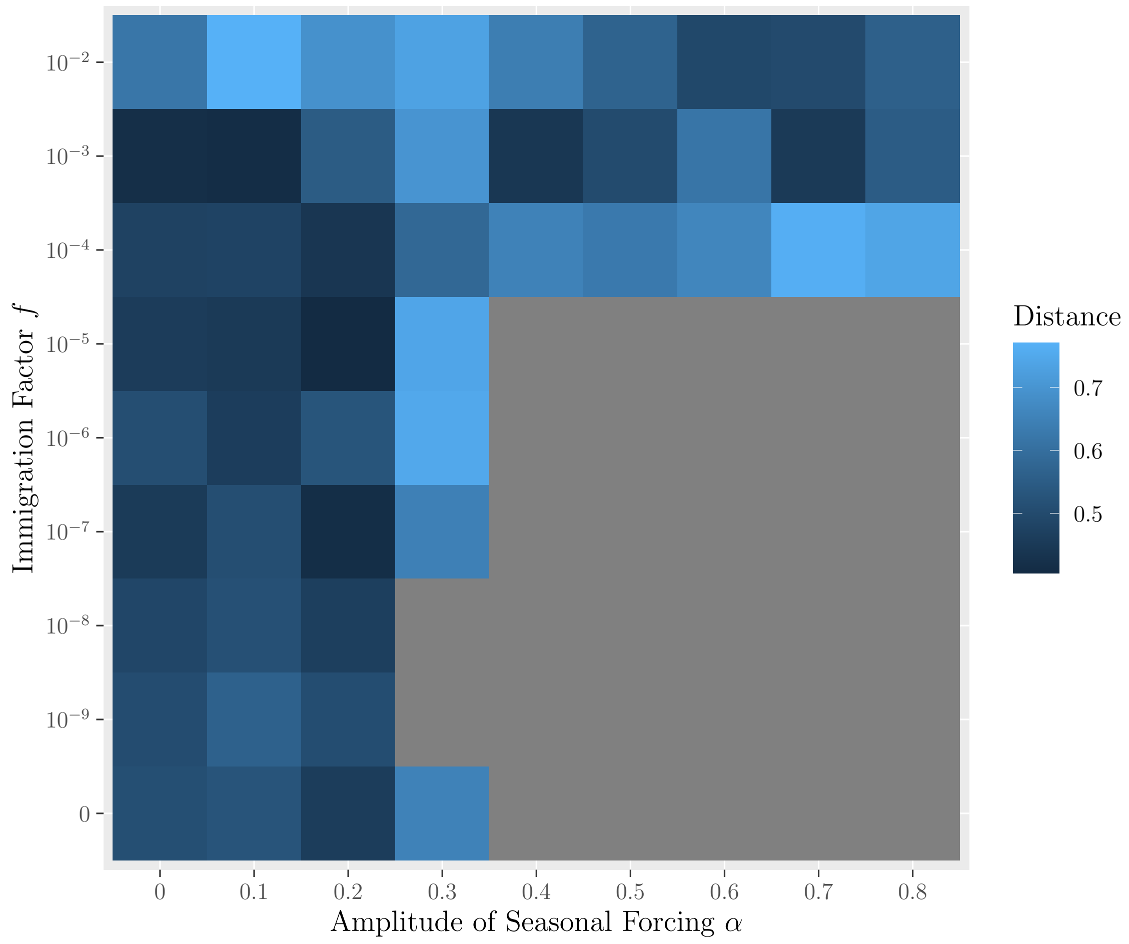
To provide a visual of the results, we create a heat map with α is on the x-axis and `immigration_factor`, f, is on the y-axis. The colour of each grid block is the distance (grey blocks represent paths that went extinct).

```r
## Create a data frame for each value of init_run so we can create
## heatmaps
results_200 <- distance_results

powers <- -c(2:9)
labs <- vector(length = length(powers)+1)

for (i in 1:length(powers)) {
  labs[i] <- paste("$10^{", powers[i],"}$", sep = "")
}

labs[length(labs)] <- "0"

label_df <- data.frame(
  Imm = as.factor(as.character(c(10^-(2:9), 0))),
  Labels = labs
)

results_200$Label <- 0

for(i in 1:nrow(results_200)) {
  for (j in 1:nrow(label_df)) {
    if (results_200$Immigration_factor[i] == label_df$Imm[j]) {
      results_200$Label[i] <- label_df$Labels[j]
    }
  }
}
```

```
## Create a heatmap
heatmap_optim <- ggplot() +
  geom_tile(data=results_200,
            aes(as.character(Alpha),
                Label,
                fill = Distance)) +
  scale_fill_continuous(
    limits = c(min(distance_results$Distance),
               max(distance_results$Distance))
                    ) +
  xlab("Amplitude of Seasonal Forcing $\\alpha$") +
  ylab("Immigration Factor $f$") +
  scale_y_discrete(limits = rev)
```

The following figure shows the heat map of distance, as shown in figure 13.

```
heatmap_optim
```



Finally, we create a data frame containing the parameter combinations resulting in the top 10 most similar paths.

```r
## Since the data frame is ordered by distance,
## take top 10 results
distance_results_ordered <-
  distance_results[order(distance_results$Distance),]
distance_results_ordered$Rank <- c(1:nrow(distance_results_ordered))
top_10_results <- distance_results_ordered[1:10,]
kable(top_10_results)
```

	Init_run	Distance	Immigration_factor	Alpha	Rank
48	200	0.4056920	1e-05	0.2	1
65	200	0.4130475	1e-03	0.1	2
30	200	0.4140507	1e-07	0.2	3
64	200	0.4179669	1e-03	0.0	4
57	200	0.4400064	1e-04	0.2	5
68	200	0.4414252	1e-03	0.4	6
47	200	0.4514070	1e-05	0.1	7
71	200	0.4516143	1e-03	0.7	8
28	200	0.4539835	1e-07	0.0	9
46	200	0.4582254	1e-05	0.0	10

We then save the results to a *.RData* file for use in other code.

```
save(distance_results,
     top_10_results, heatmap_optim,
     file = 'data/optimization_plots.RData')
```

7.5.3.2 Plotting the Top 10 Parameter Pairing Paths

To visualize the optimization results, we chose to plot the paths created for the top 10 parameter combinations. In `optimizing_parameters_new.R`, we save the monthly incidence and data frames containing weighted incidence and CV that are required to create the canonical path for each parameter combination. We begin by loading in the required *.RData* files and libraries.

```
library(tidyverse)
library(RColorBrewer)
library(knitr)
library(ggpubr)
load("Good_data/optimization_results.RData")
load("Good_data/true_path_10_3.RData")
load("data/optimization_plots.RData")
load("data/presentation_logical.RData")
```

Next, we filter each data frame to only contain the years we wish to compare (1962 to 2016).

```
## Filter the data frames to only go up to 2016
## since that is when the path created with real data ends
data_years <- c(1962:2016)
path_data_frames_filter <- path_data_list

for (i in 1:length(path_data_list)) {
  path_data_frames_filter[[i]] <- path_data_list[[i]] %>%
    filter(Year %in% data_years)
}
```

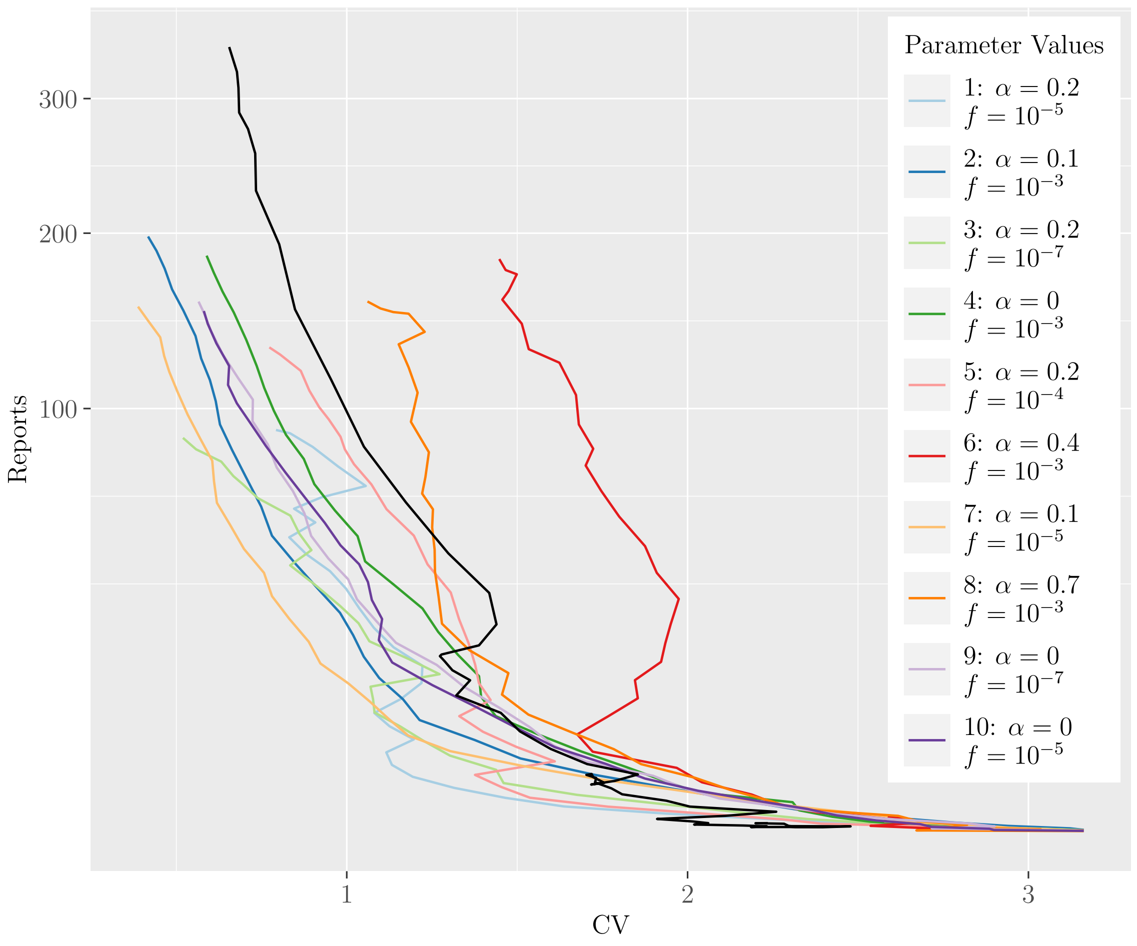
Now that each data frame contains only the years we need, we bind each data frame by row to have one data frame with the data of all simulated paths. Since, the list is named based on the parameter combinations, we index each data frame using a label the describes the value of the length of the transient, immigration factor, and α. However, we only want the path data for the top 10 parameter combinations, thus we create a column in the `top_10_results` data frame that has the same format, so we can filter by the label column. We then filter the data frame to only contain the top 10 paths, create a rank column and re-format the label column.

```
## Create a single data frame by binding each list item by row,
## add a column that indexes each list item by the label
all_path_data <- bind_rows(path_data_frames_filter, .id = "Label")

## Create a label column in top_10_results with the same format
top_10_results$Label <- 0

for (i in 1:nrow(top_10_results)) {
  top_10_results$Label[i] <-
    paste("init run: ",
          top_10_results$Init_run[i],
          ", $f$: ",
          top_10_results$Immigration_factor[i],
          ", $\\alpha$: ",
          top_10_results$Alpha[i],
          sep = "")
}

## Create a Power column to represent the power of 10 of the
## immigration factor
powers <- -c(5,3,7,3,4,3,5,3,7,5)
top_10_results$Power <- powers
```

```r
## Filter all_path_data to only include the data for
## the top 10 paths and add a rank column
all_path_data_top_10 <- all_path_data %>%
  filter(Label %in% unique(top_10_results$Label))

all_path_data_top_10$Rank <- 0

for (i in 1:nrow(all_path_data_top_10)) {
  for (j in 1:nrow(top_10_results)) {
    if (all_path_data_top_10$Label[i] == top_10_results$Label[j]) {
      all_path_data_top_10$Rank[i] <- top_10_results$Rank[j]
    }
  }
}

## Order by rank and make the rank column a factor
all_path_data_top_10 <-
  all_path_data_top_10[order(all_path_data_top_10$Rank),]

all_path_data_top_10$Rank <- factor(all_path_data_top_10$Rank,
                                    levels = as.character(1:10)
                                    )

## Use paste to make the label column provide the rank,
## immigration, and alpha for each set of data
for (i in 1:nrow(all_path_data_top_10)) {
  Rank <- all_path_data_top_10$Rank[i]
  Init_run <-
    top_10_results$Init_run[
      which(top_10_results$Rank == Rank)]
  Power <-
    top_10_results$Power[
      which(top_10_results$Rank == Rank)]
  Alpha <-
    top_10_results$Alpha[
      which(top_10_results$Rank == Rank)]


  val <- paste(Rank, ": ",
               paste(paste(
                        "$\\alpha=",
                            Alpha, "$", sep = ""),
                      paste("$f= 10^{",
                            Power, "}$",
                            sep = ""), sep = "\n"),
                sep = "")

  all_path_data_top_10$Label[i] <- val
}

## Define a vector of labels to be the label column
labels <- all_path_data_top_10$Label
```

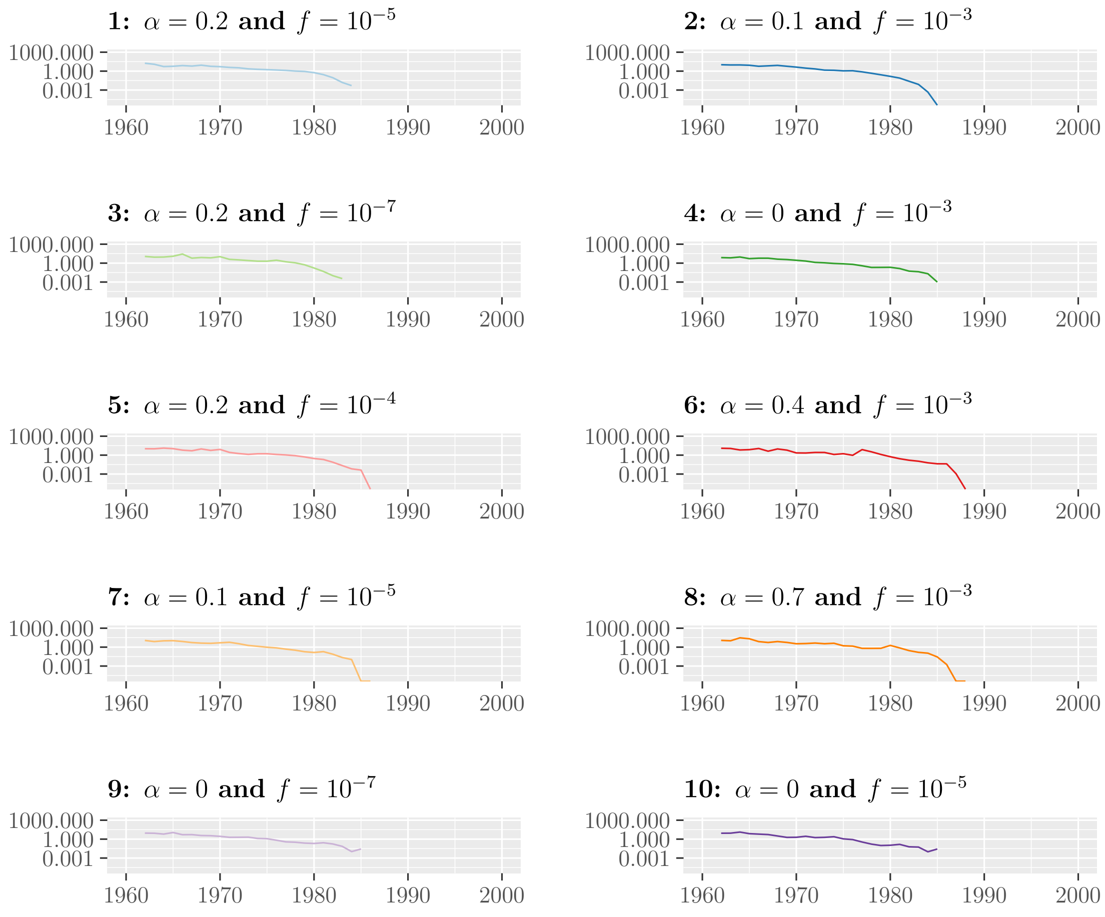
Finally, we create the plot displayed in figure 14.

```
## Define the palette used for the ggplot (from Paired brewer palette)
# palette <- palette(brewer.pal(n=10, name = "Paired"))
my_palette_top_10 <-
  c("#A6CEE3", "#1F78B4", "#B2DF8A", "#33A02C" ,
    "#FB9A99", "#E31A1C" , "#FDBF6F", "darkorange1",
    "#CAB2D6", "#6A3D9A")
## Set names of each colour to be the rank
ranks <- c(1:10)
names(my_palette_top_10) <- ranks

## Rename the data frame containing the real path data
data_us_filter <- data_us_unif

compare_top_10 <- ggplot() +
  geom_path(data = all_path_data_top_10,
            aes(CV, Reports, col = Rank), size = 0.75) +
  geom_path(data = data_us_filter, aes(CV, Incidence),
            col = "black", size = 0.75) +
  scale_y_sqrt() +
  scale_colour_manual(values = my_palette_top_10,
                      labels = unique(labels)) +
  labs(colour = "Parameter Values") +
  theme(
    legend.position = c(.99, .99),
    legend.justification = c("right", "top"),
    legend.box.just = "center",
    legend.margin = margin(6, 6, 6, 6),
    legend.title.align = 0.5,
    legend.direction = "vertical",
    legend.spacing.y = unit(0.25, 'cm'),
    legend.text = element_text(size = text_size),
    legend.title = element_text(size = text_size),
    axis.text = element_text(size = text_size),
    axis.title = element_text(size = text_size)
     ) +
  guides(colour = guide_legend(byrow = TRUE))
compare_top_10
```



Additionally, we plot the time series corresponding to each of the parameter combinations in the code below. This plot is presented in figure 15.

```r
## Define arguments used to subset the time series data
window <- 10
step <- 1/12
ts_list_top_10 <- as.list(vector(length = length(ranks)))
names(ts_list_top_10) <- ranks
ts_list_short <- as.list(vector(length = length(ranks)))

## Plot the both the full time series for each set of parameters
## and the time series for 5 years

ts_data_frames <- ts_data_list[top_10_results$Label]

for (i in 1:length(ts_data_frames)) {
  row_ind <- which(top_10_results$Rank == ranks[i])
  init_run <- top_10_results$Init_run[row_ind]
  alpha_val <- top_10_results$Alpha[row_ind]
  power_val <- top_10_results$Power[row_ind]

  ts_years <- seq((data_years[1]-init_run-window+step),
                  2019, step)

  monthly_data <- ts_data_frames[[i]]
  monthly_data$Year <- ts_years

  monthly_data_long <-
    pivot_longer(monthly_data, cols = !Year,
                 names_to = "State",
                 values_to = "Reports")

  monthly_data_long$Rank <- as.character(ranks[i])

  mean_only <- monthly_data_long %>%
    filter(State == "mean" & Year %in% data_years)

  ts_plot_top_10 <- ggplot() +
    geom_path(data = mean_only,
              aes(Year, Reports, col = Rank)) +
    scale_y_log10(limits = c(0.00001, 1000),
                  labels =
                    function(x) format(x, scientific = FALSE)) +
    scale_colour_manual(values = my_palette_top_10,
                        labels = unique(labels)) +
    ylab("") +
    xlab("") +
    xlim(c(1960, 2000)) +
    ggtitle(paste(ranks[i],
                  ": $\\alpha=",
                  alpha_val,
                  "$ and $f=10^{",
                  power_val, "}$",
                  sep="")) +
    theme(plot.title = element_text(size = 10,
                                    face = "bold"))

  ts_list_top_10[[i]] <- ts_plot_top_10
}
top_10_plot_list <- ts_list_top_10

ggarrange(plotlist = top_10_plot_list,
          ncol = 2, nrow = 5,
          common.legend = TRUE,
          legend = "none")
```



We save the results to a *.RData* file for use in other code.

```
save(true_path, data_us, point_mean_us,
     compare_top_10, all_path_data_top_10,
     my_palette_top_10, file = "data/top_10.RData")
```

7.6 Canonical Paths for Varying Seasonality

With a model that is capable of simulating the US canonical path, we decided to investigate how varying seasonality affects the trajectory of the path. In this regard, we simulate the path for the following α values: 0, 0.1, 0.2, ..., 0.8, and using an immigration factor of 10^{-3} (optimized). We use the `path_model` function, as described in section 7.5.2.2, to simulate the yearly incidence per 100,000 and the `cv_inc_fun`, as described in section 7.3.1, to create the data frames containing weighted incidence and CV for each value of α. The simulation is conducted in `model_sensitivity_alpha_new.R` as the code takes several hours to run. The calculation of weighted incidence and CV is conducted in `cv_inc_sensitivity_alpha_new.R` as it takes over an hour to run.

Once the data frames containing weighted incidence and CV are created, we begin creating the plots of the data. First, we load in the necessary *.RData* files and libraries.

```
library(stats)
library(smoother)
library(calibrate)
library(tidyverse)
library(rlist)
library(RColorBrewer)
library(knitr)
library(gridExtra)
library(ggpubr)
load("data/state_data_vacc_alphas_200_imm.RData")
load("data/cv_inc_state_alpha_sensitivity_200_imm.RData")
load("data/presentation_logical.RData")
```

Next, we determine the year in which the plotting should begin. This is not necessarily the starting year as we ran for 200 years from the initial conditions to remove transient behaviour.

```
## Determine at which data point the path should be
## plotted (want to plot after the extra run time is
## complete)
use_immigration <- TRUE
init_run <- 200
if (init_run == 0) {
    start_plot <- 1
} else {
    start_plot <- init_run + 1
}
```

Next we filter the data frames to on include the years we wish to plot.

```
years <- unique(data_us$year)
final_years <- years[start_plot:length(years)]
```

```
## Filter the data based on the year we want to begin
## plotting based on start_plot
data_us_reports <- data_us_reports %>%
    filter(year %in% final_years)
```

Before plotting the paths, we solve the deterministic version our model to determine what stable cycles (if any) are present with each value of alpha. First, we load in the necessary *.RData* files and libraries, and source the R files containing default parameters and functions.

```
library(tidyverse)
library(deSolve)
library(knitr)

source("functions.R")
source("model_parameter_inputs_new.R")
load("data/vital_stats_data.RData")
load("Good_data/US_population_data.RData")
```

Next, we set the arguments required to simulate the deterministic model.

Once we have defined all of the necessary data files, we run a for loop through each value of alpha. It begins by creating a vector of names parameters required by the following deterministic rate function. Since we observe large changes in magnitude of prevalence when α is chaotic, we logarithmically integrate E and I.

```
### Deterministic Rate function
SEIR_det <- function(t, state, params) {
  with(as.list(c(state, params)), {
    beta <- beta_fun(t, alpha, beta0, one_year)
    I <- exp(logI)
    p <- vaccination_fun(t, t_final, t_first_vacc, final_vacc_coverage)
    E <- exp(logE)

    N <- (S + E + I + R)
    dS <- N*(1 - p)*nu - S*I*beta/N - mu*S
    dlogE <- S*I*beta/(E*N) - sigma + mu
    dlogI <- sigma*E/I - gamma + mu
    dR <- N*p*nu + gamma*I - mu*R
    list(c(dS, dlogE, dlogI, dR))

  })
}
```

Next, the for loop sets the initial conditions to be the equilibrium point at $t = 0$. Additionally, we create a vector containing the time for which we wish to simulate the deterministic model. Finally, we call the `lsode` function from the `deSolve` library, to solve the deterministic model. The code also creates plots of the deterministic solution and then subsets them so that we can find the stable cycles.

```
for (alpha_val in alpha_vec) {
  ## Vector containing the parameters used in the ode function
  params_det <- c(
    nu = nu,
    mu = mu,
    N = N,
    gamma = gamma,
    sigma = sigma,
    alpha = alpha_val,
    beta0 = beta0_fun(c(nu, mu, gamma, sigma, R0)),
    one_year = one_year,
    immigration_factor = immigration_factor,
    R0 = R0
  )

  ## List version of params_det
  params_eqm <- as.list(params_det)

  ## Set initial conditions to equilibrium
  EE <- eq_prev_fun(0, params_eqm, N = params_eqm$N)

  S <- EE$S_eqm
  E <- EE$E_eqm
  I <- EE$I_eqm
  R <- EE$R_eqm

  init_values_det <- c(S = S, logE = log(E),
                       logI = log(I), R = R)

  ## Defining the time sequence used in the ode function
  times <- seq(0, t_final, by = clock_tick)

  ## Chose to use lsode to ensure it treats the
  ## model as stiff, asks the function to find
  ## the jacobian using jactype = "fullint"
  SEIR_ode <- lsode(y = init_values_det, times = times,
                    func = SEIR_det, parms = params_det,
                    jactype = "fullint")

  ## Make the output of ode a data frame
  SEIR_ode_df <- as.data.frame(SEIR_ode)

  ## Save the data frame to a list
  alpha_det_list[[which(alpha_vec == alpha_val)]] <-
    SEIR_ode_df
}

short_alphas <- c(0.025, 0.1, 0.2)

for (i in 1:length(alpha_det_list)) {
  if (alpha_vec[i] %in% short_alphas) {
    time_low <- 2015
  } else {
    time_low <- 1970
  }
  data <- alpha_det_list[[i]] %>%
    filter(time >= time_low & time <= 2020)

  plot <- ggplot() +
    geom_line(data=data, aes(time, logI)) +
    ggtitle(paste0("$\\alpha=$ ", alpha_vec[i]))

  print(plot)
}
```

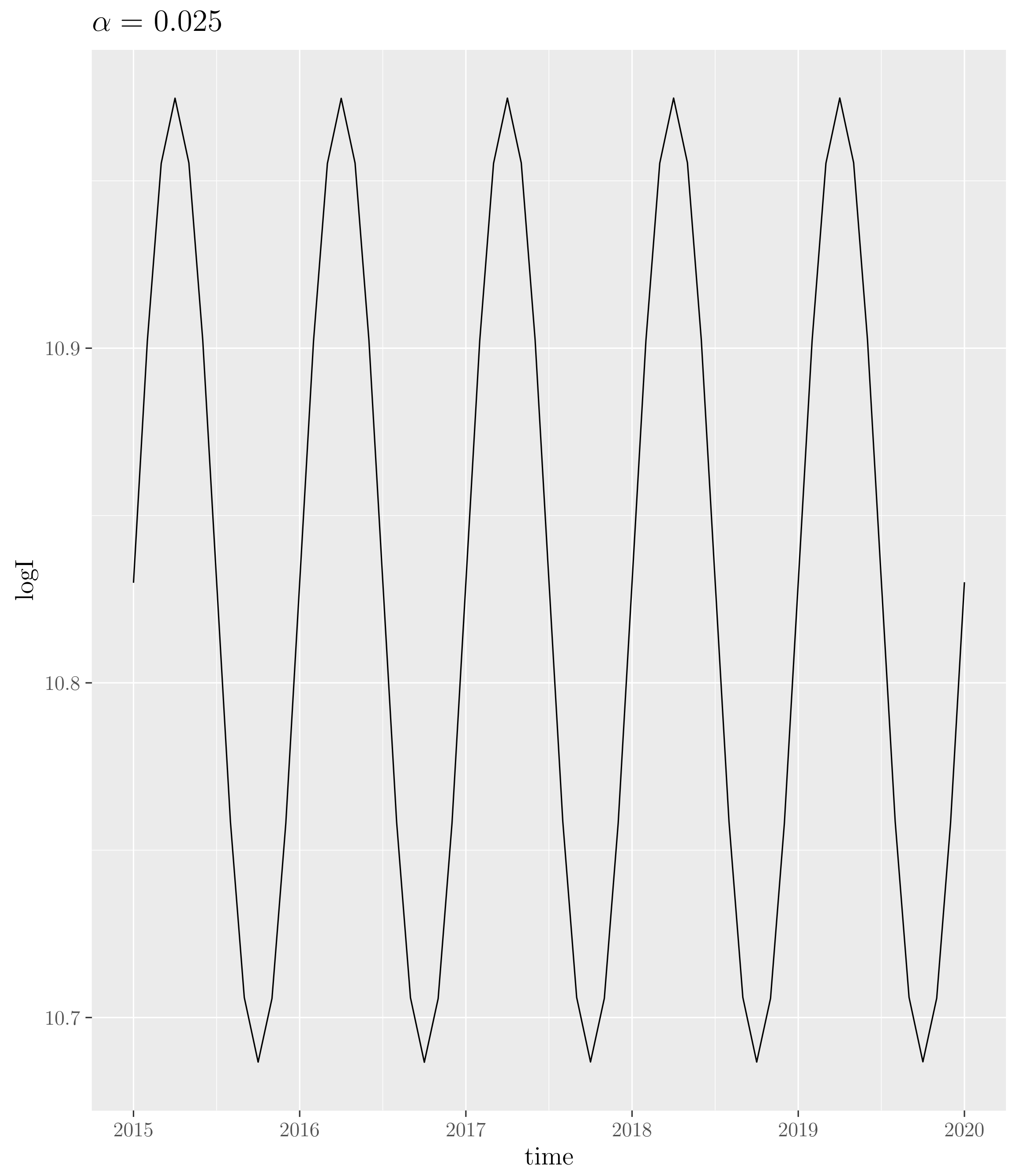


Figure 31: The deterministic solutions to the model for ranging values of α. From these plots we observe an annual cycle for α values of 0.025, 0.1 and 0.4, triennial for 0.2 and 0.3 and chaos for the rest.

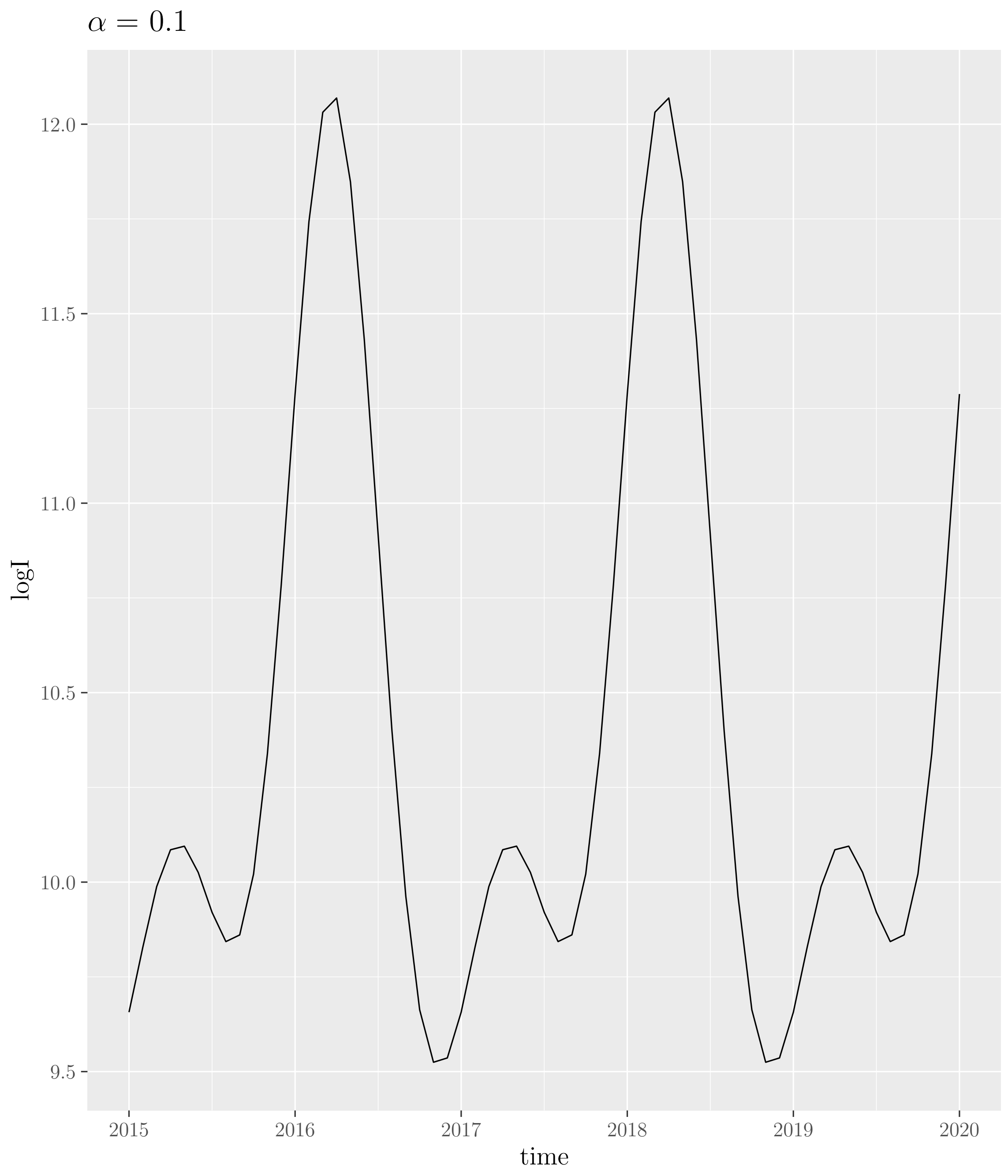


Figure 32: The deterministic solutions to the model for ranging values of α. From these plots we observe an annual cycle for α values of 0.025, 0.1 and 0.4, triennial for 0.2 and 0.3 and chaos for the rest.

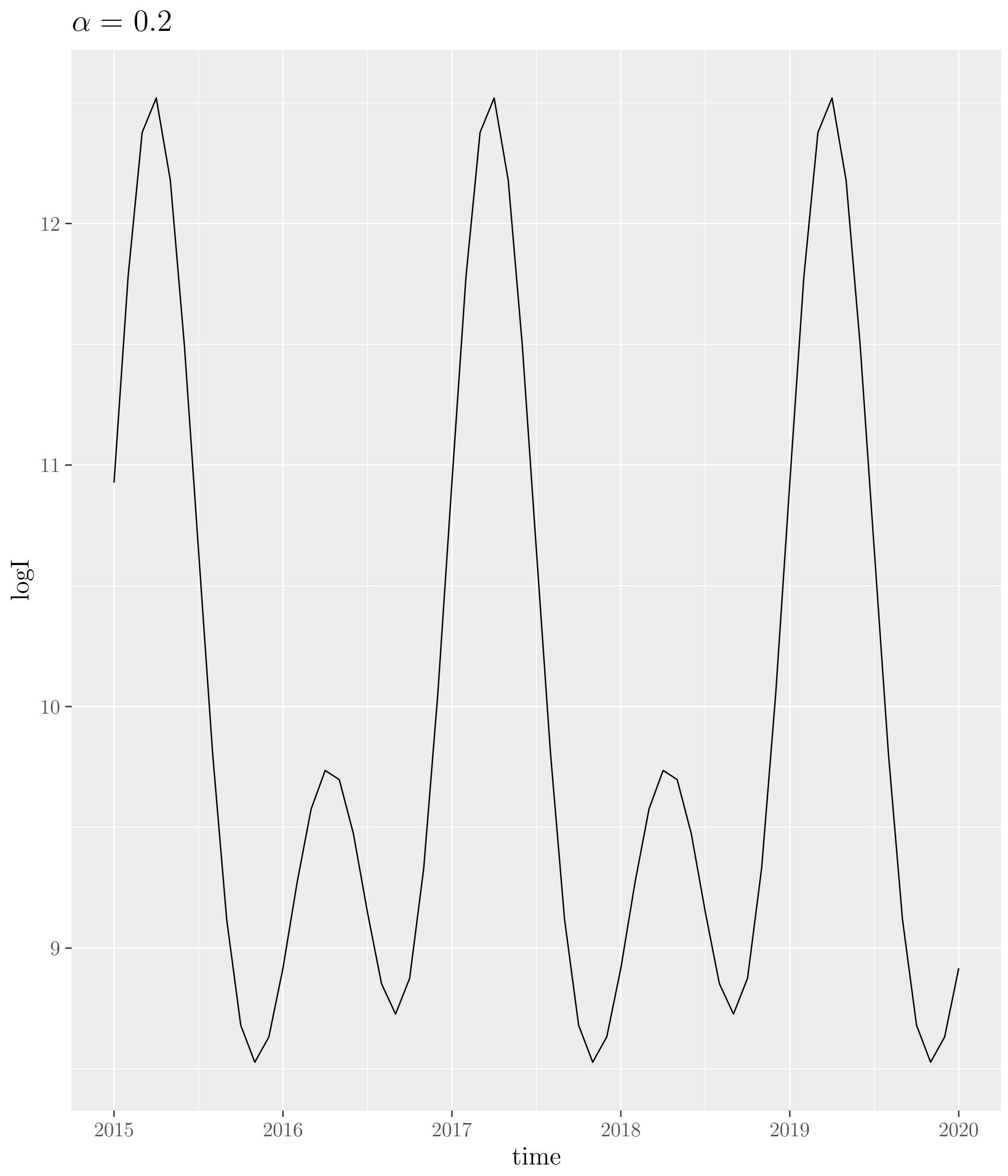


Figure 33: The deterministic solutions to the model for ranging values of α. From these plots we observe an annual cycle for α values of 0.025, 0.1 and 0.4, triennial for 0.2 and 0.3 and chaos for the rest.

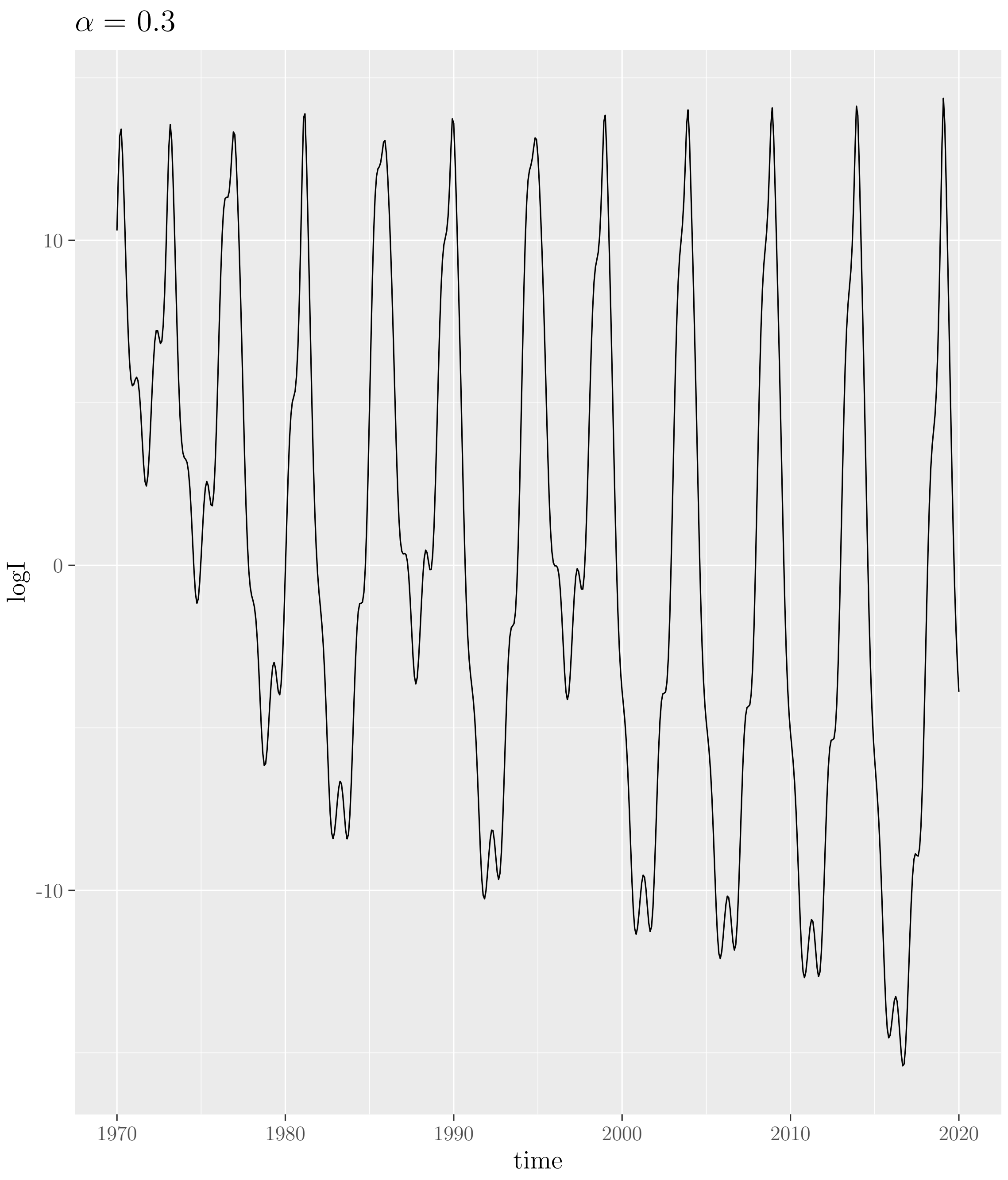


Figure 34: The deterministic solutions to the model for ranging values of α. From these plots we observe an annual cycle for α values of 0.025, 0.1 and 0.4, triennial for 0.2 and 0.3 and chaos for the rest.

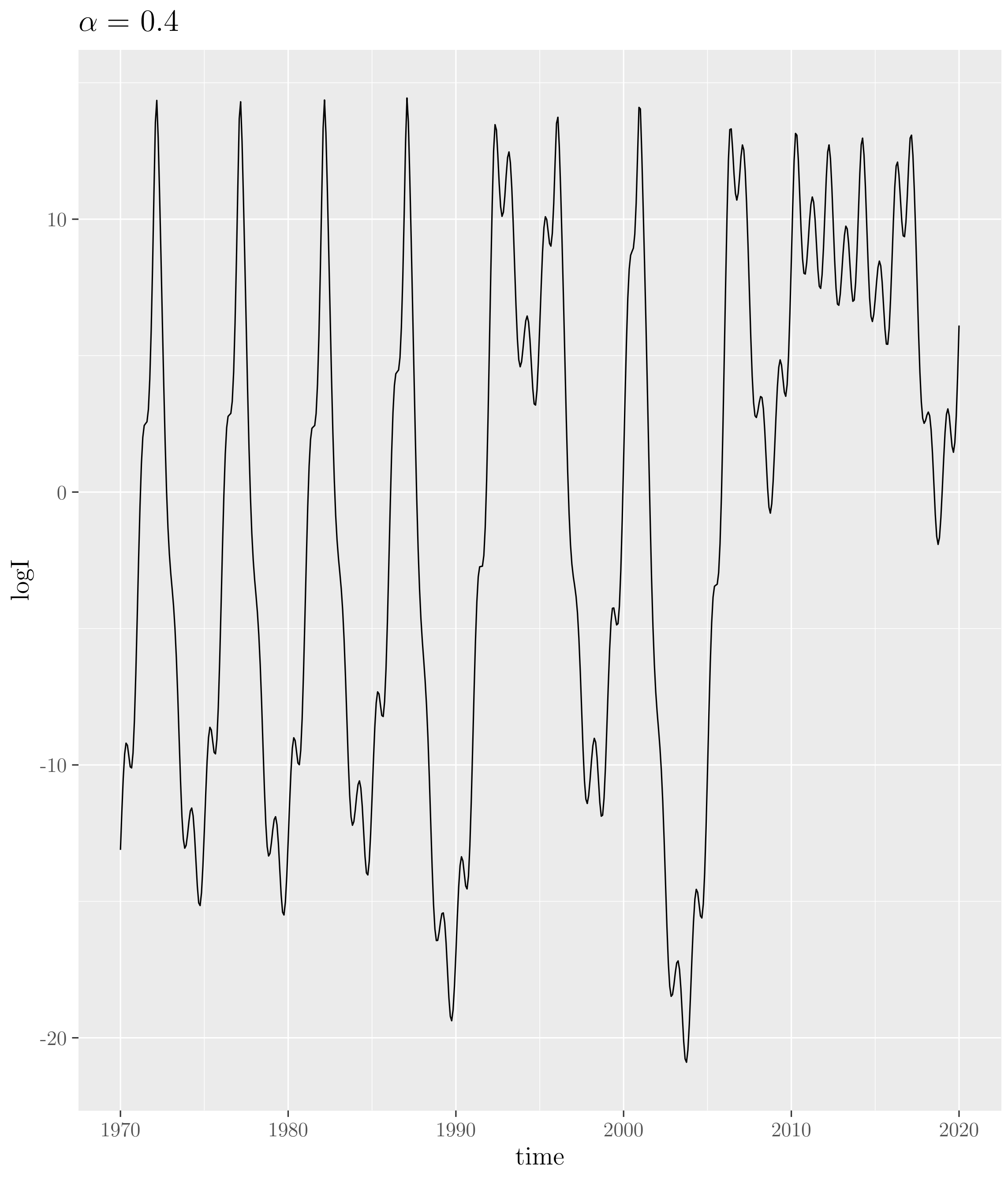


Figure 35: The deterministic solutions to the model for ranging values of α. From these plots we observe an annual cycle for α values of 0.025, 0.1 and 0.4, triennial for 0.2 and 0.3 and chaos for the rest.

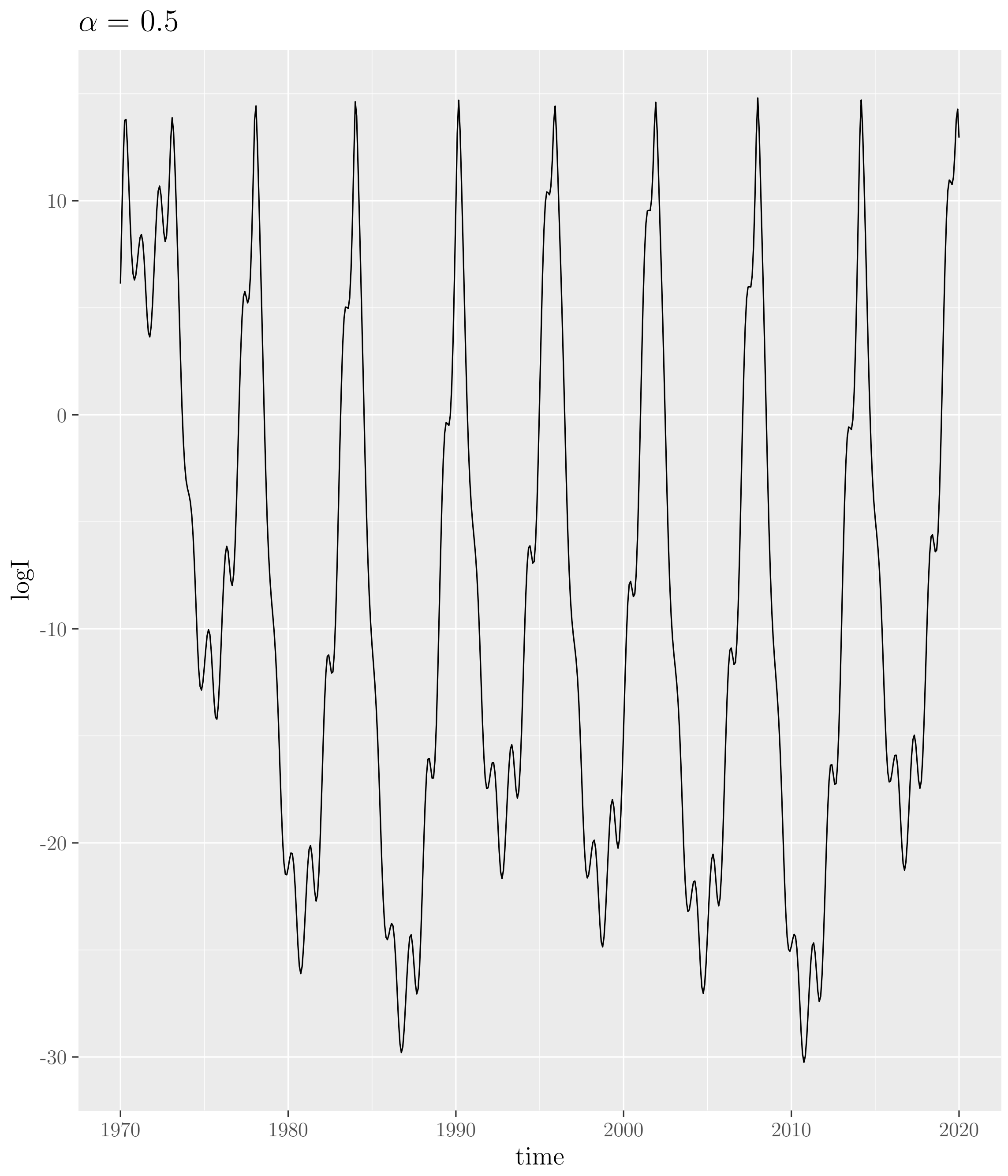


Figure 36: The deterministic solutions to the model for ranging values of α. From these plots we observe an annual cycle for α values of 0.025, 0.1 and 0.4, triennial for 0.2 and 0.3 and chaos for the rest.

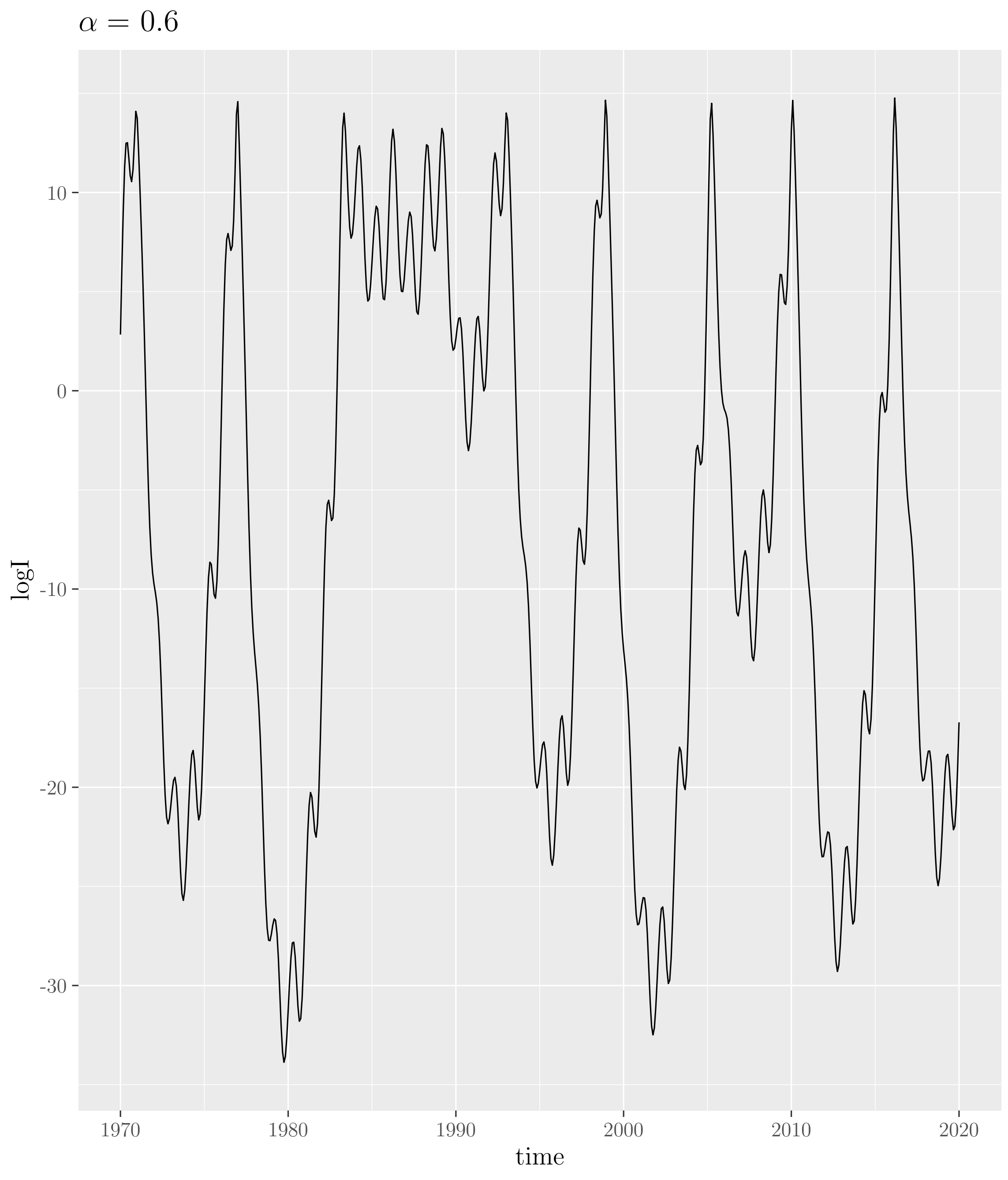


Figure 37: The deterministic solutions to the model for ranging values of α. From these plots we observe an annual cycle for α values of 0.025, 0.1 and 0.4, triennial for 0.2 and 0.3 and chaos for the rest.

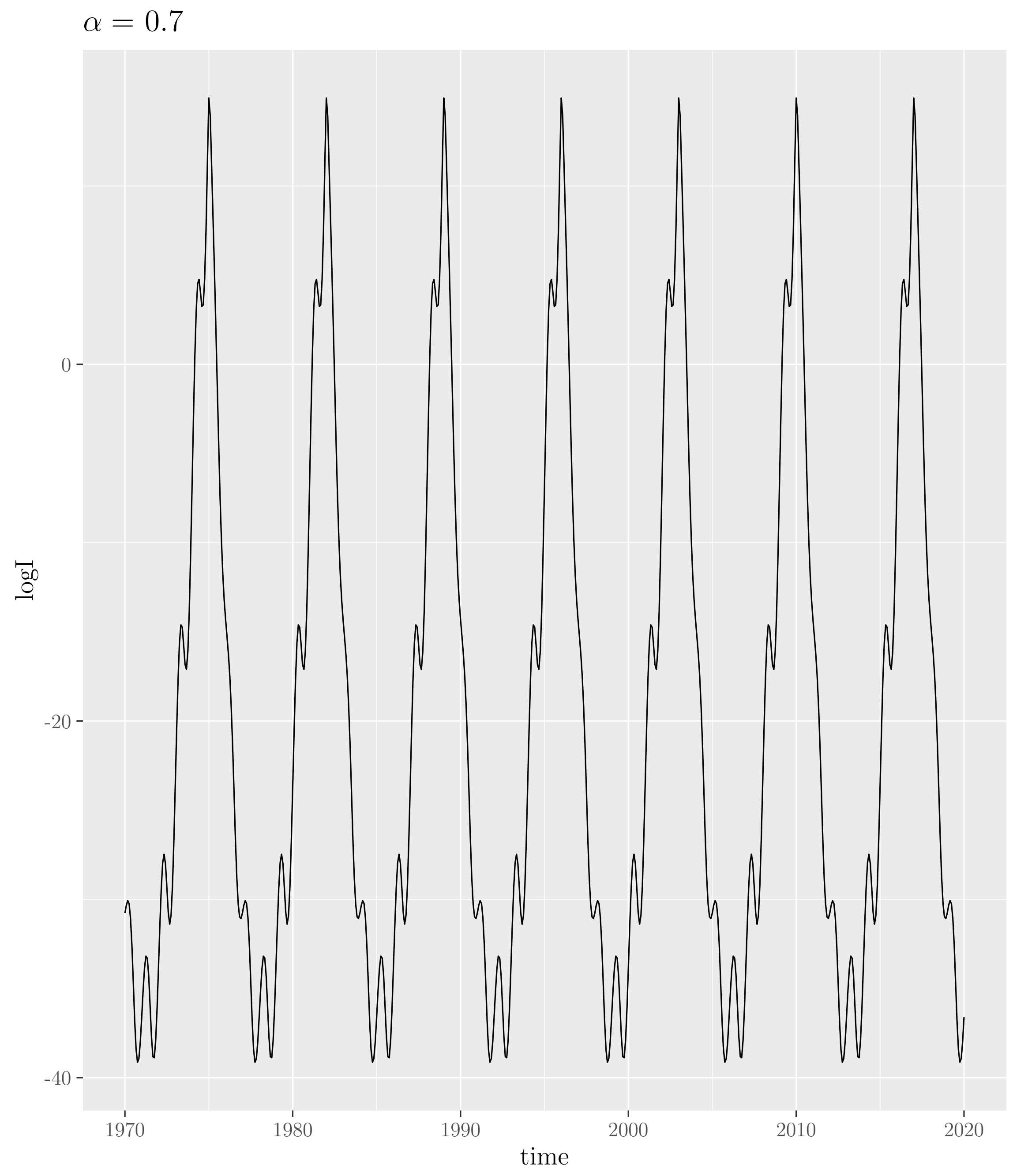


Figure 38: The deterministic solutions to the model for ranging values of α. From these plots we observe an annual cycle for α values of 0.025, 0.1 and 0.4, triennial for 0.2 and 0.3 and chaos for the rest.

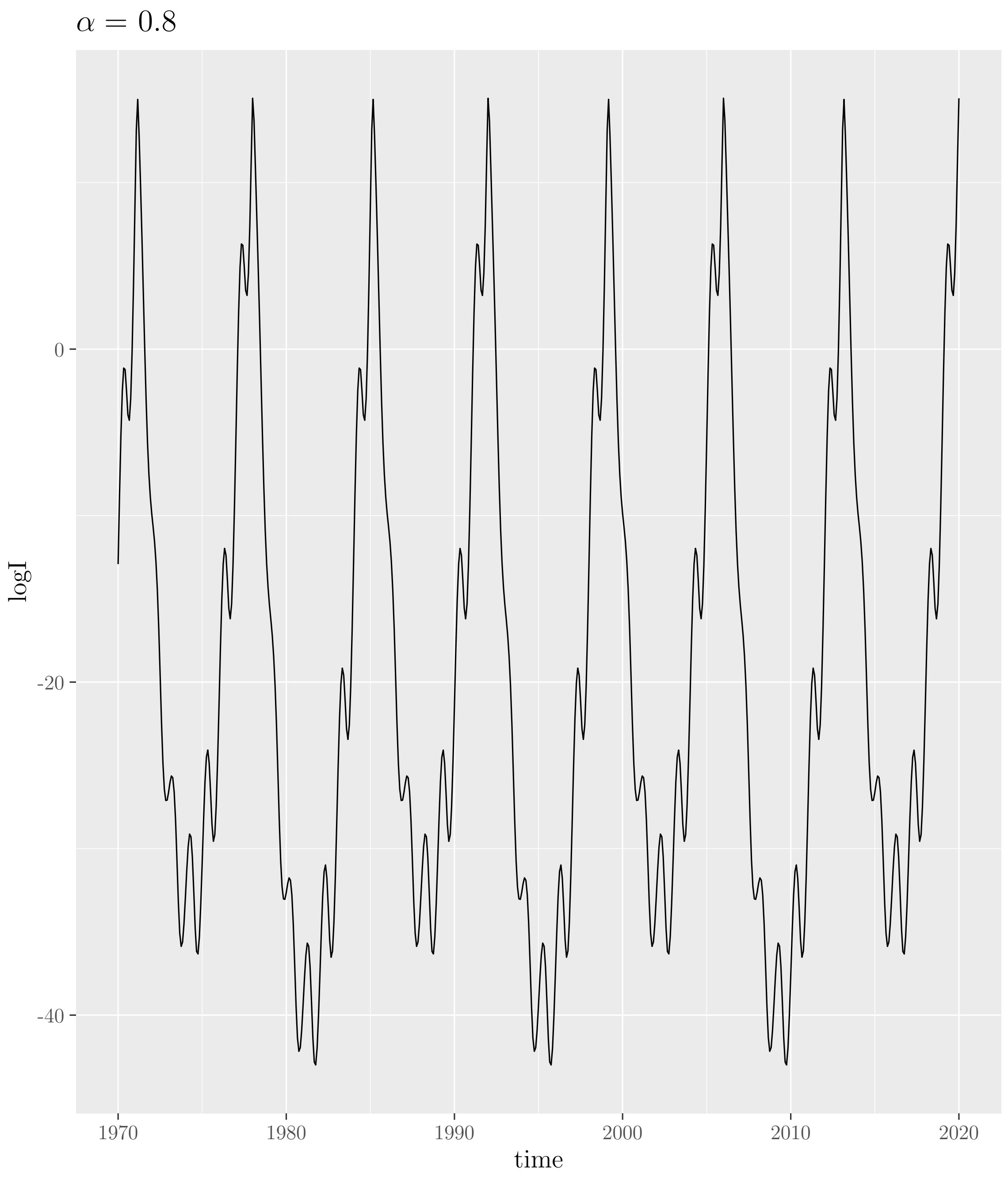


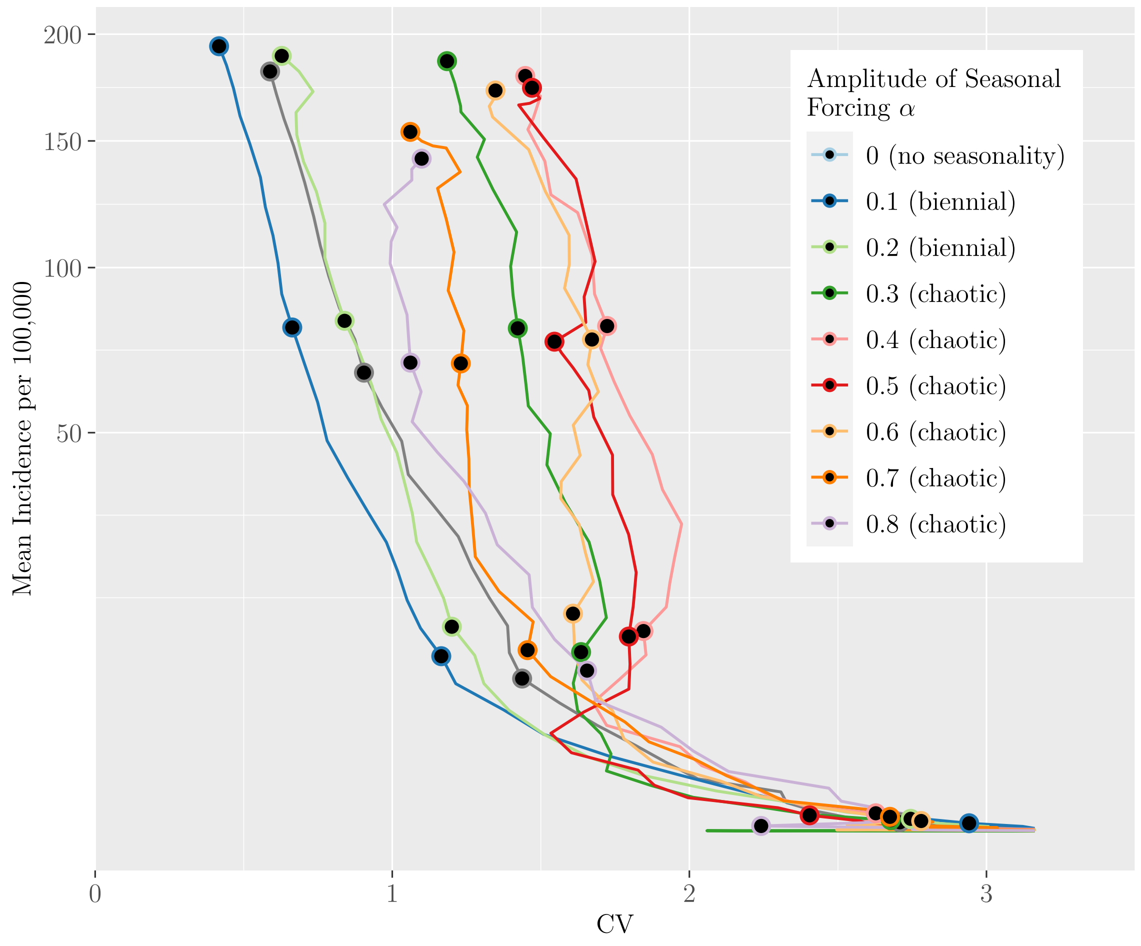
Figure 39: The deterministic solutions to the model for ranging values of α. From these plots we observe an annual cycle for α values of 0.025, 0.1 and 0.4, triennial for 0.2 and 0.3 and chaos for the rest.

From these plots we observe an annual cycle for α of 0.025, a biennial cycle for 0.1

and 0.2, and chaos for the rest.

With the data prepared, we create the figure displayed in figure 20.

```
## Define the palette used for the ggplot (from Paired brewer palette)
my_palette_alpha <- c("#A6CEE3", "#1F78B4","#B2DF8A","#33A02C",
                      "#FB9A99","#E31A1C","#FDBF6F","darkorange1",
                      "#CAB2D6")
names(my_palette_alpha) <- alpha_vec

## Create the canonical path
canonical_path_alpha_reports <- ggplot()+
  geom_path(data = data_us_reports,
        aes(CV, Incidence, col = Alpha),
        lty = 1, lwd = 1) +
  geom_point(data = point_data_us_reports,
           aes(CV, Incidence, col = Alpha),
           shape = 21, fill = "black",
           size = 2, stroke = 1.5) +
  theme(
    legend.position = c(.95, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6),
    legend.text = element_text(size = text_size),
    legend.title = element_text(size = text_size),
    axis.text = element_text(size = text_size),
    axis.title = element_text(size = text_size)
  )+
  scale_y_sqrt("Mean Incidence per 100,000",
               limits = c(min(data_us_reports$Incidence),
                          max(data_us_reports$Incidence))) +
  scale_x_continuous("CV", limits = c(0,3.5), expand = c(0,0)) +
  scale_colour_manual(values = my_palette_alpha,
                      labels = c("0 (no seasonality)",
                                 "0.1 (biennial)",
                                 "0.2 (biennial)",
                                 "0.3 (chaotic)",
                                 "0.4 (chaotic)",
                                 "0.5 (chaotic)",
                                 "0.6 (chaotic)",
                                 "0.7 (chaotic)",
                                 "0.8 (chaotic)"
  )) +
  labs(colour = "Amplitude of Seasonal \n Forcing $\\alpha$") +
  guides(colour = guide_legend(override.aes = list(linetype=1, size=1)))
canonical_path_alpha_reports
```

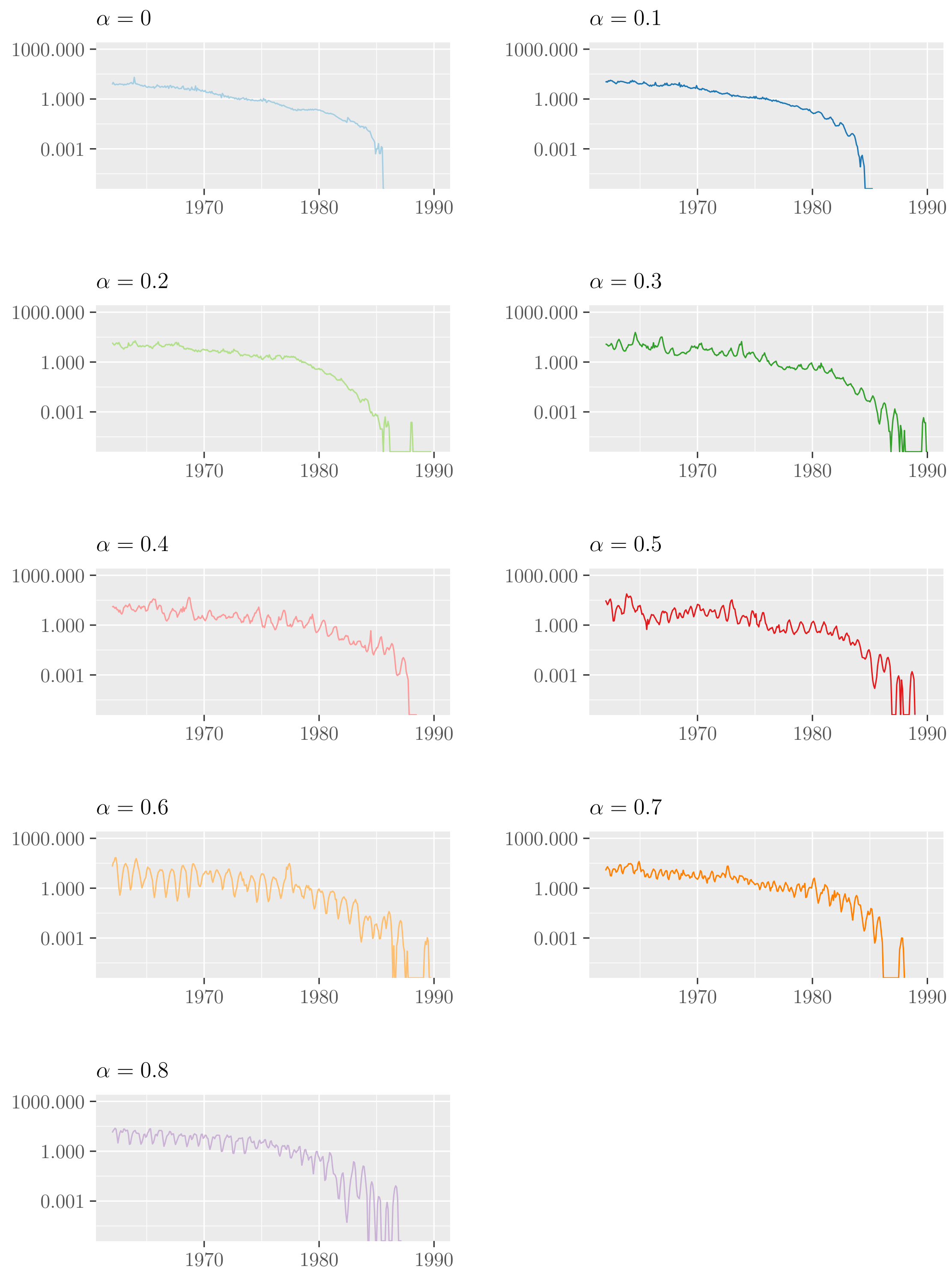


Next, we create plots of the time series on a monthly scale for each of the values of alpha, and plot them individually, as presented in figure 21.

```
## Define arguments used to subset the time series data
window <- 10
step <- 1/12
ts_list_alphas <- as.list(vector(length = length(alpha_vec)))
names(ts_list_alphas) <- alpha_vec
start_plot_year <- final_data[start_plot]
ts_list_short <- as.list(vector(length = length(alpha_vec)))
```

```r
## Plot the both the full time series for each value of alpha
## and the time series for 5 years

for (i in 1:length(monthly_reports_states_alphas)) {
  monthly_data <- monthly_reports_states_alphas[[i]]
  monthly_data$Year <-
    seq(final_data[1]-window+step,
        final_data[length(final_data)], step)

  monthly_data_long <-
    pivot_longer(monthly_data, cols = !Year,
                 names_to = "State",
                 values_to = "Reports")

  monthly_data_long$Alpha <- as.character(alpha_vec[i])

  mean_only <- monthly_data_long %>%
    filter(State == "mean" & Year >= as.numeric(start_plot_year))

  ts_plot_alpha <- ggplot() +
    geom_path(data = mean_only,
              aes(Year, Reports, col = Alpha)) +
    scale_y_log10(limits = c(0.00001, 1000),
                  labels =
                    function(x) format(x, scientific = FALSE)) +
    scale_colour_manual(values = my_palette_alpha) +
    ylab("") +
    xlab("") +
    xlim(c(1962, 1990)) +
    ggtitle(paste("$\\alpha=", alpha_vec[i], "$", sep = "")) +
    theme(
      plot.title = element_text(size = 10, face = "bold"),
      legend.position = "none")


  ts_list_alphas[[i]] <- ts_plot_alpha
}
```

```r
alpha_plot_list <- ts_list_alphas

ggarrange(plotlist = alpha_plot_list,
          ncol = 2, nrow = 5,
          common.legend = TRUE,
          legend = "none")
```


$\alpha = 0$
$\alpha = 0.1$
$\alpha = 0.2$
$\alpha = 0.3$
$\alpha = 0.4$
$\alpha = 0.5$
$\alpha = 0.6$
$\alpha = 0.7$
$\alpha = 0.8$
1000.000
1.000
0.001
1970
1980
1990

We also save the results to a *.RData* file.

```
if (use_immigration) {
  if (init_run == 0) {
    path_file <-
      paste(paste("data/canonical_path_alpha",
                  "imm", "sensitivity", sep = "_"),
            ".RData", sep = "")
  } else {
    path_file <-
      paste(paste("data/canonical_path_alpha",
                  init_run,"imm", "sensitivity", sep = "_"),
            ".RData", sep = "")
  }

} else {
  if (init_run == 0) {
    path_file <- "data/canonical_path_alpha_sensitivity.RData"
  } else {
    path_file <- paste(paste("data/canonical_path_alpha",
                             init_run, "sensitivity", sep = "_"),
                       ".RData", sep = "")
  }
}

save(canonical_path_alpha_reports, data_us_reports,
     point_data_us_reports, my_palette_alpha, ts_list_alphas,
     file = path_file)
```

7.7 Canonical Paths for Varying Natural History of Infection

The final analysis conducted in the paper investigated how changing the mean infectious and latent periods affect the trajectory of the path. Thus, we began by simulating the path for the following range of mean infectious periods: 2, 4, 8, . . . , 32 days. We use the `path_model` function, as described in section 7.5.2.2, to simulate the yearly incidence per 100,000 and the `cv_inc_fun`, as described in section 7.3.1, to create the data frames containing weighted incidence and CV for each value of the mean infectious period. The simulation is conducted in `model_sensitivity_gamma_new.R` as the code takes several hours to run. The calculation of weighted incidence and CV is conducted in `cv_inc_sensitivity_gamma_new.R` as it takes over an hour to run.

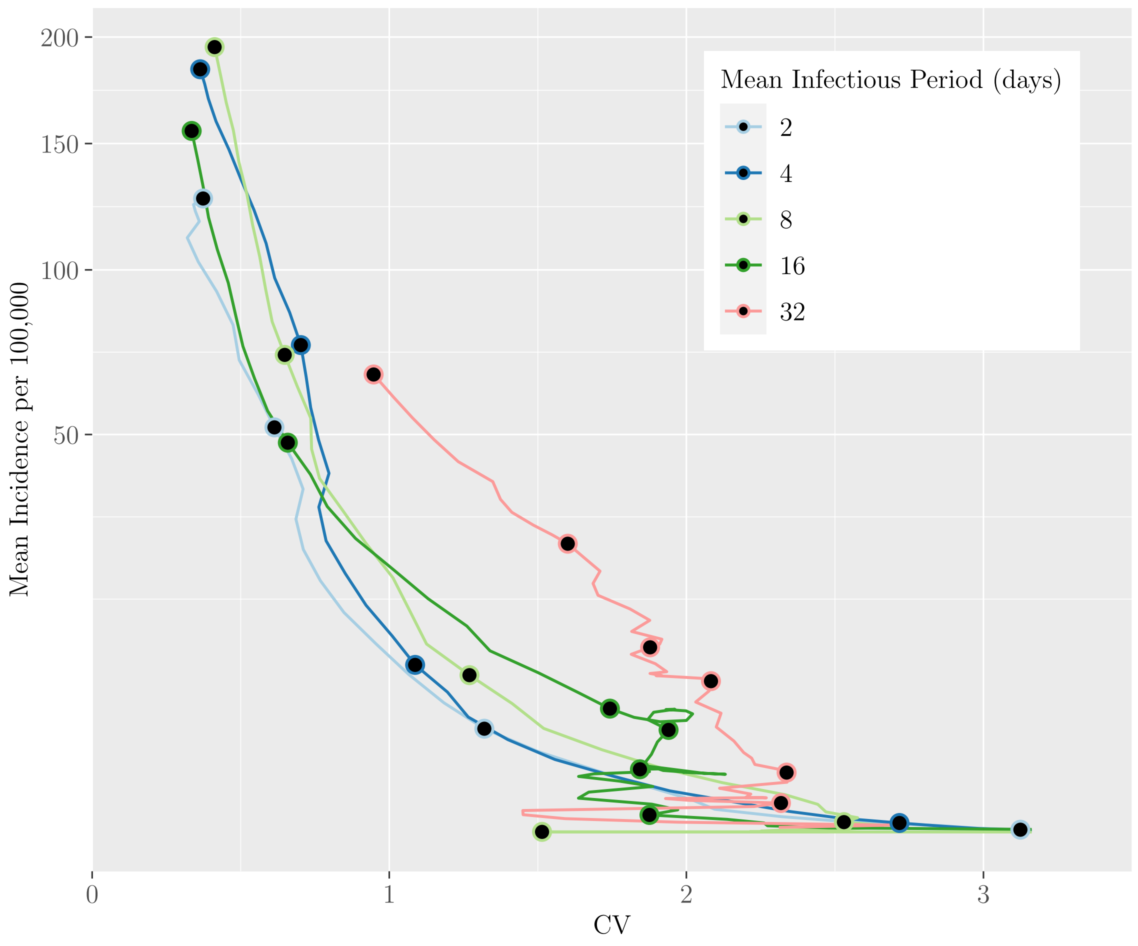
With the data frames created, we use the same methods described in section 7.6 to create figure 16.

```r
library(stats)
library(smoother)
library(calibrate)
library(tidyverse)
library(rlist)
library(RColorBrewer)
library(knitr)
library(ggpubr)
load("data/state_data_vacc_gamma_sensitivity_200.RData")
load("data/cv_inc_state_gamma_sensitivity_200.RData")
load("data/presentation_logical.RData")
```

```r
## Determine at which data point the path should be plotted
## (want to plot after the extra run time is complete)
use_immigration <- TRUE
init_run <- 200
if (init_run == 0) {
  start_plot <- 1
} else {
  start_plot <- init_run + 1
}
```

```r
## Filter the data based on the year we want to begin
## plotting based on start_plot
years <- unique(data_us$year)
final_years <- years[start_plot:length(years)]

data_us_reports <- data_us_reports %>%
  filter(year %in% final_years)

## Make the infectious period a factor
data_us_reports$t_infec <-
  as.factor(data_us_reports$t_infec)
point_data_us_reports$t_infec <-
  as.factor(point_data_us_reports$t_infec)
```

```r
## Define the palette used for the ggplot
## (from Paired brewer palette)
my_palette_gamma <- c("#A6CEE3","#1F78B4","#B2DF8A","#33A02C",
                      "#FB9A99")
t_infec <- unique(data_us_reports$t_infec)
names(my_palette_gamma) <- levels(t_infec)

## Change column names
colnames(data_us_reports) <-
  c("Gamma", "CV", "Incidence", "Year", "T_Infec")
colnames(point_data_us_reports) <-
  c("Gamma", "CV", "Incidence", "Year", "T_Infec")

canonical_path_gamma_reports <- ggplot()+
  geom_path(data = data_us_reports,
            aes(CV, Incidence, col = T_Infec), lty = 1, lwd = 1) +
  geom_point(data = point_data_us_reports,
             aes(CV, Incidence, col = T_Infec),
             shape = 21, fill = "black", size = 2, stroke = 1.5) +
  theme(
    legend.position = c(.95, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6),
    legend.text = element_text(size = text_size),
    legend.title = element_text(size = text_size),
    axis.text = element_text(size = text_size),
    axis.title = element_text(size = text_size)
  )+
  scale_y_sqrt("Mean Incidence per 100,000",
               limits = c(min(data_us_reports$Incidence),
                          max(data_us_reports$Incidence))) +
  scale_x_continuous("CV", limits = c(0,3.5), expand = c(0,0)) +
  scale_colour_manual(values = my_palette_gamma) +
  labs(color='Mean Infectious Period (days)') +
  guides(colour = guide_legend(override.aes = list(linetype=1, size=1)))
canonical_path_gamma_reports
```

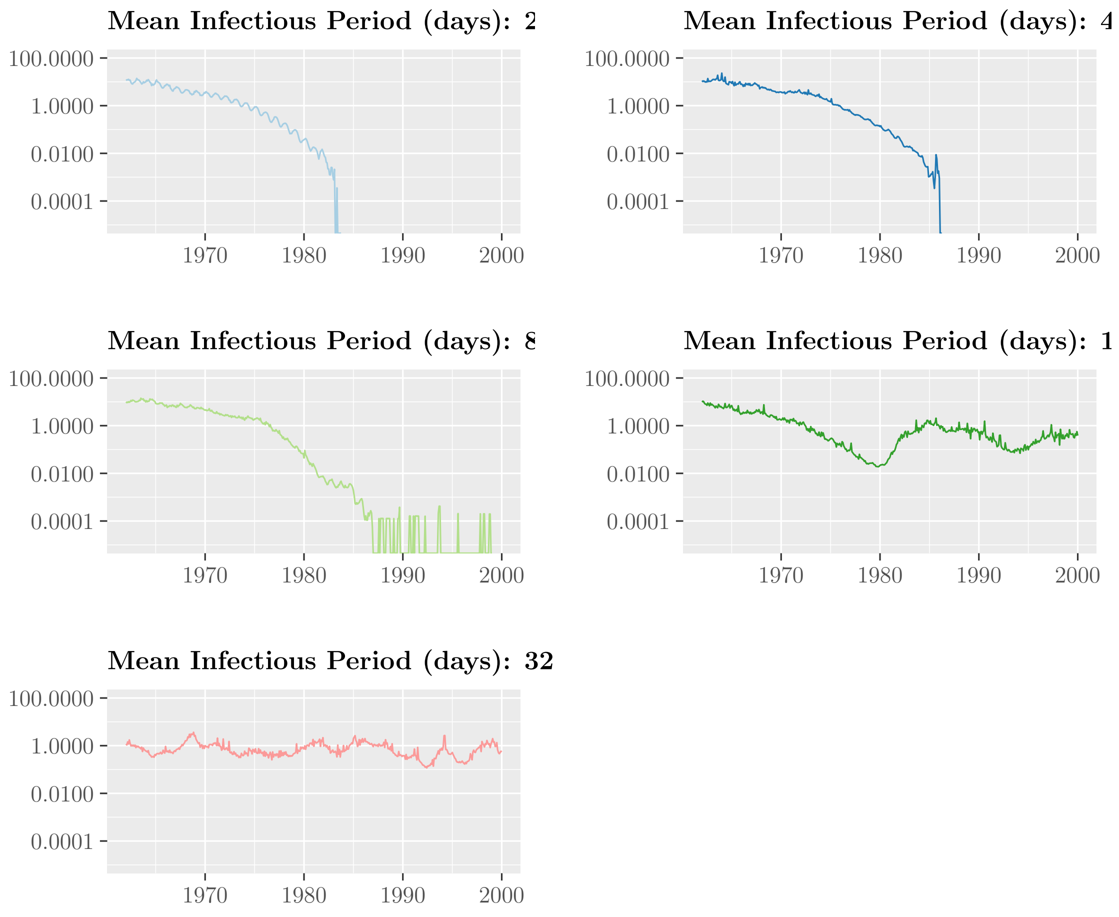


Next, we create plots of the time series on a monthly scale for each of the values of gamma, as presented in figure 17.

```
window <- 10
step <- 1/12
ts_list_gamma <- as.list(vector(length = length(gamma_vec)))
names(ts_list_gamma) <- gamma_vec
start_plot_year <- final_data[start_plot]

for (i in 1:length(monthly_reports_states_gamma)) {
  monthly_data <- monthly_reports_states_gamma[[i]]
  monthly_data$Year <-
    seq(final_data[1]-window+step,
        final_data[length(final_data)],
        step)
  monthly_data_long <-
    pivot_longer(monthly_data,
                 cols = !Year,
                 names_to = "State",
                 values_to = "Reports")

  monthly_data_long$t_infec <- t_infec[i]

  mean_only <- monthly_data_long %>%
    filter(State == "mean" & Year >= start_plot_year)

  ts_plot_gamma <- ggplot() +
    geom_path(data = mean_only, aes(Year, Reports, col = t_infec)) +
    scale_y_log10(limits = c(0.00001, 100),
                  labels =
                    function(x) format(x, scientific = FALSE)) +
    scale_colour_manual(values = my_palette_gamma) +
    ylab("") +
    xlab("") +
    labs(colour = "Mean Infectious Period") +
    xlim(c(1962, 2000)) +
    ggtitle(paste("Mean Infectious Period (days): ",
                  365/gamma_vec[i], sep = "")) +
    theme(
      plot.title = element_text(size = 10, face = "bold"),
      legend.position = "none")

  ts_list_gamma[[i]] <- ts_plot_gamma
}
```

We plot the time series for each gamma in a grid using the following code.

```
gamma_plot_list <- ts_list_gamma

ggarrange(plotlist = gamma_plot_list,
          ncol = 2, nrow = 3,
          common.legend = TRUE,
          legend = "none")
```



Additionally, we save the results to a *.RData* file.

```
if (init_run == 0) {
  sim_file <- "data/canonical_path_gamma_sensitivity.RData"
} else {
  sim_file <-
    paste(paste("data/canonical_path_gamma",
                "sensitivity", init_run, sep = "_"),
          ".RData", sep = "")
}

save(canonical_path_gamma_reports, data_us_reports,
     point_data_us_reports, ts_list_gamma, file = sim_file)
```

Lastly, we simulated the path for the following range of mean latent periods: 2, 4, 8,

. . . , 32 days. We use the `path_model` function, as described in section 7.5.2.2, to simulate the yearly incidence per 100,000 and the `cv_inc_fun`, as described in section 7.3.1, to create the data frames containing weighted incidence and CV for each value of the mean latent period. The simulation is conducted in `model_sensitivity_sigma_new.R` as the code takes several hours to run. The calculation of weighted incidence and CV is conducted in `cv_inc_sensitivity_sigma_new.R` as it takes over an hour to run.

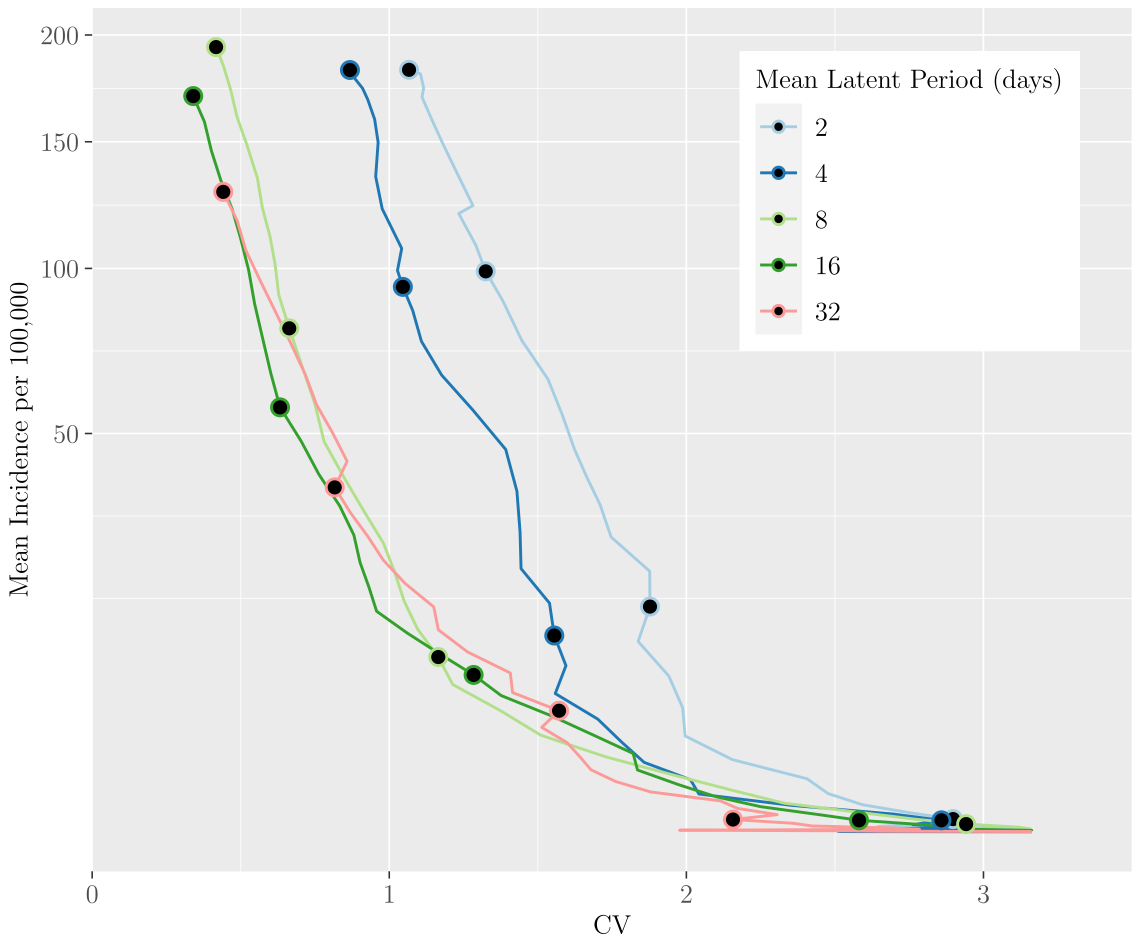
With the data frames created, we use the same methods described in section 7.6 to create figure 18.

```
library(stats)
library(smoother)
library(calibrate)
library(tidyverse)
library(rlist)
library(RColorBrewer)
library(knitr)
library(ggpubr)
load("data/state_data_vacc_sigma_sensitivity_200.RData")
load("data/cv_inc_state_sigma_sensitivity_200.RData")
load("data/presentation_logical.RData")
```

```
## Determine at which data point the path should be plotted
## (want to plot after the extra run time is complete)
use_immigration <- TRUE
init_run <- 200
if (init_run == 0) {
  start_plot <- 1
} else {
  start_plot <- init_run + 1
}
```

```
## Filter the data based on the year we want to begin plotting
## based on start_plot
years <- c(unique(data_us$year))
final_years <- years[start_plot:length(years)]

data_us_reports <- data_us_reports %>%
  filter(year %in% final_years)
```

```r
## Define the palette used for the ggplot
## (from Paired brewer palette)
my_palette_sigma <- c("#A6CEE3","#1F78B4","#B2DF8A","#33A02C",
                      "#FB9A99")

t_latent <- unique(data_us_reports$t_latent)
names(my_palette_sigma) <- levels(t_latent)

## Change column names
colnames(data_us_reports) <-
  c("Sigma", "CV", "Incidence", "Year", "t_latent")
colnames(point_data_us_reports) <-
  c("Sigma", "CV", "Incidence", "Year", "t_latent")

canonical_path_sigma_reports <- ggplot()+
  geom_path(data = data_us_reports,
            aes(CV, Incidence, col = t_latent),
            lty = 1, lwd = 1) +
  geom_point(data = point_data_us_reports,
             aes(CV, Incidence, col = t_latent),
             shape = 21, fill = "black", size = 2, stroke = 1.5) +
  theme(
    legend.position = c(.95, .95),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6),
    legend.text = element_text(size = text_size),
    legend.title = element_text(size = text_size),
    axis.text = element_text(size = text_size),
    axis.title = element_text(size = text_size)
  )+
  scale_y_sqrt("Mean Incidence per 100,000",
               limits = c(min(data_us_reports$Incidence),
                          max(data_us_reports$Incidence))) +
  scale_x_continuous("CV", limits = c(0,3.5), expand = c(0,0)) +
  scale_colour_manual(values = my_palette_sigma) +
  labs(color='Mean Latent Period (days)') +
  guides(colour = guide_legend(override.aes = list(linetype=1, size=1)))
canonical_path_sigma_reports
```

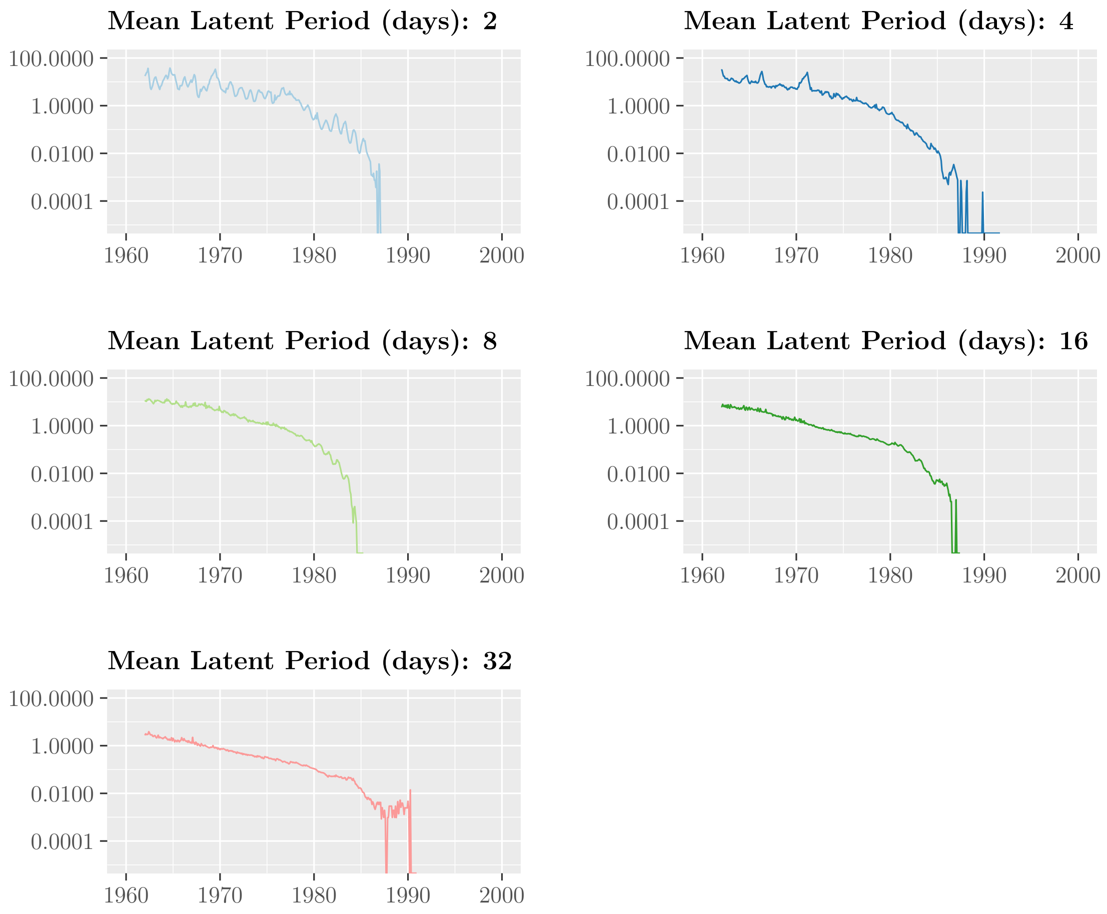


Next, we create plots of the time series on a monthly scale for each of the values of sigma.

```
window <- 10
step <- 1/12
ts_list_sigma <- as.list(vector(length = length(sigma_vec)))
names(ts_list_sigma) <- sigma_vec
start_plot_year <- final_data[start_plot]

for (i in 1:length(monthly_reports_states_sigma)) {
  monthly_data <- monthly_reports_states_sigma[[i]]
  monthly_data$Year <-
    seq(final_data[1]-window+step,
        final_data[length(final_data)],
        step)
  monthly_data_long <-
    pivot_longer(monthly_data,
                 cols = !Year,
                 names_to = "State",
                 values_to = "Reports")

  monthly_data_long$t_latent <- t_latent[i]

  mean_only <- monthly_data_long %>%
    filter(State == "mean" & Year >= start_plot_year)

  ts_plot_sigma <- ggplot() +
    geom_path(data = mean_only, aes(Year, Reports, col = t_latent)) +
    scale_y_log10(limits = c(0.00001, 100),
                  labels =
                    function(x) format(x, scientific = FALSE)) +
    scale_colour_manual(values = my_palette_sigma) +
    xlim(c(1960, 2000)) +
    ylab("") +
    xlab("") +
    labs(colour = "Mean Latent Period") +
    ggtitle(paste("Mean Latent Period (days): ",
                  365/sigma_vec[i], sep = "")) +
    theme(
      plot.title = element_text(size = 10, face = "bold"),
      legend.position = "none")

  ts_list_sigma[[i]] <- ts_plot_sigma
}
```

We plot the time series for each value of sigma in a grid using the following code, as presented in figure 19.

```
sigma_plot_list <- ts_list_sigma

ggarrange(plotlist = sigma_plot_list,
          ncol = 2, nrow = 3,
          common.legend = TRUE,
          legend = "none")
```



Additionally, we save the results to a *.RData* file.

```
if (init_run == 0) {
  sim_file <- "data/canonical_path_sigma_sensitivity.RData"
} else {
  sim_file <-
    paste(paste("data/canonical_path_sigma",
                          "sensitivity", init_run, sep = "_"),
          ".RData", sep = "")
}

save(canonical_path_sigma_reports,
     data_us_reports, point_data_us_reports,
     ts_list_sigma, file = sim_file)
```