Final project report: Dynamic regression forecasting state opioid overdose deaths in Virginia

Faysal Shaikh

CSI 678 - Spring 2022

Dynamic regression forecasting state opioid overdose deaths in Virginia

Faysal Shaikh CSI 678 - Spring 2022

Table of contents

List of figures

List of tables

1 Introduction

The purpose of this project is to create a prediction model for state opioid overdose deaths in Virginia using at most 2 predictors: unemployed persons and noninstitutionalized population. As such, we outline the opioid epidemic in the United States of America (U.S.), previous work relating socioeconomic (esp. economic) variables and opioid overdose deaths, and our modeling approach.

1.1 The U.S. opioid epidemic

Fradulent claims minimizing the addictive potential of opioid drugs from U.S. pharmaceutical companies in the 1990s have exhibited far-reaching consequences through the ensuing decades (U.S. Department of Health and Human Services, 2021). In fact the early days and months of the COVID-19 pandemic showcased a large spike in opioid overdose deaths (Haley & Saitz, 2020).

1.2 Using economic data for forecasting opioid overdose deaths

Many previous works have utilized socioeceonomic data (especially economic data) in producing forecasts for opioid overdose deaths (Yang et al., 2021; Heyman et al., 2019; Aram et al., 2020; Mensah et al., 2021; Brown and Wehby, 2019; Altekruse et al., 2020). We hope to both follow in this path while also contributing to the scholarly conversation in this area by implementing our desired approach following approaches learned from a time-series analysis & forecasting course.

1.3 Modeling approach: Dynamic [harmonic] regression

Our goal is to utilize time-series predictors (including covariates) to model a time-series outcome (or commonly-referred to as a ”forecast variable” in time-series analysis) via a linear regression with the specific structure of the error term following an AutoRegressive Integrated Moving Average (ARIMA) model. This general technique is referred to as dynamic regression (Hyndman & Athanasopoulos, 2021i).

The general description of an ARIMA model typically includes a nonseasonal component with p lag-order, d order of differencing, and q order of moving-average; for time-series data with seasonality, model description can also include a distinct seasonal component (of seasonal period m) with P lag-order, D order of differencing, and Q order of moving-average. This is typically notated as follows:

$$\text{ARIMA}(p, d, q)(P, D, Q)_m$$

Thus, a dynamic regression model with 2 predictors X_1 and X_2 with the structure of the error term as an ARIMA model capturing compound seasonal and non-seasonal time-series information would be notated as follows:

$$Y = \beta_1 X_1 + \beta_2 X_2 + \varepsilon, \text{ where } \varepsilon \sim \text{ARIMA}(p, d, q)(P, D, Q)_m$$

The above structure (including at most 2 predictors) will serve as the basis for the majority of models we examine in this work.

However, an alternative approach to handling various orders of seasonality is to utilize additional Fourier series terms as regression predictors and then re-specify the structure of the error term as an

ARIMA model handling exclusively non-seasonal time-series information; this approach is referred to as dynamic harmonic regression (Hyndman & Athanasopoulos, 2021a). We include several dynamic harmonic regression models in this work for comparison purposes with the aforementioned $(P, D, Q)_m$ seasonal-ARIMA.

1.4 Big picture: Prediction model(s) with live "crisis event" tracking

By many criteria the COVID-19 pandemic can be considered a "crisis event," disrupting many factors (including socioeconomic factors) and thus resulting in (via some unspecified causal mechanism) a spike in opioid overdose deaths. Our hope is that by demonstrating the utility of using targeted factors (especially socioeconomic factors) in the early phases of "crisis events" to adjust forecasts of opioid overdose deaths, that larger subsets of factors could be used to build a model capable of detecting more general "crisis events" and developing adjusted overdose forecasts accordingly (and thus intervene accordingly). We hope that if these techniques are useful that public health departments could gain from implementing these techniques in their drug overdose surveillance frameworks.

2 Data

Following our survey of previous literature, we identify sources for long-term opioid overdose death data and long-term unemployment data (both collected on a monthly basis). We then outline our data wrangling procedure, provide exploratory data visualizations, and execute necessary feature engineering procedures.

2.1 Data sources

2.1.1 Centers for Disease Control and Prevention (CDC)

In searching for opioid overdose death data, we identified the Centers for Disease Control and Prevention Wide-ranging ONline Data for Epidemiologic Research (CDC WONDER) source as suitable for our purposes (Centers for Disease Control and Prevention, n.d.). In terms of specific underlying cause-of-death and multiple cause-of-death coding via the International Classification of Diseases 10th Revision (ICD-10), we followed a generally-accepted combination of criteria that characterize opioid overdose deaths (Kaiser Family Foundation, n.d.).

2.1.2 U.S. Bureau of Labor Statistics (BLS)

In searching for unemployment data, we identified the U.S. Bureau of Labor Statistics (BLS) source as suitable for our purposes (U.S. BUREAU OF LABOR STATISTICS, n.d.). We were able to find a specific web page containing monthly state-level unemployment data via the search engine tool and downloaded the text file labeled "States and selected areas: Employment status of the civilian noninstitutional population, January 1976 to date, not seasonally adjusted (TXT)."

2.2 Data wrangling

Specific code segments are referenced in the Appendix to retain focus on the outcomes of this work rather than on the code. All data wrangling procedures were carried out via the R language for statistical computing version 4.1.3 (R Core Team, 2022) in the RStudio integrated development environment version 2022.02.1+461 (RStudio Team, 2022). Required package imports can be found in Appendix A.1 (p. 39). Code reading-in and preprocessing CDC WONDER data can be found in Appendix A.2.1 (p. 39) and Appendix A.3.1 (p. 40), respectively. Code reading-in and preprocessing BLS data can be found in Appendix A.2.2 (p. 39) and Appendix A.3.2 (p. 41).

2.3 Exploratory data visualization

We begin our process by visualizing our relevant variables in their initial, unprocessed forms.

2.3.1 Forecast variable: Opioid overdose deaths

Our main outcome variable obtained from the CDC WONDER data is monthly state opioid overdose deaths for Virginia ranging from a bit before January 2000 and through December 2020. In Figure 2.1 (p. 6) below we visualize this data in a standard timeplot with a vertical blue dashed-line at January 2000 representing a potential left-cutoff of data for training and a vertical red dashed-line at January 2020 representing a potential right-cutoff for train-test splitting (more on this later).

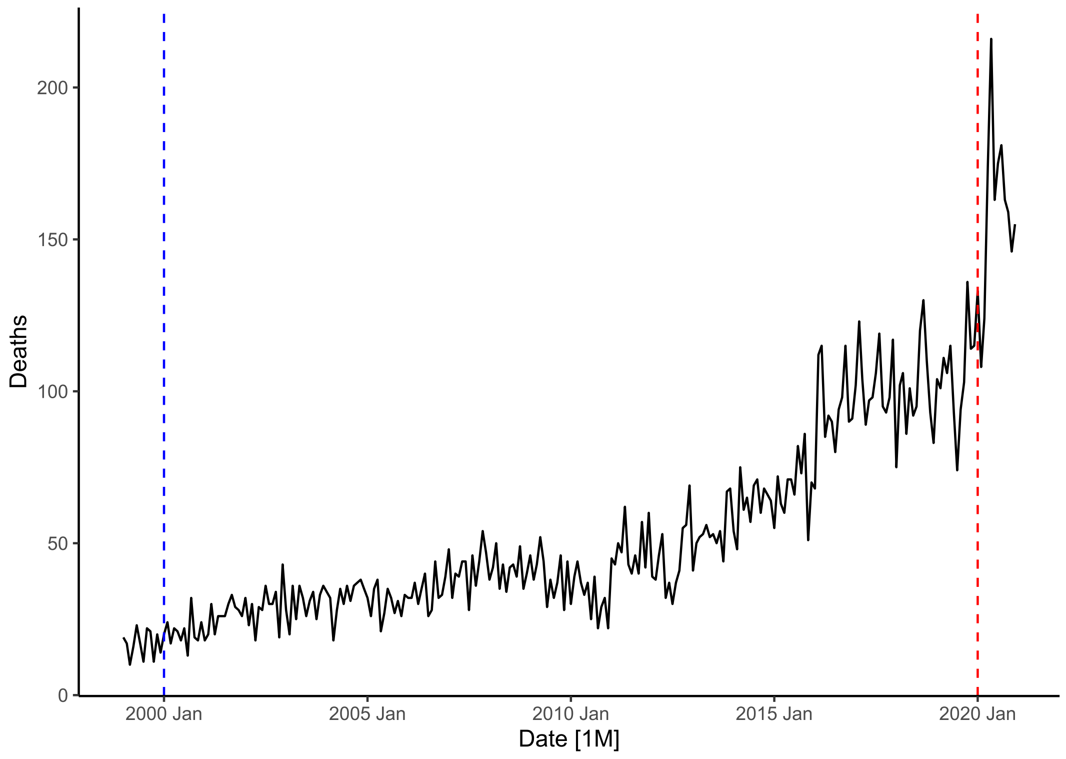

Figure 2.1: Timeplot of opioid overdose deaths

We can see an apparent spike in state opioid overdose deaths in the early months of the pandemic. We hope that we can use an additional predictor(s) along with the ARIMA error term to predict this particular spike. These are monthly data, so a suspected seasonality (period m=12 months) is visible in our seasonal and polar seasonal plots in Figure 2.2 (p. 8) and Figure 2.3 (p. 8), respectively. We additionally see an expected high degree of autocorrelation in the plot of the autocorrelation function (ACF) in Figure 2.4 (p. 9). We round out our exploratory data visualization for each variable with a subseries plot as shown in Figure 2.5 (p. 9) below; this subseries plot provides yet another view of the yearly seasonal pattern in our monthly data.

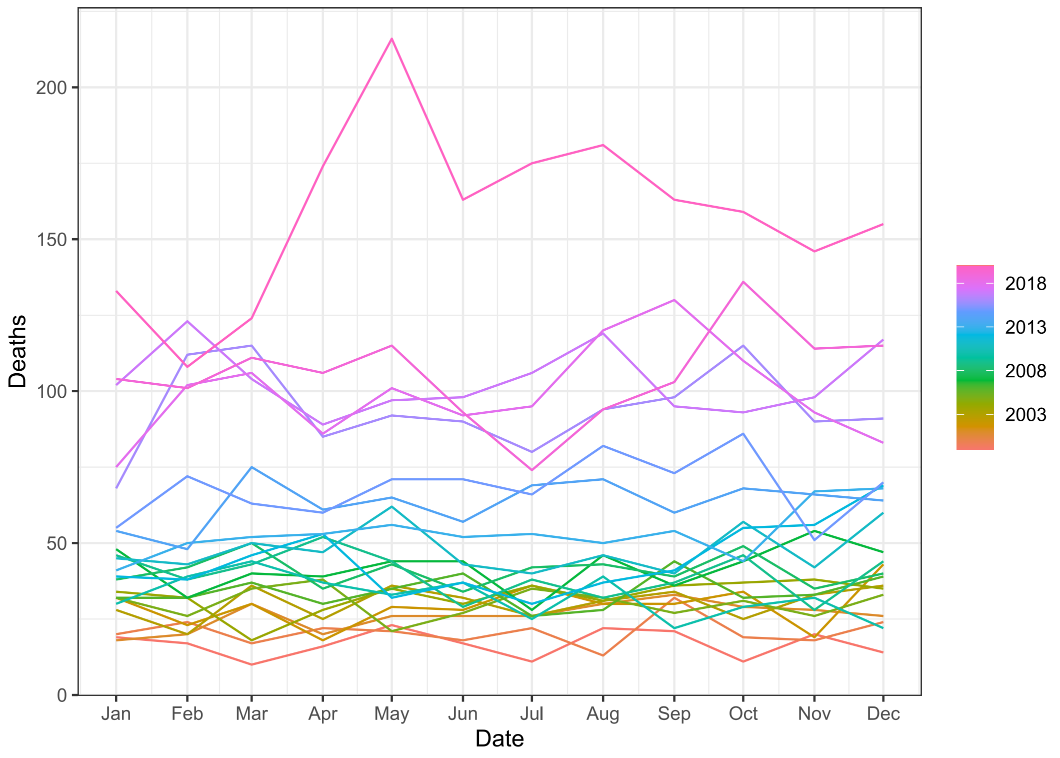

Figure 2.2: Seasonal plot of opioid overdose deaths

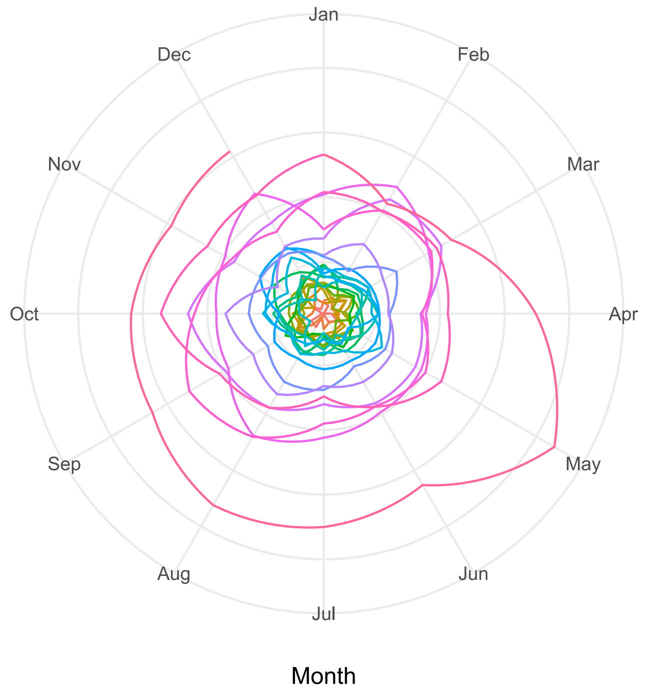

Figure 2.3: Polar seasonal plot of opioid overdose deaths

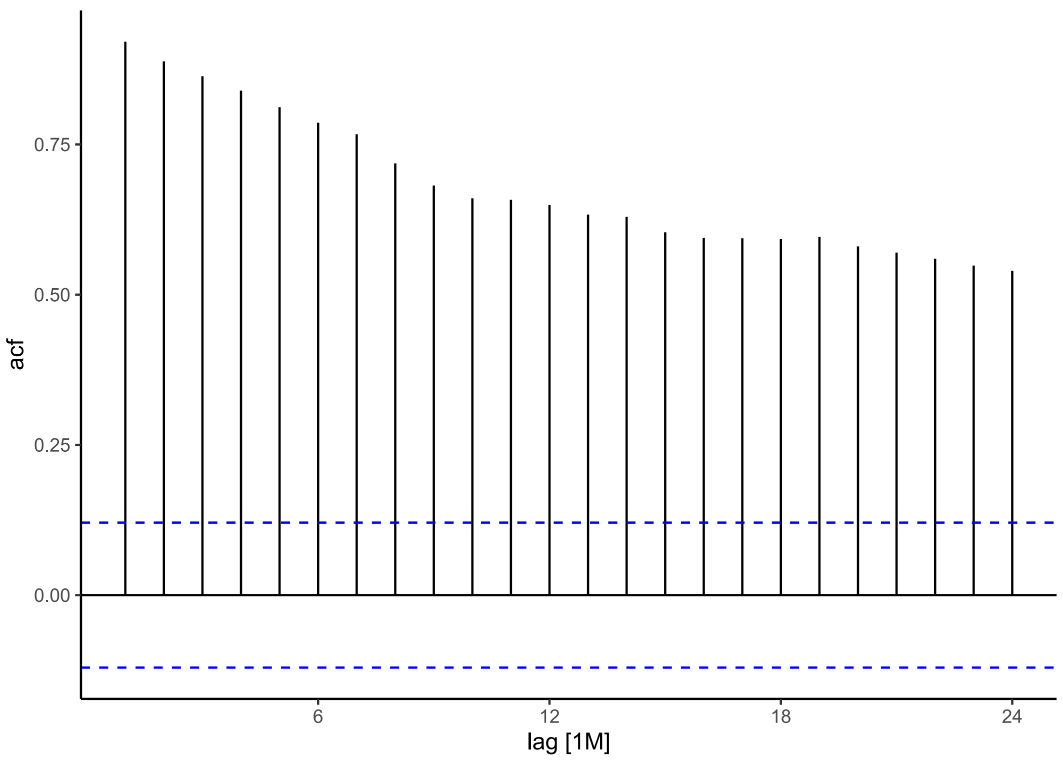

Figure 2.4: ACF plot of opioid overdose deaths

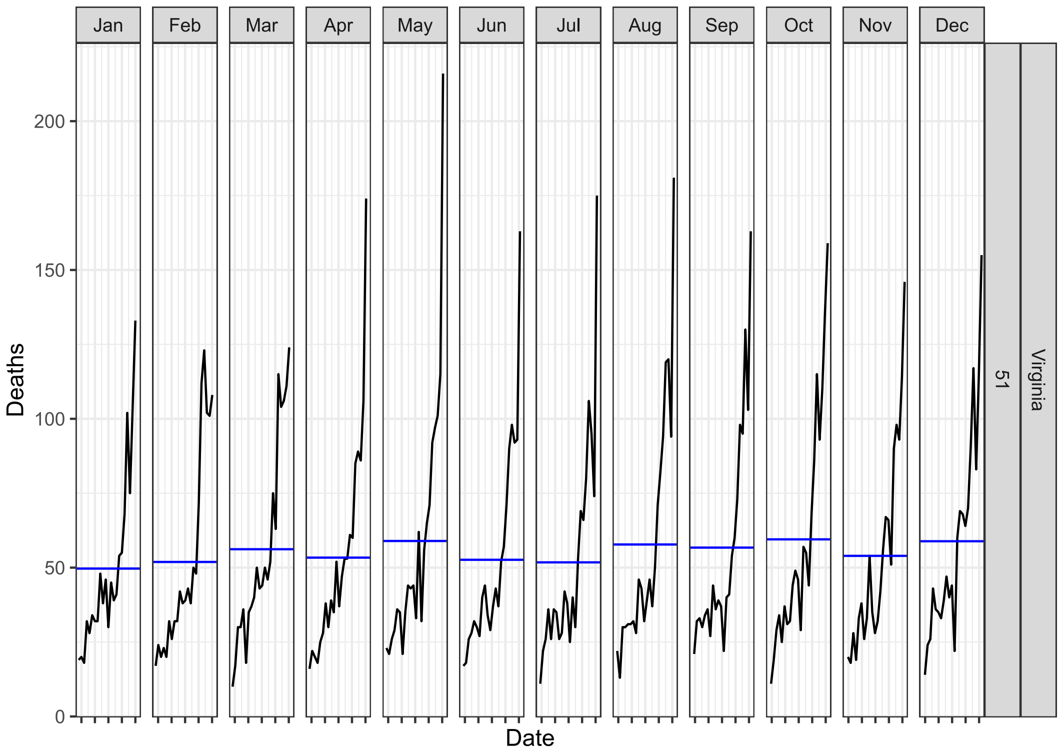

Figure 2.5: Subseries plot of opioid overdose deaths

2.3.2 Predictor variable: Unemployed

Our predictor variable of interest obtained from BLS data is monthly state unemployed persons counts for Virginia ranging from several decades before January 2000 and certainly through at least December 2020. In Figure 2.6 (p. 10) below we visualize this data in a standard timeplot our previously-described vertical dashed-lines.

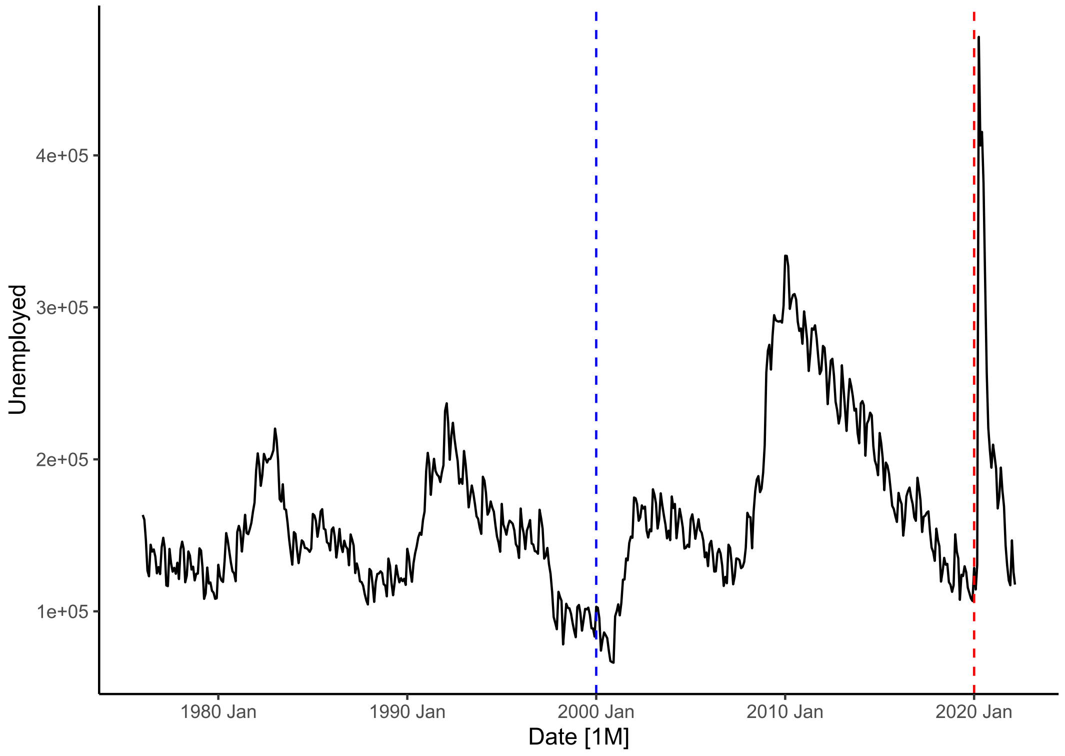

Figure 2.6: Timeplot of unemployed persons

We can notice an apparent spike in unemployment similar to the aforementioned spike in opioid overdose deaths in the early months of the pandemic. Seeing this spike reproduced in the unemployed persons measure gives us hope that this variable may serve as a suitable predictor in our model. We also see a stronger seasonal pattern of the monthly unemployment data than we saw in overdose deaths; this is reflected in the seasonal patterns visible in Figure 2.7 (p. 11), Figure 2.8 (p. 11), and Figure2.10 (p. 12) as well as in the ACF plot in Figure 2.9 (p. 12) below.

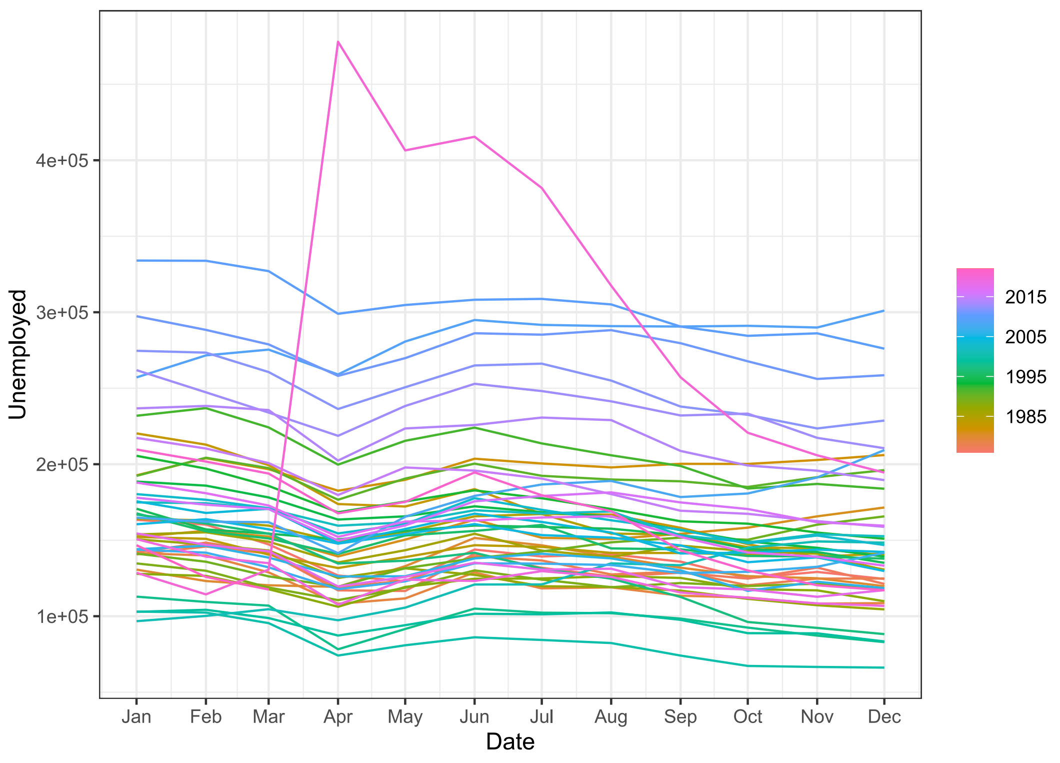

Figure 2.7: Seasonal plot of unemployed persons

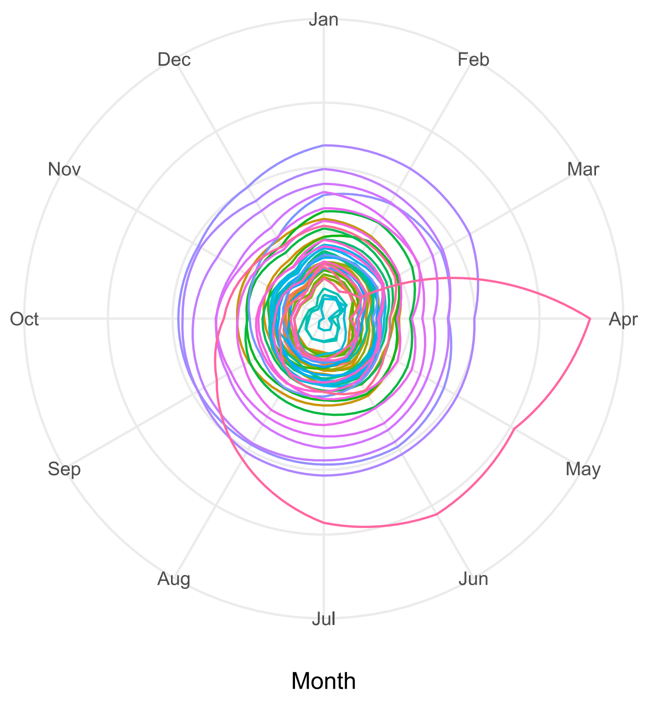

Figure 2.8: Polar seasonal plot of unemployed persons

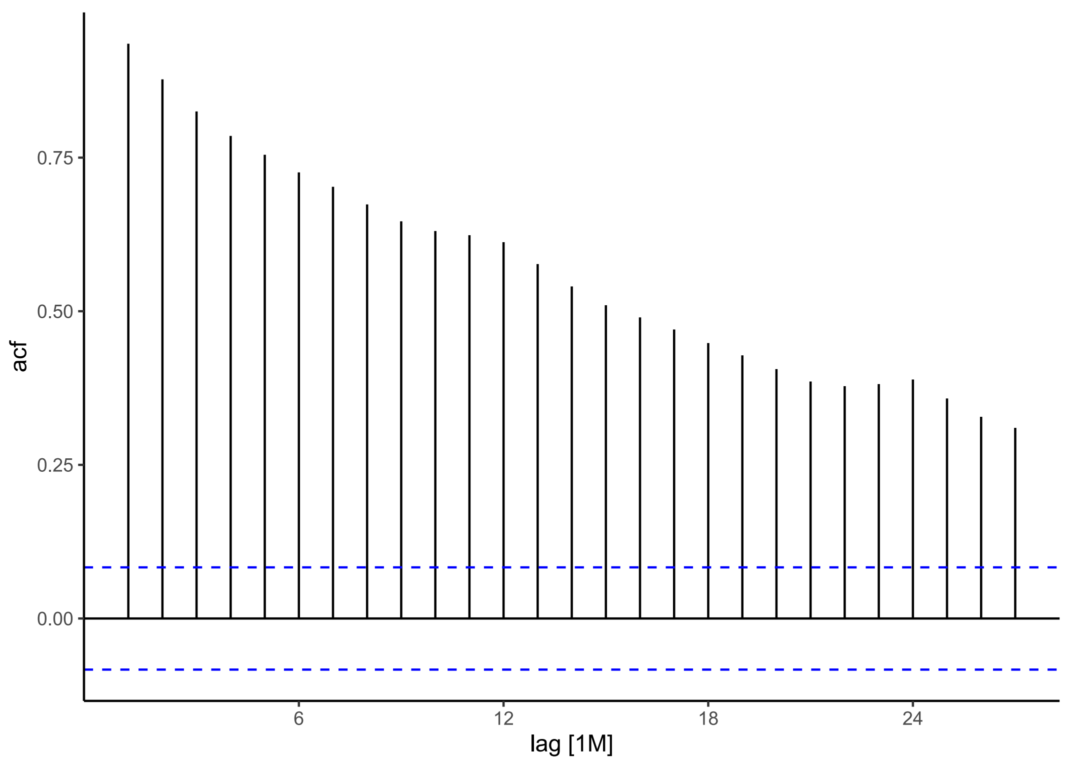

Figure 2.9: ACF plot of unemployed persons

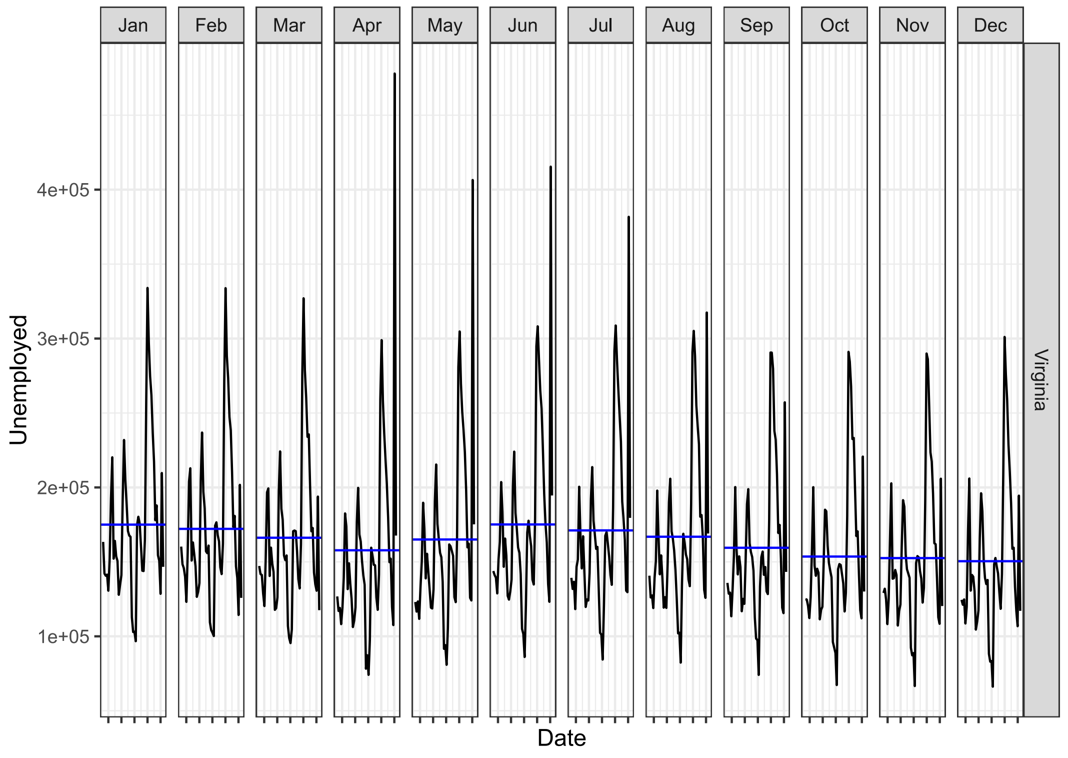

Figure 2.10: Subseries plot of unemployed persons

2.3.3 Covariate variable: Noninstitutionalized population

An additional variable of interest obtained from BLS data is monthly noninstitutionalized population counts for Virginia for the same time duration as our previously unemployed persons variable. Standard regressions typically have the main independent predictors of interest alongside relevant covariates (mathematically treated similarly to predictors but with less emphasis/interest in particular its contribution) icluded in a fully-specified model; as such, noninstutionalized population was utilized as either the single predictor or a relevant covariate for unemployment in model specification. In Figure 2.11 (p. 13) below we visualize this data in a standard timeplot our previously-described vertical dashed-lines.

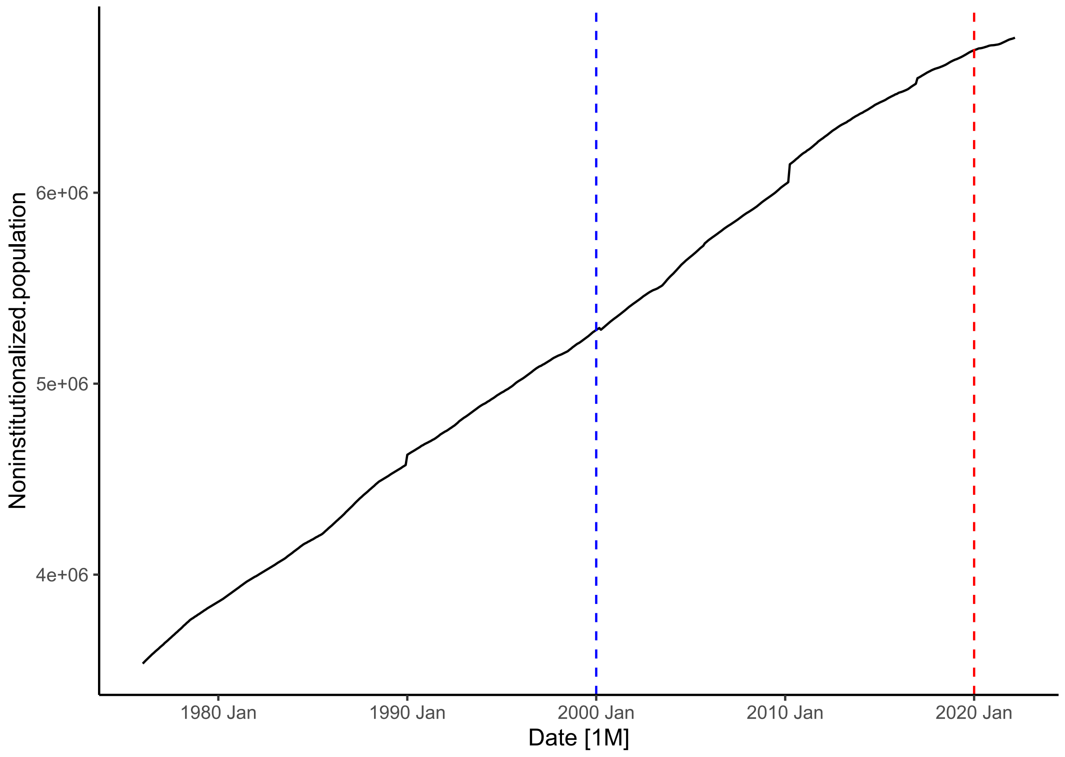

Figure 2.11: Timeplot of noninstitutionalized population

We do not notice an apparent spike in noninstitutionalized population in comparison with aforementioned spikes in opioid overdose deaths and unemployed persons in the early months of the pandemic. While we do not see a reproduction of this spike, we assume this variable helps the model understand the counts of unemployed persons as being based on the overall workforce (which is then downstream based on this noninstitutionalized population variable). As such, we are not too worried about the structure of this variable at this moment. Additional visualizations of this variable can be found in Figure 2.12 (p. 14), Figure 2.13 (p. 14), Figure 2.14 (p. 15), and Figure 2.15 (p. 15) below.

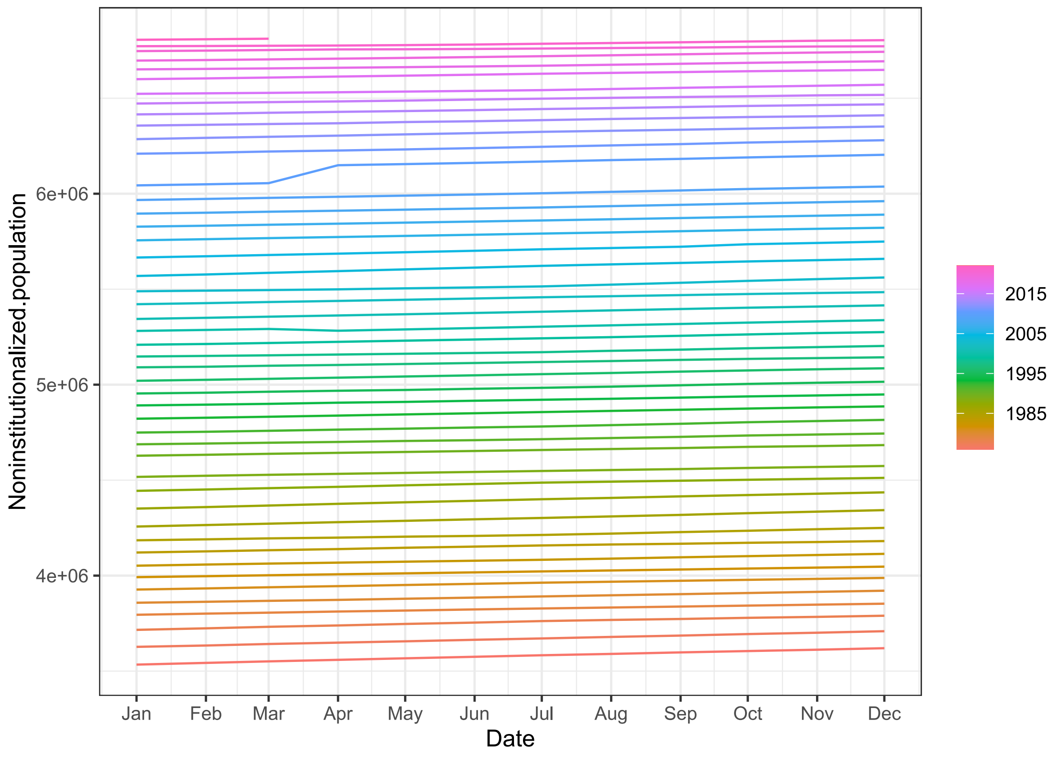

Figure 2.12: Seasonal plot of noninstitutionalized population

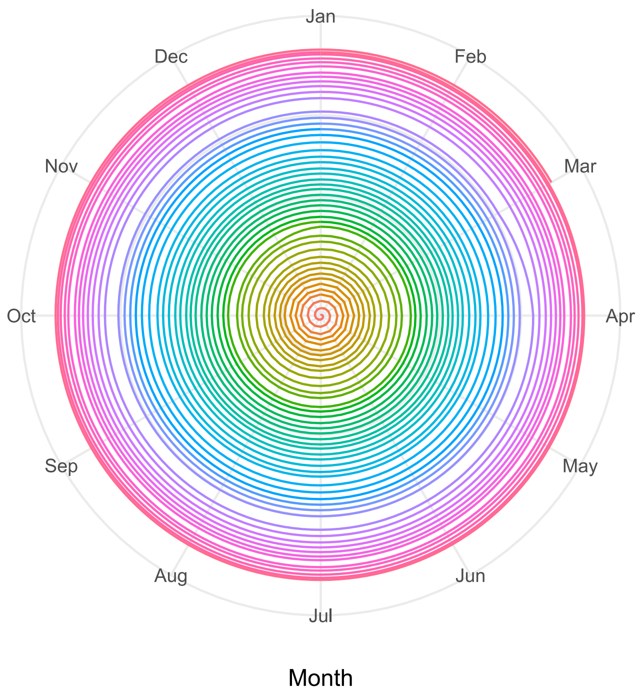

Figure 2.13: Polar seasonal plot of noninstitutionalized population

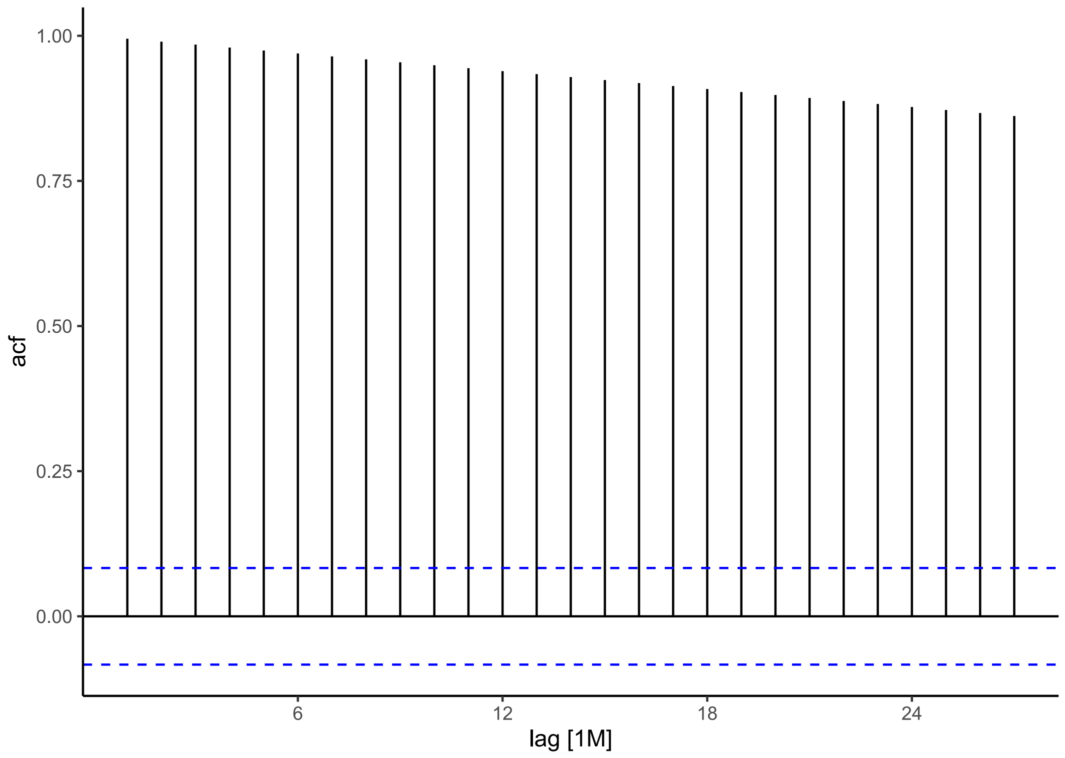

Figure 2.14: ACF plot of noninstitutionalized population

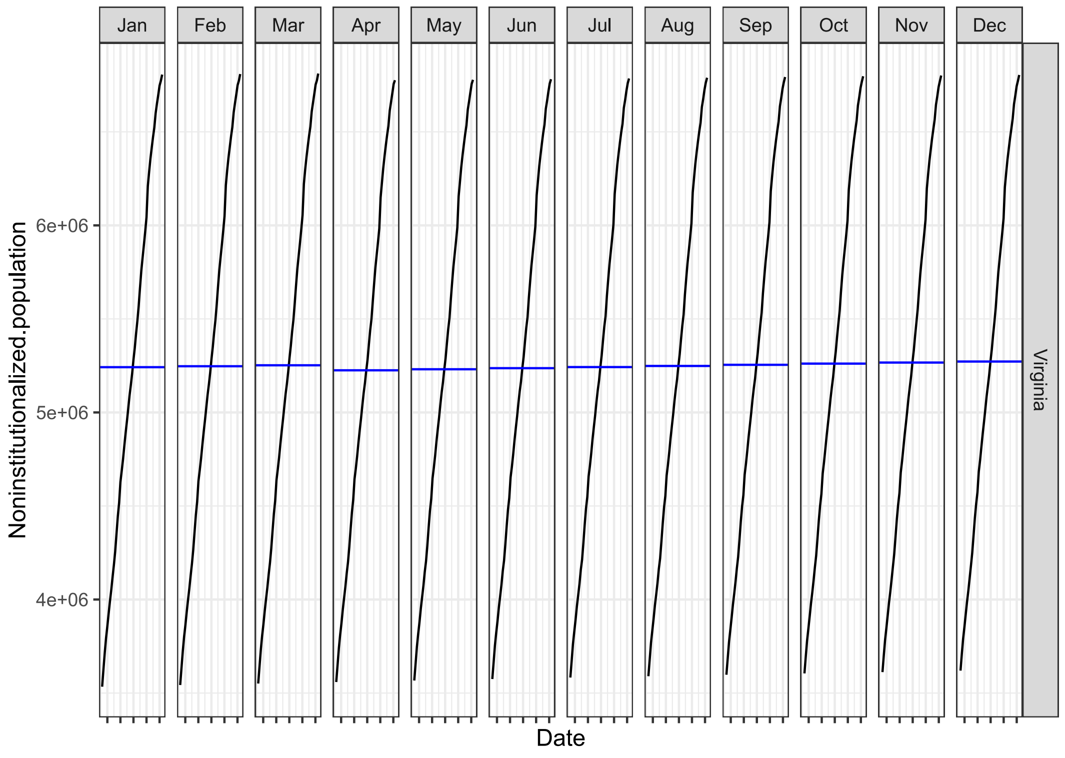

Figure 2.15: Subseries plot of noninstitutionalized population

2.4 Train-Test splitting

As our task is to forecast opioid overdose deaths for the entire year of 2020 (January 2020 through December 2020) and as we have differing availability of each of our various data measures, we choose to perform train-test splitting of our data. Train-test splitting is an important aspect of the modeling workflow to prevent data leakage from the test set (allowing the model to falsely generate predictions having secret information of the true value). We limit our training data to range from January 2000 to December 2019 and we limit our test data to range from January 2020 to December 2020 (for all data sources and variables). Code performing this train-test splitting using base R can be found in Appendix A.4 (p. 41) below.

2.5 Feature engineering

As can be observed from various exploratory data visualizations of our variables, we are dealing with nonstationary data that will present challenges to our modeling approach. We have non-seasonally-adjusted unemployment data and raw (and thus also unadjusted) opioid overdose death count data that both exhibit seasonality and high autocorrelation at most lag values. We describe our approaches of Box-Cox transformation and differencing to induce stationarity in our data to engineer suitable predictors for modeling. Additionally, we utilize the augmented Dickey-Fuller (ADF) test for stationarity (Colonescu, 2018) to aid our feature engineering process towards more stationary variables.

2.5.1 Box-Cox transformation of forecast variable

The Box-Cox transformation is a commonly-used method in time-series analysis to attempt to transform the shape of unwieldy data (Hyndman & Athanasopoulos, 2021d). We use the Box-Cox transform on our forecast variable of opioid overdose deaths to make the later model-fitting process numerically "behave" better. Code implementing the Box-Cox transform on our forecast variable via `fabletools::box_cox()` can be found in Appendix A.5.1 (p. 41) below. We also showcase comparative timeplots of untransformed and Box-Cox-transformed opioid overdose deaths in Figure 2.16 (p. 17) below.

The Box-Cox transform is modified by a parameter λ that we select in a data-driven manner via the automated `feasts::guerrero()` procedure. We once again highlight the importance of our aforementioned train-test split prior to this feature engineering step, as we ensure to fit λ based on our training data alone, as fitting λ on data that includes our test set would inadvertently provide information to our model about the true value. For our data, this ended up as a value of $\lambda = 0.3351251$.

As we are interested in improving the nonstationarity of our forecast variable via our transform, we also perform an ADF test on the untransformed and Box-Cox-transformed opioid overdose deaths variable and present p-values in Table T.1 (p. 16) below.

Variable name	ADF test p-value
Deaths	0.9797752
Box-Cox(Deaths)	0.7635537

Table T.1: ADF test p-values: Untransformed vs Box-Cox-transformed opioid overdose deaths

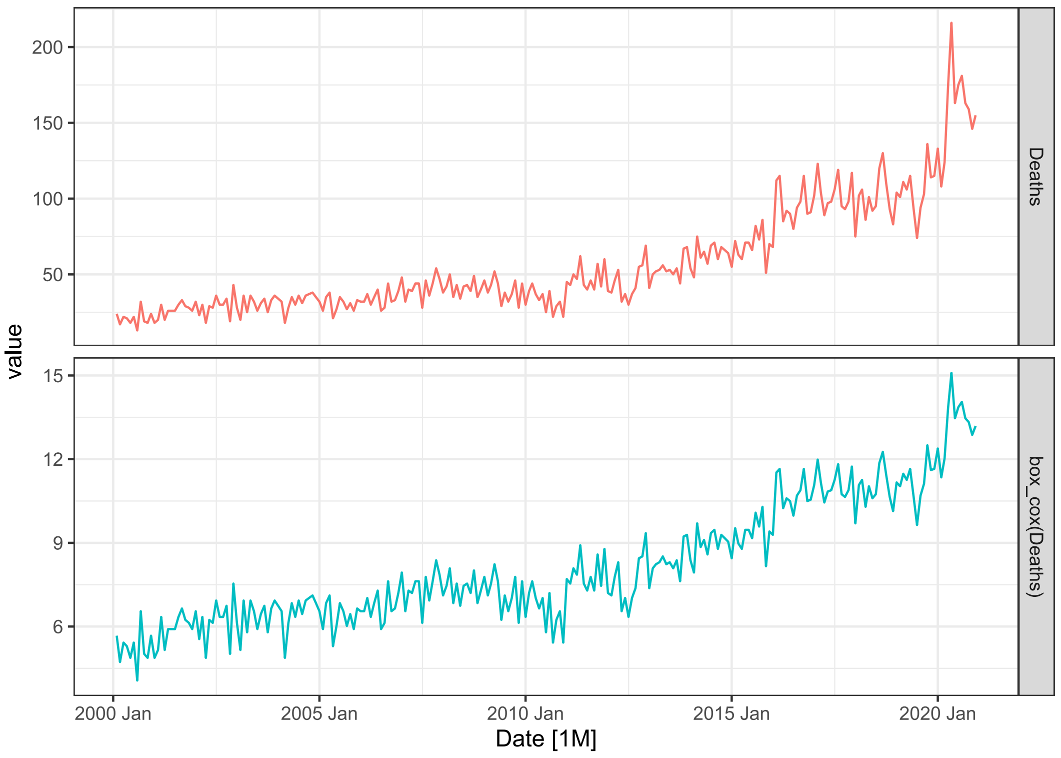

Figure 2.16: Opioid overdose deaths with Box-Cox transformation

Our p-value following transformation is still relatively high, signifying (for the ADF test) a high degree of suspicion of nonstationarity in our variable. However, our ARIMA errors approach described above will allow for differencing and other methods to handle nonstationarity in our forecast variable. As such, we can satisfactorily abandon further modifications of our forecast variable.

2.5.2 Differencing of predictor and covariate variables

As mentioned previously, we opted for non-seasonally-adjusted BLS data (including both our main predictor of unemployed persons and our covariate of noninstitutionalized population). As such, our previous exploratory data visualization for these variables also indicate a nonstationarity. While the nonstationarity in our predictor variable can be handled via the ARIMA specification of our dynamic regression error term, we must perform more rigorous feature engineering to develop stationary (as verified by ADF test p-values) predictors for our model.

We follow general advice (Hyndman & Athanasopoulos, 2021g) in first performing a seasonal difference (period m=12 months) and then performing a standard (period = 1) second-order difference. Code implementing this differencing procedure for both unemployed persons and noninstitutionalized population variables can be found in Appendix A.5.2 (p. 41) below. For the sake of clarity, at this point we transition to presenting the differencing of our unemployed person predictor variable first in completeness, prior to presenting the differencing of our noninstitutionalized population covariate variable afterwards.

We showcase comparative timeplots of undifferenced, seasonally-differenced, and doubly-differenced unemployed persons in Figure 2.17 (p. 18) below. We have also produced comparative ACF plots and partial ACF plots in Figure 2.18 (p. 19) and Figure 2.19 (p. 19) below, respectively.

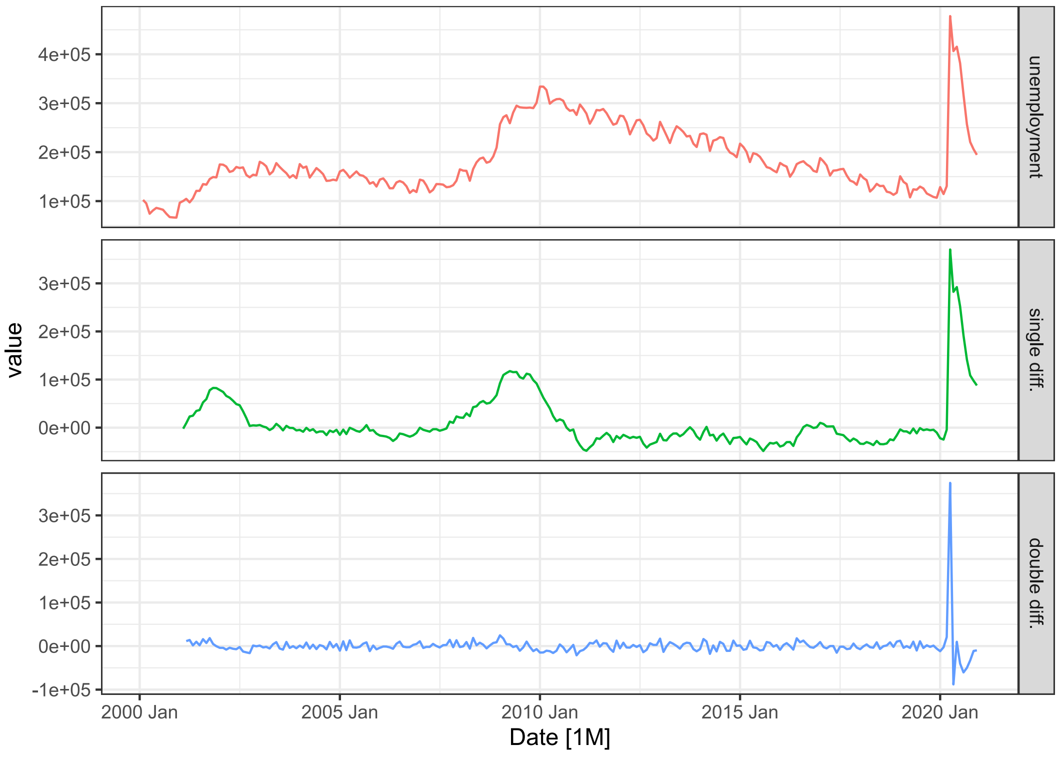

Figure 2.17: Unemployed persons with seasonal and double differencing

As we are interested in the level of stationarity we have induced in our data following our differencing procedure, we perform an ADF test on the un-differenced, seasonally-differenced, and doubly-differenced variable and present p-values in Table T.2 (p. 18) below.

Variable name	ADF test p-value
Unemployed persons	0.3885087
Seasonally-differenced	0.4248142
Doubly-differenced	<0.01

Table T.2: ADF test p-values: Undifferenced vs differenced unemployed persons

We may be surprised to see an increase in p-value following the seasonal differencing, but we do achieve an extremely low p-value following our second difference, allowing us to reject the null hypothesis of the test and drop our suspicion of nonstationarity. This is a satisfactory outcome, and we move to our next variable.

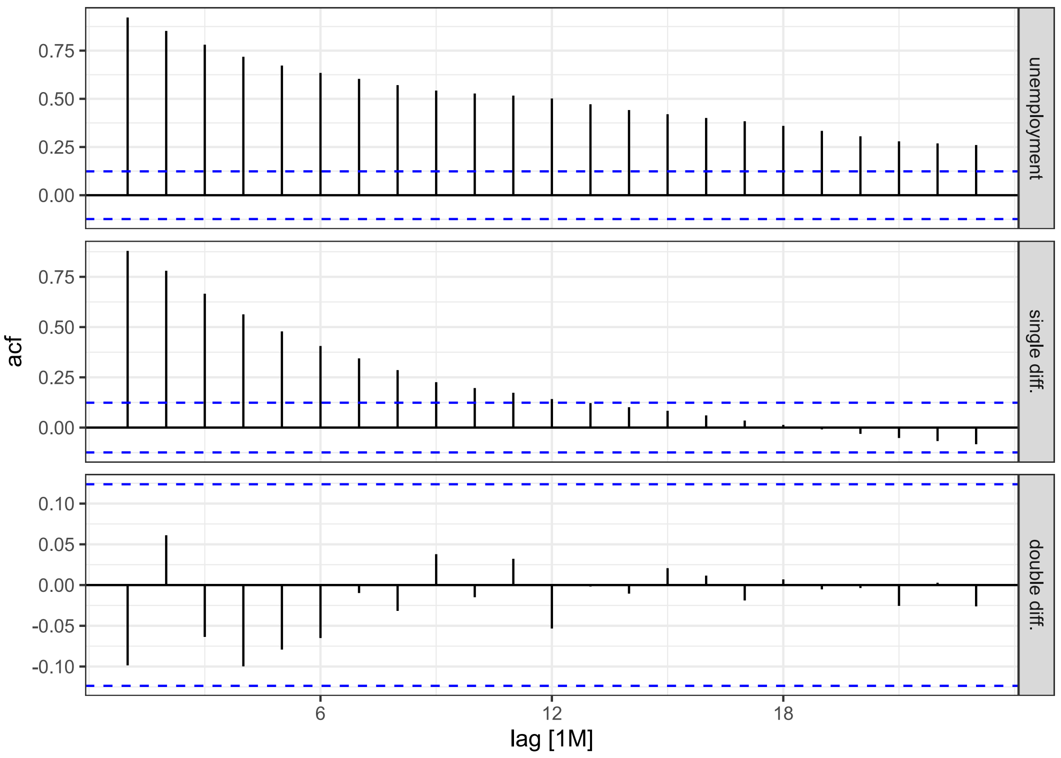

Figure 2.18: ACF plots of unemployed persons with seasonal and double differencing

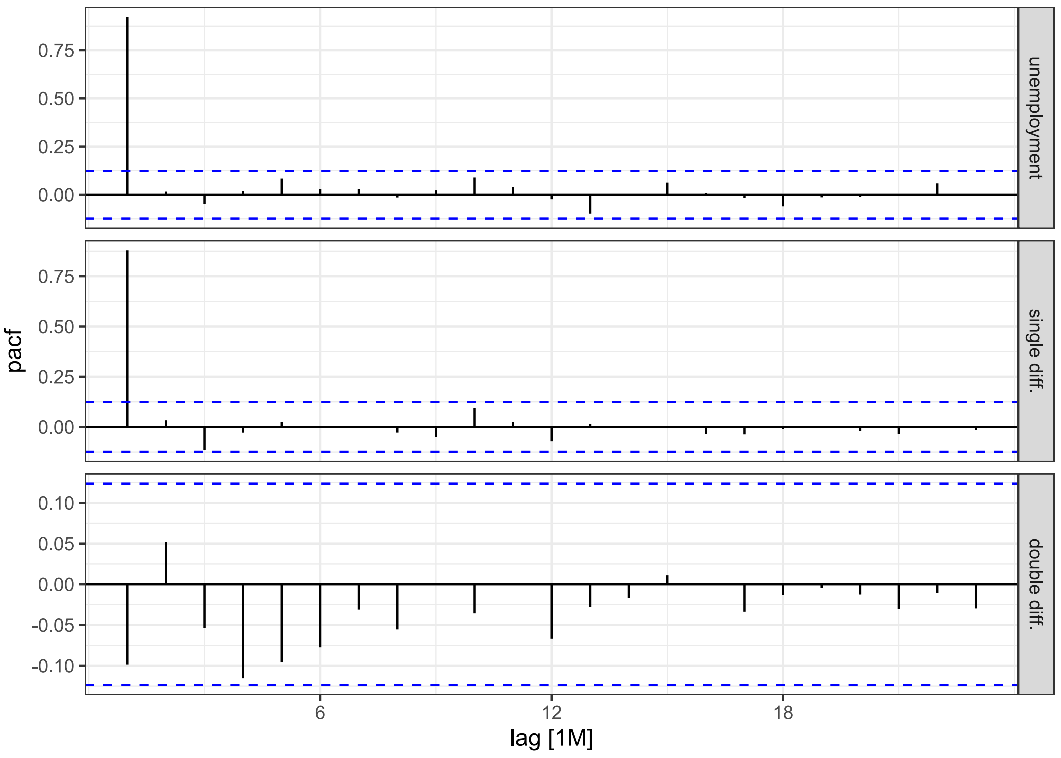

Figure 2.19: pACF plots of unemployed persons with seasonal and double differencing

We showcase comparative timeplots of undifferenced, seasonally-differenced, and doubly-differenced noninstitutionalized population in Figure 2.20 (p. 20) below. We have also produced comparative ACF plots and partial ACF plots in Figure 2.21 (p. 21) and Figure 2.22 (p. 21) below, respectively.

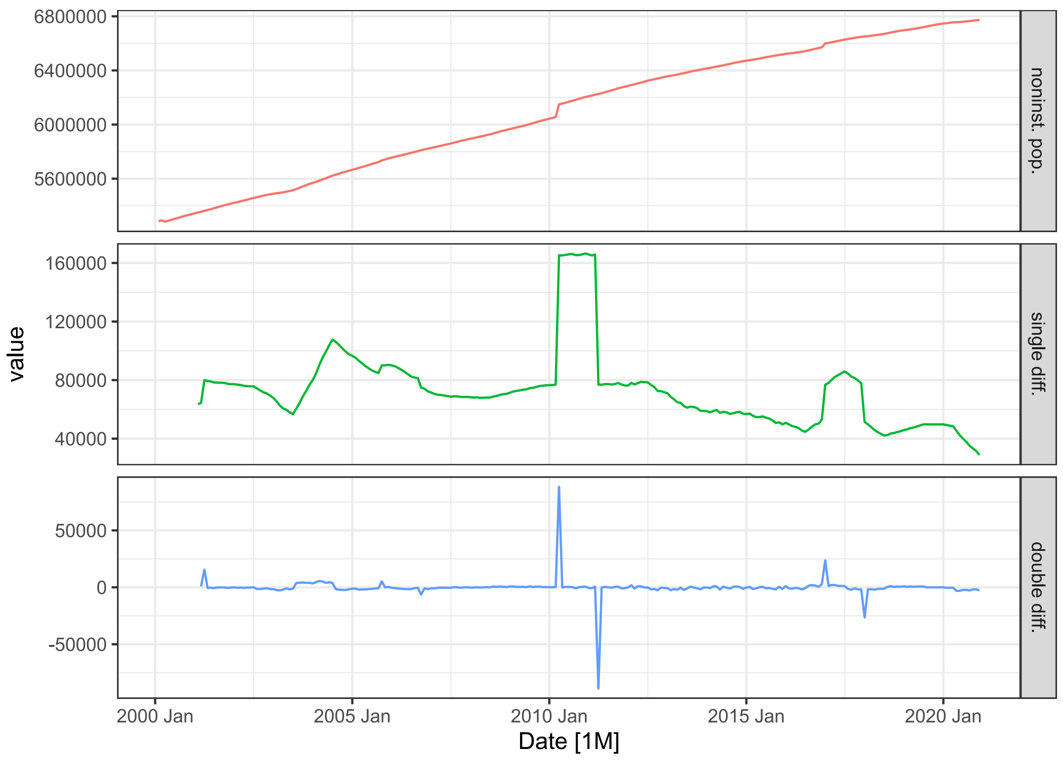

Figure 2.20: Noninstitutionalized population with seasonal and double differencing

As we are interested in the level of stationarity we have induced in our data following our differencing procedure, we perform an ADF test on the un-differenced, seasonally-differenced, and doubly-differenced variable and present p-values in Table T.3 (p. 20) below.

Variable name	ADF test p-value
Noninstitutionalized population	0.99
Seasonally-differenced	0.0486832
Doubly-differenced	<0.01

Table T.3: ADF test p-values: Undifferenced vs differenced noninstitutionalized population

We notice that our p-values decrease with each order of differencing and despite beginning with a quite problematic p-value prior to differencing we obtain a statistically significant p-value following our second differencing allowing us to reject the null hypothesis of the test and drop our suspicion of nonstationarity. We have achieved another satisfactory feature engineering outcome, and now turn our attention to the modeling process.

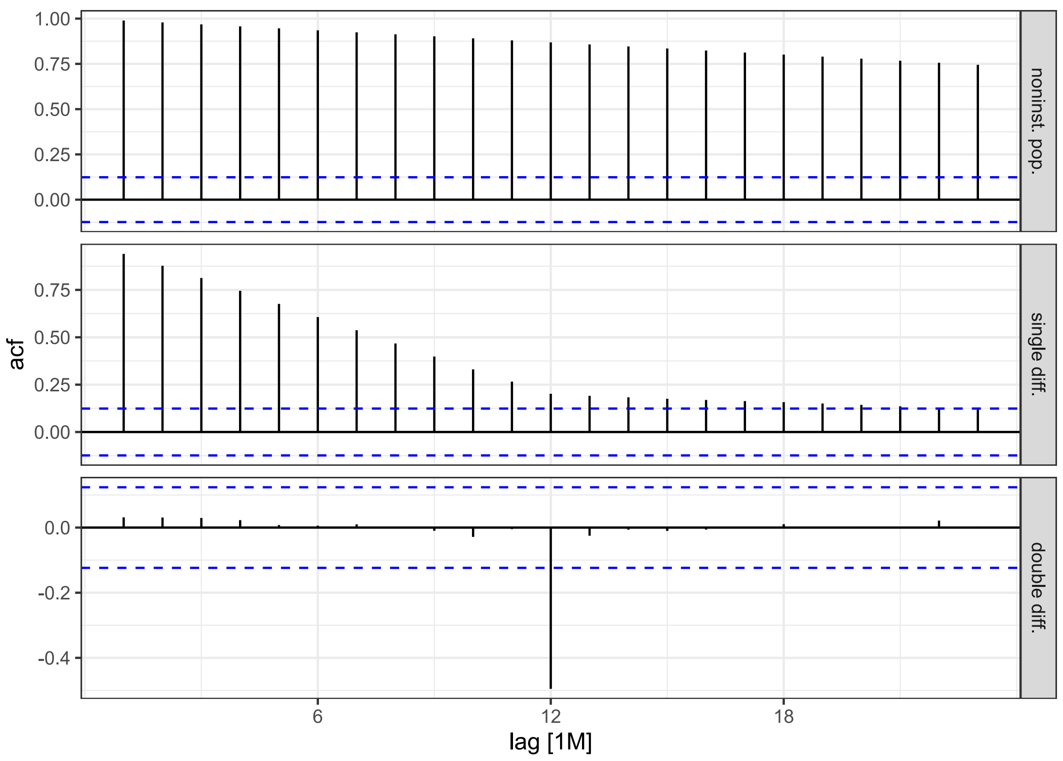

Figure 2.21: ACF plots of noninstitutionalized population with seasonal and double differencing

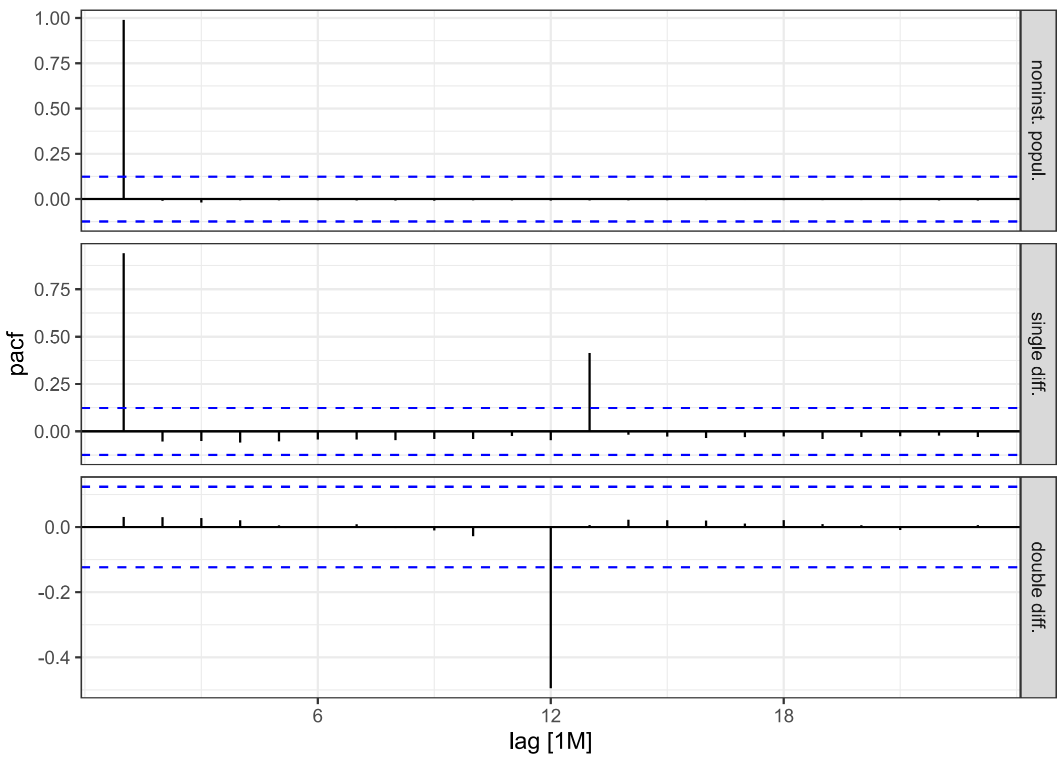

Figure 2.22: pACF plots of noninstitutionalized population with seasonal and double differencing

3 Modeling

As mentioned previously, we are utilizing a dynamic regression approach for our Box-Cox-transformed forecast variable (opioid overdose deaths) with at most 2 predictor variables (unemployed persons and noninstitutionalized population). We plan to use structure our regression model's error term as either both seasonal & nonseasonal ARIMA or nonseasonal ARIMA (with the addition of Fourier terms as predictors).

3.1 Model specification and fitting

We generate a list of candidate models based on combinations of these features (including all versions of engineered predictors: un-differenced, sesaonally-differenced, and doubly-differenced) as well as a couple of reference models (naive and seasonal naive) for later comparison. Code for model specification via `fabletools::model()` are included in Appendix A.6 (p. 41) below.

3.2 Model performance metrics

At this point we describe our model performance metrics we will use to perform selection of our "top five" candidate models below.

3.2.1 In-Sample model-fit: Corrected Akaike information criterion (AICc)

To evaluate candidate model-fit on the training data (from January 2000 to December 2019), we utilize a corrected form of the Akaike information criterion (AICc) to allow for model comparisons with differing numbers of terms (Hyndman & Athanasopoulos, 2021h). Lower values of AICc represent better in-sample model fit.

3.2.2 Out-of-Sample prediction error: Mean absolute error (MAE)

We additionally utilize a measure of mean absolute error (MAE) and a derived mean absolute percentage error (MAPE) measure, to represent prediction error (Hyndman & Athanasopoulos, 2021f). We feel this is a valuable metric to consider as MAE and MAPE are in relation to individual opioid overdose deaths may provide a tangible overview of global prediction error.

4 Evaluation

At this point, we turn our attention to evaluation of our candidate models and their predictions.

4.1 Model selection ("top five")

Following our fitting procedure described above, we sort models based on their in-sample model fit (AICc). Code for this sorting procedure is included in Appendix A.7 (p. 43) below. We then selected the 5 best models in ascending AICc order as a "top five" list of candidate models to push forward to our prediction phase of evaluation. The specific AICc values (in ascending order) are reproduced in Table T.4 (p. 23) below.

Model	AICc
unemp_double_diff	483.3742
noninst_double_diff	484.9667
noninst_single_diff	485.3166
double_diff	485.3980
unemp_double_diff_fourier	486.2854

Table T.4: "Top five" candidate models as selected by AICc (sorted by ascending AICc)

We cannot perform much interpretation from AICc values alone, but we can tell that the models have a clear ordering but with very relatively the same value for each model.

4.2 Prediction performance

We utilize our "top five" models described previously for prediction of our test data (January 2020 to December 2020) via `fabletools::forecast()`. Code for this procedure is included in Appendix A.8 (p. 43) below. We evaluate our model predictions quantitatively by global error and qualitatively based on whether the models were able to predict the spike in opioid overdose deaths in the early months of the pandemic.

4.2.1 Global test error

We evaluate predictions from our models in terms of global test error as MAE and MAPE, and reproduce these values in ascending order in Table T.5 (p. 23) below.

Model	MAE	MAPE
noninst_single_diff	40.50948	23.69582%
noninst_double_diff	41.41842	24.24299%
unemp_double_diff_fourier	41.86609	24.37542%
unemp_double_diff	42.37529	24.66329%
double_diff	42.47680	24.71573%

Table T.5: "Top five" candidate models (sorted by ascending MAE/MAPE)

While we see a clear ordering of MAE/MAPE values, we have relatively similar values (ranging in a little over than 1-[percent]-unit difference in MAPE) so it does not seem that we have much to separate these models from on global test error alone.

4.2.2 Ability to predict pandemic spike

At this point, we will attempt to qualitatively compare "top five" model predictions based on ability to predict the early-2020 spike in opioid overdose deaths. We plan to present our "top five" models in ascending MAE/MAPE order (as presented in Table T.5 above). However, we begin this section instead by examining how well simple models such as the "naive" (*naive*) and "seasonal naive" (*seasonal_naive*) models perform on this task. This will provide us with a valid frame of reference against which to evaluate our work. Without further ado, we present *naive* predictions in Figure 4.1 (p. 24) and Figure 4.2 (p. 24), as well as *seasonal_naive* predictions in Figure 4.3 (p. 25) and Figure 4.4 (p. 25) below.

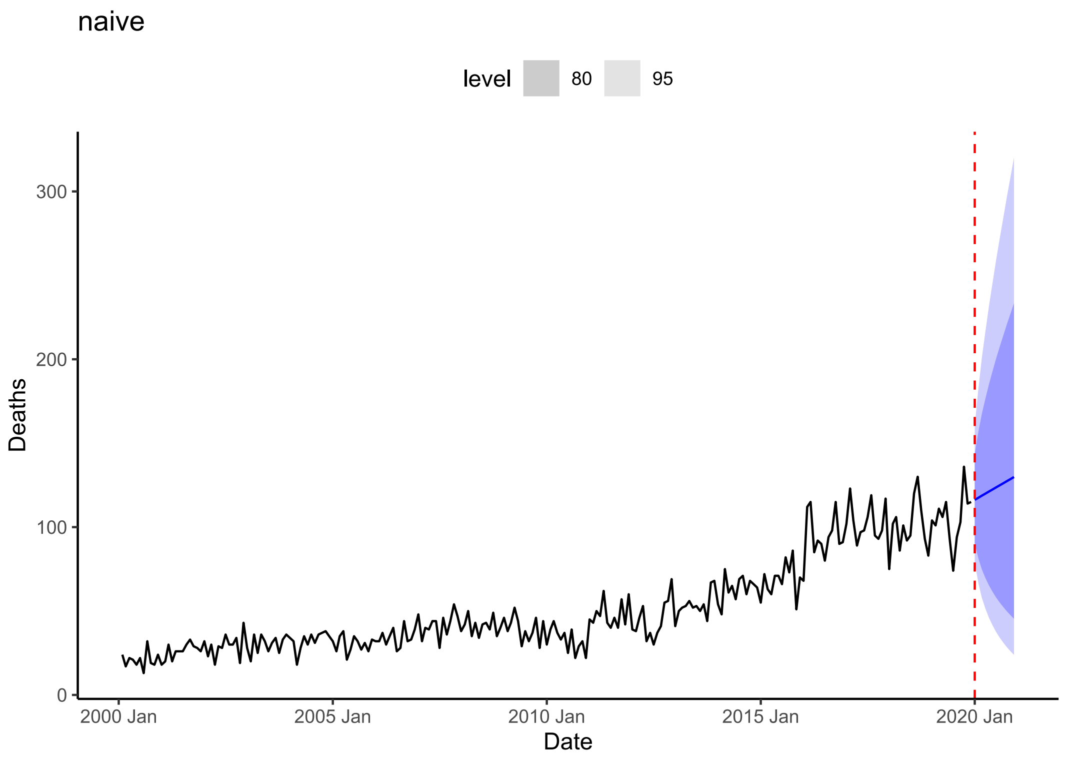

Figure 4.1: Predictions from *naive* alongside training data

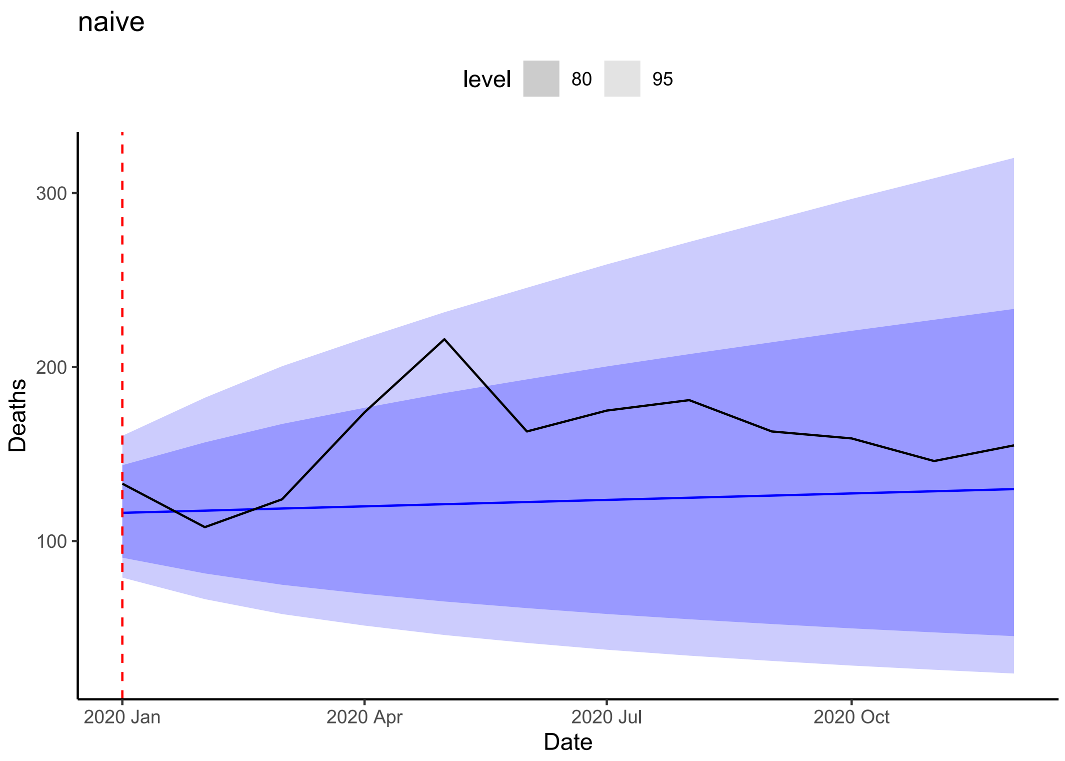

Figure 4.2: Predictions from *naive* alongside test data

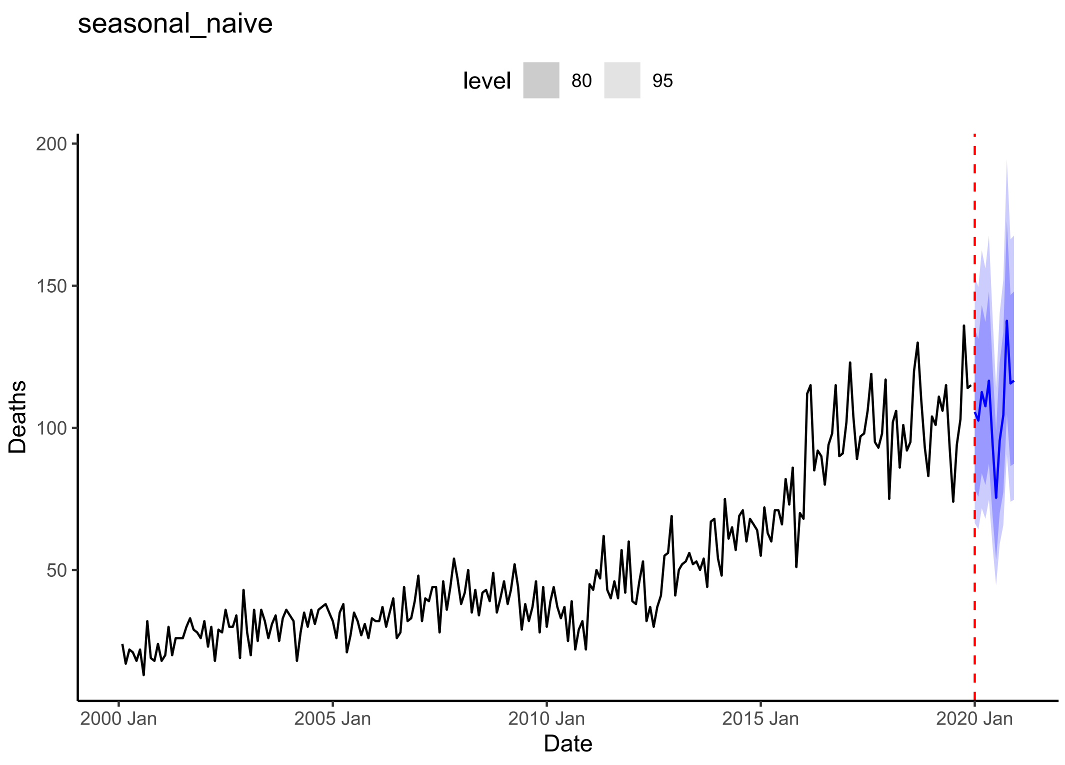

Figure 4.3: Predictions from *seasonal_naive* alongside training data

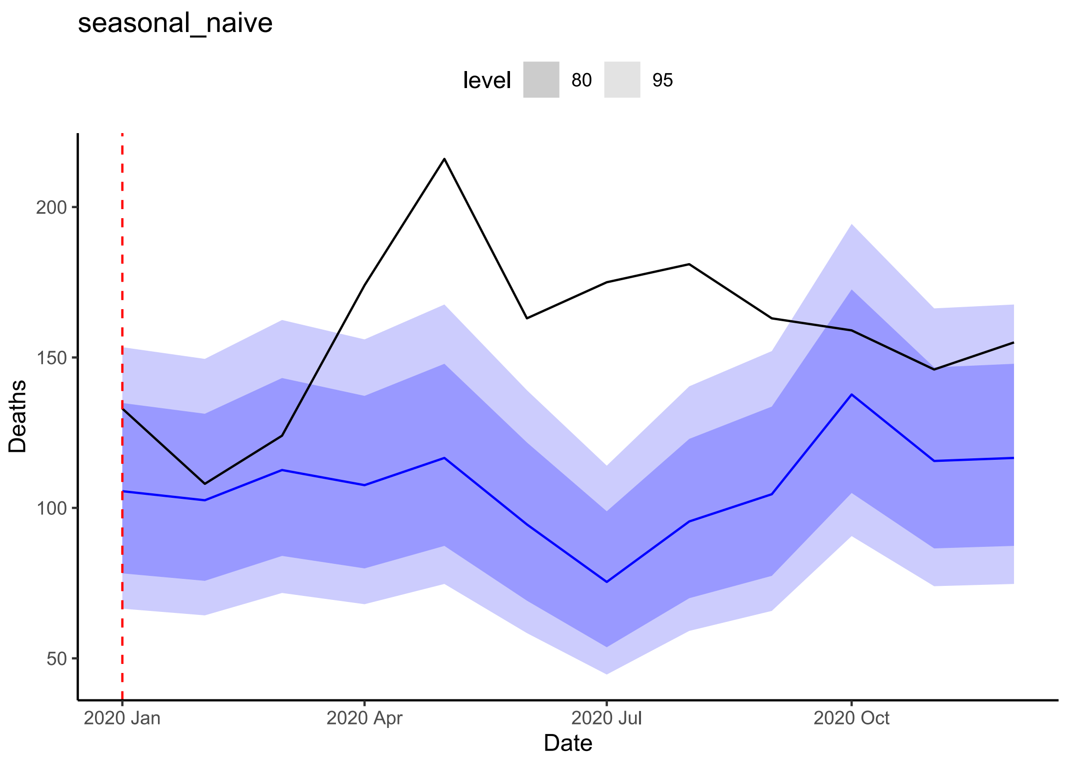

Figure 4.4: Predictions from *seasonal_naive* alongside test data

We may be surprised at how well some aspects of our simple models perform at our task. For example, despite the lack of complexity in *naive* predictions, we notice that the straight line seemed to underestimate but capture the general trend we see in the 2020 test data. Additionally, *seasonal_naive* predictions exhibit a surprisingly smaller prediction interval than the wide prediction intervals exhibited by *naive* predictions. However, when viewing *seasonal_naive* predictions next to test data we notice how improperly this model seems to perform at this task. In fact, *seasonal_naive* predictions are opposite of the test data; we would perhaps assume an approach using this model could be to simply "do the opposite" of its recommendation. While this is an entertaining proposition, it also highlights the issues with this model in performing this task, as well as potentially highlighting the benefit of our solution.

Now that we have a reference point from our simple models, we turn our attention to our "top five" models. We will present our models in ascending MAE/MAPE order, and thus our first 2 models (*noninst_single_diff* and *noninst_double_diff*) exhibit the lowest global prediction error.

We present *noninst_single_diff* predictions in Figure 4.5 (p. 27) and Figure 4.6 (p. 27) below. This model is composed of a single predictor variable: seasonally-differenced noninstitutionalized population, and the error term is structured as an ARIMA model with both nonseasonal and seasonal components.

We present *noninst_double_diff* predictions in Figure 4.7 (p. 28) and Figure 4.8 (p. 28) below. This model is composed of a single predictor variable: doubly-differenced noninstitutionalized population, and the error term is structured as an ARIMA model with both nonseasonal and seasonal components.

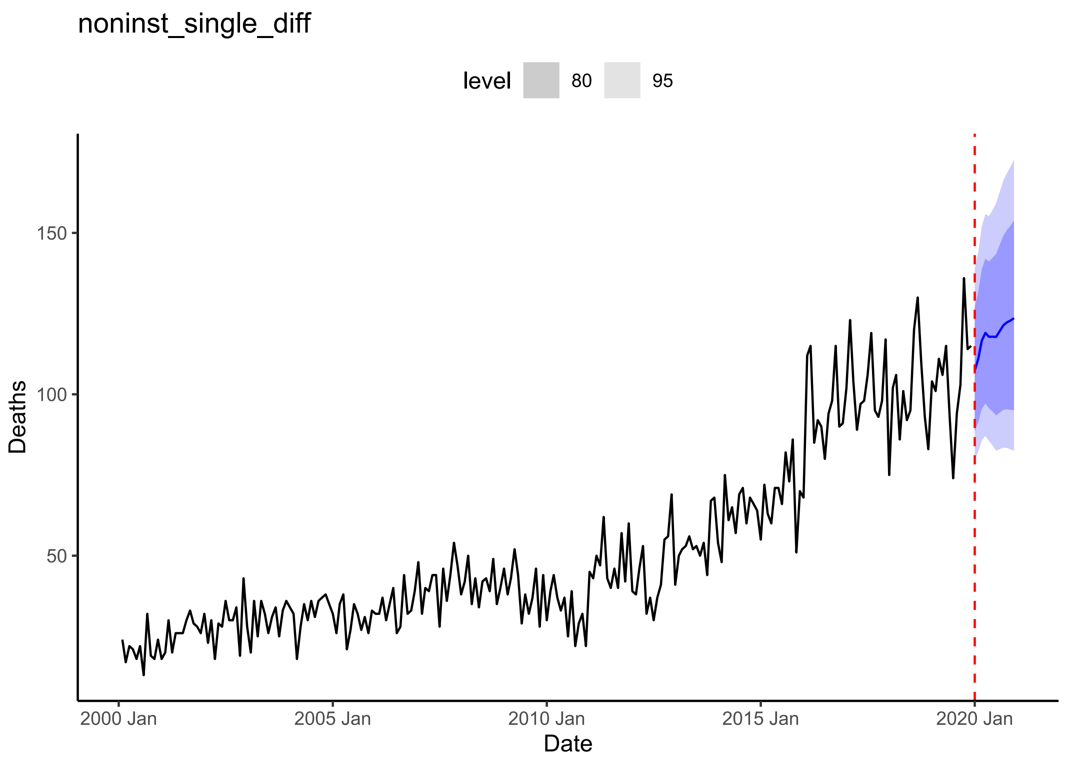

Figure 4.5: Predictions from *noninst_single_diff* alongside training data

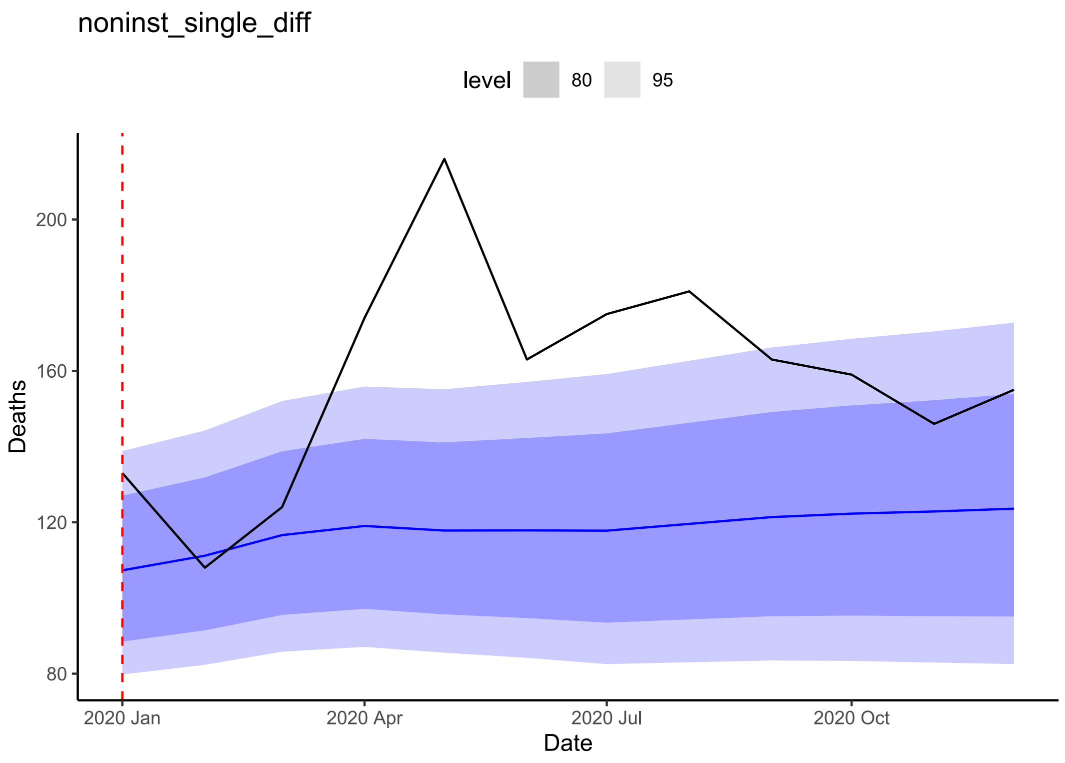

Figure 4.6: Predictions from *noninst_single_diff* alongside test data

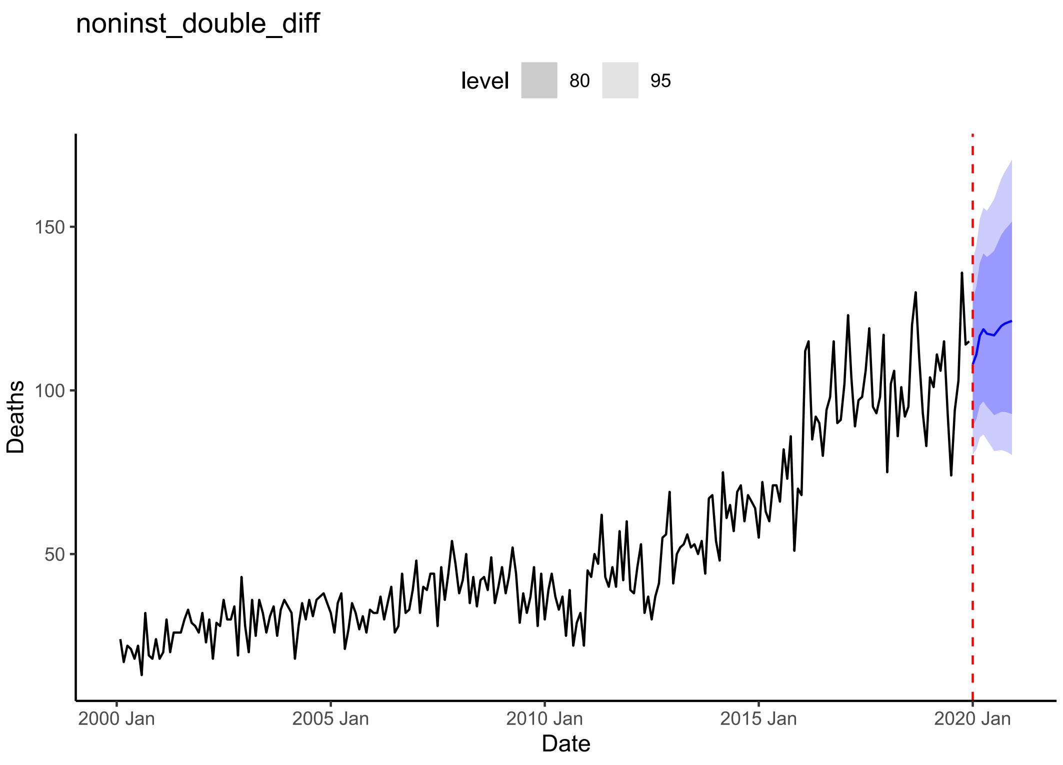

Figure 4.7: Predictions from *noninst_double_diff* alongside training data

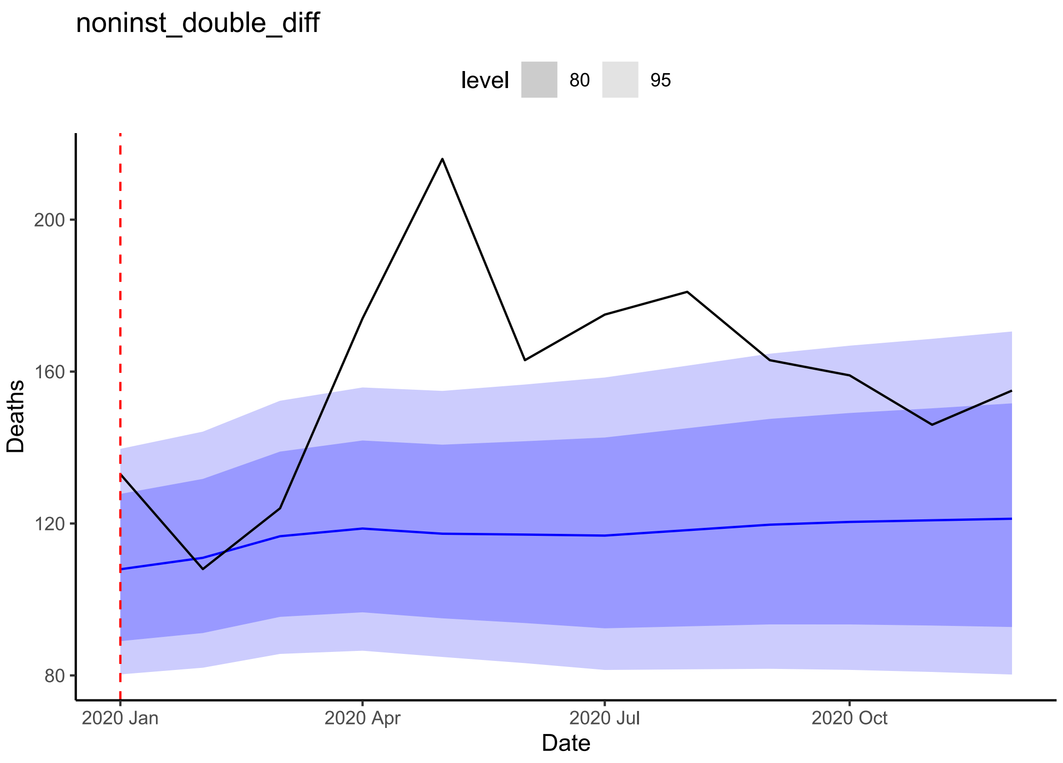

Figure 4.8: Predictions from *noninst_double_diff* alongside test data

Despite the relatively smaller prediction intervals we see of these model predictions, our first 2 models in our "top 5" do not seem to produce appreciably better predictions than *naive*. It is indeed surprising that these models have good MAE/MAPE scores, considering their only predictor is some difference of our initially-intended covariate variable: noninstitutionalized population. As such, it is not as surprising to see their poor performance in this task.

We now turn our attention to our 3 remaining models from our "top five": *unemp_double_diff_fourier*, *unemp_double_diff*, and *double_diff*.

We present *unemp_double_diff_fourier* predictions in Figure 4.9 (p. 30) and Figure 4.10 (p. 30) below. This model is composed of a single predictor variable: doubly-differenced noninstitutionalized population, accompanied by Fourier terms to capture seasonality and the error term is structured as an ARIMA model with both only a nonseasonal component.

We present *unemp_double_diff* predictions in Figure 4.11 (p. 31) and Figure 4.12 (p. 31) below. This model is composed of a single predictor variable: doubly-differenced unemployed persons, and the error term is structured as an ARIMA model with both nonseasonal and seasonal components.

We present *double_diff* predictions in Figure 4.13 (p. 32) and Figure 4.14 (p. 32) below. This model is composed of two predictor variables: doubly-differenced unemployed persons and doubly-differenced noninstitutionalized population, and the error term is structured as an ARIMA model with both nonseasonal and seasonal components.

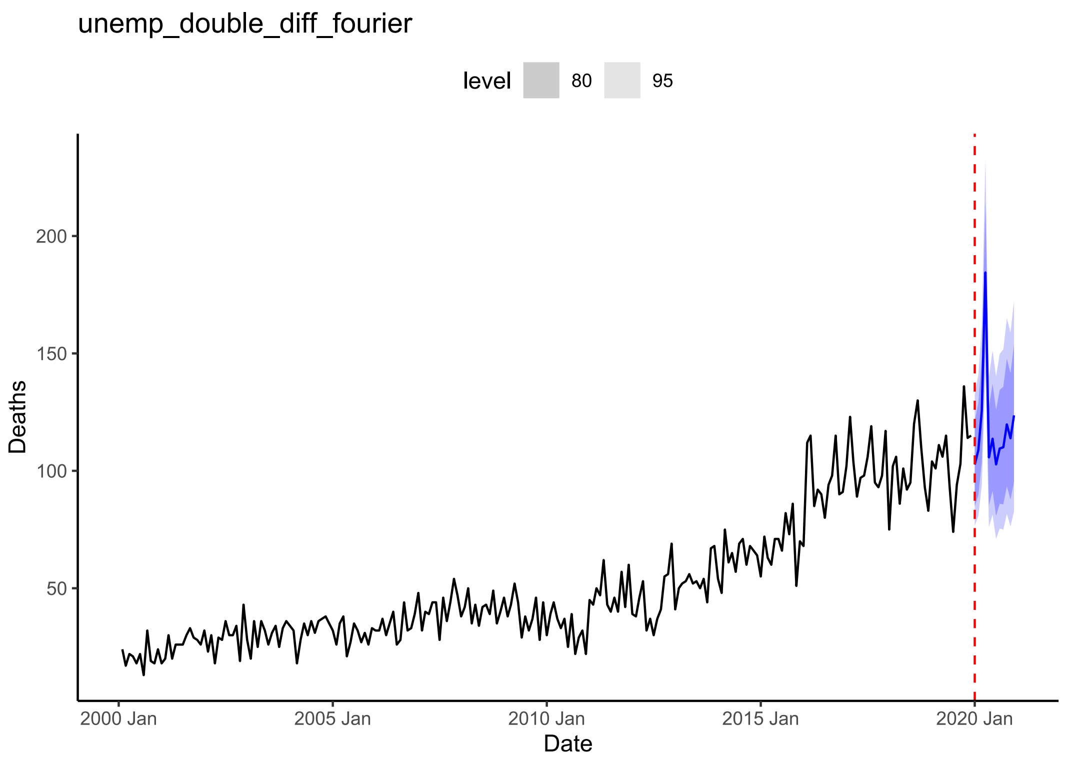

Figure 4.9: Predictions from *unemp_double_diff_fourier* alongside training data

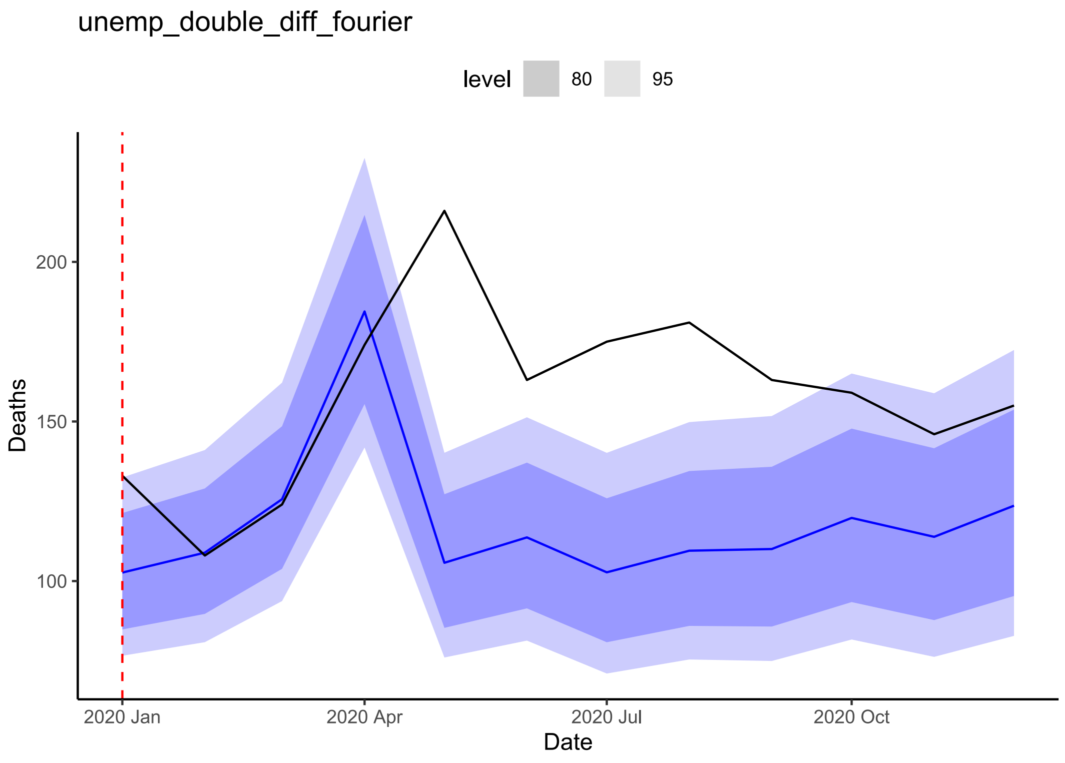

Figure 4.10: Predictions from *unemp_double_diff_fourier* alongside test data

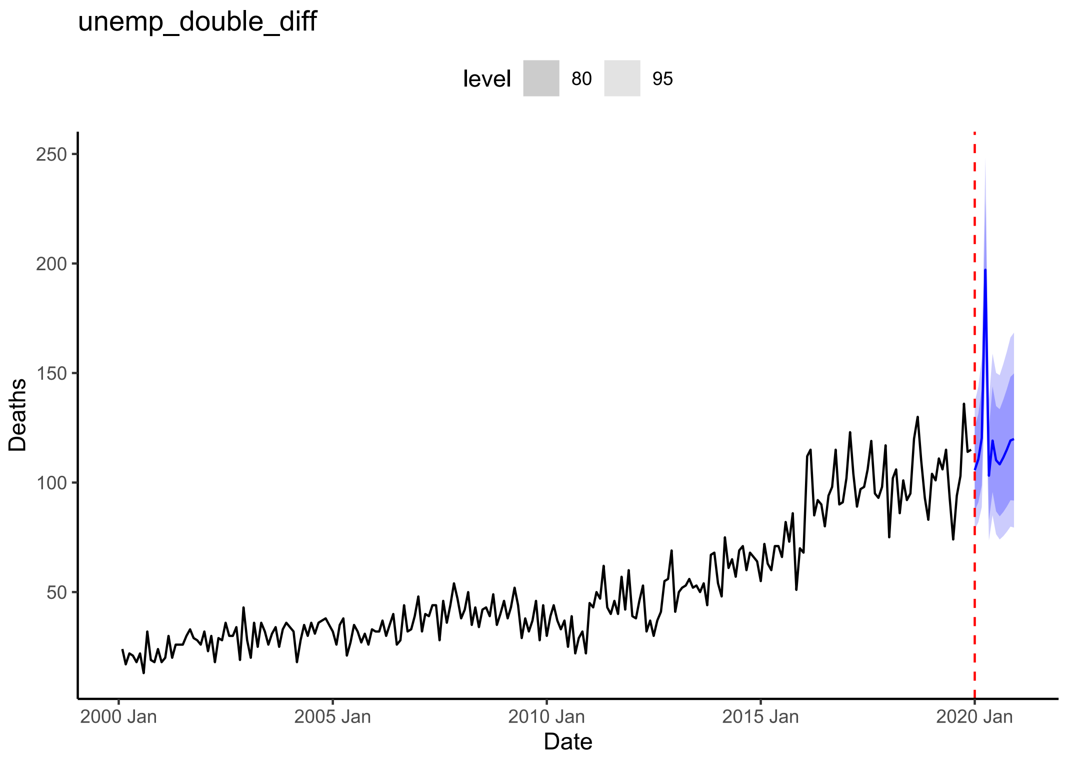

Figure 4.11: Predictions from *unemp_double_diff* alongside training data

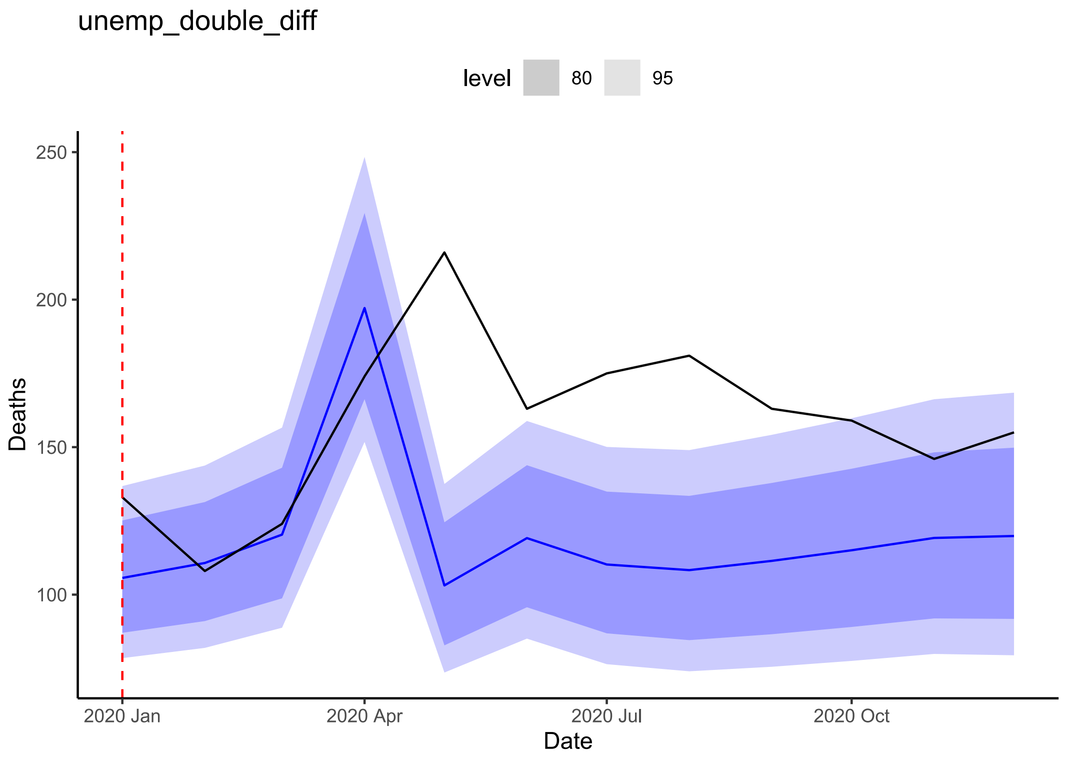

Figure 4.12: Predictions from *unemp_double_diff* alongside test data

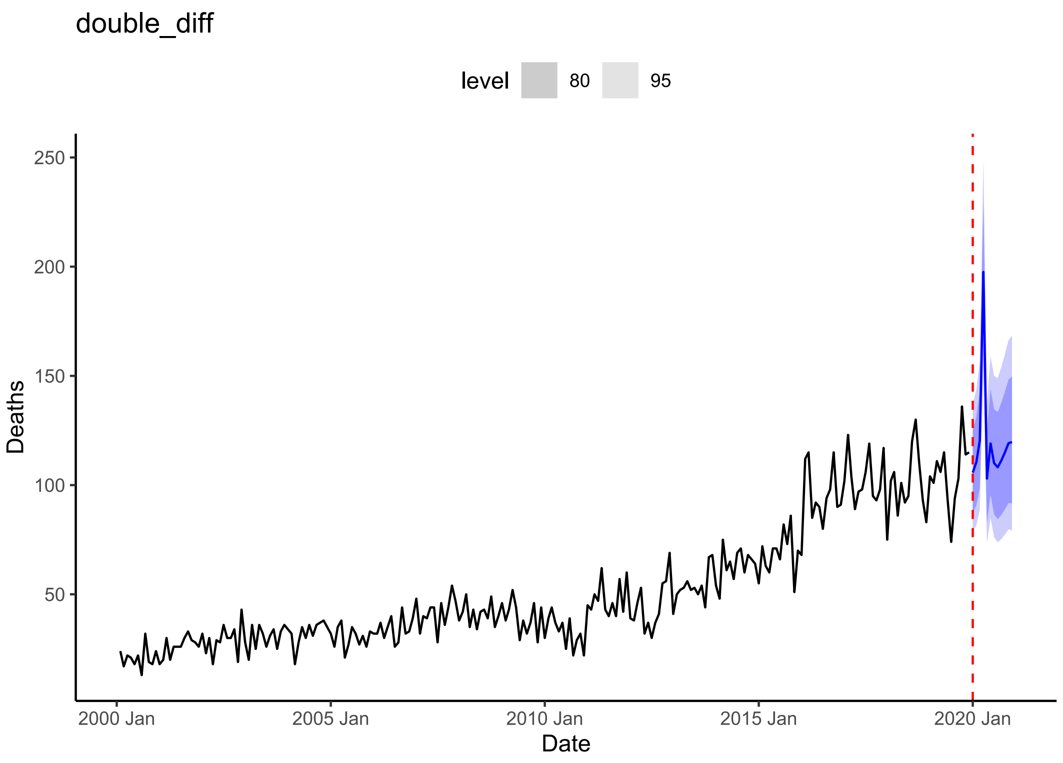

Figure 4.13: Predictions from *double_diff* alongside training data

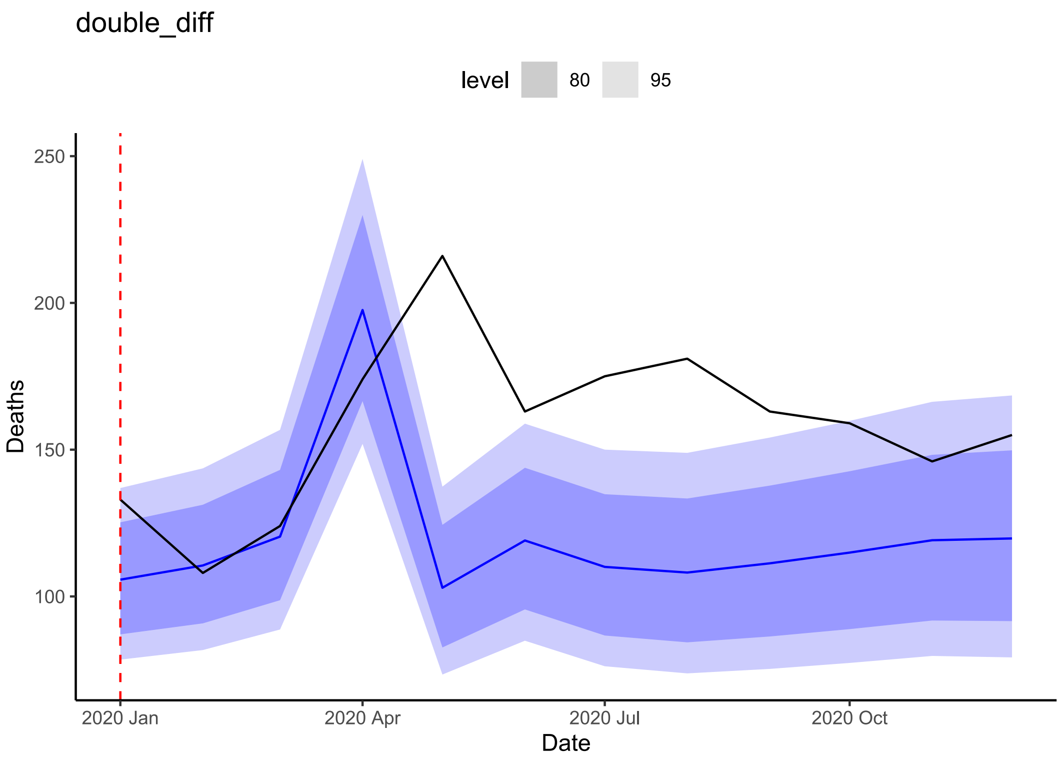

Figure 4.14: Predictions from *double_diff* alongside test data

Qualitatively, performance of these 3 models is exceedingly impressive compared to the others. These "best three" models all successfully predict the spike in opioid overdose deaths in the early months of the pandemic. All predictions look relatively similar, with the slight exception that predictions from *unemp_double_diff_fourier* showcase the "up-and-down" harmonic nature we would expect from our Fourier series predictors.

4.3 Residual diagnostics

Since we have successful performance of our model we have to verify that our model is behaving appropriately from a numerical-statistical standpoint. We now utilize the typical approach of examining residual diagnostic measures for our "best three" models we have identified as successfully predicting the pandemic spike in opioid overdose deaths: *unemp_double_diff_fourier*, *unemp_double_diff*, and *double_diff*. We have produced residual diagnostic plots (timeplots, ACF plots, and histograms) for *unemp_double_diff_fourier* in Figure 4.15 (p. 34), *unemp_double_diff* in Figure 4.16 (p. 34), and *double_diff* in Figure 4.17 (p. 35) below.

We perform the commonly-accepted Ljung-Box test of model residuals to ensure our "best three" models are properly-behaved (Hyndman & Athanasopoulos, 2021e); we present present p-values in Table T.6 (p. 33) below.

Model name	Ljung-Box test p-value
unemp_double_diff_fourier	0.7683319
unemp_double_diff	0.7438403
double_diff	0.7485437

Table T.6: Ljung-Box test p-values: "Best three" prediction models

These p-values are all indicative of model residuals not being statistically-significantly-different from white noise. As such, we can be comfortable with the predictions and numerical behavior of our models.

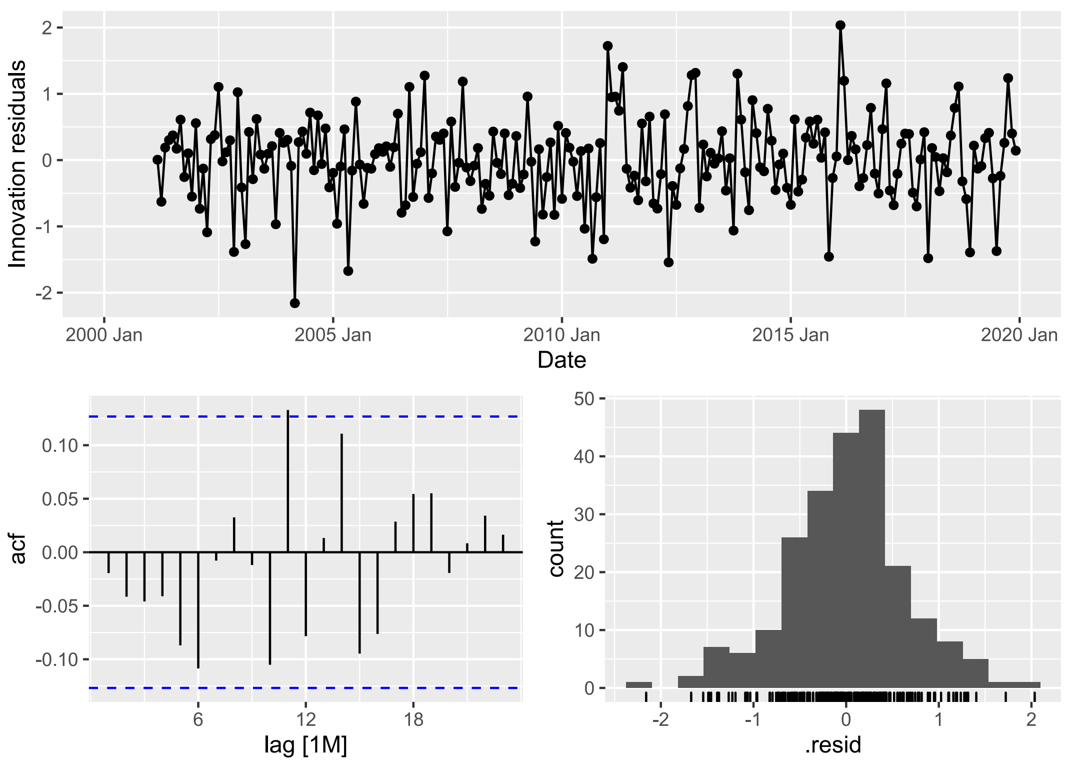

Figure 4.15: Timeplot, ACF plot, and histogram of *unemp_double_diff_fourier* residuals

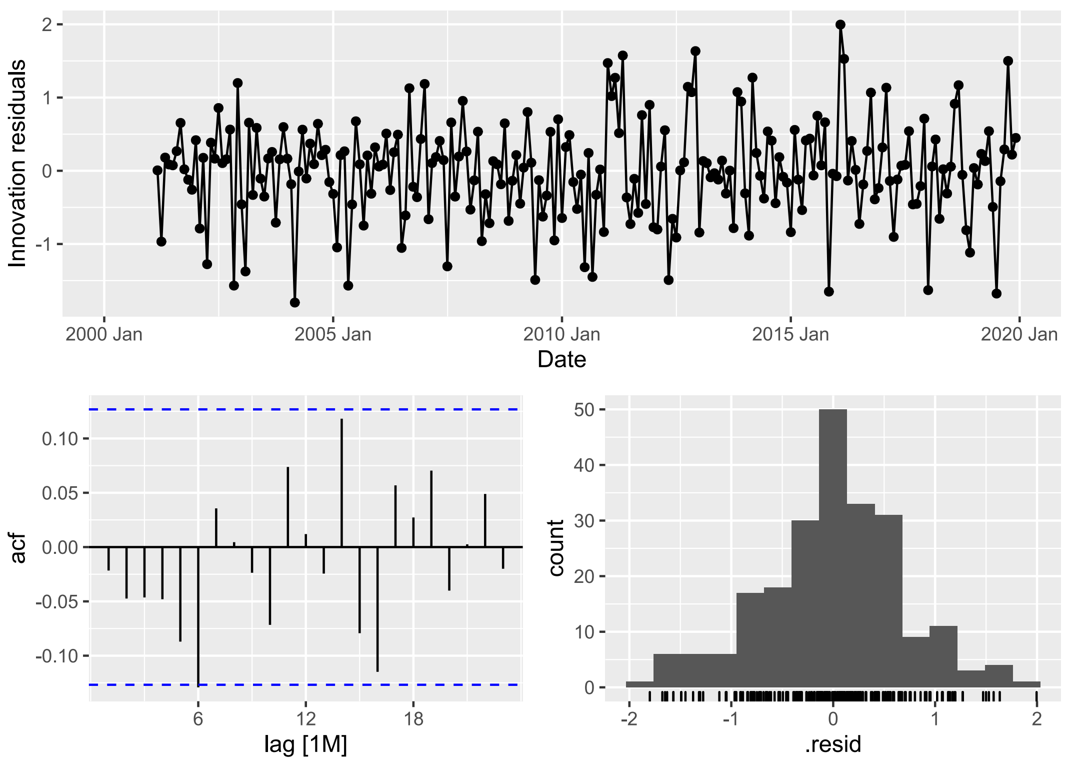

Figure 4.16: Timeplot, ACF plot, and histogram of *unemp_double_diff* residuals

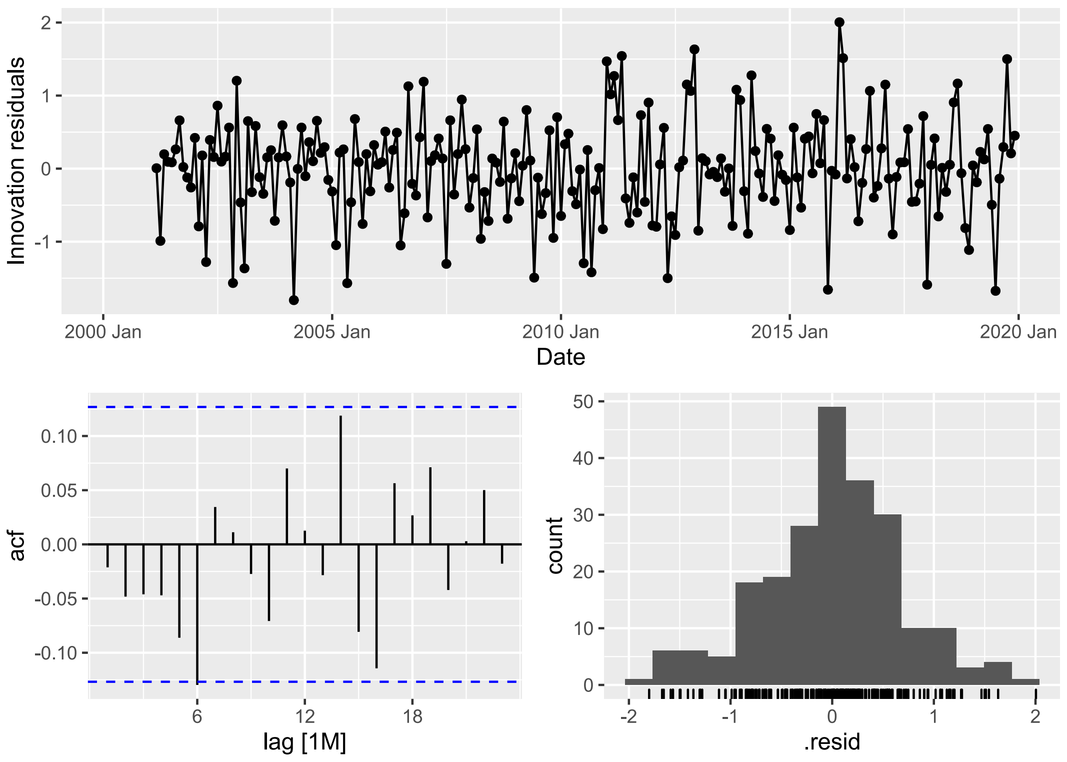

Figure 4.17: Timeplot, ACF plot, and histogram of *double_diff* residuals

5 Findings

5.1 Successful prediction of pandemic spike

At the conclusion of our procedure, we were able to successfully predict the early-pandemic spike in opioid overdose deaths in 3 out of 5 of our selected models. Although we followed generally-accepted procedures for model selection via AICc and prediction error, e.g., MAE/MAPE, (Hyndman and Athanasopoulos, 2021f; Hyndman and Athanasopoulos, 2021h), these procedures did not necessarily produce models that achieved our goal (as evidenced by 2 out of 5 of our models being unsuccessful at this task). We suspect that our combination of AICc and MAE/MAPE criteria for model selection may not be ideal for our niche goal, despite its establishment as standard practice in model selection.

5.2 Engineered predictors induced successful models

Our initial exploratory analyses suggested we utilize methods to handle nonstationarity such as Box-Cox transformation and second-order differencing. Upon performing these transformations, we noticed that 5 out of 5 of our selected models included predictors produced by these transformations. Our specially-engineered predictors proved useful as they perhaps handled stationarity better than the automatically-selected ARIMA formulations which could have attempted to find the appropriate orders of differencing for our predictors. We believe spending time on engineering predictors based on their nonstationarity and other characteristics will lead to better candidate models than throwing this task to the automated model selection algorithm on un-engineered predictors.

5.3 Harmonic predictors did model seasonality effectively

Fourier series terms (i.e., harmonic predictors) found their way into 1 out of 5 of our selected models. The Fourier terms demonstrated their efficacy in handling various types of seasonality. We assume in more complex time series tasks (e.g., with finer-grain time-series data) with more sophisticated patterns of seasonality that Fourier terms will outperform our ARIMA errors approach for seasonality. However, for our purposes, Fourier terms and ARIMA seasonality performed relatively similar to one another (despite previously-noted visual differences in model predictions).

6 Conclusions and future directions

We successfully forecasted the spike in opioid overdose deaths at the start of the pandemic (early 2020) in 3 out of 5 of our "top five" dynamic [harmonic] regression models. Our feature-engineered predictors were present in 5 out of our "top five" models. Even 1 of our "top five" models included Fourier series seasonality terms (i.e., dynamic harmonic regression).

Given the relative success of our approach, we would like to compare our work with well-regarded models in the time-series prediction space, such as Facebook's prophet model (Hyndman & Athanasopoulos, 2021b) or an artificial neural network (Hyndman & Athanasopoulos, 2021c). As these well-regarded models are known to perform well on these types of tasks, we would like to modify the challenge to implement cross-validation on a year-by-year basis. An ideal trial of this type of model would be to generate predictions a year out at a time and perhaps identify best models from performance on that specific task.

7 Acknowledgments

This work is supported in part by the NSF grant DGE: 1922598 *(NRT-HDR: Transdisciplinary Graduate Training Program in Data-Driven Adaptive Systems of Brain-Body Interactions).* Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

8 Bibliography

Altekruse, S. F., Cosgrove, C. M., Altekruse, W. C., Jenkins, R. A., & Blanco, C. (2020). Socioeconomic risk factors for fatal opioid overdoses in the United States: Findings from the Mortality Disparities in American Communities Study (MDAC). *PLOS ONE*, *15*(1), 1–16. https://doi.org/10.1371/journal.pone.0227966

Aram, J., Johnson, N. J., Lee, M. L. T., & Slopen, N. (2020). Drug overdose mortality is associated with employment status and occupation in the National Longitudinal Mortality Study. *American Journal of Drug and Alcohol Abuse*, *46*(6), 769–776. https://doi.org/10.1080/00952990.2020.1820018

Brown, E., & Wehby, G. L. (2019). Economic Conditions and Drug and Opioid Overdose Deaths. *Medical Care Research and Review*, *76*(4), 462–477. https://doi.org/10.1177/1077558717722592

Centers for Disease Control and Prevention. (n.d.). CDC WONDER – Wide-ranging Online Data for Epidemiologic Research. Retrieved May 14, 2022, from https://wonder.cdc.gov/

Colonescu, C. (2018). *Using R for principles of Econometrics.*

Haley, D. F., & Saitz, R. (2020). The Opioid Epidemic During the COVID-19 Pandemic. *JAMA*, *324*(16), 1615–1617. https://doi.org/10.1001/JAMA.2020.18543

Heyman, G. M., McVicar, N., & Brownell, H. (2019). Evidence that social-economic factors play an important role in drug overdose deaths. *International Journal of Drug Policy*, *74*, 274–284. https://doi.org/10.1016/j.drugpo.2019.07.026

Hyndman, R. J., & Athanasopoulos, G. (2021a). 10.5 Dynamic harmonic regression — Forecasting: Principles and Practice (3rd ed). Retrieved May 14, 2022, from https://otexts.com/fpp3/dhr.html

Hyndman, R. J., & Athanasopoulos, G. (2021b). 12.2 Prophet model — Forecasting: Principles and Practice (3rd ed). Retrieved May 15, 2022, from https://otexts.com/fpp3/prophet.html

Hyndman, R. J., & Athanasopoulos, G. (2021c). 12.4 Neural network models — Forecasting: Principles and Practice (3rd ed). Retrieved May 15, 2022, from https://otexts.com/fpp3/nnetar.html

Hyndman, R. J., & Athanasopoulos, G. (2021d). 3.1 Transformations and adjustments — Forecasting: Principles and Practice (3rd ed). Retrieved May 14, 2022, from https://otexts.com/fpp3/transformations.html#mathematical-transformations

Hyndman, R. J., & Athanasopoulos, G. (2021e). 5.4 Residual diagnostics — Forecasting: Principles and Practice (3rd ed). Retrieved May 14, 2022, from https://otexts.com/fpp3/diagnostics.html#portmanteau-tests-for-autocorrelation

Hyndman, R. J., & Athanasopoulos, G. (2021f). 5.8 Evaluating point forecast accuracy — Forecasting: Principles and Practice (3rd ed). Retrieved May 14, 2022, from https://otexts.com/fpp3/accuracy.html#percentage-errors

Hyndman, R. J., & Athanasopoulos, G. (2021g). 9.1 Stationarity and differencing — Forecasting: Principles and Practice (3rd ed). Retrieved May 14, 2022, from https://otexts.com/fpp3/stationarity.html#unit-root-tests

Hyndman, R. J., & Athanasopoulos, G. (2021h). 9.6 Estimation and order selection — Forecasting: Principles and Practice (3rd ed). Retrieved May 14, 2022, from https://otexts.com/fpp3/arima-estimation.html#information-criteria

Hyndman, R. J., & Athanasopoulos, G. (2021i). Chapter 10 Dynamic regression models — Forecasting: Principles and Practice (3rd ed). Retrieved May 14, 2022, from https://otexts.com/fpp3/dynamic.html

Kaiser Family Foundation. (n.d.). Opioid Overdose Death Rates and All Drug Overdose Death Rates per 100,000 Population (Age-Adjusted) — KFF. Retrieved May 11, 2022, from https://www.kff.org/other/state-indicator/opioid-overdose-death-rates/?currentTimeframe=0&sortModel=%7B%22colId%22:%22Location%22,%22sort%22:%22asc%22%7D

Mensah, E. A. M., Rahmathullah, M. J., Kumar, P., Sadeghian, R., & Aram, S. (2021). A Proactive Approach to Combating the Opioid Crisis Using Machine Learning Techniques. In H. R. Arabnia, L. Deligiannidis, H. Shouno, F. G. Tinetti, & Q.-N. Tran (Eds.), *Advances in computer vision and computational biology* (pp. 385–398). Springer International Publishing.

R Core Team. (2022, March 10). *R: A language and environment for statistical computing* (comp. software; Version 4.1.3). Vienna, Austria. https://www.R-project.org/

RStudio Team. (2022, March 17). *Rstudio: Integrated development environment for r* (Version 2022.02.1+461). Boston, MA. http://www.rstudio.com/

U.S. BUREAU OF LABOR STATISTICS. (n.d.). State Employment and Unemployment (Monthly). Retrieved May 14, 2022, from https://www.bls.gov/web/laus.supp.toc.htm

U.S. Department of Health and Human Services. (2021). What is the U.S. Opioid Epidemic? Retrieved May 14, 2022, from https://www.hhs.gov/opioids/about-the-epidemic/index.html

Yang, T.-C., Kim, S., & Matthews, S. A. (2021). Unemployment and Opioid-Related Mortality Rates in U.S. Counties: Investigating Social Capital and Social Isolation-Smoking Pathways. *Social Problems*, 1–21. https://doi.org/10.1093/socpro/spab053

A Appendix of relevant code

This section serves to provide references to code used in this project. Earlier text may reference this section to retain focus on the outcomes of the work rather than on the code. All code was built for the R language for statistical computing version 4.1.3 (R Core Team, 2022) and executed in the RStudio integrated development environment version 2022.02.1+461 (RStudio Team, 2022).

A.1 Package imports

```r
library(tidyverse) # data manipulation
library(fpp3) # FPP3 book package (time series analysis)
library(forecast) # time series forecasting
library(gridExtra) # side-by-side plots
```

A.2 Reading-in data

A.2.1 Reading-in CDC WONDER data via readr::read_tsv()

```r
# specify relevant paths
data_path = 'your_data_path_here'
opioid_file_path = paste0(data_path,
                          'state-opioid-overdose-deaths_DMV_1999-2020.tsv')

# reading-in data
raw_opioid_df <- read_tsv(opioid_file_path) # CDC WONDER data
```

A.2.2 Reading-in BLS data via base::readLines()

```r
# specify relevant paths
data_path = 'your_data_path_here'
unadjusted_unemployment_file_path = paste0(data_path,
                                           'state_unemployment_unadjusted.txt')

# read-in text file via readLines method and create data frame on-the-fly
unemployment_lines <- readLines(unadjusted_unemployment_file_path)

# define empty df to add-to later
unemployment_df <- data.frame(
  'State'=character(0),
  'Date'=character(0),
  'Noninstitutionalized population'=numeric(0),
  'Unemployed'=numeric(0)
)

# process lines from file and append to df
for (num_line in seq(length(unemployment_lines))){
  if (num_line %% 58 == 17) { # trigger at each new month of data
    # first get month and year for data point
    line_date = unemployment_lines[num_line] %>% trimws()

    # next get data for each state
    DC_line = unemployment_lines[num_line + 12] # DC data
    DC_noninst = strsplit(DC_line, "\\s{2,}")[[1]][2]
    DC_unemplo = strsplit(DC_line, "\\s{2,}")[[1]][7]
    DC_df_row = c('District of Columbia', line_date, DC_noninst, DC_unemplo)

    MD_line = unemployment_lines[num_line + 24] # Maryland data
    MD_noninst = strsplit(MD_line, "\\s{2,}")[[1]][2]
    MD_unemplo = strsplit(MD_line, "\\s{2,}")[[1]][7]
    MD_df_row = c('Maryland', line_date, MD_noninst, MD_unemplo)

    VA_line = unemployment_lines[num_line + 51] # Virginia data
    VA_noninst = strsplit(VA_line, "\\s{2,}")[[1]][2]
    VA_unemplo = strsplit(VA_line, "\\s{2,}")[[1]][7]
    VA_df_row = c('Virginia', line_date, VA_noninst, VA_unemplo)

    # append data to data frame
    unemployment_df[nrow(unemployment_df)+1,] <- as.vector(DC_df_row)
    unemployment_df[nrow(unemployment_df)+1,] <- as.vector(MD_df_row)
    unemployment_df[nrow(unemployment_df)+1,] <- as.vector(VA_df_row)
  }
}
```

A.3 Data preprocessing

A.3.1 Preprocessing for CDC WONDER data via dplyr

```r
# preprocessing via dplyr: filter, select
preproc_df <- raw_opioid_df %>%

  # select: select df cols based on names
  dplyr::select('State', 'State Code',
                'Month', 'Month Code',
                'Deaths') %>%

  # mutate: create calc cols
  mutate(
    Date = lubridate::mdy(
      paste(
        substr(raw_opioid_df$'Month', 1, 3), # month abbrev/name (3-letter)
        '01', # 1st day of every month (arbitrary)
        substr( # year (4-digit)
          raw_opioid_df$'Month',
          str_length(raw_opioid_df$'Month')-3,
          str_length(raw_opioid_df$'Month')
        ), sep='-'))) %>%

  mutate(Date = yearmonth(Date)) %>% # convert to fable time types

  # filter: selecting df rows based on logical statements of values
  na.omit() %>%

  # convert to tsibble object and fill gaps
  as_tsibble(key=dplyr::starts_with('State'), index='Date') %>%
  fill_gaps()
```

A.3.2 Preprocessing for BLS data via dplyr

```r
# convert to tsibble object and fix dtypes
unemployment_df <- unemployment_df %>%
  mutate(‘Date‘ = yearmonth(‘Date‘)) %>%
  mutate(‘Noninstitutionalized.population‘=‘Noninstitutionalized.population‘ %>%
  str_replace_all(’,’, ’’) %>% as.numeric()) %>%
  mutate(‘Unemployed‘ = ‘Unemployed‘ %>%
  str_replace_all(’,’, ’’) %>%
  as.numeric()) %>%
  as_tsibble(key=’State’, index=’Date’) # convert to tsibble
```

A.4 Data train-test splitting

```r
VA_train <- VA_df %>% # training data: 2000 to 2020
  filter(Date > yearmonth(’2000-01’) & Date < yearmonth(’2020-01’))

VA_test <- VA_df %>% # test data: year 2020
  filter(Date > yearmonth(’2019-12’) & Date < yearmonth(’2021-01’))
```

A.5 Feature engineering

A.5.1 Box-Cox transformation of forecast variable via fabletools::box_cox()

```r
guerrero_lambda <- VA_df %>%
  # filter to training data range to prevent data leakage
  filter(Date > yearmonth(’2000-01’) & Date < yearmonth(’2020-01’)) %>%
  # feasts::guerrero() for automated lambda selection
  features(Deaths, features=guerrero) %>%
  pull(lambda_guerrero)

VA_df <- VA_df %>%
  mutate(Deaths.tf = box_cox(Deaths, lambda=guerrero_lambda))
```

A.5.2 Differencing predictor variables via tsibble::difference()

```r
VA_df <- VA_df %>%
  mutate(unemp_single_diff = unemp %>% difference(12)) %>%
  mutate(unemp_double_diff = unemp %>% difference(12) %>% difference(1)) %>%
  mutate(noninst_single_diff = noninst %>% difference(12)) %>%
  mutate(noninst_double_diff = noninst %>% difference(12) %>% difference(1))
```

A.6 Model specification and estimation via fabletools::model()

```r
fit <- VA_train %>%
  # estimate models via fabletools::model()
  model(
     # naive models
     naive =NAIVE(box_cox(Deaths,lambda=guerrero_lambda)),
     seasonal_naive =SNAIVE(box_cox(Deaths,lambda=guerrero_lambda)),

     # simple ARIMA models
     arima =ARIMA(box_cox(Deaths,lambda=guerrero_lambda)),
     arima_fourier =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                          PDQ(0,0,0) + fourier(K=6)),

     # regression w/ 1 predictor & ARIMA errors (pdq) & seasonal errors (PDQ)m
     ## predictor: unemp
     unemp_simple =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                         unemp),
     unemp_single_diff =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                              unemp_single_diff),
     unemp_double_diff =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                              unemp_double_diff),
     ## predictor: noninst
     noninst_simple =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~ noninst),
     noninst_single_diff =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                                noninst_single_diff),
     noninst_double_diff =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                                noninst_double_diff),

     # regression w/ 1 predictor & ARIMA errors (pdq) & fourier seasonal errors
     ## predictor: unemp
     unemp_simple_fourier =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                                 unemp + PDQ(0,0,0) + fourier(K=6)),
     unemp_single_diff_fourier =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                                       unemp_single_diff + PDQ(0,0,0) +
                                       fourier(K=6)),
     unemp_double_diff_fourier =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                                       unemp_double_diff + PDQ(0,0,0) +
                                       fourier(K=6)),
     ## predictor: noninst
     noninst_simple_fourier =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                                    noninst + PDQ(0,0,0) + fourier(K=6)),
     noninst_single_diff_fourier =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                                         noninst_single_diff + PDQ(0,0,0) +
                                         fourier(K=6)),
     noninst_double_diff_fourier =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                                         noninst_double_diff + PDQ(0,0,0) +
                                         fourier(K=6)),

     # regression w/ 2 predictors & ARIMA & (pdq) & seasonal errors (PDQ)m
     simple =ARIMA(box_cox(Deaths, lambda=guerrero_lambda) ~ unemp + noninst),
     single_diff =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                        unemp_single_diff + noninst_single_diff),
     double_diff =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                        unemp_double_diff + noninst_double_diff),

     # regression w/ 2 predictors & ARIMA errors (pdq) & fourier seasonal errors
     simple_fourier =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~ unemp +
                           noninst + PDQ(0,0,0) + fourier(K=6)),
     single_diff_fourier =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                                unemp_single_diff + noninst_single_diff +
                                PDQ(0,0,0) + fourier(K=6)),
     double_diff_fourier =ARIMA(box_cox(Deaths,lambda=guerrero_lambda) ~
                                unemp_double_diff + noninst_double_diff +
                                PDQ(0,0,0) + fourier(K=6))
```

```
)
```

A.7 Selection of "top five" models by AICc

```
# save names of 5 lowest AICc value models
fit %>% glance() %>% slice_min(AICc, n=5) %>% select(`.model`) %>% pull() -> top
   _five
```

A.8 Generating model forecasts via `fabletools::forecast()`

```
# print top 5 forecast accuracy: MAE & MAPE
fit %>%
  forecast(VA_test) %>%
  accuracy(VA_test) %>%
  filter(`.model` %in% top_five) %>%
  arrange(`MAE`) %>%
  select(c(`.model`,`MAPE`, `MAE`)

# plot forcasts of top 5 models
for (top_model in top_five){
  test <- fit %>% # close-up plot w/ test overlay
    forecast(VA_test) %>%
    dplyr::filter(`.model` == top_model) %>%
    autoplot(VA_test) +
    geom_vline(xintercept = as.numeric(as.Date('2020-01-01')),
               color='red', linetype = 'dashed') +
    ggtitle('') +
    theme_classic() +
    theme(legend.position='none')

  train <- fit %>% # far-away plot
    forecast(VA_test) %>%
    dplyr::filter(`.model` == top_model) %>%
    autoplot(VA_train) +
    geom_vline(xintercept = as.numeric(as.Date('2020-01-01')),
               color='red', linetype = 'dashed') +
    ggtitle(top_model) +
    theme_classic() +
    theme(legend.position='top')

  grid.arrange(train, test, nrow=2)
}
```