Time Series Forecasting for Mass Shootings in the U.S. (1966-2021)

PSTAT 174 Fall 2022 Final Project - Forecasting in R

Ezra Reyes Aguimatang

University of California, Santa Barbara

Contents

1 Abstract

According to the 2^{nd} Amendment of the United States Constitution, it states *"A well regulated Militia, being necessary to the security of a free State, the right of the people to keep and bear Arms, shall not be infringed"*. In light of the increase of mass shootings and more frequent gun violence in recent years, the discussion concerning the 2^{nd} Amendment has gravitated towards becoming a large looming controversy over the American people, and in turn these discussion are seemingly intensifying a political polarization across the country.

In this study, we will be utilizing a dataset from the Kaggle Database: Mass Shootings in the United States of America, which has recorded various attributes from a whopping 398 mass shootings that occurred in the United States between the years $1966 - 2021$. With this dataset, the aim of this study is to use *Time Series Analysis* and *Time Series Forecasting* in `R` as a means of predicting the future of the United States and its people, in terms of the presence of mass shootings and gun violence.

Focusing on the `Date` and `Total_Victims` features of our dataset, this study will be forecasting the total number of victims (both fatalities and injuries) in the United States for future mass shootings if the state of the nation does not change.

PROCESS

CONCLUSION

LIMITATIONS

2 Introduction

The political divide among the people of the United States has always been present, but in more recent years, this gap between those of different political affiliations appears to widen more and more. With the growing use of social media to spread videos and ideas in a more efficient manner, new issues and conversations are brought to light, being more easily accessible than ever before. One of these topics of conversation is the issue of increased *Mass Shootings* and *Gun Violence* in the United States. According to various sources, the United States has consistently been the country awarded with the highest number of mass shootings. Although this idea is objective, factual, numerical, and has been widely known for years, the numbers of mass shootings per year seems to grow exponentially. A controversial topic that seems to be choosing between protecting the lives of primarily children in schools or protecting ones' selves with arms, is an issue in this country that has yet to be solved.

As stated prior, the Mass Shootings in the United States of America (1966-2021) dataset from Kaggle will be utilized in this study, which was originally compiled by Zeeshan-Ul-Hassan Usmani. This dataset was allegedly created using multiple sources including "Wikipedia, Mother Jones, Stanford, USA Today and other web sources" (as stated in his acknowledgements), and was given a Kaggle usability score of 7.94: a score out of 10 that Kaggle assigns datasets based on their completeness, credibility, and compatibility in order to inform users the validity of the data that is presented.

Rather than acting as a model of inference showing all the different factors that could explain the increase in the numbers of mass shootings, gun violence, and victims, this time series model will serve as a tool to forecast this dataset and show how these numbers will potentially look like in our near future if this current trend is to continue. Although this study isn't meant to take into account all or even many factors as to why this increase in mass shootings is occurring, creating a window into what this forcasted data may appear like in the future may hopefully incite some further discussion and possible urgency in the hands of all people who have the power to change the current and future state of this nation.

2.1 R Packages

For this study, data cleaning and time series analysis will be completed using the `R` programming language. We will be utilizing the following packages in `R` throughout the duration of this study...

```
library(knitr)
library(ggplot2)
library(ggfortify)
library(tinytex)
library(dplyr)
library(tidyverse)
library(MASS)
library(tseries)
library(devtools)
library(forecast)
library(MuMIn)
```

3 Dataset Overview and Cleaning

The Mass Shootings in the United States of America database has multiple files with different feature variables with some having more up to data observations than others. In this study the dataset version that will be used is the `Mass Shootings Dataset.csv` and `Mass shooting data.csv` files.

Before working with this data, we will first read the file in as an `R` dataframe to view its features and clean it as necessary.

```
Mass_Shootings_Dataset_csv <- read_csv("archive/Mass Shootings Dataset.csv")
Mass_Shootings_Dataset_csv2 <- read_csv("archive/Mass shooting data.csv")
```

When taking a look at our dataset, we will see that we have 398 rows of observations and 13 column feature variables available. Since this dataset was not created specifically for the context and purposes of this project, a little cleaning is necessary before proceeding.

This data was not explicitly meant for time series analysis, so we will begin by changing the data types of the `Date` variable and also rename a few column feature variable names for better readability.

```
Mass_Shootings_Dataset_csv$Date <- as.POSIXct(Mass_Shootings_Dataset_csv$Date,
                                              format="%m/%d/%Y")
colnames(Mass_Shootings_Dataset_csv)[1] <- "Event_ID"
colnames(Mass_Shootings_Dataset_csv)[8] <- "Total_Victims"
colnames(Mass_Shootings_Dataset_csv)[9] <- "Mental_Health_Issues"
```

	Feature Column Data Types
Event_ID	num
Title	chr
Location	chr
Date	POSIXct
Summary	chr
Fatalities	num
Injured	num
Total_Victims	num
Mental_Health_Issues	chr
Race	chr
Gender	chr
Latitude	num
Longitude	num

After looking at the structure of our data frame using the `str()` function, we see the data types of each of our column feature variables in the table above. For the intentions of this study, the variables that are most useful to us will be our `Date` and `Total_Victims` variables.

Now that we've done a little data cleaning and have gotten some insight into the feature columns of our data, we'll now take a look at the first 5 observations of our data for the 13 feature columns we have available to us.

```
head(Mass_Shootings_Dataset_csv[0:4], 5)
```

Event_ID	Title	Location	Date
1	Las Vegas Strip mass shooting	Las Vegas, NV	2017-10-01
2	San Francisco UPS shooting	San Francisco, CA	2017-06-14
3	Pennsylvania supermarket shooting	Tunkhannock, PA	2017-06-07
4	Florida awning manufacturer shooting	Orlando, Florida	2017-06-05
5	Rural Ohio nursing home shooting	Kirkersville, Ohio	2017-05-12

```
head(Mass_Shootings_Dataset_csv[5], 5)
```

Summary
NA
Jimmy Lam, 38, fatally shot three coworkers and wounded two others inside a UPS facility in San Francisco. Lam killed himself as law enforcement officers responded to the scene.
Randy Stair, a 24-year-old worker at Weis grocery fatally shot three of his fellow employees. He reportedly fired 59 rounds with a pair of shotguns before turning the gun on himself as another co-worker fled the scene for help and law enforcement responded.
John Robert Neumann, Jr., 45, a former employee of manufacturer Fiamma Inc. fatally shot five workers at the company, and then killed himself on the scene. He'd been fired from the company in April. The attack took place a week before the one-year anniversary of the Orlando nightclub massacre.
Thomas Hartless, 43, shot and killed a former girlfriend and another employee of a nursing home, and then fatally shot the Kirkersville police chief responding to the scene. Hartless' former girlfriend had recently obtained a court protection order against Hartless. Investigators later found more than 60 firearms in the home of Hartless, who was also found dead at the scene of the attack.

```
head(Mass_Shootings_Dataset_csv[6:13], 5)
```

Fatalities	Injured	Total_Victims	Mental_Health_Issues	Race	Gender	Latitude	Longitude
58	515	573	Unclear	NA	NA	NA	NA
3	2	5	Yes	Asian	M	NA	NA
3	0	3	Unclear	White	M	NA	NA
5	0	5	Unclear	NA	M	NA	NA
3	0	3	Yes	White	M	NA	NA

3.1 Making Data Time Series Appropriate

Since this data was not originally created for the purposes of Time Series Analysis, some necessary techniques of data cleaning and feature engineering are implemented before we can create our Time Series object.

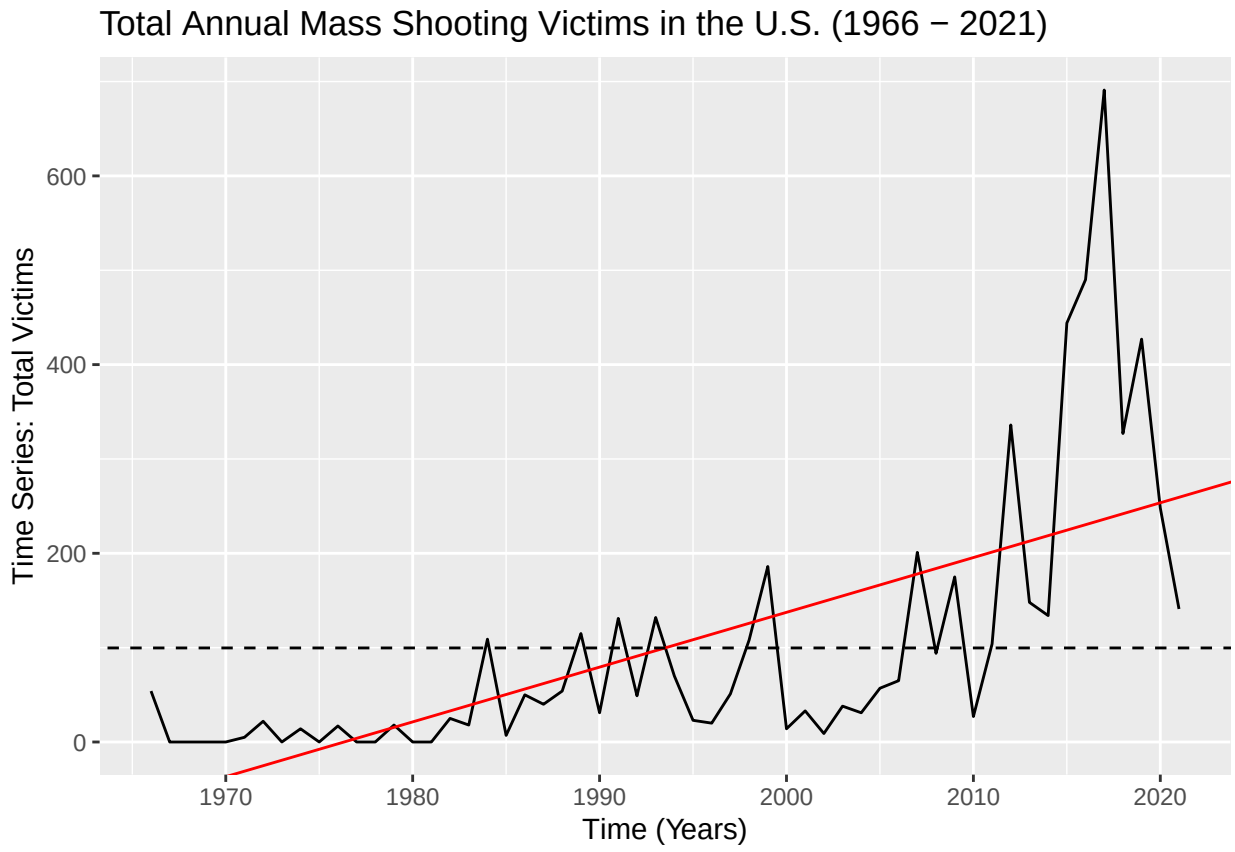
```
# Subsetting only "Total_Victims" and "Date columns"
mass_shootings_df <- Mass_Shootings_Dataset_csv1[, c("Total_Victims", "Date")]
##head(mass_shootings_df)

# Adding a Year identifying Column based on the POSIXct "Date" Column
mass_shootings_df$Year <- as.numeric(format(mass_shootings_df$Date, format="%Y"))
##head(mass_shootings_df)
##unique(mass_shootings_df$Year)

# Creating a Data Frame of Values for Missing Years
Total_Victims <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
Date <- c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA)
Year <- c(1981, 1980, 1978, 1977, 1975, 1973, 1970, 1969, 1968, 1967)
missing_obs_df <- data.frame(Total_Victims, Date, Year)

# Binding the Missing Years Data Frame to the Original Data Frame
mass_shootings_newdf <- rbind(mass_shootings_df, missing_obs_df)
mass_shootings_newdf <- mass_shootings_newdf[order(mass_shootings_newdf$Year,
                                                   decreasing=TRUE),]

# Creating a New Data Frame Summing and Grouping Row Observations by Year
Mass_Shootings_df <- mass_shootings_newdf
Mass_Shootings_df <- aggregate(Total_Victims ~ Year, Mass_Shootings_df, sum)
# Mass_Shootings_df
```

As seen above, we've subset the `Total_Victims` and `Date` columns being the two important aspects of our original data set that we want to analyze. Next, knowing that the `ts()` can only create a time series object with data that has no gaps in time, or `Date` in our case, a few steps had to be taken. Firstly, our `Date` column was originally in the format `%m/%d/%Y`, which denoted the date a mass shooting event had occurred. However, between the years $1966 - 2021$ there was thankfully not an observation available for each day in this time period. Therefore, we had started with large gaps in our data. On top of this issue, the data not only had gaps between the day of events, but also had various missing months and years. As can be seen above we had a total of 10 missing year (1981, 1980, 1978, 1977, 1975, 1973, 1970, 1969, 1968, 1967). So in order to solve all these issues at once, I decided it was best to add a new column (`Year`) to our data to ID each observation by the year of their `Date`. By doing so, it was possible to aggregate the sums of `Total_Victims` for observations that have the same value for our new `Year` column, and lastly add observations for the missing `Years` in our data and setting all their values of `Total_Victims` to `0` (indicating that there were no victims of mass shootings that year).

4 Data Split and Time Series Plot

Now that we have a good idea of what exactly our data represents and finished the process of cleaning, we now want to view a plot of our time series object to now get a visual representation of our data.

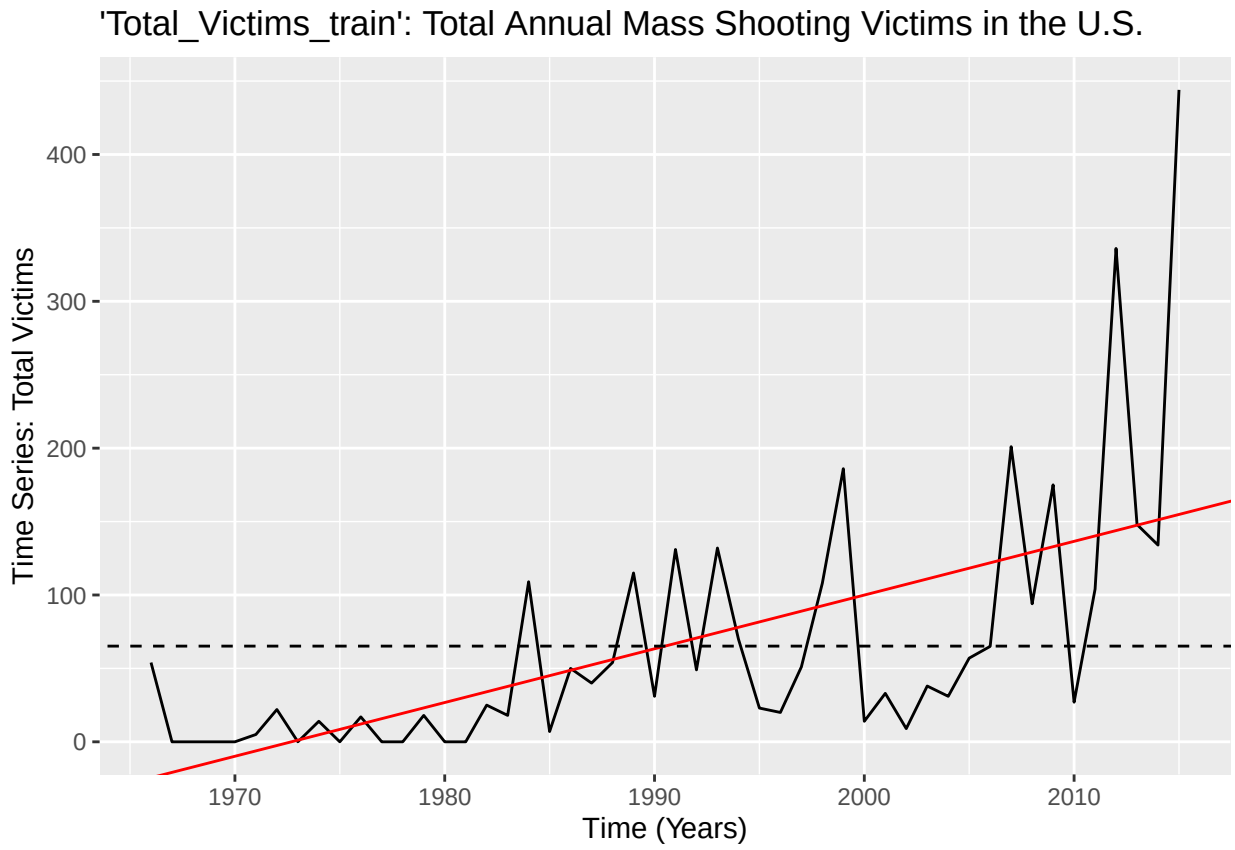
```
Total_Victims_ts <- ts(data=Mass_Shootings_df[2],
                       start=1966, end=2021, frequency=1)
```



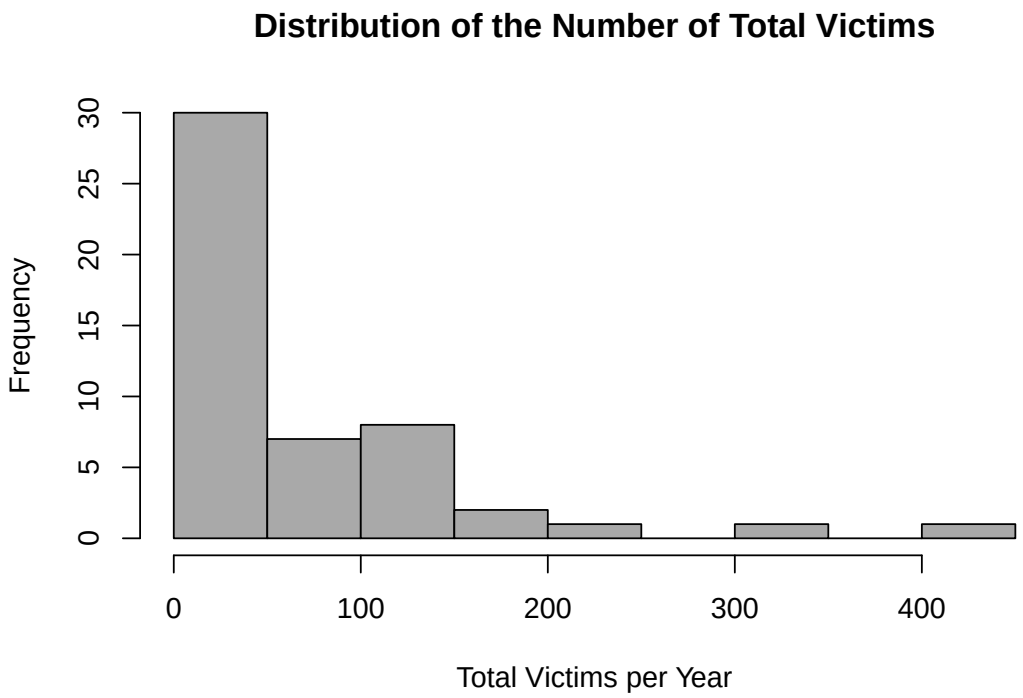
Since our time series `Total_Victims_ts` is an annual representation of total victims of mass shootings for each year, the `frequency` argument of the `ts()` function will be set to 1. From the plot above we do see a fairly strong increasing/upward trend, which allows us to conclude there has been a very clear gradual increase in the number of total victims of mass shootings (both fatalities and injuries) in the United States from 1966 − 2021. Knowing that this data pertains to the number of victims for mass shootings per year, and as we can see from the behavior of the plot, we start to conclude that this time series is most likely not seasonal but rather only has a fairly noticeable trend.

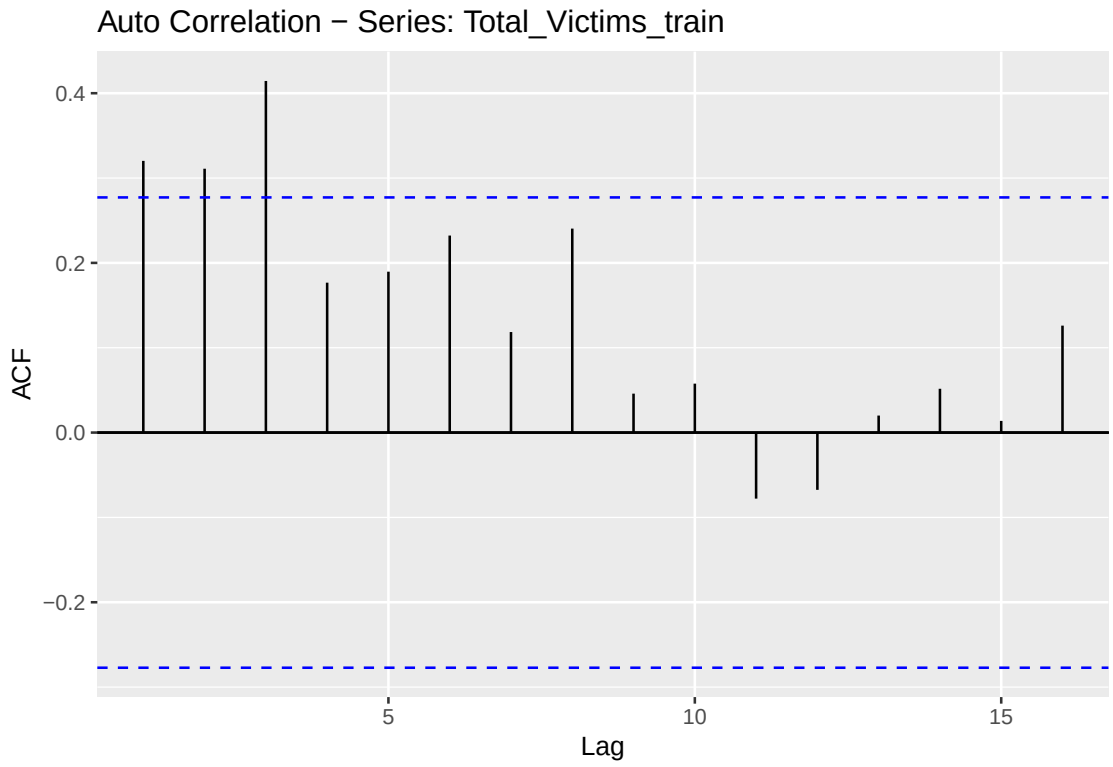
Now that we've created our Time Series object, before starting the process of analyzing our data for model selection, let us first create an 80/20 split our data into a training and testing set which we will call `Total_Victims_train` and `Total_Victims_test`.

```
Total_Victims_train <- Mass_Shootings_df[1:50, ]
Total_Victims_test <- Mass_Shootings_df[50:56, ]
```



Now, before starting the process of analyzing our data for model selection, let us first take a closer look at our time series using a histogram and ACF plot.

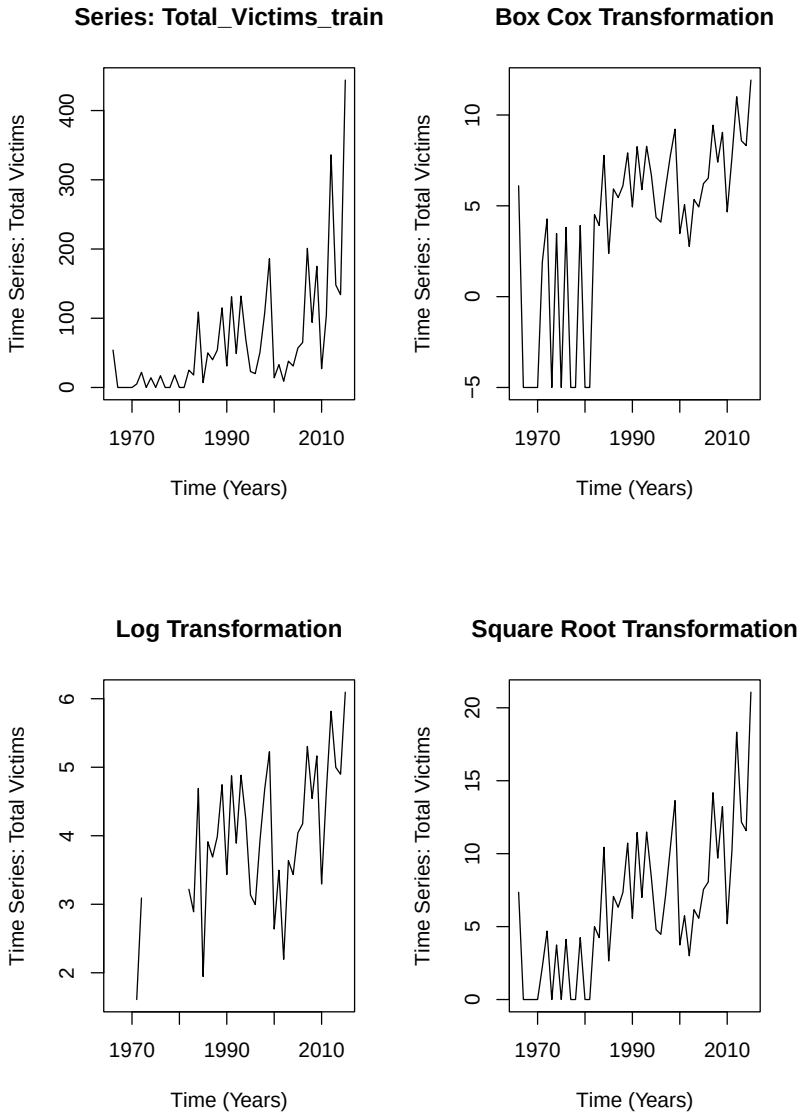

Distribution of the Number of Total Victims
Frequency
0
5
10
15
20
25
30
0
100
200
300
400
Total Victims per Year


Auto Correlation − Series: Total_Victims_train
ACF
0.4
0.2
0.0
−0.2
5
10
15
Lag

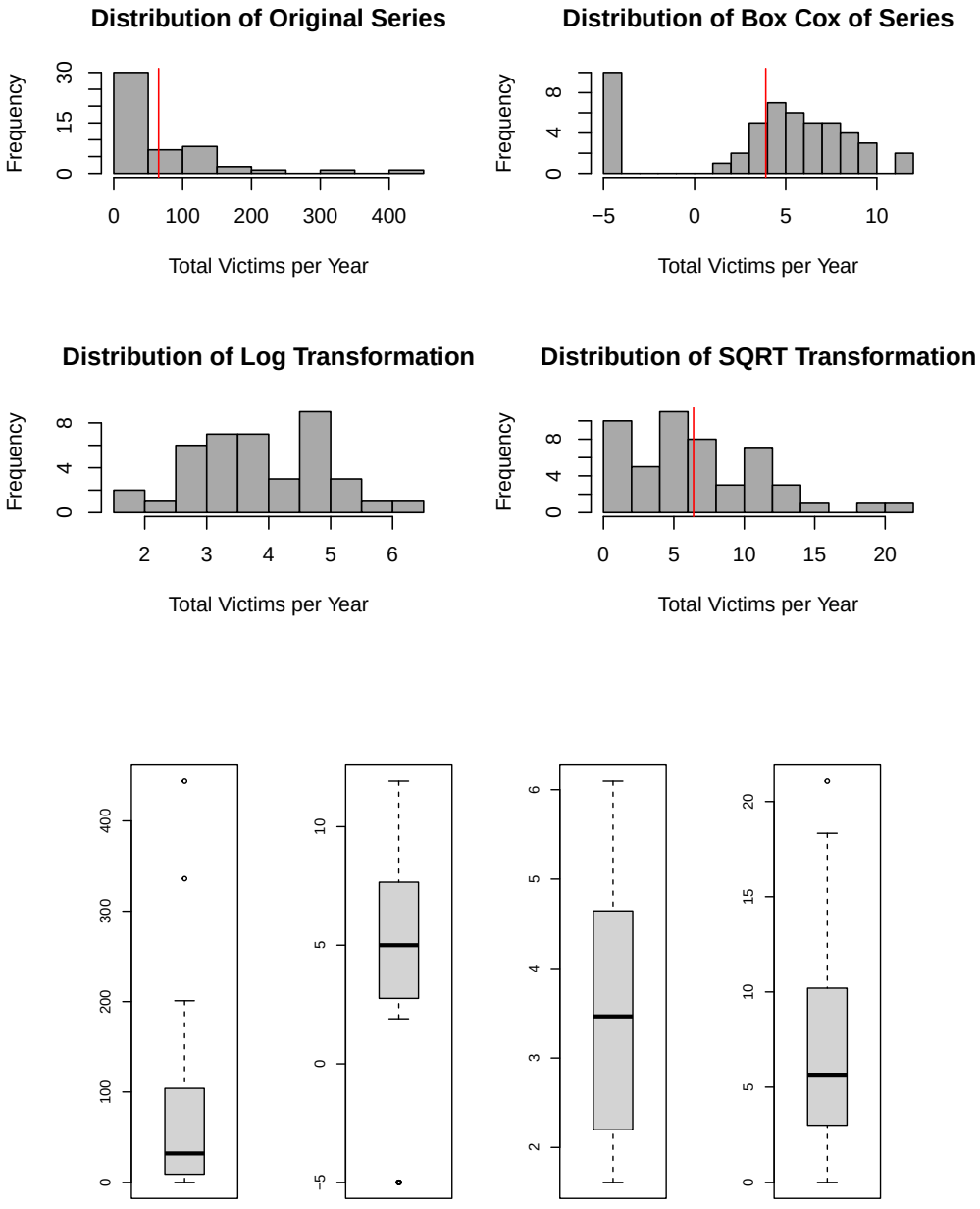
4.1 Transformations and Differencing

After looking at the initial time series plot of our data, the next step one considers in the model selection process are **Transformations** and **Differencings**. The purposes behind these techniques is to essentially manipulate our original time series model so that is it more **Stationary** and removes any trace of **Trend** or **Seasonality**. Looking at the 2 plots shown in the previous page ("Distribution of the Number of Total Victims" and "Auto Correlation - Series: Total_Victims_train"), we see the distribution of the amount of total victims per year has a very strong right-skew where many years have in between 0 and 50 victims of mass shootings. Now as we can see from the second plot, the ACF values confirm our suspicions that we do have a strong trend.

For these reasons, we want to transform our data in the hopes that we can stabilize variance and take a step closer to a stationary set of data. For this series we will attempt a Box-Cox Transformation, Log Transformation, and a Square-Root Transformation.



After examining the time series plots above of the possible transformations, we can somewhat confidently say that visually, these transformations seem to have greatly reduced variance especially for the years after 2010 when the amount of victims skyrocketed to over 600 by 2017. Although all of these plots seem to look like great transformations, we must further investigate which is the best transformation for our series.



Looking at visual the aspects of our plots above alone, we can conclude that our transformations that bring our original series, `Total_Victims_train`, looking the closest to a normal distribution, are the Box-Cox and Log Transformations. However, although our box plots for the Box-Cox

Transformation and Log Transformation (middle two box plots) seem to indicate that we have successfully normalized our data, we see from our histograms that the mean of our Log Transformation is not present in our plot and `R` informs us that we have a `-inf` mean value. In `R`, although a Log Transformation is perfect for discrete time series, it cannot handle values of zero hence our `-inf` mean value as well as an inability to calculate our variance or output an ACF or PACF plot. For this reason, we are not able utilize a Log Transformation for this study.

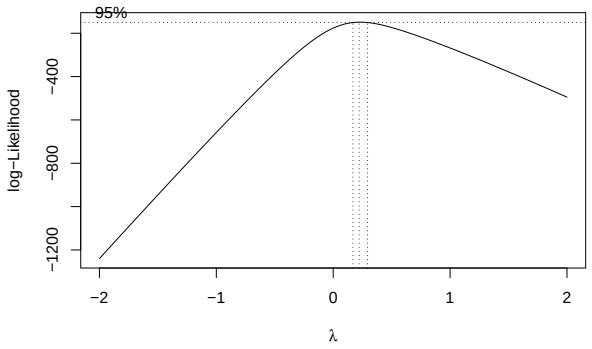
Now to choose the best transformation for our series, well take a look at the table below:

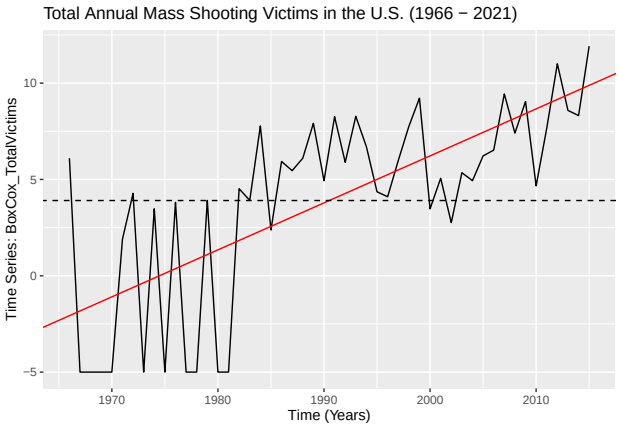
Series	Variance	Mean	Description
Total_Victims_train	7553.334	65.180	Original Time Series
BoxCox_TotalVictims	24.571	3.906	Box Cox Transformed Series
sqrt_TotalVictims	24.726	6.399	Square Root Transformed Series

Looking at the table above as well as keeping our previous plots in mind. Although both the Box-Cox Transformation and the Square Root Transformation stabilize our variance, we can conclude the Box-Cox Transformation is most suitable as it not only stabilizes our variance best, but it also brings the mean much closer to 0. To calculate our transformation we will use our lambda value of $\lambda \approx 0.2$ and substitute it into the following transformation:

$$\frac{1}{\lambda}(X_t^{\lambda} - 1), \text{ where } X_t = \text{ Total\_Victims\_ts}$$

we can the 95% confidence interval for our value of λ in our Log-Likelihood plot below, and we can also see that the time series plot of our Box-Cox transformation sufficiently seems to reduce variance and trend in the visual sense as well.





Now that we have narrowed down the suitable transformations down, finally choosing a Box-Cox Transformation for our series, we take a look at the levels of differencing that further stabilize the variance of our series while also bringing the mean to approximately zero or removing trend. Additionally, although we do not suspect seasonality, we will be checking for any sign of its presence in our series.

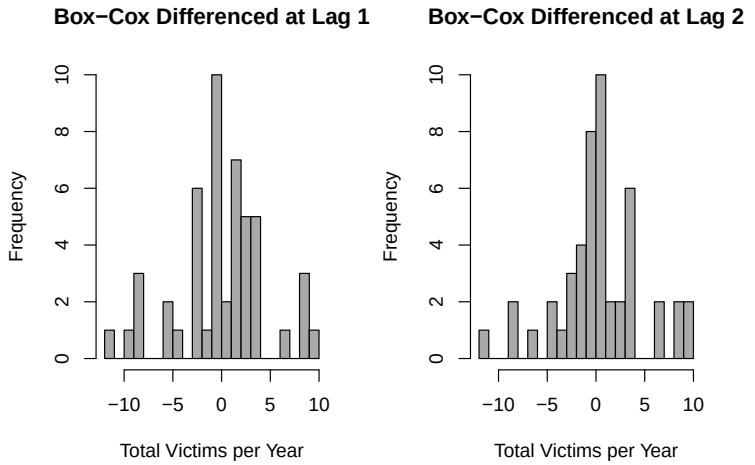
As we can see from our 4 Time Series plots above of our original Box-Cox transformed series, its differencing at lag 1, its differencing at lag 2, and at lag 3; we see that (visually) a differencing at lag 1, 2, and 3 all seem to stabilize our variance and reduce trend best in combination with a Box-Cox transformation: where at lag 4 and above there was a fairly significant visual increase in trend. Additionally, for the sake of discovering any seasonality, a first order differencing for higher values of lag were also calculated and plotted, but as we had guessed there did not seem to be evidence of seasonality after differencing and concluded that there was clear over differencing for values of lag greater than 4. However, after calculating the variances at each level of differencing, we can take a closer look to verify at which lag value(s) does differencing most reduce trend and stabilize variance:

Series	Variance	Mean	Description
Total_Victims_train	7553.334	65.180	Original Time Series Object
BoxCox_TotalVictims	24.571	3.906	Box-Cox Transformation (prior to differencing)
BoxCox_TotalVictims_diff1	21.718	0.119	Box-Cox Transformation (difference at lag 1)
BoxCox_TotalVictims_diff2	19.405	0.399	Box-Cox Transformation (difference at lag 2)
BoxCox_TotalVictims_diff3	21.114	0.696	Box-Cox Transformation (difference at lag 3)

As evidenced by the table above, when differencing at lag 3, we see an increase in variance and can therefore suspect over-differencing: concluding that differencing at lag 1 and lag 2 are most suitable to making our series stationary. Now we can take a look at some plots to begin our model selection process.

4.2 Distribution of Data (after transforming and differencing)

Now that we've taken a quick look at our ACF and PACF plots, lets do a check for normality by viewing our Box-Cox Transformed data after differencing. To do this we'll see our data's distribution using a histogram and we will also view a Normal Q-Q Plot.



To no sur

```
##
##  Box-Ljung test
##
## data:  BoxCox_TotalVictims
## X-squared = 89.836, df = 20, p-value = 7.917e-11

## Warning in kpss.test(BoxCox_TotalVictims, null = "Trend"): p-value greater than
## printed p-value
```

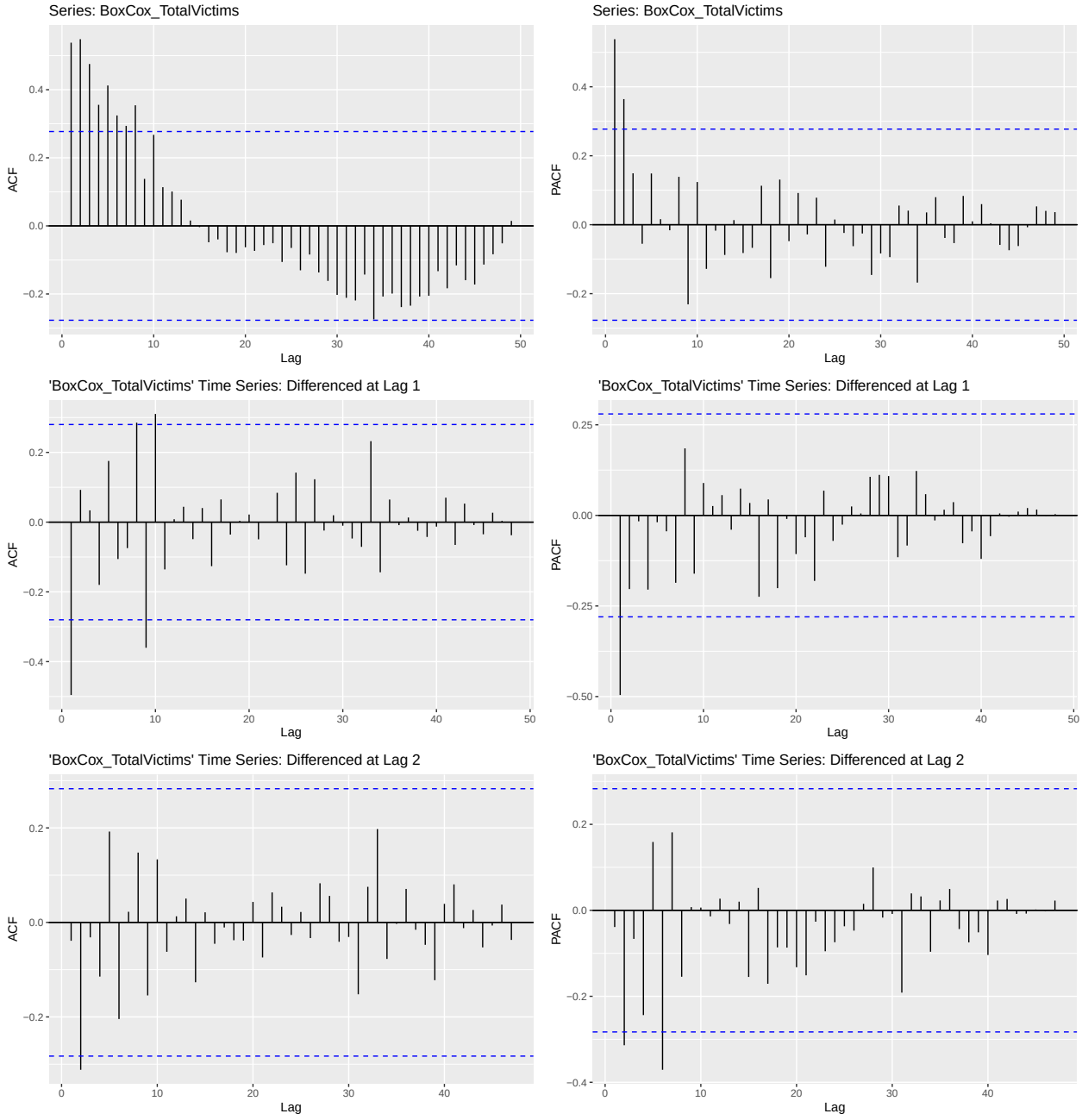
```
##
##  KPSS Test for Trend Stationarity
##
## data:  BoxCox_TotalVictims
## KPSS Trend = 0.11341, Truncation lag parameter = 3, p-value = 0.1

##
##  Augmented Dickey-Fuller Test
##
## data:  BoxCox_TotalVictims
## Dickey-Fuller = -2.8468, Lag order = 3, p-value = 0.2346
## alternative hypothesis: stationary

##
##  Box-Ljung test
##
## data:  BoxCox_TotalVictims_diff1
## X-squared = 40.387, df = 20, p-value = 0.004461

## Warning in kpss.test(BoxCox_TotalVictims_diff1, null = "Trend"): p-value
## greater than printed p-value

##
##  KPSS Test for Trend Stationarity
##
## data:  BoxCox_TotalVictims_diff1
## KPSS Trend = 0.062511, Truncation lag parameter = 3, p-value = 0.1

## Warning in adf.test(BoxCox_TotalVictims_diff1): p-value smaller than printed
## p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  BoxCox_TotalVictims_diff1
## Dickey-Fuller = -5.8826, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary

##
##  Box-Ljung test
##
## data:  BoxCox_TotalVictims_diff2
## X-squared = 16.461, df = 20, p-value = 0.6876

## Warning in kpss.test(BoxCox_TotalVictims_diff2, null = "Trend"): p-value
## greater than printed p-value
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  BoxCox_TotalVictims_diff2
## KPSS Trend = 0.058457, Truncation lag parameter = 3, p-value = 0.1

## Warning in adf.test(BoxCox_TotalVictims_diff2): p-value smaller than printed
## p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  BoxCox_TotalVictims_diff2
## Dickey-Fuller = -4.9993, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary

##
##  Box-Ljung test
##
## data:  BoxCox_TotalVictims_diff3
## X-squared = 15.949, df = 20, p-value = 0.7198

## Warning in kpss.test(BoxCox_TotalVictims_diff3, null = "Trend"): p-value
## greater than printed p-value

##
##  KPSS Test for Trend Stationarity
##
## data:  BoxCox_TotalVictims_diff3
## KPSS Trend = 0.05725, Truncation lag parameter = 3, p-value = 0.1

##
##  Augmented Dickey-Fuller Test
##
## data:  BoxCox_TotalVictims_diff3
## Dickey-Fuller = -3.9654, Lag order = 3, p-value = 0.01915
## alternative hypothesis: stationary
```

5 Model Identification

5.1 ACF and PACF (after transforming and differencing)

The next step in the model identification process is looking at the **ACF** and **PACF** plots of our time series model, also known as the **Auto Correlation Function** and **Partial Correlation Function**. As we discussed in the previous section, we saw in our table of variances and means that a differencing at lag 3 appeared to be over differencing, so well take a look at the ACF and PACF plots for differencing at lags 1 and 2 in comparison to our original transformation of the time series.



In comparison to the ACF and PACF plots of our original Box-Cox transformed series, differencing

at lag 1 and lag 2 both seem to have made our plots of ACF and PACF much more stationary, as evidenced by rapid the decay of auto correlation and partial auto correlations. We also seem to have successfully first order differenced our series seeing no presence of seasonality. However, although not a defining factor of a non-stationary series, it should be noted that the ACF and PACF plot of the series differenced at lag 2 has a value of 0 (or below the confidence interval) at the first lag and a value above the confidence interval at the second lag. Due to this, it is possible that our series may not be completely stationary or the best combination of differencing, but this can be verified later through testing.

Knowing this, we'll begin coming up with some suitable models.

5.2 Model Selection

AKA Fit Model

CORRECT PARAGRAPHS BELOW In R, the boxcox() function takes a time series object as input, and returns a list containing the transformed data, along with the lambda value. The lambda value is used to reverse the Box-Cox transformation after the model is fit, in order to make predictions on the original scale of the data.

To calculate the optimal lambda value, the boxcox() function performs a search over a range of lambda values and selects the one that maximizes the log-likelihood function. This is done by using the MASS package in R.

Once the optimal lambda value has been determined using the boxcox() function, it can be used in the arima() or other modeling functions in R to fit a stationary time series model. The lambda value can be passed to the lambda argument in the arima() function, which will apply the inverse Box-Cox transformation to the model predictions.

```
##
## Call:
## arima(x = BoxCox_TotalVictims_diff1, order = c(1, 1, 0), method = "ML")
##
## Coefficients:
##           ar1
##       -0.7004
## s.e.   0.1026
##
## sigma^2 estimated as 31.77:  log likelihood = -151.45,  aic = 306.9
##
## Call:
## arima(x = BoxCox_TotalVictims_diff1, order = c(0, 1, 1), method = "ML")
##
## Coefficients:
##           ma1
##       -1.0000
## s.e.   0.0572
##
## sigma^2 estimated as 21.72:  log likelihood = -143.93,  aic = 291.86
```

```
##
## Call:
## arima(x = BoxCox_TotalVictims_diff1, order = c(1, 1, 1), method = "ML")
##
## Coefficients:
##          ar1      ma1
##       -0.5477  -1.0000
## s.e.   0.1290   0.0597
##
## sigma^2 estimated as 15.57:  log likelihood = -136.55,  aic = 279.1
##
## Call:
## arima(x = BoxCox_TotalVictims_diff1, order = c(1, 0, 0), method = "ML")
##
## Coefficients:
##          ar1  intercept
##       -0.5585     0.1761
## s.e.   0.1271     0.3606
##
## sigma^2 estimated as 15.24:  log likelihood = -136.46,  aic = 278.91
##
## Call:
## arima(x = BoxCox_TotalVictims_diff1, order = c(1, 0, 1), method = "ML")
##
## Coefficients:
##          ar1      ma1  intercept
##       -0.0680  -0.8208     0.2402
## s.e.   0.2771   0.2383     0.0949
##
## sigma^2 estimated as 12.6:  log likelihood = -132.22,  aic = 272.44
##
## Call:
## arima(x = BoxCox_TotalVictims_diff1, order = c(0, 0, 1), method = "ML")
##
## Coefficients:
##          ma1  intercept
##       -0.8818     0.2415
## s.e.   0.1859     0.0721
##
## sigma^2 estimated as 12.54:  log likelihood = -132.24,  aic = 270.48

## [1] Worst/Largest AICc: ARIMA(1, 1, 0)

## [1] 307.1699

## [1] 5th Lowest AICc: ARIMA(0, 1, 1)

## [1] 292.1284
```

```
## [1] 4th Lowest AICc: ARIMA(1, 1, 1)
```

```
## [1] 279.644
```

```
## [1] 3rd Lowest AICc: ARIMA(1, 0, 0)
```

```
## [1] 279.4466
```

```
## [1] 2nd Lowest AICc: ARIMA(1, 0, 1)
```

```
## [1] 273.3521
```

```
## [1] Best/Lowest AICc: ARIMA(0, 0, 1)
```

```
## [1] 271.0137
```

```
# use ar() function?
## ex. ar(data, aic=TRUE, order.max=NULL, method=c("...")
```

5.3 Model Estimation

checking model parameters

5.4 Model Diagnostics

Using different performance metrics (ex. `Box.test()` or `shapiro.test()`), or testing the normality assumption through plotting (i.e. `hist()`, `qqnorm()`, or `qqline()`).

```
# Box.test()
# shapiro.test()
# hist()
# qqnorm()
# qqline()
```

```
# Evaluating the residuals of ARIMA(1, 0, 1), our 2nd best fitting model, according to AICc
res_arima101_fit <- residuals(arima101_fit)

mean_res_arima101 <- mean(res_arima101_fit)
stdev_res_arima101 <- sqrt(var(res_arima101_fit))
```

```
# Evaluating the residuals of ARIMA(0, 0, 1), our best fitting model, according to AICc
res_arima001_fit <- residuals(arima001_fit)
```

6 Data Forecasting

Forecast $n + 1 \ldots$

```
# pred()
# points()
# lines()
```

7 Conclusion

BLANK

8 References

BLANK

9 Appendix

BLANK