THE UNIVERSITY OF CHICAGO

Chicago Traffic Crashes Forecasting 2016 – 2022

MSCA 31006 Time Series Analysis – Spring 2022

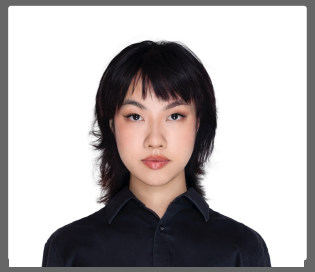
Allen Dong, Xin Ge, Kenneth Jin, Ren Liu, Pete Ryan

OUR TEAM

Allen Dong

Xin Ge

Kenneth Jin



Ren Liu

Pete Ryan

AGENDA

Introduction	Insights for improving traffic safety in Chicago based on weekly crash analysis and weather conditions.
Data	Chicago Data Portal – Regularly updated records of daily traffic crashes and related details.
Analysis	Experimental algorithm and results
Model Selection	Final model based on selection criteria
Conclusion	Project summarization and future work

PROBLEM STATEMENT

Combine weekly crash totals over 6-years with weekly averages of visibility and precipitation to analyze trends and patterns in Chicago vehicle crashes. Use time-series techniques to forecast future events and provide insights for safer driving.

BUSINESS USE CASE

- Improve city transportation efficiency

- Setting up additional precautions in the futures to prevent spikes in traffic crashes for seasons & trends

Data

Traffic Crashes

Chicago Data Portal – Regularly updated records of daily traffic crashes and related details such as Admin. Info, Road Info, Environment, Location, Damage ...

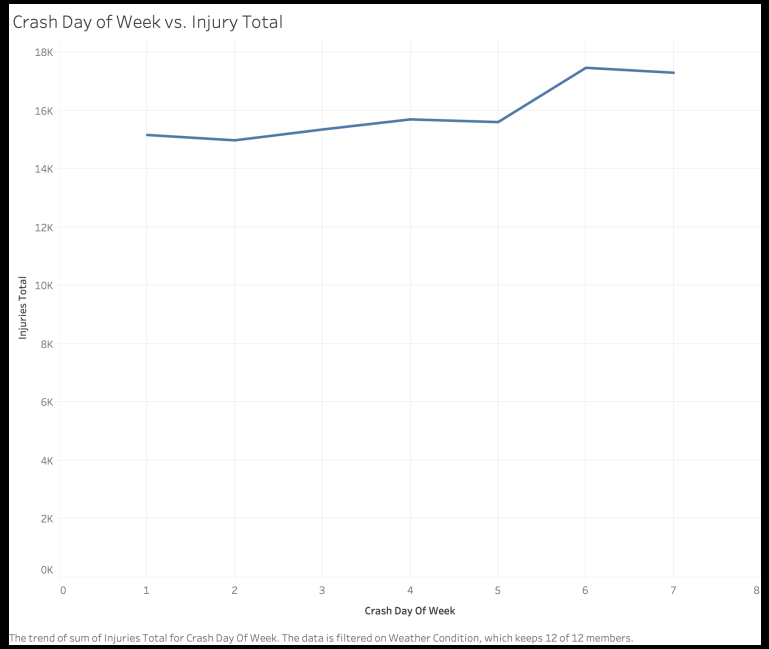
Rows 614K
Columns 49

Weather

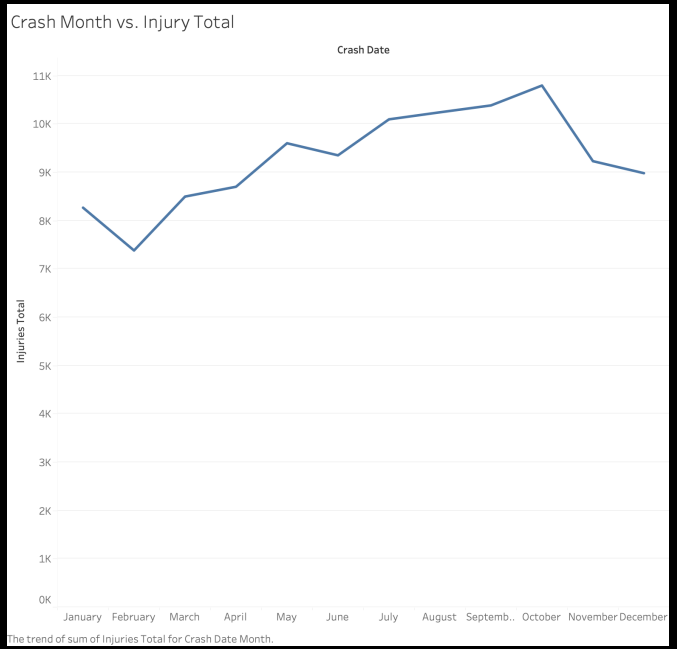
Chicago hourly weather history

- Total precipitation (In)
- Visibility (Miles)

EXPLORATORY DATA ANALYSIS

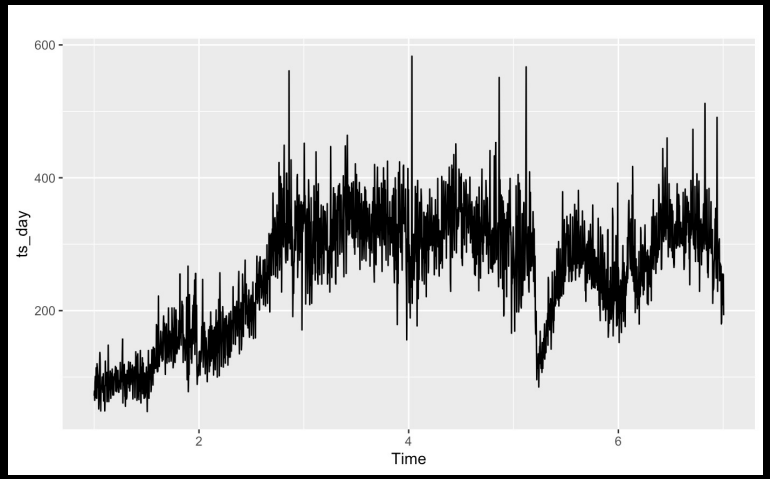


- Crash Day of Week vs. Total Injury

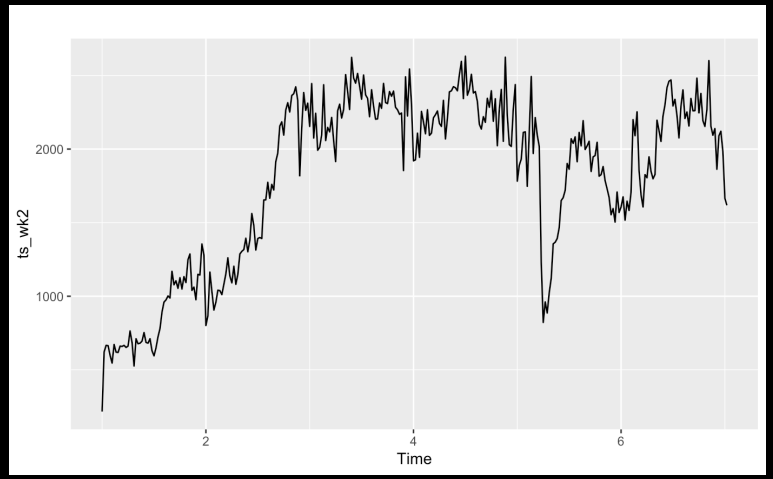


- Crash Month vs. Total Injury

DATA PREPROCESSING



Time Series plot of daily crash number



Time Series plot of weekly crash number

By comparing daily and weekly data, daily data shows more complex seasonal patterns due to the large volume. So, we will use the weekly data for experimental analysis

DATA PREPROCESSING

STEP 1

- Extract Date from Traffic Crashes Dataset and export it

```
df = pd.read_csv('Traffic_Crashes.csv')

ts = df['CRASH_DATE']

df['CRASH_DATE'] = df['CRASH_DATE'].str.slice(0, 10)

ts = df['CRASH_DATE']

ts = pd.to_datetime(ts)

ts

0         2022-04-20
1         2021-07-02
2         2022-04-20
3         2022-04-20
4         2022-04-20
             ...
611987    2022-05-05
611988    2022-05-05
611989    2022-05-03
611990    2022-04-29
611991    2022-05-04
Name: CRASH_DATE, Length: 611992, dtype: datetime64[ns]

df_date = pd.DataFrame(ts)

df_date.to_csv("/Users/xinge/Time Series Project/date.csv")
```

- Separate of Date from 2016-2022 on new Traffic Crashes Dataset and aggregate the crash number into weekly sum
- 2016.01.01 - 2022.01.02 for training
- 2022.01.03 - 2022.05.01 for test

```
```{r}
df_wk_seg <- df[df$CRASH_DATE >= "2016-01-01"]
df_wk_seg <- df_wk_seg[df_wk_seg$CRASH_DATE < "2022-01-03"]

df_wk_seg_fc <- df[df$CRASH_DATE >= "2022-01-03"]
df_wk_seg_fc <- df_wk_seg_fc[df_wk_seg_fc$CRASH_DATE < "2022-05-01"]
```

```{r}
df_wk_seg <- df_wk_seg[order(df_wk_seg$CRASH_DATE)]
```

```{r}
df_wk_seg$week <- as.Date(cut(df_wk_seg$CRASH_DATE, "week"))
```

```{r}
df_wk_gp <- df_wk_seg %>% group_by(week) %>% dplyr::summarize(crashes = n())
```
```

DATA PREPROCESSING

STEP 2

- Separate of Date from 2016-2022 on Weather Dataset and aggregate the precipitation and visibility into weekly mean
- 2016.01.01 - 2022.01.02 for training
- 2022.01.03 - 2022.05.01 for test

```r
df2_wk_seg <- df2[df2$date >= "2016-01-01"]
df2_wk_seg <- df2_wk_seg[df2_wk_seg$date < "2022-01-03"]
df2_wk_seg_fc <- df2[df2$date >= "2022-01-03"]
df2_wk_seg_fc <- df2_wk_seg_fc[df2_wk_seg_fc$date < "2022-05-01"]
```

```r
df2_wk_seg <- df2_wk_seg[order(df2_wk_seg$date)]
df2_wk_seg$week <- as.Date(cut(df2_wk_seg$date, "week"))
df2_wk_seg_fc <- df2_wk_seg_fc[order(df2_wk_seg_fc$date)]
df2_wk_seg_fc$week <- as.Date(cut(df2_wk_seg_fc$date, "week"))
```

```r
df2_wk_gp <- df2_wk_seg %>% group_by(week) %>%dplyr:: summarise(across(everything(), mean))
df2_wk_gp_fc <- df2_wk_seg_fc %>% group_by(week) %>%dplyr:: summarise(across(everything(), mean))
```

STEP 3

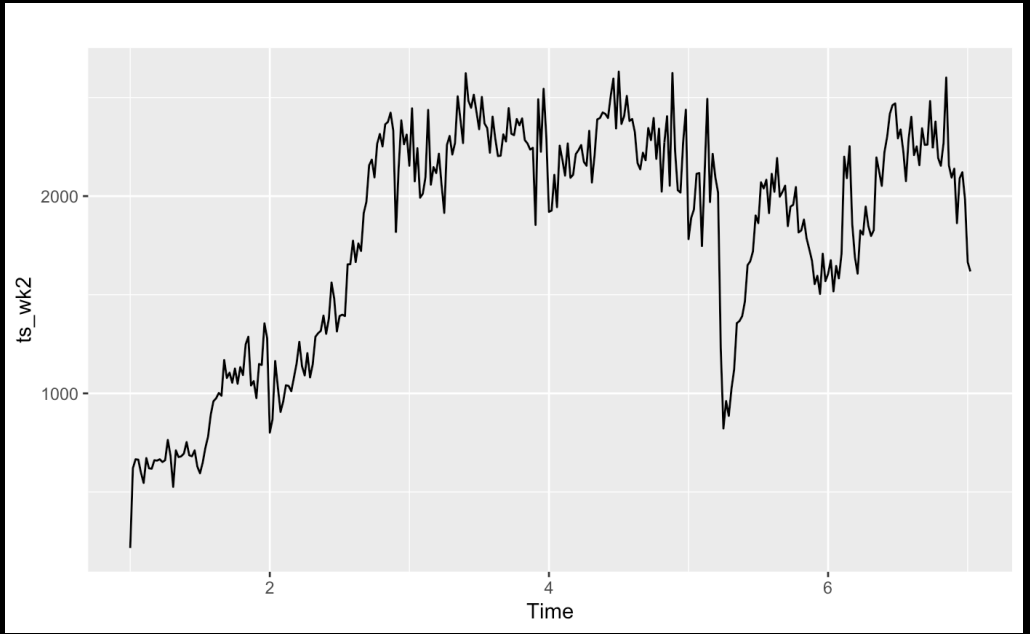
- Combine the previous two processed datasets. The new dataset contains the weekly dates, number of crashes, mean precipitation and mean visibility per week

	week	totalprecipIn	crashes	visibilityMiles
1	2015-12-28	0.000000000	216	5.000000
2	2016-01-04	0.120000000	621	4.714286
3	2016-01-11	0.071428571	666	4.000000
4	2016-01-18	0.015714286	664	5.142857
5	2016-01-25	0.052857143	598	3.857143
6	2016-02-01	0.174285714	546	5.285714
7	2016-02-08	0.067142857	672	3.000000
8	2016-02-15	0.014285714	620	5.285714
9	2016-02-22	0.090000000	618	4.428571
10	2016-02-29	0.132857143	661	4.285714

DATA PREPROCESSING

Initial Observation

Overall, we see an upward trend and some seasonality. There is a significant drop which is most likely caused by external event COVID-19.

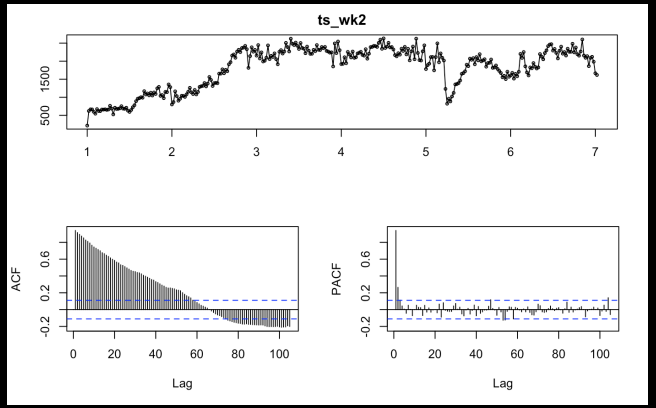


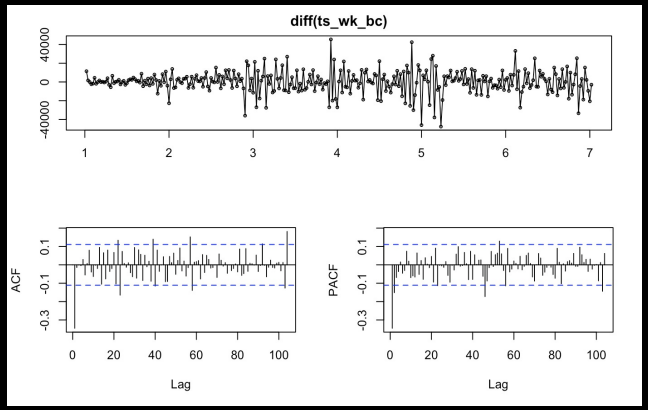
Plot of the car crash training dataset (2016.01.01 - 2022.01.02)

DATA PREPROCESSING

Box-Cox Transformation

- lambda = 1.555455

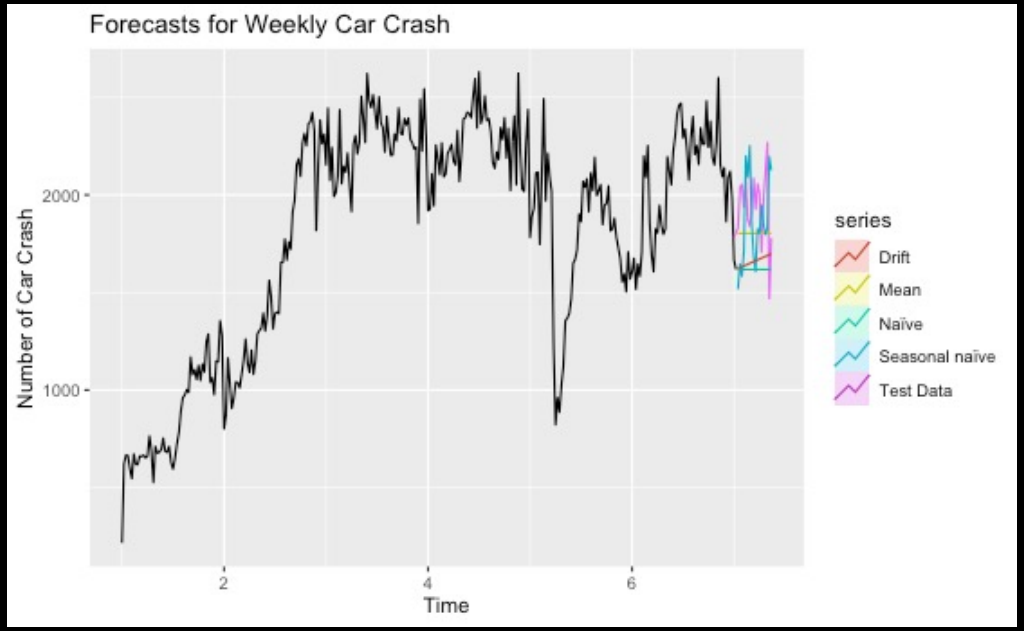




Differencing

- After first order differencing, we get a stationary time series.

BASELINE FORECAST METHODS



Seasonal and non-seasonal forecast methods, including Mean, Drift, Naïve, Seasonal Naïve.

```
err_naive <- sqrt(mean((fc_naive$mean - df_wk_gp_fc$crashes)^2))
err_naive
```

```
[1] 205.5478
```

01

Exponential Smoothing

Estimating time series component directly from the data without predetermined structure.

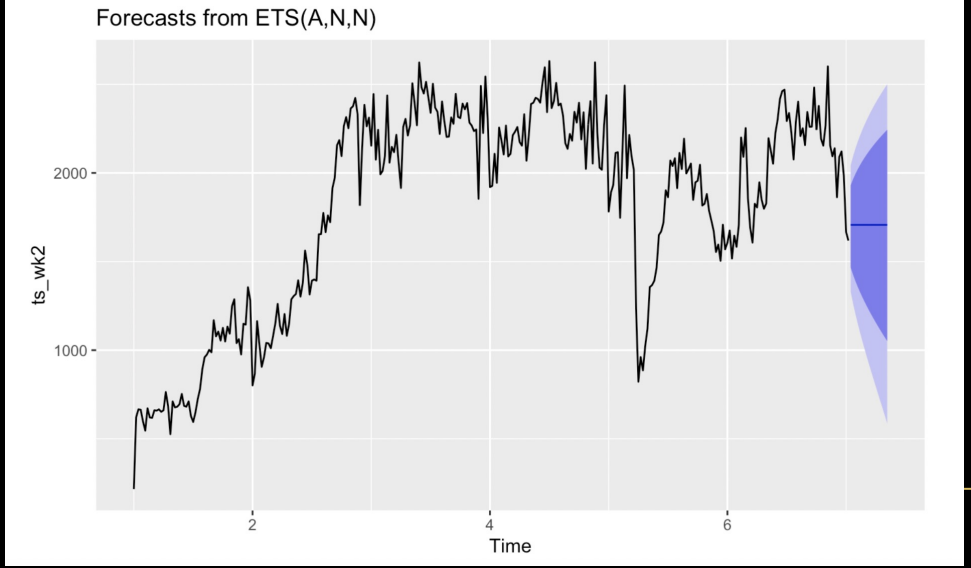
ETS

The outliers and noise will have less impact than the naïve method.

```{r}
fit_ets <- ets(ts_wk2, lambda = "auto")
```

```
Warning in ets(ts_wk2, lambda = "auto") :
  I can't handle data with frequency greater than 24. Seasonality will be ignored. Try stlf() if you need seasonal forecasts.
```

```
autoplot(forecast(fit_ets, h=17))
```



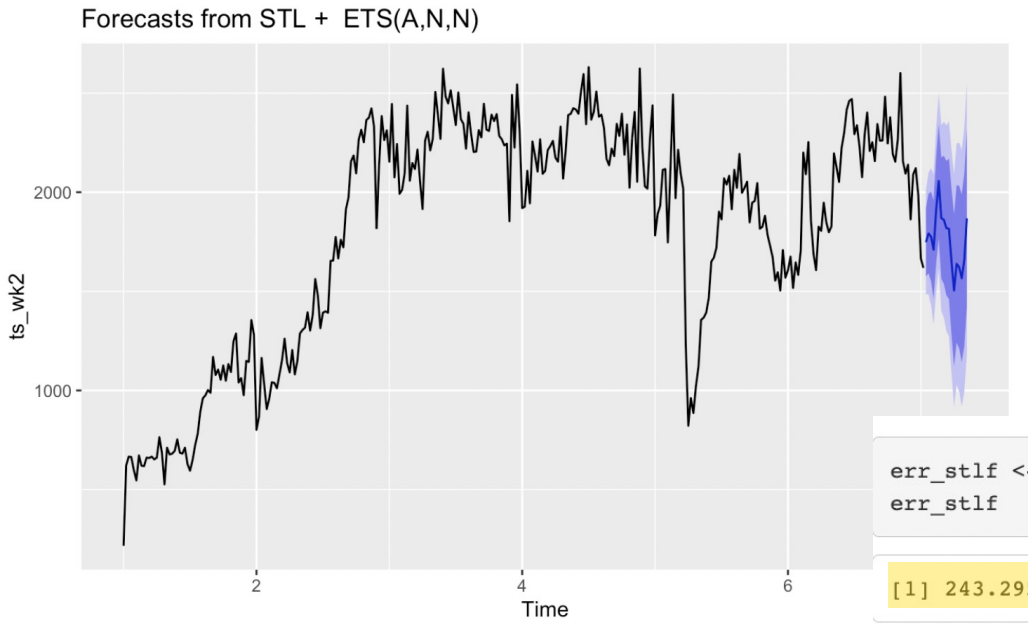
- Advantage: The weight put on each observation decreases **exponentially** over time (the most recent observation has the highest weight).
- This seasonality rules out the use of ETS models which can't handle data with frequency greater than 24, unless used in combination with STL (Seasonal and Trend decomposition using Loess)

STLF + ETS

```
fit_stlf <- stlf(ts_wk2, method='ets', h=17)
```

Hide

```
autoplot(fit_stlf)
```



- STLF can be defined as Seasonal and Trend decomposition using Loess Forecasting model.
- STLF will decompose the time series using STL, and forecast the seasonally adjusted series with ETS, and return the re-seasonalized forecasts.

```
err_stlf <- sqrt(mean((fit_stlf$mean- df_wk_gp_fc$crashes)^2))
err_stlf
```

```
[1] 243.2953
```

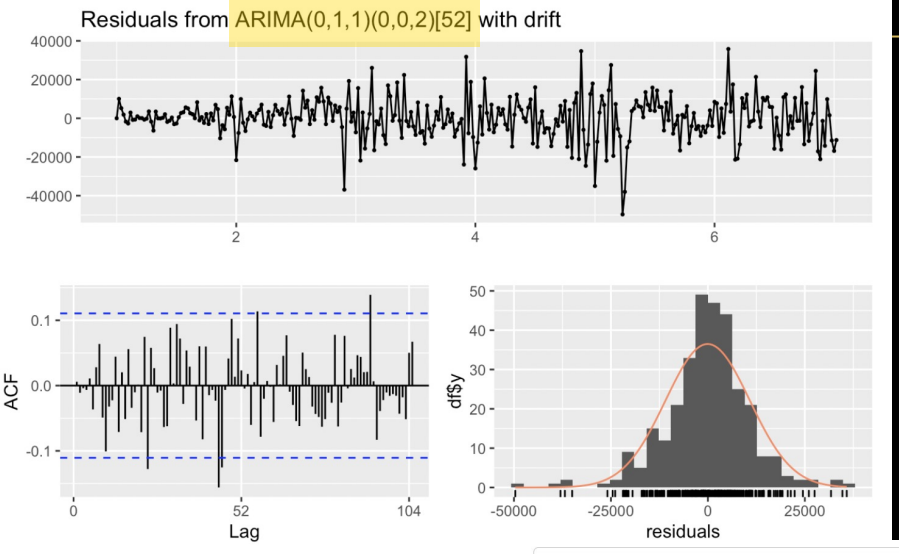
02

ARIMA Models

AutoRegressive Integrated Moving Average models are simple yet effective

AUTO SARIMA

The model can use its past values, past forecasting error and seasonality pattern to predict future values




```
    Ljung-Box test

data:  Residuals from ARIMA(0,1,1)(0,0,2)[52] with drift
Q* = 72.928, df = 59, p-value = 0.105

Model df: 4.   Total lags used: 63
```

```
Series: ts_wk2
ARIMA(0,1,1)(0,0,2)[52] with drift
Box Cox transformation: lambda= 1.555455

Coefficients:
          ma1     sma1     sma2     drift
      -0.4243   0.0773   0.2260  236.0806
s.e.   0.0525   0.0579   0.0683  433.6952

sigma^2 = 119664852:  log likelihood = -3355.98
AIC=6721.95   AICc=6722.15   BIC=6740.68
```

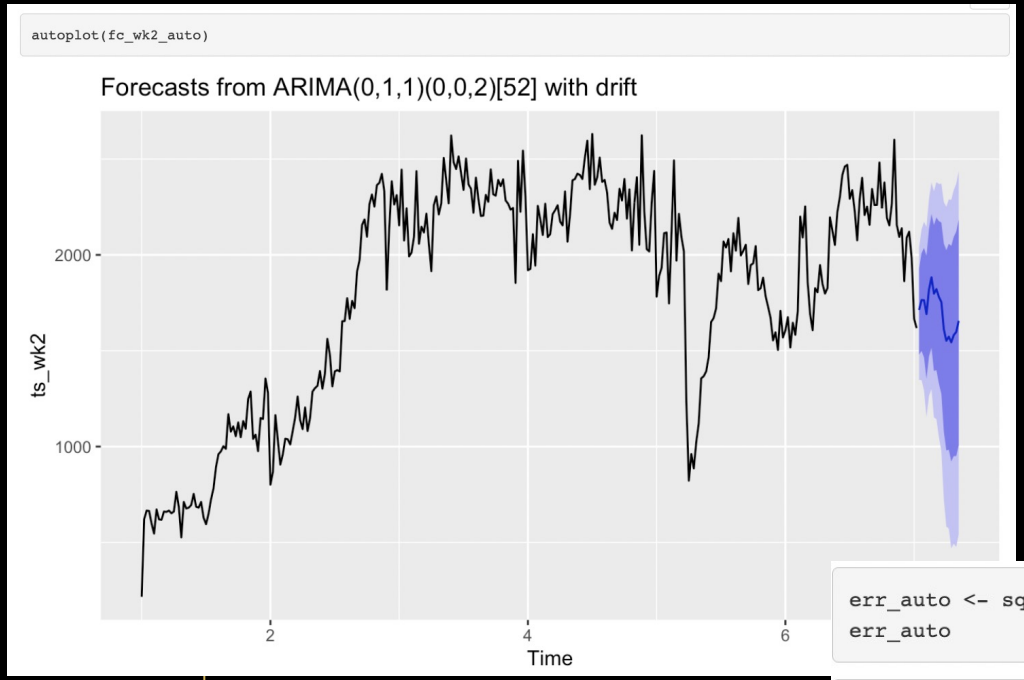
SARIMA(0,1,1)(0,0,2)[52]

Auto SARIMA think a first order of differencing was needed as well.

The ACF shows several significant spikes within the lags. However, the Ljung-Box test gave p value > 0.05 and therefore, the residual looks much like the white noise.

AUTO SARIMA FORECAST

```
autoplot(fc_wk2_auto)
```



Forecast Explanation

- The forecast of the Auto SARIMA model shows some seasonality and seems to be a good base model representation.

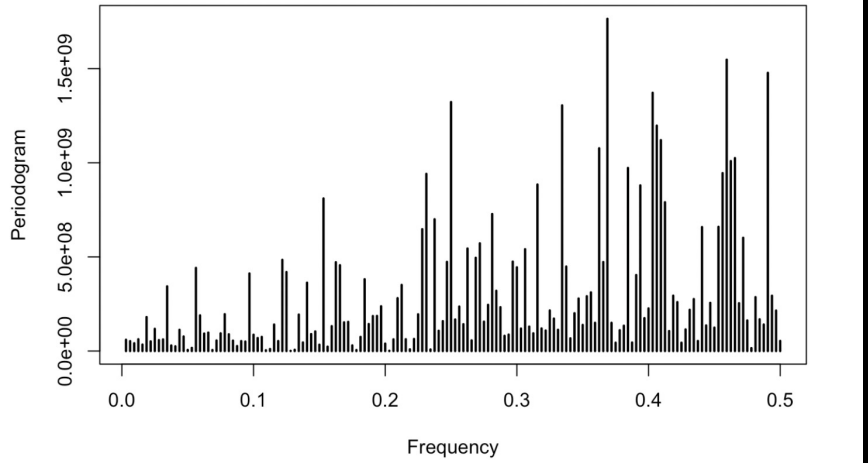
```
err_auto <- sqrt(mean((fc_wk2_auto$mean - df_wk_gp_fc$crashes)^2))
err_auto
```

```
[1] 276.825
```

AUTO SARIMA W/ FOURIER TRANSFORMATION

Since our series has dynamic seasonality, the transformation will change the data from a time to frequency domain representation.

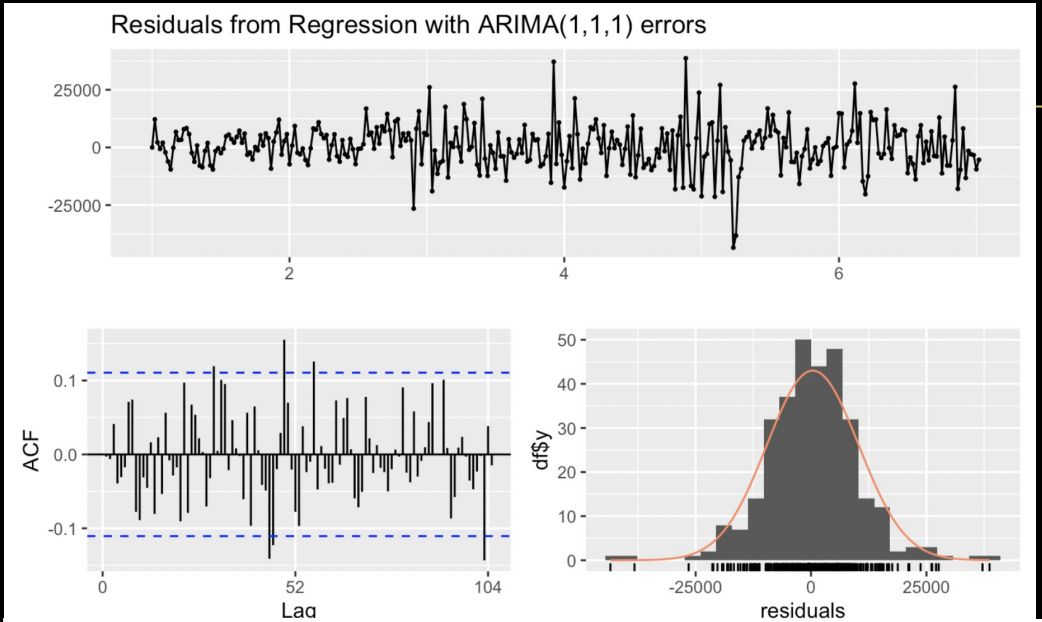
```
temp <- periodogram(ts_wk_bc_diff)
```



The Spectrum Analysis shows that there is too much noise within our data, looping through model to find the best K value.

```{r}
bestfit <- list(aicc=Inf)
k = 0
for(i in 1:25)
{
  fit_fourier_best <-auto.arima(ts_wk2, xreg=fourier(ts_wk2, i), seasonal = FALSE, lambda = 1.368373)

  if(fit_fourier_best$aicc < bestfit$aicc){
    bestfit <- fit_fourier_best
    k = i
  }
}
```

AUTO SARIMA W/ FOURIER TRANSFORMATION



Ljung-Box test

data: Residuals from Regression with ARIMA(1,1,1) errors
Q* = 91.313, df = 35, p-value = 6.426e-07

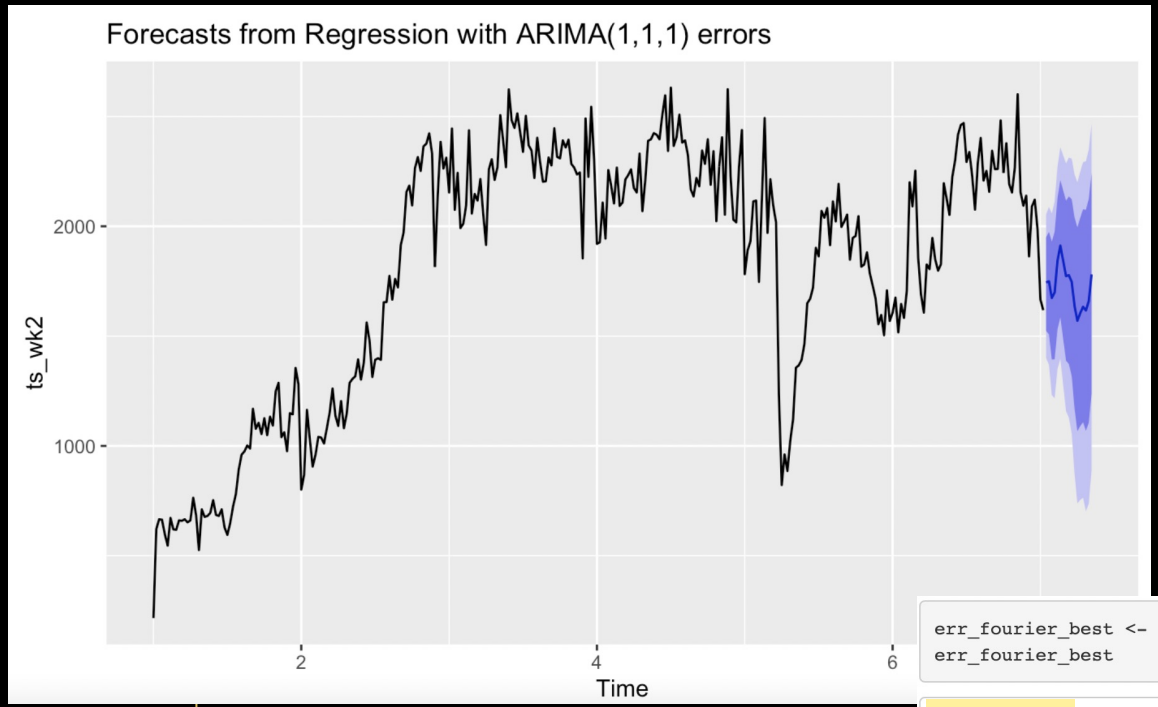
Model df: 28. Total lags used: 63

```
Series: ts_wk2
Regression with ARIMA(1,1,1) errors
Box Cox transformation: lambda= 1.555455

Coefficients:
          ar1      ma1      S1-52     C1-52     S2-52     C2-52     S3-52     C3-52     S4-52     C4-52
      -0.1680  -0.3727  -9948.558  -1981.271  -133.413   716.6996   934.6597  -4236.877  -2421.367  -176.1832
s.e.   0.1075   0.1006   3586.074   3580.949  1836.836  1834.8779  1272.3847   1271.672   1003.376  1003.2829
          S5-52     C5-52     S6-52     C6-52     S7-52     C7-52     S8-52     C8-52     S9-52     C9-52
      -1037.0726  2042.7483  -119.8836  1302.395  642.4525  1142.2846  -1234.1802  1743.877  -1063.8719  25.3034
s.e.    851.8568   852.0661   758.5380   758.831  698.0701   698.2922   657.8543   657.909   630.9506  630.8152
         S10-52    C10-52    S11-52    C11-52    S12-52    C12-52    S13-52    C13-52
      -1734.638  -109.8238  -1652.6035  -35.5724  -1672.6484  -917.4102  -1167.566  1741.9501
s.e.    613.237   612.9466   602.1258  601.7811   595.9155  595.6520   593.417   593.4143

sigma^2 = 109337835:  log likelihood = -3326.42
AIC=6710.84   AICc=6716.99   BIC=6819.48
```

ARIMA (1,1,1), K = 13

The Ljung Box Test shows that the model is not White Noise $P < 0.05$ with the ACF also showing multiple spikes.

AUTO SARIMA FORECAST W/ FOURIER TRANSFORMATION



Forecast Explanation

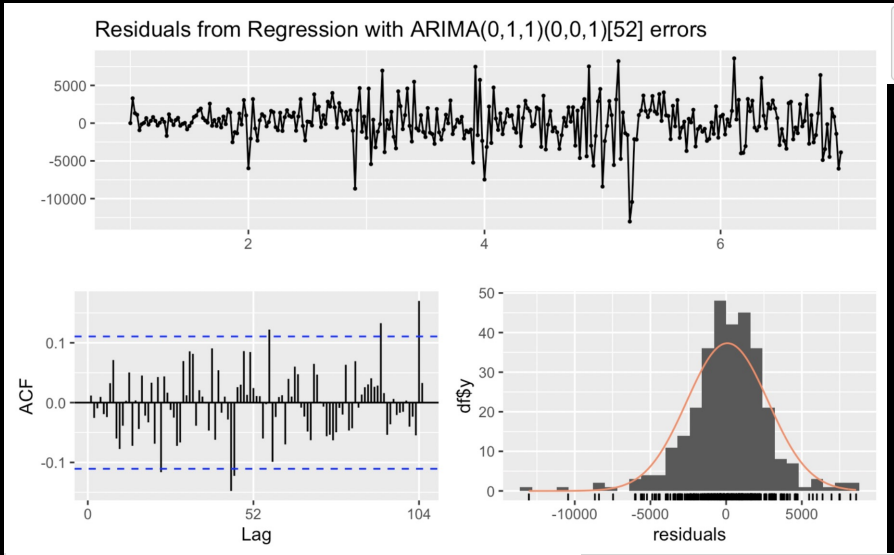
The forecast of the Fourier Arima Model seems to show a better forecast representation of seasonality. It has a lower error value and lower AICc value than the base model.

```
err_fourier_best <- sqrt(mean((fc_wk2_fourier_best$mean - df_wk_gp_fc$crashes)^2))
err_fourier_best
```

```
[1] 250.393
```

REGRESSION WITH ARIMA ERROR W/ PRECIPITATION & VISIBILITY

Able to now include other information that might be relevant in predicting the crashes, on time series modeling, to allow for more accurate forecasting.




```
xreg = cbind(Percipitation = ts_wk3, Visibility = ts_wk4)
fit_wk_auto_both <- auto.arima(ts_wk2, xreg = xreg, lambda = 1.368373, seasonal=TRUE)
```

```
Ljung-Box test

data:  Residuals from Regression with ARIMA(0,1,1)(0,0,1)[52] errors
Q* = 70.201, df = 59, p-value = 0.1508

Model df: 4.   Total lags used: 63
```

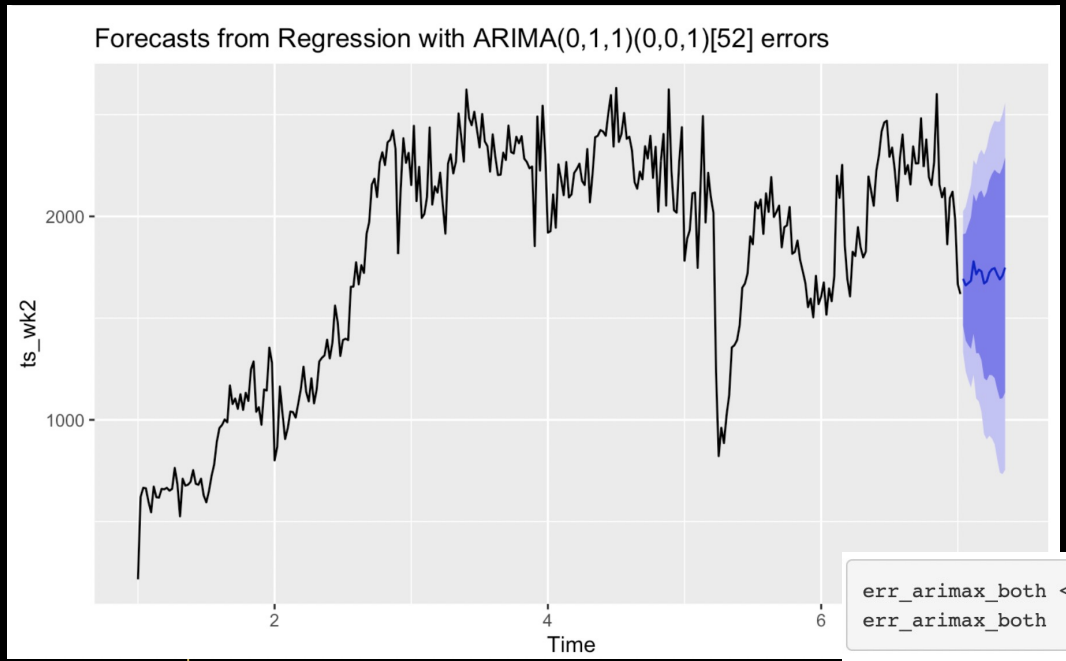
```
Series: ts_wk2
Regression with ARIMA(0,1,1)(0,0,1)[52] errors
Box Cox transformation: lambda= 1.368373

Coefficients:
          ma1     sma1  Percipitation  Visibility
      -0.4069   0.1007       -134.4997   -534.2200
s.e.   0.0540   0.0512        774.4471    287.7701

sigma^2 = 7326247:  log likelihood = -2916.26
AIC=5842.52   AICc=5842.72   BIC=5861.25
```

ARIMA(0,1,1)(0,0,1)[52]

Different than the other ARIMA model this model includes the two other XREG values of Precipitation & Visibility.

The ACF shows several significant spikes within the lags. However, the Ljung-Box test gave p value > 0.05 and therefore, the residual looks much like the white noise.

REGRESSION WITH ARIMA ERROR FORECAST W/ PRECIPITATION & VISIBILITY



Forecast Explanation

The forecast of the regression with ARIMA error model doesn't show much trend but some seasonality. It has one of the lowest AICc & RMSE so far compared to the other models.

```
err_arimax_both <- sqrt(mean((fc_wk_auto_both$mean - df_wk_gp_fc$crashes)^2))
err_arimax_both
```

```
[1] 236.3234
```

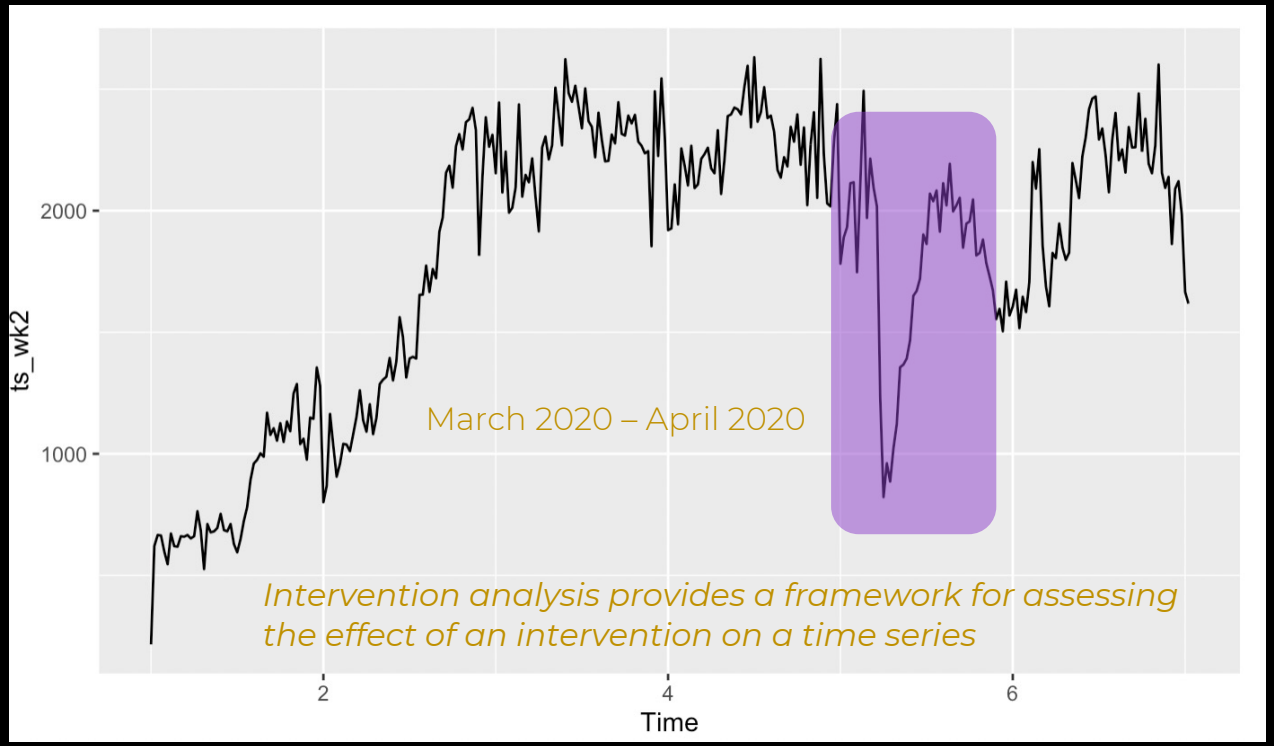
03

Intervention Analysis

Assessing the effect of an intervention or event that significantly alters the mean function or trend of a time series.

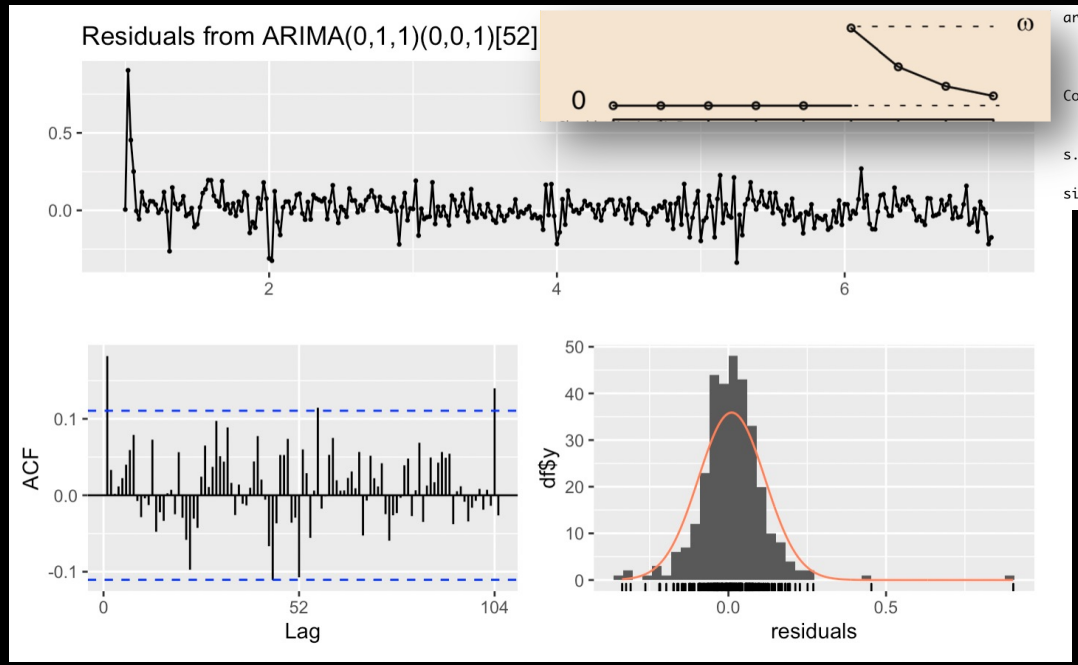
Drastic Drop caused by External Event

There is a drastic shift in the series caused by external effect that slowly decays



Intervention Analysis - Pulse Function

An intervention model with a pulse function is typically employed if the effects are expected to be only temporary, and decay over time




```
arimax(x = log(ts_wk2), order = c(0, 1, 1), seasonal = list(order = c(0, 0,
    1), period = 52), xreg = xreg, method = "ML", xtransf = covid, transfer = list(c(1,
    0)))

Coefficients:
          ma1     sma1  Percipitation  Visibility  T1-AR1   T1-MA0
      -0.5170  0.2495         0.0061     -0.0245  0.9397  -0.7534
s.e.   0.0638  0.0667         0.0308      0.0115  0.0293   0.0933

sigma^2 estimated as 0.01126:  log likelihood = 256.15,  aic = -500.3
```

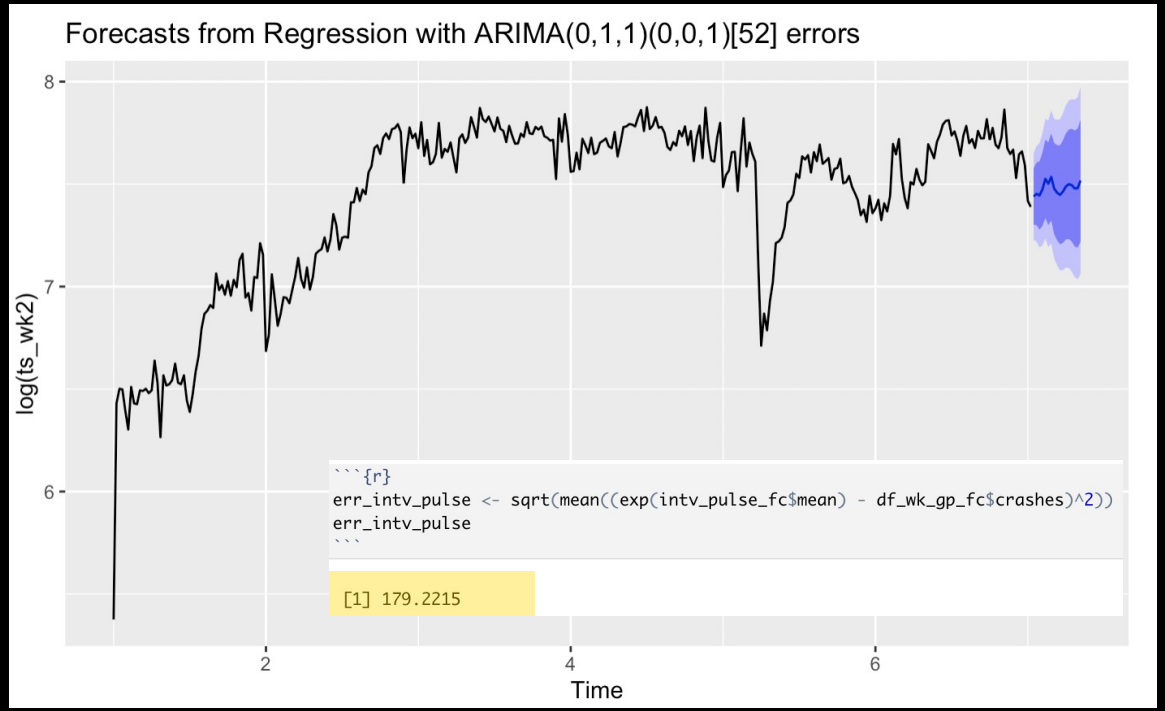
Parameters:
xtransf is the intervention series, flagging Pt = 1 to indicate March 16, 2020
transfer = list(c(1,0)) indicates the functional form of the transfer function.

Pulse function:
w = -0.7534; δ = 0.9397

$$m_t = \frac{\omega B}{1 - \delta B} P_t^{(T)}$$

Results:
The model has AIC value = -500.
The residual check gave us a p-value = 0.1834 which indicated that the residual is white noise.

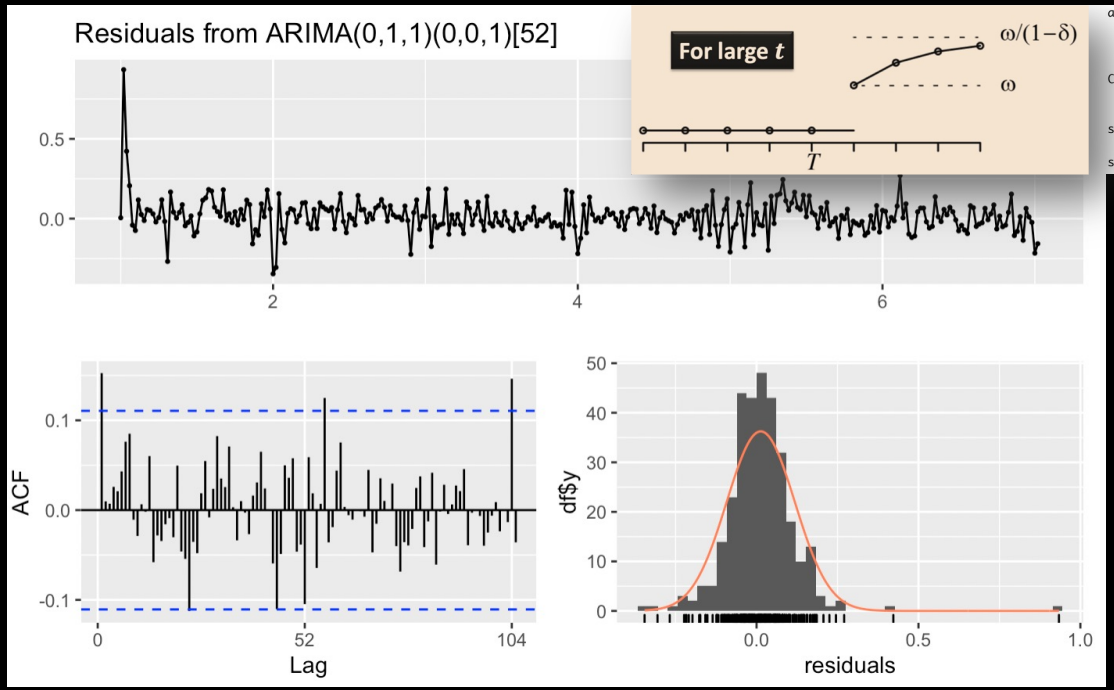
Intervention Analysis – Pulse Function Forecasting



The forecast of intervention with pulse function gives us the lowest forecasting RMSE error.

Intervention Analysis - Step Function

An intervention model with a step function is typically employed if the effects are expected to be immediate and permanent shift in the mean




```
arimax(x = log(ts_wk2), order = c(0, 1, 1), seasonal = list(order = c(0, 0,
    1), period = 52), xreg = xreg, method = "ML", xtransf = covid_step, transfer = list(c(1,
    0)))

Coefficients:
          ma1     sma1  Percipitation  Visibility  T1-AR1   T1-MA0
      -0.4415  0.2360         0.0046     -0.0236  0.2591  -0.6278
s.e.   0.0636  0.0655         0.0303      0.0113  0.0979   0.0914

sigma^2 estimated as 0.01126:  log likelihood = 256.45,  aic = -500.9
```

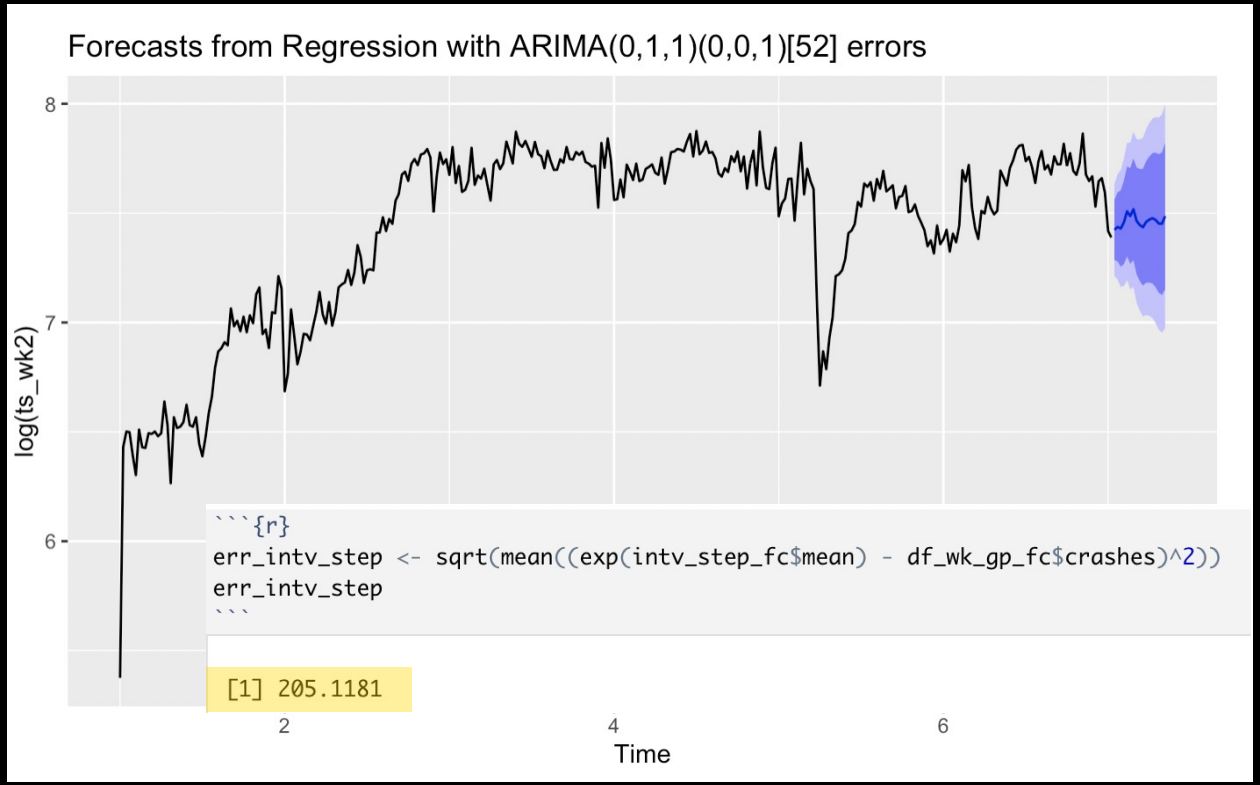
Parameters:
xtransf is the intervention series, flagging Pt = 1 for all the date starting from March 16, 2020
transfer = list(c(1,0)) indicates the functional form of the transfer function.

Step Funtion:
w = -0.6278; δ = 0.2591

$$\frac{\omega B}{1-\delta B} S_t^{(T)}$$

Results:
The model has AIC value = -500.
The residual check gave us a p-value = 0.3325 which indicated that the residual is white noise.

Intervention Analysis – Step Function Forecasting



The forecast of intervention model with pulse function gives us the second lowest forecasting RMSE error.

04

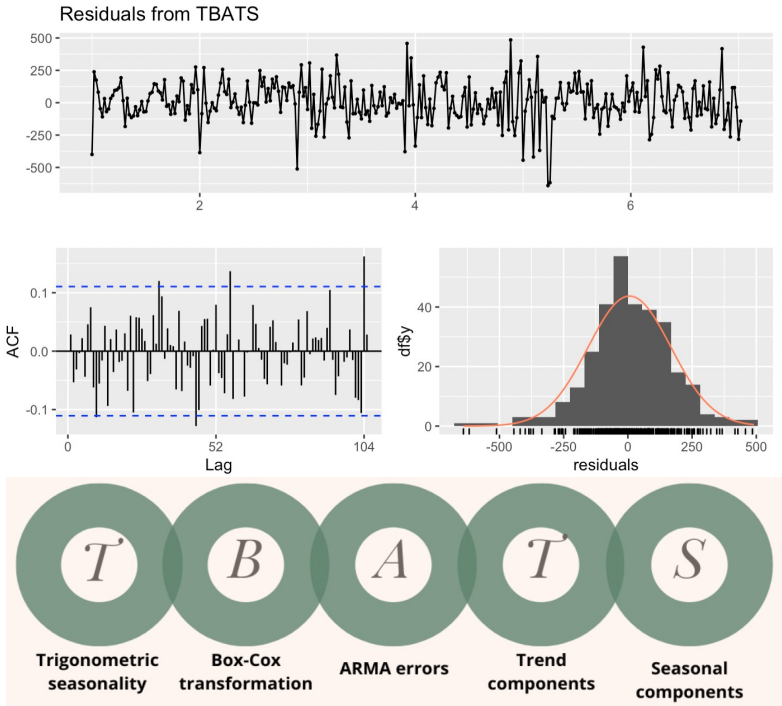
TBATS Model

Forecast time series with complex seasonal patterns using exponential smoothing

TBATS

The variance has changed over time, the dynamic seasonality has also changed over time, and there is a strong trend.

```
TBATS(1, {0,0}, -, {<52,5>})

Call: tbats(y = ts_wk2)

Parameters
  Alpha: 0.5480914
  Gamma-1 Values: -0.008646206
  Gamma-2 Values: -0.007217537

Seed States:
              [,1]
 [1,]  727.285960
 [2,]  -47.979778
 [3,]    3.329315
 [4,]  -56.997318
 [5,]  -27.286547
 [6,]   17.268421
 [7,] -149.014345
 [8,]    4.959360
 [9,]   38.502313
[10,]  -32.771923
[11,]  -29.996358

Sigma: 161.2386
AIC: 5025.361
```

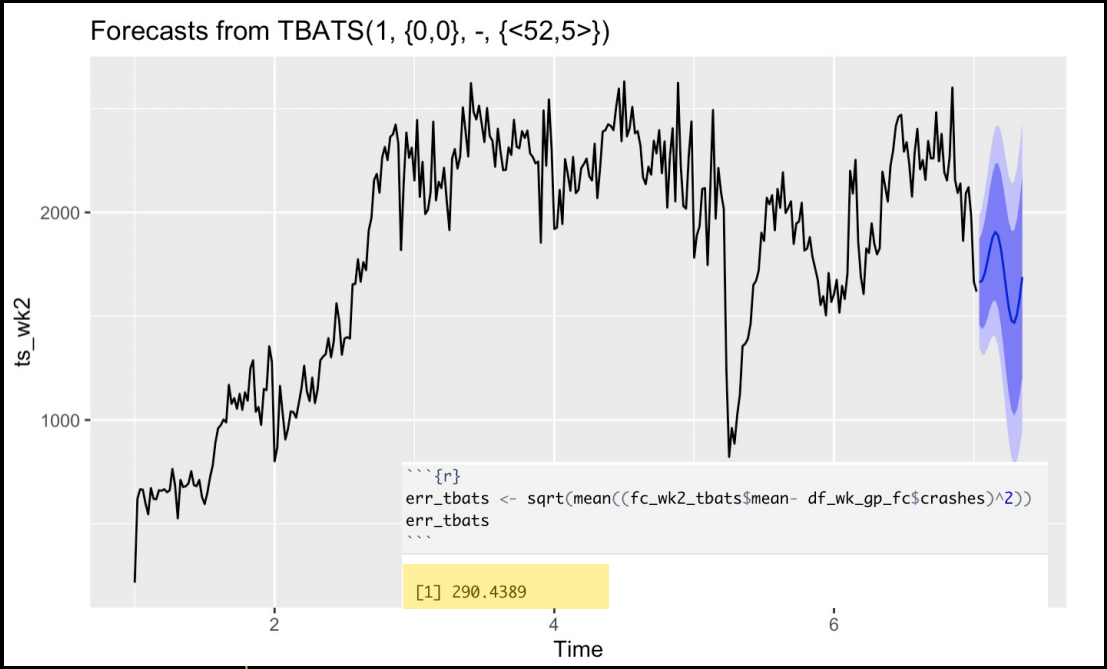


TBATS (1, {0,0}, -, {<52,5>})

1: No Box-Cox Transformation
{0,0}: ARMA Error
-: Damping parameter
{<52,5>}: seasonal period, fourier terms

The residual analysis gave us p-value = 0.0084 which indicates the residual is not white noise

TBATS FORECAST



The TBATS gave us RMSE 290.

Prediction intervals often too wide;
Very slow on long series;
Covariates: Neither ETS nor TBATS models allow for covariates. A state space model of the same form as TBATS but with multiple sources of error and covariates could be used.

Summary of Model Results

Model	Ljung Box Test P-Value	AIC	AICc	BIC	RMSE	PARAMETERS
Naïve						
Seasonal Naïve	N/A	N/A	N/A	N/A	205.5478	N/A
Exponential Smoothing						
ETS	N/A	N/A	N/A	N/A	N/A	N/A
ETS + STLF	N/A	N/A	N/A	N/A	243.2953	N/A
Arima						
SARIMA	0.105	6721.95	6722.15	6740.68	276.825	(0,1,1)(0,0,2)[52]
SARIMA w/ Fourier	6.43E-07	6710.84	6716.99	6819.48	250.393	(1,1,1) Errors
Regression w/ ARIMA Error (P & V)	0.1508	5842.52	5842.72	5861.25	236.3234	(0,1,1)(0,0,1)[52]
Intervention Analysis						
Pulse Function	0.1928	-498.34	-498.21	-483.35	179.2215	(0,1,1)(0,0,1)[52]
Step Function	0.3325	-499.26	-499.13	-484.27	205.1181	(0,1,1)(0,0,1)[52]
TBATS						
TBATS	N/A	5025.36	N/A	N/A	290.4389	(1, {0,0}, -, {<52,5>})

Principle of parsimony
All other things being equal. simple models are preferred to complex ones.

Conclusions

- Based on daily seasonal complexity and frequency, selected Weekly Data analysis as best approach for dataset.
- Overall, Intervention Analysis with pulse function yielded some of the lowest metrics and accurately captured seasonality of the data in the forecast.
- From a principle of parsimony or simplicity perspective, Seasonal Naïve model is most efficient and close to Intervention Analysis for metrics.
- Regression with ARIMA Error also performs well with potential for improvement.

Future Work

- Explore additional regressor(s) to determine if Regression with ARIMA Error can be improved further – possibly improving accuracy beyond Intervention Analysis.
- Consider Ensemble Approach or grouping together of different methods for improved forecasting.
- Evaluate potential for model integration with other transportation data, e.g., traffic-flow modeling, to help improve forecast accuracy or influence traffic controls.
- Determine how we can incorporate cost into our models for potential use by insurance companies or for roadway design and traffic planning purposes.

Thank You

QUESTIONS?