## horizontal line

Time Series Analysis: California Covid-19 Cases

Nicholas Benelli

Department of Mathematical Sciences, Stevens Institute of Technology, Hoboken, NJ

Project Supervisor: Dr. Hadi Safari Katesari

August 22, 2022

# Abstract

The CHHS reports COVID-19 cases in the state of California on a daily basis. This project is a time series analysis of the statewide COVID-19 cases. The goal is to provide a realistic forecast and predict the reported COVID-19 cases for the next 13 weeks. The daily data is grouped to make the data weekly. The data is checked for stationarity. Seasonal models are fitted to the data and tested. Some of the models require a seasonal difference, while others do not. Trends, volatility, and seasonality will be analyzed. Residual analysis is performed to determine if the SARIMA model will fit the data.

# Introduction and Motivation

The COVID-19 pandemic has been a generational-changing event. The novelty and contagiousness of the virus have put stress on family life, the healthcare system, and the economy but in particular the response to such a devastating pandemic.

As a result, the qualities required for our information and reporting systems had to be completely revamped. This in turn has led to new challenges when it comes to modeling how the pandemic works. Predictive modeling can be vital and even life-saving because accurate predictions can allow for policy to be made quickly and for resources to be allocated to healthcare facilities if a large wave of cases is near.

The burden of reporting has fallen to the state level. Each state is in charge of distributing tests and reporting back positive cases and even deaths from COVID-19. The California Health and Human Services department has been tasked with collecting and reporting COVID-19 tests and positive cases within California.

The aim of this paper and the time series model is to take the data reported by the CHHS and build a predictive model in order to forecast the next 13 weeks of COVID-19 cases.

# Data Description

**Data Range:** January 27, 2011 - August 8, 2022 (Present)

**Link to Database:** https://data.chhs.ca.gov/dataset/covid-19-time-series-metrics-by-county-and-state/resource/046cdd2b-31e5-4d34-9ed3-b48cdbc4be7a

## Data Fields

The data fields description are pulled directly from the CHHS website’s data dictionary.

**Date:** Reporting time period (Values: Date in YYYY-MM-DD format)

**Area:** County of Residence of Case, Test, or Death of COVID-19 (Value: Text)

**Area\_Type:** Geographic type of the Area field. (Values: “State” if the Area value is “California” “County” for all other values)

**Population:** California Department of Finance (DOF) population estimates 2020 (Values: Positive Numeric)

**Cases:** Total number of laboratory-confirmed COVID-19 cases with episode date on the provided date. “Episode date” is defined as the earliest of the following dates (if the dates exist): date received, date of diagnosis, date of symptom onset, specimen collection date, or date of death. (Values: Positive Numeric)

**Cumulative\_Cases:**  Cumulative total of Cases field. (Values: Positive Numeric)

**Deaths:** Total number of confirmed COVID-19 associated deaths with date of death on the provided date. Local health departments use multiple sources to confirm that a death is COVID-associated, including speaking with physicians, reviewing medical records, and consulting with medical examiners. COVID-associated deaths are also counted in the “Cases” field. (Values: Positive Numeric)

**Cumulative\_Deaths:** Cumulative total of Deaths field. (Values: Positive Numeric)

**Total\_Test:** Total number of COVID-19 molecular tests (polymerase chain reaction [PCR] tests only) performed by laboratories with specimen collection date (estimated testing date) on the provided date. (Values: Positive Numeric)

**Cumulative\_Total\_Tests:** Cumulative total of Total Tests field.

(Values: Positive Numeric)

**Positive\_Tests:** Total number of positive COVID-19 molecular tests (polymerase chain reaction [PCR] tests only) with specimen collection date (estimated testing date) on the provided date. (Values: Positive Numeric)

**Cumulative\_Positive\_Tests:** Cumulative total of Positive Tests field. (Values: Positive Numeric)

**Reported\_Cases:**

Total number of laboratory-confirmed COVID-19 cases reported to the California Department of Public Health on the provided date. (Values: Positive or Negative Numeric)

**Cumulative\_Reported\_Cases:** Cumulative total of Reported Cases field. (Values: Positive Numeric)

**Reported\_Deaths:** Total number of confirmed COVID-19-associated deaths reported to the California Department of Public Health on the provided date. Local health departments use multiple sources to confirm that a death is COVID-associated, including speaking with physicians, reviewing medical records, and consulting with medical examiners. COVID-associated deaths are also counted in the “Cases” field. (Values: Positive or Negative Numeric)

**Cumulative\_Reported\_Deaths:** Cumulative total of Reported Deaths field.

(Values: Positive Numeric)

**Reported\_Tests:** Total number of COVID-19 molecular tests reported to the California Department of Public Health on the provided date. (Values: Positive or Negative Numeric)

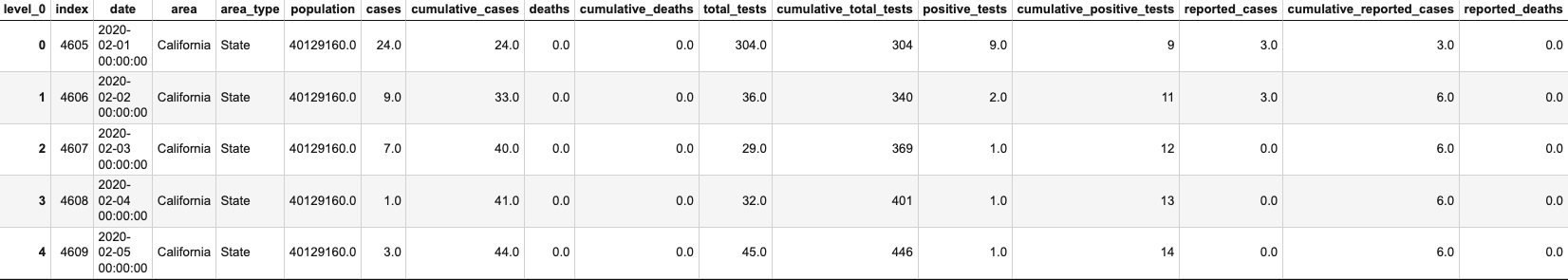
## Initial Look at Data

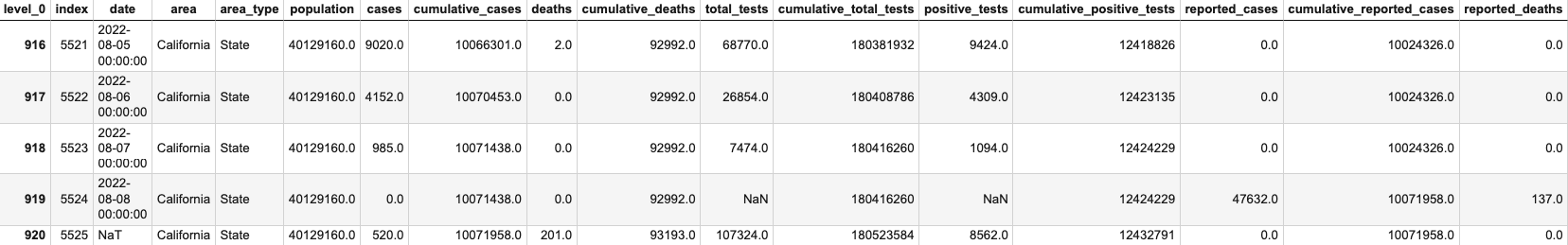
### Table View of Raw Data



The dataset starts on February 1, 2020, and continues to the present (August 8, 2022). The data is broken up by county and state. The primary keys of the table are date and county. Each county has one row with no date to make up the difference between total cases and the sum of daily reported cases. For this paper, we are only concerned with analyzing the state of California and not a specific county. The dataset is subsetted on “state” to only show statewide data.

### Table View of State-Wide Data

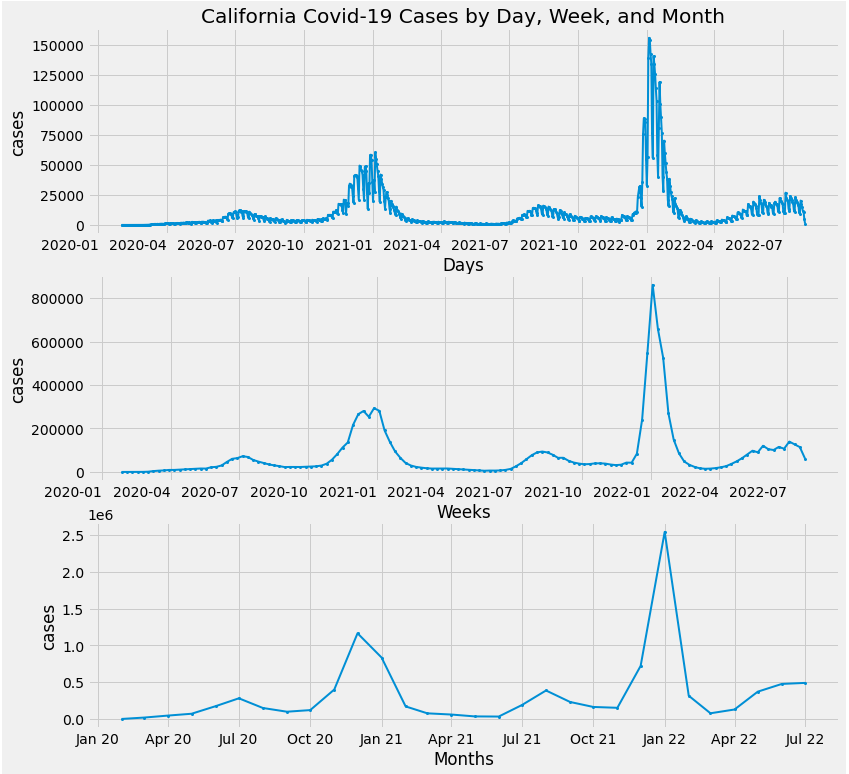




|  |
| --- |
| display(df\_california.describe()) |

### 

### Plot View of Daily, Weekly, and Monthly Cases



Seeing as the trends are generally the same, the rest of the paper will use the weekly cases as the default data to build off. Weekly cases will remove any daily reporting variances and trends during the week.

|  |
| --- |
| # Weekly Cases data\_weekly = data\_daily.copy().reset\_index() data\_weekly['date'] = data\_weekly['date'] - pd.to\_timedelta(7, unit='d') data\_weekly = data\_weekly.groupby([ pd.Grouper(key='date', freq='W-MON')])['cases'].sum().reset\_index().sort\_values('date') data\_weekly = data\_weekly.set\_index(['date']) display(data\_weekly.tail(5)) |

## 

## Pre-Processing

Within the state-wide dataset there are two rows that contain blank data or data that is not relevant to the overall dataset, so the first step is to remove those rows. The last date does not have any reporting and is a placeholder value for the next report. The “NaN” row is used to match the total daily data with the official total cases. These are not relevant to the time series. The remaining dataset is 132 samples.

|  |
| --- |
| # remove last 2 rows - not reported df\_california.drop(df\_california.tail(2).index, inplace = **True**) |

The data is then split into training, validation, and testing segments. In time series analysis the data cannot be random. The final 13 data points will be the testing group. The 15 data points prior to that will be the validation group. The remaining 104 data points will be the training sample. This is roughly an 80/20 percent split for training and testing.

|  |
| --- |
| # Test Train split val\_size = 15 test\_size = 13 data\_train = data.iloc[:-(val\_size + test\_size), :] data\_val = data.iloc[-(val\_size + test\_size):-test\_size] data\_test = data.iloc[-test\_size:, :] |

After the data is cleaned up, we apply various transformations and used the Dickey-Fuller test to check if the data is stationary with the transformation.

|  |
| --- |
| **def** **dickey\_fuller\_test\_results**(p\_value):  **if** p\_value < 0.05:  print(f'The null hypothesis can be rejected. The time series is stationary. ({round(p\_value, 4)} < 0.05)')  **else**:  print(f'The null hypothesis cannot be rejected. The time series is not stationary. ({round(p\_value, 4)} > 0.05)')  **return** **None** |

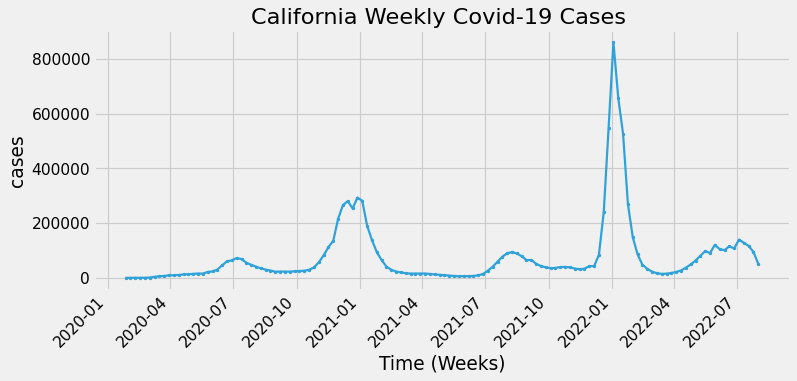
### Seasonal Decomposition:

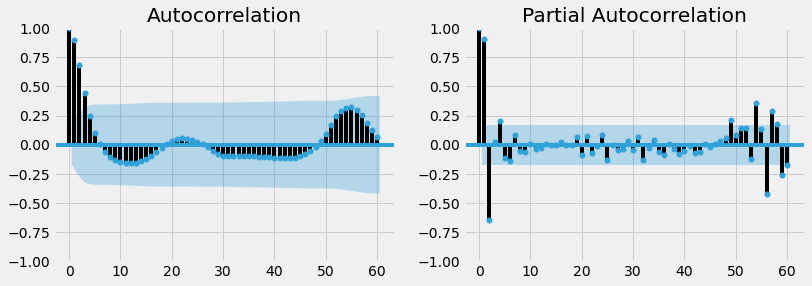
A seasonal decomposition is completed to see if there are any trends or seasonality in the data. There is no real trend in the data, but there may be some seasonality in the data. The season would be 52 weeks because the data is weekly.

|  |
| --- |
| result = seasonal\_decompose(data\_weekly, model='multiplicative') fig = result.plot() |

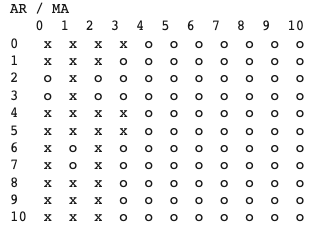
### 

### No Transformation:





EACF - No Transformation



|  |
| --- |
| result = adfuller(data\_weekly) print(f'p-value: {result[1]}') dickey\_fuller\_test\_results(result[1]) |

p-value: 1.0952288813797513e-05

The null hypothesis can be rejected. The time series is stationary. (0.0 < 0.05)

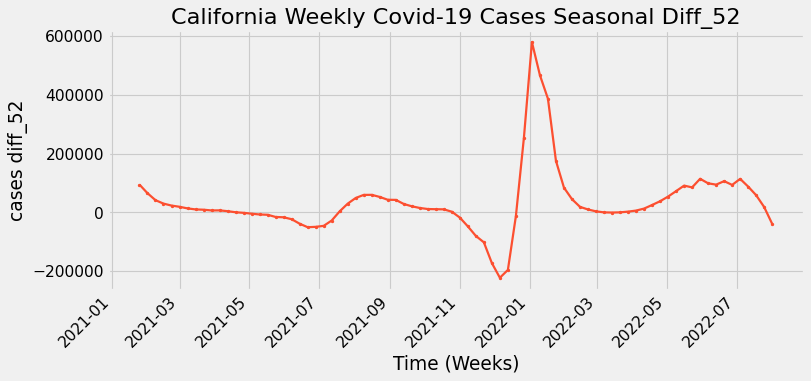
The time series is stationary without any transformation. It is not necessary to take a transformation or difference the data. The autocorrelation plot shows an exponential decay with some noise one season out. The noise is not significant, however. The ACF plot would suggest there is no moving average (MA) component in the model. The partial autocorrelation plot shows the first two lags are strongly correlated and the next two lags are moderately correlated. There are lags one season out that are significant. This would suggest a seasonal model would be required to model the data. The extended autocorrelation suggests an AR(2), AR(3), or ARMA(3,3) model. The ARMA(3,3) can be disregarded because it does not agree with the ACF plot.

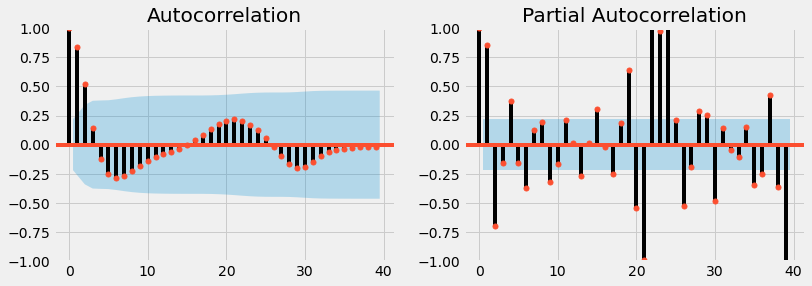
### 

### Seasonal Transformation:

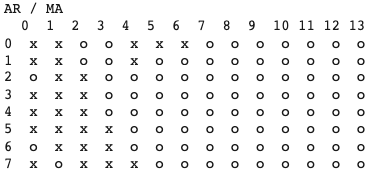
The data appears to have some seasonality from the seasonal decomposition above. Intuition would say that datasets tracking viruses will have a seasonality component to them. For example, cases of the common cold generally increase in the winter and decrease in the summer. A seasonal difference can remove some of the volatility and harsh spikes in the data.

|  |
| --- |
| data\_seasondiff = data.loc[:, ['cases']].diff(52).iloc[52:, :] |





EACF - Seasonal Transformation



|  |
| --- |
| result = adfuller(data\_seasondiff) print(f'p-value: {result[1]}') dickey\_fuller\_test\_results(result[1]) |

Dickey-Fuller test:

(-4.015081448346416, 0.0013335669560792648, 5, 74, {'1%': -3.5219803175527606, '5%': -2.9014701097664504, '10%': -2.58807215485756}, 1636.9166974696273)

p-value: 0.0013335669560792648

The null hypothesis can be rejected. The time series is stationary. (0.0013 < 0.05)

The data is stationary without a transformation and with a seasonal transformation according to the Augmented Dickey-Fuller test. The test’s p-value fall below an alpha of 0.05 so the null hypothesis can be rejected. It is worth noting the 2021 spike in the time series does not fully match up with the 2022 spike in the time series.

The ACF plot shows an exponential decay and then oscillation. The PACF plot for the differenced data shows a strong correlation at the first two lags and a moderate correlation at lags three and four. There appears to be noise around the twentieth lag. The EACF plot suggests a SARMA(2,0,1)X(0,1,0).

# Box Jenkins Analysis

Multiple accuracy metrics will be used to compare the prediction data to the validation data. The mean percentage error (mpe) and the root mean square error (rmse) will be used as the main metrics.

|  |
| --- |
| **def** **find\_prediction\_acc**(y\_pred, y\_true, print\_result=True):  mape = np.mean(np.abs(y\_pred - y\_true) / np.abs(y\_true)) # Mean absolute percentage error  mae = np.mean(np.abs(y\_pred- y\_true)) # mean absolute error  mpe = np.mean((y\_pred - y\_true) / y\_true) # Mean percentage error  rmse = np.mean((y\_pred - y\_true) \*\*2)\*\*(1/2) # RMSE  corr = np.corrcoef(y\_pred, y\_true)[0, 1] # Correlation Coefficient   mins = np.amin(np.hstack([y\_pred.reshape(-1,1), y\_true.reshape(-1,1)]), axis=1)  maxs = np.amax(np.hstack([y\_pred.reshape(-1,1), y\_true.reshape(-1,1)]), axis=1)  minmax = 1 - np.mean(mins.reshape(-1,1) / maxs.reshape(-1,1)) # minmax |

## Model Selection

Models will take the form of SARIMA(p, d, q)x(P, D, Q)[s] where p is the AR component, d is the integration, q is the Moving Average (MA). The capital variables are the seasonal AR, Integration, and MA components. The variable s is the season which will be 52. A loop is used via the python package “pmdarima” to minimize the AIC of the hyperparameters on the training data. Separate loops are run for no seasonal difference and one seasonal difference to find the best two models for each scenario. Starting parameters are given to start the minimization. These parameters are selected in accordance with the ACF, PACF, and EACF. Max parameters are given to not exceed when minimizing the AIC.

|  |
| --- |
| **import** pmdarima **as** pm model\_no\_seasondiff= pm.auto\_arima(  data\_train,  start\_p=3,d=0,start\_q=0,  max\_p=5,max\_d=1,max\_q=3,  start\_P=1, D=0, start\_Q=0,  max\_P=5,max\_D=1, max\_Q=3, m=season,  seasonal=**True**, error\_action='warn',trace=**True**, supress\_warnings=**True**,stepwise=**True**, random\_state=20,n\_fits=50)  model\_seasondiff= pm.auto\_arima(  data\_train,  start\_p=2,d=0,start\_q=0,  max\_p=6,max\_d=1,max\_q=3,  start\_P=0, D=1, start\_Q=0,  max\_P=5,max\_D=1, max\_Q=3, m=season,  seasonal=**True**, error\_action='warn',trace=**True**, supress\_warnings=**True**,stepwise=**True**, random\_state=20,n\_fits=50) |

### 

### 

### 

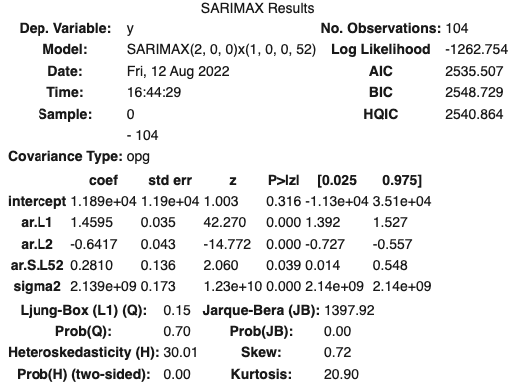
### 

### SARIMA AIC Minimization - No Seasonal Difference



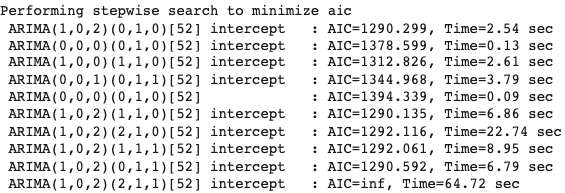
The AIC is similar for all these models. The third coefficient parameter (AR(3)) is not necessary because the AIC is not very different from the AR(2) models. There is not enough data to have multiple seasonal parameters. The two models that stand out are the SARIMA(2,0,0)x(0,0,1)[52] and the SARIMA(2,0,0)X(1,0,0)[52]. Their AICs are 2538.010 and 2538.507 respectively. That difference is negligible. The ACF, PACF, and EACF suggest the model should look more like SARIMA(2,0,0)x(1,0,0)[52].

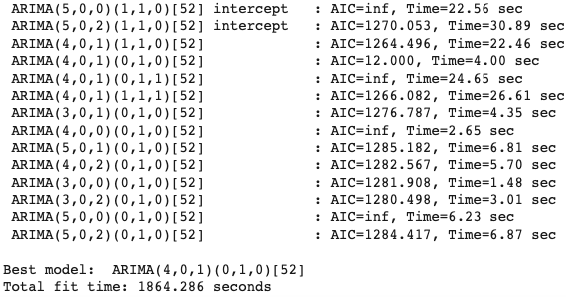
|  |
| --- |
| model\_seasdiff = pm.arima.ARIMA(order=(2,0,0), seasonal\_order=(1, 0, 0, 52)).fit(data\_train) model\_seasdiff.summary() |

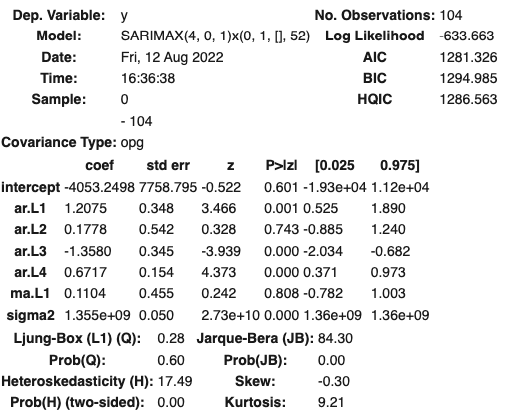


All the parameters are significant (p-value< alpha of 0.05) and the model is viable for further testing.

### SARIMA AIC Minimization - 1 Seasonal Difference



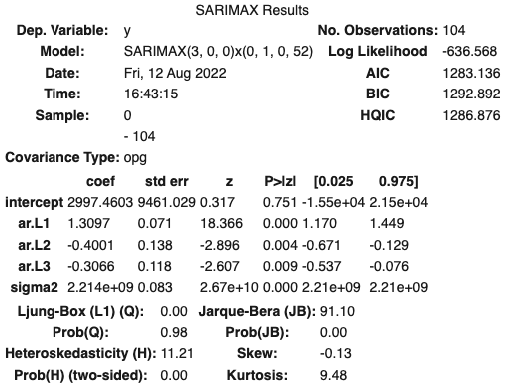




The best model is a SARIMA(4,0,1)X(0,1,0)[52] with an AIC of 1281. A deeper dive into the model shows that not all coefficient parameters are significant. The p-value for the ma.L1 is not less than alpha (0.05), and phi\_2 (ar.L2) is not less than alpha. The model parameters need to be reduced.

AR and MA parameters were reduced until all parameters’ p-values were significant. The final model is SARIMA(3, 0, 0)X(0,1,0)[2]

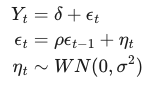
|  |
| --- |
| model\_seasdiff = pm.arima.ARIMA(order=(3,0,0), seasonal\_order=(0, 1, 0, 52)).fit(data\_train) model\_seasdiff.summary() |



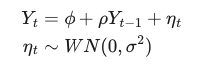
An AR(2) model was used as a comparison, but there is clearly a seasonal trend in the dataset so the model would underfit the data. The AIC and BIC cannot be directly compared because of the seasonal integration. Therefore, further testing needs to be done between these models.

|  |  |  |
| --- | --- | --- |
| **model** | **aic** | **bic** |
| SARIMA(3,0,0)x(0,1,0)[52] | 1306.1 | 1316 |
| SARIMA(2,0,0)x(1,0,0)[52] | 2562.5 | 2575.7 |
| AR(2) | 2565.9 | 2576.5 |

SARIMAX is the standard python package for fitting seasonal models. There is an extra exogenous regressor term. An AR(1) ARIMA model not using the SARIMAX model can be represented by the following:



SARMIX models are represented differently when estimating. An AR(1) SARIMAX model can be represented below:



SARIMAX fits the model by maximum likelihood via Kalman filter (linear quadratic estimation.)

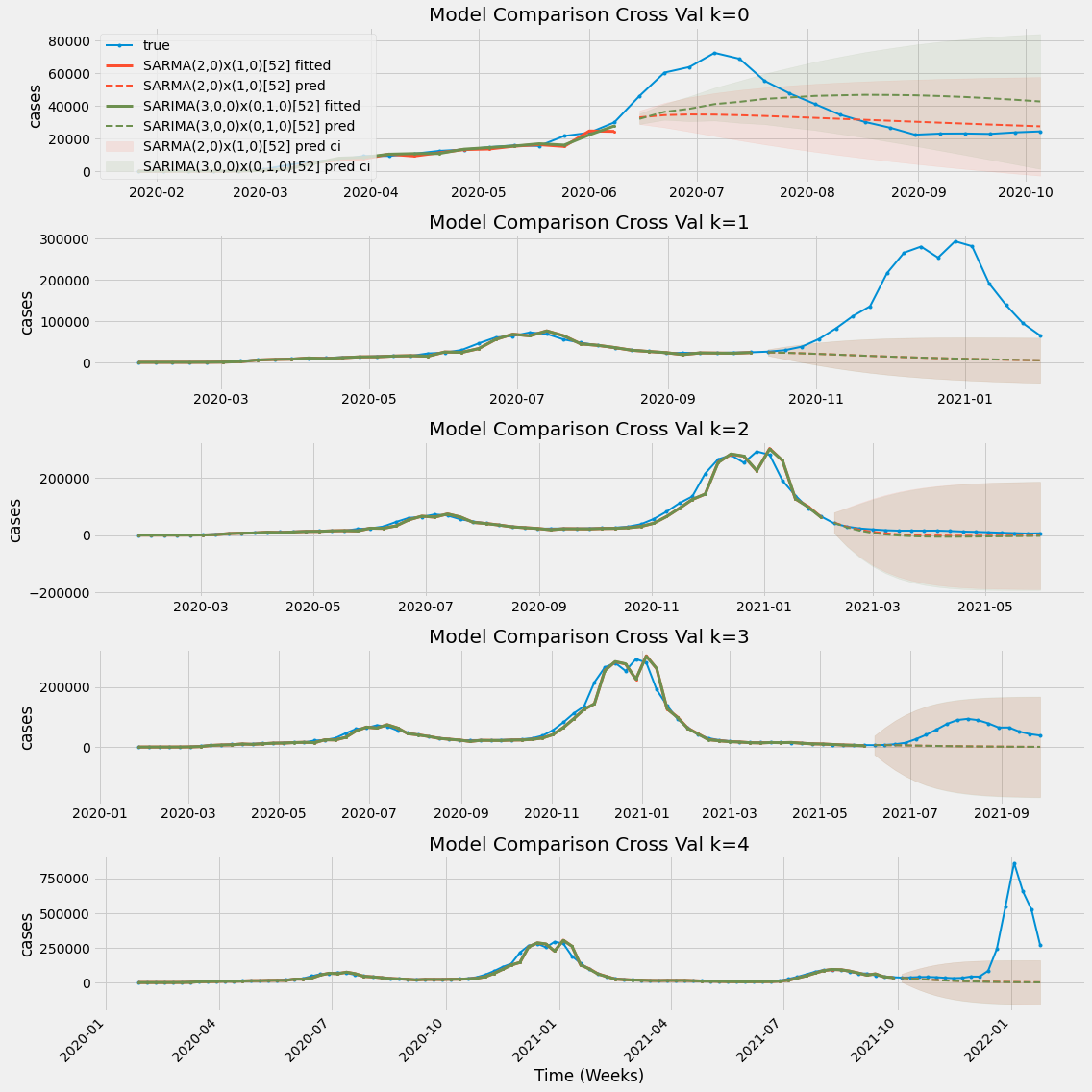
## Model Comparison -SARMA (No Difference) vs SARIMA (Season Difference)

Comparing these models via K-folds cross-validation will determine if the seasonal difference is useful for handling the volatility of the dataset. Intuition would argue the data should not be seasonally integrated because the dataset is already stationary. The two models being compared will be the SARIMA(2,0,0)x(1,0,0)[52] (SARMA model) and SARIMA(3,0,0)(0,1,0)[52] (SARIMA model).

|  |
| --- |
| # Model Parameter Variables pdqParams\_list = [(2, 0 , 0), (3, 0, 0),] PDQParams\_list = [(1, 0, 0, season),(0, 1, 0, season),] model\_names = ['SARMA(2,0)x(1,0)[52]','SARIMA(3,0,0)x(0,1,0)[52]',] |

Time series K-folds cross-validation will be used to compare the two models. This method consists of cutting the time series data into 5 different sections. Half of one section is used to train and the other half is used to validate. The first loop uses the first section of data only. The model is trained on the first training section and validated on the first validation section. The next loop uses the entire first section plus the first half (training section) of the second section to train and the second half of the second section to test. This process will be done 5 times. (k=5) The code is illustrated below.

|  |
| --- |
| **from** sklearn.model\_selection **import** TimeSeriesSplit n\_splits = 5 tscv = TimeSeriesSplit(n\_splits=n\_splits, test\_size=**None**)  **for** k,( tr\_idx, val\_idx) **in** enumerate(tscv.split(data\_train)):  data\_train\_k, data\_val\_k = data\_train.iloc[tr\_idx, :], data\_train.iloc[val\_idx, :]   # true values to validate prediction  y\_true = data\_val\_k.values   # loop through models  **for** inc, pdqParams **in** enumerate(pdqParams\_list):   # fit model  model = SARIMAX(data\_train\_k, order=pdqParams, season=PDQParams\_list[inc]).fit()   # forecast  data\_pred\_k = model.forecast(len(data\_val\_k))  data\_pred\_list.append(data\_pred\_k)  y\_pred = data\_pred\_k.values   # measure accuracy  acc\_dict = find\_prediction\_acc(y\_pred, y\_true, print\_result=**False**) |

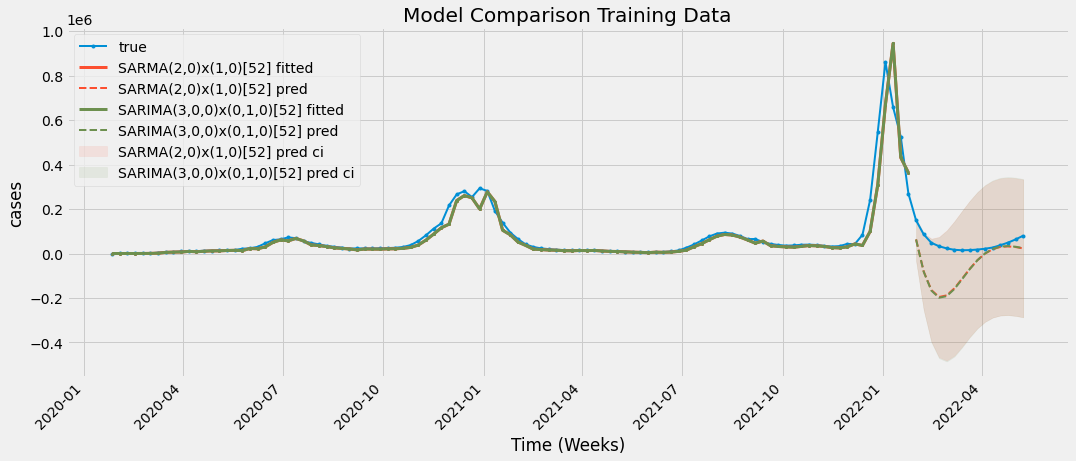


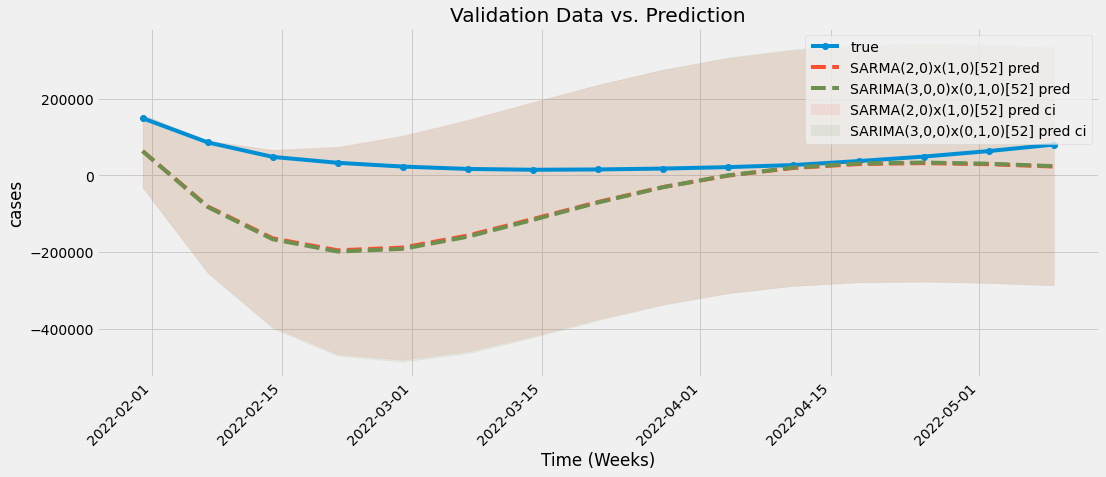
Accuracy Metrics from K-Folds Cross Validation:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **model** | **k** | **mape** | **mae** | **mpe** | **rmse** | **minmax** | **aic** |
| SARMA(2,0)x(1,0)[52] | 0 | 0.284 | 13318.26 | -0.088 | 17609.3 | 0.265 | 373.81 |
| SARIMA(3,0,0)x(0,1,0)[52] | 0 | 0.56 | 18680.665 | 0.305 | 20129.267 | 0.365 | 367.54 |
| SARMA(2,0)x(1,0)[52] | 1 | 0.787 | 137030.376 | -0.787 | 168736.359 | 0.787 | 725.97 |
| SARIMA(3,0,0)x(0,1,0)[52] | 1 | 0.787 | 136938.365 | -0.787 | 168634.155 | 0.787 | 728.03 |
| SARMA(2,0)x(1,0)[52] | 2 | 0.87 | 10418.019 | -0.862 | 11445.527 | 0.87 | 1221.57 |
| SARIMA(3,0,0)x(0,1,0)[52] | 2 | 1.092 | 13261.854 | -1.086 | 14480.494 | 1.091 | 1223.57 |
| SARMA(2,0)x(1,0)[52] | 3 | 0.795 | 47075.006 | -0.789 | 56225.213 | 0.795 | 1583.8 |
| SARIMA(3,0,0)x(0,1,0)[52] | 3 | 0.798 | 47181.622 | -0.792 | 56333.236 | 0.798 | 1585.79 |
| SARMA(2,0)x(1,0)[52] | 4 | 0.698 | 197015.501 | -0.698 | 331721.916 | 0.698 | 1945.32 |
| SARIMA(3,0,0)x(0,1,0)[52] | 4 | 0.714 | 197959.114 | -0.714 | 332331.859 | 0.714 | 1947.28 |

The models are very close. The SARMA(2,0)x(1,0)[52] model gives a better prediction when there are fewer data points. The seasonal integration is not helping the results of the model. It is clear that the seasonal difference is not enhancing the model after being trained on the full dataset. The SARMA(2, 0)x(2,0) is the better model for the dataset.

The dataset has periods of high volatility most notably in January 2021 and 2022. The model struggles to anticipate those rapid peaks in COVID-19 cases. There are only two full seasons (2020 and 2021) available. The seasonal parameters would need to be trained over more seasons in order to account for these sharp peaks.





|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **model** | **k** | **mape** | **mae** | **mpe** | **rmse** | **minmax** | **aic** | **bic** | **llf** |
| SARMA(2,0)x(1,0)[52] | train | 3.57 | 98979.49 | -3.57 | 126039.90 | 3.57 | 2569.85 | 2577.81 | -1281.93 |
| SARIMA(3,0,0)x(0,1,0)[52] | train | 3.61 | 99710.19 | -3.61 | 127668.69 | 3.61 | 2571.87 | 2582.49 | -1281.94 |

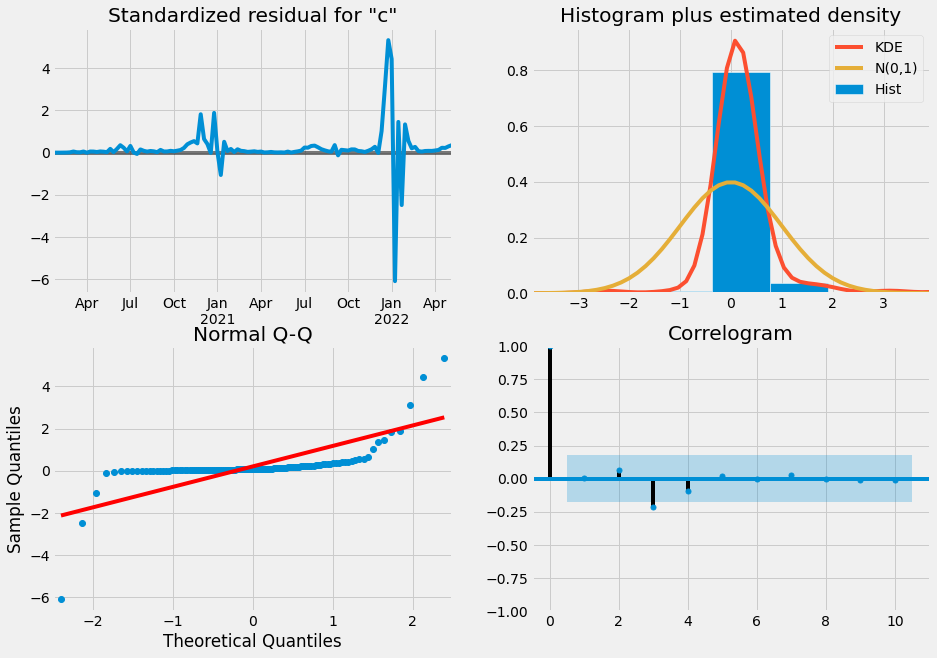
## 

## 

## 

## Residual Analysis - SARMA(2,0)x(1,0)[52]

|  |
| --- |
| model.plot\_diagnostics(figsize=(14, 10)) |



|  |
| --- |
| **def** **shapiro\_wilk\_test**(data, alpha= 0.05):  test\_results = stats.shapiro(data\_train)  print('Shapiro-Wilk Test:')  print(test\_results)  **if** test\_results.pvalue < alpha:  print(f"p-value: {round(test\_results.pvalue, 4)}. The null-hypothesis can be rejected. The data is not normally distributed.")  **else**:  print(f"p-value: {round(test\_results.pvalue, 4)}. The null-hypothesis cannot be rejected. The data is normally distributed.")   **return** test\_results shapiro\_wilk\_test(model.resid) |

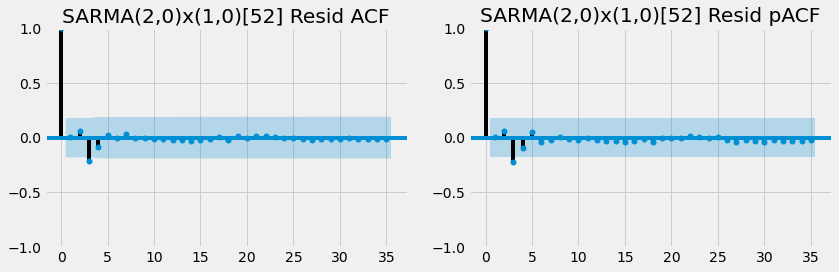
Shapiro-Wilk Test:

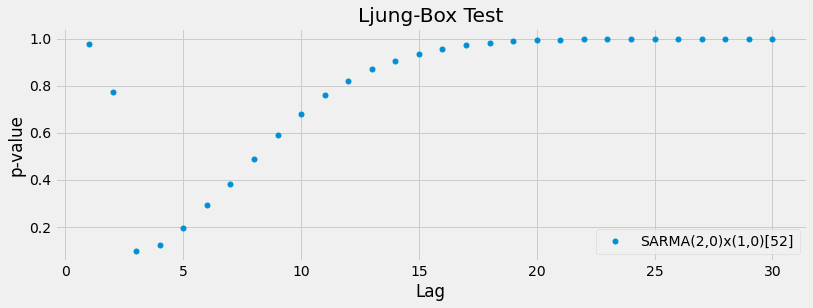
ShapiroResult(statistic=0.5359777808189392, pvalue=1.202452074738043e-16)

p-value: 0.0. The null-hypothesis can be rejected. The data is not normally distributed.

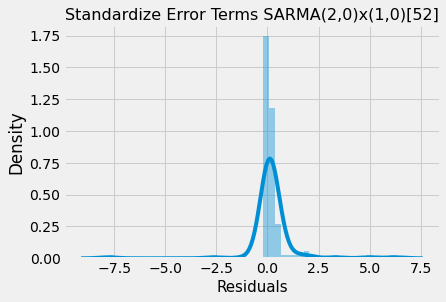
The time series of the model’s residuals are relatively stable except for two periods of high volatility. These two periods correspond with the two spikes in the time series. These two periods also account for the outliers in the Q-Q Plot. The Shapiro-Wilk test confirms the residuals are not normally distributed due to the outliers.

The outliers can be explained because the period of late December 2021 to January 2022 is when the highly contagious Omicron Variant started to spread in California. Also, January 2021 is in the first season of the dataset so the seasonal factors cannot be modeled. December is a month of high travel which leads to a higher spread of coronavirus.





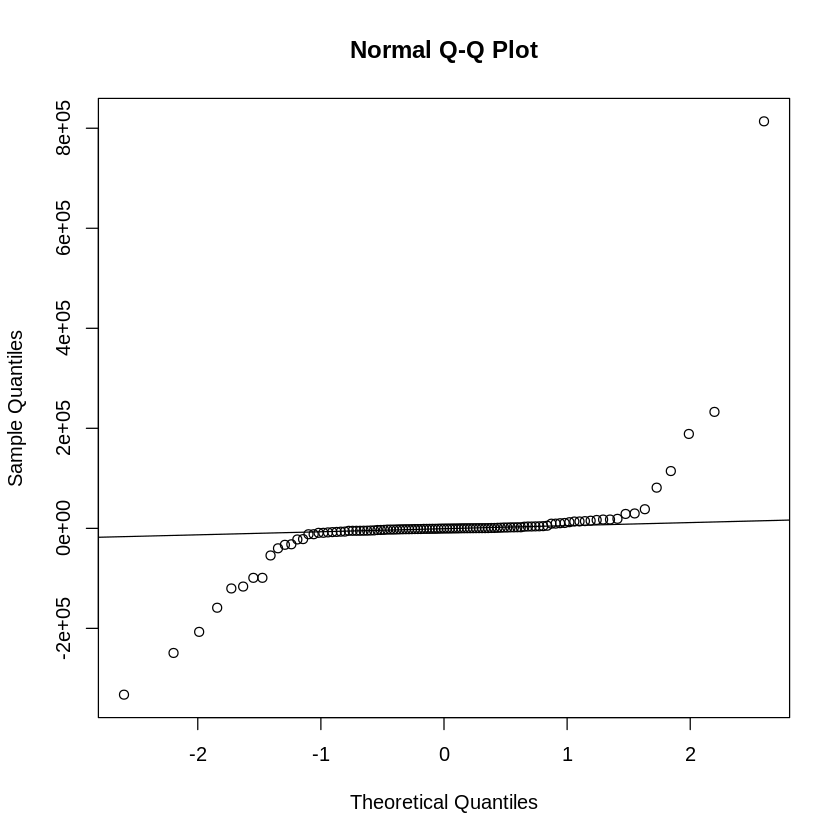
The model residual ACF and PACF plot show the dataset is being fully captured in the model. There is no correlation amongst the residuals except for significant lags at three and four. The Ljung-Box Test confirms the ACF and PACF residual plots.



The histogram of the standardized residuals shows the outliers on either end. The rest of the model seems to be normally distributed.

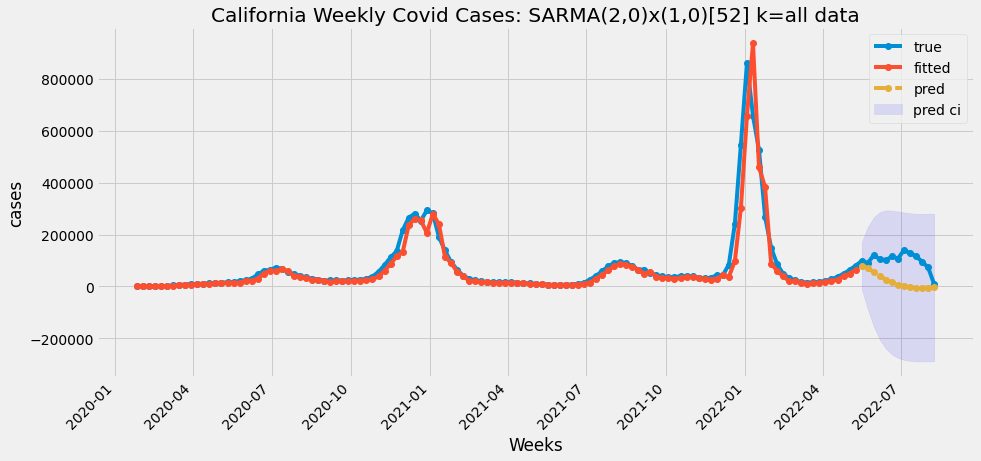
A GARCH(1,1) was fit in R on the residuals of an AR(2) model. The normality of the residuals did not improve and there were still outliers on both ends of the plot.

|  |
| --- |
| **library**(ugarch) qpParam <- c(2, 0) qpGarchParam <- c(1,1) spec <- ugarchspec(variance.model = list(model = "sGARCH",  garchOrder = qpGarchParam,  submodel = NULL,  external.regressors = NULL,  variance.targeting = FALSE),    mean.model = list(armaOrder = qpParam,  external.regressors = NULL,  distribution.model = "norm",  start.pars = list(),  fixed.pars = list()))  garch <- ugarchfit(spec = spec, data = ts\_train, solver.control = list(trace=0)) garchResiduals <- garch@fit$residuals qqnorm(garchResiduals) qqline(garchResiduals) |



## Prediction - SARMA(2,0)x(1,0)[52]

|  |
| --- |
| # SARIMA(2, 0, 0)X(1,0,0)[52] pdqParams = (2, 0, 0); PDQParams = (1, 0, 0, season); model\_name = "SARMA(2,0)x(1,0)[52]"  # train + val data\_train\_all = pd.concat([data\_train, data\_val], axis = 0) y\_true = data\_test.values  # fit model model = SARIMAX(data\_train\_all, order=pdqParams, season\_order= PDQParams).fit() # forecast data\_pred = model.forecast(len(data\_test)) fcast = model.get\_forecast(len(data\_test)) ci = fcast.conf\_int() y\_pred = data\_pred.values |



## 

The model does an adequate job forecasting the next few weeks. The forecast deteriorates after about 6 weeks because the forecast approaches zero after that period. The true values do fall in the prediction confidence interval. The prediction accuracy values are listed below.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **model** | **k** | **mape** | **mae** | **mpe** | **rmse** | **minmax** | **aic** | **bic** | **llf** |
| SARIMA(3,0,0)x(0,1,0)[52] | train+val | 0.82 | 78648.46 | -0.82 | 88574.77 | 0.82 | 2923.31 | 2931.67 | -1458.65 |

# Conclusions

## Statistical Conclusion

Ultimately the SARMA(2,0)x(1, 0)[52] is a good fit for the data according to the residual analysis. The testing data falls into the final forecast’s confidence interval. The forecast is adequate for about six weeks but deteriorates after that time period. COVID-19 is still a relatively new virus and there are only two full seasons (2020 and 2021) of data. As the sample size grows the seasonal parameters in the SARMA model will be more accurate or a better model can be specified.

## Further Recommendations

The model strictly uses past confirmed COVID-19 cases to predict future cases. It does not look at the number of positive tests or tests administered. Future models can take these factors into account. A multivariate model can be fit to include tests administered, positive tests, and deaths. Also, the data can be modeled around a different metric completely, like positive tests. Another suggestion would model the daily data instead of weekly data. The daily data might need to include a GARCH component.

One final suggestion would be to model another highly contagious virus with more seasons of data. That model can then be transferred to the COVID-19 dataset. An example would be to model SARs or other epidemics and see if those models can be applied to COVID-19. This technique could lead to better forecasts in the short term.

## Conclusions

There are so many factors that go into predicting COVID-19 cases. There is clearly a seasonal component in the data. It is worth noting that strict lockdown restrictions were implemented from March 2020 to May 2021. Large gatherings were not permitted and social distancing was widely practiced. Given the circumstances, the model is relatively successful in forecasting future cases.

A vaccine was introduced in mid-2021. Finally, a very contagious Omincron Variant of the virus began to spread in December of 2021. This contagious virus strand led to an outlier amount of cases the model struggled to predict. This outlier event led to the residuals not being normally distributed. The model will improve as more data is introduced and the seasonal variables will be tuned better as the years go on.

# References

Seabold, Skipper, and Josef Perktold. “statsmodels: Econometric and statistical modeling with python.” Proceedings of the 9th Python in Science Conference. 2010.

Smith, Taylor G., et al. pmdarima: ARIMA estimators for Python, 2017-, http://www.alkaline-ml.com/pmdarima [Online; accessed 2022-08-12].

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261-272.

Brownlee, J. (2020, September 9). *Time series forecasting performance measures with python*. Machine Learning Mastery. Retrieved August 12, 2022, from https://machinelearningmastery.com/time-series-forecasting-performance-measures-with-python/

Bui, C. (n.d.). *Chibui191/bitcoin\_volatility\_forecasting: GARCH and multivariate LSTM forecasting models for Bitcoin realized volatility with potential applications in crypto options trading, hedging, portfolio management, and Risk Management*. GitHub. Retrieved August 12, 2022, from https://github.com/chibui191/bitcoin\_volatility\_forecasting

Cryer, J. D., & Chan, K.-S. (2008). *Time series analysis with applications in R*. Springer.

Keith, M. (2022, March 16). *Forecast with Arima in python more easily with Scalecast*. Medium. Retrieved August 12, 2022, from https://towardsdatascience.com/forecast-with-arima-in-python-more-easily-with-scalecast-35125fc7dc2e

Pawar, A. (2022). Seasonal and Nonseasonal GARCH Time Series Analysis: Case Study of Bitcoin Historical and S&P 500 stock datasets. *ResearchGate*.

Radecic, D. (2022, March 23). *Time series from scratch - train/test splits and Evaluation Metrics*. Medium. Retrieved August 12, 2022, from https://towardsdatascience.com/time-series-from-scratch-train-test-splits-and-evaluation-metrics-4fd654de1b37

*Sarimax and Arima: Frequently asked questions (FAQ)¶*. statsmodels. (n.d.). Retrieved August 12, 2022, from https://www.statsmodels.org/stable/examples/notebooks/generated/statespace\_sarimax\_faq.html

Telmo-Correa. (n.d.). *Telmo-Correa/Time-Series-analysis: Self Study on Cryer and Chan's "Time Series analysis with applications in R"*. GitHub. Retrieved August 12, 2022, from https://github.com/telmo-correa/time-series-analysis

Verma, Y. (2021, September 6). *Complete guide to dickey-fuller test in time-series analysis*. Analytics India Magazine. Retrieved August 12, 2022, from https://analyticsindiamag.com/complete-guide-to-dickey-fuller-test-in-time-series-analysis/