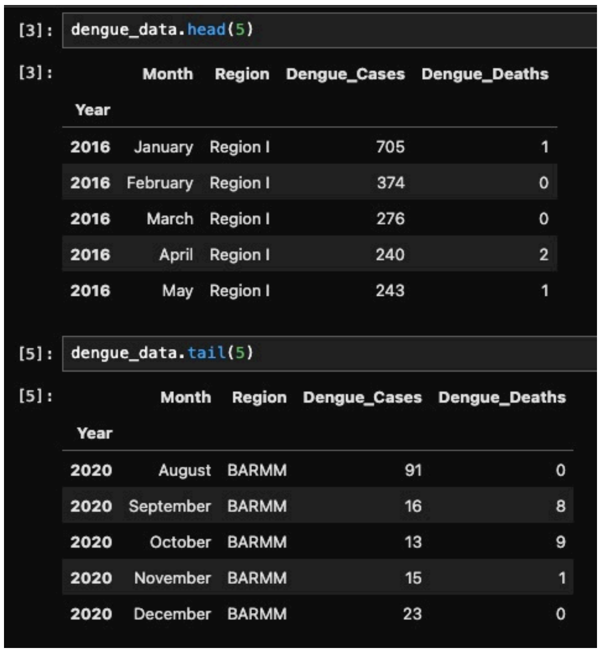
Data set contains the recorded number of dengue cases per region of the Philippines from year 2016 to 2020. It can be used to find trends about the disease as well as spatiotemporal analysis that can result into data-driven solution about the trends of the desease for the past 5 years.

Source:https://www.kaggle.com/datasets/vincentgupo/dengue-cases-in-the-philippines

DATA INSPECTIONS

- We read in the dataset using the "Year" col as the index value col. This would allow us to see things as a change of time in years.
- We can then inspect our data using the .head()
    - In this case we decided to look at the first 5 and last 5 of the data to ensure our ordering is correct.



```
[3]: dengue_data.head(5)
```

```
[3]:            Month  Region  Dengue_Cases  Dengue_Deaths
     Year
     2016     January  Region I           705              1
     2016    February  Region I           374              0
     2016       March  Region I           276              0
     2016       April  Region I           240              2
     2016         May  Region I           243              1
```

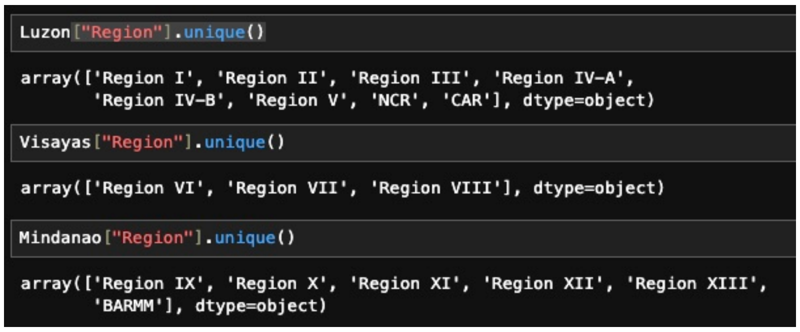
```
[5]: dengue_data.tail(5)
```

```
[5]:            Month  Region  Dengue_Cases  Dengue_Deaths
     Year
     2020      August   BARMM            91              0
     2020   September   BARMM            16              8
     2020     October   BARMM            13              9
     2020    November   BARMM            15              1
     2020    December   BARMM            23              0
```

- From here we can see that we are working with 5 cols. Our years are ordered correctly and it looks like our "Month" are also ordered for us.
- We have a "Region" cols but do not know how many regions are there.
- We can use the .unique() to see all the regions available in that col.

- From a quick google search we see that the Philippines are divided into 3 mains regions
    - Regions of Luzon
        - Region 1, 2, 3, 4A, 4B, 5, CAR (Cardillera Administrative Region), NCR (National Capital Region)
    - Regions of Visayas
        - Region 6, 7, 8, NIR (Negros Island Region)
    - Regions of Mindanao
        - Region 9, 10, 11, 12, 13, NARMM (Bangsamoro Autonomous Region in Muslim Mindanao

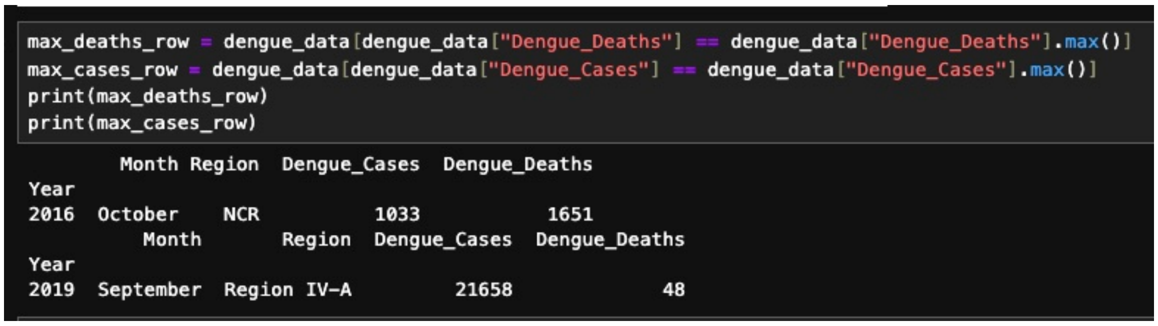
- We can approach this from sub-region to main region. But, I think its better to first observe the bigger picture thus, look at the larger main regions.
- We will create 3 regions
  - Use .unique() to ensure the sub-regions are allocated correctly.



```
Luzon["Region"].unique()

array(['Region I', 'Region II', 'Region III', 'Region IV-A',
       'Region IV-B', 'Region V', 'NCR', 'CAR'], dtype=object)

Visayas["Region"].unique()

array(['Region VI', 'Region VII', 'Region VIII'], dtype=object)

Mindanao["Region"].unique()

array(['Region IX', 'Region X', 'Region XI', 'Region XII', 'Region XIII',
       'BARMM'], dtype=object)
```

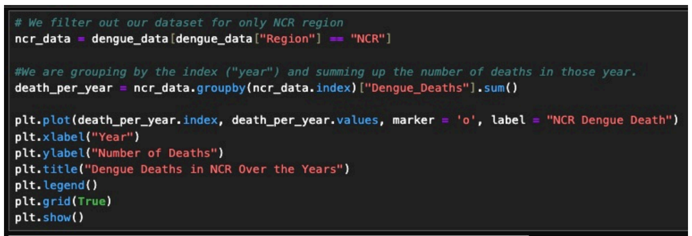
Analysis of Luzon Region

- What year did the highest cases on death and number of cases. Which subregion did it occur in?

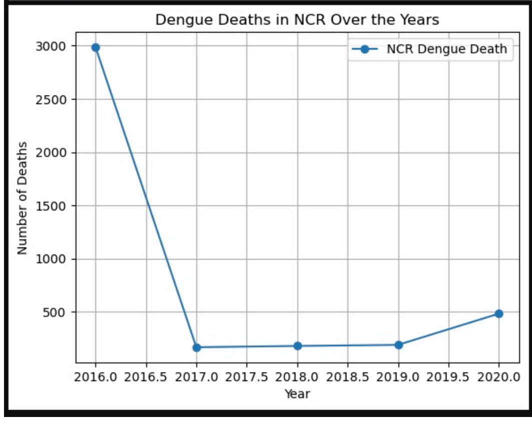


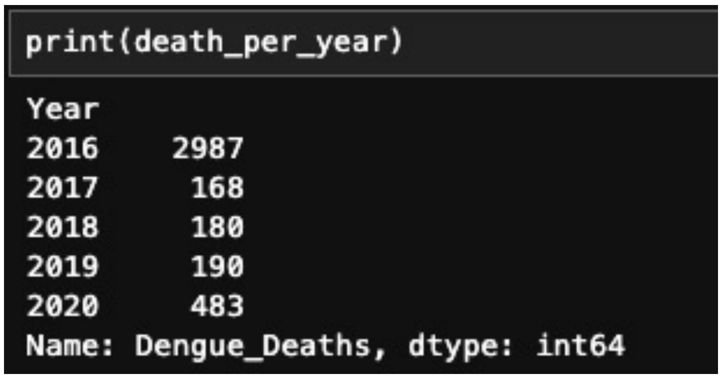
```
max_deaths_row = dengue_data[dengue_data["Dengue_Deaths"] == dengue_data["Dengue_Deaths"].max()]
max_cases_row = dengue_data[dengue_data["Dengue_Cases"] == dengue_data["Dengue_Cases"].max()]
print(max_deaths_row)
print(max_cases_row)

        Month Region  Dengue_Cases  Dengue_Deaths
Year
2016  October    NCR          1033           1651
          Month      Region  Dengue_Cases  Dengue_Deaths
Year
2019  September  Region IV-A         21658             48
```

  - We can see that the year 2016, the region NCR had the most number of Dengue fever deaths in the Luzon region
  - 2019 is the year with the highest number of Dengue fever cases, which occurred in region IV-A.



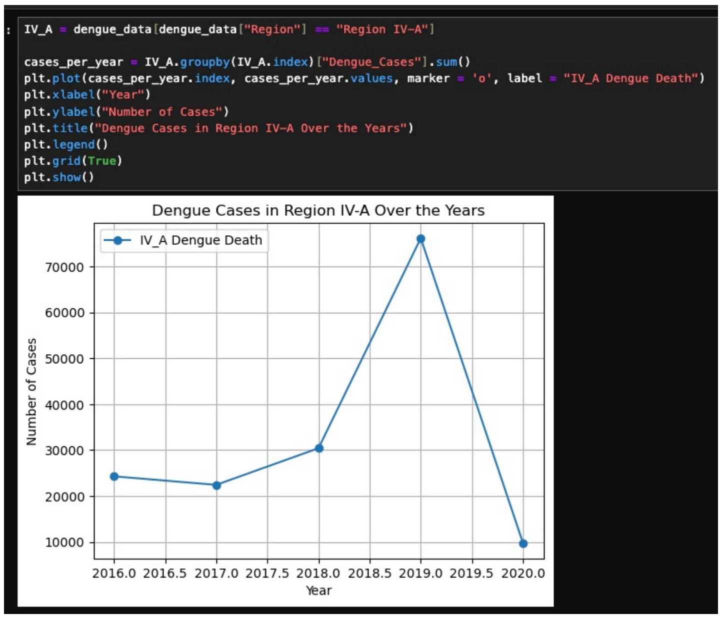
```
# We filter out our dataset for only NCR region
ncr_data = dengue_data[dengue_data["Region"] == "NCR"]

#We are grouping by the index ("year") and summing up the number of deaths in those year.
death_per_year = ncr_data.groupby(ncr_data.index)["Dengue_Deaths"].sum()

plt.plot(death_per_year.index, death_per_year.values, marker = 'o', label = "NCR Dengue Death")
plt.xlabel("Year")
plt.ylabel("Number of Deaths")
plt.title("Dengue Deaths in NCR Over the Years")
plt.legend()
plt.grid(True)
plt.show()
```



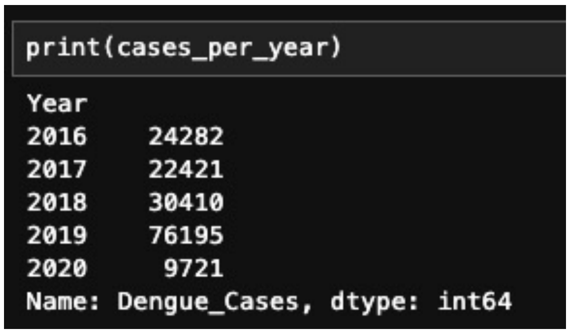


```
print(death_per_year)

Year
2016    2987
2017     168
2018     180
2019     190
2020     483
Name: Dengue_Deaths, dtype: int64
```

- What we see here is very interesting. The total number of deaths per year decreased by 94.4% from 2016 -1017.
    - WHAT MAY HAVE CAUSE THIS?

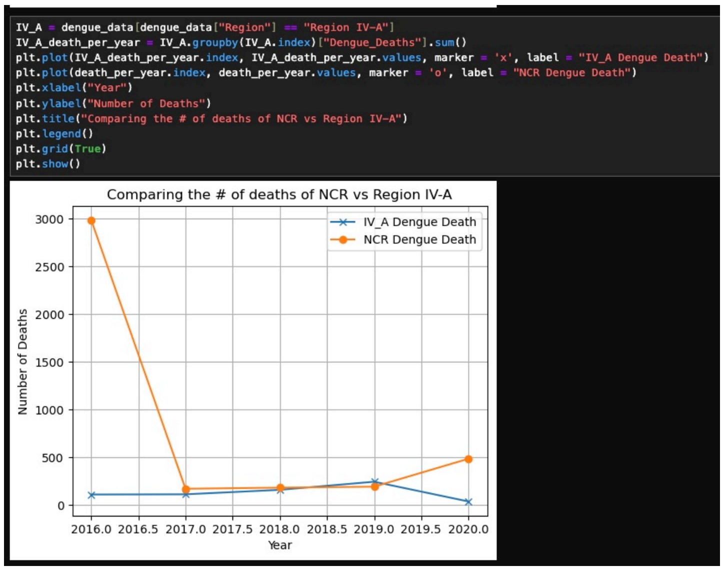


```
IV_A = dengue_data[dengue_data["Region"] == "Region IV-A"]

cases_per_year = IV_A.groupby(IV_A.index)["Dengue_Cases"].sum()
plt.plot(cases_per_year.index, cases_per_year.values, marker = 'o', label = "IV_A Dengue Death")
plt.xlabel("Year")
plt.ylabel("Number of Cases")
plt.title("Dengue Cases in Region IV-A Over the Years")
plt.legend()
plt.grid(True)
plt.show()
```



```
print(cases_per_year)

Year
2016     24282
2017     22421
2018     30410
2019     76195
2020      9721
Name: Dengue_Cases, dtype: int64
```

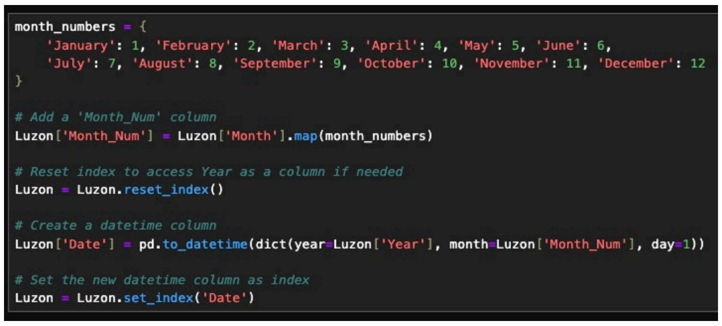
- From this data we see there is a large case of Dengue fever in 2019. This is the same year as COVID.



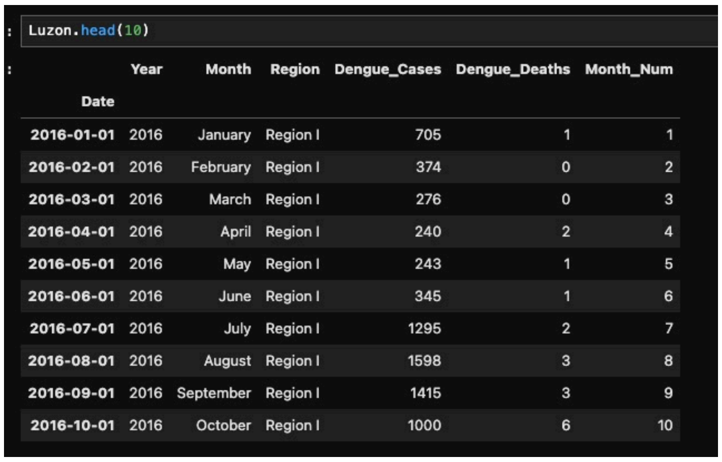
```
IV_A = dengue_data[dengue_data["Region"] == "Region IV-A"]
IV_A_death_per_year = IV_A.groupby(IV_A.index)["Dengue_Deaths"].sum()
plt.plot(IV_A_death_per_year.index, IV_A_death_per_year.values, marker = 'x', label = "IV_A Dengue Death")
plt.plot(death_per_year.index, death_per_year.values, marker = 'o', label = "NCR Dengue Death")
plt.xlabel("Year")
plt.ylabel("Number of Deaths")
plt.title("Comparing the # of deaths of NCR vs Region IV-A")
plt.legend()
plt.grid(True)
plt.show()
```

- We can see there is a slight increase in the numbers of death in the NCR region while it is the opposite for region IV-A.

- Restructure the dataframe to where my index value is a Time Series value.



```
month_numbers = {
    'January': 1, 'February': 2, 'March': 3, 'April': 4, 'May': 5, 'June': 6,
    'July': 7, 'August': 8, 'September': 9, 'October': 10, 'November': 11, 'December': 12
}

# Add a 'Month_Num' column
Luzon['Month_Num'] = Luzon['Month'].map(month_numbers)

# Reset index to access Year as a column if needed
Luzon = Luzon.reset_index()

# Create a datetime column
Luzon['Date'] = pd.to_datetime(dict(year=Luzon['Year'], month=Luzon['Month_Num'], day=1))

# Set the new datetime column as index
Luzon = Luzon.set_index('Date')
```

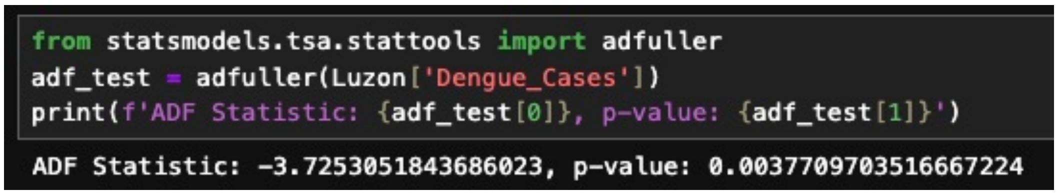


```
Luzon.head(10)
```

Date	Year	Month	Region	Dengue_Cases	Dengue_Deaths	Month_Num
2016-01-01	2016	January	Region I	705	1	1
2016-02-01	2016	February	Region I	374	0	2
2016-03-01	2016	March	Region I	276	0	3
2016-04-01	2016	April	Region I	240	2	4
2016-05-01	2016	May	Region I	243	1	5
2016-06-01	2016	June	Region I	345	1	6
2016-07-01	2016	July	Region I	1295	2	7
2016-08-01	2016	August	Region I	1598	3	8
2016-09-01	2016	September	Region I	1415	3	9
2016-10-01	2016	October	Region I	1000	6	10

Using ADF (Augmented Dickey-Fuller) test to see if my times series data has something call a "unit root".

- This statistical test proposes 2 events
  - 1: Null Hypotheses: My data has a unit root thus it is NON-STATIONARY.
    - has trends/seasonality that needs to be removed
  - 2: Alternative Hypothesis: My dad has no unit root thus it is STATIONARY
    - mean and variance are constant over time.
- Need a p-values < 0.05
-



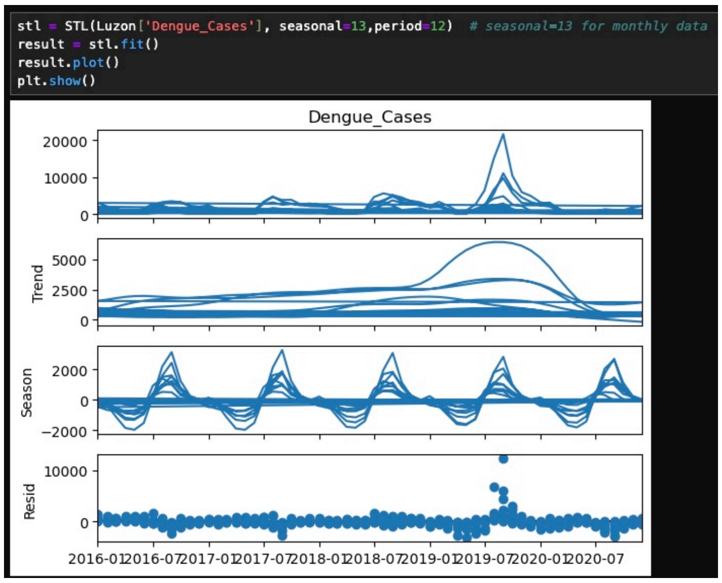
```
from statsmodels.tsa.stattools import adfuller
adf_test = adfuller(Luzon['Dengue_Cases'])
print(f'ADF Statistic: {adf_test[0]}, p-value: {adf_test[1]}')

ADF Statistic: -3.7253051843686023, p-value: 0.0037709703516667224
```

  - Our data is stationary, thus we do not need to apply a

Using STL Decomposition to separate trend, seasonality and residuals

- Seasonal and Trend decomposition using Loess
- Break down a time series into 3 main parts
  - Trend
    - long term direction of data (is it going up, down or plateau?)
  - Seasonality
    - Regular, repeating patterns that happen at the same time each cycle.
  - Remainder
    - The left over after extracting the first 2. This are the random "noise"
- We decompose a time series
  - to understand patterns
  - to improve forecasting
    - works better when they can focus on each component separately.
  - spot anomalies
  - To build better models
    - Some advance models use decomposed components to make more accurate predictions.

- LOESS
    - A technique to estimate the trend and seasonality, even as it change with time.
    - Good for complex or changing patterns.

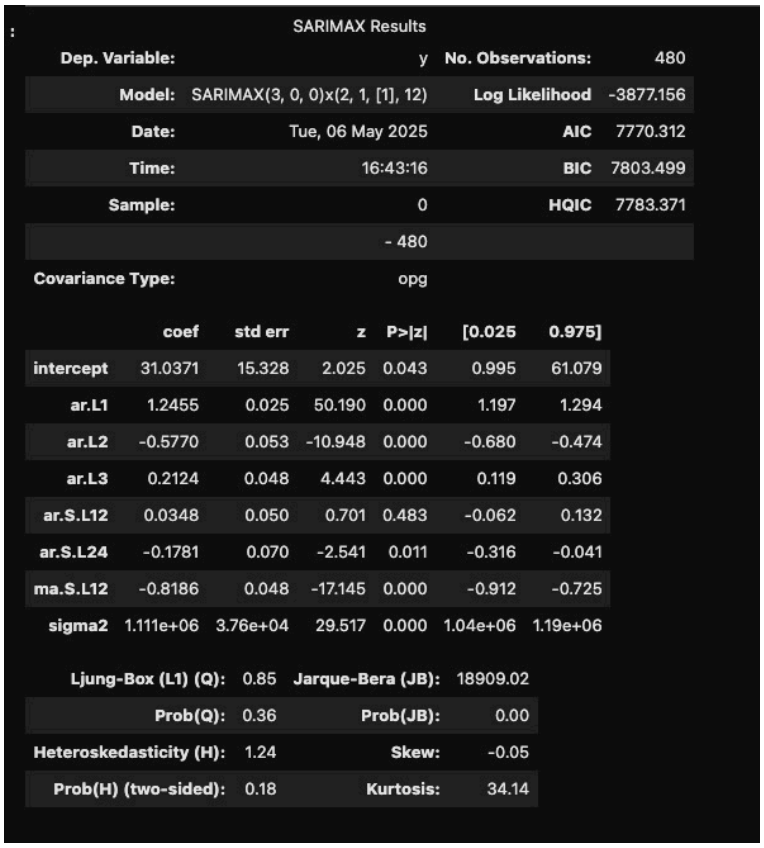


```
stl = STL(Luzon['Dengue_Cases'], seasonal=13,period=12)  # seasonal=13 for monthly data
result = stl.fit()
result.plot()
plt.show()
```

- The season graph tells us that Dengue does seem to be a seasonal disease due to its oscillating pattern. If we look at the trend graph, it follows our original data.

Modeling Fitting

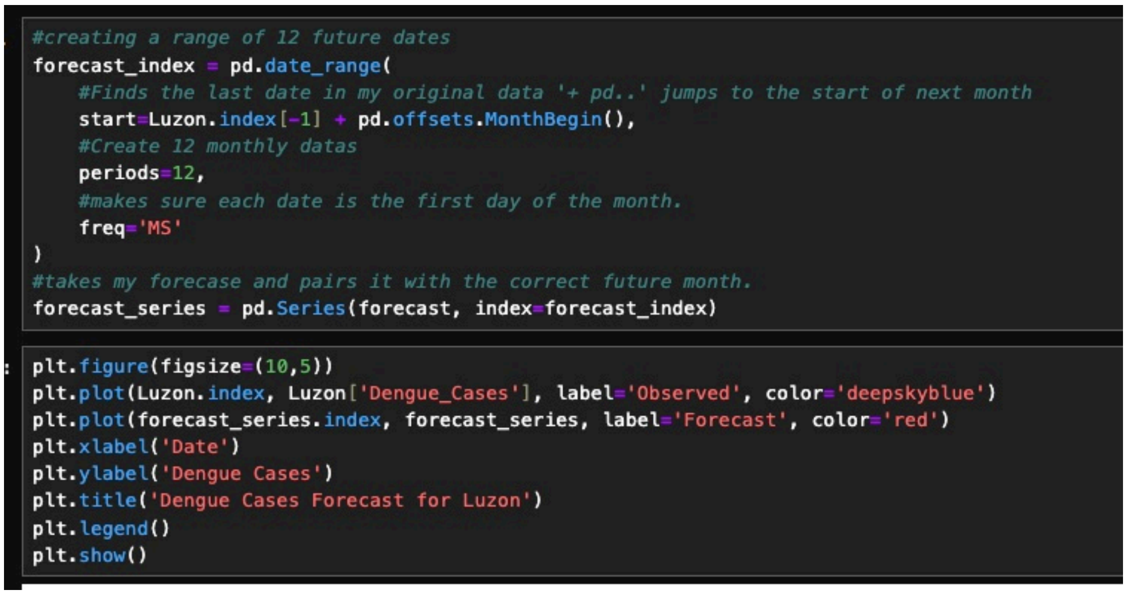
- This is building a math model using ARIMA to capture patterns in our time series data to make forecasts.
- The ARIMA is using past dengue case count to learn the underlying patterns to predict future values.
- The reason for doing this is to build a model that can accurately forecast future dengue cases based on historical trends and seasonal cycles.
- ARIMA uses (p,d,q) parameters
    - p = Number of **autoregressive** terms (how much past values affect current)
        - Ex: Guessing day x by using past two days data
    - d = degree of **differencing** (to remove trends)
    - q = number of **moving average** terms (how much past errors affect current)



```
:                              SARIMAX Results
      Dep. Variable:                          y   No. Observations:       480
             Model:  SARIMAX(3, 0, 0)x(2, 1, [1], 12)   Log Likelihood   -3877.156
              Date:              Tue, 06 May 2025               AIC    7770.312
              Time:                      16:43:16               BIC    7803.499
            Sample:                             0              HQIC    7783.371
                                           - 480
   Covariance Type:                           opg

                  coef    std err        z    P>|z|    [0.025    0.975]
   intercept   31.0371     15.328    2.025    0.043     0.995    61.079
       ar.L1    1.2455      0.025   50.190    0.000     1.197     1.294
       ar.L2   -0.5770      0.053  -10.948    0.000    -0.680    -0.474
       ar.L3    0.2124      0.048    4.443    0.000     0.119     0.306
    ar.S.L12    0.0348      0.050    0.701    0.483    -0.062     0.132
    ar.S.L24   -0.1781      0.070   -2.541    0.011    -0.316    -0.041
    ma.S.L12   -0.8186      0.048  -17.145    0.000    -0.912    -0.725
      sigma2 1.111e+06   3.76e+04   29.517    0.000  1.04e+06  1.19e+06

      Ljung-Box (L1) (Q):   0.85   Jarque-Bera (JB):   18909.02
                Prob(Q):   0.36           Prob(JB):       0.00
 Heteroskedasticity (H):   1.24               Skew:      -0.05
    Prob(H) (two-sided):   0.18           Kurtosis:      34.14
```

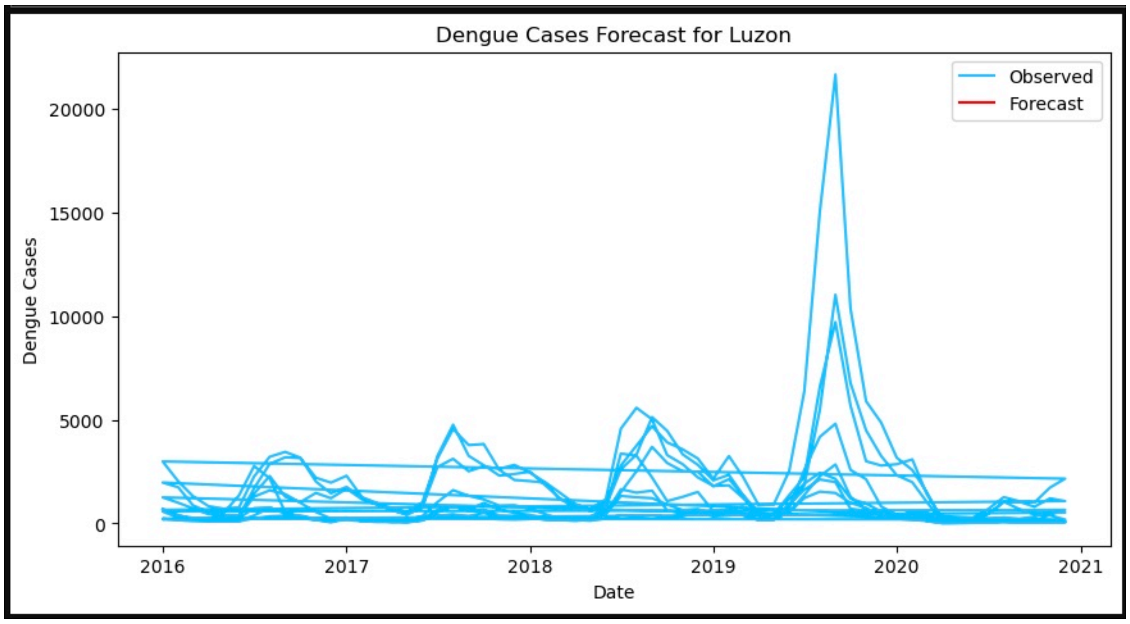
- Model Parameter (3, 0, 0) (0,0,1) with autoresgressive terms: 3 = looking into past 3 months. I (differencing : 0 (no regular differencing. MA (moving average) 0 (no regular MA terms).
- Season AR: 2 looks at the past 2 years, since seasonality is 12.
- Seasonal I : 1 (seasonal differencing)
- Seasonal MA: 1
- Seasonal period = 12.
- Log Likeihood, AIC and BIC are goodness of fits metrics.
    - lower AIC/BIC values indicate better models.
    - ar.L1, ar.L2, ar.L3: Regular autoregressive terms (how much last 3 months affect now)
    - ar.S.L12, ar.S.L24: Seasonal AR terms (how much the same month last year, and two years ago, affect now)
    - ar.L1 (1.24) How strongly the value from 1 period ago affects my current prediction. The positive value means they move in the same direction
    - ar.L2 (-0.577) The influence of the value from 2 periods ago. The negative value suggests a slight reversal effect
    - ma.L1 (0.4824) How much past prediction errors affect current prediction
    - ma.S.L12 (0.2191) Seasonal effect (at 12-month intervals)
    - sigma2 (8.244e+05) Variance of the error term (how much unexplained variation exists)
    - std err :Lower values mean more confident estimates
    - z : The strength of the effect (coefficient ÷ std err)
    - P>|z| - If below 0.05, the coefficient is statistically significant
    - Prob(Q), Prob(H): greater than 0.05 is good.
        - errors are random and variance is stable.
    - Heteroskedasticity (H):
        - Tests if variance of residuals is constant. Prob(H) > 0.05 is good (no evidence variance changes over time).

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
- Method to fix
    - Simplify model, reduce the number of p and q
    - Make sure time series is stationary
    - Avoid redundant terms.
    - Increase data size



```
#creating a range of 12 future dates
forecast_index = pd.date_range(
    #Finds the last date in my original data '+ pd..' jumps to the start of next month
    start=Luzon.index[-1] + pd.offsets.MonthBegin(),
    #Create 12 monthly datas
    periods=12,
    #makes sure each date is the first day of the month.
    freq='MS'
)
#takes my forecase and pairs it with the correct future month.
forecast_series = pd.Series(forecast, index=forecast_index)
```

```
plt.figure(figsize=(10,5))
plt.plot(Luzon.index, Luzon['Dengue_Cases'], label='Observed', color='deepskyblue')
plt.plot(forecast_series.index, forecast_series, label='Forecast', color='red')
plt.xlabel('Date')
plt.ylabel('Dengue Cases')
plt.title('Dengue Cases Forecast for Luzon')
plt.legend()
plt.show()
```



- Need to revisit data set to see why forecast was not plotted.