index

April 27, 2022

1 Testing for Trends - Lab

1.1 Introduction

In this lab, you'll practice your knowledge of testing for stationarity.

1.2 Objectives

You will be able to:

- Use rolling statistics as a check for stationarity
- Use the Dickey-Fuller test and conclude whether or not a dataset is exhibiting stationarity

1.3 Importing the data

Let's look at some new data. In this lab, we'll work with a time series in Python by using the popular Air Passengers dataset.

Start by running the cell below to import the necessary libraries.

[2]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
%matplotlib inline
```

The dataset is stored in `'passengers.csv'`. Import it and view the first five rows.

[3]:
```
# Import 'passengers.csv'
data = pd.read_csv('passengers.csv')

# View the first five rows
data.head()
```

[3]:

	Month	#Passengers
0	1949-01-01	112
1	1949-02-01	118
2	1949-03-01	132
3	1949-04-01	129
4	1949-05-01	121

Change the `'Month'` column over to a `datetime` type and make sure it is set as the index of the DataFrame.

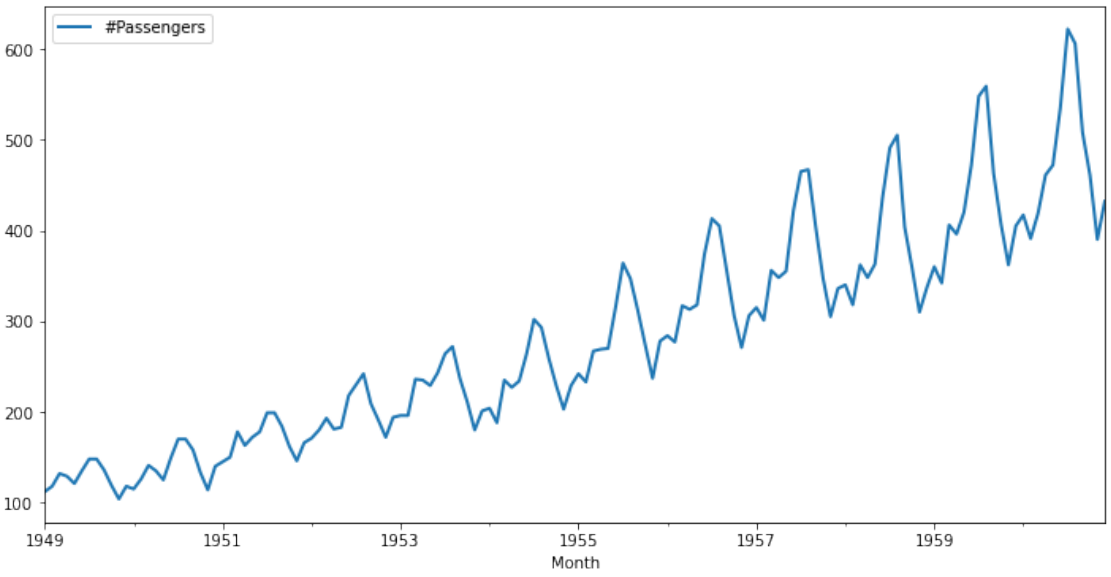
[5]:
```
# Change the type of 'Month' to datetime
data["Month"] = pd.to_datetime(data["Month"])

# Set 'Month' as the index
data.set_index("Month", inplace = True)
```

[7]:
```
# Check the index
data.index
```

[7]:
```
DatetimeIndex(['1949-01-01', '1949-02-01', '1949-03-01', '1949-04-01',
               '1949-05-01', '1949-06-01', '1949-07-01', '1949-08-01',
               '1949-09-01', '1949-10-01',
               …
               '1960-03-01', '1960-04-01', '1960-05-01', '1960-06-01',
               '1960-07-01', '1960-08-01', '1960-09-01', '1960-10-01',
               '1960-11-01', '1960-12-01'],
              dtype='datetime64[ns]', name='Month', length=144, freq=None)
```

Now that we have successfully created a time series, we can use the `.plot()` method in pandas to visually inspect this time series.

[10]:
```
# Plot the time series data
data.plot(kind = "line",figsize=(12,6), linewidth=2, fontsize=10);
```



Wec can see that that there is an overall increasing trend in the data along with some seasonal variations. However, it might not always be possible to make such visual inferences. Let's reconfirm

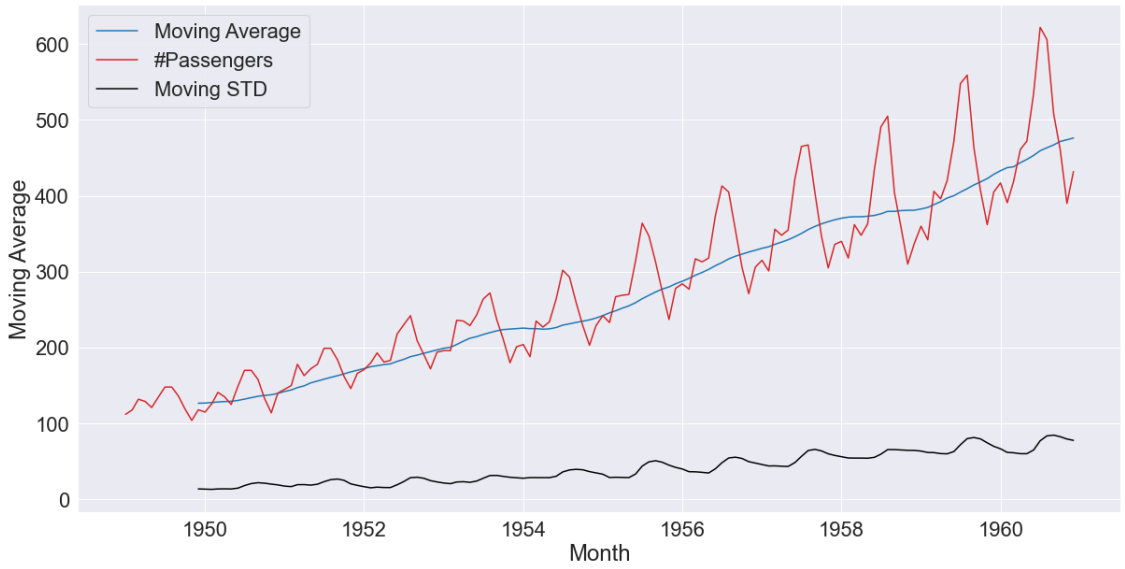
this here using both **rolling statistics** and the **Dickey-Fuller test**.

1.4 Rolling Statistics

Use the `.rolling()` method to find the rolling mean and rolling std with a window of 12 months. Plot the original curve along with the rolling mean and standard error.

[28]:
```
# Determine rolling statistics
data["Moving Average"] = data["#Passengers"].rolling(12).mean()
data["Moving STD"] = data["#Passengers"].rolling(12).std()
```

[31]:
```
# Plot rolling statistics
import seaborn as sns
sns.set(rc={"figure.figsize":(20, 10)}, font_scale=2)

sns.lineplot(x = data.index, y = data["Moving Average"],
             color = "tab:blue", label = "Moving Average");
sns.lineplot(x = data.index, y = data["#Passengers"],
             color = "tab:red", label = "#Passengers");
sns.lineplot(x = data.index, y = data["Moving STD"],
             color = "black", label = "Moving STD");

plt.legend();
```



[33]:
```
## G

# # Determine rolling statistics
# roll_mean = data.rolling(window=12, center=False).mean()
# roll_std = data.rolling(window=12, center=False).std()
```

```
# # Plot rolling statistics
# fig = plt.figure(figsize=(12,6))
# plt.plot(data, color='blue',label='Original')
# plt.plot(roll_mean, color='red', label='Rolling Mean')
# plt.plot(roll_std, color='black', label = 'Rolling Std')
# plt.legend(loc='best')
# plt.title('Rolling Mean & Standard Deviation')
# plt.show()
```

Though the variation in standard deviation is small, the mean is increasing with time and thus, this is not a stationary series.

1.5 Dickey-Fuller Test

Use the Dickey-Fuller test to verify your visual result.

[74]:
```
from statsmodels.tsa.stattools import adfuller

adfuller_result = adfuller(data["#Passengers"])

lis = ["Test Statistics:", "P Value:",
       "Number of Lags:", "Number of Observations:",
       "Critical values", "Maximized Information Criterion:"]

for i, item in enumerate(adfuller_result):
    if lis[i] == "Critical values":
        print(lis[i] + " " + list(item.keys())[0] + ":     " , item[list(item.
 ↪keys())[0]])
        print(lis[i] + " " + list(item.keys())[1] + ":     " , item[list(item.
 ↪keys())[1]])
        print(lis[i] + " " + list(item.keys())[2] + ":     " , item[list(item.
 ↪keys())[2]])
    else:
        print(lis[i], "      ", item)




# G
# from statsmodels.tsa.stattools import adfuller

# # Perform Dickey-Fuller test:
# print ('Results of Dickey-Fuller Test: \n')
# dftest = adfuller(data['#Passengers'])
```

```
# # Extract and display test results in a user friendly manner
# dfoutput = pd.Series(dftest[0:4], index=['Test Statistic', 'p-value', '#Lags␣
 ↪Used', 'Number of Observations Used'])
# for key,value in dftest[4].items():
#     dfoutput['Critical Value (%s)'%key] = value
# print(dfoutput)
```

```
Test Statistics:                  0.815368879206047
P Value:                  0.991880243437641
Number of Lags:                  13
Number of Observations:                 130
Critical values 1%:               -3.4816817173418295
Critical values 5%:               -2.8840418343195267
Critical values 10%:               -2.578770059171598
Maximized Information Criterion:                  996.6929308390189
```

1.6 Summary

In this lab, you checked for the stationarity of a time series in Python. Next, we'll further explore stationarity and how to make sure to make time series stationary!