Practical 10 - JosiahTeh

January 4, 2022

1 First Name: Josiah

2 Last Name: Teh

3 Import Libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
```

4 Read in champagne.CSV File

[3]:
```
# hint lecture cell 2
champagne = pd.read_csv('champagne.csv')
champagne.head()
```

[3]:

	Month	champagne
0	1964-01	2815
1	1964-02	2672
2	1964-03	2755
3	1964-04	2721
4	1964-05	2946

5 Data Management

6 convert champagne['Month'] to datetime format

[4]:
```
# hint lecture cell 3
from datetime import datetime
champagne['Month'] = pd.to_datetime(champagne['Month'], format = '%Y-%m')
champagne.head()
```

[4]:

	Month	champagne
0	1964-01-01	2815
1	1964-02-01	2672

```
2 1964-03-01       2755
3 1964-04-01       2721
4 1964-05-01       2946
```

7 Set 'Month' column as index

[5]:
```
# hint lecture cell 4
champagne.set_index('Month', inplace = True)
champagne.head()
```

[5]:
```
            champagne
Month
1964-01-01       2815
1964-02-01       2672
1964-03-01       2755
1964-04-01       2721
1964-05-01       2946
```

8 Convert champagne['champagne'] to numeric and print the description of champagne['champagne'] column

[6]:
```
# hint lecture cell 5
champagne['champagne'] = pd.to_numeric(champagne['champagne'])
print(champagne.describe())
```

```
          champagne
count    105.000000
mean    4761.152381
std     2553.502601
min     1413.000000
25%     3113.000000
50%     4217.000000
75%     5221.000000
max    13916.000000
```

9 print the index of champagne

[7]:
```
# hint lecture cell 6
champagne.index
```

[7]:
```
DatetimeIndex(['1964-01-01', '1964-02-01', '1964-03-01', '1964-04-01',
               '1964-05-01', '1964-06-01', '1964-07-01', '1964-08-01',
               '1964-09-01', '1964-10-01',
               …
               '1971-12-01', '1972-01-01', '1972-02-01', '1972-03-01',
```

```
              '1972-04-01', '1972-05-01', '1972-06-01', '1972-07-01',
              '1972-08-01', '1972-09-01'],
             dtype='datetime64[ns]', name='Month', length=105, freq=None)
```

10 Print rows from 1965-07-01 to 1965-12-01

[8]:
```
# hint lecture cell 7
champagne['1965-07-01':'1965-12-01']
```

[8]:
```
            champagne
Month
1965-07-01       3028
1965-08-01       1759
1965-09-01       3595
1965-10-01       4474
1965-11-01       6838
1965-12-01       8357
```

11 Print from begining of data till 1966-07-01

[9]:
```
# hint lecture cell 8
champagne[:'1966-07-01']
```

[9]:
```
            champagne
Month
1964-01-01       2815
1964-02-01       2672
1964-03-01       2755
1964-04-01       2721
1964-05-01       2946
1964-06-01       3036
1964-07-01       2282
1964-08-01       2212
1964-09-01       2922
1964-10-01       4301
1964-11-01       5764
1964-12-01       7312
1965-01-01       2541
1965-02-01       2475
1965-03-01       3031
1965-04-01       3266
1965-05-01       3776
1965-06-01       3230
1965-07-01       3028
1965-08-01       1759
1965-09-01       3595
```

```
1965-10-01        4474
1965-11-01        6838
1965-12-01        8357
1966-01-01        3113
1966-02-01        3006
1966-03-01        4047
1966-04-01        3523
1966-05-01        3937
1966-06-01        3986
1966-07-01        3260
```

12 Print data for the entire year 1972

[10]:
```
# hint lecture cell 9
champagne['1972']
```

```
<ipython-input-10-2787fdfe0c3c>:2: FutureWarning: Indexing a DataFrame with a
datetimelike index using a single string to slice the rows, like
`frame[string]`, is deprecated and will be removed in a future version. Use
`frame.loc[string]` instead.
  champagne['1972']
```

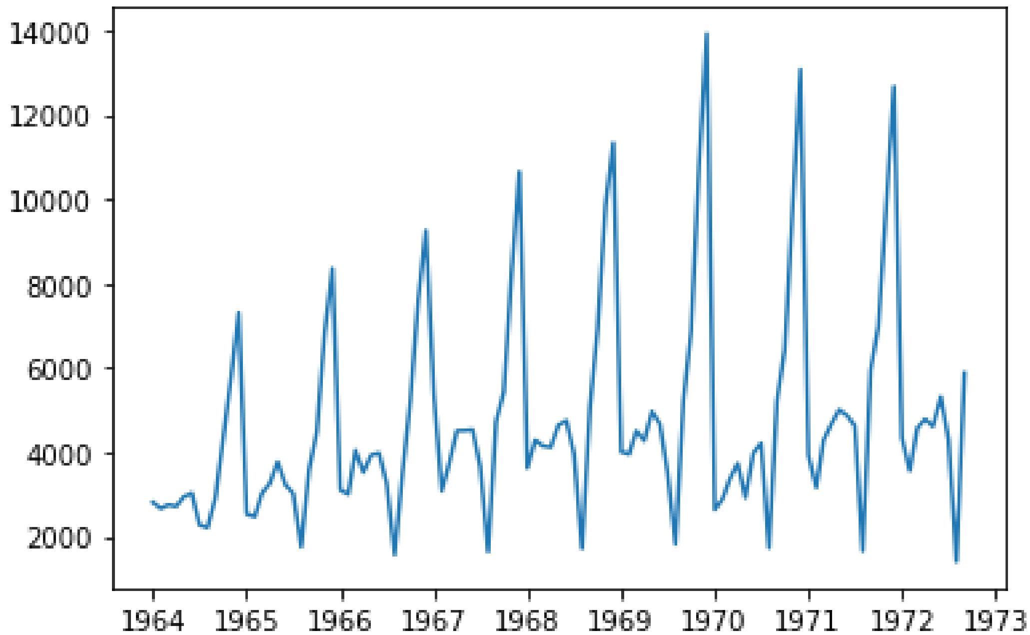
[10]:

	champagne
Month	
1972-01-01	4348
1972-02-01	3564
1972-03-01	4577
1972-04-01	4788
1972-05-01	4618
1972-06-01	5312
1972-07-01	4298
1972-08-01	1413
1972-09-01	5877

13 1. Plot champagne Time Series

[11]:
```
# hint lecture cell 10
%matplotlib inline
plt.plot(champagne)
```

[11]:
```
[<matplotlib.lines.Line2D at 0x1927125c820>]
```



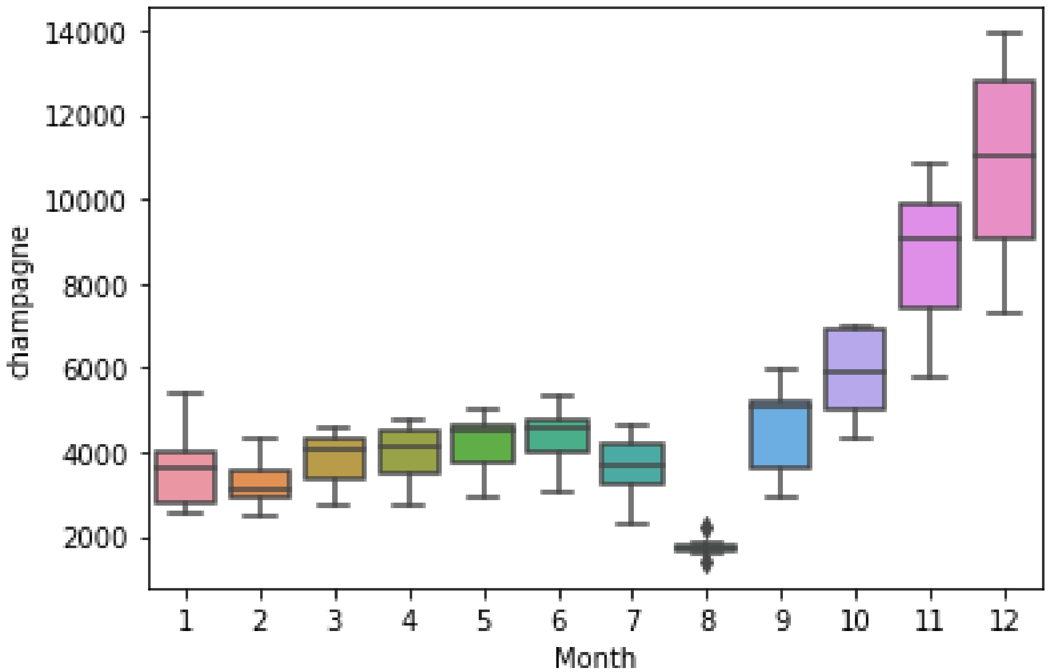
14 Create a column called 'Month' that is just the month of sale

[13]:
```
# hint lecture cell 11
champagne['Month'] = champagne.index.month
champagne.head()
```

[13]:
```
            champagne  Month
Month
1964-01-01       2815      1
1964-02-01       2672      2
1964-03-01       2755      3
1964-04-01       2721      4
1964-05-01       2946      5
```

15 Box plot of monthly champagne sale

[16]:
```
# hint lecture cell 12
import seaborn as sns
ax = sns.boxplot(data = champagne, x='Month', y='champagne')
```

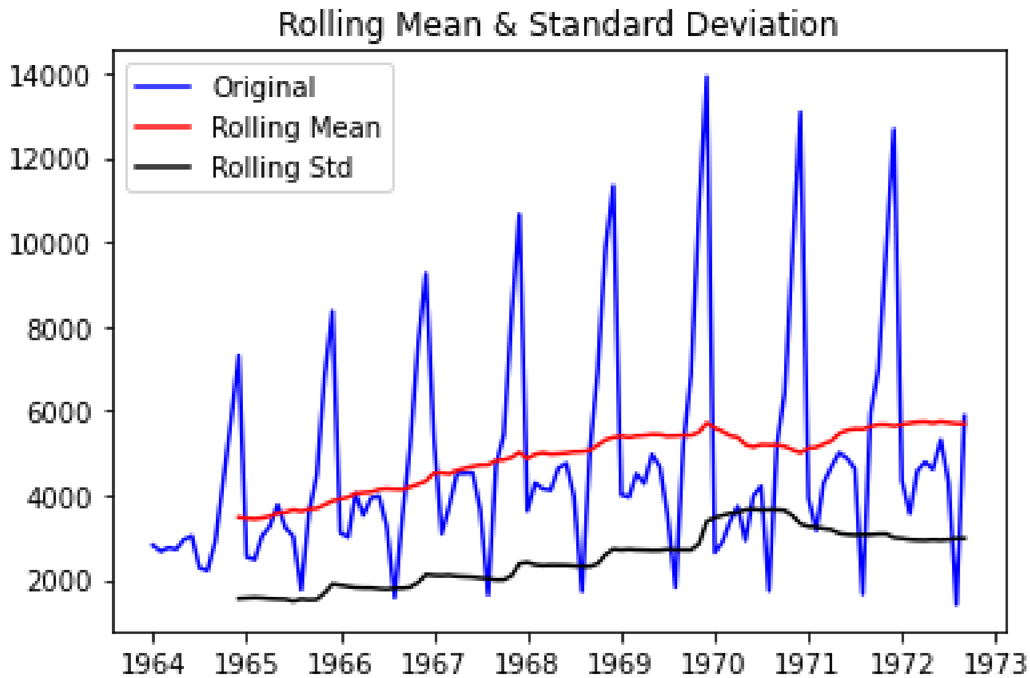


16 2. Stationarity - Check

[17]:
```
def test_stationarity(timeseries):

    #Determing rolling statistics
    rolmean = timeseries.rolling(window=12).mean()
    rolstd = timeseries.rolling(window=12).std()

    #Plot rolling statistics:
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)
```

17 Perform test_stationarity on champagne sale

[18]:
```
test_stationarity(champagne['champagne'])
```



[19]:
```
from statsmodels.tsa.stattools import adfuller

#Perform Dickey-Fuller test:
def test_Dickey_Fuller(timeseries):
    print('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    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)
```

18 Perform test_Dickey_Fuller on champagne sale

[20]:
```
test_Dickey_Fuller(champagne['champagne'])
```

```
Results of Dickey-Fuller Test:
Test Statistic                 -1.833593
p-value                         0.363916
#Lags Used                     11.000000
Number of Observations Used    93.000000
Critical Value (1%)            -3.502705
Critical Value (5%)            -2.893158
Critical Value (10%)           -2.583637
```

```
dtype: float64
```

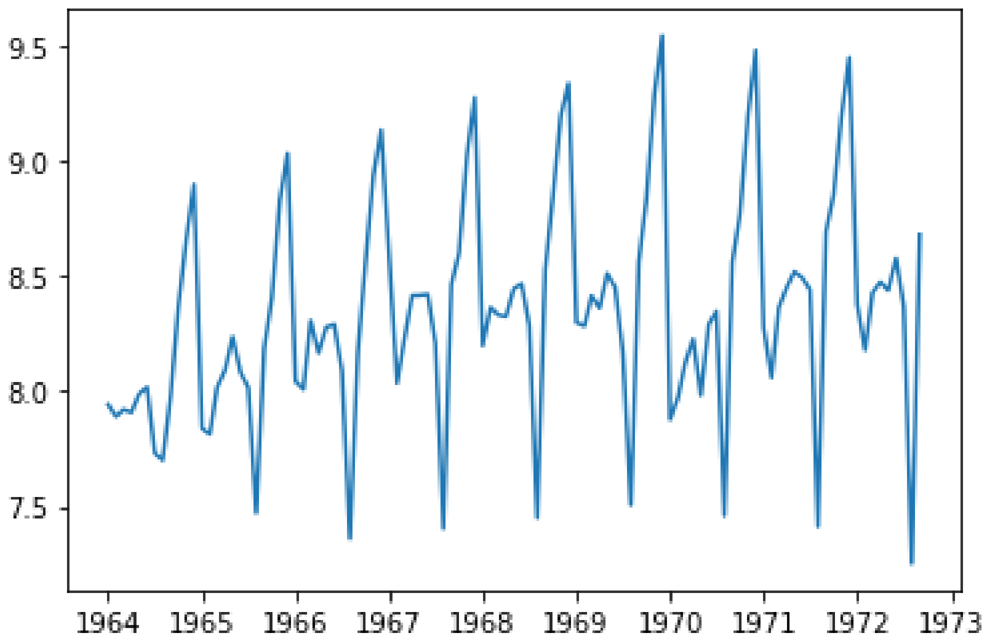
19 Make Time Series Stationary

20 Decomposing

21 get log of champagne sales and print the log time series (ts_log)

[22]:
```python
# hint lecture cell 17
ts_log = np.log(champagne['champagne'])
plt.plot(ts_log)
```

[22]:
```
[<matplotlib.lines.Line2D at 0x19274076100>]
```



22 Decompose log of champagne sales to obtain trend, seasonal, residual

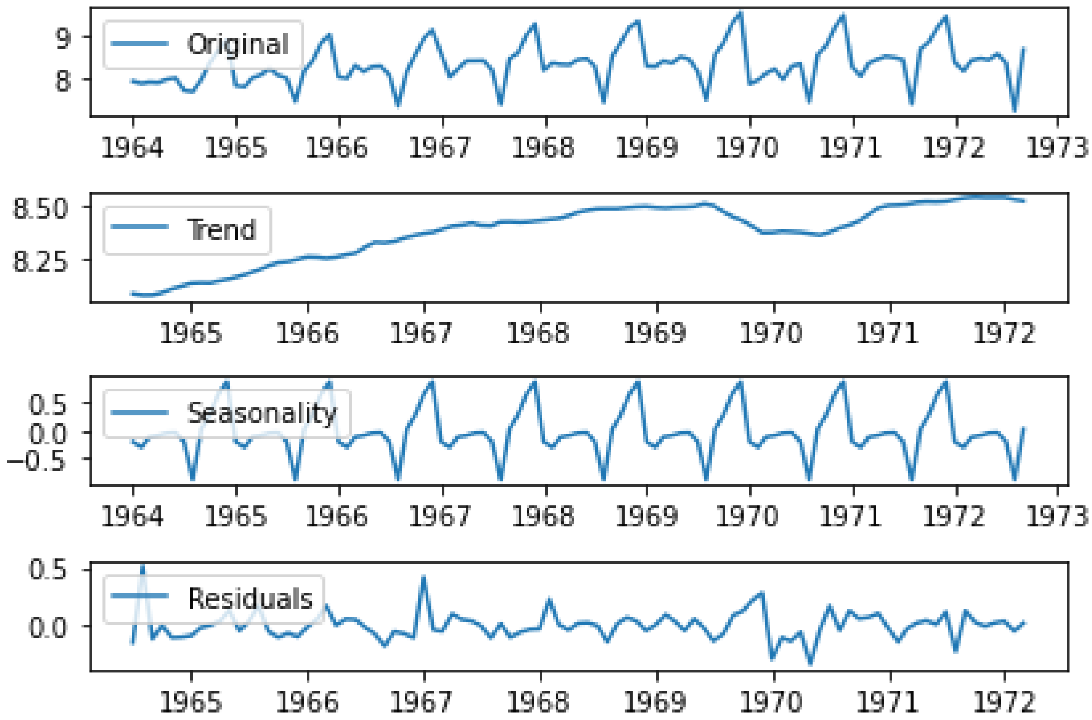
23 plot 'Original' (ts_log)

24 plot trend

25 plot seasonality

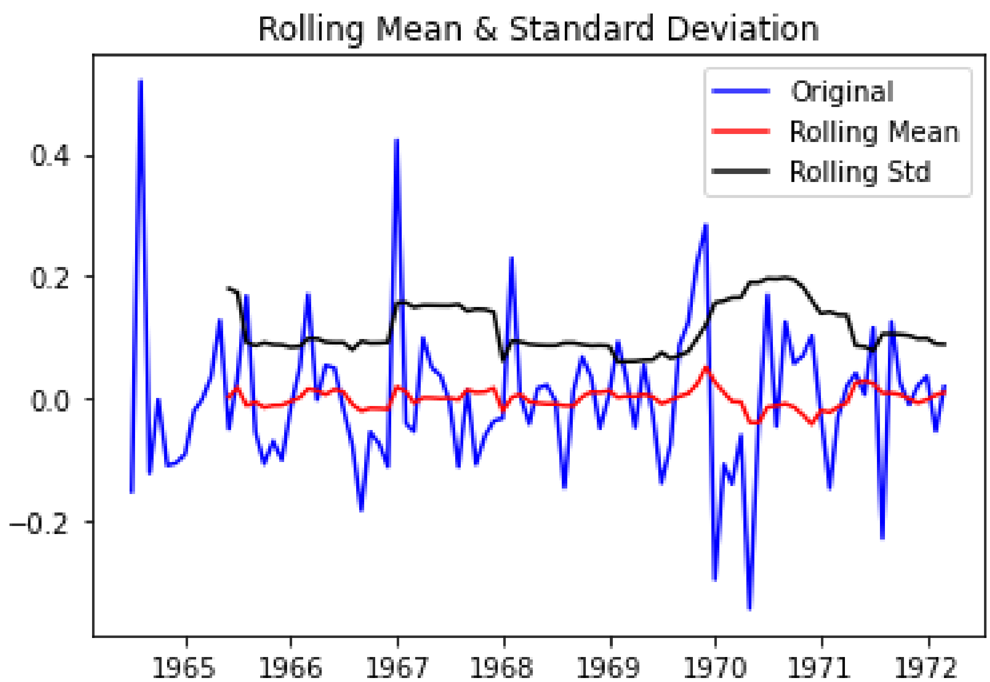
26 plot residuals

[28]:
```
# hint lecture cell 18
from statsmodels.tsa.seasonal import seasonal_decompose

decomposition = seasonal_decompose(ts_log)
trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

plt.subplot(411)
plt.plot(ts_log, label='Original')
plt.legend(loc='best')
plt.subplot(412)
plt.plot(trend, label='Trend')
plt.legend(loc='best')
plt.subplot(413)
plt.plot(seasonal,label='Seasonality')
plt.legend(loc='best')
plt.subplot(414)
plt.plot(residual, label='Residuals')
plt.legend(loc='best')
plt.tight_layout()
```



27 Perform test_stationarity on residual of champagne sale

[29]:
```
# hint lecture cell 19
#use only residual data
ts_log_decompose = residual
ts_log_decompose.dropna(inplace=True)
test_stationarity(ts_log_decompose)
```



28 Perform test_Dickey_Fuller on residual of champagne sale

[30]:
```
test_Dickey_Fuller(ts_log_decompose)
```

```
Results of Dickey-Fuller Test:
Test Statistic                -6.275488e+00
p-value                        3.910002e-08
#Lags Used                     7.000000e+00
Number of Observations Used    8.500000e+01
Critical Value (1%)           -3.509736e+00
Critical Value (5%)           -2.896195e+00
Critical Value (10%)          -2.585258e+00
dtype: float64
```

29 Plot ACF & PACF chart & find optimal parameter

[31]:
```
from statsmodels.tsa.stattools import acf, pacf
```

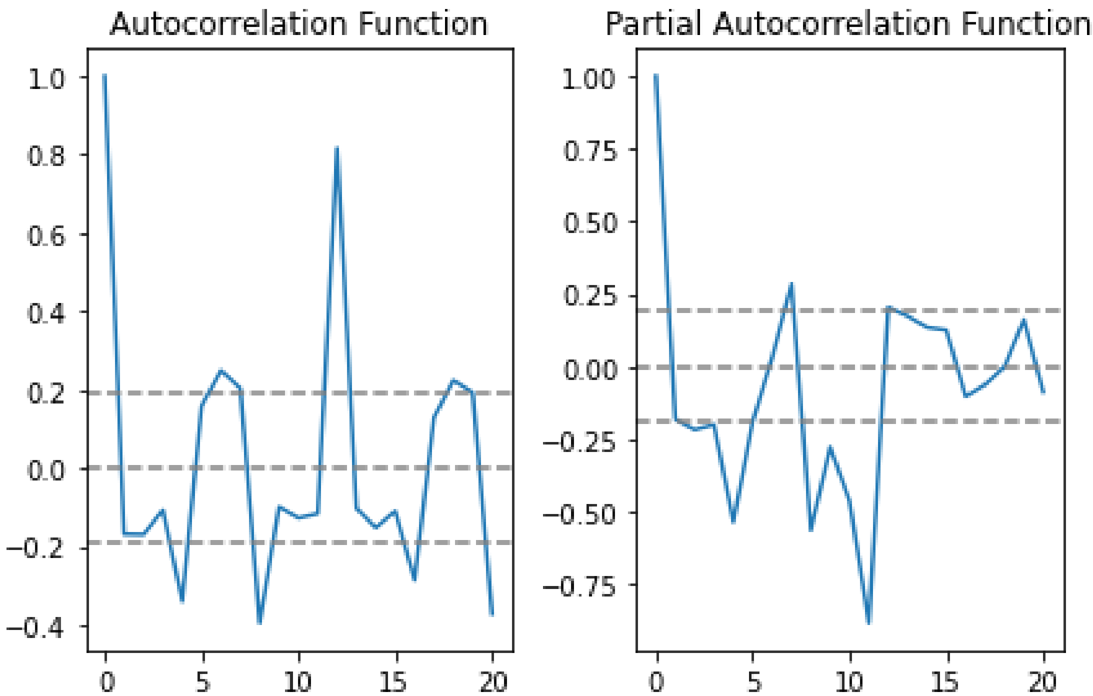
30 Obtain partical autocorrelation (pacf) and autocorrelation (acf)

[32]:
```
# hint lecture cell 22
ts_log_diff = ts_log - ts_log.shift()
ts_log_diff.dropna(inplace=True)
lag_acf = acf(ts_log_diff, nlags=20)
lag_pacf = pacf(ts_log_diff, nlags=20, method='ols')
```

```
C:\Users\Admin\anaconda3\lib\site-packages\statsmodels\tsa\stattools.py:667:
FutureWarning: fft=True will become the default after the release of the 0.12
release of statsmodels. To suppress this warning, explicitly set fft=False.
  warnings.warn(
```

31 Plot partical autocorrelation (pacf) and autocorrelation (acf)

[33]:
```
# hint lecture cell 23
#Plot ACF:
plt.subplot(121)
plt.plot(lag_acf)
plt.axhline(y=0,linestyle='--',color='gray')
plt.axhline(y=-1.96/np.sqrt(len(ts_log_diff)),linestyle='--',color='gray')
plt.axhline(y=1.96/np.sqrt(len(ts_log_diff)),linestyle='--',color='gray')
plt.title('Autocorrelation Function')

#Plot PACF:
plt.subplot(122)
plt.plot(lag_pacf)
plt.axhline(y=0,linestyle='--',color='gray')
plt.axhline(y=-1.96/np.sqrt(len(ts_log_diff)),linestyle='--',color='gray')
plt.axhline(y=1.96/np.sqrt(len(ts_log_diff)),linestyle='--',color='gray')
plt.title('Partial Autocorrelation Function')
plt.tight_layout()
```



[34]:
```python
from statsmodels.tsa.arima_model import ARIMA
```

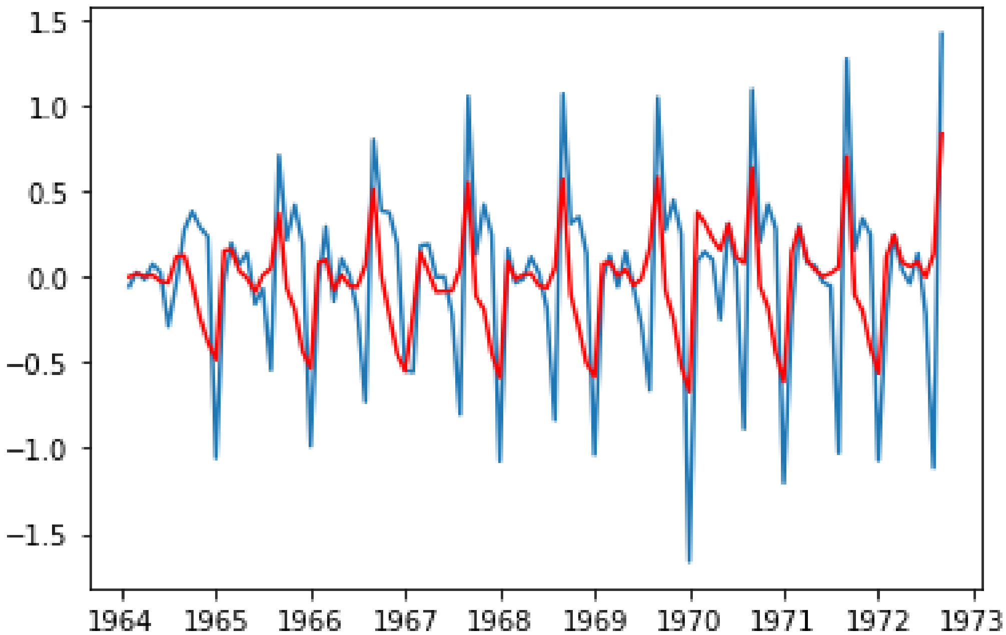
32 Build ARIMA model using ts_log using p and q values from acf and pacf

[36]:
```python
# hint lecture cell 25
#ARIMA
model = ARIMA(ts_log, order=(1, 1, 1))  #(p,d,q)
results_ARIMA = model.fit(disp=-1)
plt.plot(ts_log_diff)
plt.plot(results_ARIMA.fittedvalues, color='red')
```

```
C:\Users\Admin\anaconda3\lib\site-packages\statsmodels\tsa\arima_model.py:472:
FutureWarning:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
```

```
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
C:\Users\Admin\anaconda3\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:524: ValueWarning: No frequency
information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
C:\Users\Admin\anaconda3\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:524: ValueWarning: No frequency
information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
C:\Users\Admin\anaconda3\lib\site-packages\statsmodels\tsa\arima_model.py:472:
FutureWarning:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
```

```
[36]: [<matplotlib.lines.Line2D at 0x192746b88b0>]
```



33 Make predictions

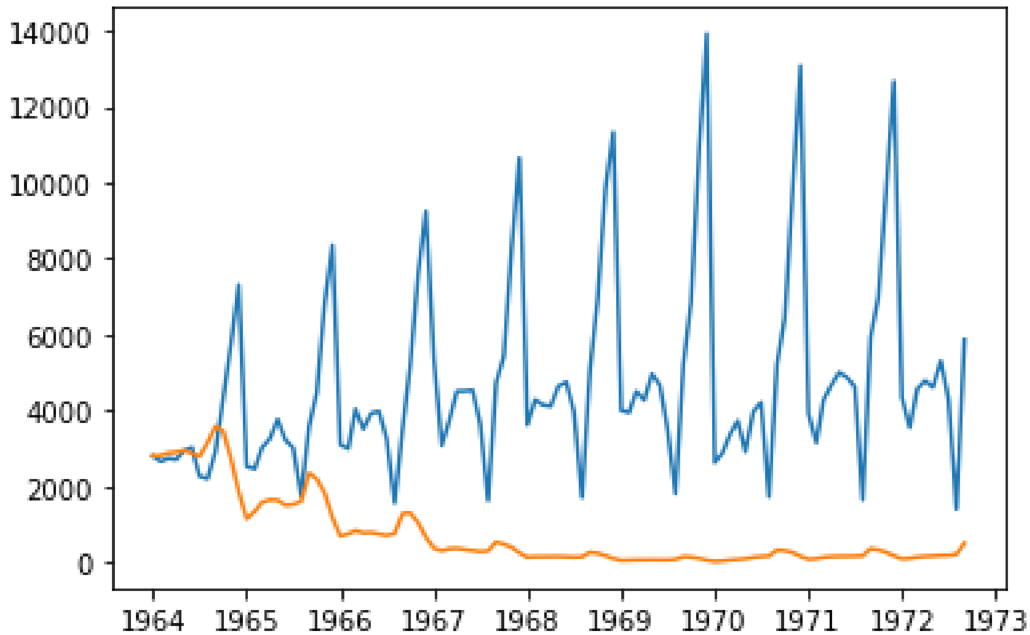
[37]:
```
predictions_ARIMA_diff = pd.Series(results_ARIMA.fittedvalues, copy=True)
```

[38]:
```
predictions_ARIMA_diff_cumsum = predictions_ARIMA_diff.cumsum()
```

[43]:
```
predictions_ARIMA_log = pd.Series(ts_log.iloc[0], index=ts_log.index)
predictions_ARIMA_log = predictions_ARIMA_log.
 ↪add(predictions_ARIMA_diff_cumsum,fill_value=0)
```

[44]:
```
predictions_ARIMA = np.exp(predictions_ARIMA_log)
plt.plot(champagne['champagne'])
plt.plot(predictions_ARIMA)
```

[44]: [<matplotlib.lines.Line2D at 0x192748214f0>]


14000
12000
10000
8000
6000
4000
2000
0
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973