GOKHALE INSTITUTE OF POLITICS AND ECONOMICS

Assignment-3, Advanced Econometrics

Author: Balaji Bahirwal

Reg no. AE2007


```python
import warnings
import itertools
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.tsa.api as smt
import statsmodels.formula.api as smf
from io import StringIO
import requests
import matplotlib.pyplot as plt
%matplotlib inline

plt.style.use('bmh')
```

Analysis of Rural & Urban wages

```python
wages = pd.read_excel("C:\\Users\\bahirwal\\Desktop\\Balaji\\Advanced ECotrix\\Task-3\\Wages_rural & urban.xlsx
wages.head()
```

	Date	Rural	Urban
0	2013-01-01	105.1	104.0
1	2013-02-01	105.8	104.7
2	2013-03-01	106.0	105.0
3	2013-04-01	106.4	105.7
4	2013-05-01	107.2	106.6

```python
wages['Date'] = pd.to_datetime(wages['Date'])
```

```python
wages.set_index('Date', inplace=True)
```

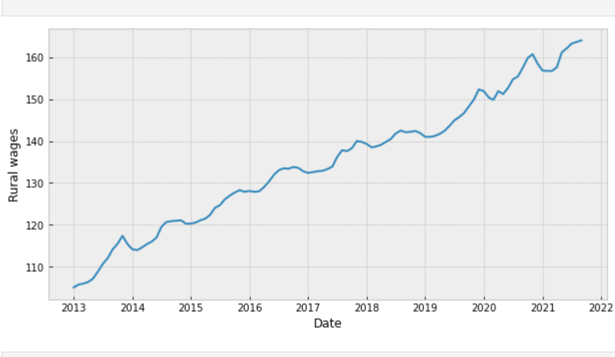
```python
wages.head()
```

Date	Rural	Urban
2013-01-01	105.1	104.0
2013-02-01	105.8	104.7
2013-03-01	106.0	105.0
2013-04-01	106.4	105.7
2013-05-01	107.2	106.6

For Rural

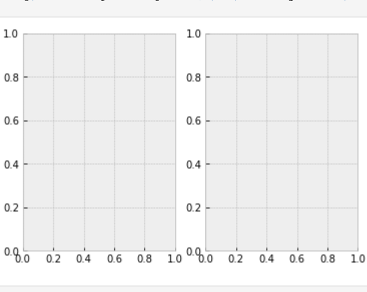
```python
# extract out the time-series
wages_ts = wages['Rural']
```

```python
# Plot the time series
plt.figure(figsize=(10,5))
plt.plot(wages_ts)
plt.xlabel('Date')
plt.ylabel('Rural wages');
```

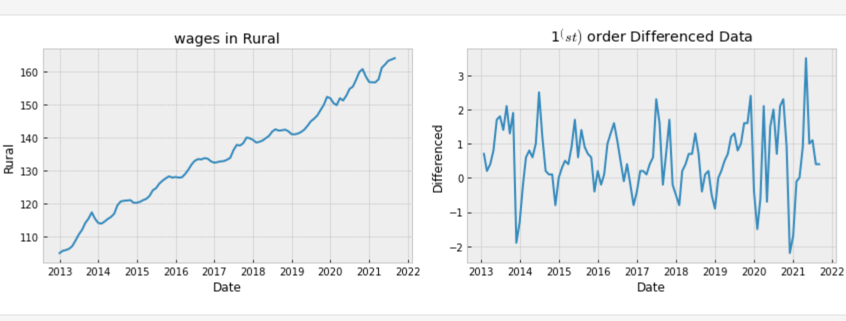



```python
random_df = wages
```

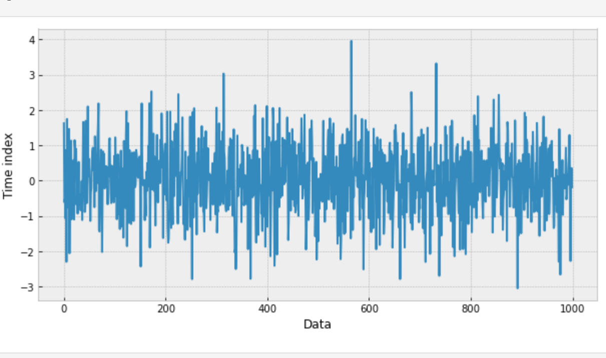
```python
fig, axes = plt.subplots(1, 2, sharey=False, sharex=False)
```



```python
fig, axes = plt.subplots(1, 2, sharey=False, sharex=False)
fig.set_figwidth(12)
fig.set_figheight(4)
axes[0].plot(random_df.index, random_df['Rural'])
axes[0].set_xlabel('Date')
axes[0].set_ylabel('Rural')
axes[0].set_title('wages in Rural')
axes[1].plot(random_df.index, random_df['Rural'].diff(periods=1))
axes[1].set_xlabel('Date')
axes[1].set_ylabel('Differenced')
axes[1].set_title('1$^(st)$ order Differenced Data')
plt.tight_layout()
plt.show()
```

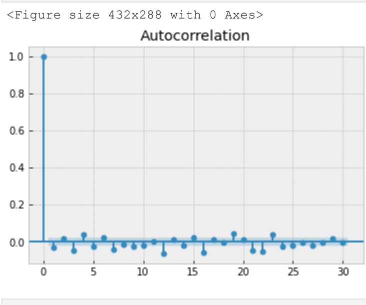



```python
# Generate white noise
np.random.seed(1)

# Plot of discrete white noise
plt.figure(figsize=(10, 5))
white_noise = np.random.normal(size = 1000)
plt.plot(white_noise)
plt.xlabel('Data')
plt.ylabel('Time index')
plt.show()
```



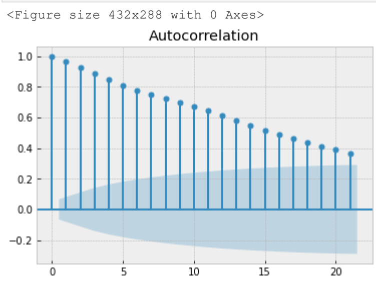

```python
# there is no autocorrelation as it is a noise a no pattern can be drawn
plt.figure()
smt.graphics.plot_acf(white_noise, alpha=0.5)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

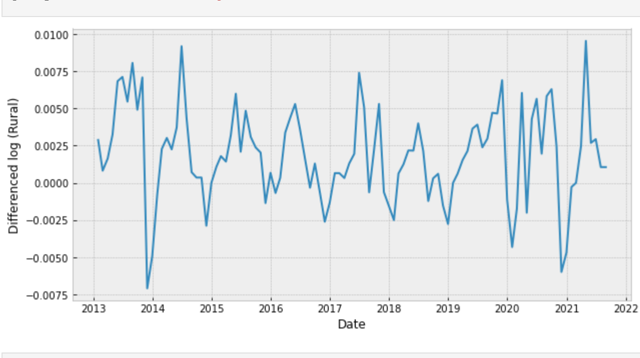



```python
# plot autocorrelation for rural wages data
plt.figure()
smt.graphics.plot_acf(wages_ts, alpha=0.5)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

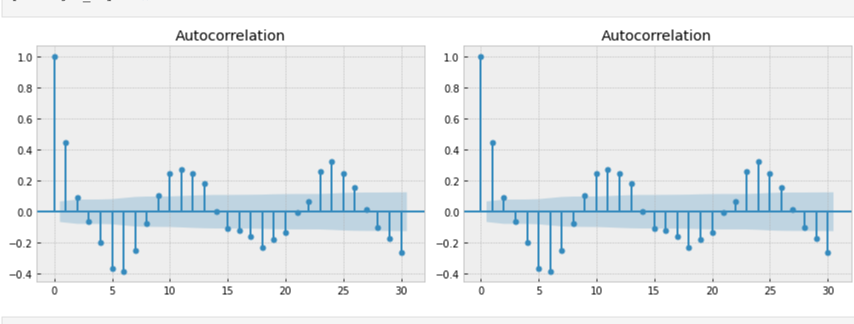



```python
# when variance is not stationary we took log transformation to find differentiation
# Now time series become stationary by making mean and variance constant
plt.figure(figsize=(10, 5))
plt.plot(np.log10(wages_ts).diff(periods=1))
plt.xlabel('Date')
plt.ylabel('Differenced log (Rural)');
```

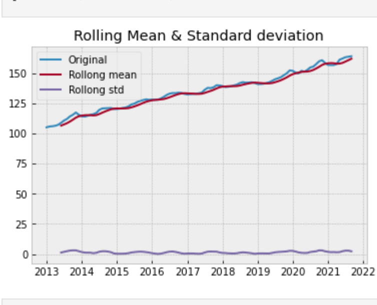



```python
wages_ts_log = np.log10(wages_ts)
wages_ts_log.dropna(inplace=True)

wages_ts_log_diff = wages_ts_log.diff(periods=1)
wages_ts_log_diff.dropna(inplace=True)
```

```python
# ACF plot
fig, axes = plt.subplots(1, 2, sharey=False, sharex=False)
fig.set_figwidth(12)
fig.set_figheight(4)
smt.graphics.plot_acf(wages_ts_log_diff, lags=30, ax=axes[0], alpha=0.5)
smt.graphics.plot_acf(wages_ts_log_diff, lags=30, ax=axes[1], alpha=0.5)
plt.tight_layout()
```

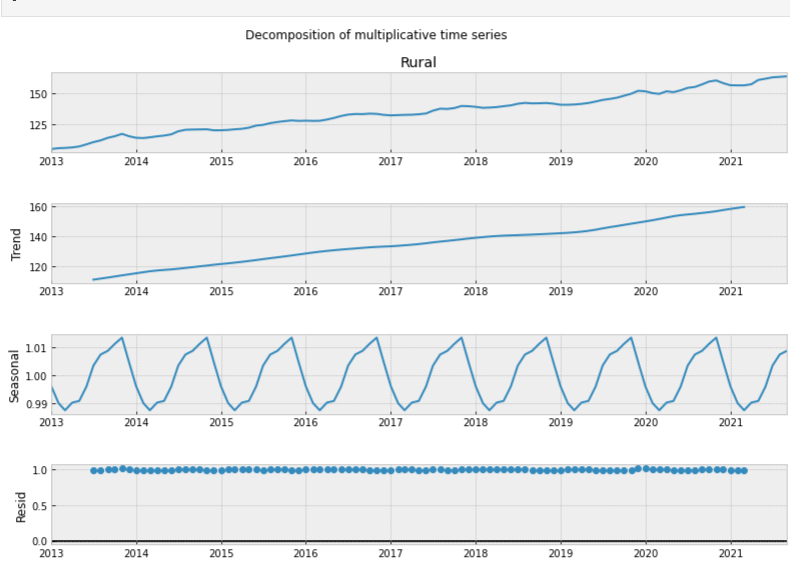



```python
# Determing rolling statistics
rolmean = wages_ts.rolling(window=6).mean()
rolstd = wages_ts.rolling(window=6).std()

# plot rolling statistics
orig = plt.plot(wages_ts, label='Original')
mean = plt.plot(rolmean, label='Rollong mean')
std = plt.plot(rolstd, label='Rollong std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard deviation')
plt.show(block=False)
```




```python
decomposition = sm.tsa.seasonal_decompose(wages_ts, model = 'multiplicative')
```

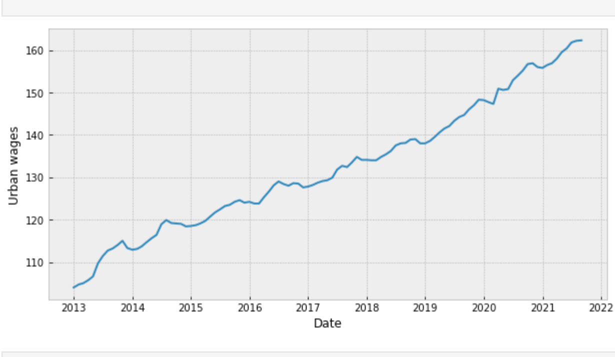
```python
fig = decomposition.plot()
fig.set_figwidth(12)
fig.set_figheight(8)
fig.suptitle('Decomposition of multiplicative time series')
plt.show()
```



Wages Urban

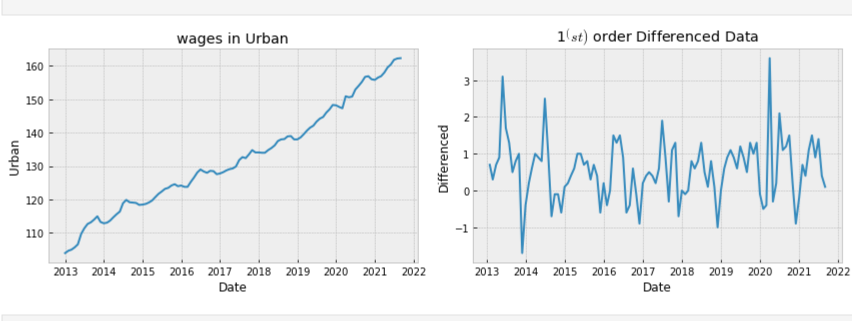
```python
# extract out the time-series
wages_ts_ur = wages['Urban']
```

```python
# Plot the time series
plt.figure(figsize=(10,5))
plt.plot(wages_ts_ur)
plt.xlabel('Date')
plt.ylabel('Urban wages');
```



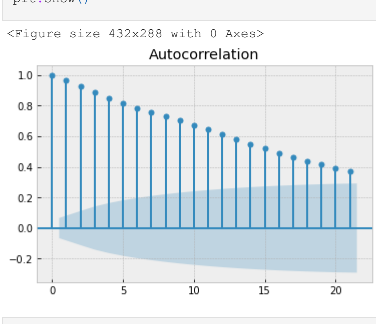

```python
random_df = wages
```

```python
# First order differenced data to make mean and variance constant and converting time series into stationary
fig, axes = plt.subplots(1, 2, sharey=False, sharex=False)
fig.set_figwidth(12)
fig.set_figheight(4)
axes[0].plot(random_df.index, random_df['Urban'])
axes[0].set_xlabel('Date')
axes[0].set_ylabel('Urban')
axes[0].set_title('wages in Urban')
axes[1].plot(random_df.index, random_df['Urban'].diff(periods=1))
axes[1].set_xlabel('Date')
axes[1].set_ylabel('Differenced')
axes[1].set_title('1$^(st)$ order Differenced Data')
plt.tight_layout()
plt.show()
```

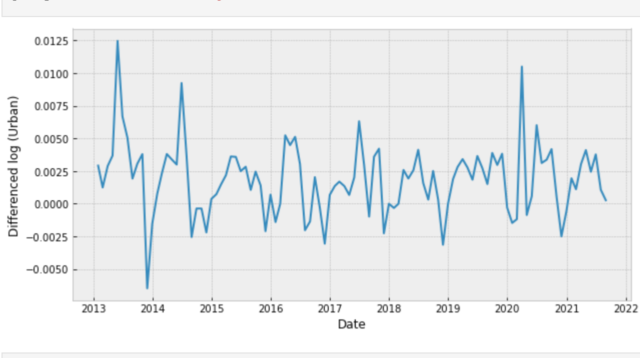



```python
# plot autocorrelation for uban wages data
plt.figure()
smt.graphics.plot_acf(wages_ts_ur, alpha=0.5)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

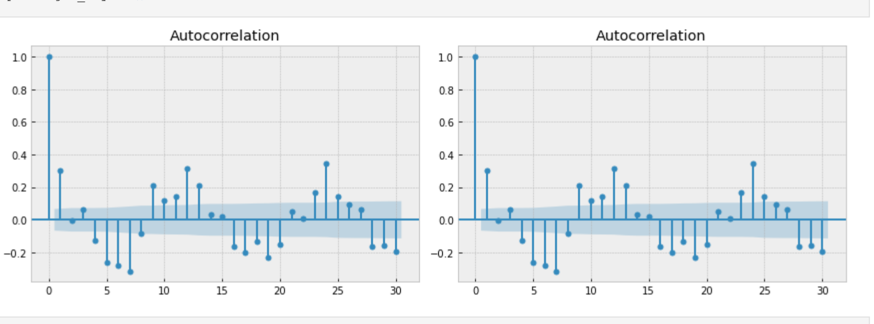



```python
# we will also try making time series stationary by taking its log transformation
plt.figure(figsize=(10, 5))
plt.plot(np.log10(wages_ts_ur).diff(periods=1))
plt.xlabel('Date')
plt.ylabel('Differenced log (Urban)');
```

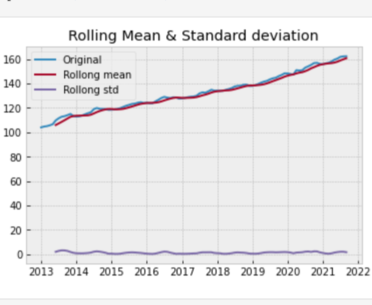



```python
wages_ts_log_ur = np.log10(wages_ts_ur)
wages_ts_log_ur.dropna(inplace=True)

wages_ts_log_diff_ur = wages_ts_log_ur.diff(periods=1)
wages_ts_log_diff_ur.dropna(inplace=True)
```

```python
# ACF plot
fig, axes = plt.subplots(1, 2, sharey=False, sharex=False)
fig.set_figwidth(12)
fig.set_figheight(4)
smt.graphics.plot_acf(wages_ts_log_diff_ur, lags=30, ax=axes[0], alpha=0.5)
smt.graphics.plot_acf(wages_ts_log_diff_ur, lags=30, ax=axes[1], alpha=0.5)
plt.tight_layout()
```

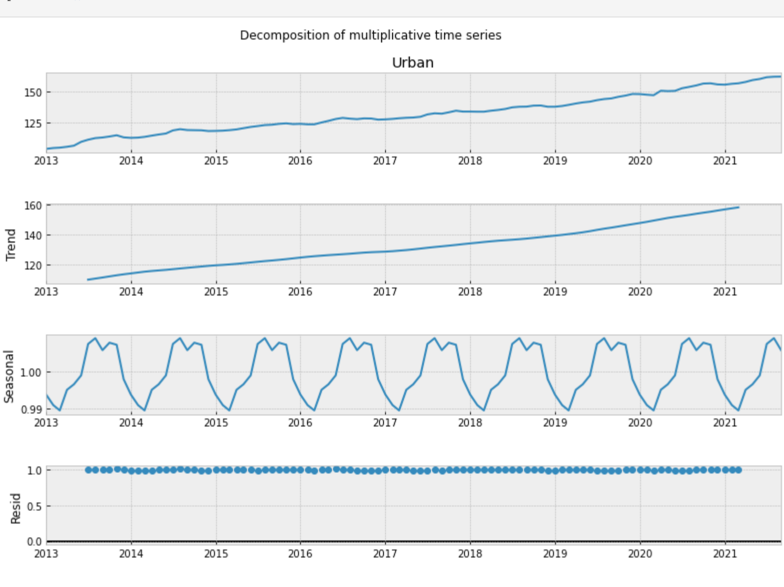



```python
# Determing rolling statistics
rolmean_ur = wages_ts_ur.rolling(window=6).mean()
rolstd_ur = wages_ts_ur.rolling(window=6).std()

# plot rolling statistics
orig = plt.plot(wages_ts_ur, label='Original')
mean = plt.plot(rolmean_ur, label='Rollong mean')
std = plt.plot(rolstd_ur, label='Rollong std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard deviation')
plt.show(block=False)
```




```python
decomposition_ur = sm.tsa.seasonal_decompose(wages_ts_ur, model = 'multiplicative')
```

```python
fig = decomposition_ur.plot()
fig.set_figwidth(12)
fig.set_figheight(8)
fig.suptitle('Decomposition of multiplicative time series')
plt.show()
```



We can see that urban wages show a increasing trend and also a seasonality as there wages are increasing from january to june-july and then decreasing to its original level at time of starting of year.

CPI & WPI

```python
WPI = pd.read_excel("C:\\Users\\bahirwal\\Desktop\\Balaji\\Advanced ECotrix\\Task-3\\WPI annually.xlsx")
```

```python
WPI.head()
```

	Year	wpi
0	1960	3.068381
1	1961	3.141976
2	1962	3.255201
3	1963	3.374086
4	1964	3.725082

```python
WPI.tail()
```

	Year	wpi
56	2016	122.940363
57	2017	127.167193
58	2018	132.592399
59	2019	135.091338
60	2020	146.345357

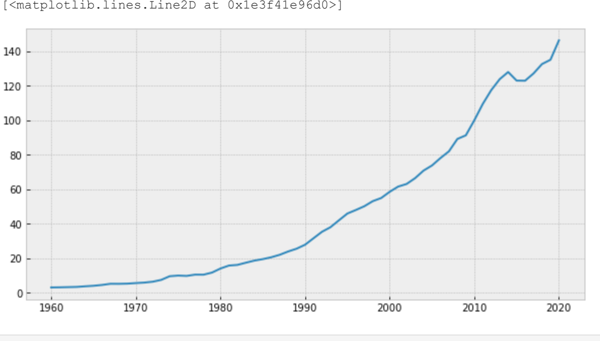
```python
WPI.set_index('Year', inplace=True)
```

```python
WPI.head()
```

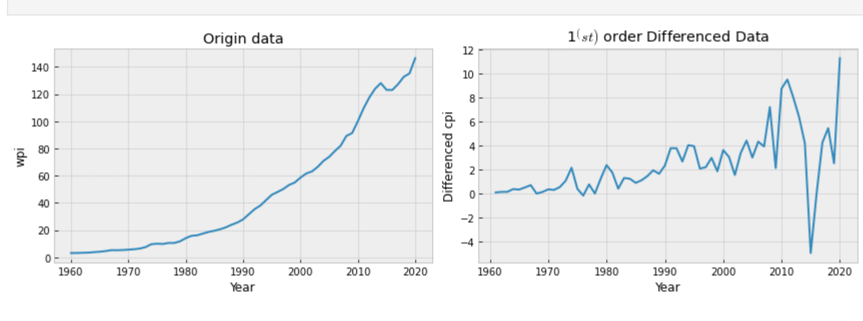
Year	wpi
1960	3.068381
1961	3.141976
1962	3.255201
1963	3.374086
1964	3.725082

```python
plt.figure(figsize=(10,5))
plt.plot(WPI)
```

```
[<matplotlib.lines.Line2D at 0x1e3f41e96d0>]
```



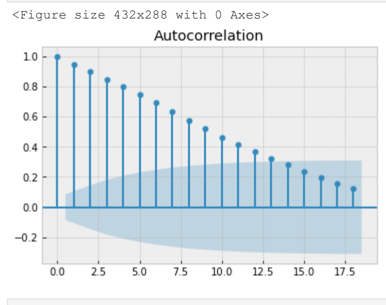
```python
random_df = WPI
fig, axes = plt.subplots(1, 2, sharey=False, sharex=False)
fig.set_figwidth(12)
fig.set_figheight(4)
axes[0].plot(random_df.index, random_df['wpi'])
axes[0].set_xlabel('Year')
axes[0].set_ylabel('wpi')
axes[0].set_title('Origin data')
axes[1].plot(random_df.index, random_df['wpi'].diff(periods=1))
axes[1].set_xlabel('Year')
axes[1].set_ylabel('Differenced cpi')
axes[1].set_title('1$^(st)$ order Differenced Data')
plt.tight_layout()
plt.show()
```



wholesale price index shows a continuous increase from 1960 till 2020. the time series is not stationary and trend is clearly visible. 1st order differenced data is also not stationary and so we have to go for log transformation to make time series stationary.

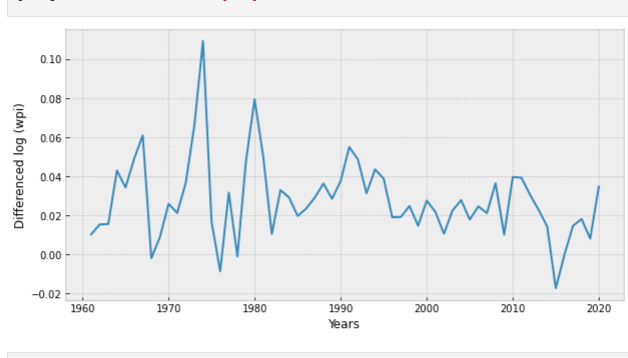
```python
# plot autocorrelation for cpi data
plt.figure()
smt.graphics.plot_acf(random_df, alpha=0.5)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

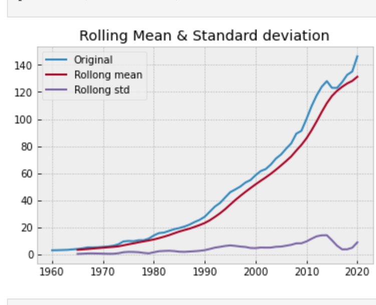



```python
# extract out the time-series
WPI_ts = WPI['wpi']
```

```python
# when variance is not stationary we took log transformation to find differentiation
# Now time series become stationary by making mean and variance constant
plt.figure(figsize=(10, 5))
plt.plot(np.log10(WPI_ts).diff(periods=1))
plt.xlabel('Years')
plt.ylabel('Differenced log (wpi)');
```




```python
# Determing rolling statistics for both cpi and wpi
rolmean = random_df.rolling(window=6).mean()
rolstd = random_df.rolling(window=6).std()

# plot rolling statistics
orig = plt.plot(random_df, label='Original')
mean = plt.plot(rolmean, label='Rollong mean')
std = plt.plot(rolstd, label='Rollong std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard deviation')
plt.show(block=False)
```




```python
decomposition_WPI = sm.tsa.seasonal_decompose(WPI_ts, model = 'multiplicative')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-133-23119d2b2855> in <module>
----> 1 decomposition_WPI = sm.tsa.seasonal_decompose(WPI_ts, model = 'multiplicative')

~\Anaconda3\lib\site-packages\pandas\util\_decorators.py in wrapper(*args, **kwargs)
    197                 else:
    198                     kwargs[new_arg_name] = new_arg_value
--> 199             return func(*args, **kwargs)
    200
    201         return cast(F, wrapper)

~\Anaconda3\lib\site-packages\statsmodels\tsa\seasonal.py in seasonal_decompose(x, model, filt, period, two_sided, extrapolate_trend)
    141             period = pfreq
    142         else:
--> 143             raise ValueError("You must specify a period or x must be a "
    144                              "pandas object with a DatetimeIndex with "
    145                              "a freq not set to None")

ValueError: You must specify a period or x must be a pandas object with a DatetimeIndex with a freq not set to None
```

```python
fig = decomposition_ur.plot()
fig.set_figwidth(12)
fig.set_figheight(8)
fig.suptitle('Decomposition of multiplicative time series')
plt.show()
```

```python
from statsmodels.tsa.stattools import adfuller
X = WPI.values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: 2.021913
p-value: 0.998701
Critical Values:
	1%: -3.571
	5%: -2.923
	10%: -2.599
```

As result of ....test, with p-value 0.998701 indicates that wpi data is non stationary

In [57]:
```python
def adfuller_test(WPI):
    result = adfuller(wpi)
    labels = ['ADF Test Statistics', 'P-value', '#Lags used', 'Number of Observations Used']
    for value, label in zip(result, labels):
        print(label+' : '+str(value) )
    if result[1] <= 0.05:
        print("Strong evidence against the null hypothesis(HO), reject the null hypothesis, Data has no unit ro
    else :
        print("Weak evidence aginst null hypothesis, time series has unit root, indicating it is non-stationary
```

In [58]:
```python
test_result_1 = adfuller_test(WPI['wpi'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-58-40f8c61a4e41> in <module>
----> 1 test_result_1 = adfuller_test(WPI['wpi'])

<ipython-input-57-b0156a370839> in adfuller_test(WPI)
      1 def adfuller_test(WPI):
----> 2     result = adfuller(wpi)
      3     labels = ['ADF Test Statistics', 'P-value', '#Lags used', 'Number of Observations Used']
      4     for value, label in zip(result, labels):
      5         print(label+' : '+str(value) )

NameError: name 'wpi' is not defined
```

In [ ]:
```python
adfuller_test(inflation['cpi'])
```

In [ ]:

CPI

In [59]:
```python
CPI = pd.read_excel("C:\\Users\\bahirwal\\Desktop\\Balaji\\Advanced ECotrix\\Task-3\\CPI annually.xlsx")
```

In [60]:
```python
CPI
```

Out[60]:

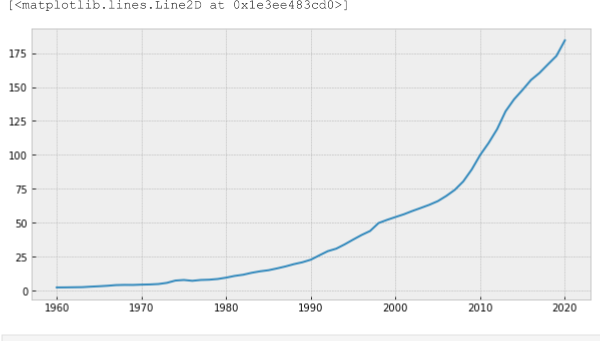
	Year	cpi
0	1960	2.527487
1	1961	2.570333
2	1962	2.663693
3	1963	2.742170
4	1964	3.108394
...	...	...
56	2016	155.182121
57	2017	160.346852
58	2018	166.672645
59	2019	172.878328
60	2020	184.328815

61 rows × 2 columns

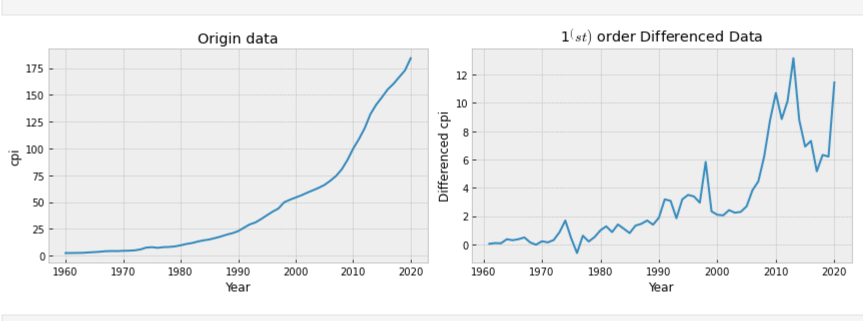
In [61]:
```python
CPI.set_index('Year', inplace=True)
```

In [62]:
```python
plt.figure(figsize=(10,5))
plt.plot(CPI)
```

Out[62]: [<matplotlib.lines.Line2D at 0x1e3ee483cd0>]

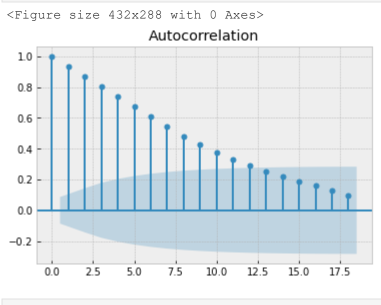


In [63]:
```python
random_df_1 = CPI
fig, axes = plt.subplots(1, 2, sharey=False, sharex=False)
fig.set_figwidth(12)
fig.set_figheight(4)
axes[0].plot(random_df_1.index, random_df_1['cpi'])
axes[0].set_xlabel('Year')
axes[0].set_ylabel('cpi')
axes[0].set_title('Origin data')
axes[1].plot(random_df_1.index, random_df_1['cpi'].diff(periods=1))
axes[1].set_xlabel('Year')
axes[1].set_ylabel('Differenced cpi')
axes[1].set_title('1$^(st)$ order Differenced Data')
plt.tight_layout()
plt.show()
```



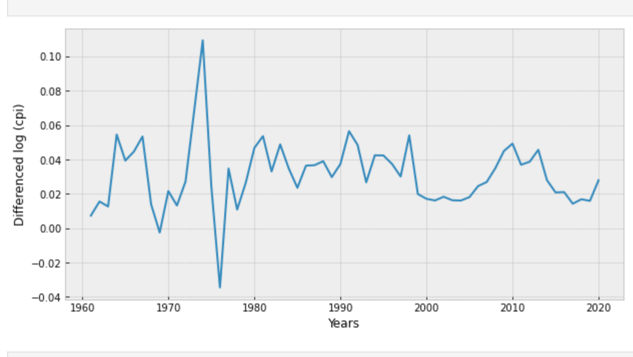
In [64]:
```python
# plot autocorrelation for cpi data
plt.figure()
smt.graphics.plot_acf(random_df_1, alpha=0.5)
plt.show()
```

<Figure size 432x288 with 0 Axes>

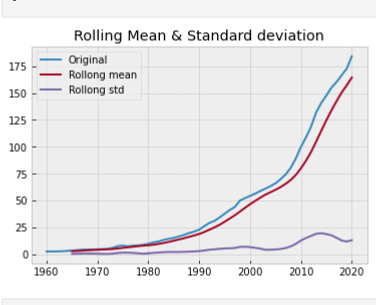


In [65]:
```python
# extract out the time-series
CPI_ts = CPI['cpi']
```

In [66]:
```python
# when variance is not stationary we took log transformation to find differentiation
# Now time series become stationary by making mean and variance constant
plt.figure(figsize=(10, 5))
plt.plot(np.log10(CPI_ts).diff(periods=1))
plt.xlabel('Years')
plt.ylabel('Differenced log (cpi)');
```



In [67]:
```python
# Determing rolling statistics for both cpi and wpi
rolmean_1 = random_df_1.rolling(window=6).mean()
rolstd_1 = random_df_1.rolling(window=6).std()

# plot rolling statistics
orig = plt.plot(random_df_1, label='Original')
mean = plt.plot(rolmean_1, label='Rollong mean')
std = plt.plot(rolstd_1, label='Rollong std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard deviation')
plt.show(block=False)
```



In [68]:
```python
from statsmodels.tsa.stattools import adfuller
X_1 = CPI.values
result = adfuller(X_1)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: 6.007442
p-value: 1.000000
Critical Values:
	1%: -3.558
	5%: -2.917
	10%: -2.596
```

In [ ]:

Combined Wages of rural and urban region

In [69]:
```python
WAGES = pd.read_excel("C:\\Users\\bahirwal\\Desktop\\Balaji\\Advanced ECotrix\\Task-3\\Book1.xlsx")
WAGES
```

Out[69]:

	Date	wages
0	2013-01-01	104.6
1	2013-02-01	105.3
2	2013-03-01	105.5
3	2013-04-01	106.1
4	2013-05-01	106.9
...	...	...
100	2021-05-01	160.4
101	2021-06-01	161.3
102	2021-07-01	162.5
103	2021-08-01	162.9
104	2021-09-01	163.2

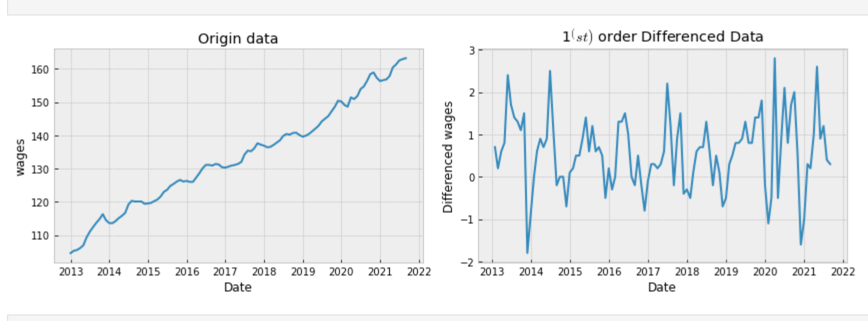
105 rows × 2 columns

In [70]:
```python
WAGES.set_index('Date', inplace=True)
```

In [71]:
```python
plt.figure(figsize=(10,5))
plt.plot(WAGES)
```

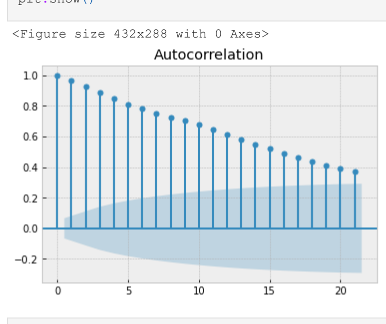
Out[71]: [<matplotlib.lines.Line2D at 0x1e3f40829d0>]

In [72]:
```python
random_df_2 = WAGES
fig, axes = plt.subplots(1, 2, sharey=False, sharex=False)
fig.set_figwidth(12)
fig.set_figheight(4)
axes[0].plot(random_df_2.index, random_df_2['wages'])
axes[0].set_xlabel('Date')
axes[0].set_ylabel('wages')
axes[0].set_title('Origin data')
axes[1].plot(random_df_2.index, random_df_2['wages'].diff(periods=1))
axes[1].set_xlabel('Date')
axes[1].set_ylabel('Differenced wages')
axes[1].set_title('1$^(st)$ order Differenced Data')
plt.tight_layout()
plt.show()
```



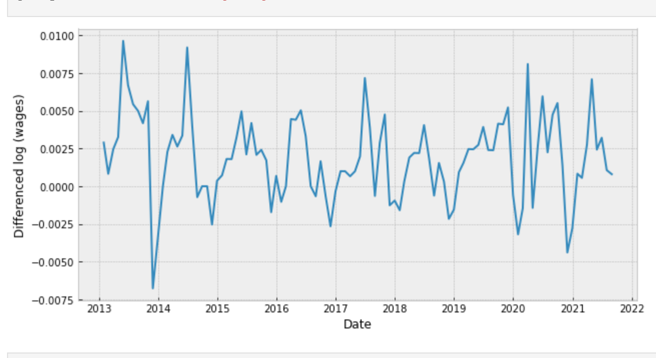
In [73]:
```python
# plot autocorrelation for cpi data
plt.figure()
smt.graphics.plot_acf(random_df_2, alpha=0.5)
plt.show()
```

<Figure size 432x288 with 0 Axes>

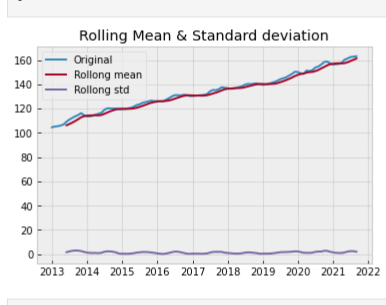


In [74]:
```python
# extract out the time-series
WAGES_ts = WAGES['wages']
```

In [75]:
```python
# when variance is not stationary we took log transformation to find differentiation
# Now time series become stationary by making mean and variance constant
plt.figure(figsize=(10, 5))
plt.plot(np.log10(WAGES_ts).diff(periods=1))
plt.xlabel('Date')
plt.ylabel('Differenced log (wages)');
```



In [76]:
```python
# Determing rolling statistics for both cpi and wpi
rolmean_2 = random_df_2.rolling(window=6).mean()
rolstd_2 = random_df_2.rolling(window=6).std()

# plot rolling statistics
orig = plt.plot(random_df_2, label='Original')
mean = plt.plot(rolmean_2, label='Rollong mean')
std = plt.plot(rolstd_2, label='Rollong std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard deviation')
plt.show(block=False)
```



In [82]:
```python
X_2 = WAGES.values
result_2 = adfuller(X_2)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: 6.007442
p-value: 1.000000
Critical Values:
	1%: -3.558
	5%: -2.917
	10%: -2.596
```

In [83]:
```python
from statsmodels.tsa.stattools import adfuller
X_2 = WAGES.values
result_2 = adfuller(X_2)
print('ADF Statistic: %f' % result_2[0])
print('p-value: %f' % result_2[1])
print('Critical Values:')
for key, value in result_2[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: 0.590408
p-value: 0.987377
Critical Values:
	1%: -3.500
	5%: -2.892
	10%: -2.583
```

Housing prices

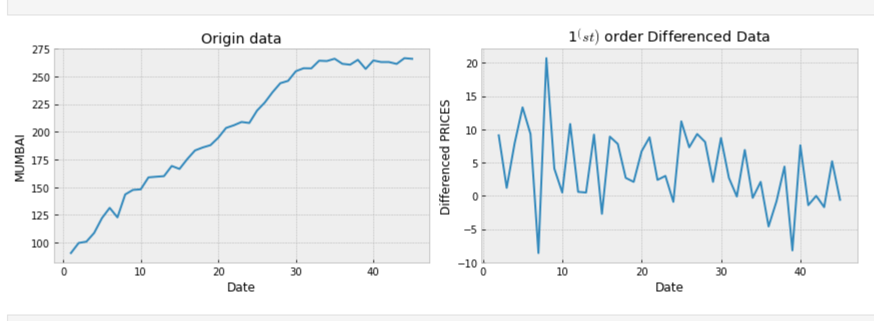
In [117…
```python
house_price = pd.read_excel("C:\\Users\\bahirwal\\Desktop\\Balaji\\Advanced ECotrix\\Task-3\\Mumbai housing qu
house_price.head()
```

Out[117…

	Date	MUMBAI
0	1	90.6
1	2	99.7
2	3	100.9
3	4	108.8
4	5	122.1

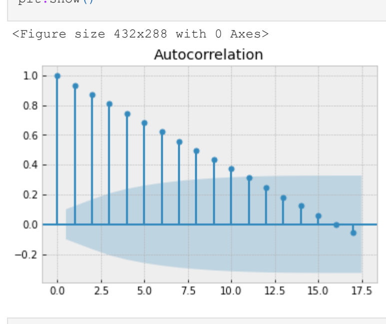
In [119…
```python
house_price.set_index('Date', inplace=True)
```

In [132…
```python
random_df_HP = house_price
fig, axes = plt.subplots(1, 2, sharey=False, sharex=False)
fig.set_figwidth(12)
fig.set_figheight(4)
axes[0].plot(random_df_HP.index, random_df_HP['MUMBAI'])
axes[0].set_xlabel('Date')
axes[0].set_ylabel('MUMBAI')
axes[0].set_title('Origin data')
axes[1].plot(random_df_HP.index, random_df_HP['MUMBAI'].diff(periods=1))
axes[1].set_xlabel('Date')
axes[1].set_ylabel('Differenced PRICES')
axes[1].set_title('1$^(st)$ order Differenced Data')
plt.tight_layout()
plt.show()
```



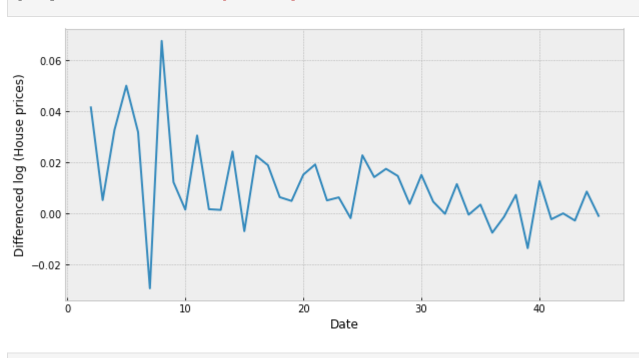
In [121…
```python
# plot autocorrelation for cpi data
plt.figure()
smt.graphics.plot_acf(random_df_HP, alpha=0.5)
plt.show()
```

<Figure size 432x288 with 0 Axes>

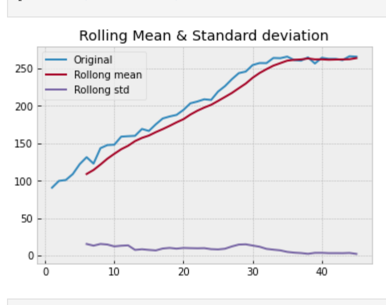


In [122…
```python
# extract out the time-series
HP_ts = house_price['MUMBAI']
```

In [123…
```python
# when variance is not stationary we took log transformation to find differentiation
# Now time series become stationary by making mean and variance constant
plt.figure(figsize=(10, 5))
plt.plot(np.log10(HP_ts).diff(periods=1))
plt.xlabel('Date')
plt.ylabel('Differenced log (House prices)');
```



In [124…
```python
# Determing rolling statistics for both cpi and wpi
rolmean_HP = random_df_HP.rolling(window=6).mean()
rolstd_HP = random_df_HP.rolling(window=6).std()

# plot rolling statistics
orig = plt.plot(random_df_HP, label='Original')
mean = plt.plot(rolmean_HP, label='Rollong mean')
std = plt.plot(rolstd_HP, label='Rollong std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard deviation')
plt.show(block=False)
```



In [125…
```python
X_HP =house_price.values
result_HP = adfuller(X_HP)
print('ADF Statistic: %f' % result_HP[0])
print('p-value: %f' % result_HP[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: -2.609357
p-value: 0.091079
Critical Values:
	1%: -3.558
	5%: -2.917
	10%: -2.596
```

Housing prices in mumbai are not stationary as p-value is 0.091079 and ADF statistic of -2.609357. we made the time series stationary using first order differencing.

In [ ]: