Time Series Analysis

IDS.131 P-Set

Problem Set 5

Yash Dixit

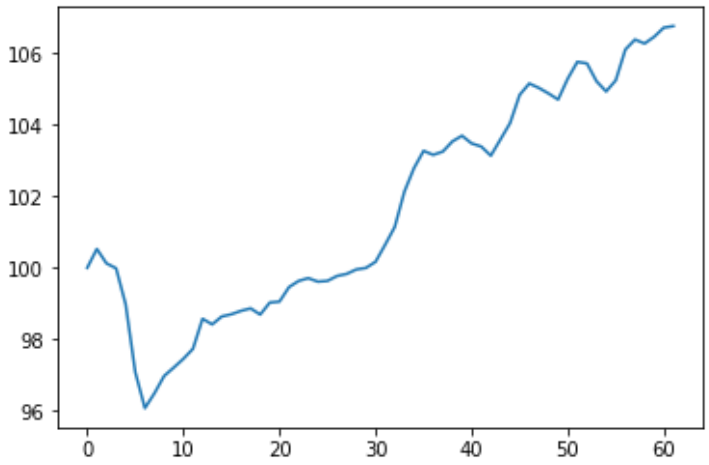
Problem 5.1: BPP Data Analysis

(a) Monthly CPI time series (w/o BER or PriceStats)

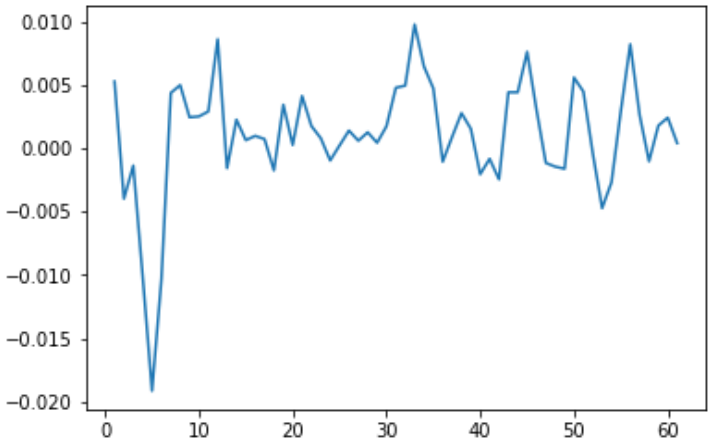
- Sections of the code

```
CPIdiff=CPIseries.pct_change()[1:len(CPIseries)]
CPIdiff.plot()
pyplot.show()
CPIseries
from statsmodels.graphics.tsaplots import plot_pacf
from pandas.plotting import autocorrelation_plot

#plot_pacf(CPIdiff, lags=20)
#CPIdiff
autocorrelation_plot(CPIseries)
plot_pacf(CPIseries, lags=15)
pyplot.show()
```

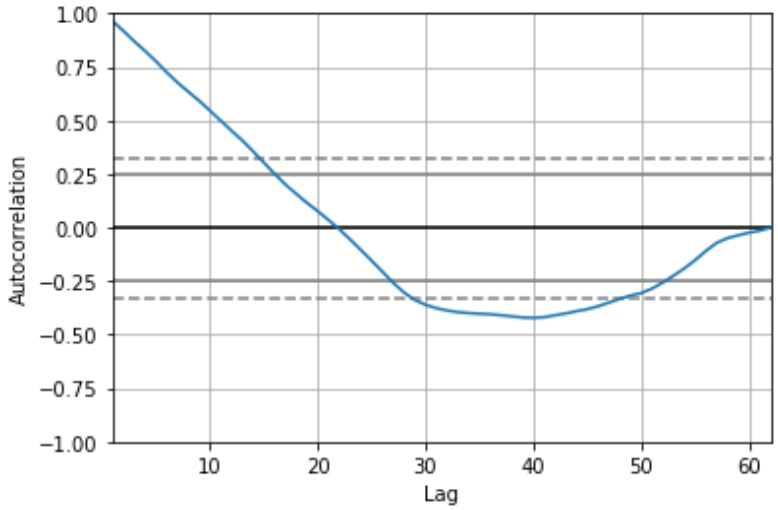
- Relevant plots for illustration



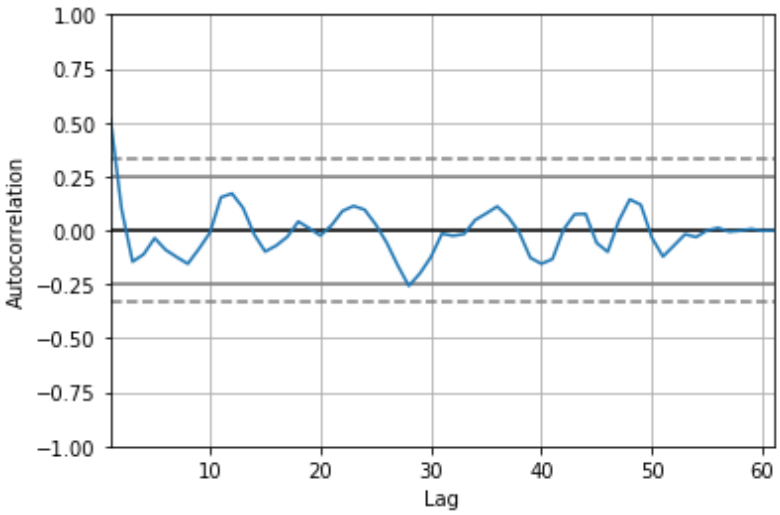
Monthly CPI (w/o log or differencing)



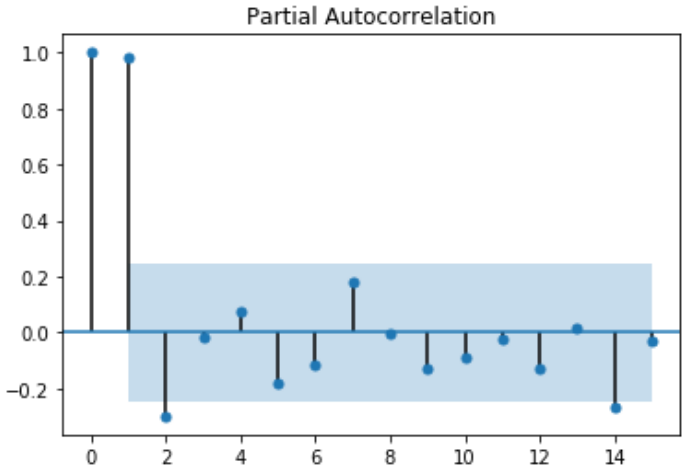
Differenced series – Monthly CPI



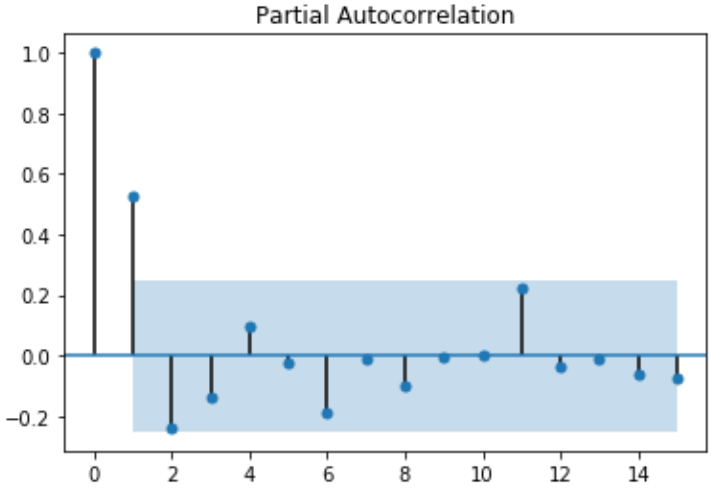
Correlogram (w/o differencing)



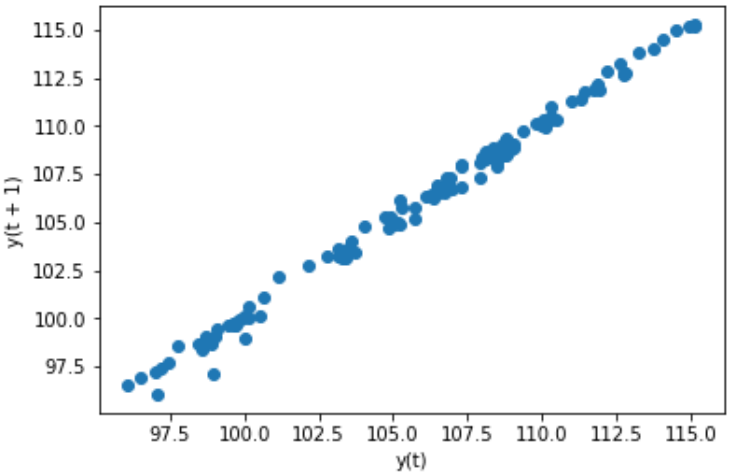
Correlogram (after differencing)



PACF (w/o differencing)



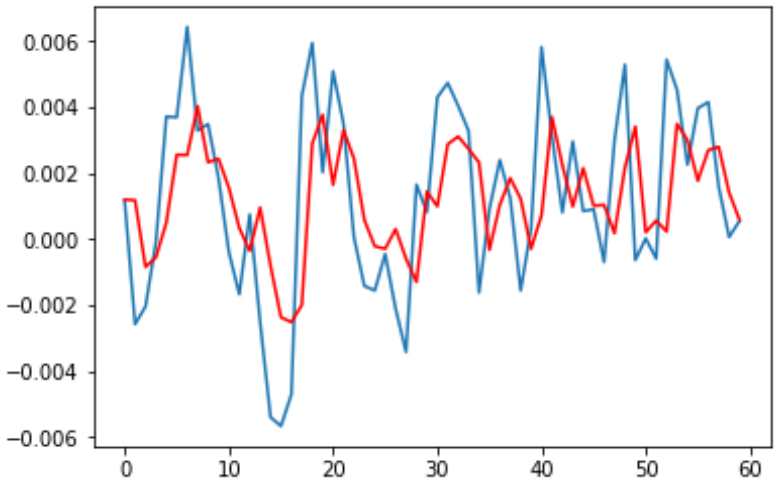
PACF (after differencing) – The correlogram and PACF form the basis for concluding that the transformed time series is stationary



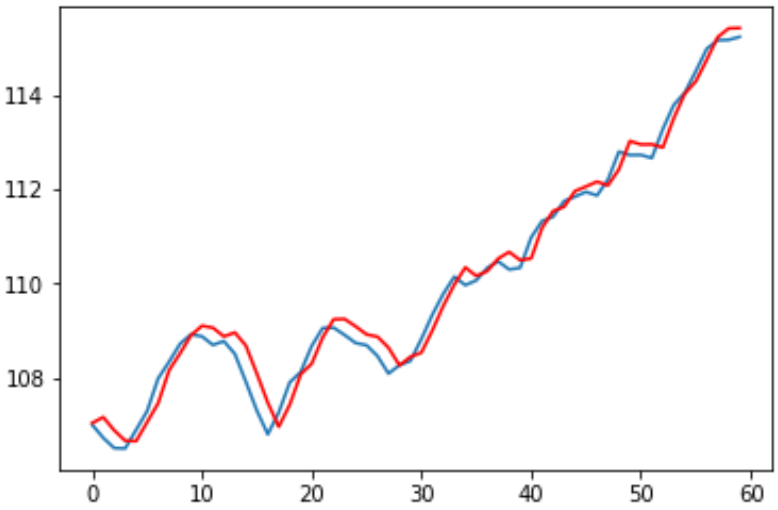
Lag Plot (Illustrative of the time series relationship)

- Sample code for fitting the AR1 model

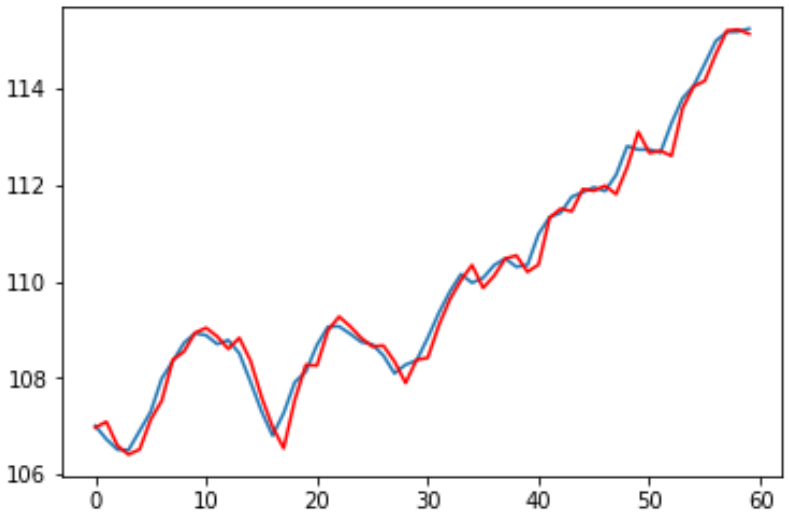
```
cpilogdiff = cpilogdiff.dropna()
X = cpilogdiff.values
train, test = X[0:round(len(X)/2)], X[round(len(X)/2):-1]
# train autoregression
#
model = AR(train)
model_fit = model.fit(maxlag=1)
window = model_fit.k_ar
coef = model_fit.params
```



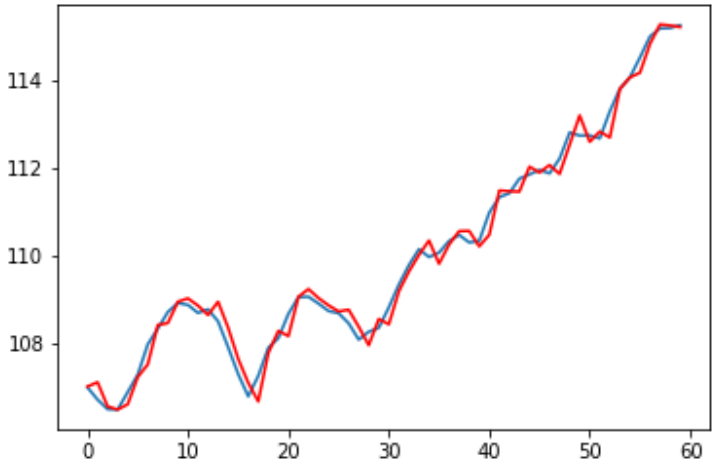
AR1 fit on the stationarized time series data



AR1 fit after re-transformation (MSE = .099)



AR2 fit (MSE = 0.076)



AR3 (MSE = 0.069)

- We can conclude that the AR3 fit has the lowest MSE (after checking for subsequent models)

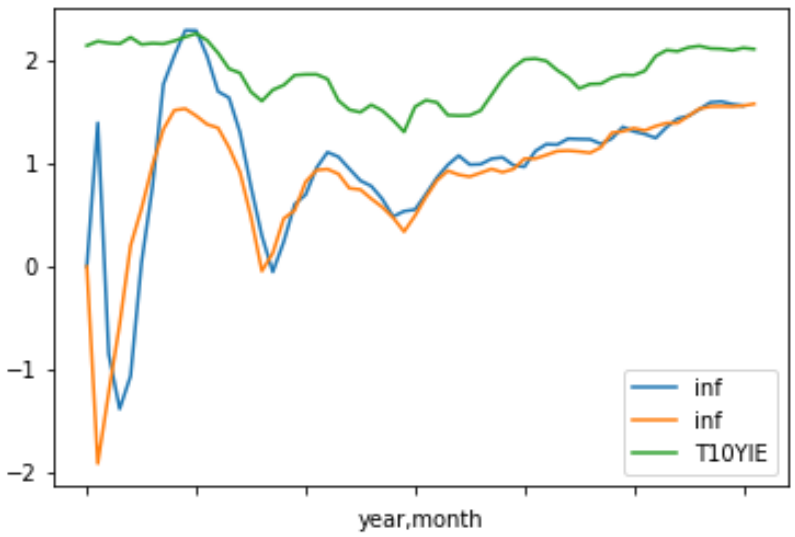
(b) Monthly inflation rates

- Approach is to consider monthly averages as a proxy for the monthly inflation rate
- Snippets of the code

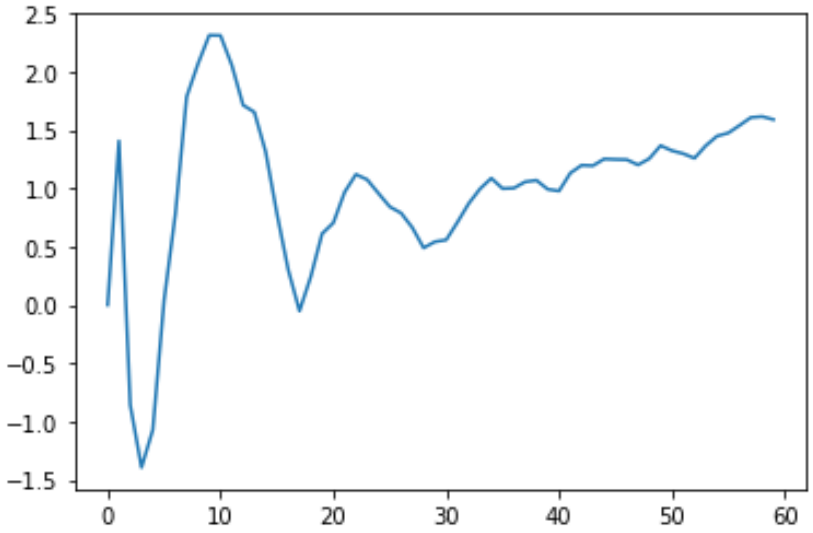
```
or i in range(0,m):
    for j in range(0,1):
```

```
        ps['date'] = pd.to_datetime(ps['date'])
        ps['day1'] = pd.DatetimeIndex(ps['date']).day == 1
        ps['year'] = ps['date'].apply(lambda x: x.strftime('%Y'))
        ps['month'] = ps['date'].apply(lambda x:
x.strftime('%m'))
        pss = ps[ (ps['day1'] == True) ]

psg = ps.groupby(['year', 'month'])
cpimav = psg.aggregate({'CPI':np.mean})
pricemav = psg.aggregate({'PriceStats':np.mean})
cpimav = cpimav.drop(cpimav.index[range(0,62)])
pricemav = pricemav.drop(pricemav.index[range(0,62)])


g,h = cpimav.shape

for i in range(0,m1):
    for j in range(0,1):

        ty['DATE'] = pd.to_datetime(ty['DATE'])
#        ps['day1'] = pd.DatetimeIndex(ps['date']).day == 1
        ty['year'] = ty['DATE'].apply(lambda x: x.strftime('%Y'))
        ty['month'] = ty['DATE'].apply(lambda x:
x.strftime('%m'))
#        pss = ps[ (ps['day1'] == True) ]

tyg = ty.groupby(['year', 'month'])
tmav = tyg.aggregate({'T10YIE':np.mean})
tmav = tmav.drop(tmav.index[range(0,128)])
```

- Results



Inflation rates of the given data sets



Inflation rate using the AR predictions (CPI)

(c) External Regressors and

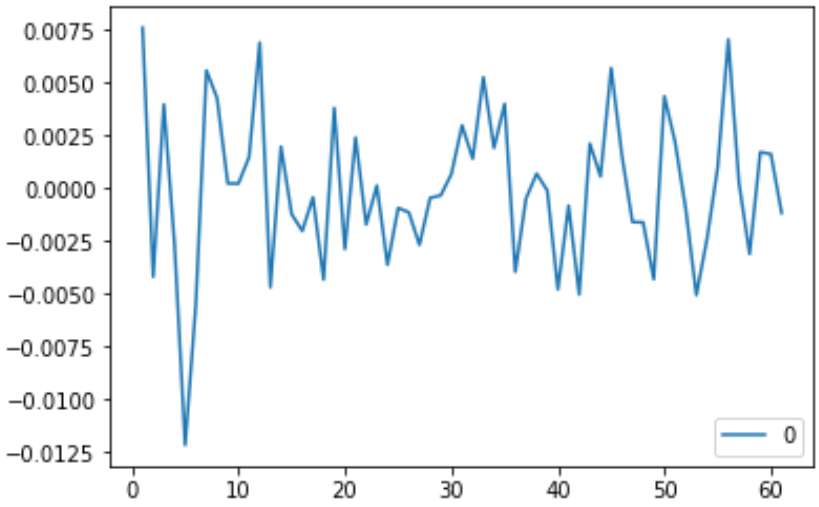
(d) Improving the Model

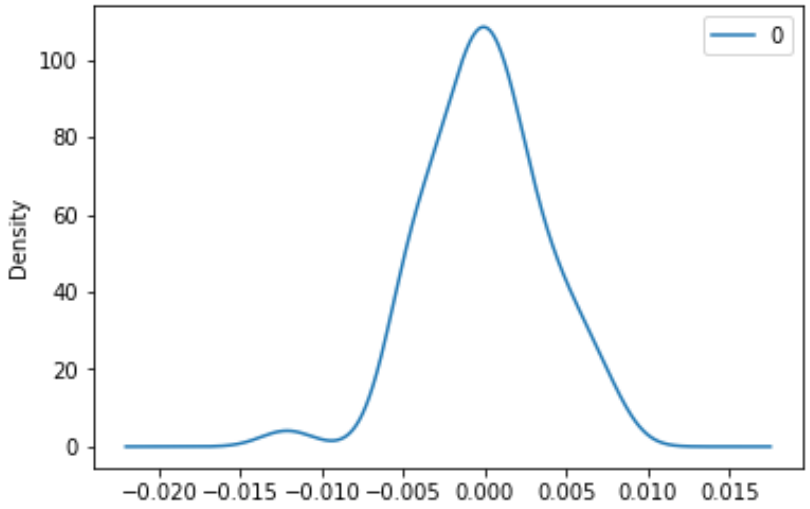
- Model with the code and results:

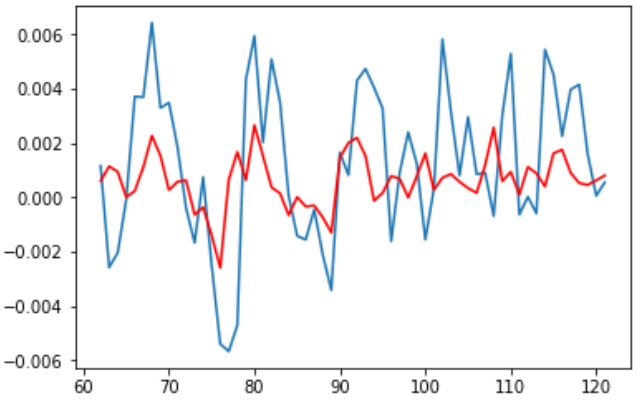
```
from statsmodels.tsa.arima_model import ARIMA
model2=ARIMA(CPIdiff, order=(1,0,0),exog= exregtrain)
#model3=ARIMA(CPIseries, order=(1,1,0))

model_2 = model2.fit()
```

```
#model_3= model3.fit(maxlag=1)

predictions2 = model_2.predict(start=len(CPIdiff),
end=len(CPIdiff)+len(CPItest)-1, dynamic=True, exog=exregtest)
error2 = mean_squared_error((CPItest), (predictions2))

residuals = pd.DataFrame(model_2.resid)
```

Residual plots for the above model



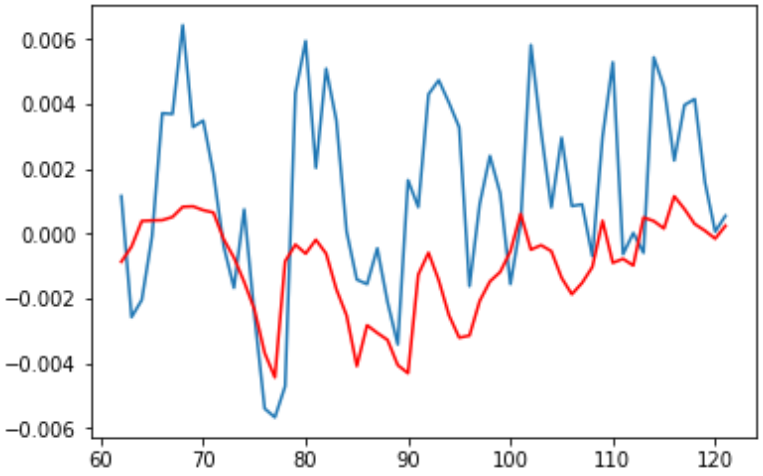




Predictions

- Model with the code and results:

```
model3=ARIMA(CPIdiff, order=(1,0,0),exog= exregtrain1)
#model3=ARIMA(CPIseries, order=(1,1,0))

model_3 = model3.fit()
#model_3= model3.fit(maxlag=1)

predictions3 = model_3.predict(start=len(CPIdiff),
end=len(CPIdiff)+len(CPItest)-1, dynamic=True, exog=exregtest1)
error2 = mean_squared_error((CPItest), (predictions3))
```

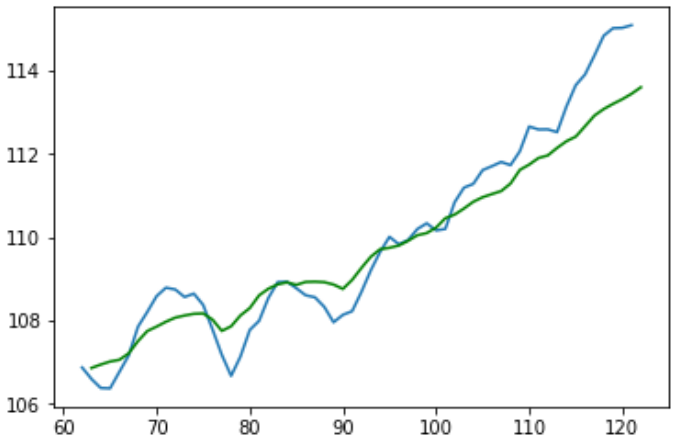


Error value = 1.148563168316832e-05

- Model with the code and results:

```
#from statsmodels.tsa.statespace import SARIMAX
import statsmodels.api as sm
```

```
model2=sm.tsa.statespace.SARIMAX(CPIdiff[1:len(CPIdiff)],
order=(1,0,1), seasonal_order=(1,0,1,12),
exog=exregtrain[1:len(exregtrain)])
#model3=ARIMA(CPIseries, order=(1,1,0))

model_2 = model2.fit()

predictions2 = model_2.predict(start=len(CPIdiff),
end=len(CPIdiff)+len(CPItest)-1, dynamic=True, exog=exregtest)
error2 = mean_squared_error((CPItest), (predictions2))

predictions4 = predictions2.copy()
predictions4[63]=106.75*(1+predictions4[63])

for i in range(64,123):
    predictions4[i]=predictions4[i-1]*(1+predictions4[i])
```



Error: 6.661318899477206e-06

The models progressively become better in prediction accuracy.

(e) Autocovariance of the MA1 Model

$$x_t = w_t + bw_{t-1}$$

$$\gamma(k) = E(x_t x_{t-k})$$

$$\gamma(k) = E[(w_t + bw_{t-1})(w_{t-k} + bw_{t-k-1})]$$
$$= E(w_t w_{t-k} + bw_t w_{t-k-1} + bw_{t-1} w_{t-k} + b^2 w_{t-1} w_{t-k-1})$$

$$= E(w_t w_{t-k}) + E(bw_t w_{t-k-1}) + E(bw_{t-1} w_{t-k}) + E(b^2 w_{t-1} w_{t-k-1})$$

$$\gamma(0)=\sigma^2 =E(w^2)+bE(w\,w)+bE(w\,w)+b^2E(w^2)$$

$$\gamma(1) = E(w_tw_{t-1}) + bE(w_tw_{t-2}) + bE(w_t{}^2{}_{-1}w_{t-1}) + b^2E(w_{t-1}w_{t-2})$$

$$\gamma(1) = b\sigma_w{}^2$$

FOr MA(1), the autocovariance function truncates (i.e., it is zero) after lag 1.

(f) Autocovariance of the AR1 Model

$$X_t = \varphi_1X_{t-1} + \varrho_t\,.$$

$$X_t =\varrho_t +\varphi_1(\varrho_{t-1} +\varphi_1(\varrho_{t-2} +\cdots)) = \varrho_t +\varphi_1\varrho_{t-1} +\varphi^2{}_1\varrho_{t-2} +\cdots$$

Stationarity indicates $E(X_t) = 0$

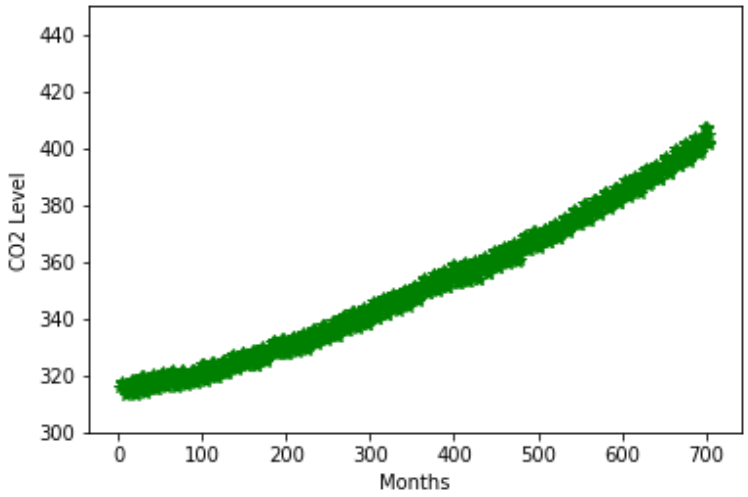
$$\gamma_0 =E(\varrho_t +\varphi_1\varrho_{t-1} +\varphi^2{}_1\varrho_{t-2} +\cdots)^2 = (1+\varphi^2{}_1 +\varphi^4{}_1 +\cdots)\sigma^2 =$$

$$\frac{\sigma^2}{1\ -\ \varphi^2{}_1}$$

$$\gamma_k = \mathbb{E}\left(\sum_{r=0}^{\infty}\phi_1^r\epsilon_{t-r}\sum_{s=0}^{\infty}\phi_1^s\epsilon_{t+k-s}\right) = \frac{\sigma^2\phi_1^k}{1-\phi_1^2}$$

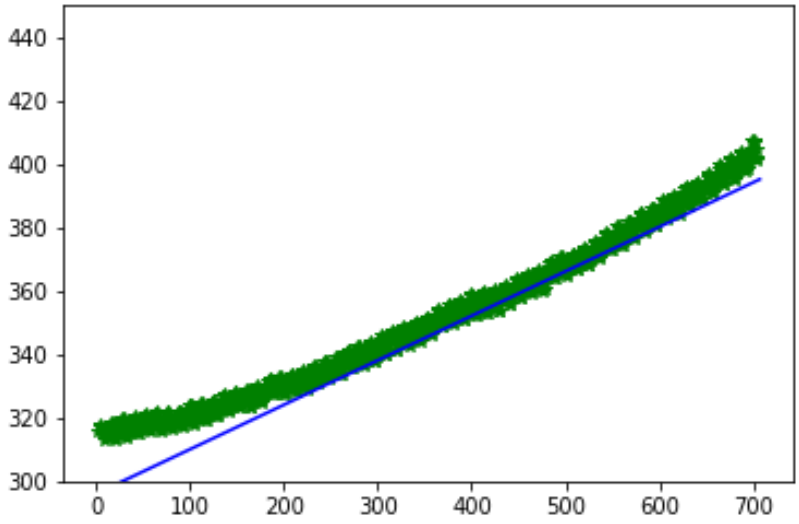
Problem 5.2: The Mauna Loa CO2 concentration

- General visualization of the CO2 data



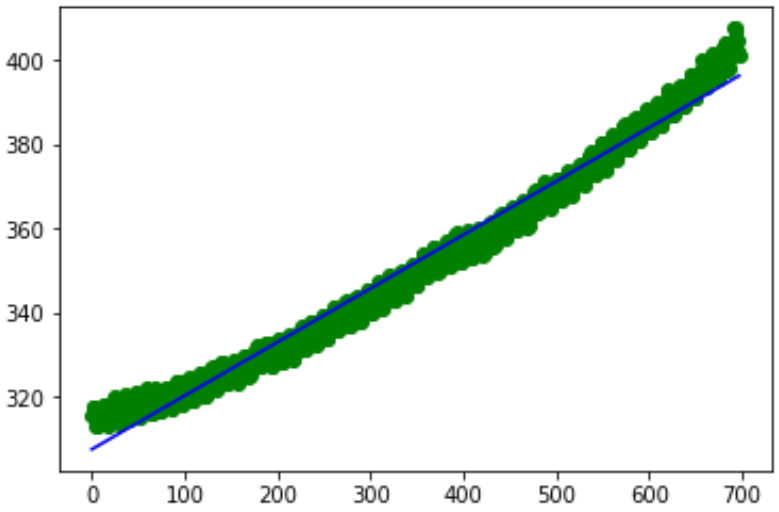
- Linear Fits (Code with coefficients + Plots)

```
plt.plot(Mn_Length, 296.0171 + 0.1403 * Mn_Length, 'b-')
plt.ylim(top=450)
plt.ylim(bottom=300)
```

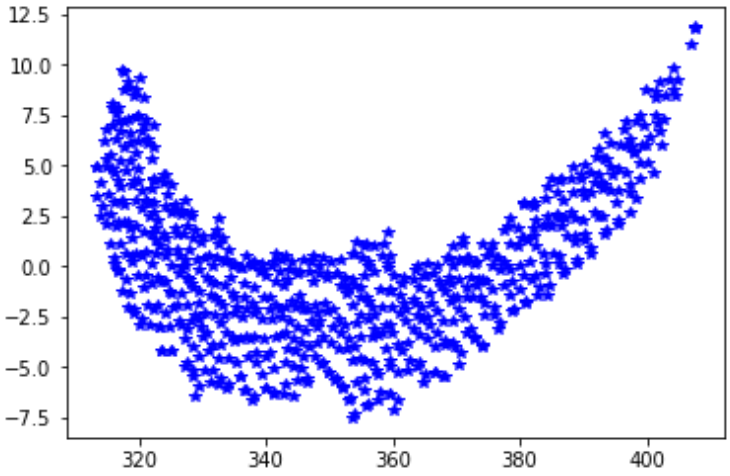


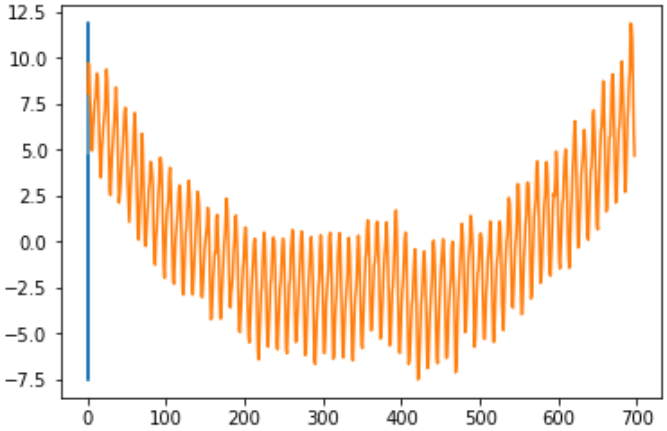
	coef	std err	t	P>|t|	[0.025	0.975]
const	296.0171	3.897	75.964	0.000	288.366	303.668
x1	0.1403	0.010	14.703	0.000	0.122	0.15

Manage outliers and then re-run the regression



	coef	std err	t	P>|t|	[0.025	0.975]
const	307.6219	0.289	1063.749	0.000	307.054	308.190
x1	0.1273	0.001	177.270	0.000	0.126	0.129





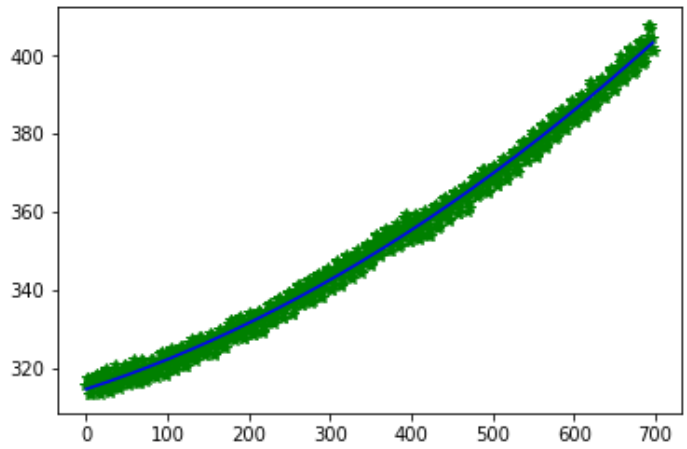
Residuals plotted with CO2 conc and time scale

- The residual plot shows non-trivial pattern (vaguely, U-shaped) that implies non-random residuals.

- This is an indicator to try other non-linear models (which also seems intuitive from the preliminary visualization, wherein the plot shows a slight increasing curvature – akin to a quadratic / exponential)

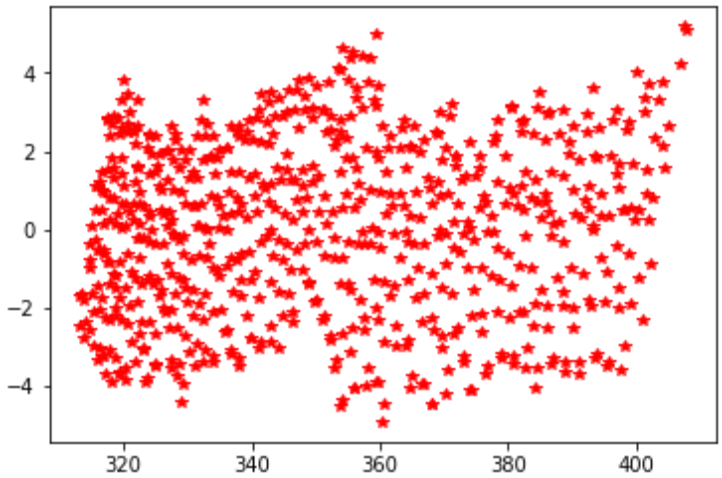
- Quadratic Fit

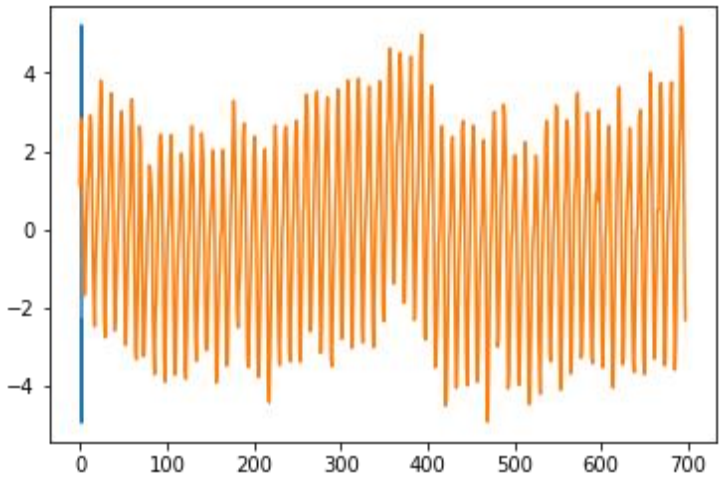
```
z = np.polyfit(range(0,len(CO2_NOoutliner)), CO2_NOoutliner,2)
print(z)
#y = β1 + β2*t + β3*t^2
β3, β2, β1 = z
print(β3, β2,β1)
```



Coeff and Fit: [8.58498708e-05 6.75086292e-02 3.14563005e+02]

Residuals plotted with CO2 conc and time scale

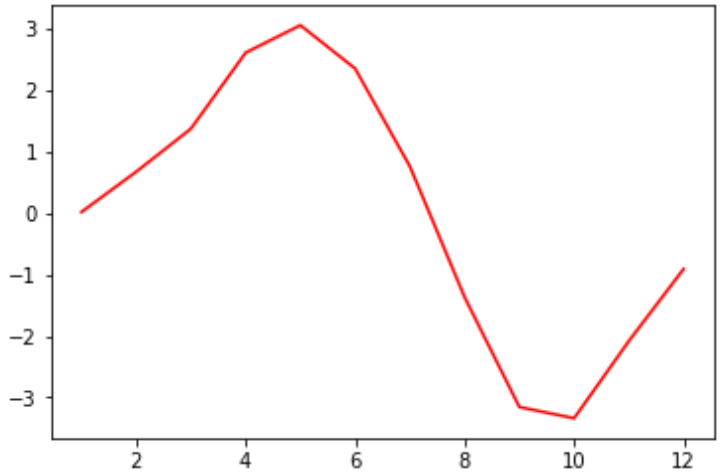


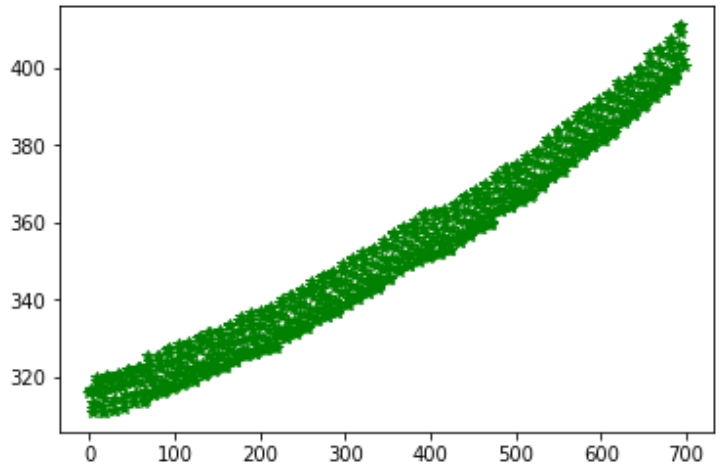


- We can infer that the quad fit is better (there is no perceivable pattern to the residuals unlike the linear fit which had a U-shaped plot)

- Having said that, we can also see some periodicity to the residual plot possibly indicating a salient seasonal element to the CO2 concentration

- Monthly averaged residuals with the de-seasoned plot





- We can draw the following conclusions about the CO2 concentration i.e. though the seasonal component is non-trivial, we can clearly see an increasing trend (as verified after de-seasonalizing the time series)