Homework Report #4: Time Series

Author: Isaac Han

Email: cogitoergosum01001@gmail.com

Class: Data Analysis: Statistical Modeling and Computation in applications

Professors: Prof. Uhler, Prof. Jegelka

Table of Contents

DATA:

Information about the data set

Named as `CO2.csv`, it provides concentration of CO_2 in ppm unit, derived from in situ air measurements, at Mauna Loa, Observatory, Hawaii. The observatory is located at Lattidue of 19.5'' North and Longitude of 155.6'' W with elavation of 3397m.

The concentration of CO_2 was recorded each month starting from March 1958. And only the concentration given in column 5 was used as instructed by the course. The data set contains NA(Not A Number Type) and appropriate data processing was required.

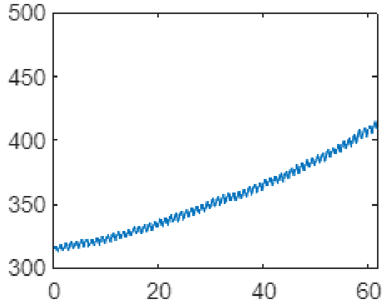
Data Parameters

- $C_i = F(t_i) + P_i + R_i$ where $F :\to F(t)$ accounts for the ong-term trend
- t_i is time at the middle of the ith month, measured in fractions of years after Jan 15,1958. $t_i = \frac{i+0.5}{12}$
- P_i is periodic in i with a fixed period, accounting for the seasonal pattern.
- R_i is the remaining residual that accounts for all other influences.
- Data and test sets are splited into 80:20

Pre-Processing

```
opts = detectImportOptions('CO2.csv');

T = readtable('CO2.csv',opts);
T = T(3:end,:);
dim = size(T);
T.count = (0:dim(1)-1)' ;
T.time = (T.count + 0.5)/12;

processed_T = T(T.CO2~=-99.99,:);
any(processed_T.CO2 == -99.99);
writetable(processed_T,'CO2Processed.csv');
writetable(T,'CO2notprocessed.csv');
clear
```

```
opts = detectImportOptions('CO2Processed.csv');
% preview('CO2Processed.csv',opts)
T = readtable('CO2processed.csv');
plot(T.time,T.CO2)
ylim([300 500])
```



Partitioning

```
cutoff = round(size(T,1)*0.8)
```

```
cutoff = 587
```

```
training = {T.time(1:cutoff), T.CO2(1:cutoff)};
test = {T.time(cutoff+1:end), T.CO2(cutoff+1:end)};
```

Detrending: Quadratic fitting

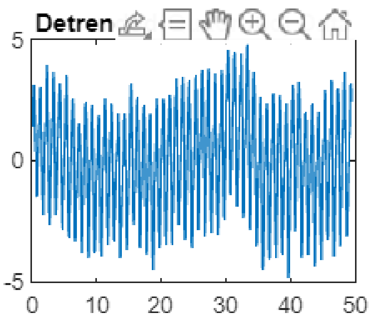
```
x = [ones(length(training{1}),1),training{1},training{1}.^2];
y = training{2}
```

```
y = 587×1
  315.7000
  317.4500
  317.5100
  315.8600
  314.9300
  313.2100
  313.3300
  314.6700
  315.5800
  316.4800
      ⋮
```

```
b = x\y
```

```
b = 3×1
  314.1006
    0.8021
    0.0121
```

```
T_cutoff = T(1:cutoff, :);
T_cutoff.detrend = y - x*b;
plot(T_cutoff.time,T_cutoff.detrend)
title('Detrended(Quadratic) Data')
xlim([0 50])
```



Written Exercise:

Q1: Plot the periodic signal for each month

From previous section, detrending,

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}. \text{ In Matrix form, let's call } Y = A * \beta$$

By performing the linear regression, one can obtain parameters of $\beta' s$ that minimizes the squared error. Therefore,

$\text{Residual}_i = y_i - A * \widehat{\beta}$ where $\widehat{\beta}$ is 3 by 1 matrix that minimizes the least squred error.

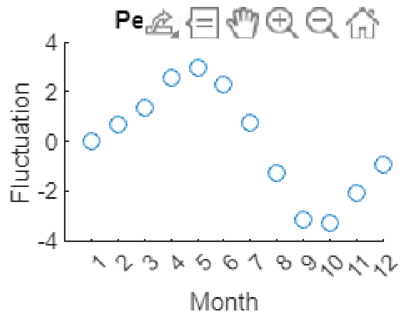
The results of residuals are added to the data.frame as Table.detrend. Now, the data frame is filtered based on column entries named Mn.

$$P_i = \frac{1}{\left\lfloor \frac{\text{length}(y)}{12} \right\rfloor + 1} \sum_{k=0}^{\left\lfloor \frac{\text{length}(y)}{12} \right\rfloor} y_{i+12k}$$

Or simply just a biased mean of samples corresponds to $\text{For } y_j,\ j \equiv i \bmod(12)$.

```
P_i = zeros(1,12);
for i = 1:12
    T_month = T_cutoff(T_cutoff.Mn ==i,:);
    P_i(i) = sum(T_month.detrend)/size(T_month,1);
end
scatter(1:12,P_i)
```

```
xlabel('Month')
ylabel('Fluctuation')
title('Periodic signal P_i')
xticks(1:12)
```




```
k = table(P_i');
k.Properties.VariableNames = "P";
k.Properties.RowNames =
["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"];
disp(k)
```

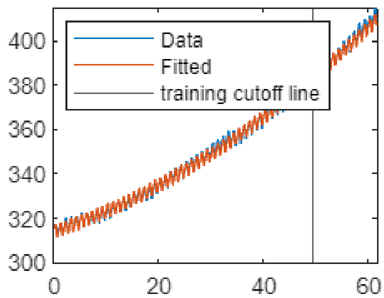
	P
Jan	-0.012919
Feb	0.64641
Mar	1.3556
Apr	2.5619
May	2.9829
Jun	2.3165
Jul	0.7763
Aug	-1.3012
Sep	-3.1281
Oct	-3.3095
Nov	-2.0815
Dec	-0.92151

Q2: Plot the final fit

The final fit is defined as $F_n(t_i) + P_i$. Plot final fit on top of the entire time series with a vertical line indicating the split between the training and test data.

```
clear x y T_month
clf

figure()
p1 = plot(T.time,T.CO2);
x = [ones(length(T.time),1),T.time,T.time.^2];
T.extra = P_i([T.Mn])';

hold on
```

```
p2 = plot(T.time,x*b +T.extra);
xline(T.time(cutoff))
legend('Data','Fitted', 'training cutoff line',Location='northwest')
hold off
xlim([0 max(T.time)])
```




```
clear extra
```

Q3: Reported error of RMSE and MAPE(Max 200 words)

RMSE(Root-Mean-Square Error) is defined as following.

$$E = \sqrt{\frac{1}{n}\sum_{i=1}^{n}|A_i - F_i|^2}$$

MAPE(Mean Absolute Percentage Error) is defined as following.

$$E = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - F_i}{A_i}\right| \times 100$$

Notice that error was applied only to the training set.

```
clear x
x = [ones(length(training{1}),1),training{1},training{1}.^2];
predict = x*b +T.extra(1:cutoff)
```

```
predict = 587×1
  315.6238
  316.8974
  317.3860
  315.3149
  313.3054
  311.5467
  312.7302
  313.9589
  314.9363
  315.6647
      ⋮
```

```
RMSE_Error = rmse(x*b,training{2});
```

```
MAPE_Error = mape(x*b, training{2});
fprintf('Reported RMSE and MAPE errors, (%.3f, %.3f)',RMSE_Error,MAPE_Error)
```

```
Reported RMSE and MAPE errors, (2.188, 0.537)
```

```
RMSE_Error2 = rmse(predict,training{2});
MAPE_Error2 = mape(predict, training{2});
fprintf('Reported RMSE and MAPE errors, (%.3f, %.3f)',RMSE_Error2,MAPE_Error2)
```

```
Reported RMSE and MAPE errors, (0.714, 0.168)
```

Considering seasonal variation greatly reduced errors, as they are reduced to about 1/3 of their original errors. It is intuitive, as ziggles were shown in the plot of CO_2 level; A linear fitting would ignore such ziggles.

To reason quantitatively, consider the plot of P_i where we observed seasonal variation closely resembles a sine curve.

$P_{\text{continuous}} = \mathrm{A}\sin\left(\phi + \frac{2\pi t}{p}\right) + B$ which is a typical sine wave form introduced in class.

Then, the error generated by ignoring seasonal variation for one period would be,

$\frac{1}{N}\sum |P_{\text{continuous}}| = \frac{1}{N}\left|\sum \mathrm{A}\sin\left(\phi + \frac{2\pi t}{p}\right)\right| \approx \frac{1}{N}\left|\mathrm{A}\cos\left(\phi + \frac{2\pi t}{p}\right) * \frac{2\pi}{p}\right|$ sum may be approximated to integral when time intervals are sufficiently small compared to our interest. Notice by the definition of time, $p = 1$.

Also notice that having unaccounted error term produce a product of errors in RMSE:

$$(E_{\text{seasonal}} + E_{\text{Other}})^2 = (E_{\text{seasonal}})^2 + (E_{\text{Other}})^2 + 2E_{\text{seasonal}}E_{\text{Other}}$$

Therefore, having errors that are not independent might contribute to the error by forming products.

Q4: Ratio of values

1. The ratio of range of values of F to amplitude of P
2. The ratio of the amplitude of P to the range of residual R(both the trend and periodic signal

8u7

Recall the following matrix equation of F,

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}. \text{ In Matrix form, let's call } Y = A * \beta$$

In this paper, $F_n(t_i)$ is equivalent to y_i. Therefore, the range of F is $\max(Y) - \min(Y) = 74.6814$

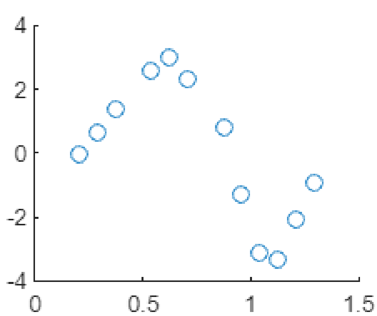
```
P_i
```

```
P_i = 1×12
   -0.0129    0.6464    1.3556    2.5619    2.9829    2.3165    0.7763   -1.3012 ⋯
```

```
t = T.time(1:12)
```

```
t = 12×1
    0.2083
    0.2917
    0.3750
    0.5417
    0.6250
    0.7083
    0.8750
    0.9583
    1.0417
    1.1250
      ⋮
```

```
scatter(t,P_i)
```

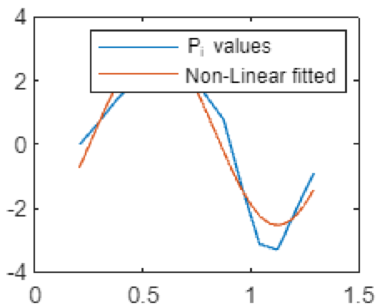



```
beta0 = [3, 0, 1.5, 0];
modelfun = @(b,t) b(1)*sin(b(2)+2*pi.*t/b(3)) + b(4) ;
sine_fit = fitnlm(t, P_i, modelfun,beta0)
```

```
sine_fit = 
Nonlinear regression model:
    y ~ b1*sin(b2 + 2*pi*t/b3) + b4

Estimated Coefficients:
          Estimate      SE        tStat       pValue  
          ________    _______    _______    __________

    b1     2.8274     0.28192     10.029    8.3061e-06
    b2    -1.5152      0.2151    -7.0444    0.00010777
```

```
    b3      1.1357      0.058085      19.553    4.8656e-08
    b4     0.29852       0.21047      1.4183       0.19385


Number of observations: 12, Error degrees of freedom: 8
Root Mean Squared Error: 0.679
R-Squared: 0.928,  Adjusted R-Squared 0.901
F-statistic vs. constant model: 34.4, p-value = 6.4e-05
```

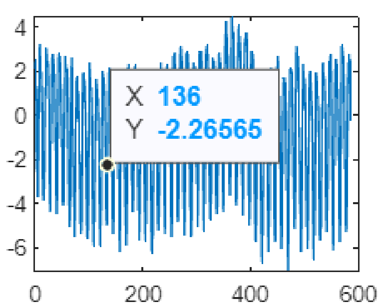
```
clear predict
plot(t,P_i)
hold on
x1 = linspace(t(1),t(12));
y1 = sine_fit.Coefficients.Estimate;
plot(x1, modelfun(y1,x1))
hold off
legend('P_i values', 'Non-Linear fitted')
```



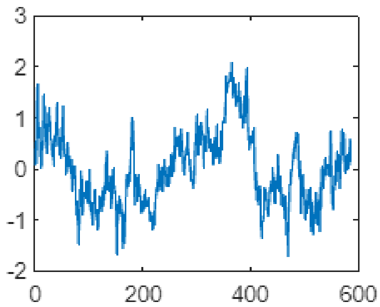

```
clear x1 y1
```

From non-linear fitting, estimated parameter states that amplitude is 2.8274.

```
clear x
x = [ones(length(training{1}),1),training{1},training{1}.^2];
residual1 = training{2} - x*b - predict(sine_fit, (mod(T_cutoff.count, 12) +
0.5)/12 );
plot(residual1)
```




```
figure()
residual2 = training{2} - x*b - T.extra(1:cutoff);
plot(residual2)
```




```
fprintf('Non-linear fitting gave the range of residuals %.4f',max(residual1)-
min(residual1))
```

```
Non-linear fitting gave the range of residuals 11.5666
```

```
fprintf('Subtracting the average of each months(constant) gave the range of
residual %.4f',max(residual2)-min(residual2))
```

```
Subtracting the average of each months(constant) gave the range of residual 3.8364
```

It was found that the range of residuals would be smaller by subtracting $P_i's$ for each month. It was found that $R = 3.8364$

Rest of them are trivial arithmatics;

1. $\frac{F}{P} = \frac{74.6814}{2.8274} \approx 26.41$
2. $\frac{P}{R} = \frac{2.8274}{3.8364} \approx 0.7370$

```
F = 74.6814;
P = 2.8274;
F/P;
R = 3.8364;
P/R;
```

Decomposition of the variation of the $\mathrm{CO_2}$ concentration can be meaningful based on applications. For an application to show the global trend of $\mathrm{CO_2}$ for 80 years, it may not be important as $F >> P$. However, forecasting $\mathrm{CO_2}$ concentration of a few months in the future, the seasonal variation plays a great role as $\frac{\mathrm{P}}{\Delta F}$ and $\frac{P}{R}$ are considerable. Meaning, the change of the level from a month to another is greatly dependent on seasonal variation.

Written Exercise: ARIMA

Q1: Consider MA(1) model, find autocovariance function

Consider MA(1) model,

$y_t = W_t + \theta W_{t-1} = W_t(1 + \theta L)$ where L is a lag operator with properties $L^k X_t = X_{t-k}$

And lag polynomial is defined as following;

$$\theta(L) = 1 + \sum_{i=1}^{q} \theta_i L^i \quad \varphi(L) = 1 - \sum_{i=1}^{p} \varphi_i L^i$$

```
Mdl = arima(0,0,1)
```

```
Mdl = 
  arima with properties:

     Description: "ARIMA(0,0,1) Model (Gaussian Distribution)"
    Distribution: Name = "Gaussian"
               P: 0
               D: 0
               Q: 1
        Constant: NaN
              AR: {}
             SAR: {}
              MA: {NaN} at lag [1]
             SMA: {}
     Seasonality: 0
            Beta: [1×0]
        Variance: NaN
```

```
MA_residual = estimate(Mdl,residual2)
```

```
    ARIMA(0,0,1) Model (Gaussian Distribution):
```

	Value	StandardError	TStatistic	PValue
Constant	-0.00071834	0.035087	-0.020473	0.98367
MA{1}	0.69687	0.031515	22.112	2.4309e-108
Variance	0.23754	0.015796	15.037	4.1818e-51

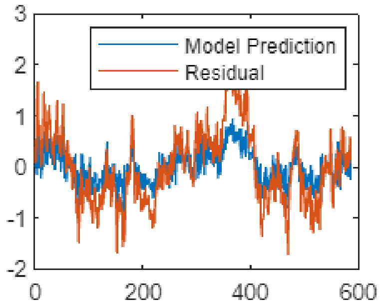
```
MA_residual = 
  arima with properties:

     Description: "ARIMA(0,0,1) Model (Gaussian Distribution)"
    Distribution: Name = "Gaussian"
               P: 0
               D: 0
               Q: 1
        Constant: -0.000718339
              AR: {}
             SAR: {}
              MA: {0.696865} at lag [1]
             SMA: {}
     Seasonality: 0
            Beta: [1×0]
        Variance: 0.237537
```

```
leftover = infer(MA_residual,residual2,'Y0',residual2(1:2));
y_hat = residual2 - leftover
```

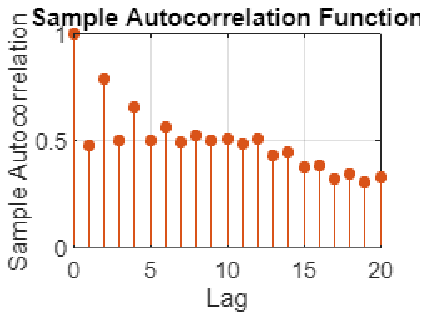
```
y_hat = 587×1
```

```
-0.0007
 0.0529
 0.3475
-0.1564
 0.4882
 0.7912
 0.6070
-0.0057
 0.4988
 0.1002
   ⋮
```

```
plot(y_hat)
hold on
plot(residual2)
hold off
legend('Model Prediction','Residual')
```




```
figure()
[acf ,~] = autocorr(leftover,NumSTD=0);
autocorr(leftover,NumSTD=0)
```




```
[xcf, lag] = xcov(leftover);
```

This paper reports the following results;

$$y_t = W(\mu, \sigma^2)_t(1 + \theta L) = W(0, 0.2375^2)_t(1 + 0.6969 * L)$$

For autocovariance, XCF(cross correlation function) it is defined as following;

$$C_{xy}(m) = \begin{cases} \frac{1}{T}\sum_{t=1}^{T+k}(y_{1,t}-\overline{y_1})(y_{2,t}-\overline{y_2}); & k = 0, 1, 2, \cdots \\ \frac{1}{T}\sum_{t=1}^{T+k}(y_{1,t}-\overline{y_1})(y_{2,t}-\overline{y_2}); & k = 0, -1, -2, \cdots \end{cases}$$

From simple algebra, $\text{autocorr} = \frac{C_k}{C_o}$ where C was defined previously. Therefore,

$C_k = \text{autocorr} * C_0$

```
fprintf('Autocovariance of the first 10 lags: ')
```

```
Autocovariance of the first 10 lags:
```

```
disp(xcf(587)*acf(1:10))
```

```
  139.4338
   66.2124
  109.7654
   69.5042
   91.4393
   69.3541
   79.0661
   69.1657
   72.8785
   69.9372
```

Q2: AR(1) model

For AR(1) model,

$X_t = \phi X_{t-1} + W_t$ or equivalently, for the consistancy of equation format,

$(1 - \phi L)X_t = W_t$

```
AR = arima(1,0,0)
```

```
AR = 
  arima with properties:

     Description: "ARIMA(1,0,0) Model (Gaussian Distribution)"
    Distribution: Name = "Gaussian"
               P: 1
               D: 0
               Q: 0
        Constant: NaN
              AR: {NaN} at lag [1]
             SAR: {}
              MA: {}
             SMA: {}
     Seasonality: 0
```

```
        Beta: [1×0]
    Variance: NaN
```

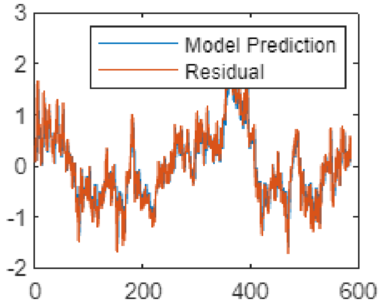
```
AR_residual = estimate(AR,residual2)
```

```
 
    ARIMA(1,0,0) Model (Gaussian Distribution):
 
                  Value      StandardError    TStatistic      PValue  
                __________   _____________    __________    __________

    Constant    0.00013199      0.012331       0.010704        0.99146
    AR{1}          0.90864      0.016249          55.92              0
    Variance      0.088947     0.0048973         18.163     1.0215e-73
AR_residual = 
  arima with properties:

     Description: "ARIMA(1,0,0) Model (Gaussian Distribution)"
    Distribution: Name = "Gaussian"
               P: 1
               D: 0
               Q: 0
        Constant: 0.00013199
              AR: {0.908643} at lag [1]
             SAR: {}
              MA: {}
             SMA: {}
     Seasonality: 0
            Beta: [1×0]
        Variance: 0.088947
```

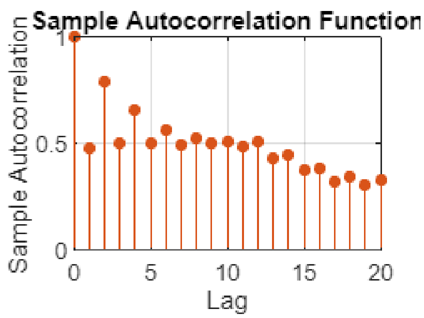
```
leftover2 = infer(AR_residual,residual2);
y_hat2 = residual2 - leftover2
```

```
y_hat2 = 587×1
    0.0632
    0.0694
    0.5022
    0.1128
    0.4954
    1.4763
    1.5114
    0.5451
    0.6463
    0.5850
      ⋮
```

```
plot(y_hat2)
hold on
plot(residual2)
hold off
legend('Model Prediction','Residual')
```




```
figure()
[acf2 ,~] = autocorr(leftover2,NumSTD=0);
autocorr(leftover,NumSTD=0)
```




```
[xcf2, lag2] = xcov(leftover2);
```

This paper reports the following parameters;

$$(1 - 0.908643L)X_t = W(0, 0.088947^2)_t$$

For autocovariance,

```
fprintf('Autocovariance of the first 10 lags: ')
```

```
Autocovariance of the first 10 lags:
```

```
disp(xcf2(587)*acf2(1:10))
```

```
   52.2119
   -9.5560
    2.0354
   -1.1709
    1.9369
   -0.8885
   -0.3401
   -0.0754
   -0.5350
    3.1220
```

Written Exercise: CPI and BER

Data Explanation:

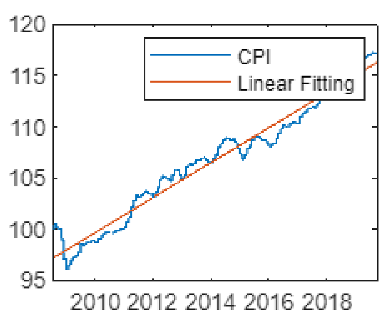
1. CPI(consumer price index)
2. BER(break-even rate)

Data loading

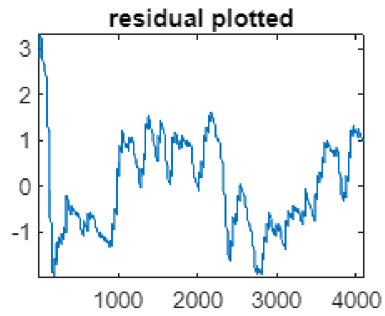
```
clc
clear
opts = detectImportOptions('CPI.csv');
T = readtable('CPI.csv',opts);
dim = size(T);

processed_T = rmmissing(T);
clear T
T = table2timetable(processed_T);
T.scalartime = seconds(T.date - T.date(1))/(3600*24);
```

Q0: Veryfy Your Result By plotting(As recommended by Alan)

```
plot(T.date,T.CPI)
x = [ones(size(T.date,1),1), T.scalartime];
y = T.CPI;
b = x\y;
hold on
plot(T.date,x*b)
hold off
legend('CPI','Linear Fitting')
```




```
figure()
residual = y - x*b;
plot(residual)
axis('tight')
title('residual plotted')
```



AR Model, AR model find parameters, MRSE, 1month ahead forecast

Q1:Repeat the model fitting and evaluation procedure, with rate

1: Description of calculation

Following the definition of inflation rate given by the course, we have;

$$\mathrm{IR}_t = \frac{\mathrm{CPI}_t - \mathrm{CPI}_{t-1}}{\mathrm{CPI}_{t-1}}$$

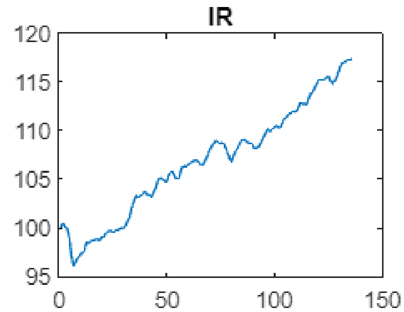
1. Find the start date and end date, which turned out to be 07/24/2008 and 10/01/2019.
2. Extract Month and year from table.date
3. Use 'Group_by' to group datas by month and year. In MATLAB, the function is called grpstats.
4. Pass Grouped datas to 'sapply' function. It is called varfun in MATLAB. Since grpstats calculates mean, it was not necessary to call.
5. Having vector(list) with CPIs in order, run a simple for loop to calculate monthly IR. And plot.

```
mth = month(T.date);
yr = year(T.date);
[monthlyCPIMean, groupName] = grpstats(T.CPI, [yr, mth], {'mean','gname'})
```

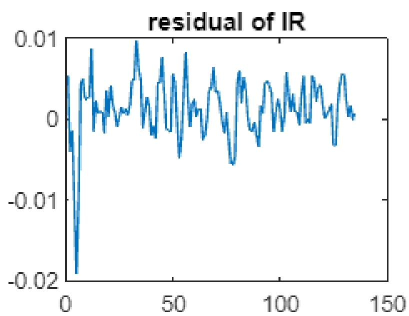
```
monthlyCPIMean = 136×1
  100.0000
  100.5251
  100.1238
   99.9854
   98.9754
   97.0797
   96.0757
   96.4938
   96.9737
   97.2095
      ⋮

groupName = 136×2 cell
'2008'      '7'
'2008'      '8'
'2008'      '9'
'2008'      '10'
'2008'      '11'
'2008'      '12'
'2009'      '1'
'2009'      '2'
'2009'      '3'
```

```
'2009'      '4'
    ⋮
```

```
plot(monthlyCPIMean)
title('IR')
```




```
IR = zeros(length(monthlyCPIMean)-1,1);
for i = 1:length(monthlyCPIMean)-1
    IR(i) = (monthlyCPIMean(i+1) - monthlyCPIMean(i))/monthlyCPIMean(i);
end

plot(IR)
title('residual of IR')
```



Q2: Describe how the data has been detrended and plot

Since regression was already performed on CPI data, I would assume that the data that the question is referring is IR data. If one is interested in detrending of CPI data, refer to Q0.

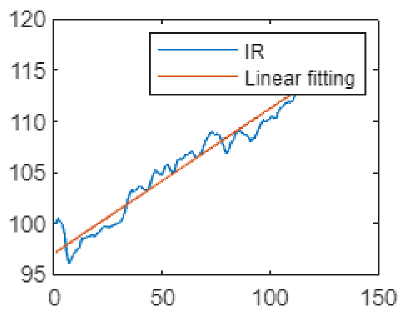
Since mathematical backgrounds of regressions were already inroduced previously, without redundancy, regression would be performed. Please recall the expression given already.

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}.$$ In Matrix form, let's call $Y = A * \beta$

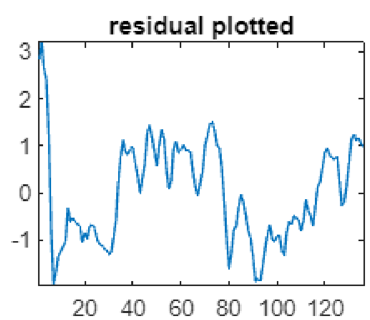
I detrended residual graph by finding inverse of matrix A after projecting Y and A into the column space of A.

AKA $(A^T A)^{-1} A^T Y = \beta$.

```
x = [ones(size(monthlyCPIMean,1),1), (1:length(monthlyCPIMean))'];
y = monthlyCPIMean;
b = x\y;

figure()
plot(monthlyCPIMean)
hold on
plot(x*b)
hold off
legend('IR','Linear fitting')
```




```
figure()
residual = y - x*b;
plot(residual)
axis('tight')
title('residual plotted')
```



Notice that the linear fitting seems to fail reduce the magnitide of oscillation.

Q3: Statement of and justification for the chosen AR(p) model

1. Includes: plots and reasoning

After searching for methodology online, it is recommended to use PCAF(partial Autocorrelation function) in conjunction with Yule-Walker equations. The estimation of the PACF involves solving Yule-Walker equations with respect to the autocorrelations. PCAF measures the correlation between y_t and $L^k y_t$ after adjusting for the linear effects of $L^k y_t$. L is a lag operator introduced previously.

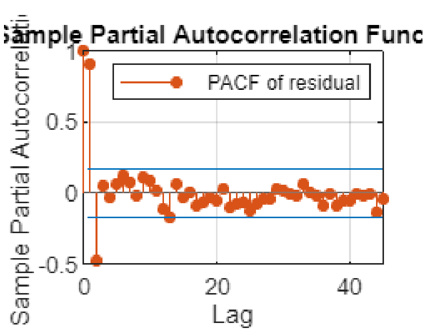
Introducing Yule-Walker equations, the equation can be written as $\gamma_m = \sum_{k=1}^{p} \varphi_k \gamma_{m-k} + \sigma_\epsilon^2 \delta_{m,0}$ where m =0, ..., p. To see how the equations yields p+1 equations, the equation was written in a matrix format.

γ_m is the autocovariance function of X_t, σ_ϵ is the standard deviation of the input noise processes, and δ is a Kronecker delta.

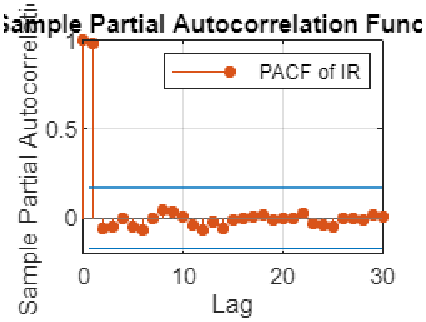
$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix} = \begin{bmatrix} \gamma_0 & \gamma_{-1} & \gamma_{-2} & \cdots \\ \gamma_1 & \gamma_0 & \gamma_{-}1 & \cdots \\ \gamma_2 & \gamma_1 & \gamma_0 & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \vdots \\ \varphi_p \end{bmatrix}. \quad \text{and} \quad \gamma_0 = \sum_{k=1}^{p} \varphi_k \gamma_{-k} + \sigma_\epsilon^2$$

This equation provides p + 1 equations for a model with p+1 unknowns($\varphi's$ and σ) and none of them are redundant as each equations involves a new term. This concludes the reasoning for choosing AR(p) model.

```
parcorr(residual,Method="yule-walker", numlags = 45);
legend('PACF of residual')
```




```
parcorr(monthlyCPIMean,Method="yule-walker", numlags = 30);
legend('PACF of IR')
```



Analyzing with PCAF revealed AR(2) for residual(detrended data) and AR(1) for IR data. Therefore, one can come up with following scheme;

$$\text{IR} = \text{AR(2) model} + \text{linear trend} + \text{other residual}$$

Therefore, this paper procedes with AR(2) model

Q4: Describe the final model;

1. Model specification
2. Plot 1month-ahead forecasts
3. Plot prediction and data together

Let's use the first 100 months for training and predict IR of 101 st month.

```
cutoff = 100
```

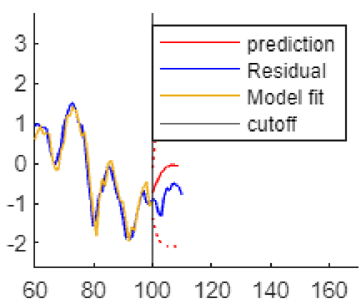
```
cutoff = 100
```

```
AR = arima(2,0,0);
AR_residual = estimate(AR, residual(1:cutoff))
```

```
    ARIMA(2,0,0) Model (Gaussian Distribution):

                  Value      StandardError    TStatistic      PValue
                _________    _____________    __________    ___________

    Constant    -0.017102      0.034547        -0.49503        0.62058
    AR{1}          1.4948      0.059866          24.969    1.3409e-137
    AR{2}        -0.62139      0.057431          -10.82     2.7739e-27
    Variance      0.10064      0.013149          7.6539     1.9501e-14
AR_residual = 
  arima with properties:

     Description: "ARIMA(2,0,0) Model (Gaussian Distribution)"
    Distribution: Name = "Gaussian"
               P: 2
               D: 0
               Q: 0
        Constant: -0.0171017
              AR: {1.49476 -0.621394} at lags [1 2]
             SAR: {}
```

```
             MA: {}
            SMA: {}
    Seasonality: 0
           Beta: [1×0]
       Variance: 0.10064
```

```
leftover = infer(AR_residual, residual(1:cutoff));
y_hat2 = residual(1:cutoff) - leftover;

[YF,YMSE] = forecast(AR_residual,10,residual(1:cutoff));

figure()
hold on
plot(100:109, YF,'r')
plot(residual(1:110),'b')
plot(y_hat2)
xline(cutoff)
plot(100:109,YF +1.96*sqrt(YMSE),'r:')
plot(100:109,YF -1.96*sqrt(YMSE),'r:')
k = legend('prediction','Residual','Model fit','cutoff',Location='northeast');
hold off

xlim([60 170])
ylim([-2.6 3.8])
```



Notice that the red dotted lines are 95 confidence intervals for forecasting.

This paper reports the results as follows;

The model found:

$$\mathrm{AR}(2) : \left(1 - L\varphi_1 - L^2\varphi_2\right)X_t = (1 - 1.49476L + 0.621394L^2)X_t =$$

$$W(0, 0.10064^2)_t - 0.0171017$$

```
linear_predict = x*b;
```

```
fprintf('The model predicted 101th month\''s as: %f ', linear_predict(101) + YF(1))
```

```
The model predicted 101th month's as: 110.650578
```

```
fprintf('whereas the data indicates: %f ', monthlyCPIMean(101))
```

```
whereas the data indicates: 110.471900
```

Q5: Which AR(p) model to choose

1. Include a plot of the RSME against different lags p for the model

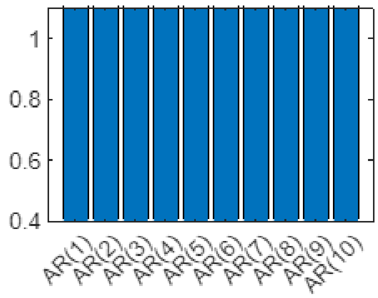
In conjunction with Yule-Walker equation, this paper stated why would PCAF give sufficient information to determine the degree of AR models.

Let's approach model selection based on RMSE this time. Like before, the training sets are 100 months of IR datas.

```
 clear AR AR_residual leftover y_hat2 y_hat YF YMSE
error_mat = zeros(30,10);
% for i = 1:10
%     AR = arima(i,0,0);
%     AR_residual = estimate(AR, residual(1:cutoff));
%     YF = forecast(AR_residual,30,residual(1:cutoff));
%     error_mat(:,i) = linear_predict(100:130-1) + YF;
% end
disp(rmse(monthlyCPIMean(100:130-1),error_mat))
```

```
  113.1257   113.1257   113.1257   113.1257   113.1257   113.1257   113.1257   113.1257   113.1257   113.1257
```

```
X = NaN(1,10);
X = categorical(X);
for i =1:10
    X(i) = sprintf("AR(%d)",i);
end
graph1 = bar(X,rmse(monthlyCPIMean(100:130-1),error_mat));

ylim([0.40 1.1 ])
```



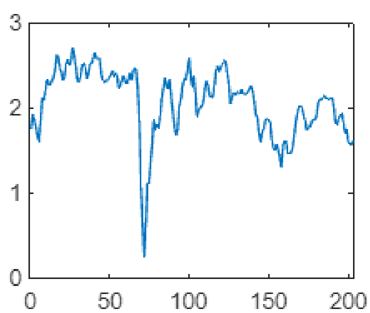
From AR(1) to AR(10) models, RMSE errors are listed as followed. Notice that I predicted upto 30 months in the future, which would explain the high level of errors in general. Secondly, the RMSE error does not monotonically decreases with increasing the order since higher than necessary orders would overfit the errors in training data set and wrongly predict.

In fact, the AR(10) model has the highest RMSE error.

Q6: Overlay your estimates

1. plot IR data/predictions/BER

```
clear T2 mth yr groupName residual
T2 = readtable('T10YIE.csv');
T2 = rmmissing(T2);

mth = month(T2.DATE);
yr = year(T2.DATE);

[monthlyBERmean, groupName] = grpstats(T2.T10YIE, [yr, mth], {'mean','gname'});
plot(monthlyBERmean)
```




```
BER = zeros(length(monthlyBERmean)-1,1);
for i = 1:length(monthlyBERmean)-1
    BER(i) = (monthlyBERmean(i+1) - monthlyBERmean(i))/monthlyBERmean(i);
end
```

```
cutoff = 128;
x = [ones(cutoff,1), (1:cutoff)'];
y = IR(1:cutoff);
b = x\y;
residual = y - x*b
```

```
residual = 128×1
    0.0047
   -0.0045
   -0.0019
   -0.0107
   -0.0197
   -0.0109
    0.0038
    0.0044
    0.0018
    0.0019
      ⋮
```

```
clear linear_predict
linear_predict = x*b
```

```
linear_predict = 128×1
    0.0005
    0.0005
    0.0005
    0.0006
    0.0006
    0.0006
    0.0006
    0.0006
    0.0006
    0.0006
      ⋮
```

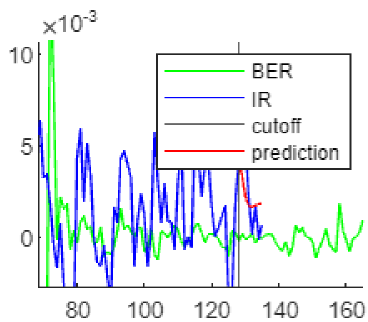
```
AR = arima(2,0,0);
AR_residual = estimate(AR, residual)
```

```
    ARIMA(2,0,0) Model (Gaussian Distribution):

                   Value       StandardError    TStatistic      PValue
                 __________    _____________    __________    __________

    Constant     3.0882e-05     0.00028805       0.10721         0.91462
    AR{1}           0.65816       0.071846        9.1608      5.1528e-20
    AR{2}          -0.24811       0.070262       -3.5312      0.00041364
    Variance        9.33e-06    5.1659e-07        18.061      6.4957e-73
AR_residual = 
  arima with properties:

     Description: "ARIMA(2,0,0) Model (Gaussian Distribution)"
    Distribution: Name = "Gaussian"
               P: 2
               D: 0
               Q: 0
        Constant: 3.08817e-05
              AR: {0.658164 -0.24811} at lags [1 2]
```

```
           SAR: {}
            MA: {}
           SMA: {}
   Seasonality: 0
          Beta: [1×0]
      Variance: 9.32997e-06
```

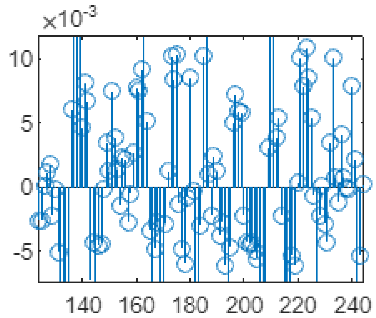
```
leftover = infer(AR_residual, residual);
y_hat2 = residual - leftover;
[YF,~] = forecast(AR_residual,135-cutoff,residual);
```

```
figure()
% plot(BER)
hold on
% plot(BER)
plot(BER/100,'g')
plot(IR,'b')
xline(cutoff)
clear x
x = [ones(7,1), (cutoff+1:length(IR))'];
plot(cutoff:length(IR), [IR(cutoff);(x*b+YF)],'r')
hold off
legend('BER','IR','cutoff', 'prediction')

xlim([68.7 165.0])
ylim([-0.0027 0.0108])
```



As I do not understand what three graphs were supposed to be plotted, I plotted BER rate, IR rate, and AR(2) model fitted by IR.

```
stem(xcov(IR,BER))
xlim([124 244])
ylim([-0.0074 0.0118])
```

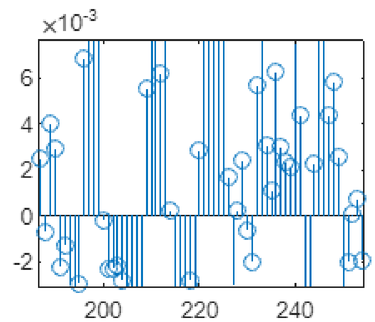


Since I didn't see a clear pattern between IR rate and BER rate, I plotted covariance between them. It seems like there are a seasonal pattern hiding.

Written Assignment #3

Q1: Plot the cross correlation function between the and inflation rate

```
xdata = xcorr(IR,BER);
stem(xdata)

xlim([186.6 254.1])
ylim([-0.0031 0.0077])
```



What is shown is a magnified portion of cross-correlation function between IR and BER. For me, the lag term is not super concise, but I occasionally counted 12 terms before sin-like curve ends. And it does make sense to have a strong seasonal lag with 12, such as black friday sale and so on.

Q2: Fit SARIMAX

```
data = [IR(1:cutoff),BER(1:cutoff)];

clear AR AR_residual

AR = SARIMAX_data1;
```

```
Unrecognized function or variable 'SARIMAX_data1'.
```

```
AR_residual = estimate(AR, IR)
leftover = infer(AR_residual, IR);
y_hat2 = IR - leftover;
```

```
[YF,~] = forecast(AR_residual,5,IR);
```

```
cutoff = 128;
x = [ones(cutoff,1), (1:cutoff)'];
y = IR(1:cutoff);
b = x\y;
residual = y - x*b
clear linear_predict
linear_predict = x*b


figure()
hold on
plot(IR,'b')
xline(cutoff)
clear x
x = [ones(5,1), (cutoff:cutoff+4)'];
plot(cutoff:cutoff+5, [IR(cutoff);(x*b+YF)],'r')
hold off
legend('IR','cutoff', 'prediction')

xlim([88.1 175.1])
ylim([-0.0063 0.0123])
```

This papaer reports SARIMAX model of;

$$\left(1-\phi_1 L-\phi_2 L^2\right)(1-L^{12})y_t = c + X_1\beta_1 + \epsilon_t =$$

$$(1-0.672424L+0.289171L^2)(1-L^{12})y_t = 0.00082287 - 0.000870617X_1 + \epsilon(0,(8.4e-06)^2)_t$$

For prediction;

```
fprintf('the prediction is %f', (linear_predict(1)+YF(1)))
fprintf('whereas the actual data is %f', IR(101))
```

Q3: RMSE

This paper reports rmse between IR data and SARIMAX model to be;

```
rmse_data = vertcat(y_hat2(1:cutoff), x*b+YF)
fprintf('The RMSE error is : %.10f',rmse(IR(1:133),rmse_data))
```

Q4: Improving Your Model

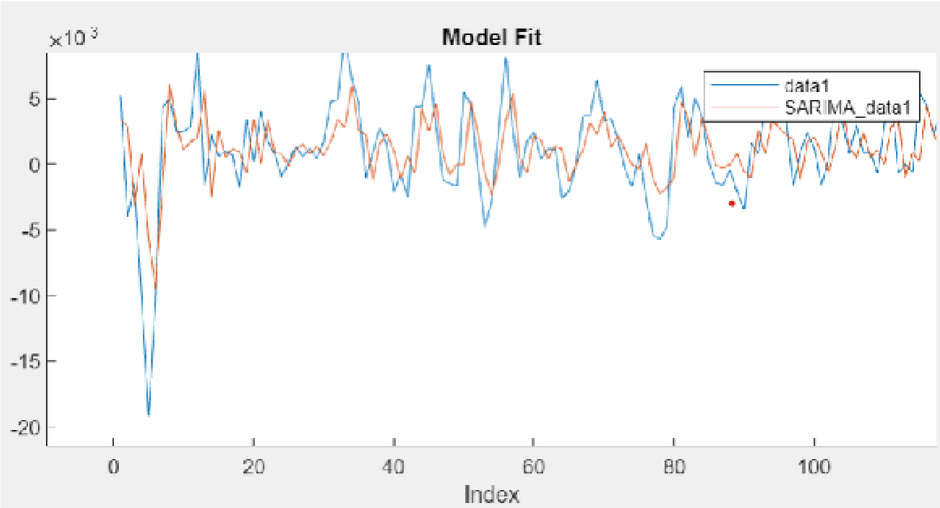
1. What are other steps to improve?
2. What is the smallest error you can obtain?

3. Describe the model that performs best

There were several arima parameters that I have not considered. The arima models allows lags for $X_t, \epsilon_t,$ and degree of integration, along with seasonal parameters with exogenous parameter called betas.

As I am not fully comprehend the dynamics of parameters, I wasn't capable of exploring. But I would imagine that with sufficient tiral and errors, one can come up with a better model if such parameters are adequately investigated.

After trying several models, following model came out to be the most successful.



Parameters

Parameter	Value	Standard Error
Constant	0.0007	0.0003
AR{1}	0.6621	0.0709
AR{2}	-0.2463	0.0711
Variance	0.0000	0.0000

```
clear AR AR_residual
AR = SARIMAX_data1;
AR_residual = estimate(AR, IR)
leftover = infer(AR_residual, IR);
y_hat2 = IR - leftover;
fprintf('RMSE error: %.10f', rmse(y_hat2,IR))
```