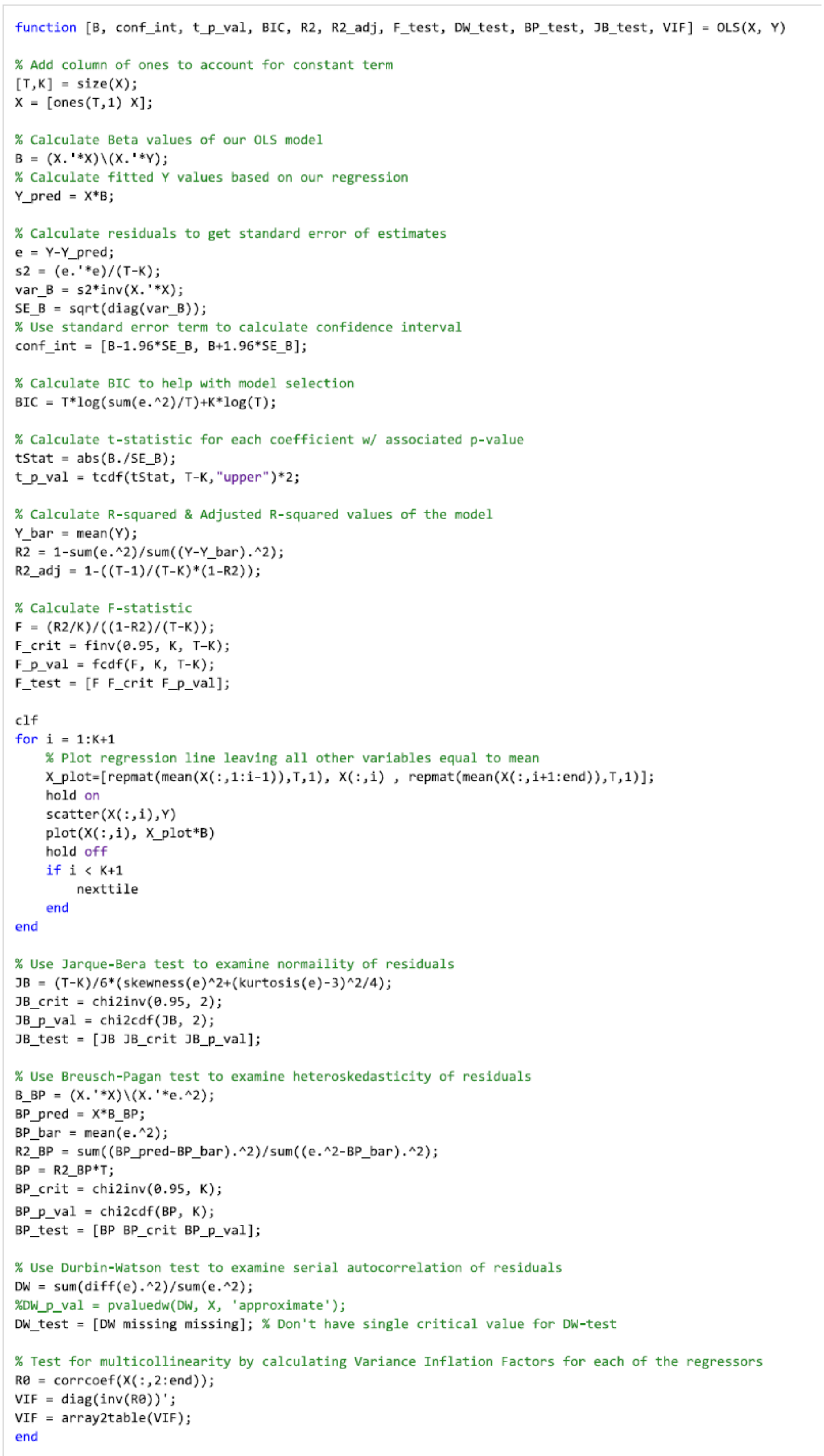
FIN41660: Financial Econometrics – Individual Assignment

Part 1: MATLAB FUNCTIONS



```matlab
function [B, conf_int, t_p_val, BIC, R2, R2_adj, F_test, DW_test, BP_test, JB_test, VIF] = OLS(X, Y)

% Add column of ones to account for constant term
[T,K] = size(X);
X = [ones(T,1) X];

% Calculate Beta values of our OLS model
B = (X.'*X)\(X.'*Y);
% Calculate fitted Y values based on our regression
Y_pred = X*B;

% Calculate residuals to get standard error of estimates
e = Y-Y_pred;
s2 = (e.'*e)/(T-K);
var_B = s2*inv(X.'*X);
SE_B = sqrt(diag(var_B));
% Use standard error term to calculate confidence interval
conf_int = [B-1.96*SE_B, B+1.96*SE_B];

% Calculate BIC to help with model selection
BIC = T*log(sum(e.^2)/T)+K*log(T);

% Calculate t-statistic for each coefficient w/ associated p-value
tStat = abs(B./SE_B);
t_p_val = tcdf(tStat, T-K,"upper")*2;

% Calculate R-squared & Adjusted R-squared values of the model
Y_bar = mean(Y);
R2 = 1-sum(e.^2)/sum((Y-Y_bar).^2);
R2_adj = 1-((T-1)/(T-K)*(1-R2));

% Calculate F-statistic
F = (R2/K)/((1-R2)/(T-K));
F_crit = finv(0.95, K, T-K);
F_p_val = fcdf(F, K, T-K);
F_test = [F F_crit F_p_val];

clf
for i = 1:K+1
    % Plot regression line leaving all other variables equal to mean
    X_plot=[repmat(mean(X(:,1:i-1)),T,1), X(:,i) , repmat(mean(X(:,i+1:end)),T,1)];
    hold on
    scatter(X(:,i),Y)
    plot(X(:,i), X_plot*B)
    hold off
    if i < K+1
        nexttile
    end
end

% Use Jarque-Bera test to examine normaility of residuals
JB = (T-K)/6*(skewness(e)^2+(kurtosis(e)-3)^2/4);
JB_crit = chi2inv(0.95, 2);
JB_p_val = chi2cdf(JB, 2);
JB_test = [JB JB_crit JB_p_val];

% Use Breusch-Pagan test to examine heteroskedasticity of residuals
B_BP = (X.'*X)\(X.'*e.^2);
BP_pred = X*B_BP;
BP_bar = mean(e.^2);
R2_BP = sum((BP_pred-BP_bar).^2)/sum((e.^2-BP_bar).^2);
BP = R2_BP*T;
BP_crit = chi2inv(0.95, K);
BP_p_val = chi2cdf(BP, K);
BP_test = [BP BP_crit BP_p_val];

% Use Durbin-Watson test to examine serial autocorrelation of residuals
DW = sum(diff(e).^2)/sum(e.^2);
%DW_p_val = pvaluedw(DW, X, 'approximate');
DW_test = [DW missing missing]; % Don't have single critical value for DW-test

% Test for multicollinearity by calculating Variance Inflation Factors for each of the regressors
R0 = corrcoef(X(:,2:end));
VIF = diag(inv(R0))';
VIF = array2table(VIF);
end
```

Part 2: Empirical Analysis

For our analysis we will be looking at UN life expectancy data, and consider data relating to the year 2015. The data was sourced online from: https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who

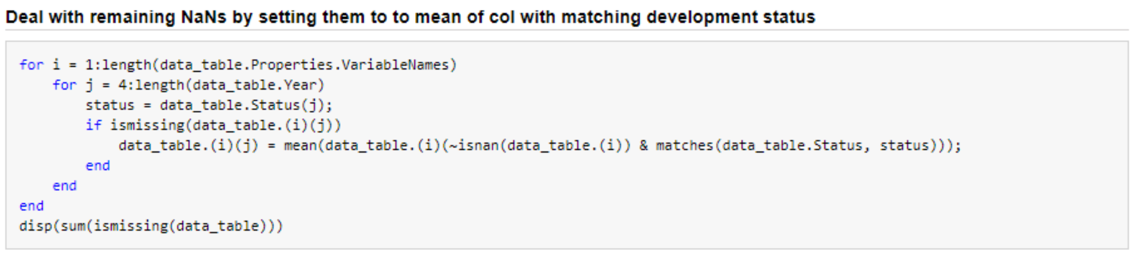
The data contains information pertaining to 183 countries, with the following 22 variables:

Year, Development Status, Life Expectancy, Adult Mortality, Infant Deaths, Alcohol Consumption, Health Expenditure as % of GDO, Hepatitis B Immunisation Coverage, Measles Prevalence, BMI, Deaths under 5 Years old, Prevalence of Polio, Health Expenditure as % of Total Expenditure, Diphtheria Immunisation Coverage, Prevalence of HIV/AIDS, GDP, Population, Prevalence of Thinness among 10-19 Year olds, Prevalence of Thinness among 5-9 Year olds, HDI in terms of Income Composition of Resources & Number of Years of Schooling.

We are looking to build a regression model to predict life expectancy based on the other factors.

Data Preparation:

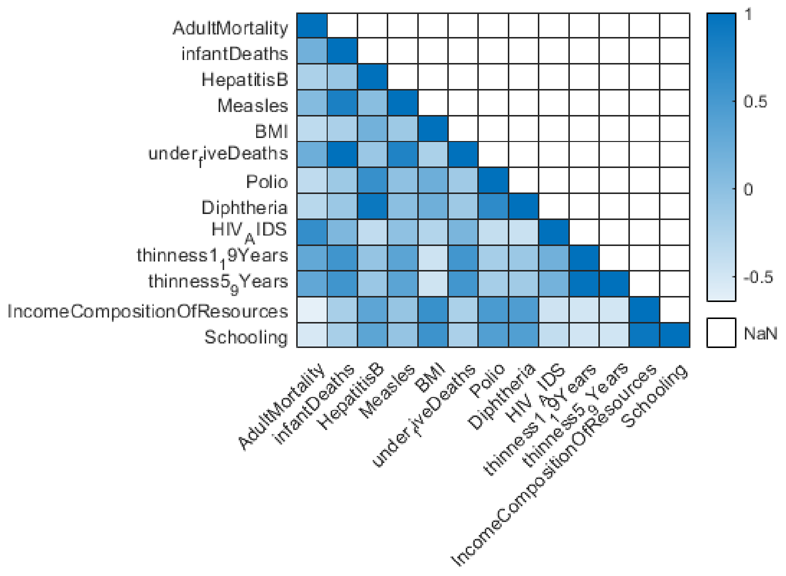
Before we could run our linear regression model, we first had to prepare the data. There was a large number of missing values in the data, so the first step was to deal with these. Columns with more than 15% NA values for omitted from further consideration. The remaining missing values were replaced by the mean of their respective column given the country's UN development status (whether the country was a "Developed" or "Developing" country).



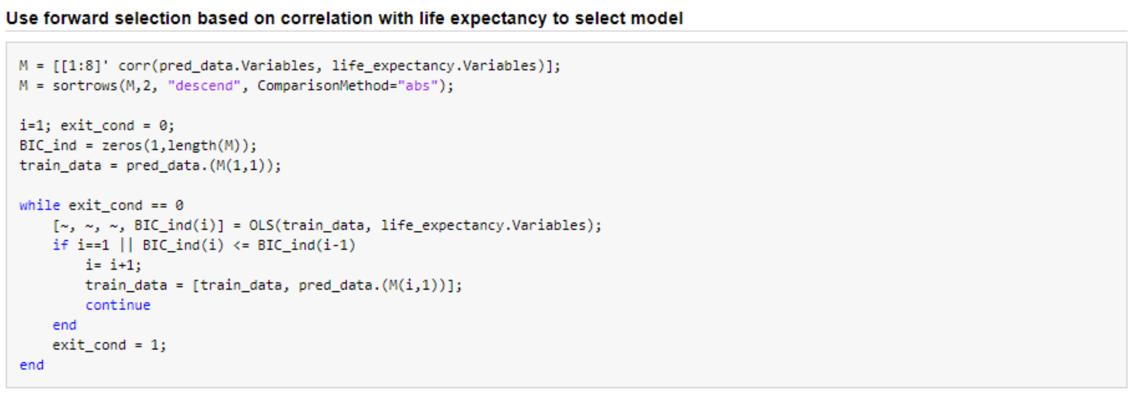
Deal with remaining NaNs by setting them to to mean of col with matching development status

```
for i = 1:length(data_table.Properties.VariableNames)
    for j = 4:length(data_table.Year)
        status = data_table.Status(j);
        if ismissing(data_table.(i)(j))
            data_table.(i)(j) = mean(data_table.(i)(~isnan(data_table.(i)) & matches(data_table.Status, status)));
        end
    end
end
disp(sum(ismissing(data_table)))
```

In order to aid in the interpretation of our data/model, predictor variables which were highly correlated with other predictor variables were removed from the data. More sophisticated methods such as Principal Components Analysis could have been used for this, but I felt that this would hinder interpretation of regression coefficients. This led to the Income Composition, Thinness among 5-9 year olds, Measles Prevalence, Deaths under 5 & Diphtheria Immunization columns being removed.



We then ranked the predictor variables in terms of their correlation with life expectancy, and then used this ordering to inform a forward stepwise model selection procedure based on minimisation of BIC values. I chose BIC as our criterion here as I felt that it appropriately penalizes model for using excessive parameters, while also ensuring the model captures as much of the variance as possible.

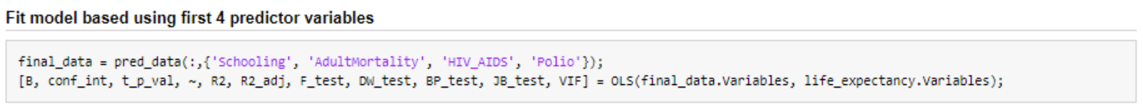


Use forward selection based on correlation with life expectancy to select model

```
M = [[1:8]' corr(pred_data.Variables, life_expectancy.Variables)];
M = sortrows(M,2, "descend", ComparisonMethod="abs");

i=1; exit_cond = 0;
BIC_ind = zeros(1,length(M));
train_data = pred_data.(M(1,1));

while exit_cond == 0
    [~, ~, ~, BIC_ind(i)] = OLS(train_data, life_expectancy.Variables);
    if i==1 || BIC_ind(i) <= BIC_ind(i-1)
        i= i+1;
        train_data = [train_data, pred_data.(M(i,1))];
        continue
    end
    exit_cond = 1;
end
```

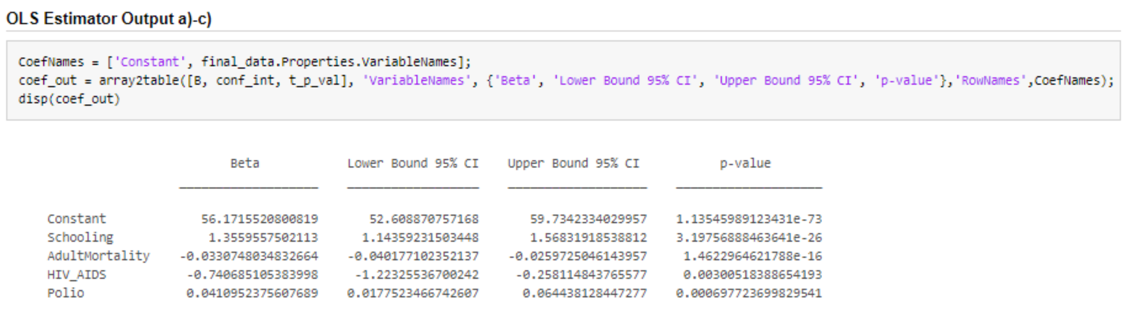
Through this, we found the optimal trade-off between including additional parameters in our model, and model accuracy was found by including the following 4 predictor variables: Years of Schooling, Adult Mortality, Prevalence of HIV/AIDS & Prevalence of Polio.

We then fitted a regression model using the aforementioned parameters using our OLS function outlined in **Part 1**, resulting in the following outputs:



Fit model based using first 4 predictor variables

```
final_data = pred_data(:,{'Schooling', 'AdultMortality', 'HIV_AIDS', 'Polio'});
[B, conf_int, t_p_val, ~, R2, R2_adj, F_test, DW_test, BP_test, JB_test, VIF] = OLS(final_data.Variables, life_expectancy.Variables);
```



OLS Estimator Output a)-c)

```
CoefNames = ['Constant', final_data.Properties.VariableNames];
coef_out = array2table([B, conf_int, t_p_val], 'VariableNames', {'Beta', 'Lower Bound 95% CI', 'Upper Bound 95% CI', 'p-value'},'RowNames',CoefNames);
disp(coef_out)
```

```
                             Beta            Lower Bound 95% CI    Upper Bound 95% CI          p-value
                      __________________    __________________    __________________    ____________________

    Constant            56.1715520800819       52.608870757168      59.7342334029957    1.13545989123431e-73
    Schooling            1.3559557502113      1.14359231503448      1.56831918538812    3.19756888463641e-26
    AdultMortality   -0.0330748034832664   -0.0401771023521370   -0.0259725046143957     1.4622964621788e-16
    HIV_AIDS          -0.740685105383998     -1.22325536700242    -0.258114843765577     0.00300518388654193
    Polio             0.0410952375607689    0.0177523466742607     0.064438128447277    0.000697723699829541
```

From the p-values outlined in our output we can see that each coefficient is statistically significant at the 1% level. The coefficient of our model implies a baseline global life expectancy of 56.17 years, i.e. that setting all other variables equal to 0 the average person in 2015 would have a life expectancy of 56.17.

The marginal effect of years of schooling has a positive coefficient of 1.356, implying that for each additional year of schooling a person can expect to live 1.35 years longer. The 95% confidence interval surrounding the coefficient estimate, with a lower bound of 1.4, still shows a strong positive effect of education on life expectancy, giving us plenty of evidence for a relationship between the variables.

As would be expected, adult mortality & the prevalence of HIV/AIDs, both had a negative relationship with life expectancy. The relatively low coefficient for adult mortality was more down to the scale of the variable (mean value ≈ 150), rather than any sort of weakness of the relationship. Both estimates suggest that as the adult mortality or the prevalence of HIV/AIDs in a country increases, the life expectancy of its populace decreases.

The marginal effect of Polio immunization was 0.041, implying a positive correlation with life expectancy. Similar to above, the low parameter estimate was in large part due to the variables scale. The figure suggests that each % increase in the immunization coverage for Polio increases, the life expectancy increases by 0.041 years.

R-Squared Output d)

```
R2_out = array2table([R2, R2_adj], 'VariableNames',{'R-Squared','Adjusted R-Squared'});
disp(R2_out)
```

```
       R-Squared          Adjusted R-Squared
    _________________     __________________

    0.827697450660637     0.824809698437073
```

We calculated our model's R^2 value to be 0.8277. This implies that our model explains roughly 82.77% of the variance seen within the data. Such a high R2 value lends credence to our model, and it suggest that we are capturing most of the variance in the data.

We have also calculated an adjusted R2 value of 0.8248. Unlike the traditional R2 measure which is monotonically non-decreasing with the number of parameters, adjusted R2 seeks to penalize models for including additional/redundant parameters. The fact that our adjusted R2 value and R2 values are so similar gives us confidence that we are not including redundant parameters in our model.

Residual Analysis Output e) & g)

```
res_out = array2table([F_test; DW_test; BP_test; JB_test], 'VariableNames',{'Test Statistic', 'Critical Value*', 'p-value'}, ...
    'RowNames',{'F-Statistic', 'Durbin-Watson', 'Breusch-Pagan', 'Jacque-Bera'});
res_out = convertvars(res_out, @isnumeric, @nanblank);

disp(res_out)
```

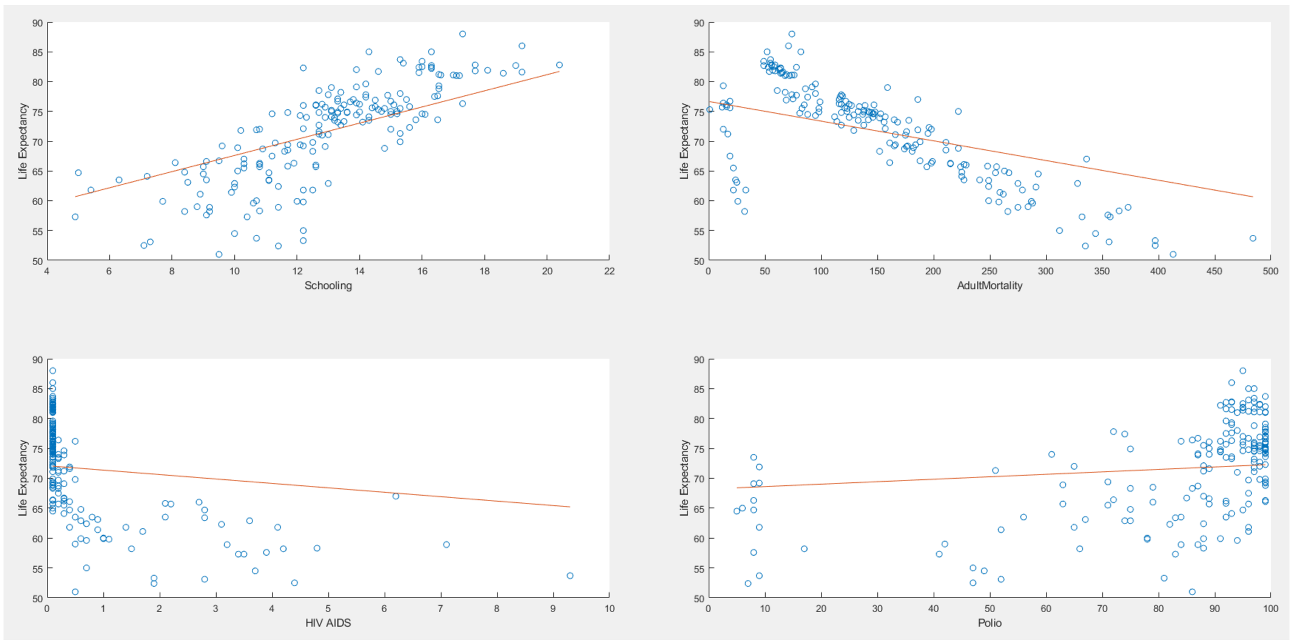
```
                      Test Statistic     Critical Value*    p-value
                     ________________    _______________    _______

    F-Statistic      214.967573370673        2.4221         1
    Durbin-Watson    2.00503863657107
    Breusch-Pagan    22.3694775527474        9.4877         0.99983
    Jacque-Bera      1.81794381840244        5.9915         0.59706
```

The F-test is designed to test the overall significance of the parameters in our model. The calculated F-statistic for our model of 214.968, which is greater than the critical value of 2.4221, meaning that our estimated coefficients are jointly significant at the 5% level. As such, we conclude that the parameter estimates are jointly statistically different from 0, and that our model is useful in predicting life expectancy.

The Durbin-Watson statistic of our model is 2.005 ≈ 2, meaning we fail to reject the null hypothesis of no serial autocorrelation between our residuals.

In order to test for heteroskedasticity within our residuals, we calculate the Breusch-Pagan test statistic of 22.3765. As this exceeds our critical value of 9.4877, we reject the null hypothesis of heteroskedasticity and conclude that our residuals are distributed with constant variance.

The Jarque-Bera test examines the normality of our residuals. We calculate the test statistic to be 1.8179. As this lies below our critical value, we fail to reject the null hypothesis of our residuals being distributed with skewness 0 & kurtosis 3, and conclude that our residuals are normally distributed.



In the above plots, we show the distribution of life expectancy with respect to each of the predictor variables in our model. We also plot the regression line of each variable, assuming all other variables are kept constant at their mean values. From the plots, we can see that the first two variables' relationship with life expectancy is well approximated by a linear assumption, with the regression line passing through most of the mass of the data.

The relationship between the variables in the 2nd set of plots is less clear, as we see that there is a high density of points at the extremes of each, with relatively sparse areas between them, making it hard to determine how well the relationship is approximated by a line. Overall though, a linear relationship seems at least somewhat appropriate for each variable.

The final condition we want to test our model on, is the presence of multicollinearity within our data, which could lead to increased variance around the estimation of the parameters of our model, and potentially make our model extremely sensitive to changes in model specification. To examine the degree of multicollinearity within our data, we have calculated the Variance Inflation Factors seen below.

Multicollinearity Output h)

```
disp(VIF)
```

```
        VIF1                VIF2                VIF3               VIF4
  ________________    ________________    ________________    _______________

  1.52461063542148    1.96740051873801    1.72549637572785    1.3125694653969
```

As the VIFs are all less than 2, we surmise that multicollinearity is not an issue within our model, and that the variance of our parameter estimates is not being affected by the degree of correlation between our variables.

Part 3: ARIMA & Forecasting

The time series I have chosen to examine as part of this section is the FRED-MD: 6-Month Treasury Bill (TB6MS). This is a monthly frequency time series of the Federal Reserve 6-Month T-Bill rate.

The data was imported from the current.csv file on the FED St. Louis website to ensure we have the most up to date data available.

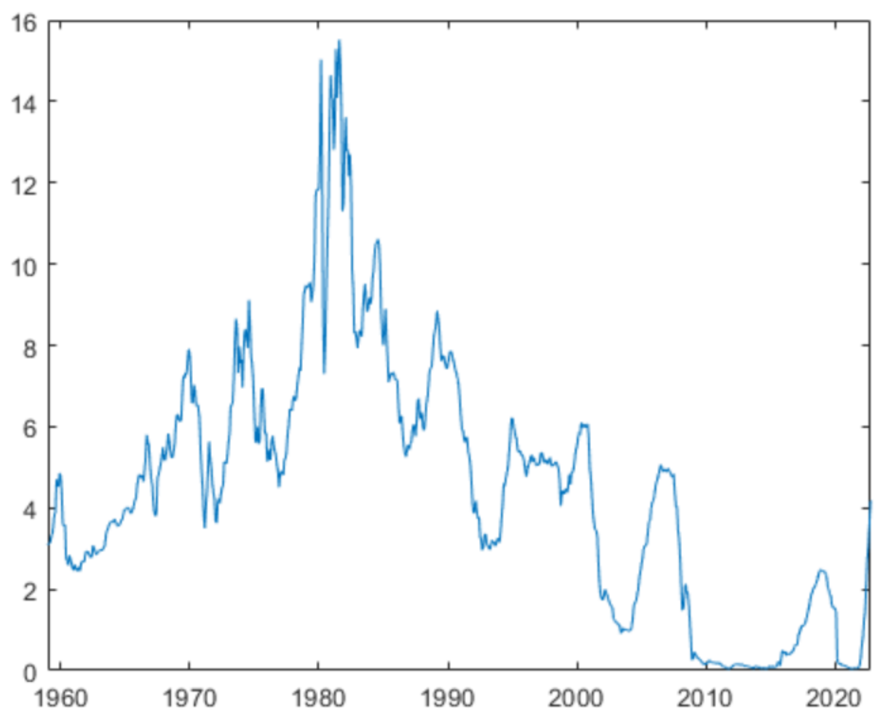
Initial importation of Data

```
clearvars()

% Set import options - Import only columns under consideration in our analysis
opts = detectImportOptions('current.csv');
opts = setvaropts(opts,'sasdate','InputFormat','MM/dd/uuuu');
opts.SelectedVariableNames = {'sasdate','TB6MS'};

[data] = readtable('current.csv', opts);
data(1,:) = []; % Remove TCode information from dataframe

Date = table2array(data(:,1));
TBill = table2array(data(:,2));
```

An initial plot of the data, along with the ACF & PACF plots (linear decay of ACF function) showed evidence that the time series was integrated and that we would need to difference the series before further analysis could be conducted. To make certain of this, I ran an Augmented Dickey-Fuller test to look for a unit root. We failed to reject the null hypothesis of a unit root within the series and concluded that we would need to difference the series before proceeding further.

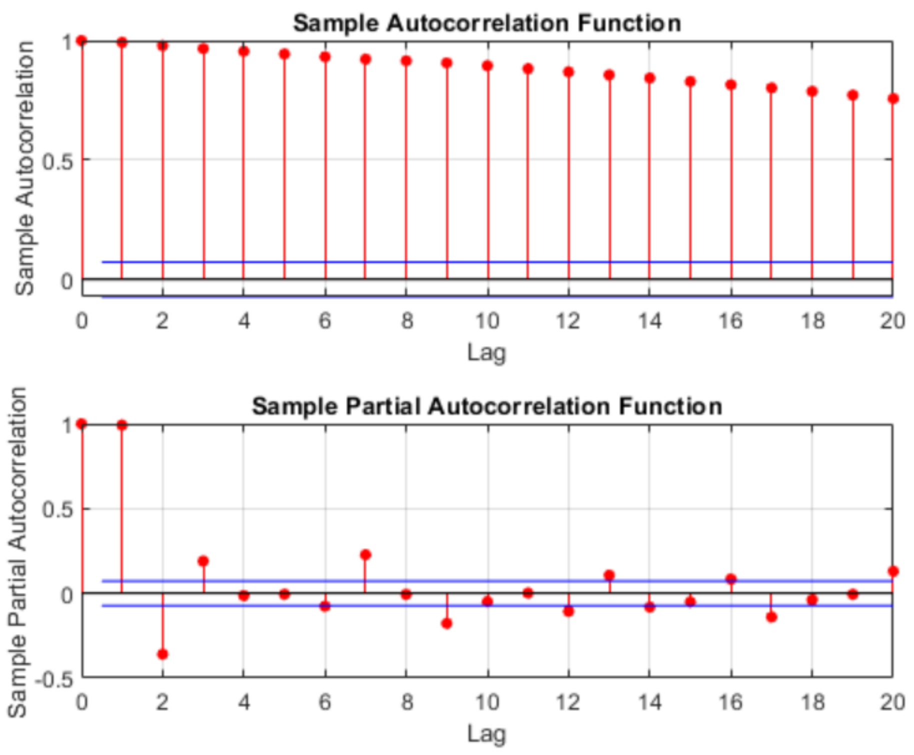
Plot time-series of historical 6-Month Treasury Bill Rate

```
plot(Date, TBill)
```



Plot sample ACF & sample PACF of 6-Month Treasury Bill Rate

```
tiledlayout(2,1)
nexttile
autocorr(TBill)
nexttile
parcorr(TBill)
```



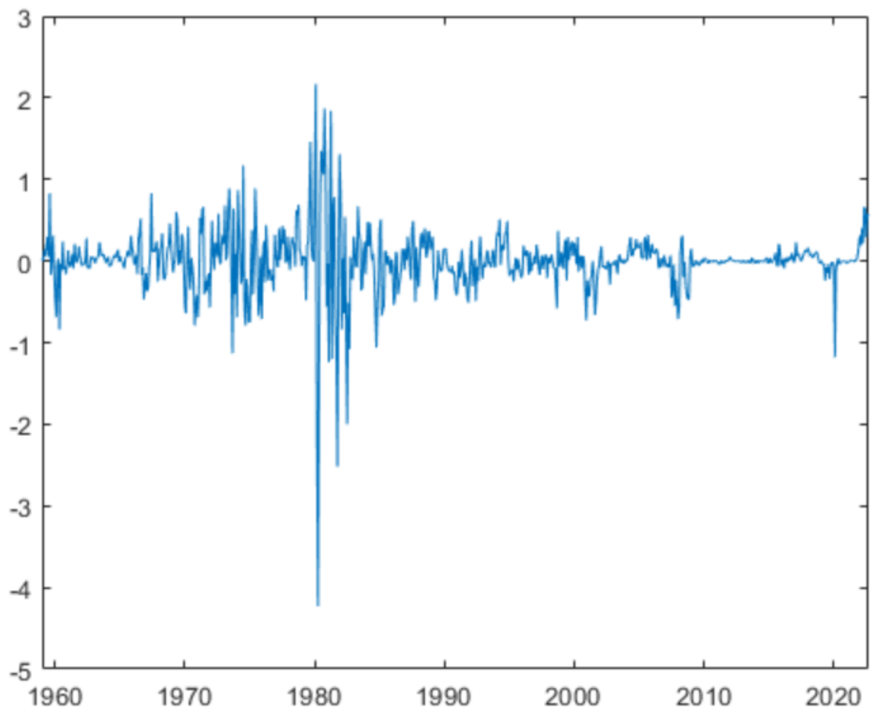
Check for integration using ADF test

```
[h,pValue] = adftest(TBill);
dTBill = diff(TBill);
```

Plotting the differenced series, it was clear that the series was now more in line with what we would expect from a stationary series. After running a second ADF test to confirm the stationarity of the differenced series, we proceeded with our analysis.

Plot differenced series

```
clf
plot(Date(1:end-1), dTBill)
```



The next step was to fit a number of different ARMA processes to the series to determine what type of process we are dealing with. Using a historical time period of 20 years worth of data as input, we fit ARMA models of order 0 to 3 for each of Autoregressive & Moving Average components.

Fit model based on previous 20 years of data

```
TBill_AIC_20Y=zeros(4,4);
TBill_BIC_20Y=zeros(4,4);

options =  optimset('lsqnonlin');
options.Display='none';

for p=0:3
    for q=0:3
    [~, ~, ~, ~, diagnostics] = armaxfilter(dTBill(end-240:end),1,[1:p],[1:q],[],[],options);
    TBill_AIC_20Y(p+1,q+1)=diagnostics.AIC; % store AIC values
    TBill_BIC_20Y(p+1,q+1) = diagnostics.SBIC; % store BIC values
    end
end
```

Based off of the AIC & BIC values, we decided to proceed with the following 3 models:

1. ARMA(1,1)
2. ARMA(1,2)
3. ARMA(3,1).

Examine predictive performance on previous data to determine best model to use.

In the following, we will examine the out-of-sample forecast performance of the model Use a forecast horizon of 3 years (36 months)

```
forecast_horizon = 36;
OOS_iter = length(dTBill)-forecast_horizon;
sample_Data = dTBill(OOS_iter-240:end);
n = length(sample_Data)-forecast_horizon;

% Estimate models based on 20 years of data, excluding most recent year
ToEstMdl_arma11 = arima('ARLags',1,'MALags',1);
[EstMdl_arma11, logL1] = estimate(ToEstMdl_arma11,sample_Data(1:end-forecast_horizon));

ToEstMdl_arma21 = arima('ARLags',1,'MALags',[1:2]);
EstMdl_arma21 = estimate(ToEstMdl_arma21,sample_Data(1:end-forecast_horizon));

ToEstMdl_arma31 = arima('ARLags',[1:3],'MALags',1);
EstMdl_arma31 = estimate(ToEstMdl_arma31,sample_Data(1:end-forecast_horizon));
```

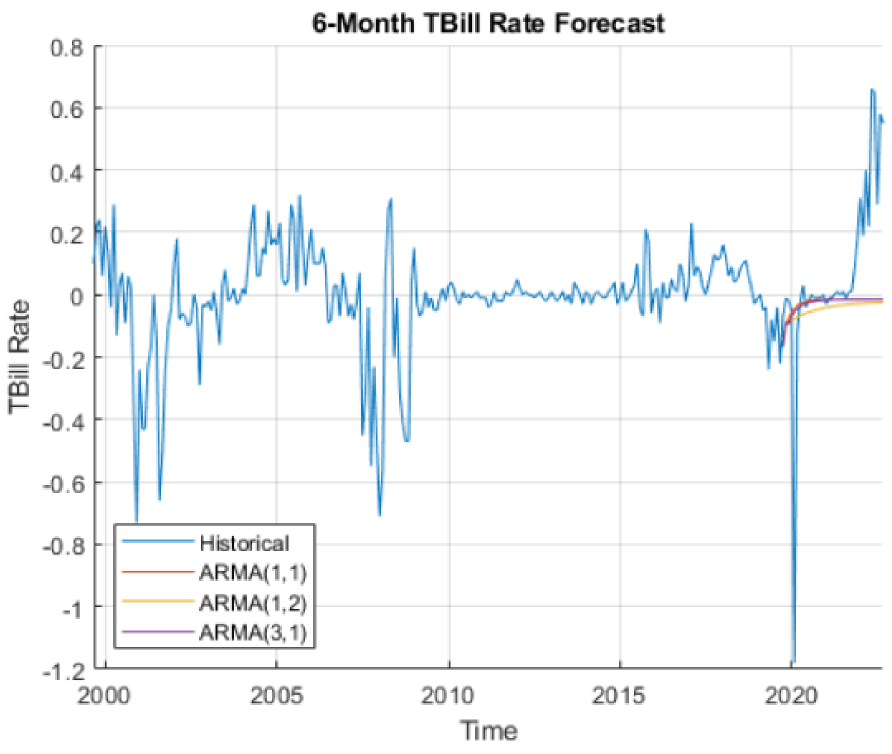
```
    ARIMA(1,0,1) Model (Gaussian Distribution):

                        Value              StandardError            TStatistic              PValue
                  __________________    ____________________    __________________    __________________

    Constant      -0.0046801696514501     0.00926580323432107    -0.505101342333116      0.613487658618503
    AR{1}           0.697653421645259      0.0469647222472488      14.8548397235784                      0
    MA{1}           -0.12130823108142       0.063556479268705     -1.90866820310406     0.0563049074952773
    Variance       0.0163536257914016    0.000920384839117849      17.7682476898206                      0


    ARIMA(1,0,2) Model (Gaussian Distribution):

                        Value               StandardError            TStatistic                PValue
                  ___________________    ____________________    __________________    ____________________

    Constant      -0.00218890310020238     0.00435680940911772    -0.502409652261023       0.615379390781044
    AR{1}            0.903975544864572       0.037083684774251      24.3766376067419                       0
    MA{1}           -0.314335200796737      0.0527122869732793      -5.963224493676     2.47308262757429e-09
    MA{2}           -0.284234971841262      0.0494697412515374     -5.74563287881415     9.15778075594176e-09
    Variance        0.0158343700647712    0.000899781037841501      17.5980259628015                       0


    ARIMA(3,0,1) Model (Gaussian Distribution):

                        Value               StandardError            TStatistic                PValue
                  ___________________    ____________________    __________________    ____________________

    Constant      -0.00665487222310807      0.0149975389643844    -0.443730950718771       0.657237115572382
    AR{1}            0.179680773328985       0.221604407045078      0.81081768961587       0.417470374268221
    AR{2}             0.20237104999148       0.137685030309508      1.46981156583662       0.141612795830272
    AR{3}            0.170527406019753      0.0404442355075062      4.21635874383395     2.48278684804593e-05
    MA{1}            0.425678443108293       0.218556488214702      1.94768156546385      0.0514530772298478
    Variance         0.015935061060378    0.000908111460796653       17.547472692833                       0
```

Calculate the forecasts for each model.

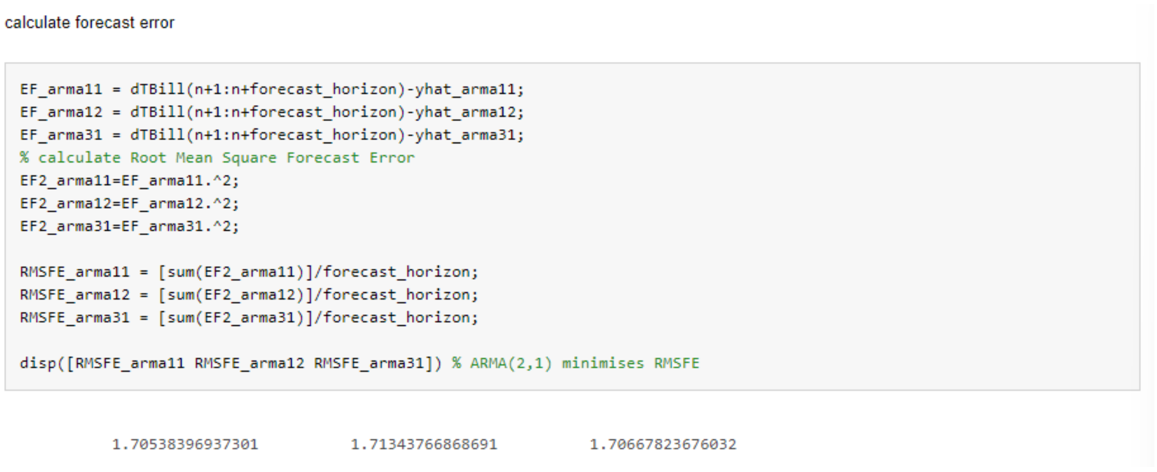
```
[yhat_arma11, yMSE_arma11] = forecast(EstMdl_arma11,forecast_horizon,'Y0',sample_Data(1:end-forecast_horizon));
[yhat_arma12, yMSE_arma12] = forecast(EstMdl_arma21,forecast_horizon,'Y0',sample_Data(1:end-forecast_horizon));
[yhat_arma31, yMSE_arma31] = forecast(EstMdl_arma31,forecast_horizon,'Y0',sample_Data(1:end-forecast_horizon));
```

Plot forecasts for each model

```
t1 = datetime(Date(OOS_iter-240));
t2 = datetime(Date(OOS_iter+forecast_horizon));
fore_Dates = [t1:calmonths(1):t2]';

clf

hold on
plot(fore_Dates, sample_Data);
plot(fore_Dates,[repmat(missing,1, n) yhat_arma11']);
plot(fore_Dates,[repmat(missing,1, n) yhat_arma12']);
plot(fore_Dates,[repmat(missing,1, n) yhat_arma31']);
title('6-Month TBill Rate Forecast')
xlabel('Time')
ylabel('TBill Rate')
legend('Historical','ARMA(1,1)','ARMA(1,2)', 'ARMA(3,1)', Location='southwest')
grid on
hold off
```



We see that the each model forecasts that the series will revert to the mean value before staying relatively constant. This is contrary to what actually happened over the next three months, as we see a sharp rise in the series. This is one of the main issues with using pure ARIMA models for forecasting, as they will just expect the series to revert to its mean level before remaining constant. If we were instead using some more complicated model such as a GARCH or ARMAX model, we may be able to get more interesting results.

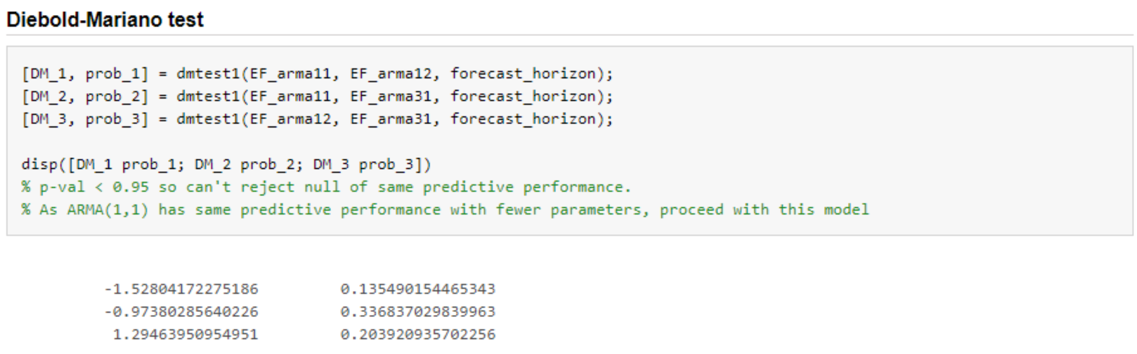


calculate forecast error

```
EF_arma11 = dTBill(n+1:n+forecast_horizon)-yhat_arma11;
EF_arma12 = dTBill(n+1:n+forecast_horizon)-yhat_arma12;
EF_arma31 = dTBill(n+1:n+forecast_horizon)-yhat_arma31;
% calculate Root Mean Square Forecast Error
EF2_arma11=EF_arma11.^2;
EF2_arma12=EF_arma12.^2;
EF2_arma31=EF_arma31.^2;

RMSFE_arma11 = [sum(EF2_arma11)]/forecast_horizon;
RMSFE_arma12 = [sum(EF2_arma12)]/forecast_horizon;
RMSFE_arma31 = [sum(EF2_arma31)]/forecast_horizon;

disp([RMSFE_arma11 RMSFE_arma12 RMSFE_arma31]) % ARMA(2,1) minimises RMSFE
```

```
        1.70538396937301          1.71343766868691          1.70667823676032
```

We then calculated the root mean squared error of each forecast in order to assess their predictive performance against the realised data. Each model resulted in an RMSE of roughly 1.7, indicating that no model was particularly outstanding in comparison to the others. In order to get a more rigorous assessment of their relative performance, we ran a Diebold-Mariano test on the forecast error.



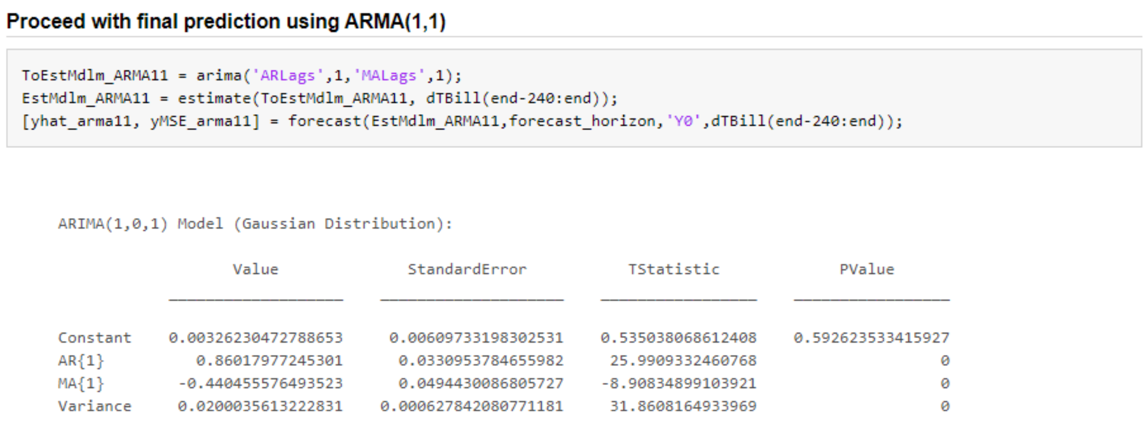
Diebold-Mariano test

```
[DM_1, prob_1] = dmtest1(EF_arma11, EF_arma12, forecast_horizon);
[DM_2, prob_2] = dmtest1(EF_arma11, EF_arma31, forecast_horizon);
[DM_3, prob_3] = dmtest1(EF_arma12, EF_arma31, forecast_horizon);

disp([DM_1 prob_1; DM_2 prob_2; DM_3 prob_3])
% p-val < 0.95 so can't reject null of same predictive performance.
% As ARMA(1,1) has same predictive performance with fewer parameters, proceed with this model
```

```
        -1.52804172275186          0.135490154465343
        -0.97380285640226          0.336837029839963
         1.29463950954951          0.203920935702256
```

According to the test, we fail to reject the null that each model has the same predictive accuracy at a significance level of 5%. As a result, we decided to choose the model which contained the least number of parameters in the ARMA(1,1).

Running the model based on the most recent 20-years of data, we arrive at our final model:



Proceed with final prediction using ARMA(1,1)

```
ToEstMdlm_ARMA11 = arima('ARLags',1,'MALags',1);
EstMdlm_ARMA11 = estimate(ToEstMdlm_ARMA11, dTBill(end-240:end));
[yhat_arma11, yMSE_arma11] = forecast(EstMdlm_ARMA11,forecast_horizon,'Y0',dTBill(end-240:end));
```

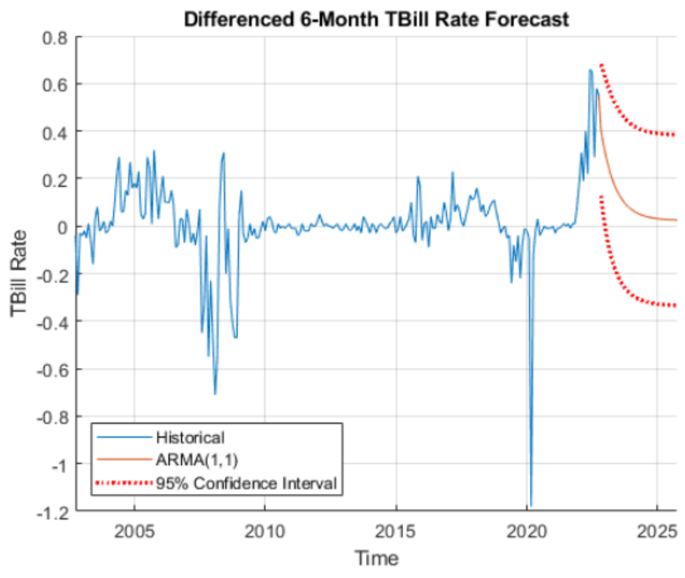
```
    ARIMA(1,0,1) Model (Gaussian Distribution):

                        Value              StandardError          TStatistic            PValue
                  ___________________    ___________________    _________________    _________________

    Constant      0.00326230472788653    0.00609733198302531    0.535038068612408    0.592623533415927
    AR{1}            0.86017977245301     0.0330953784655982     25.9909332460768                    0
    MA{1}          -0.440455576493523     0.0494430086805727    -8.90834899103921                    0
    Variance       0.0200035613222831   0.000627842080771181     31.8608164933969                    0
```

The constant included in the model was found to be not statistically different from 0 at a 5% confidence level. The autoregressive component had a positive coefficient of 0.86 while the moving average component had a coefficient of -0.44. This implies the series' previous value plays a large part in the current value, & that the process has a relatively resistant to shocks (negative MA coefficient means a proportion of the previous yesterday's shock term is counteracted in today's time series value).

Our forecast for the differenced series is shown below, along with a 95% confidence interval on the mean of the prediction. It shows the previously seen high values of the series reverting back to the mean value of 0.

Plot forecasts for differenced values

```
n = 240+forecast_horizon;
t1 = datetime(Date(end-240));
t2 = datetime(Date(end)+calmonths(forecast_horizon));
fore_Dates = [t1:calmonths(1):t2]';

clf

hold on
plot(fore_Dates, [dTBill(end-240:end)' repmat(missing, 1, forecast_horizon)]);
plot(fore_Dates, [repmat(missing,1, 240) dTBill(end) yhat_arma11']);
plot(fore_Dates, [repmat(missing,1, 241) (yhat_arma11 + 1.96*sqrt(yMSE_arma11))'],'r:','LineWidth',2);
plot(fore_Dates, [repmat(missing,1, 241) (yhat_arma11 - 1.96*sqrt(yMSE_arma11))'],'r:','LineWidth',2);
title('Differenced 6-Month TBill Rate Forecast')
xlabel('Time')
ylabel('TBill Rate')
legend('Historical','ARMA(1,1)','95% Confidence Interval', Location='southwest')
grid on
hold off

% Forecast shows high values reverting to the mean
```



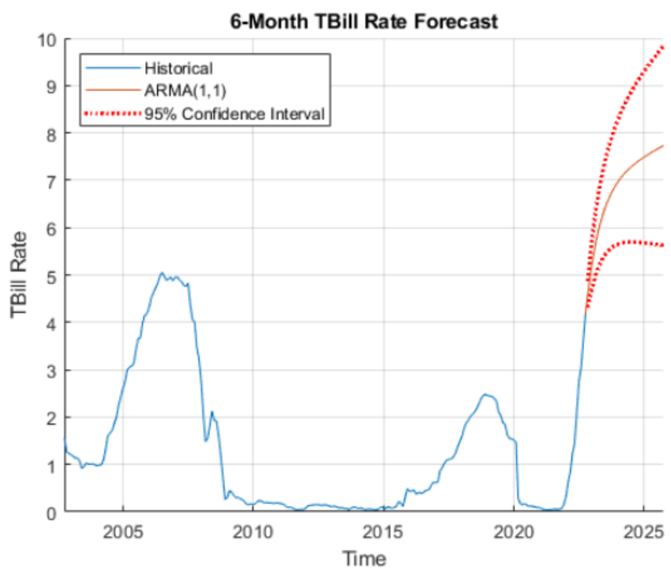
In order to convert this forecast into something more useful/less abstract, we undifference the series, and plot the forecast for the actual 6-Month Treasury Bill rate. This will allow us to examine the predicted dynamics of the T-Bill rate rather than the somewhat abstract differenced series. To undifference the series we use the formulae provided below:

$$\Delta y_t = y_t - y_{t-1}$$

$$\Rightarrow y_t = \Delta y_t + y_{t-1}$$

Plot forecast for integrated series

```
TBill_hat = cumsum(yhat_arma11)+TBill(end);
TBill_MSE = cumsum(yMSE_arma11);

clf

hold on
plot(fore_Dates, [TBill(end-240:end)' repmat(missing, 1, forecast_horizon)]);
plot(fore_Dates, [repmat(missing,1, 240) TBill(end) TBill_hat']);
plot(fore_Dates, [repmat(missing,1, 241) (TBill_hat + 1.96*sqrt(TBill_MSE))'],'r:','LineWidth',2);
plot(fore_Dates, [repmat(missing,1, 241) (TBill_hat - 1.96*sqrt(TBill_MSE))'],'r:','LineWidth',2);
title('6-Month TBill Rate Forecast')
xlabel('Time')
ylabel('TBill Rate')
legend('Historical','ARMA(1,1)','95% Confidence Interval', Location='northwest')
grid on
hold off
```



From the plot, we see that the model forecasts the Treasury Bill rate to continue to increase, up to a value of around 7.75%, before beginning to level off. Based off of this forecast, we expect the current interest rate hikes to continue for about another 3 years.

Examining the 95% confidence interval for our prediction, we see that there is quite a wide interval around our estimate. We could see rates continue to shoot up over 10% in the next 36 months, or we could rates peak within the next 18 months before beginning to fall back down.

A full outline of the code used for **Parts 2 & 3** is provided below.