Multivariate Time Series Analytics

Saransh Goel	2019A7PS0988P
Ayush Sharma	2018A3PS0326P

Submitted to

Dr. Navneet Goyal

Department of Computer Science & Information Systems

For the partial fulfillment of the course

CS F320 Foundations of Data Science

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

Multivariate Time Series Analysis

Time series

A time series is a set of data points which occur successively over a period of time. The thing which differentiates the time series data from other simple data is the dependence of future data on historical data which handicapped the normal statistical techniques to estimate a future value because now there is a need of considering the correlation between future and past data.

Multivariate time series

It can also be called as vector time series with each data point as a vector $\underline{Z}_{i,t}$.

$$\underline{Z}_t = [Z_{1,t}, Z_{2,t}, \ldots\ldots\ldots\ldots, Z_{m,t}]$$

Vector AutoRegression (VAR)

In a VAR model, each variable is a linear function of the past values of itself and the past values of all the other variables. Assuming k lags:

$$\underline{Z}_t = \underline{A}_1\underline{Z}_{t-1} + \underline{A}_2\underline{Z}_{t-2}. \ldots\ldots. + \underline{A}_k\underline{Z}_{t-k}$$

Where $\underline{Z}_t$ is a (n x 1) matix and

A_k is an (n x n) matrix

Applications

- Predicting Shifts in the Mean of a Multivariate Time Series Process
  An Application in Predicting Business Failures. Every firm needs to know are they going well or there is some trend of sliding towards failure, this could be done by analysing the time series of other companies which were failed in history and could be compared with ours company if there is any similarity, if a significant similarity is present so there could be the case that our company may fail in future.

- Detection of Changes in Multivariate Time Series
  Detection of changes in multivariate time series could be very important for economists to analyse various trends in time series to detect future trends especially in share values.

- International tourism demand
  Tourism sector is a great application of time series analysis, especially multivariate time series where the tourism graph of one place could be related to the tourism graph of other places. This can also help governments to find the things important to attract tourists by analysing other time series and find any distinct difference between both which leads to a high tourist count.

- Multivariate time series models for studies on stochastic generators in power systems.

- Short-term wind speed forecasting system based on multivariate time series and multi-objective optimization.
  Wind speed forecasting is very important for wind energy generation, a government planning to establish a number of wind energy power plants in the country would need a spatial analysis of wind speed throughout the year, of course government will want to establish these plants in those areas where the wind speed is high and consistent for high and consistent supply of electricity in adjacent areas.

- The variation in the atmospheric temperature can be caused by humidity and season factors.

- Stock market data
  In the stock market time series is used to forecast the future graph of any share. This could be based on many factors other than time. It can also depend on the stock graph of other companies and also depends on the monetary policies of the government.

- In e-commerce platforms where they analyse the rise and fall in business during festivals and during various seasons of the year. This analysis helps them to find the best time to come out with a new product and offers according to the festivals and seasons.

- Solar energy data of various states to find which areas are the best for construction of solar energy power plants, which can provide a reasonable amount of solar energy throughout the year.

- Forecasting GDP is also a key application of multivariate time series analysis, GDP(Gross Domestic Product) is a very good measure of the growth of a country. Many of the time there is a need for forecasting GDP to estimate future

growth and try to take necessary steps to increase the speed of the growth. Also the time series analysis of history data will tell what monetary and social policies affect the GDP the most. For example increasing the GST over luxury items will affect the most or GST increased on daily items. This is becoming more important during natural disasters like Covid-19 to find the necessary steps taken for instant healing of the economy otherwise a large unemployment could lead to a country wide slow and can have many harmful consequences.

Application - Forecasting Macroeconomic Data

Vector autoregression (VAR) is a statistical model for multivariate time series analysis, especially in a time series where the variables have a relationship that affects each other to time. VAR models are different from univariate autoregressive models because they allow analysis and make predictions on multivariate time series data. VAR models are often used in economics and weather forecasting.

Basic requirements to use the VAR model are :

- Time series with at least two variables.
- Relationship between variables.

It is considered an autoregressive model because the predictions made by the model are dependent on the past values, which means that each observation is modelled as the function of its lagged value.

The basic difference between the ARIMA family and VAR models is that all the ARIMA models are used for univariate time series, where the VAR models work with multivariate time series. In addition, ARIMA models are unidirectional models, which means that the dependent variables are influenced by their past or lag values itself, where VAR is a bi-directional model, which means a dependent variable is affected by its past value or by another variable's value or influenced by both of the things.

Vector Autoregression

A typical autoregression model(AR(p)) for univariate time series can be represented by

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + e_t$$

Where

- *yt*−i indicates the variable value at periods earlier.
- *Ai* is a time-invariant (*k* × *k*)-matrix.
- *et* is an error term.
- *c* is an intercept of the model.

Here the order p means, up to p-lags of y is used.

As we know, the VAR model deals with multivariate time series, which means there will be two or more variables affecting each other. Therefore, the VAR model equation increases with the number of variables in the time series.

Let's suppose there are two time-series variables, y1 and y2, so to calculate y1(t), the VAR model will use the lags of both time-series variables.

For example, the equation for the VAR(1) model with two time-series variables (y1 and y2) will look like this:

$$Y_{1,t} = \alpha_1 + \beta_{11,1} Y_{1,t-1} + \beta_{12,1} Y_{2,t-1} + \epsilon_{1,t}$$
$$Y_{2,t} = \alpha_2 + \beta_{21,1} Y_{1,t-1} + \beta_{22,1} Y_{2,t-1} + \epsilon_{2,t}$$

Where, Y{1,t-1} is the first lag value for yy1 and Y{2,t-1} is the first lag value for y2.

And the VAR(2) with y1 and y2 time series variables, the equation of the model will look like :

$$Y_{1,t} = \alpha_1 + \beta_{11,1} Y_{1,t-1} + \beta_{12,1} Y_{2,t-1} + \beta_{11,2} Y_{1,t-2} + \beta_{12,2} Y_{2,t-2} + \epsilon_{1,t}$$
$$Y_{2,t} = \alpha_2 + \beta_{21,1} Y_{1,t-1} + \beta_{22,1} Y_{2,t-1} + \beta_{21,2} Y_{1,t-2} + \beta_{22,2} Y_{2,t-2} + \epsilon_{2,t}$$

Here we can clearly understand how the model's equation will increase with variables and the lag values.

Defining the Problem

Forecasting the gross domestic product, personal consumption expenditure and the gross private domestic investment from the US Macroeconomic data for 1959 Q1 to 2009 Q3. These values are represented as columns named realgdp, realcons and realinv respectively in the dataset.

For this we use the VAR model, but before that we must analyse the data. For that we perform Granger's Causality test and Cointegration test.

Details about the Dataset

US Macroeconomic Data for 1959Q1 - 2009Q3

Number of Observations - 203

Number of Variables - 14

Variable name definitions:

```
year      - 1959q1 - 2009q3
quarter   - 1-4
realgdp   - Real gross domestic product (Bil. of chained 2005 US$,
            seasonally adjusted annual rate)
realcons  - Real personal consumption expenditures (Bil. of chained
            2005 US$, seasonally adjusted annual rate)
realinv   - Real gross private domestic investment (Bil. of chained
            2005 US$, seasonally adjusted annual rate)
realgovt  - Real federal consumption expenditures & gross investment
            (Bil. of chained 2005 US$, seasonally adjusted annual rate)
realdpi   - Real private disposable income (Bil. of chained 2005
            US$, seasonally adjusted annual rate)
cpi       - End of the quarter consumer price index for all urban
            consumers: all items (1982-84 = 100, seasonally adjusted).
m1        - End of the quarter M1 nominal money stock (Seasonally
            adjusted)
tbilrate  - Quarterly monthly average of the monthly 3-month
            treasury bill: secondary market rate
unemp     - Seasonally adjusted unemployment rate (%)
pop       - End of the quarter total population: all ages incl. armed
            forces over seas
infl      - Inflation rate (ln(cpi_{t}/cpi_{t-1}) * 400)
realint   - Real interest rate (tbilrate - infl)
```

Importing the data

```
mdata = sm.datasets.macrodata.load_pandas().data
mdata.head()
```

Output:

	year	quarter	realgdp	realcons	realinv	realgovt	realdpi	cpi	m1	tbilrate	unemp	pop	infl	realint
0	1959.0	1.0	2710.349	1707.4	286.898	470.045	1886.9	28.98	139.7	2.82	5.8	177.146	0.00	0.00
1	1959.0	2.0	2778.801	1733.7	310.859	481.301	1919.7	29.15	141.7	3.08	5.1	177.830	2.34	0.74
2	1959.0	3.0	2775.488	1751.8	289.226	491.260	1916.4	29.35	140.5	3.82	5.3	178.657	2.74	1.09
3	1959.0	4.0	2785.204	1753.7	299.356	484.052	1931.3	29.37	140.0	4.33	5.6	179.386	0.27	4.06
4	1960.0	1.0	2847.699	1770.5	331.722	462.199	1955.5	29.54	139.6	3.50	5.2	180.007	2.31	1.19

Here we can see how our data looks. Here we are considering three variable real gdp real cons and realinv for further modeling processing. And also need to make a datetime value using the year and quarter columns before going for further processes.

Implementing Vector Autoregression(VAR) in Python

To build the model, we can use python's statsmodel package, which provides most of the module to work on time series analysis and provides some data with the package to practice on the time series analysis.

Importing libraries

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from statsmodels.tsa.api import VAR
from statsmodels.tsa.base import  datetools
```

Making a datetime value using the year and quarter columns

```
mdata = mdata[['year','quarter']].astype(int)
mdata = mdata[['year','quarter']].astype(str)
quarterly = mdata["year"] + "Q" + mdata["quarter"]
quarterly = datetools.dates_from_str(quarterly)
quarterly
```

Output:

```
[datetime.datetime(1959, 3, 31, 0, 0),
 datetime.datetime(1959, 6, 30, 0, 0),
 datetime.datetime(1959, 9, 30, 0, 0),
 datetime.datetime(1959, 12, 31, 0, 0),
 datetime.datetime(1960, 3, 31, 0, 0),
 datetime.datetime(1960, 6, 30, 0, 0),
 datetime.datetime(1960, 9, 30, 0, 0),
 datetime.datetime(1960, 12, 31, 0, 0),
 datetime.datetime(1961, 3, 31, 0, 0),
 datetime.datetime(1961, 6, 30, 0, 0),
 datetime.datetime(1961, 9, 30, 0, 0),
 datetime.datetime(1961, 12, 31, 0, 0),
 datetime.datetime(1962, 3, 31, 0, 0),
 datetime.datetime(1962, 6, 30, 0, 0),
 datetime.datetime(1962, 9, 30, 0, 0),
 datetime.datetime(1962, 12, 31, 0, 0),
 datetime.datetime(1963, 3, 31, 0, 0),
 datetime.datetime(1963, 6, 30, 0, 0),
 datetime.datetime(1963, 9, 30, 0, 0),
 datetime.datetime(1963, 12, 31, 0, 0),
 datetime.datetime(1964, 3, 31, 0, 0),
 datetime.datetime(1964, 6, 30, 0, 0),
 datetime.datetime(1964, 9, 30, 0, 0),
 datetime.datetime(1964, 12, 31, 0, 0),
 datetime.datetime(1965, 3, 31, 0, 0),
 datetime.datetime(1965, 6, 30, 0, 0),
 datetime.datetime(1965, 9, 30, 0, 0),
 datetime.datetime(1965, 12, 31, 0, 0),
 datetime.datetime(1966, 3, 31, 0, 0),
 datetime.datetime(1966, 6, 30, 0, 0),
 datetime.datetime(1966, 9, 30, 0, 0),
 datetime.datetime(1966, 12, 31, 0, 0),
```

Now we can use the datetime values as the index of our data.

```
mata = mdata[['realgdp','realcons','realinv']]
mdata.index = pandas.DatetimeIndex(quarterly)
mdata.head()
```

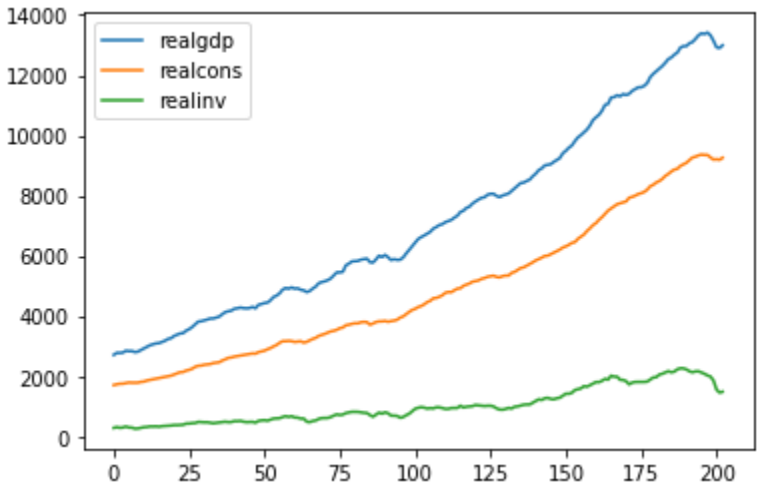
Output:

	realgdp	realcons	realinv
1959-03-31	2710.349	1707.4	286.898
1959-06-30	2778.801	1733.7	310.859
1959-09-30	2775.488	1751.8	289.226
1959-12-31	2785.204	1753.7	299.356
1960-03-31	2847.699	1770.5	331.722

Visualizing the data.

```
mdata.plot()
```

Output:

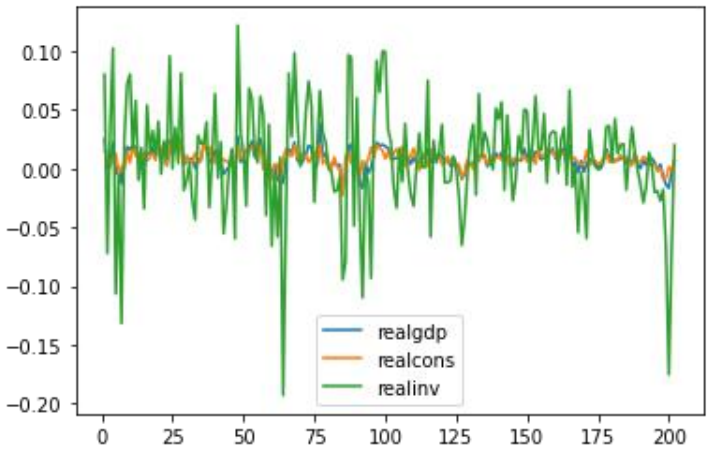


Here we can see that both realgdp and realcons have a high correlation, and there is a slight correlation between reading and other variables. Because of the trend we are seeing, we can understand that all of them are steadily growing until the last point, but they all have some sort of decrement.

So we can check the pattern of time series in their log values.

```
data = np.log(mdata).diff().dropna()
data.plot()
```

Output:



Here we can see how the normalized values are being plotted in the time series. The relationship between real gdp and realcons is as strong as they follow the same line pattern, which clearly says the change in real gdp will affect realcons or vice-versa. For more analysis, we can perform some tests on the time series to statistically define the relationship between the variables.

Granger's causality test

By using granger's causality test, we can find the relationship between the variables before building the model because it is known that if there is no relationship between the variables, we can drop the variables and separately do the modeling. If there is a relationship between them, we need to consider the variable in the modeling part.

In mathematics, the test provides the p-value between the variables, and if the p-value is higher than 0.05 then we will be required to accept the null hypothesis, and if the p-value is lesser than 0.05 we are required to reject the null hypothesis.

```
from statsmodels.tsa.stattools import grangercausalitytests
data = mdata[["realgdp", "realcons"]].pct_change().dropna()
#Performing test on for realgdp and realcons.
gc_res = grangercausalitytests(data, 12)
```

Output:

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=28.7248 , p=0.0000  , df_denom=198, df_num=1
ssr based chi2 test:   chi2=29.1600 , p=0.0000  , df=1
likelihood ratio test: chi2=27.2295 , p=0.0000  , df=1
parameter F test:         F=28.7248 , p=0.0000  , df_denom=198, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=18.9880 , p=0.0000  , df_denom=195, df_num=2
ssr based chi2 test:   chi2=38.9498 , p=0.0000  , df=2
likelihood ratio test: chi2=35.5873 , p=0.0000  , df=2
parameter F test:         F=18.9880 , p=0.0000  , df_denom=195, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=13.5015 , p=0.0000  , df_denom=192, df_num=3
ssr based chi2 test:   chi2=41.9812 , p=0.0000  , df=3
likelihood ratio test: chi2=38.0914 , p=0.0000  , df=3
parameter F test:         F=13.5015 , p=0.0000  , df_denom=192, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=10.9646 , p=0.0000  , df_denom=189, df_num=4
ssr based chi2 test:   chi2=45.9467 , p=0.0000  , df=4
likelihood ratio test: chi2=41.3192 , p=0.0000  , df=4
parameter F test:         F=10.9646 , p=0.0000  , df_denom=189, df_num=4
```

Here we can see that p-values for every lag are zero. So now, let's move forward for the causality test between realgdp and realinv.

```
data = mdata[["realgdp", "realinv"]].pct_change().dropna()
```

Output:

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=1.7235  , p=0.1908  , df_denom=198, df_num=1
ssr based chi2 test:   chi2=1.7496  , p=0.1859  , df=1
likelihood ratio test: chi2=1.7421  , p=0.1869  , df=1
parameter F test:         F=1.7235  , p=0.1908  , df_denom=198, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=2.1601  , p=0.1181  , df_denom=195, df_num=2
ssr based chi2 test:   chi2=4.4309  , p=0.1091  , df=2
likelihood ratio test: chi2=4.3825  , p=0.1118  , df=2
parameter F test:         F=2.1601  , p=0.1181  , df_denom=195, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=1.5522  , p=0.2024  , df_denom=192, df_num=3
ssr based chi2 test:   chi2=4.8263  , p=0.1850  , df=3
likelihood ratio test: chi2=4.7687  , p=0.1895  , df=3
parameter F test:         F=1.5522  , p=0.2024  , df_denom=192, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=1.4008  , p=0.2353  , df_denom=189, df_num=4
ssr based chi2 test:   chi2=5.8701  , p=0.2091  , df=4
likelihood ratio test: chi2=5.7847  , p=0.2158  , df=4
parameter F test:         F=1.4008  , p=0.2353  , df_denom=189, df_num=4
```

Here we can see p values for every lag is higher than 0.05, which means we need to accept the null hypothesis.

And also, we can say that a similar thing will happen if we perform the test between realcons and realinv.

```
data = mdata[["realcons", "realinv"]].pct_change().dropna()
```

Output:

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=5.9571  , p=0.0155  , df_denom=198, df_num=1
ssr based chi2 test:   chi2=6.0474  , p=0.0139  , df=1
likelihood ratio test: chi2=5.9582  , p=0.0146  , df=1
parameter F test:         F=5.9571  , p=0.0155  , df_denom=198, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=1.0988  , p=0.3353  , df_denom=195, df_num=2
ssr based chi2 test:   chi2=2.2539  , p=0.3240  , df=2
likelihood ratio test: chi2=2.2413  , p=0.3261  , df=2
parameter F test:         F=1.0988  , p=0.3353  , df_denom=195, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=0.2671  , p=0.8490  , df_denom=192, df_num=3
ssr based chi2 test:   chi2=0.8306  , p=0.8421  , df=3
likelihood ratio test: chi2=0.8289  , p=0.8425  , df=3
parameter F test:         F=0.2671  , p=0.8490  , df_denom=192, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=0.3127  , p=0.8693  , df_denom=189, df_num=4
ssr based chi2 test:   chi2=1.3103  , p=0.8596  , df=4
likelihood ratio test: chi2=1.3060  , p=0.8604  , df=4
parameter F test:         F=0.3127  , p=0.8693  , df_denom=189, df_num=4
```

Here we are getting p-values higher than before. This means that the realcons are affecting the realinv. In graphs, we were estimating that there is no significant relationship between the realinv and other values, and hence we can understand the significance of the granger's causality test.

Cointegration test

Cointegration helps to find out the statistical connection between two or more time series. When two or more time series are cointegrated, they have a long run, statistically significant relationship.

We can perform this test using the statsmodel package.

```
data = mdata[["realgdp","realcons",
"realinv"]].pct_change().dropna()
from statsmodels.tsa.vector_ar.vecm import coint_johansen

def cointegration_test(data, alpha=0.05):

   """Perform Johanson's Cointegration Test and Report
Summary"""
  out = coint_johansen(data,-1,5)
  d = {'0.90':0, '0.95':1, '0.99':2}
   traces = out.lr1
   cvts = out.cvt[:, d[str(1-alpha)]]
def adjust(val, length= 6): return str(val).ljust(length)

   # Summary
    print('Name   ::  Test Stat > C(95%)    =>   Signif  \n', '-
-'*20)
    for col, trace, cvt in zip(data.columns, traces, cvts):
        print(adjust(col), ':: ', adjust(round(trace,2), 9),
">", adjust(cvt, 8), ' =>  ' , trace > cvt)
cointegration_test(data)
```

Output:

```
Name   ::  Test Stat > C(95%)    =>   Signif
 ----------------------------------------
realgdp ::  69.55     > 24.2761   =>   True
realcons ::  22.74     > 12.3212   =>   True
realinv ::  2.6       > 4.1296    =>   False
```

Here we can see the significance of the variables on the whole system.

So here we have seen how we can use these two tests; next, we can further proceed with the modeling part.

Modeling

We can directly put the preprocessed data in the VAR module for modeling purposes.

```
var = VAR(data)
```

After this step, one thing comes up in the procedure: how to select the order. One thing that is usually performed is to check for the best-fit lag value. We need to compare the different AIC(Akaike Information Criterion), BIC(Bayesian Information criterion), FPE(Focused Prediction Error) and HQIC(Hannan–Quinn information criterion). These all are the parameters that help to select the best-fit lag value.

Statsmodel provides the select_order module to analyze these values.

```
x= var.select_order()
x.summary()
```

Output:

VAR Order Selection (* highlights the minimums)

	AIC	BIC	FPE	HQIC
0	-27.70	-27.64	9.365e-13	-27.68
1	-28.02	-27.82*	6.747e-13	-27.94*
2	-28.02	-27.66	6.758e-13	-27.88
3	-28.04*	-27.52	6.675e-13*	-27.83
4	-28.02	-27.35	6.779e-13	-27.75
5	-28.01	-27.19	6.844e-13	-27.68
6	-27.96	-26.98	7.235e-13	-27.56
7	-27.92	-26.78	7.530e-13	-27.46
8	-27.92	-26.63	7.503e-13	-27.40
9	-27.94	-26.50	7.368e-13	-27.36
10	-27.90	-26.30	7.695e-13	-27.25
11	-27.84	-26.09	8.176e-13	-27.13
12	-27.81	-25.90	8.452e-13	-27.04
13	-27.78	-25.72	8.780e-13	-26.94
14	-27.79	-25.57	8.733e-13	-26.89

Here we can see the minimum values in combination with the AIC, BIC, FPE and HQIC are given with the * sign. Here we can see we have that sign in the third row and the first row.

Here we select the third row, which means that the value of lag valueVAR(p) is three because it is suggested to go with the combinations where AIC with other parameters are generating minimums.

```
results = var.fit(3)
#We can check the summary of the model by.
results.summary()
```

Output:

```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Tue, 10, Aug, 2021
Time:                     10:07:41
--------------------------------------------------------------------
No. of Equations:         3.00000    BIC:                   -27.4257
Nobs:                     199.000    HQIC:                  -27.7212
Log likelihood:           1961.15    FPE:                7.47593e-13
AIC:                     -27.9222    Det(Omega_mle):     6.45336e-13
--------------------------------------------------------------------

Results for equation realgdp
================================================================================
                 coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------------
const               0.001281         0.001295            0.989           0.322
L1.realgdp         -0.286148         0.171582           -1.668           0.095
L1.realcons         0.673869         0.132245            5.096           0.000
L1.realinv          0.030578         0.026428            1.157           0.247
L2.realgdp          0.025691         0.174478            0.147           0.883
L2.realcons         0.295441         0.147990            1.996           0.046
L2.realinv         -0.014443         0.026963           -0.536           0.592
L3.realgdp         -0.180031         0.174857           -1.030           0.303
L3.realcons         0.183702         0.148048            1.241           0.215
L3.realinv          0.012632         0.026449            0.478           0.633
================================================================================
```

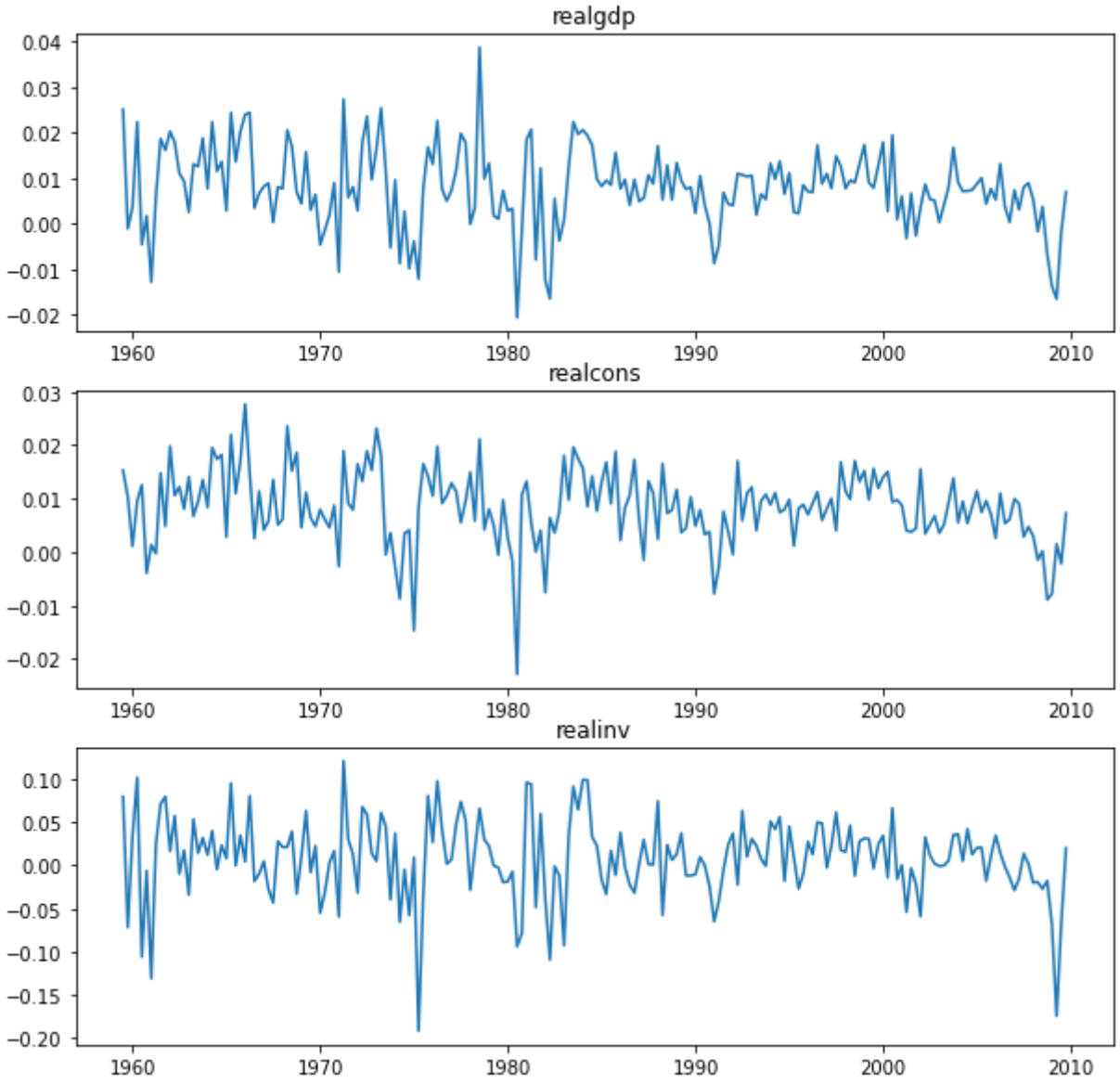
```
Results for equation realcons
================================================================================
                coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------------
const               0.004837         0.001094            4.420           0.000
L1.realgdp         -0.127156         0.144955           -0.877           0.380
L1.realcons         0.256394         0.111722            2.295           0.022
L1.realinv          0.024043         0.022326            1.077           0.282
L2.realgdp         -0.086634         0.147402           -0.588           0.557
L2.realcons         0.205707         0.125024            1.645           0.100
L2.realinv          0.003846         0.022779            0.169           0.866
L3.realgdp         -0.359067         0.147722           -2.431           0.015
L3.realcons         0.418452         0.125073            3.346           0.001
L3.realinv          0.041906         0.022345            1.875           0.061
================================================================================

Results for equation realinv
================================================================================
                coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------------
const              -0.020597         0.006812           -3.024           0.002
L1.realgdp         -1.862537         0.902338           -2.064           0.039
L1.realcons         4.403374         0.695465            6.332           0.000
L1.realinv          0.223717         0.138981            1.610           0.107
L2.realgdp          0.331425         0.917568            0.361           0.718
L2.realcons         0.878198         0.778270            1.128           0.259
L2.realinv         -0.096555         0.141797           -0.681           0.496
L3.realgdp         -0.488310         0.919562           -0.531           0.595
L3.realcons        -0.123787         0.778572           -0.159           0.874
L3.realinv          0.033453         0.139095            0.241           0.810
================================================================================

Correlation matrix of residuals
             realgdp  realcons   realinv
realgdp     1.000000  0.599898  0.759619
realcons    0.599898  1.000000  0.142964
realinv     0.759619  0.142964  1.000000
```

Here we can see all coefficient standard error value t-test and the model's probabilities at every lag till 3 lag. Then, at last, there is a confusion matrix that shows the correlation between the variables. In the results, we found that the correlation between realgdp and realinv is high, the relatable effect in the probability also we can cross-check for the same thing. We can also plot the model, which will be a better way to understand the model's performance.

Visualizing the input:

```
results.plot();
```

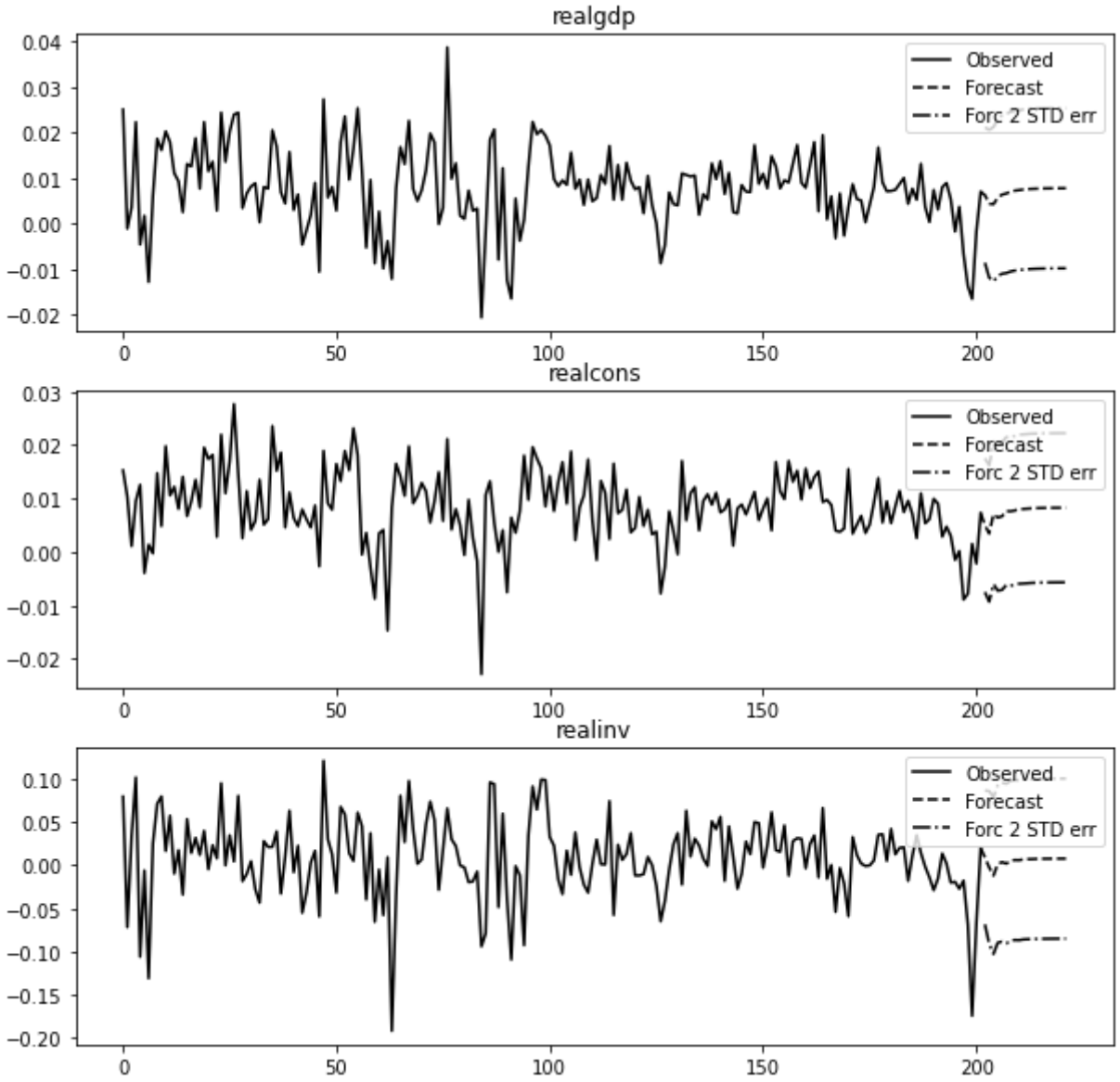
Output:



Plotting values forecasted by the model.

```
results.plot_forecast(20);
```

Output:



Here we can see the results by the model in the plot for every variable. The lines for forecasted values for the next 20 steps are going in such a steady manner, so here we can see the lags we have decided are quite satisfactory and providing good results.

References

- Data Source: FRED, Federal Reserve Economic Data, Federal Reserve Bank of St. Louis; http://research.stlouisfed.org/fred2/; accessed December 15, 2009.
- https://www.sciencedirect.com/science/article/pii/S0378779609002132
- https://www.sciencedirect.com/science/article/pii/S0169207002000572
- https://www.tandfonline.com/doi/abs/10.1080/01621459.2014.957545
- https://www.tandfonline.com/doi/abs/10.1080/01621459.1993.10476294