# VAR Model

**Date:** Thu Jul 21 13:45:34 2022

**Author:** kkitonga and ewayagi

In [1]:
```
#===========================libraries==================================#
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.api import VAR
from statsmodels.tsa.vector_ar.vecm import coint_johansen
from statsmodels.tsa.stattools import adfuller, grangercausalitytests
```

In [2]:
```
#====================exploring the datasets in statmodels=====================#

from statsmodels import datasets
print(dir(datasets))
```

```
['PytestTester', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__p
ackage__', '__path__', '__spec__', 'anes96', 'cancer', 'ccard', 'check_internet', 'china_smoking', 'clear_data_
home', 'co2', 'committee', 'copper', 'cpunish', 'danish_data', 'elnino', 'engel', 'fair', 'fertility', 'get_dat
a_home', 'get_rdataset', 'grunfeld', 'heart', 'interest_inflation', 'longley', 'macrodata', 'modechoice', 'nile
', 'randhie', 'scotland', 'spector', 'stackloss', 'star98', 'statecrime', 'strikes', 'sunspots', 'test', 'utils
', 'webuse']
```

In [3]:
```
#========loading macrodata from sm package and store in vardata===============#
vardata = datasets.macrodata.load_pandas().data
```

In [6]:
```
#==============================Exploring the data=============================#

#use these compands if you want to view all rows and all columns
#pd.set_option('display.max_columns', None)
#pd.set_option('display.max_rows' , None)

#view data
print(vardata)
```

```
       year  quarter    realgdp  realcons   realinv  realgovt  realdpi  \
0    1959.0      1.0   2710.349    1707.4   286.898   470.045   1886.9
1    1959.0      2.0   2778.801    1733.7   310.859   481.301   1919.7
2    1959.0      3.0   2775.488    1751.8   289.226   491.260   1916.4
3    1959.0      4.0   2785.204    1753.7   299.356   484.052   1931.3
4    1960.0      1.0   2847.699    1770.5   331.722   462.199   1955.5
..      ...      ...        ...       ...       ...       ...      ...
198  2008.0      3.0  13324.600    9267.7  1990.693   991.551   9838.3
199  2008.0      4.0  13141.920    9195.3  1857.661  1007.273   9920.4
200  2009.0      1.0  12925.410    9209.2  1558.494   996.287   9926.4
201  2009.0      2.0  12901.504    9189.0  1456.678  1023.528  10077.5
202  2009.0      3.0  12990.341    9256.0  1486.398  1044.088  10040.6

         cpi      m1  tbilrate  unemp      pop  infl  realint
0     28.980   139.7      2.82    5.8  177.146  0.00     0.00
1     29.150   141.7      3.08    5.1  177.830  2.34     0.74
2     29.350   140.5      3.82    5.3  178.657  2.74     1.09
3     29.370   140.0      4.33    5.6  179.386  0.27     4.06
4     29.540   139.6      3.50    5.2  180.007  2.31     1.19
..       ...     ...       ...    ...      ...   ...      ...
198  216.889  1474.7      1.17    6.0  305.270 -3.16     4.33
199  212.174  1576.5      0.12    6.9  305.952 -8.79     8.91
200  212.671  1592.8      0.22    8.1  306.547  0.94    -0.71
201  214.469  1653.6      0.18    9.2  307.226  3.37    -3.19
202  216.385  1673.9      0.12    9.6  308.013  3.56    -3.44

[203 rows x 14 columns]
```

In [7]:
```
#checking for more information about the data
vardata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 203 entries, 0 to 202
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   year      203 non-null    float64
 1   quarter   203 non-null    float64
 2   realgdp   203 non-null    float64
 3   realcons  203 non-null    float64
 4   realinv   203 non-null    float64
 5   realgovt  203 non-null    float64
 6   realdpi   203 non-null    float64
 7   cpi       203 non-null    float64
 8   m1        203 non-null    float64
 9   tbilrate  203 non-null    float64
 10  unemp     203 non-null    float64
 11  pop       203 non-null    float64
 12  infl      203 non-null    float64
 13  realint   203 non-null    float64
dtypes: float64(14)
memory usage: 22.3 KB
```

In [8]:
```
#describing the vardata
print(vardata.describe())
```

```
              year     quarter       realgdp     realcons      realinv  \
count   203.000000  203.000000    203.000000   203.000000   203.000000   
mean   1983.876847    2.492611   7221.171901  4825.293103  1012.863862   
std      14.686817    1.118563   3214.956044  2313.346192   585.102267   
min    1959.000000    1.000000   2710.349000  1707.400000   259.764000   
25%    1971.000000    1.500000   4440.103500  2874.100000   519.147500   
50%    1984.000000    2.000000   6559.594000  4299.900000   896.210000   
75%    1996.500000    3.000000   9629.346500  6398.150000  1436.681500   
max    2009.000000    4.000000  13415.266000  9363.600000  2264.721000   

          realgovt       realdpi         cpi           m1    tbilrate  \
count   203.000000    203.000000  203.000000   203.000000  203.000000   
mean    663.328640   5310.540887  105.075788   667.927586    5.311773   
std     140.863655   2423.515977   61.278878   455.346381    2.803071   
min     460.400000   1886.900000   28.980000   139.600000    0.120000   
25%     527.959500   3276.950000   41.050000   228.650000    3.515000   
50%     662.412000   4959.400000  104.100000   540.900000    5.010000   
75%     773.049000   6977.850000  159.650000  1102.100000    6.665000   
max    1044.088000  10077.500000  218.610000  1673.900000   15.330000   

            unemp         pop        infl     realint  
count  203.000000  203.000000  203.000000  203.000000  
mean     5.884729  239.724153    3.961330    1.336502  
std      1.458574   37.390450    3.253216    2.668799  
min      3.400000  177.146000   -8.790000   -6.790000  
25%      4.900000  208.631000    2.270000   -0.085000  
50%      5.700000  236.348000    3.240000    1.340000  
75%      6.800000  271.721500    4.975000    2.630000  
max     10.700000  308.013000   14.620000   10.950000  
```

In [9]:
```
#checking for missing values in the data
print(vardata.isnull())
```

```
      year  quarter  realgdp  realcons  realinv  realgovt  realdpi    cpi  \
0    False    False    False     False    False     False    False  False   
1    False    False    False     False    False     False    False  False   
2    False    False    False     False    False     False    False  False   
3    False    False    False     False    False     False    False  False   
4    False    False    False     False    False     False    False  False   
..     ...      ...      ...       ...      ...       ...      ...    ...   
198  False    False    False     False    False     False    False  False   
199  False    False    False     False    False     False    False  False   
200  False    False    False     False    False     False    False  False   
201  False    False    False     False    False     False    False  False   
202  False    False    False     False    False     False    False  False   

        m1  tbilrate  unemp    pop   infl  realint  
0    False     False  False  False  False    False  
1    False     False  False  False  False    False  
2    False     False  False  False  False    False  
3    False     False  False  False  False    False  
4    False     False  False  False  False    False  
..     ...       ...    ...    ...    ...      ...  
198  False     False  False  False  False    False  
199  False     False  False  False  False    False  
200  False     False  False  False  False    False  
201  False     False  False  False  False    False  
202  False     False  False  False  False    False  

[203 rows x 14 columns]
```

In [10]:
```
#view the first 5 rows
print(vardata.head())
```

```
     year  quarter   realgdp  realcons  realinv  realgovt  realdpi    cpi  \
0  1959.0      1.0  2710.349    1707.4  286.898   470.045   1886.9  28.98
1  1959.0      2.0  2778.801    1733.7  310.859   481.301   1919.7  29.15
2  1959.0      3.0  2775.488    1751.8  289.226   491.260   1916.4  29.35
3  1959.0      4.0  2785.204    1753.7  299.356   484.052   1931.3  29.37
4  1960.0      1.0  2847.699    1770.5  331.722   462.199   1955.5  29.54

      m1  tbilrate  unemp      pop  infl  realint
0  139.7      2.82    5.8  177.146  0.00     0.00
1  141.7      3.08    5.1  177.830  2.34     0.74
2  140.5      3.82    5.3  178.657  2.74     1.09
3  140.0      4.33    5.6  179.386  0.27     4.06
4  139.6      3.50    5.2  180.007  2.31     1.19
```

In [11]:
```python
#view the last 5 rows
print(vardata.tail())
```

```
       year  quarter    realgdp  realcons   realinv  realgovt  realdpi  \
198  2008.0      3.0  13324.600    9267.7  1990.693   991.551   9838.3
199  2008.0      4.0  13141.920    9195.3  1857.661  1007.273   9920.4
200  2009.0      1.0  12925.410    9209.2  1558.494   996.287   9926.4
201  2009.0      2.0  12901.504    9189.0  1456.678  1023.528  10077.5
202  2009.0      3.0  12990.341    9256.0  1486.398  1044.088  10040.6

         cpi      m1  tbilrate  unemp      pop  infl  realint
198  216.889  1474.7      1.17    6.0  305.270 -3.16     4.33
199  212.174  1576.5      0.12    6.9  305.952 -8.79     8.91
200  212.671  1592.8      0.22    8.1  306.547  0.94    -0.71
201  214.469  1653.6      0.18    9.2  307.226  3.37    -3.19
202  216.385  1673.9      0.12    9.6  308.013  3.56    -3.44
```

In [19]:
```python
#check the class ot type of the dataset
print(type(vardata))
#view the names of the variables
print(vardata.columns.values)
#check the number of rows and columns
print(vardata.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
['year' 'quarter' 'realgdp' 'realcons' 'realinv' 'realgovt' 'realdpi'
 'cpi' 'm1' 'tbilrate' 'unemp' 'pop' 'infl' 'realint']
(203, 14)
```

In [16]:
```python
#check data types
print(vardata.dtypes)
```

```
year        float64
quarter     float64
realgdp     float64
realcons    float64
realinv     float64
realgovt    float64
realdpi     float64
cpi         float64
m1          float64
tbilrate    float64
unemp       float64
pop         float64
infl        float64
realint     float64
dtype: object
```

In [17]:
```python
#selecting columns
selection = vardata[["realgdp" , "unemp" , "infl"]]
print(selection)
```

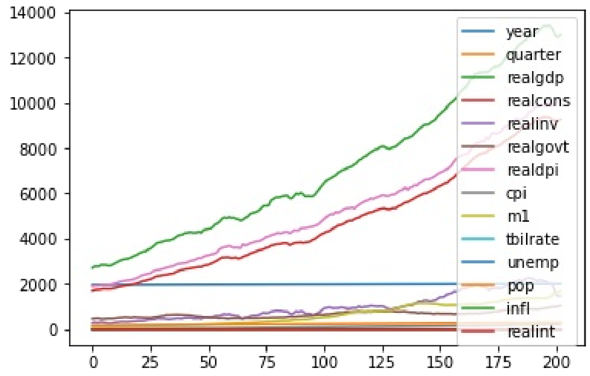
```
       realgdp  unemp  infl
0     2710.349    5.8  0.00
1     2778.801    5.1  2.34
2     2775.488    5.3  2.74
3     2785.204    5.6  0.27
4     2847.699    5.2  2.31
..         ...    ...   ...
198  13324.600    6.0 -3.16
199  13141.920    6.9 -8.79
200  12925.410    8.1  0.94
201  12901.504    9.2  3.37
202  12990.341    9.6  3.56

[203 rows x 3 columns]
```

In [18]:
```python
#obtain means of variables
print(vardata.mean())
```

```
year        1983.876847
quarter        2.492611
realgdp     7221.171901
realcons    4825.293103
realinv     1012.863862
realgovt     663.328640
realdpi     5310.540887
cpi          105.075788
m1           667.927586
tbilrate       5.311773
unemp          5.884729
pop          239.724153
infl           3.961330
realint        1.336502
dtype: float64
```

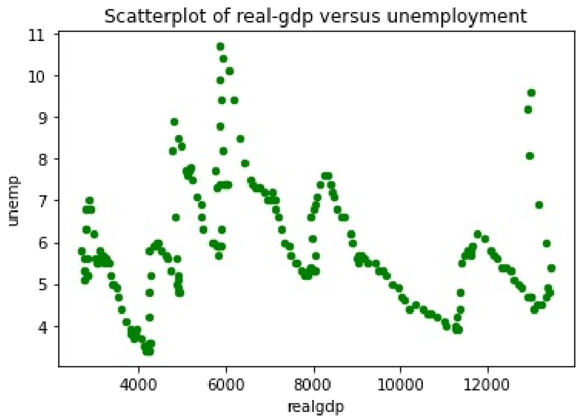
In [20]:
```python
#==============================plots=====================================#
#1. simple plot of all variables
vardata.plot()
```

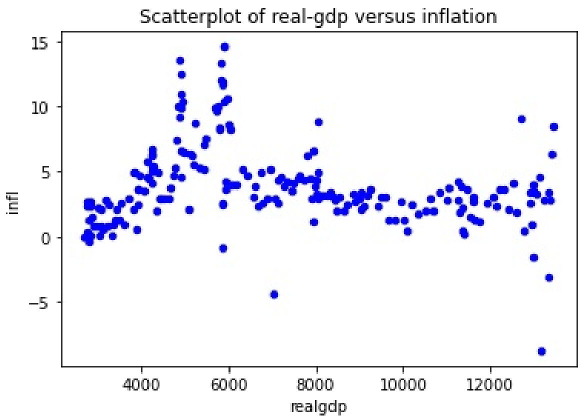
Out[20]:
```
<AxesSubplot:>
```

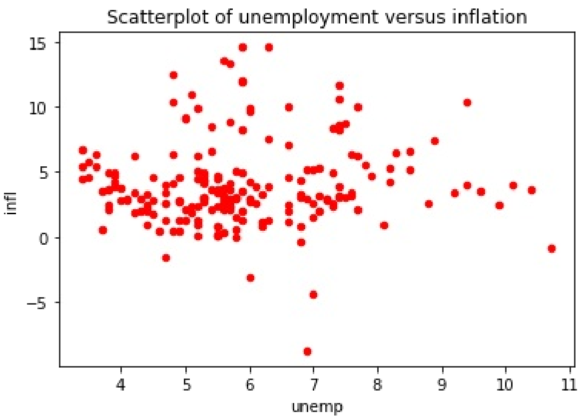



In [21]:
```python
#2. Scatter plots

vardata.plot(x = "realgdp", y ="unemp", kind = "scatter", color = "green")
plt.title("Scatterplot of real-gdp versus unemployment")

vardata.plot(x = "realgdp", y ="infl", kind = "scatter", color = "blue")
plt.title("Scatterplot of real-gdp versus inflation")

vardata.plot(x = "unemp", y ="infl", kind = "scatter", color = "red")
plt.title("Scatterplot of unemployment versus inflation")
```

Out[21]:
```
Text(0.5, 1.0, 'Scatterplot of unemployment versus inflation')
```






In [22]:
```
#====================Testing for stationarity================================#
# ADF method
adfgdp1 = adfuller(vardata['realgdp'])
print(adfgdp1[0])
print(adfgdp1[1])

adfunemp1 = adfuller(vardata['unemp'])
print(adfunemp1[0])
print(adfunemp1[1])

adfinfl1 = adfuller(vardata['infl'])
print(adfinfl1[0])
print(adfinfl1[1])
```

```
1.7504627967647144
0.9982455372335032
-2.53645846733464
0.10685366457233386
-3.0545144962572364
0.03010762086348588
```

In [23]:
```
#===============ADF test with differenced the non-stationary variables========#
adfdgdp2 = adfuller(vardata['realgdp'].diff()[1:])
print(adfdgdp2[0])
print(adfdgdp2[1])

adfdunemp2 = adfuller(vardata['unemp'].diff()[1:])
print(adfdunemp2[0])
print(adfdunemp2[1])

adfinfl2 = adfuller(vardata['infl'].diff()[1:])
print(adfinfl2[0])
print(adfinfl2[1])
```

```
-6.305695561658106
3.327882187668224e-08
-4.168474748074203
0.0007447109360995933
-17.155662786065147
6.895349138508994e-30
```

In [24]:
```
#==================Granger Causality test====================================#
granger1 = grangercausalitytests(vardata[["realgdp", "unemp"]], 5)

granger2 = grangercausalitytests(vardata[["realgdp", "infl"]], 5)

granger3 = grangercausalitytests(vardata[["infl", "unemp"]], 5)

granger4 = grangercausalitytests(vardata[["unemp", "infl"]], 5)
```

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=0.4286  , p=0.5134  , df_denom=199, df_num=1
ssr based chi2 test:   chi2=0.4351  , p=0.5095  , df=1
likelihood ratio test: chi2=0.4346  , p=0.5097  , df=1
parameter F test:         F=0.4286  , p=0.5134  , df_denom=199, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=4.8368  , p=0.0089  , df_denom=196, df_num=2
ssr based chi2 test:   chi2=9.9203  , p=0.0070  , df=2
likelihood ratio test: chi2=9.6833  , p=0.0079  , df=2
parameter F test:         F=4.8368  , p=0.0089  , df_denom=196, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=3.9284  , p=0.0094  , df_denom=193, df_num=3
ssr based chi2 test:   chi2=12.2127 , p=0.0067  , df=3
```

```
likelihood ratio test: chi2=11.8544 , p=0.0079  , df=3
parameter F test:         F=3.9284  , p=0.0094  , df_denom=193, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=2.9612  , p=0.0210  , df_denom=190, df_num=4
ssr based chi2 test:   chi2=12.4057 , p=0.0146  , df=4
likelihood ratio test: chi2=12.0344 , p=0.0171  , df=4
parameter F test:         F=2.9612  , p=0.0210  , df_denom=190, df_num=4

Granger Causality
number of lags (no zero) 5
ssr based F test:         F=2.5681  , p=0.0283  , df_denom=187, df_num=5
ssr based chi2 test:   chi2=13.5959 , p=0.0184  , df=5
likelihood ratio test: chi2=13.1494 , p=0.0220  , df=5
parameter F test:         F=2.5681  , p=0.0283  , df_denom=187, df_num=5

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=1.6140  , p=0.2054  , df_denom=199, df_num=1
ssr based chi2 test:   chi2=1.6384  , p=0.2006  , df=1
likelihood ratio test: chi2=1.6317  , p=0.2015  , df=1
parameter F test:         F=1.6140  , p=0.2054  , df_denom=199, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=2.9019  , p=0.0573  , df_denom=196, df_num=2
ssr based chi2 test:   chi2=5.9518  , p=0.0510  , df=2
likelihood ratio test: chi2=5.8654  , p=0.0533  , df=2
parameter F test:         F=2.9019  , p=0.0573  , df_denom=196, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=3.4352  , p=0.0180  , df_denom=193, df_num=3
ssr based chi2 test:   chi2=10.6793 , p=0.0136  , df=3
likelihood ratio test: chi2=10.4039 , p=0.0154  , df=3
parameter F test:         F=3.4352  , p=0.0180  , df_denom=193, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=2.8095  , p=0.0268  , df_denom=190, df_num=4
ssr based chi2 test:   chi2=11.7704 , p=0.0191  , df=4
likelihood ratio test: chi2=11.4355 , p=0.0221  , df=4
parameter F test:         F=2.8095  , p=0.0268  , df_denom=190, df_num=4

Granger Causality
number of lags (no zero) 5
ssr based F test:         F=2.4363  , p=0.0363  , df_denom=187, df_num=5
ssr based chi2 test:   chi2=12.8979 , p=0.0244  , df=5
likelihood ratio test: chi2=12.4952 , p=0.0286  , df=5
parameter F test:         F=2.4363  , p=0.0363  , df_denom=187, df_num=5

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=0.1681  , p=0.6823  , df_denom=199, df_num=1
ssr based chi2 test:   chi2=0.1706  , p=0.6796  , df=1
likelihood ratio test: chi2=0.1705  , p=0.6797  , df=1
parameter F test:         F=0.1681  , p=0.6823  , df_denom=199, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=0.5088  , p=0.6020  , df_denom=196, df_num=2
ssr based chi2 test:   chi2=1.0436  , p=0.5935  , df=2
likelihood ratio test: chi2=1.0409  , p=0.5943  , df=2
parameter F test:         F=0.5088  , p=0.6020  , df_denom=196, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=2.5585  , p=0.0564  , df_denom=193, df_num=3
ssr based chi2 test:   chi2=7.9538  , p=0.0470  , df=3
likelihood ratio test: chi2=7.7997  , p=0.0503  , df=3
parameter F test:         F=2.5585  , p=0.0564  , df_denom=193, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=1.9359  , p=0.1061  , df_denom=190, df_num=4
ssr based chi2 test:   chi2=8.1102  , p=0.0876  , df=4
likelihood ratio test: chi2=7.9493  , p=0.0935  , df=4
parameter F test:         F=1.9359  , p=0.1061  , df_denom=190, df_num=4

Granger Causality
number of lags (no zero) 5
ssr based F test:         F=2.7191  , p=0.0213  , df_denom=187, df_num=5
ssr based chi2 test:   chi2=14.3954 , p=0.0133  , df=5
likelihood ratio test: chi2=13.8962 , p=0.0163  , df=5
parameter F test:         F=2.7191  , p=0.0213  , df_denom=187, df_num=5

Granger Causality
number of lags (no zero) 1
```

```
ssr based F test:         F=3.1613  , p=0.0769  , df_denom=199, df_num=1
ssr based chi2 test:   chi2=3.2090  , p=0.0732  , df=1
likelihood ratio test: chi2=3.1837  , p=0.0744  , df=1
parameter F test:         F=3.1613  , p=0.0769  , df_denom=199, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=5.0238  , p=0.0075  , df_denom=196, df_num=2
ssr based chi2 test:   chi2=10.3038 , p=0.0058  , df=2
likelihood ratio test: chi2=10.0484 , p=0.0066  , df=2
parameter F test:         F=5.0238  , p=0.0075  , df_denom=196, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=3.8169  , p=0.0109  , df_denom=193, df_num=3
ssr based chi2 test:   chi2=11.8660 , p=0.0079  , df=3
likelihood ratio test: chi2=11.5273 , p=0.0092  , df=3
parameter F test:         F=3.8169  , p=0.0109  , df_denom=193, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=6.4844  , p=0.0001  , df_denom=190, df_num=4
ssr based chi2 test:   chi2=27.1660 , p=0.0000  , df=4
likelihood ratio test: chi2=25.4649 , p=0.0000  , df=4
parameter F test:         F=6.4844  , p=0.0001  , df_denom=190, df_num=4

Granger Causality
number of lags (no zero) 5
ssr based F test:         F=4.8501  , p=0.0003  , df_denom=187, df_num=5
ssr based chi2 test:   chi2=25.6769 , p=0.0001  , df=5
likelihood ratio test: chi2=24.1432 , p=0.0002  , df=5
parameter F test:         F=4.8501  , p=0.0003  , df_denom=187, df_num=5
```

In [25]:

```
#========================Cointegration test==================================#

def cointegration_test(vardata, alpha=0.05):
    out = coint_johansen(vardata, -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)

    print('Name   ::  Test Stat > C(95%)    =>   Signif  \n', '--'*20)
    for col, trace, cvt in zip(vardata[["realgdp" , "unemp" , "infl"]],
                               traces, cvts):
        print(adjust(col), ':: ', adjust(round(trace,2), 9), ">",
              adjust(cvt, 8), ' =>  ' , trace > cvt)

cointegration_test(vardata)
```

```
C:\Users\ewaya\Anaconda3\lib\site-packages\statsmodels\tsa\vector_ar\vecm.py:648: HypothesisTestWarning: Critic
al values are only available for time series with 12 variables at most.
  warnings.warn(
Name   ::  Test Stat > C(95%)    =>   Signif
 ----------------------------------------
realgdp ::  887.09    > nan       =>   False
unemp  ::  682.18    > nan       =>   False
infl   ::  553.9     > 311.1288  =>   True
```

In [26]:

```
#========================Splitting the train and test data===================#
variables = vardata[["realgdp" , "unemp" , "infl"]]
print(variables.shape)

test_obs = 12
train_data = variables[:-test_obs]
test_data = variables[-test_obs:]

print(train_data)
print(test_data)
```

```
(203, 3)
        realgdp  unemp  infl
0      2710.349    5.8  0.00
1      2778.801    5.1  2.34
2      2775.488    5.3  2.74
3      2785.204    5.6  0.27
4      2847.699    5.2  2.31
..          ...    ...   ...
186   12683.153    5.0  9.14
187   12748.699    4.9  0.40
188   12915.938    4.7  2.60
189   12962.462    4.7  3.97
190   12965.916    4.7 -1.58

[191 rows x 3 columns]
        realgdp  unemp  infl
191   13060.679    4.4  3.30
192   13099.901    4.5  4.58
193   13203.977    4.5  2.75
194   13321.109    4.7  3.45
195   13391.249    4.8  6.38
196   13366.865    4.9  2.82
197   13415.266    5.4  8.53
198   13324.600    6.0 -3.16
199   13141.920    6.9 -8.79
200   12925.410    8.1  0.94
201   12901.504    9.2  3.37
202   12990.341    9.6  3.56
```

In [27]:

```
#====================determine the number of lags to be used==================#
modellag = VAR(train_data)
order = modellag.select_order(maxlags = 10)
print(order.summary())

#lags = 2
```

```
 VAR Order Selection (* highlights the minimums)
==================================================
       AIC         BIC         FPE         HQIC
--------------------------------------------------
0        18.90       18.95   1.617e+08       18.92
1        6.744       6.956       849.2       6.830
2        5.952       6.323       384.5       6.102
3       5.754*      6.284*      315.5*      5.969*
4        5.772       6.462       321.6       6.052
5        5.784       6.632       325.6       6.128
6        5.845       6.852       346.3       6.253
7        5.896       7.062       365.0       6.369
8        5.951       7.276       386.1       6.488
9        5.997       7.481       405.2       6.599
10       6.036       7.679       422.5       6.702
--------------------------------------------------
```

In [28]:

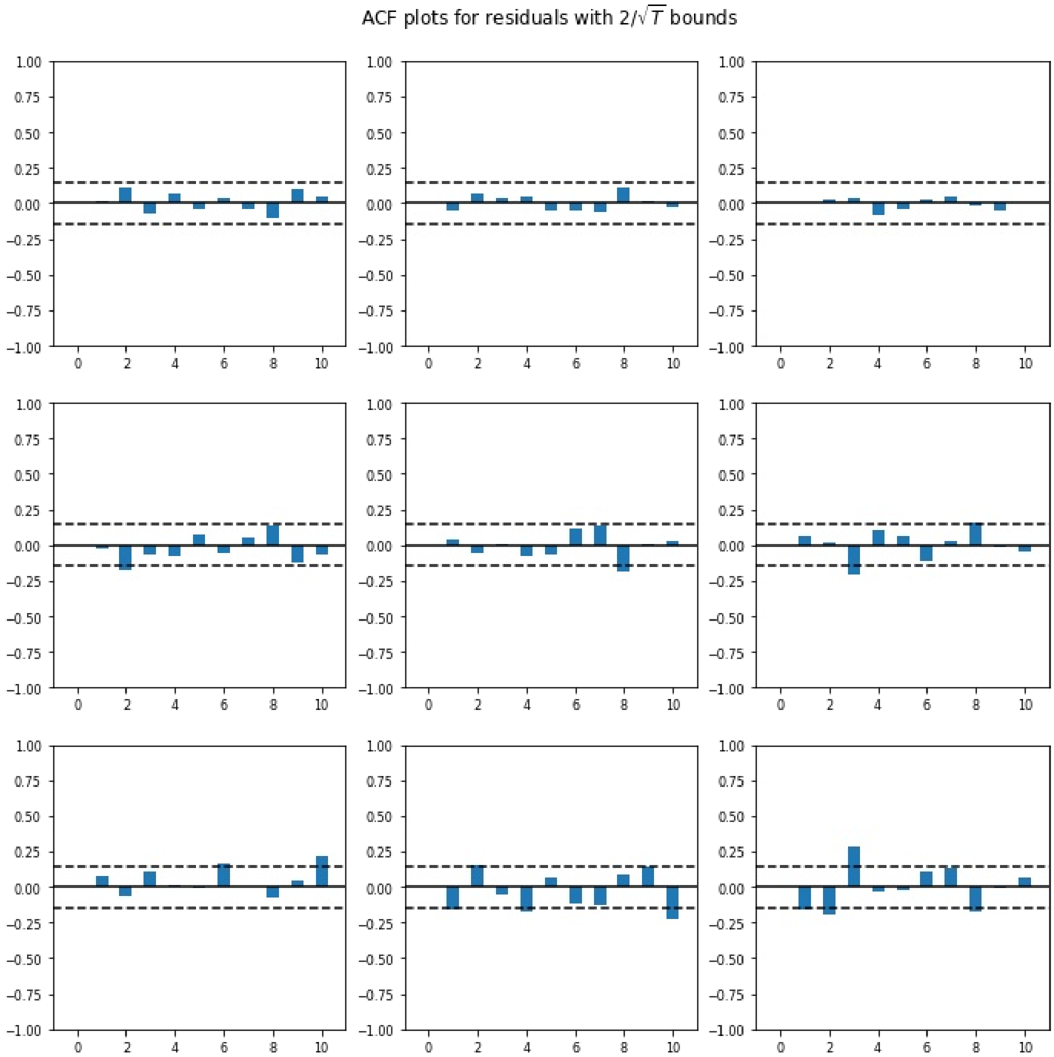
```
#==============================VAR model===================================#
VAR_model = modellag.fit(2)
print(VAR_model.summary())
```

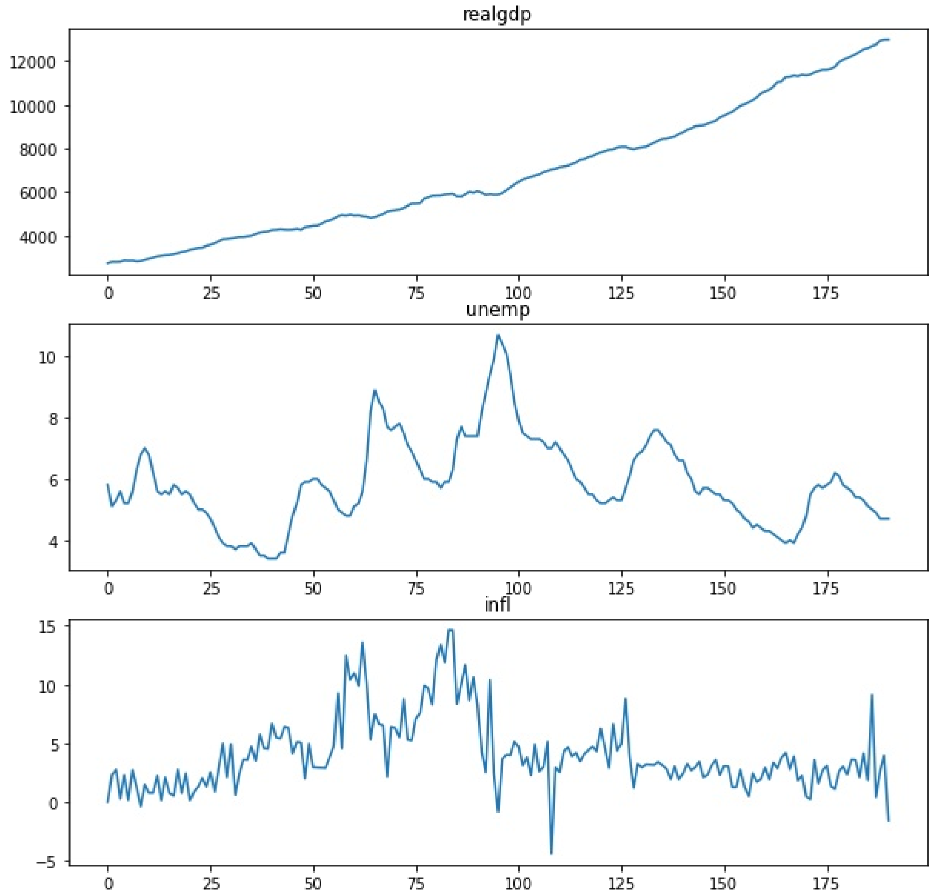
```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Mon, 05, Sep, 2022
Time:                     05:02:25
--------------------------------------------------------------------
No. of Equations:         3.00000    BIC:                    6.36432
Nobs:                     189.000    HQIC:                   6.15005
Log likelihood:          -1350.93    FPE:                    405.139
AIC:                      6.00413    Det(Omega_mle):         363.263
--------------------------------------------------------------------
Results for equation realgdp
=============================================================================
                coefficient       std. error           t-stat            prob
-----------------------------------------------------------------------------
const             -0.385066        17.950947           -0.021           0.983
L1.realgdp         1.035418         0.086438           11.979           0.000
L1.unemp         -34.582266        13.784381           -2.509           0.012
L1.infl           -1.811716         1.539806           -1.177           0.239
L2.realgdp        -0.030026         0.086961           -0.345           0.730
L2.unemp          40.477990        13.803529            2.932           0.003
L2.infl           -2.843963         1.586663           -1.792           0.073
=============================================================================

Results for equation unemp
=============================================================================
                coefficient       std. error           t-stat            prob
-----------------------------------------------------------------------------
const              0.175627         0.088836            1.977           0.048
L1.realgdp        -0.000891         0.000428           -2.082           0.037
L1.unemp           1.436192         0.068216           21.054           0.000
L1.infl            0.006009         0.007620            0.789           0.430
L2.realgdp         0.000896         0.000430            2.081           0.037
L2.unemp          -0.482377         0.068311           -7.061           0.000
L2.infl            0.020754         0.007852            2.643           0.008
=============================================================================

Results for equation infl
=============================================================================
                coefficient       std. error           t-stat            prob
-----------------------------------------------------------------------------
const              1.300788         0.798934            1.628           0.103
L1.realgdp         0.001845         0.003847            0.480           0.632
L1.unemp          -0.709542         0.613495           -1.157           0.247
L1.infl            0.384984         0.068531            5.618           0.000
L2.realgdp        -0.001917         0.003870           -0.495           0.620
L2.unemp           0.668754         0.614347            1.089           0.276
L2.infl            0.447822         0.070617            6.342           0.000
=============================================================================

Correlation matrix of residuals
            realgdp     unemp      infl
realgdp    1.000000 -0.507819  0.136025
unemp     -0.507819  1.000000 -0.256924
infl       0.136025 -0.256924  1.000000
```

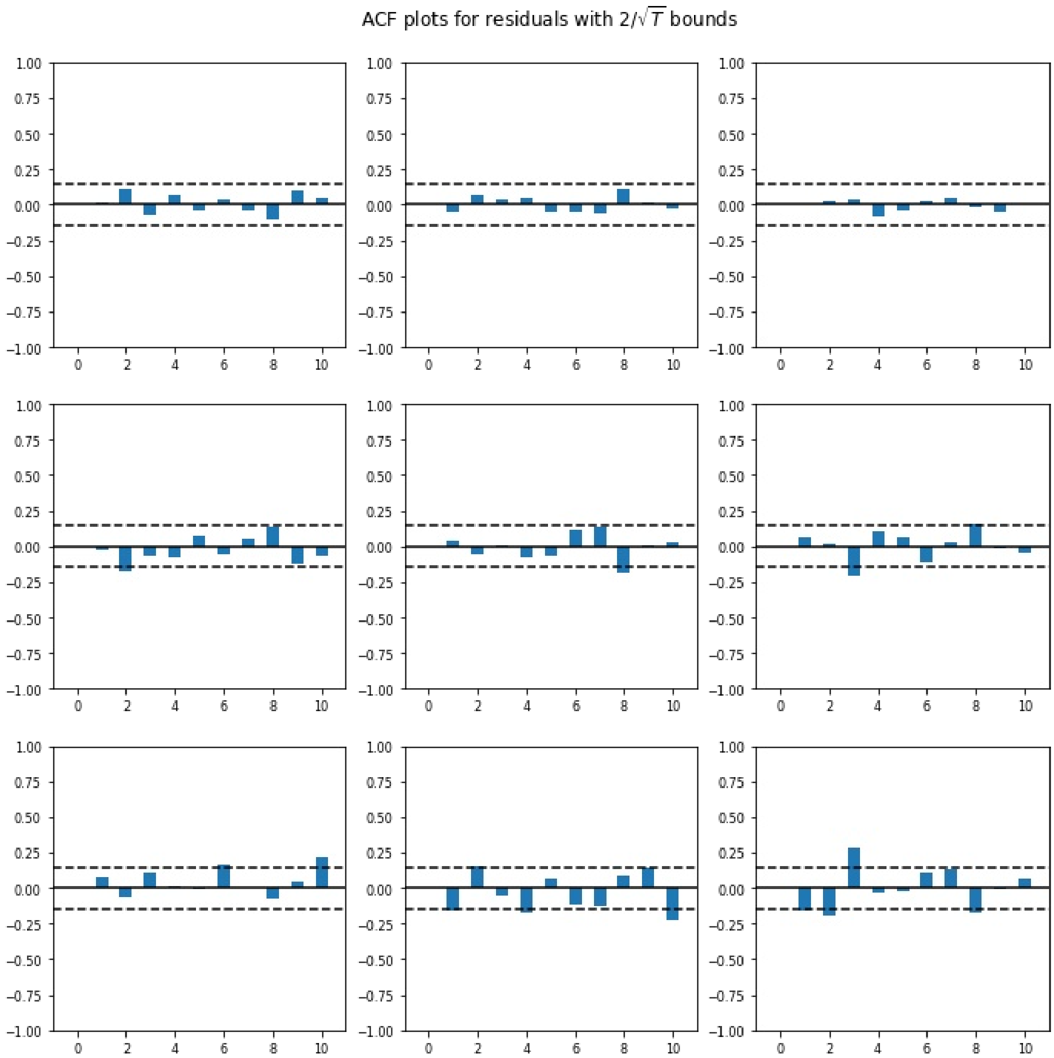
In [29]:

```
#===================================plotting the results======================#

VAR_model.plot()

#Autocorrelation plots

VAR_model.plot_acorr()
```

Out[29]:


ACF plots for residuals with $2/\sqrt{T}$ bounds

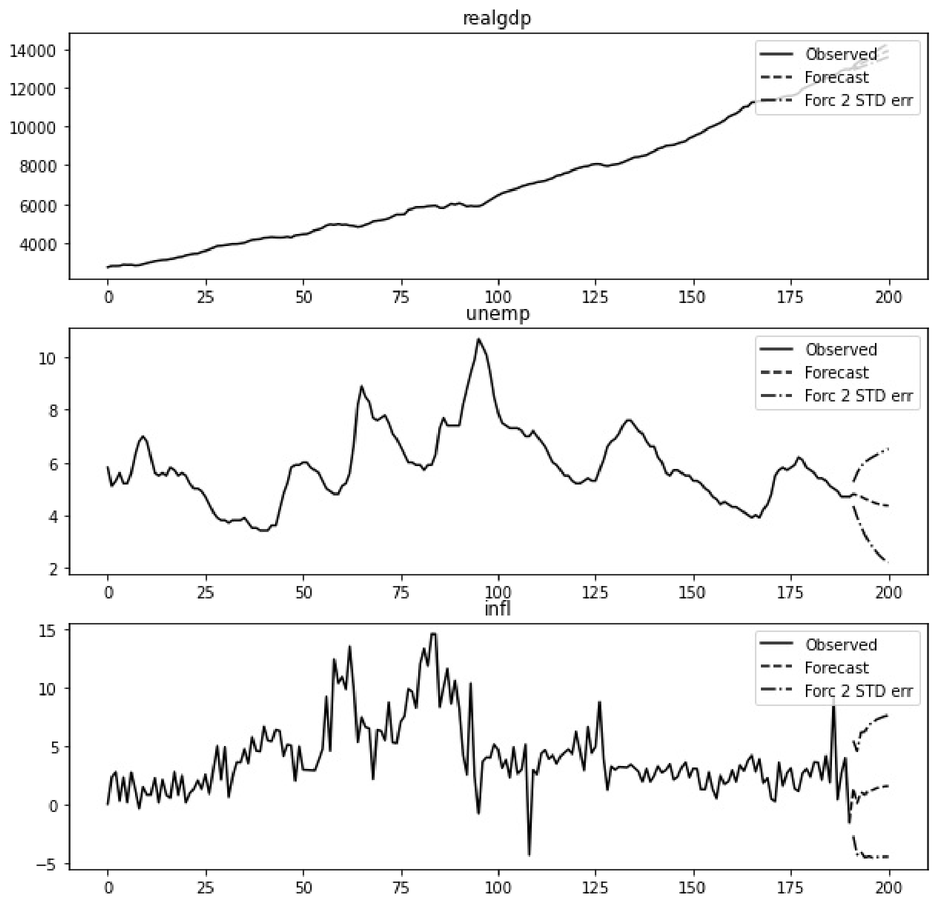

realgdp
unemp
infl

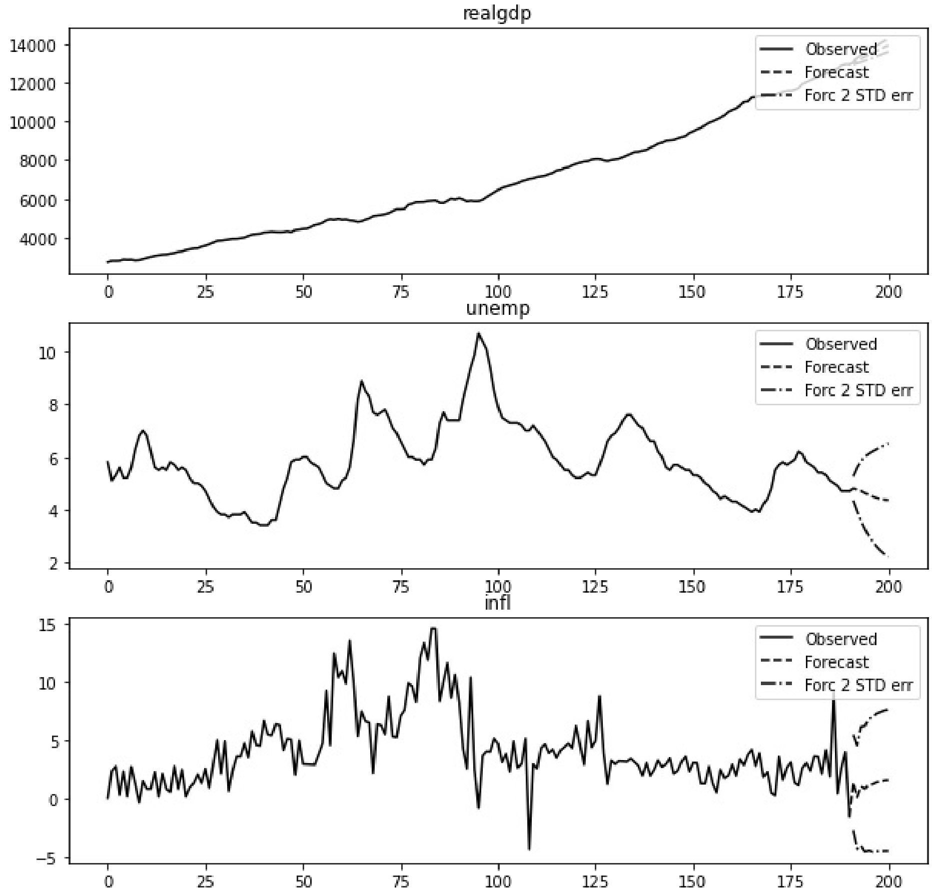



In [30]:

```
#==============================forecasting==================================#
#ploting the forecast

VAR_model.plot_forecast(10)

#=======================================END==================================#
```

Out[30]:






In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js