In [1]:

```
# import datetime
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datlib.FRED import *
from datlib.plots import *
import pandas_datareader.data as web

%matplotlib inline

# Import Statsmodels

from statsmodels.tsa.api import VAR
from statsmodels.tsa.stattools import adfuller
from statsmodels.tools.eval_measures import rmse, aic
```

In [2]:
```
#FRED.py
#. . .
def bil_to_mil(series):
    return series* 10**3
# . . .
#fedProject.py
# . . .
data_codes  = {# Assets
               "Balance Sheet: Total Assets ($ Mil)": "WALCL",
               "Balance Sheet Securities, Prem-Disc, Repos, and Loans ($ Mil)": '
               "Balance Sheet: Securities Held Outright ($ Mil)": "WSHOSHO",
               ### breakdown of securities holdings ###
               "Balance Sheet: U.S. Treasuries Held Outright ($ Mil)":"WSHOTSL",
               "Balance Sheet: Federal Agency Debt Securities ($ Mil)" : "WSHOFAD
               "Balance Sheet: Mortgage-Backed Securities ($ Mil)": "WSHOMCB",
               # other forms of lending
               "Balance Sheet: Repos ($ Mil)": "WORAL",
               "Balance Sheet: Central Bank Liquidity Swaps ($ Mil)" : "SWPT",
               "Balance Sheet: Direct Lending ($ Mil)" : "WLCFLL",
               # unamortized value of securities held (due to changes in interest
               "Balance Sheet: Unamortized Security Premiums ($ Mil)": "WUPSHO",
               # Liabilities
               "Balance Sheet: Total Liabilities ($ Mil)" : "WLTLECL",
               "Balance Sheet: Federal Reserve Notes Outstanding ($ Mil)" : "WLFN
               "Balance Sheet: Reverse Repos ($ Mil)": "WLRRAL",
               ### Major share of deposits
               "Balance Sheet: Deposits from Dep. Institutions ($ Mil)":"WLODLL",
               "Balance Sheet: U.S. Treasury General Account ($ Mil)": "WDTGAL",
               "Balance Sheet: Other Deposits ($ Mil)": "WOTHLB",
               "Balance Sheet: All Deposits ($ Mil)": "WLDLCL",
               # Capital
               "Balance Sheet: Total Capital": "WCTCL",
               # Interest Rates
               "Unemployment Rate": "UNRATE",
               "Nominal GDP ($ Bil)":"GDP",
               "Real GDP ($ Bil)":"GDPC1",
               "GDP Deflator":"GDPDEF",
               "CPI":"CPIAUCSL",
               "Core PCE":"PCEPILFE",
               "Private Investment":"GPDI",
               "Base: Total ($ Mil)": "BOGMBASE",
               "Base: Currency in Circulation ($ Bil)": "WCURCIR",
               "1 Month Treasury Rate (%)": "DGS1MO",
               "3 Month Treasury Rate (%)": "DGS3MO",
               "1 Year Treasury Rate (%)": "DGS1",
               "2 Year Treasury Rate (%)": "DGS2",
               "10 Year Treasury Rate (%)": "DGS10",
               "30 Year Treasury Rate (%)": "DGS30",
               "Effective Federal Funds Rate (%)": "DFF",
               "Federal Funds Target Rate (Pre-crisis)":"DFEDTAR",
               "Federal Funds Upper Target":"DFEDTARU",
               "Federal Funds Lower Target":"DFEDTARL",
               "Interest on Reserves (%)": "IOER",
               "VIX": "VIXCLS",
                "5 Year Forward Rate": "T5YIFR"
               }
```

```
inflation_target = 2

unemployment_target = 4
# Select start and end dates
start = datetime.datetime(2000, 1, 1)
end = datetime.datetime.today()

## year variable automatically adjusts the numper of periods
#   per year in light of data frequency
annual_div = {"Q":4,
             "W":52,
             "M":12}
### choose frequency
freq = "M"
### set periods per year
year = annual_div[freq]
```

In [22]:
```
#data cleaning, importing

d_parser = lambda x: pd.datetime.strptime(x, '%m/%d/%Y')
df = pd.read_csv('M4-4.csv', parse_dates=['Date'], date_parser=d_parser)
df
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_7108/1718697172.py:3: FutureWarning: T
he pandas.datetime class is deprecated and will be removed from pandas in a fut
ure version. Import from datetime module instead.
  d_parser = lambda x: pd.datetime.strptime(x, '%m/%d/%Y')
```

Out[22]:

	Date	M4	Log Total Assets	Effective Federal Funds Rate (%)	Log Currency in Circulation ($ Bil)	Unemployment Rate
0	2010-01-31	7.07	14.63	0.11	6.83	9.8
1	2010-02-28	7.06	14.63	0.13	6.83	9.8
2	2010-03-31	7.05	14.65	0.17	6.84	9.9
3	2010-04-30	7.06	14.66	0.20	6.84	9.9
4	2010-05-31	7.06	14.66	0.20	6.84	9.6
...	...	...	...	...	...	...
115	2019-08-31	7.40	15.14	2.13	7.47	3.7
116	2019-09-30	7.40	15.15	2.04	7.47	3.5
117	2019-10-31	7.41	15.19	1.83	7.48	3.6
118	2019-11-30	7.42	15.21	1.55	7.49	3.6
119	2019-12-31	7.42	15.23	1.55	7.49	3.6

120 rows × 6 columns

In [23]:
```
df['Date_at_year_month'] = df['Date'].dt.strftime('%Y-%m')
```

In [24]:

```
column_names = {'Date_at_year_month':'DATE',
                'M4':'M4',
                'Log Total Assets': 'TA',
                'Log Currency in Circulation ($ Bil)':'CC',
                'Effective Federal Funds Rate (%)':'FFR',
                'Unemployment Rate':'Unemployment Rate'}

# rename columns
df = df.rename(columns = column_names)

df
```

Out[24]:

	Date	M4	TA	FFR	CC	Unemployment Rate	DATE
0	2010-01-31	7.07	14.63	0.11	6.83	9.8	2010-01
1	2010-02-28	7.06	14.63	0.13	6.83	9.8	2010-02
2	2010-03-31	7.05	14.65	0.17	6.84	9.9	2010-03
3	2010-04-30	7.06	14.66	0.20	6.84	9.9	2010-04
4	2010-05-31	7.06	14.66	0.20	6.84	9.6	2010-05
...	...	...	...	...	...	...	...
115	2019-08-31	7.40	15.14	2.13	7.47	3.7	2019-08
116	2019-09-30	7.40	15.15	2.04	7.47	3.5	2019-09
117	2019-10-31	7.41	15.19	1.83	7.48	3.6	2019-10
118	2019-11-30	7.42	15.21	1.55	7.49	3.6	2019-11
119	2019-12-31	7.42	15.23	1.55	7.49	3.6	2019-12

120 rows × 7 columns

In [25]:
```
df = df.set_index('DATE')
df
```

Out[25]:
```
               Date    M4     TA   FFR    CC  Unemployment Rate
DATE
2010-01  2010-01-31  7.07  14.63  0.11  6.83                9.8
2010-02  2010-02-28  7.06  14.63  0.13  6.83                9.8
2010-03  2010-03-31  7.05  14.65  0.17  6.84                9.9
2010-04  2010-04-30  7.06  14.66  0.20  6.84                9.9
2010-05  2010-05-31  7.06  14.66  0.20  6.84                9.6
    ...         ...   ...    ...   ...   ...                ...
2019-08  2019-08-31  7.40  15.14  2.13  7.47                3.7
2019-09  2019-09-30  7.40  15.15  2.04  7.47                3.5
2019-10  2019-10-31  7.41  15.19  1.83  7.48                3.6
2019-11  2019-11-30  7.42  15.21  1.55  7.49                3.6
2019-12  2019-12-31  7.42  15.23  1.55  7.49                3.6

120 rows × 6 columns
```

In [26]:
```
df = df.drop(['Date'], axis = 1)
df
```

Out[26]:
```
           M4     TA   FFR    CC  Unemployment Rate
DATE
2010-01  7.07  14.63  0.11  6.83                9.8
2010-02  7.06  14.63  0.13  6.83                9.8
2010-03  7.05  14.65  0.17  6.84                9.9
2010-04  7.06  14.66  0.20  6.84                9.9
2010-05  7.06  14.66  0.20  6.84                9.6
    ...   ...    ...   ...   ...                ...
2019-08  7.40  15.14  2.13  7.47                3.7
2019-09  7.40  15.15  2.04  7.47                3.5
2019-10  7.41  15.19  1.83  7.48                3.6
2019-11  7.42  15.21  1.55  7.49                3.6
2019-12  7.42  15.23  1.55  7.49                3.6

120 rows × 5 columns
```

In [27]:
```
data = df
```

In [28]:

```
#ADF test

X = data["M4"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")


X = data["FFR"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")


X = data["TA"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")


X = data["CC"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")
```

```
X = data["Unemployment Rate"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")
```

```
ADF Statistic: 2.548278
p-value: 0.999064
Critical Values:
	1%: -3.487
	5%: -2.886
	10%: -2.580
Failed to Reject Ho - Time Series is Non-Stationary
ADF Statistic: -2.503304
p-value: 0.114673
Critical Values:
	1%: -3.492
	5%: -2.889
	10%: -2.581
Failed to Reject Ho - Time Series is Non-Stationary
ADF Statistic: -2.081078
p-value: 0.252191
Critical Values:
	1%: -3.489
	5%: -2.887
	10%: -2.580
Failed to Reject Ho - Time Series is Non-Stationary
ADF Statistic: -3.653973
p-value: 0.004809
Critical Values:
	1%: -3.492
	5%: -2.889
	10%: -2.581
Reject Ho - Time Series is Stationary
ADF Statistic: -2.223388
p-value: 0.197867
Critical Values:
	1%: -3.492
	5%: -2.889
	10%: -2.581
Failed to Reject Ho - Time Series is Non-Stationary
```

In [31]:

```
## 1st diff
data_diff = data.diff().dropna()
data_diff.to_csv("data-diff-1.csv")
```

In [ ]:

In [29]:

```
#ADF test

X = data_diff["M4"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")


X = data_diff["FFR"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")


X = data_diff["TA"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")


X = data_diff["CC"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")
```

```python
X = data_diff["Unemployment Rate"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")
```

```
ADF Statistic: -15.211102
p-value: 0.000000
Critical Values:
        1%: -3.487
        5%: -2.886
        10%: -2.580
Reject Ho - Time Series is Stationary
ADF Statistic: -1.657520
p-value: 0.453122
Critical Values:
        1%: -3.492
        5%: -2.889
        10%: -2.581
Failed to Reject Ho - Time Series is Non-Stationary
ADF Statistic: -2.732047
p-value: 0.068655
Critical Values:
        1%: -3.490
        5%: -2.888
        10%: -2.581
Failed to Reject Ho - Time Series is Non-Stationary
ADF Statistic: -5.133268
p-value: 0.000012
Critical Values:
        1%: -3.490
        5%: -2.888
        10%: -2.581
Reject Ho - Time Series is Stationary
ADF Statistic: -1.758765
p-value: 0.401117
Critical Values:
        1%: -3.492
        5%: -2.889
        10%: -2.581
Failed to Reject Ho - Time Series is Non-Stationary
```

```
In [ ]:
```

In [30]:

```
##2nd diff
data_new = data_diff.diff().dropna()

data_new.to_csv("data-diff2.csv")
```

In [ ]:

In [15]:

```
#ADF test

X = data_new["M4"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")


X = data_new["FFR"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")


X = data_new["TA"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")


X = data_new["CC"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")
```

```
X = data_new["Unemployment Rate"].values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if result[0] < result[4]["5%"]:
    print ("Reject Ho - Time Series is Stationary")
else:
    print ("Failed to Reject Ho - Time Series is Non-Stationary")
```

```
ADF Statistic: -8.381034
p-value: 0.000000
Critical Values:
        1%: -3.491
        5%: -2.888
        10%: -2.581
Reject Ho - Time Series is Stationary
ADF Statistic: -3.943879
p-value: 0.001735
Critical Values:
        1%: -3.492
        5%: -2.889
        10%: -2.581
Reject Ho - Time Series is Stationary
ADF Statistic: -6.016185
p-value: 0.000000
Critical Values:
        1%: -3.490
        5%: -2.887
        10%: -2.581
Reject Ho - Time Series is Stationary
ADF Statistic: -6.497273
p-value: 0.000000
Critical Values:
        1%: -3.494
        5%: -2.889
        10%: -2.582
Reject Ho - Time Series is Stationary
ADF Statistic: -7.922465
p-value: 0.000000
Critical Values:
        1%: -3.492
        5%: -2.889
        10%: -2.581
Reject Ho - Time Series is Stationary
```

```
In [17]: df = data_new
```

In [18]:

```
## Johansen test

from statsmodels.tsa.vector_ar.vecm import coint_johansen

def cointegration_test(df, alpha=0.05):
    """Perform Johanson's Cointegration Test and Report Summary"""
    out = coint_johansen(df,-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(df.columns, traces, cvts):
        print(adjust(col), ':: ', adjust(round(trace,2), 9), ">", adjust(cvt, 8),

cointegration_test(df)
```

```
Name   ::  Test Stat > C(95%)    =>   Signif
 ----------------------------------------
M4     ::  224.01    > 60.0627   =>   True
TA     ::  147.03    > 40.1749   =>   True
FFR    ::  85.81     > 24.2761   =>   True
CC     ::  41.46     > 12.3212   =>   True
Unemployment Rate ::  18.09     > 4.1296    =>   True
```

In [19]:

```
##Testing Causation using Granger’s Causality Test

from statsmodels.tsa.stattools import grangercausalitytests
maxlag=12
test = 'ssr_chi2test'
def grangers_causation_matrix(data_new, variables, test='ssr_chi2test', verbose=F
    df = pd.DataFrame(np.zeros((len(variables), len(variables))), columns=variabl
    for c in df.columns:
        for r in df.index:
            test_result = grangercausalitytests(data_new[[r, c]], maxlag=maxlag,
            p_values = [round(test_result[i+1][0][test][1],4) for i in range(maxl
            if verbose: print(f'Y = {r}, X = {c}, P Values = {p_values}')
            min_p_value = np.min(p_values)
            df.loc[r, c] = min_p_value
    df.columns = [var + '_x' for var in variables]
    df.index = [var + '_y' for var in variables]
    return df

grangers_causation_matrix(df, variables = df.columns)
```

Out[19]:

	M4_x	TA_x	FFR_x	CC_x	Unemployment Rate_x
M4_y	1.0000	0.0000	0.3917	0.0120	0.0026
TA_y	0.0031	1.0000	0.0080	0.3248	0.1002
FFR_y	0.1405	0.2517	1.0000	0.0217	0.2008
CC_y	0.0060	0.0103	0.6505	1.0000	0.0671
Unemployment Rate_y	0.0567	0.0000	0.5694	0.2324	1.0000

In [ ]:

```
#Now , I wanted to see the effects of unconventional monetary policy on the M4.
#Here, the row are the Response (Y) and the columns are the predictor series (X).
#If a given p-value is < significance level (0.05), then, the corresponding X ser
#For example, P-Value of 0.0 means (column 1, row 2) represents the p-value of th
#M4 causing Log Total Asset_y, which is less that the significance level of 0.05.
#So, we can reject the null hypothesis and conclude M4 causes Log Total Asset _y.
#Looking at the P-Values in the above table, we can pretty much observe that all
#are interchangeably causing each other. This makes this system of multi time ser

#Here, 0.0 means (column 1, row 3): M4 has an effect on Log Currency In Circulati
#Likewise, we can see that Log Total Asset has effects on VIX, EFFR, Currency in
#However, it has insignificant effect on Loss Function as the p value is greater
#Currency in Circulation causes M4, Loss Function, FFR, Log Total Asset;
#Loss Function causes VIX, FFR, Currency in Circulation but it does not have effe
#Also, Federal Funds Rate has significant effect on M4, Total Asset, Currency in
```

In [ ]: