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]:
```
#data cleaning, importing

d_parser = lambda x: pd.datetime.strptime(x, '%m/%d/%Y')
df = pd.read_csv('M4-5.csv', parse_dates=['Date'], date_parser=d_parser)
df
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_2724/2883410668.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[2]:

	Date	M4	Log Total Assets	Effective Federal Funds Rate (%)	Log Currency in Circulation ($ Bil)	Unemployment Rate
0	2010-01-31	0.00	0.000000e+00	0.00	6.83	0.0
1	2010-02-28	-0.01	0.000000e+00	0.00	6.83	0.0
2	2010-03-31	-0.01	2.000000e-02	0.02	6.84	0.1
3	2010-04-30	0.01	-1.000000e-02	-0.01	6.84	-0.1
4	2010-05-31	0.00	-1.000000e-02	-0.03	6.84	-0.3
...	...	...	...	...	...	...
115	2019-08-31	0.01	0.000000e+00	-0.29	7.47	0.1
116	2019-09-30	0.00	2.000000e-02	0.18	7.47	-0.3
117	2019-10-31	0.01	3.000000e-02	-0.12	7.48	0.3
118	2019-11-30	0.01	-2.000000e-02	-0.07	7.49	-0.1
119	2019-12-31	0.00	-1.780000e-15	0.28	7.49	0.0

120 rows × 6 columns

In [3]:
```
df['Date_at_year_month'] = df['Date'].dt.strftime('%Y-%m')
```

In [4]:

```
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': 'U'}

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

df
```

Out[4]:

	Date	M4	TA	FFR	CC	U	DATE
0	2010-01-31	0.00	0.000000e+00	0.00	6.83	0.0	2010-01
1	2010-02-28	-0.01	0.000000e+00	0.00	6.83	0.0	2010-02
2	2010-03-31	-0.01	2.000000e-02	0.02	6.84	0.1	2010-03
3	2010-04-30	0.01	-1.000000e-02	-0.01	6.84	-0.1	2010-04
4	2010-05-31	0.00	-1.000000e-02	-0.03	6.84	-0.3	2010-05
...	...	...	...	...	...	...	...
115	2019-08-31	0.01	0.000000e+00	-0.29	7.47	0.1	2019-08
116	2019-09-30	0.00	2.000000e-02	0.18	7.47	-0.3	2019-09
117	2019-10-31	0.01	3.000000e-02	-0.12	7.48	0.3	2019-10
118	2019-11-30	0.01	-2.000000e-02	-0.07	7.49	-0.1	2019-11
119	2019-12-31	0.00	-1.780000e-15	0.28	7.49	0.0	2019-12

120 rows × 7 columns

In [5]:

```
df = df.set_index('DATE')
df
```

Out[5]:

	Date	M4	TA	FFR	CC	U
DATE						
2010-01	2010-01-31	0.00	0.000000e+00	0.00	6.83	0.0
2010-02	2010-02-28	-0.01	0.000000e+00	0.00	6.83	0.0
2010-03	2010-03-31	-0.01	2.000000e-02	0.02	6.84	0.1
2010-04	2010-04-30	0.01	-1.000000e-02	-0.01	6.84	-0.1
2010-05	2010-05-31	0.00	-1.000000e-02	-0.03	6.84	-0.3
...	...	...	...	...	...	...
2019-08	2019-08-31	0.01	0.000000e+00	-0.29	7.47	0.1
2019-09	2019-09-30	0.00	2.000000e-02	0.18	7.47	-0.3
2019-10	2019-10-31	0.01	3.000000e-02	-0.12	7.48	0.3
2019-11	2019-11-30	0.01	-2.000000e-02	-0.07	7.49	-0.1
2019-12	2019-12-31	0.00	-1.780000e-15	0.28	7.49	0.0

120 rows × 6 columns

In [6]:
```
df = df.drop(['Date'], axis = 1)
df
```

Out[6]:

	M4	TA	FFR	CC	U
DATE					
2010-01	0.00	0.000000e+00	0.00	6.83	0.0
2010-02	-0.01	0.000000e+00	0.00	6.83	0.0
2010-03	-0.01	2.000000e-02	0.02	6.84	0.1
2010-04	0.01	-1.000000e-02	-0.01	6.84	-0.1
2010-05	0.00	-1.000000e-02	-0.03	6.84	-0.3
...	...	...	...	...	...
2019-08	0.01	0.000000e+00	-0.29	7.47	0.1
2019-09	0.00	2.000000e-02	0.18	7.47	-0.3
2019-10	0.01	3.000000e-02	-0.12	7.48	0.3
2019-11	0.01	-2.000000e-02	-0.07	7.49	-0.1
2019-12	0.00	-1.780000e-15	0.28	7.49	0.0

120 rows × 5 columns

In [7]:
```
data = df
data
```

Out[7]:

	M4	TA	FFR	CC	U
DATE					
2010-01	0.00	0.000000e+00	0.00	6.83	0.0
2010-02	-0.01	0.000000e+00	0.00	6.83	0.0
2010-03	-0.01	2.000000e-02	0.02	6.84	0.1
2010-04	0.01	-1.000000e-02	-0.01	6.84	-0.1
2010-05	0.00	-1.000000e-02	-0.03	6.84	-0.3
...	...	...	...	...	...
2019-08	0.01	0.000000e+00	-0.29	7.47	0.1
2019-09	0.00	2.000000e-02	0.18	7.47	-0.3
2019-10	0.01	3.000000e-02	-0.12	7.48	0.3
2019-11	0.01	-2.000000e-02	-0.07	7.49	-0.1
2019-12	0.00	-1.780000e-15	0.28	7.49	0.0

120 rows × 5 columns

In [8]:

```
#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["U"].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: -14.773444
p-value: 0.000000
Critical Values:
        1%: -3.487
        5%: -2.886
        10%: -2.580
Reject Ho - Time Series is Stationary
ADF Statistic: -3.996996
p-value: 0.001426
Critical Values:
        1%: -3.491
        5%: -2.888
        10%: -2.581
Reject Ho - Time Series is Stationary
ADF Statistic: -10.281226
p-value: 0.000000
Critical Values:
        1%: -3.487
        5%: -2.886
        10%: -2.580
Reject Ho - Time Series is 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: -8.219365
p-value: 0.000000
Critical Values:
        1%: -3.491
        5%: -2.888
        10%: -2.581
Reject Ho - Time Series is Stationary
```

```
In [9]: df = data
```

In [10]:
```
## 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     ::  166.56    > 60.0627  =>   True
TA     ::  97.62     > 40.1749  =>   True
FFR    ::  60.85     > 24.2761  =>   True
CC     ::  33.67     > 12.3212  =>   True
U      ::  12.02     > 4.1296   =>   True
```

In [11]:

```
##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[11]:

	M4_x	TA_x	FFR_x	CC_x	U_x
M4_y	1.0000	0.0000	0.3924	0.0004	0.0108
TA_y	0.0002	1.0000	0.0077	0.2116	0.1110
FFR_y	0.0625	0.2477	1.0000	0.0156	0.1985
CC_y	0.0019	0.0216	0.7152	1.0000	0.2312
U_y	0.1152	0.0000	0.5653	0.0048	1.0000

In [12]:

```
from statsmodels.tsa.stattools import kpss

def kpss_test(df):
    statistic, p_value, n_lags, critical_values = kpss(df.values)

    print(f'KPSS Statistic: {statistic}')
    print(f'p-value: {p_value}')
    print(f'num lags: {n_lags}')
    print('Critial Values:')
    for key, value in critical_values.items():
        print(f'   {key} : {value}')

print('KPSS Test: M4')
kpss_test(df['M4'])
print('KPSS Test: FFR')
kpss_test(df['FFR'])
print('KPSS Test: TA')
kpss_test(df['TA'])
```

```
KPSS Test: M4
KPSS Statistic: 0.7310020008542962
p-value: 0.010727090831427614
num lags: 3
Critial Values:
   10% : 0.347
   5% : 0.463
   2.5% : 0.574
   1% : 0.739
KPSS Test: FFR
KPSS Statistic: 0.07556965458914269
p-value: 0.1
num lags: 19
Critial Values:
   10% : 0.347
   5% : 0.463
   2.5% : 0.574
   1% : 0.739
KPSS Test: TA
KPSS Statistic: 0.08352134889349987
p-value: 0.1
num lags: 13
Critial Values:
   10% : 0.347
   5% : 0.463
   2.5% : 0.574
   1% : 0.739
```

```
C:\Users\HP\anaconda3\lib\site-packages\statsmodels\tsa\stattools.py:2022: In
terpolationWarning: The test statistic is outside of the range of p-values av
ailable in the
look-up table. The actual p-value is greater than the p-value returned.

  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\statsmodels\tsa\stattools.py:2022: In
terpolationWarning: The test statistic is outside of the range of p-values av
ailable in the
```

```
look-up table. The actual p-value is greater than the p-value returned.

  warnings.warn(
```

In [ ]: