VAR Model of Inflation in Python

Dr. Chen-Sheng Lin
Postdoctoral Research Fellow
The Research Institute for the Humanities and Social Sciences

June 22, 2022

Introduction

- A vector autoregressive (VAR) model of order p:

$$y_t = \mu + \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + \cdots + \Phi_p y_{t-p} + \varepsilon_t$$

where y_t is a multivariate time series.

- Originally proposed by Sims (1980)
- Treat all variables as "endogenous" and "stationary"
- Is a AR structure
- Useful for forecasting
- Called a *reduced-form* VAR (no contemporaneous effect)

Economic Theory of Inflation

- Phillips Curve (1958)

  A Phillips curve shows the tradeoff between unemployment and inflation in an economy.

- Okun‘s Law (1962)

  Okun's law implies a stable negative relationship between unemployment rate and GDP growth rate.

- Taylor’s rule (1993)

  The short-term interest rate is determined by the values of inflation and economic slack such as the output gap or unemployment gap.

Empirical Analysis: Data

- Four variables:
  - Unemployment rate (y_{1t})
  - GDP growth rate (y_{2t})
  - Inflation rate (y_{3t})
  - Interest rate (y_{4t})
- Sample period: 1978 Q1 to 2022 Q1
  - Training set: 1978 Q1 to 2009 Q4
  - Test set: 2010 Q1 to 2022Q1
- Data frequency: Quarterly
- Data resource: AREMOS

Empirical Analysis: VAR(4) Model

- A VAR (12) model

$$\begin{bmatrix} y_{1,t} \\ y_{2,t} \\ y_{3,t} \\ y_{4,t} \end{bmatrix} = \mu + \boldsymbol{\Phi}_1 \begin{bmatrix} y_{1,t-1} \\ y_{2,t-1} \\ y_{3,t-1} \\ y_{4,t-1} \end{bmatrix} + \boldsymbol{\Phi}_2 \begin{bmatrix} y_{1,t-2} \\ y_{2,t-2} \\ y_{3,t-2} \\ y_{4,t-2} \end{bmatrix} + \boldsymbol{\Phi}_3 \begin{bmatrix} y_{1,t-3} \\ y_{2,t-3} \\ y_{3,t-3} \\ y_{4,t-3} \end{bmatrix} +$$

$$\boldsymbol{\Phi}_4 \begin{bmatrix} y_{1,t-4} \\ y_{2,t-4} \\ y_{3,t-4} \\ y_{4,t-4} \end{bmatrix} + \cdots + \boldsymbol{\Phi}_{12} \begin{bmatrix} y_{1,t-12} \\ y_{2,t-12} \\ y_{3,t-12} \\ y_{4,t-12} \end{bmatrix} + \boldsymbol{\varepsilon}_t$$

where y_1 is unemployment rate, y_2 is GDP growth rate, y_3 is inflation rate, and y_4 is interest rate.

Python Code

- Install packages and enable us to use it

```
import numpy as np   # 矩陣處理包
import matplotlib.pyplot as plt  # 繪圖包
%matplotlib inline      #內嵌繪圖
import pandas as pd   #建立時序與資料結構包
import seaborn as sns #繪圖函式包
import datetime as dt #處理日期和時間包
from sklearn import (linear_model, metrics) #引入機器學習演算法
import statsmodels.formula.api as sm #線性迴歸程式包
from statsmodels.tsa.stattools import adfuller as adf #單根檢定包
from statsmodels.tsa.api import VAR #VAR模型
from sklearn.metrics import mean_squared_error as mse  #計算 MSE
```

Empirical Analysis: Time Series Plot

- Input data

```
df = pd.read_csv("VAR.csv", parse_dates=['Date'], index_col='Date')
print(df.shape)
df.head( )
```

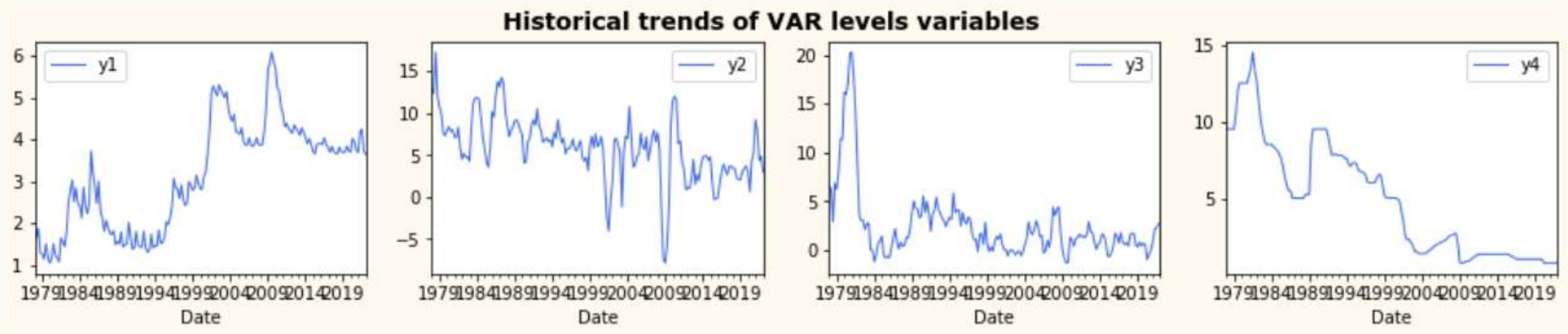
```
(177, 4)
```

Date	y1	y2	y3	y4
1978-01-01	1.870000	12.90	6.533589	9.5
1978-04-01	1.643333	12.35	6.251459	9.5
1978-07-01	1.870000	17.26	2.889159	9.5
1978-10-01	1.283333	11.92	6.862898	9.5
1979-01-01	1.260000	10.73	6.218566	9.5

Empirical Analysis: Time Series Plot

- Need to ensure stationarity of all time series variables.

```
def plot_vars(data, levels, color, leveltype):
    fig, ax = plt.subplots(1, 4, figsize=(16,2.5), sharex=True)
    palettes = ["blue", "green", "red", "orange", "purple", "black"]
    for col, i in dict(zip(levels, list(range(4)))).items():
      data[col].plot(ax=ax[i], legend=True, linewidth=1.0, color=color, sharex=True)
    fig.set_facecolor("floralwhite")
   fig.suptitle(f"Historical trends of VAR {leveltype} variables", fontsize=14,
   fontweight="bold", fontname="Verdana")
new_names = ["y1", "y2", "y3", "y4"]
plot_vars(df, levels=new_names, color="royalblue", leveltype="levels")
```



Empirical Analysis: ADF Test

Augmented Dickey-Fuller Test (ADF test)

```
def adf_test(series, title=''):
 print (f'Augmented Dickey-Fuller Test: {title}')
 result = adf(series.dropna(), autolag='AIC') # .dropna() handles differenced data
 labels = ['ADF test statistic', 'p-value', 'Number of lags used',
           'Number of observations used']
 out = pd.Series(result[0:4], index=labels)
 for k, v in result[4].items():
   out[f'critical value ({k})'] = v
 print(out)
 if result[1] <= 0.10:  # 有顯著性，推翻虛無假設
   print("Data has no unit root and is stationary")
 else:
   print("Data has a unit root and is non-stationary")
for col in df.columns:
adf_test(df[col], title=col)
print()
```

Empirical Analysis: ADF Test

- Augmented Dickey-Fuller Test (ADF test)

```
Augmented Dickey-Fuller Test: y1
ADF test statistic              -1.704744
p-value                          0.428670
Number of lags used              5.000000
Number of observations used    171.000000
critical value (1%)             -3.469181
critical value (5%)             -2.878595
critical value (10%)            -2.575863
dtype: float64
Data has a unit root and is non-stationary
```

```
Augmented Dickey-Fuller Test: y2
ADF test statistic              -2.269381
p-value                          0.182033
Number of lags used             10.000000
Number of observations used    166.000000
critical value (1%)             -3.470370
critical value (5%)             -2.879114
critical value (10%)            -2.576139
dtype: float64
Data has a unit root and is non-stationary
```

```
Augmented Dickey-Fuller Test: y3
ADF test statistic              -4.271813
p-value                          0.000497
Number of lags used              9.000000
Number of observations used    167.000000
critical value (1%)             -3.470126
critical value (5%)             -2.879008
critical value (10%)            -2.576083
dtype: float64
Data has no unit root and is stationary
```

```
Augmented Dickey-Fuller Test: y4
ADF test statistic              -1.358409
p-value                          0.602045
Number of lags used              1.000000
Number of observations used    175.000000
critical value (1%)             -3.468280
critical value (5%)             -2.878202
critical value (10%)            -2.575653
dtype: float64
Data has a unit root and is non-stationary
```

Empirical Analysis: ADF Test

- Differencing non-stationary variables:

```
for c in list(df.columns.values):
    df[c + "_diff"] =df[c] - df[c].shift(1)
# differencing(1)
    df_differenced = df.diff().dropna()
# do ADF test on differencing(1) dataframe
    for col in df_differenced.columns:
    adf_test(df_differenced[col], title=col)
    print( )
```

```
Augmented Dickey-Fuller Test: y1
ADF test statistic            -6.515933e+00
p-value                        1.071381e-08
Number of lags used            4.000000e+00
Number of observations used    1.710000e+02
critical value (1%)           -3.469181e+00
critical value (5%)           -2.878595e+00
critical value (10%)          -2.575863e+00
dtype: float64
Data has no unit root and is stationary
```

```
Augmented Dickey-Fuller Test: y2
ADF test statistic            -7.475963e+00
p-value                        4.913056e-11
Number of lags used            9.000000e+00
Number of observations used    1.660000e+02
critical value (1%)           -3.470370e+00
critical value (5%)           -2.879114e+00
critical value (10%)          -2.576139e+00
dtype: float64
Data has no unit root and is stationary
```

```
Augmented Dickey-Fuller Test: y4
ADF test statistic            -7.872408e+00
p-value                        4.944776e-12
Number of lags used            0.000000e+00
Number of observations used    1.750000e+02
critical value (1%)           -3.468280e+00
critical value (5%)           -2.878202e+00
critical value (10%)          -2.575653e+00
dtype: float64
Data has no unit root and is stationary
```

Empirical Analysis: Out-of-Sample Forecasting

```
start_date = '2010-01-01'
end_date = '2022-01-01'
```

← test set

```
def var_create(columns, data):
data = data[columns]
data = data.dropna(axis=0)
data.index.to_period(freq='Q')

# Split dataset and run VAR on the trained part
  data_train = data.loc['1978-04-01':'2009-10-01', : ]  # training set
  var_train = VAR(data_train)
  results = var_train.fit(12) # run a VAR (12) model or model.select_order (maxlags=12)
                              .summary()
  lag_order = results.k_ar
  forecasted = pd.DataFrame(results.forecast(data_train.values[-lag_order:],
   49))  # Forecast 49 Quarters

 # Rename forecasted columns
    forecasted_names = list(forecasted.columns.values)
    data_train_names = list(data_train.columns.values)
```

Empirical Analysis: Out-of-Sample Forecasting

```
 var_dict = dict(zip(forecasted_names, data_train_names))
    for f,t in var_dict.items():
    forecasted = forecasted.rename(columns={f:t + '_fcast'})

  forecasted.index = pd.date_range(start=start_date, periods=forecasted.shape[0],
  freq='QS')
  forecasted.index.to_period(freq='Q')
  forecasted.index.names = ['date']

# Parse together forecasted data with original dataset
  final_data = pd.merge(forecasted, data, left_index=True, right_index=True)
  final_data = final_data.sort_index(axis=0, ascending=True)
  final_data = pd.concat([data_train, final_data], sort=True, axis=0)
  final_data = final_data.sort_index(axis=0, ascending=True)

  var_mse = mse(final_data.loc[start_date:end_date,'y3_fcast'],
                final_data.loc[start_date:end_date,'y3']) # 計算 MSE value
  return   var_mse, final_data
```

Empirical Analysis: Forecasting Results

- Estimate VAR(12) model and print out the MSE value

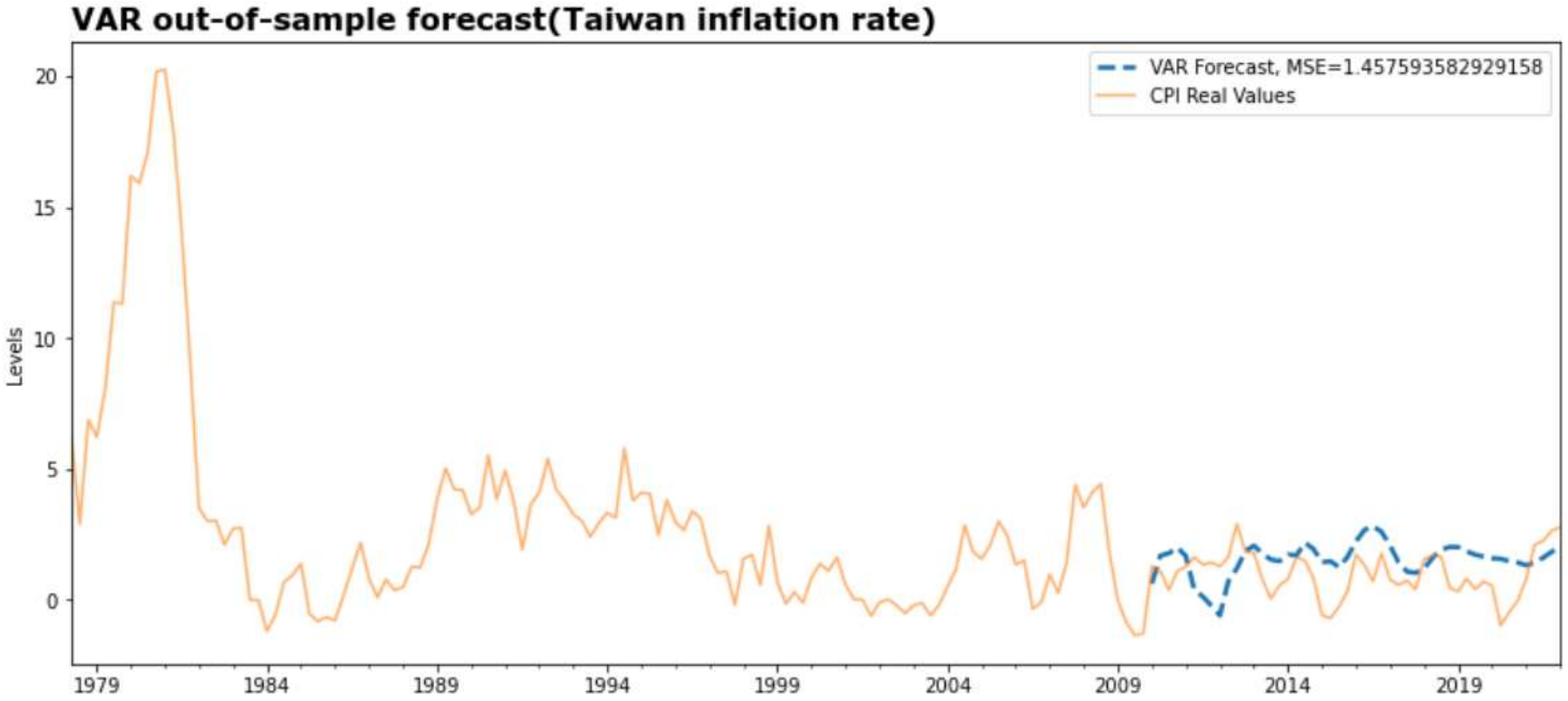
```
mse1, df1 = var_create(columns=['y1_diff','y2_diff','y3','y4_diff'],data=df)
print(f"The mean squared error between the forecasted and actual values is {mse1}")
```

The mean squared error between the forecasted and actual values is 1.45759.

- Visualize the forecasts against the actual values

```
def plot_cpi(final_data, var_mse):
    fig, ax = plt.subplots(figsize=(14,6))
    colors = sns.color_palette('deep', 8)
    final_data['y3_fcast'].plot(ax=ax, legend=True, linewidth=2.5, linestyle='dashed')
    final_data['y3'].plot(ax=ax, legend=True, alpha=0.6, linestyle='solid')
    ax.set_title("VAR out-of-sample forecast(Taiwan inflation rate)", fontsize=16,
                 fontweight="bold", fontname="Verdana", loc="left")
    ax.set_ylabel('Levels', fontname='Verdana')
    ax.legend([f'VAR Forecast, MSE={var_mse}', 'CPI Real Values'])
    plot_cpi(final_data=df1, var_mse=mse1)
```

Empirical Analysis: Forecasting Results



Overall, we can see that the VAR model provides a similar trend of changes with the actual data.

Thank You for Your Time and Attention!