Time Series Final Project

0853411 劉書維

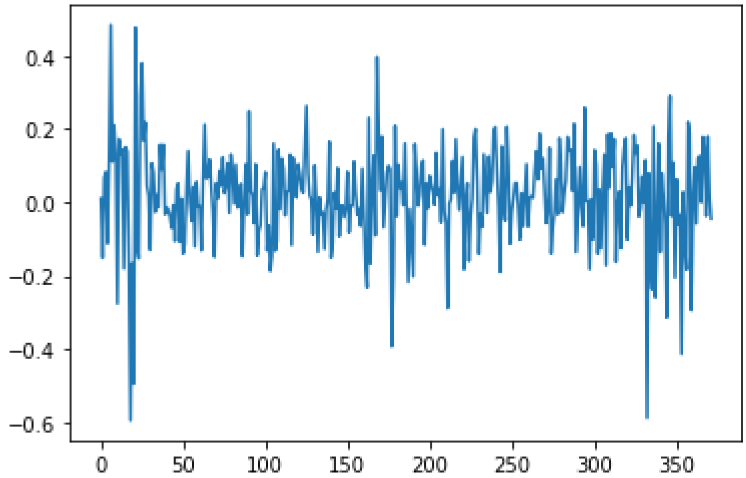
Problem 1. GARCH and SV models

1. **Transform the returns into log-return, and plot-out the time series of the log-return.**

利用計算式將 returns 轉換成 log-returns，程式如下：

```
for i in range(len(n_return)):
    b = np.log(n_return[i]+1)
    log_return.append(b)
```

並且畫出結果：



2. **Build a GARCH model on this data.**

要建一個 GARCH model 需要決定 p、q 的 order，在不知道機率分布的情況下，假設是 normal distribution，所以我使用兩層 for 迴圈，將 p= 1~10 與 q= 1~10 走訪一次，以最低的 AIC 當指標，來決定最好的 p 和 q，程式碼如下：

```python
import arch

low_aic = 0
for i in range(1,11):
    for j in range(1,11):
        model = arch.arch_model(log_return,vol='Garch', p=i, o=0, q=j, dist='Normal')
        result = model.fit(update_freq=0)
        if result.aic < low_aic:
            low_aic = result.aic
            best_p = i
            best_q = j
```

最後得到 p = 1 和 q = 1，AIC = -475.28 然後套入 model 中得到模型資訊如下：

```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                      y   R-squared:                      -0.000
Mean Model:             Constant Mean   Adj. R-squared:                 -0.000
Vol Model:                      GARCH   Log-Likelihood:                241.639
Distribution:                  Normal   AIC:                          -475.278
Method:            Maximum Likelihood   BIC:                          -459.603
                                        No. Observations:                  372
Date:                Mon, Jun 22 2020   Df Residuals:                      368
Time:                        21:40:34   Df Model:                            4
                               Mean Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
mu             0.0164  6.797e-03      2.416  1.568e-02 [3.101e-03,2.975e-02]
                            Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
omega      8.2428e-04  3.968e-04      2.077  3.777e-02 [4.656e-05,1.602e-03]
alpha[1]       0.0618  2.569e-02      2.405  1.617e-02   [1.144e-02, 0.112]
beta[1]        0.8853  3.747e-02     23.629 1.935e-123     [ 0.812, 0.959]
============================================================================

Covariance estimator: robust
```

並且可以知道各係數，如：mu = 0.0164、omega = 0.00083、alpha = 0.0618、beta = 0.8853

3. **Based on the GARCH model you fit, compute 1-step to 5-step ahead volatility forecasts at the forecast origin December 2003.**

藉由 forecast 可以預測最後一天的數值，而且可以做出 1~5 step 的預測，程式碼如下：

```
forecasts = result.forecast(horizon=5)
print(forecasts.mean.iloc[-1:])
```

結果輸出如下圖：

```
In [4]: forecasts = result.forecast(horizon=5)
   ...: print(forecasts.mean.iloc[-1:])
          h.1       h.2       h.3       h.4       h.5
371  0.016423  0.016423  0.016423  0.016423  0.016423
```

神奇的是他預測的五個 log returns 數值都是 0.016423，實際數值是-0.045，所以還是有蠻大的落差。

4. Build an SV model on this data instead

可以建立一個函式，決定要預測的步數，和建立 volatility、nu 之後再將可以觀察到的 log returns 輸入，程式碼如下：

```
import pymc3 as pm

def make_stochastic_volatility_model(data):
    with pm.Model() as model:
        step_size = pm.Exponential('step_size', 1)
        volatility = pm.GaussianRandomWalk('volatility', sigma=step_size, shape=len(data))
        nu = pm.Exponential('nu', 0.1)
        returns = pm.StudentT('returns', nu=nu, lam=np.exp(-2*volatility), observed=data)

    return model

stochastic_vol_model = make_stochastic_volatility_model(log_return)
```

5. Based on the SV model you fit, compute 1-step to 5-step ahead volatility forecasts at the forecast origin December 2003.

可以從剛剛的 model 中進行 sample 然後多次預測出所需結果，程式碼如下：

```
with stochastic_vol_model:
    prior = pm.sample_prior_predictive(500)

with stochastic_vol_model:
    trace = pm.sample(1, tune=10, cores=1)

with stochastic_vol_model:
    posterior_predictive = pm.sample_posterior_predictive(trace)

print(posterior_predictive['returns'][1][-1])
```

即可求得最後一天的預測數字，實際數值是-0.045，結果如下表：

1 step	2 step	3 step	4 step	5 step
0.0089	-0.055	-0.163	0.056	-0.024

可以看出以 2 step 的結果較接近真實數值，所以更具有參考性。

6. **For this data, between the GARCH and SV models, which one will you prefer? Why?**

我比較喜歡 SV 的 model，因為就我的程式出來的數值，SV 的 5 個 steps 都有變化，而且 SV 的 function 是我依需要的資訊求出來，並用 sm 的套件做 sample 與預測，也因此，我對結果的準確率較有信心！

Problem 2. VAR model and Cointegration

1. **Fit a VAR model on this data**

要 fit 一個 VAR model 必須先檢查它的 attributes 是否都是 stationary，如果是 non-stationary，就要先做差分，來維持穩定性，檢查的方法是利用著名的 Augmented Dickey-Fuller Test，然後使用 p-value 來決定接受或拒絕此假設，檢查的程式碼如下：

```
def adfuller_test(series, signif=0.05, name='', verbose=False):
    r = adfuller(series, autolag='AIC')
    output = {'test_statistic':round(r[0], 4), 'pvalue':round(r[1], 4),
    p_value = output['pvalue']
    def adjust(val, length= 6): return str(val).ljust(length)
```

第一次結果如下：

```
   Augmented Dickey-Fuller Test on "1-year"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -2.0947
No. Lags Chosen       = 19
Critical value 1%     = -3.441
Critical value 5%     = -2.866
Critical value 10%    = -2.569
=> P-Value = 0.2467. Weak evidence to reject the Null Hypothesis.
=> Series is Non-Stationary.

   Augmented Dickey-Fuller Test on "3-year"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -2.0329
No. Lags Chosen       = 12
Critical value 1%     = -3.441
Critical value 5%     = -2.866
Critical value 10%    = -2.569
=> P-Value = 0.2724. Weak evidence to reject the Null Hypothesis.
=> Series is Non-Stationary.
```

可以看出一年期的殖利率和三年期的殖利率都是 Non-Stationary，所以必須做差分，可以用 pandas 中的 diff() 來協助，並再做一次 Augmented Dickey-Fuller Test 檢查是否穩定，結果如下：

```
   Augmented Dickey-Fuller Test on "1-year"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -6.1392
No. Lags Chosen       = 19
Critical value 1%     = -3.441
Critical value 5%     = -2.866
Critical value 10%    = -2.569
=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.

   Augmented Dickey-Fuller Test on "3-year"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -6.3655
No. Lags Chosen       = 19
Critical value 1%     = -3.441
Critical value 5%     = -2.866
Critical value 10%    = -2.569
=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.
```

所以我們就可以開始建立一個 VAR model 了！利用 VAR 的套件即可建立，程式碼如下：

```
model = VAR(df_differenced)

x = model.select_order(maxlags=20)
print(x.summary())
```

我們將 lag 最大設置為 20 並且去觀察，若是以 AIC 當參考指標的話，是 19 的參數最好，結果如下：

	AIC	BIC	FPE	HQIC
0	-5.590	-5.575	0.003735	-5.584
1	-5.754	-5.709	0.003171	-5.736
2	-5.825	-5.751*	0.002953	-5.796
3	-5.834	-5.730	0.002926	-5.794
4	-5.839	-5.706	0.002911	-5.787
5	-5.850	-5.687	0.002879	-5.787
6	-5.916	-5.723	0.002695	-5.841*
7	-5.922	-5.699	0.002681	-5.835
8	-5.923	-5.671	0.002676	-5.825
9	-5.928	-5.646	0.002663	-5.818
10	-5.925	-5.613	0.002672	-5.804
11	-5.931	-5.590	0.002656	-5.798
12	-5.950	-5.579	0.002607	-5.805
13	-5.959	-5.559	0.002582	-5.803
14	-5.956	-5.526	0.002591	-5.789
15	-5.958	-5.498	0.002587	-5.779
16	-5.963	-5.474	0.002572	-5.773
17	-5.959	-5.440	0.002582	-5.757
18	-5.967	-5.419	0.002562	-5.754
19	$\boxed{-5.991^*}$	-5.412	0.002503*	-5.765
20	-5.988	-5.380	0.002510	-5.751

2. **Use the fitted VAR model to produce 1-step to 12-step ahead forecasts of the interest rates, assuming that the forecast origin is March 2004.**

我們將剛剛建置的 model，更改一下參數就可以多次計算，預測出 1-step to 12-step ahead 的一年期與三年期殖利率，原值是：1.19%與 2%，程式碼與預測結果如下：

```python
lag_order = model_fitted.k_ar

forecast_input = df_differenced.values[-lag_order:]
fc = model_fitted.forecast(y=forecast_input, steps=lag_order)
df_forecast = pd.DataFrame(fc, index=df.index[-lag_order:], columns=df.columns + '_2d')
print(df_forecast)
```

1 step	-0.09796	-0.106791	7 step	0.040793	0.024522
2 step	-0.007397	0.021481	8 step	0.007999	0.009345
3 step	0.001081	-0.013013	9 step	-0.014226	-0.025953
4 step	-0.011423	-0.003908	10 step	-0.022509	-0.026345
5 step	-0.025273	0.013386	11 step	-0.053975	-0.017804
6 step	-0.055334	-0.034494	12 step	0.090832	0.063017

模型預測出來的結果可以說是差強人意，因為利率基本上都是正值，但是模型預測出來卻是負值，且一年期的殖利率居然會高於三年期的，也是不合理的地方，所以就只有 8 step 出來的數值較合理。

3. **Are the two interest rate series cointegrated ? Use 5 % significance level to perform the test.**

我是用套件來做 cointegration test，程式碼如下：

```python
def cointegration_test(df, alpha=0.05):
    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), ' =>  ' , trace >

cointegration_test(df)
```

結果如下圖，可以看出只有 1-year 的殖利率是 True（cointegration）：

```
Name    ::  Test Stat > C(95%)    =>   Signif
 ----------------------------------------
1-year ::  25.44      > 12.3212   =>   True
3-year ::  0.76       > 4.1296    =>   False
```

Problem 3. ARIMA model and Kalman Filter

1. Fit an ARIMA(0, 1, 1) model on this data.

可以直接使用 ARIMA 的套件協助我們建立 model，程式碼如下：

```
model = ARIMA(returns, order=(0,1,1))
model_fit = model.fit(disp=0)
print(model_fit.summary())
```

Model 資訊如下圖：

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                    D.y   No. Observations:                  339
Model:                 ARIMA(0, 1, 1)   Log Likelihood                -761.182
Method:                       css-mle   S.D. of innovations              2.278
Date:                Mon, 22 Jun 2020   AIC                           1528.365
Time:                        23:17:11   BIC                           1539.843
Sample:                             1   HQIC                          1532.939

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0018      0.007     -0.244      0.807      -0.016       0.012
ma.L1.D.y     -0.9448      0.027    -35.538      0.000      -0.997      -0.893
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
MA.1            1.0584           +0.0000j            1.0584            0.0000
-----------------------------------------------------------------------------
```

可以觀察到模型截距是-0.0018，MA 的係數是-0.9448。

2. Estimate the local trend model in Equations (11.1) and (11.2) in the slide Week 11-1.

接下來，將 data 丟入 pyflux 中的 local trend 的套件中，並假設此分布是常態分佈，就可以得到這個模型的資訊，程式碼如下：

```
np_returns = np.array(returns)
model = pf.LocalTrend(data=np_returns, family=pf.Normal())
result = model.fit()
print(result.summary())
```

結果如下圖，可以觀察到其預測的數值和其他資訊：

```
LLT
=======================================================
==================================================
Dependent Variable: Series                              Method: MLE
Start Date: 0                                           Log Likelihood:
-776.3499
End Date: 339                                           AIC: 1558.6998
Number of observations: 340                             BIC: 1570.1866
===============================================================================
===========================
Latent Variable                         Estimate   Std Error  z        P>|z|
95% C.I.
======================================== ========== ========== ========
======== =========================
Sigma^2 irregular                       4.88754113
Sigma^2 level                           0.02138289
Sigma^2 trend                           3.3623e-06
===============================================================================
```

3 . Obtain time plots for the filtered variables and smoothed variables with

pointwise 95 % confidence interval.

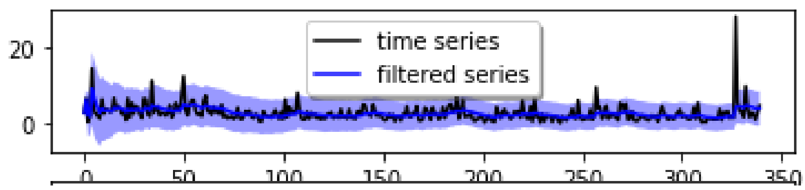
可以利用 python 中的 pydlm 套件，就可以繪出 filtered 和 smoothed 的 95 信

賴區間，程式碼如下：

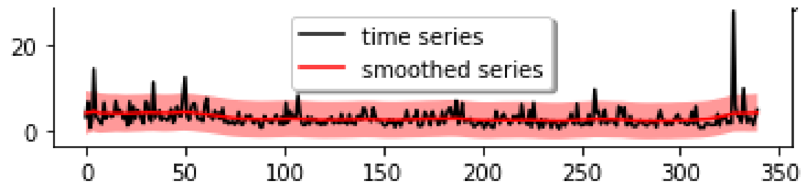
```
linear_trend = trend(degree=1, discount=0.95, name='linear_trend', w=10)
simple_dlm = dlm(returns)+ linear_trend
simple_dlm.fit()

simple_dlm.turnOff('data points')
simple_dlm.plot()
```

Filtered 畫出的結果：



Smoothed 畫出的結果：


time series
smoothed series
20
0
0
50
100
150
200
250
300
350