一一五、Python AI 金融交易策略应用(5)：时间数列交易策略应用范例

回想我当年在台湾银行交易室的时光，不论是在中国信托银行、中华开发银行，还是永丰银行，其实我几乎没有看到有交易员使用象样的量化技巧，来进行交易预测。就以我离开银行前的最后一份工作为例，记得那时刚好遇到美国总统大选，外汇交易主管就跟大家一起起哄，要来赌汇率的走势。结果交易员就开始各抒己见，各自猜测涨跌的想法。

我对此并不太意外，一方面那时候量化交易的技巧在台湾并不流行，交易室的工作方式大都是单兵作战，另一方面，因为大家好像都不熟悉使用量化模型来进行预测。当然，我也是觉得很可惜，大家都读到硕士以上的学历，计量与数量方法也学得不少，为什么真的上战场了，却只有小米加步枪。那些可以当成加农炮的重武器，怎么全都不知道丢到哪里去了？

以我的认知，国外的计量学者之所以要发展出这些时间数列的方法，目的就是要能预测金融市场的走势。尤其，当你手上只有价格的历史数据时，你能否由资料本身的统计特性，搭配这些时间数列的模型，做出可以信赖的预测。

因此，在这份课程中，我整理了各类与交易策略有关的时间数列分析方法，来搭配我们可能发展的趋势与反转策略使用。例如，使用 ADF(Augmented Dicky-Fuller)检定，来判定一个价格数列是否具有均数回复(Mean Reversion)的特性(图一)。又如使用自我序列相关函数(Auto-Correlation Function)来研究股票的前后期价格，自我相关的现象(图二)。另外，我也介绍如何使用时间数列分解的技巧，将其分解为趋势、季节与残差的部分(图三)，然后再各个预测，分别击破。

当然，ARIMA 与 GARCH 模型是整个时间数列分析方法的重头戏，自然不能错过。我也举例说明，如何撰写自动执行的 R 语言程序，帮我们估计出 ARIMA(3,0,3)的形式，是 SP500 对数报酬数列的最佳选择(图四)。而针对 FTSE100 指数，以 GARCH(1,1)的形式，最能描述其波动性的结构(图五)。

圖一

✧ 使用 ADF(Augmented Dicky-Fuller)測試 Amazon 股價的均數回復特性

甲、ADF測試

◆ 針對時間數列本身是否具有均數回復的特性，我們可以使用 Augmented Dickey-Fuller (ADF)測試。

➢ 一個具有均數回復特性的價格數列，可以用下面的線性模型來描述其價格變動，

$$\Delta y_t = \mu + \beta t + \lambda y_{t-1} + \alpha_1 \Delta y_{t-1} + \alpha_2 \Delta y_{t-2} + \ldots + \alpha_k \Delta y_{t-k} + \varepsilon_t$$

➢ ADF 測試是針對 λ 是否為 0 所進行的。

✓ 如果 λ 不為 0，則下一期的變動量，與本期數值有關，則時間數列就不是隨機漫步。

✓ 如果 λ 為 0，則下一期的變動量，與本期數值無關，則時間數列就是隨機漫步。

➢ 檢定量是將 λ 的估計值，除以 λ 的標準差，我們預期 λ 為負值。

✓ 因此檢定量負值愈大且愈顯著，帶表隨機漫步性愈低。

✓ 這也是我們希望看到的結果，表示時間數列具有較高的可測性。

In [1]:
```
# Import the Time Series library
import statsmodels.tsa.stattools as ts
from datetime import datetime
import pandas as pd

# Download the Amazon OHLCV data from 1/1/2000 to 1/1/2015
amzn = pd.read_csv("amazon.csv", index_col=0, parse_dates=True, infer_datetime_format=True)

# Output the results of the Augmented Dickey-Fuller test for Amazon
# with a lag order value of 1
result = ts.adfuller(amzn['Adj Close'], 1)

print(result)
print(result[0])
print(result[4]['5%'])
```

◆ 輸出

```
(0.049177575166449786,          // test-statistics
 0.9624149463256304,            // p-value
 1,
 3771,                          // # of data points
 {'1%': -3.4320852842548395,    //  1% critical value of test statistics
  '5%': -2.8623067530084247,    //  5% critical value of test statistics
  '10%': -2.5671781529820348    // 10% critical value of test statistics
 },
 19576.11604147387)

0.049177575166449126
-2.8623067530084247
```

➢ 測試統計量 0.04917 高於 1%，5%，10%的統計值，無法拒絕虛無假設，$\lambda = 0$。

➢ λ 不是顯著的異於 0，有相當的隨機性，沒有 Mean-Reversion 的現象。

◆ 大部分 Equity 為 Geometric Brownian Motion，Random Walk。

圖二

✧ 使用 Autocorrelation Function 來研究 Microsoft 與 SP500 價格的序列相關性

丙、Financial Data

◆ Microsoft & SP500

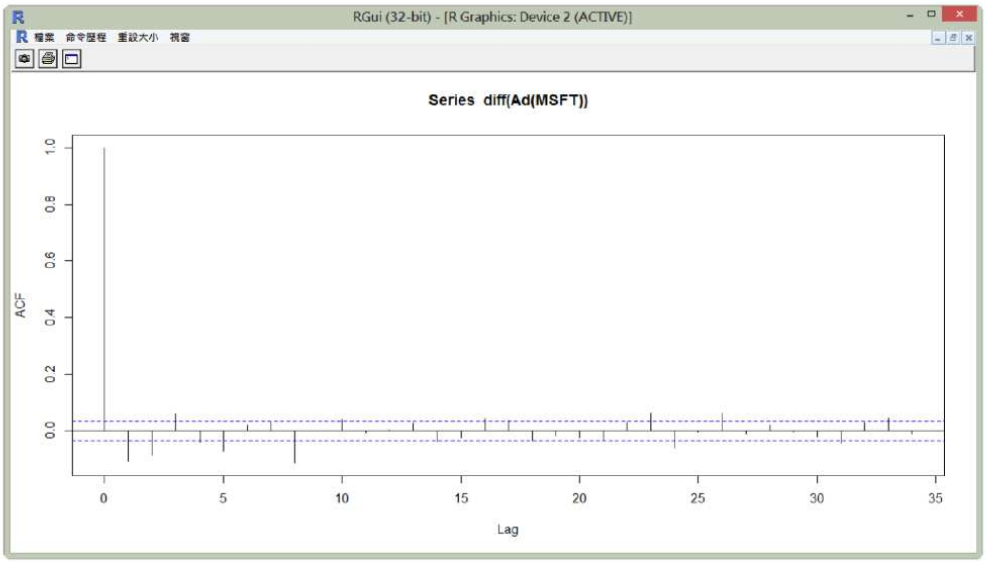
➢ 安裝

```
> install.packages('quantmod')
```

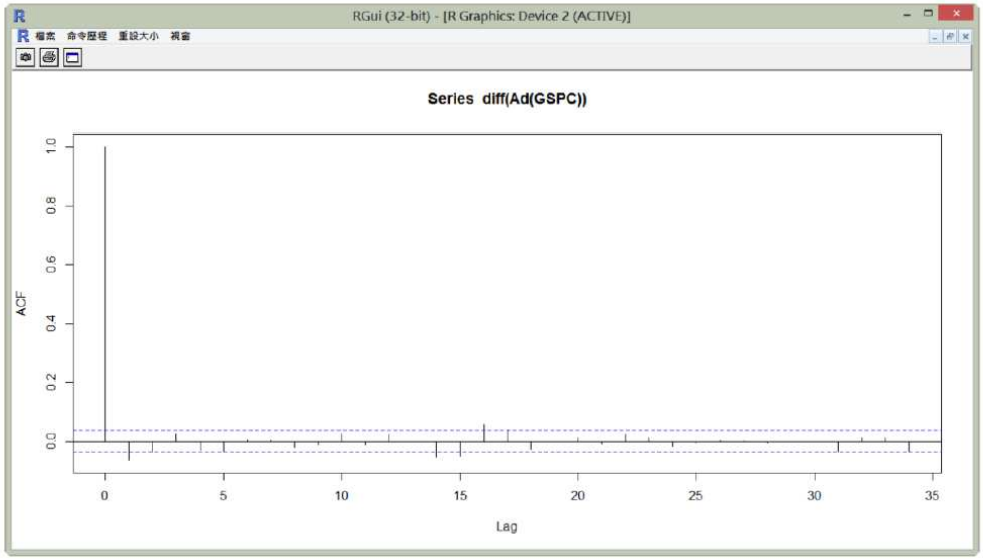
➢ 執行

```
> require('quantmod')
> getSymbols('MSFT', src='yahoo', from="2007-01-03", to="2019-02-22")
> head(MSFT)
> tail(MSFT)
> acf(diff(Ad(MSFT)), na.action = na.omit)

> getSymbols('^GSPC', src='yahoo', from="2007-01-03", to="2019-02-22")
> acf(diff(Ad(GSPC)), na.action = na.omit)
```

➢ MSFT 有序列相關，負值明顯。



➢ SP500 有序列相關，但不太明顯。



圖三

✧ 使用 Statsmodels 套件 Seasonal_Decompose 方法，分解時間數列

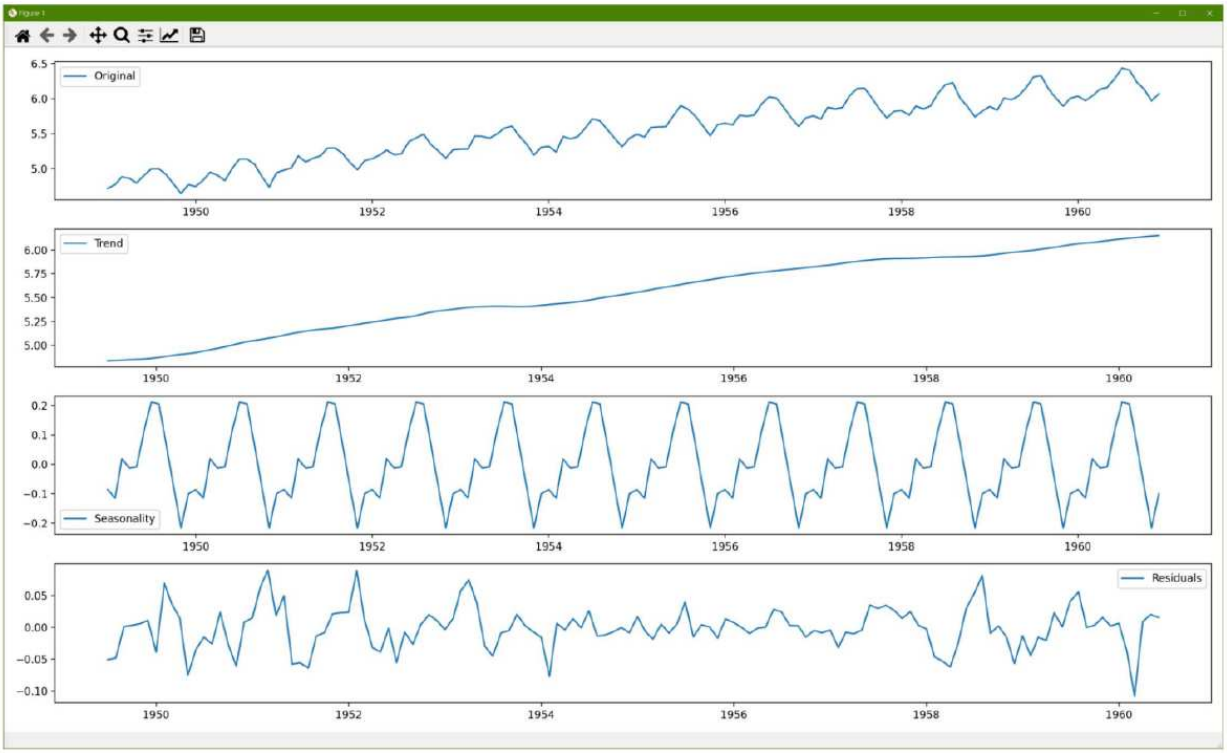
◆ 分解：

- ➢ 可以將時間數列分解成三個主要成份
  - ✓ 趨勢：長期趨勢
  - ✓ 季節：季節性變動
  - ✓ 以及剩餘的殘差成份
- ➢ 可以有相乘與相加的函數型式

$$y_t = T_t \times S_t \times e_t$$

$$y_t = T_t + S_t + e_t$$

```
>>> from statsmodels.tsa.seasonal import seasonal_decompose

>>> decomposition = seasonal_decompose(ts_log)
>>> trend = decomposition.trend
>>> seasonal = decomposition.seasonal
>>> residual = decomposition.resid

>>> print(trend)
>>> print(seasonal)
```



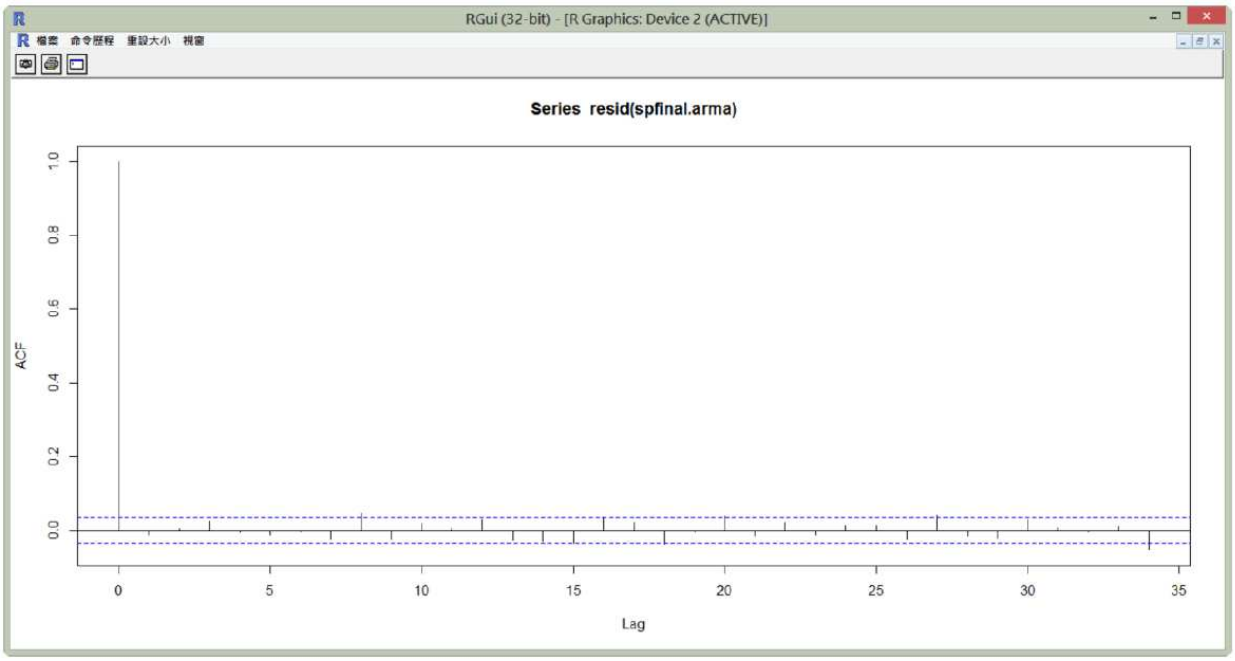
圖四

✧ 使用 AIC 數值，選取 SP500 報酬時間數列 ARIMA 模型的 Order(3,0,3)

丁、Financial Data

◆ SP500 估計，Code：R3_ARIMA.R。

```
> require('quantmod')

> getSymbols('^GSPC', src='yahoo')
 [1] "GSPC"

> sp = diff(log(Cl(GSPC)))
>
> spfinal.aic <-Inf
> spfinal.order <- c(0, 0, 0)
> for (i in 0:4) for (j in 0:4) {
+   spcurrent.aic <- AIC(arima(sp, order=c(i, 0, j)))
+   if (spcurrent.aic < spfinal.aic) {
+     spfinal.aic <- spcurrent.aic
+     spfinal.order <- c(i, 0, j)
+     spfinal.arma <- arima(sp, order=spfinal.order)
+   }
+ }

> spfinal.aic
[1] -18173.01

> spfinal.order
[1] 3 0 3

> acf(resid(spfinal.arma), na.action=na.omit)
```



圖五

✧ 使用 R 的 tseries 套件，估計 FTSE100 報酬的 GARCH 模型 Order(1,1)

甲、GARCH模型

◆ GARCH 表“一般化的 ARCH 模型”(Generalized Auto-Regressive Conditional Heteroskedasticity Model)，由 Bollerslev (1986)所提出。

➢ 針對前式中的殘差向，定義一個 GARCH(p, q)過程如下，

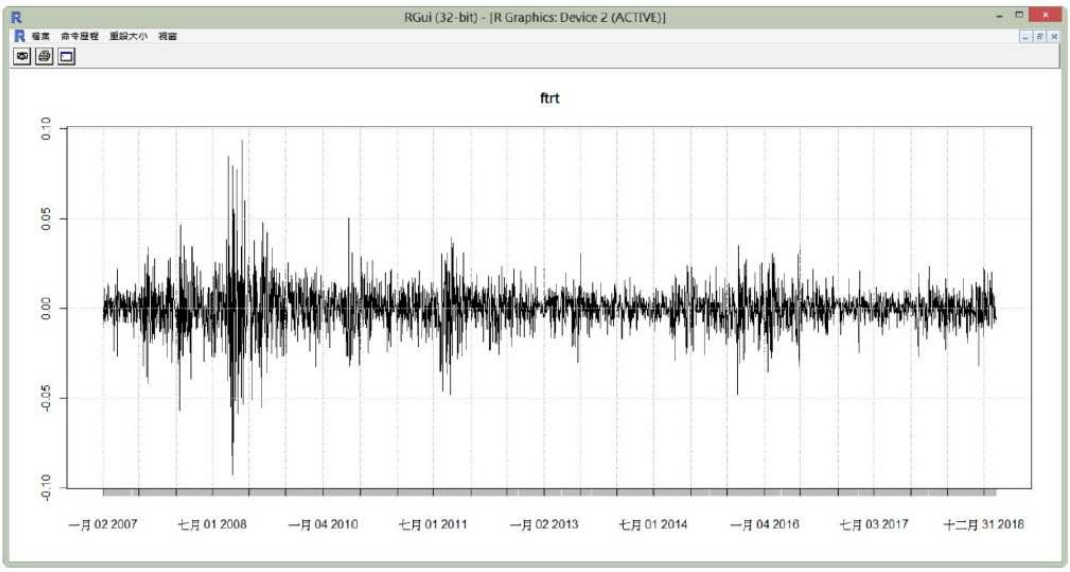
$$\varepsilon_t = \sigma_t e_t \ , \ e_t \sim N(0,1)$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2$$

➢ 對於所有的 i，e_t 與 ε_{t-i} 為獨立的。

◆ Code：R4_GARCH.R。

```
> require(quantmod)
> getSymbols("^FTSE")
> ftrt = diff(log(Cl(FTSE)))
> plot(ftrt)
```




```
> require(tseries)
> ft.garch <- garch(ft, trace=F)

> summary(ft.garch)

Call:
garch(x = ft, trace = F)

Model:
GARCH(1,1)
```