EXCEL VBA 進階班

Lecture 5

SECTION 7. USER-DEFINED TYPE

TimeSeries (2)

回顧 TIMESERIES

- 上次講述了時間序列資料的建立與資料整理搜尋。
- 這次開始利用時間序列資料進行一些指標計算與回測。
- 由於多筆資料較為複雜，此處用最出建構的單筆時間序列來進行與改進。

回顧 TIMESERIES

- 回顧現在的時間序列資料長相：

```
' 為了可以有 Null，我們把 dates 與 values 改為 Variant，但記住:
' dates 裡面要使用日期陣列或 Null
' values 裡面要是 double 陣列或 Null
Public Type TimeSeries
    dates As Variant
    values As Variant
End Type
```

回顧 TIMESERIES

- 在開始之前，為方便我們輸入資料，我們可以把建構時間序列的方式寫成一個函數。
- 如下頁所示：

```
'輸入的函數
Function fromWorksheet(dateRange As Range, valueRange As Range) As TimeSeries
Dim nData As Integer: nData = dateRange.Count
ReDim dates(nData) As Date
ReDim values(nData) As Double
Dim output As TimeSeries

Call rangeToArray(dateRange, dates)
Call rangeToArray(valueRange, values)

output.dates = dates
output.values = values
fromWorksheet = output
End Function
```

```
' 輸入陣列
Private Sub rangeToArray(thisArray As Variant, inputRange As Range)
Dim inputValues As Variant: inputValues = inputRange.Value
Dim elem As Variant
Dim counts As Integer

For Each elem In inputValues
    counts = counts + 1
    thisArray(counts) = elem
Next elem
```

USER-DEFINED TYPE

- 回到了 TimeSeries 的資料型態，現在我們要建立一些函數，讓我們可以對時間序列資料做更多的分析。
- 第一步最簡單的就是計算報酬率函數，我們可以寫一個 calculateReturnRate 函數來計算報酬率。
- 一開始只讓他有計算日報酬的功能。

USER-DEFINED TYPE

```
Function calculateReturnRate(inputSeries As TimeSeries) As TimeSeries

Dim nData As Integer: nData = UBound(inputSeries.dates)
ReDim returnDates(nData - 1) As Date
ReDim returnRates(nData - 1) As Double
Dim i As Integer

For i = 2 To nData
    returnDates(i - 1) = inputSeries.dates(i)
    returnRates(i - 1) = Log(inputSeries.values(i)) - Log(inputSeries.values(i - 1))
Next i

Dim outputSeries As TimeSeries
outputSeries.dates = returnDates
outputSeries.values = returnRates
calculateReturnRate = outputSeries
End Function
```

USER-DEFINED TYPE：練習

- 如果要計算的為 N 天的報酬，要如何修改函數？
- 當資料數量其實不足以算那麼多天的報酬率時該如何是好？(後半部錯誤處理會詳述可以參考方法)

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 接下來，擁有了資料篩選跟輸入方法，以及簡單的報酬率計算後，我們可以來精進時間序列資料的用途。
- 現在來創造一個自我迴歸模型（ autoregressive model ）。
- 自我迴歸模型是統計上一種時間序列的模型，用同一變數，的前幾期來預測該其結果。

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 另 r_t 為第 t 期之報酬率，一個一階自我迴歸模型（first-order autoregressive model，簡寫為 AR(1)）如下：

$$r_t = \beta_0 + \beta_1 r_{t-1} + \varepsilon_t$$

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- ε_t 也叫白噪音 (Whit Niose)，有以下特徵 :

$$E(\varepsilon_t) = 0$$

$$E(\varepsilon_t^2) = \sigma^2$$

$$E(\varepsilon_i \varepsilon_j) = 0, \forall i \neq j$$

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 同理，越多階代表利用前面越多期之歷史資料 :
- AR(2) :

$$r_t = \beta_0 + \beta_1 r_{t-1} + \beta_2 r_{t-2} + \varepsilon_t$$

- AR(3) :

$$r_t = \beta_0 + \beta_1 r_{t-1} + \beta_2 r_{t-2} + \beta_3 r_{t-3} + \varepsilon_t$$

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 對於迴歸分析 Excel 本身已經有很方便的函數名為 LINEST。
- 我們需要的只有把資料整理為它所需的格式。
- LINEST 在使用上有幾點須注意：

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 現在我們有以下資料 :

Y	X1	X2
0.111884	0.902502	0.195786
0.221955	0.088567	0.259479
0.803004	0.823109	0.237763
0.104557	0.286423	0.391374
0.369698	0.323302	0.283083
0.491955	0.891184	0.816136
0.529646	0.042012	0.06004
0.358225	0.56602	0.480351
0.358317	0.536334	0.01811

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 我們想要估計 :

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

- 在使用 LINEST 上可以如以下使用方式 :

{=LINEST(A2:A10,B2:C10,TRUE,FALSE)}

E	F	G	H
Beta 2	Beta 1	Beta 0	
-0.07439	0.165617	0.312741	

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 第一組資料為因變數 Y，後面的矩陣為自變數 X，每一個 column 代表一個自變數。
- 第三個引數為是否需要常數項 β_0，True 為需要。
- 第四個為是否需要額外的統計檢定結果，False 為不需要。
- 要注意的是，估計參數的順序是反的。

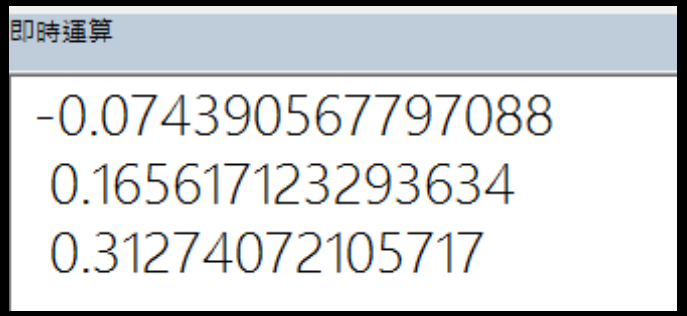
USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 現在來逐一測試各種情況下 LINEST 在 VBA 的結果，一般用二維陣列輸入的情況：

```
Sub linestExample()

Dim y As Variant: y = Range("A2:A10").Value
Dim x As Variant: x = Range("B2:C10").Value

Dim betaVec As Variant: betaVec = Application.LinEst(y, x, True, False)
Dim beta As Variant
For Each beta In betaVec
    Debug.Print beta
Next beta
End Sub
```

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 出來結果與原來 Excel 上一樣。



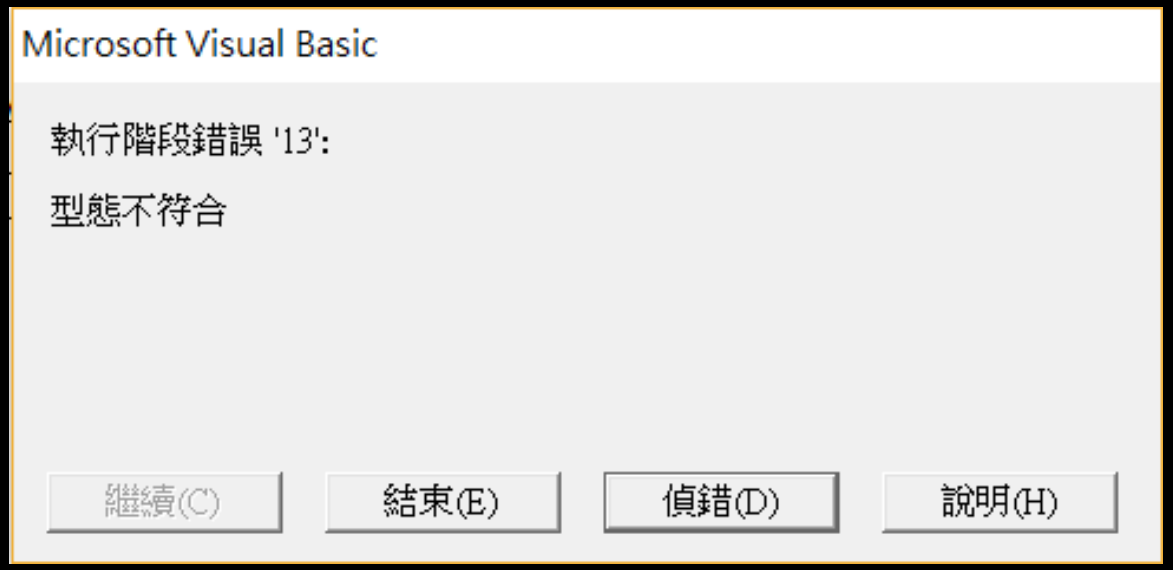
USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 但如果我們將 Y 變為一個一維陣列呢？

```
Sub linestExample()

Dim y As Variant: y = Range("A2:A10").Value
Dim x As Variant: x = Range("B2:C10").Value
Dim i As Integer
Dim yVec(9) As Double
For i = 1 To 9
    yVec(i) = y(i, 1)
Next i

Dim betaVec As Variant: betaVec = Application.LinEst(yVec, x, True, False)
Dim beta As Variant
For Each beta In betaVec
    Debug.Print beta
Next beta
End Sub
```

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 會出現以下錯誤：



USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

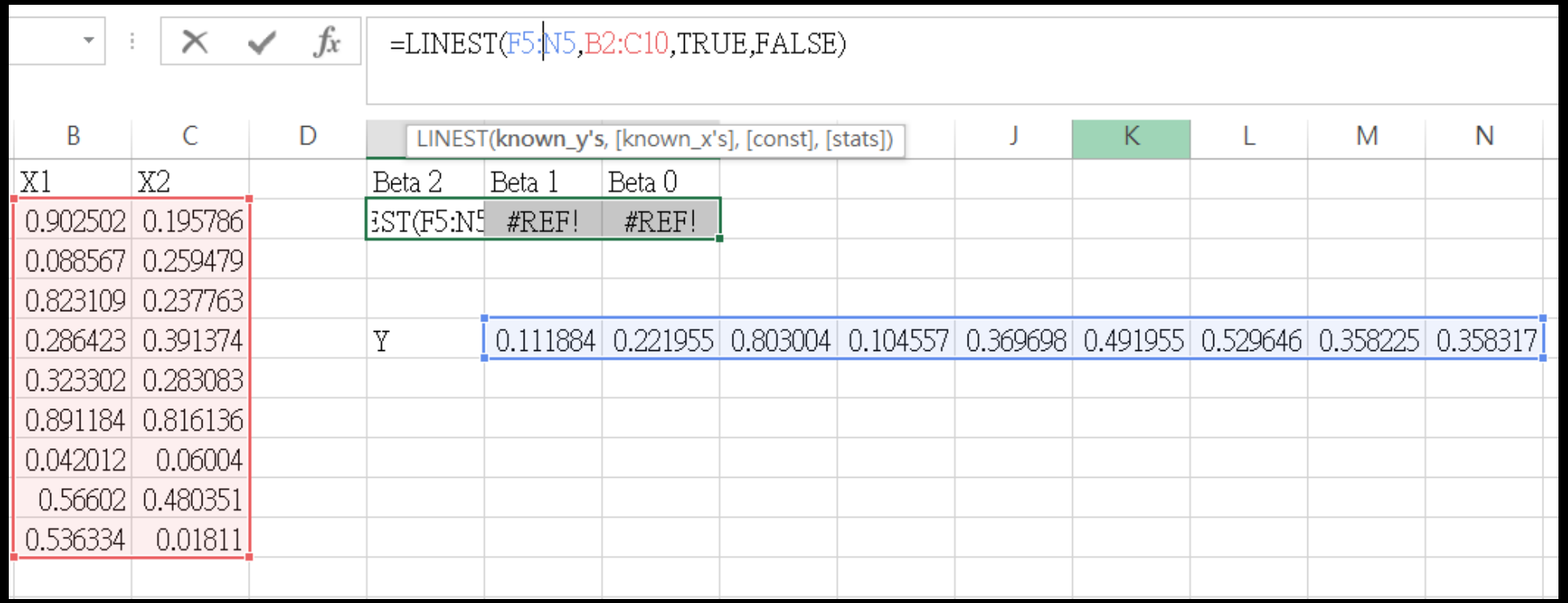
- 按下偵錯錯誤處是在於 beta :

```
Dim betaVec As Variant: betaVec = Application.LinEst(yVec, x, True, False)
Dim beta As Variant
For Each beta In betaVec
    Debug.Print beta
Next beta
End Sub
```

- 為何會錯誤呢？因為在 Excel 上，一維陣列被當作一個 row vector。

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

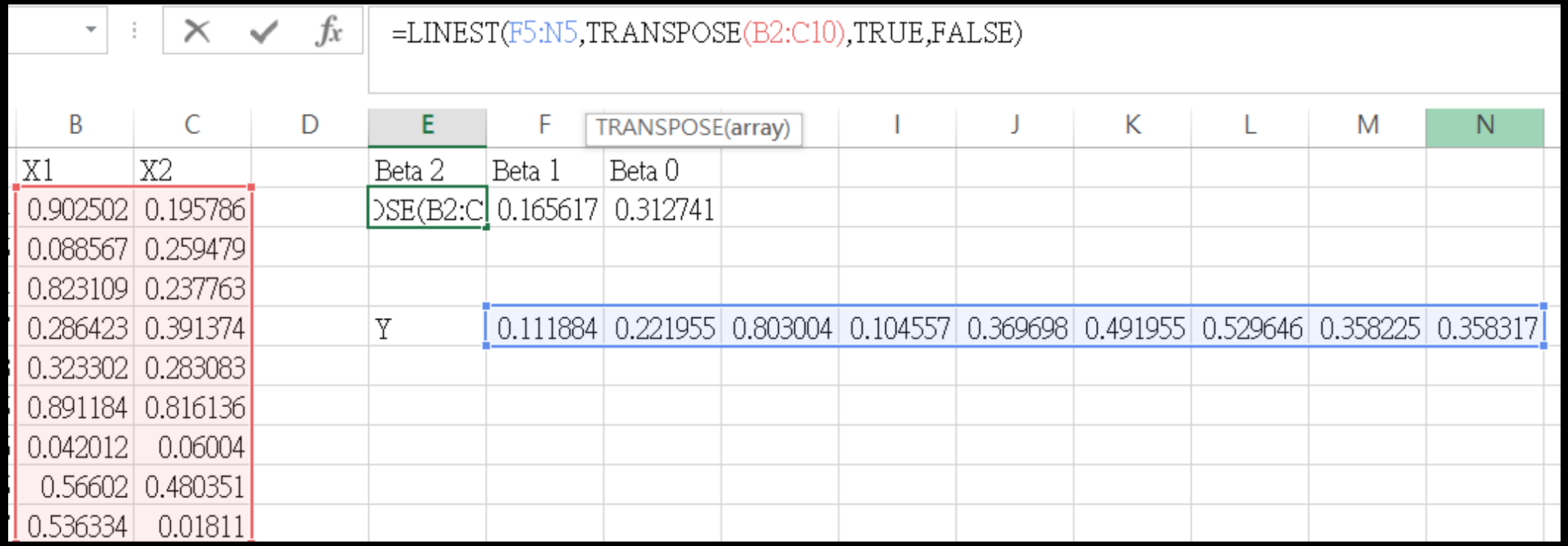
- 來看看以下的情況 :



USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 原因在於 LINEST 資料是以 Y 來對其的，當 Y 為 row vector 時，會預設 X 的每一個 row 為一組資料。
- 因此只要改變資料放置的方式即可。

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL



USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 在 VBA 中也是相同 :

```
Sub linestExample()

Dim y As Variant: y = Range("A2:A10").Value
Dim x As Variant: x = Range("B2:C10").Value
Dim i As Integer
Dim yVec(9) As Double
For i = 1 To 9
    yVec(i) = y(i, 1)
Next i

Dim betaVec As Variant: betaVec = Application.LinEst(yVec, Application.Transpose(x), True, False)
Dim beta As Variant
For Each beta In betaVec
    Debug.Print beta
Next beta
End Sub
```

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 因此接下來，我們可以寫一個 AR 函數來估計該時間序列之參數。

```
Function setARModel(inputSeries As TimeSeries, order As Integer) As Double()
Dim seriesLength As Integer: seriesLength = UBound(inputSeries.dates)
Dim nData As Integer: nData = seriesLength - order
Dim dataValues As Variant: dataValues = inputSeries.values
Dim nParam As Integer: nParam = order + 1
ReDim y(nData) As Double
ReDim x(order, nData) As Double
ReDim paramVec(order + 1) As Double
Dim regressionResult As Variant
Dim paramElem As Variant
Dim i As Integer
Dim j As Integer

For j = 1 To nData
   y(j) = dataValues(i + order)

   For i = 1 To order
      x(i, j) = dataValues(i + order - 1)
   Next i
Next j
regressionResult = Application.LinEst(y, x, True, False)

For Each paramElem In regressionResult
   paramVec(nParam) = paramElem
   nParam = nParam - 1
Next paramElem

setARModel = paramVec
End Function
```

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 接下來可以利用目前的時間序列函數做個小回測 :
  - 第一步 : 利用 fromWorksheet 輸入想要的資料轉成時間序列格式。
  - 第二步 : 利用 inInterval 函數篩選出用來估計參數的資料區間。
  - 第三步 : 利用 calculateReturnRate 函數計算每段區間的報酬率。
  - 第四步 : 利用 setARModel 估計報酬率的自我迴歸模型參數。
  - 接下來剩下的時間 :
    - 當 AR 模型預測下期為正報酬 ➔ 本期作多，到下期平倉
    - 當 AR 模型預測下期為負報酬 ➔ 本期放空，到下期平倉

USER-DEFINED TYPE : AUTOREGRESSIVE MODEL

- 試試看寫一個回測函數完成接下來的部分，並把損益變為一個時間序列

SECTION 8. 錯誤處理

Err 與 On Error Resume Next

回顧 TIMESERIES

- 但在時間序列越來越多功能越來越自由時，反之也因為使用 Range 與 Variant 造成可能會出現很多未預期的錯誤。
  - 1. dates 與 values 兩組資料長度不同
  - 2. dates 內容並非日期或是 values 內容並非 double

回顧 TIMESERIES

- 例如不小心因手誤將兩資料長度設為不同。

```
Dim adjColsedTimeSeries As TimeSeries: adjColsedTimeSeries = fromWorksheet(Range("A2", Cells(110, "A")), _
                                                                           Range("B2", Cells(100, "B")))

dataFile.Close
'找 2330<= 100 的股價
Dim findPrices As TimeSeries
findPrices = inInterval(adjColsedTimeSeries, byValue, , 100)
```

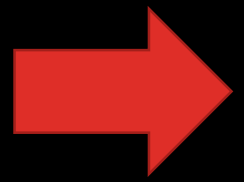
回顧 TIMESERIES

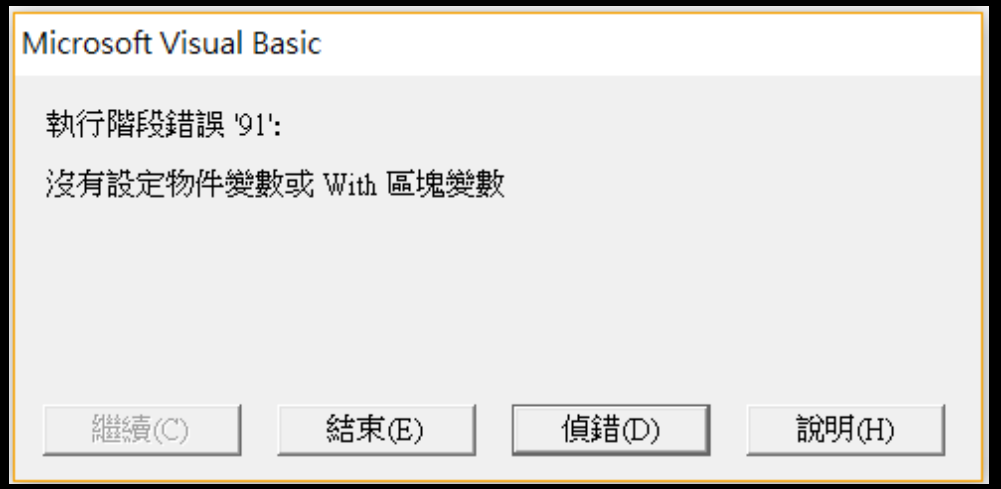
- 上述結果造成很多莫名的 0 元股價出現，因為陣列長度是以 dates 為主，dates 長度較長時會使 values 多於的地方皆為 0。
- 但由於程式在執行時並不會產生任何造成中斷的錯誤，可能會難以察覺。
- 要如何改善這個問題呢？
- 可以利用 VBA 中的 Err 物件，將某些應該是錯誤的情況定義為錯誤。

ERR

- VBA 發生錯誤時，會有類似以下系統提示。
- 可以利用 Err 建立專屬於某些情境的錯誤。

```
Sub example()

Dim x As Worksheet
x = 100

End Sub
```





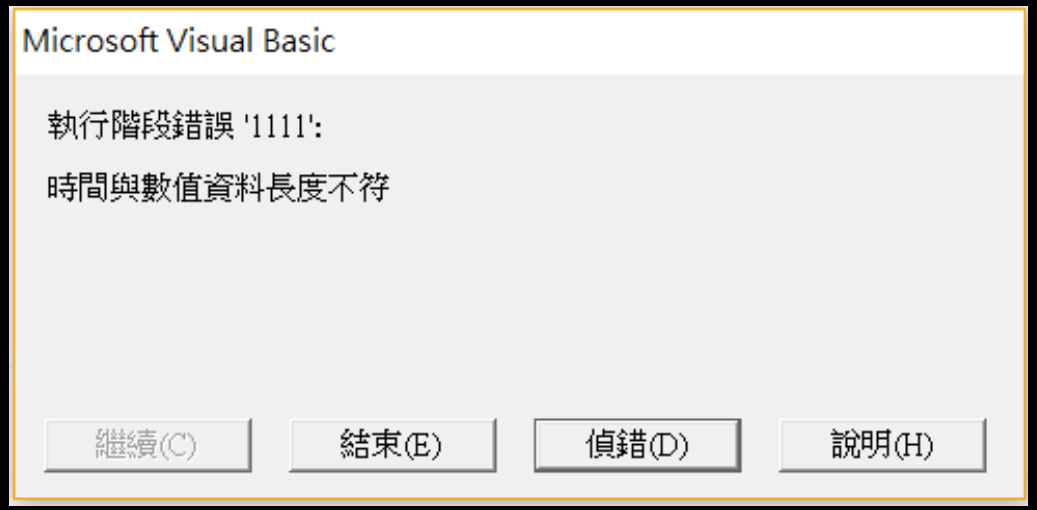
ERR

- Err.Rise 方法可以在需要的地方建立自己想要的系統提醒。
- 例如剛才 fromWorksheet 函數的問題，我們可以在當兩串儲存格長度不同時顯示系統錯誤提醒並暫停在錯誤處。

```
'輸入的函數
Function fromWorksheet(dateRange As Range, valueRange As Range) As TimeSeries

Dim nData As Integer: nData = dateRange.Count

If nData <> valueRange.Count Then
    Err.Raise Number:=1111, Description:="時間與數值資料長度不符"
End If
```

ERR

- 剛才的手誤部分就會出現以下的訊息：



ERR

- 按下「偵錯」會連到 Err.Rise 的地方：

```
If nData <> valueRange.Count Then
    Err.Raise Number:=1111, Description:="時間與數值資料長度不符"
End If
```

- 而其中的 Number 與 Description 等內容分別代表甚麼意思呢？

ERR

- Description 代表的意思非常淺顯易懂，即是錯誤時顯現的對話框內容，可用時告知詳細出錯原因。
- Number 則有更大的用處，當錯誤時我們可以將一組錯誤代碼給予 Err，不同的情況可以給予不同的代碼，而在 On Error Goto 時，我們可以依據該代碼給予不同的錯誤不同的處理，Number 必須為 Long。

ERR

- 如在上述函數中，利用 On Error GoTo 錯誤處理， 發現錯誤代碼為 1111 時，可以自動將兩者變為使用 valueRange 的長度。
- 一般在沒任何錯誤時 Number 為 0。

```
If nData <> valueRange.Count Then
    Err.Raise Number:=1111, Description:="時間與數值資料長度不符"
End If

reStart:

ReDim dates(nData) As Date
ReDim values(nData) As Double
Dim output As TimeSeries

Call rangeToArray(dates, dateRange)
Call rangeToArray(values, valueRange)

output.dates = dates
output.values = values
fromWorksheet = output
Exit Function

reRange:
If Err.Number = 1111 Then
    nData = valueRange.Count
    GoTo reStart
Else
    MsgBox "不明錯誤"
End If
```

ERR

- 注意：當函數處理完該錯誤後，錯誤已不存在，但 Err 內還是存有該錯誤的相關資訊。
- 此時可以使用 Err.Clear 方法將現存錯誤清除。
- Err.Rise 內還有許多引數，但實用性較不大，以下稍微介紹：

ERR

- HelpContext : VBA Help 文件的 ID。
- HelpFile : Help 文件之路徑。
- Source : 用來記錄出厝的模組或類別之名稱。

ON ERROR RESUME NEXT

- 在寫程式時常會要處理批次的問題，利用迴圈重複處理大量的事，但當迴圈某步突然出問題時，可能會造成前功盡棄。
- 例如，我們想利用迴圈把 data 1 ~ data 4 的資料整併成一張表。

data
0.75
0.73
0.06
0.79
0.43
0.15
0.1

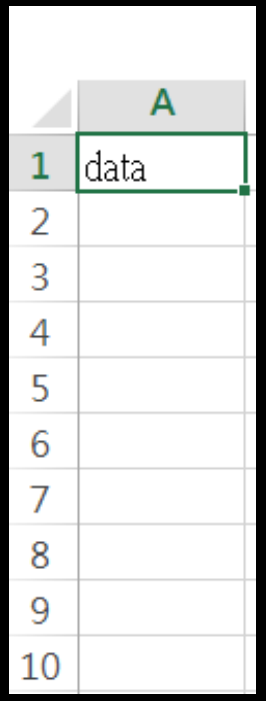
ON ERROR RESUME NEXT

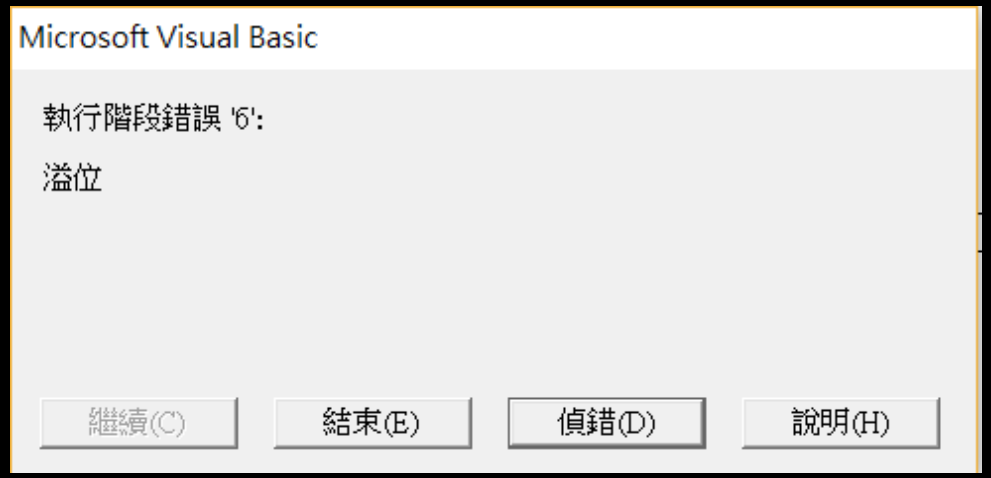
- 他們理應都是一串數值，因此我們可以利用以下程式碼來進行：

```
Sub resumeExample()

Dim collectionSheet As Worksheet: Set collectionSheet = ActiveSheet
Dim dataFile As Workbook
Dim dataPathCollection As Variant: dataPathCollection = Application.GetOpenFilename(MultiSelect:=True)
Dim i As Integer
Dim oneFilePath As Variant
Dim colCounts As Integer
Dim lastRow As Integer

For Each oneFilePath In dataPathCollection
    colCounts = colCounts + 1
    Cells(1, colCounts).Value = "Data " & colCounts
    Set dataFile = Workbooks.Open(oneFilePath)
    lastRow = dataFile.Worksheets(1).Cells(1, 1).End(xlDown).Row
    dataFile.Worksheets(1).Range("A2", Cells(lastRow, "A")).Copy Destination:=collectionSheet.Cells(2, colCounts)
    dataFile.Close
Next oneFilePath

End Sub
```

ON ERROR RESUME NEXT

- 但 data 2 檔案其實有問題，他的內容為空白的，因此 End(XlDown).Row 會回傳工作表能到達的最後一個 Row，造成溢位。





ON ERROR RESUME NEXT

- 在錯誤時我們不希望程式因此停下來，可以繼續將剩下的資料跑完避免前功盡棄。
- 此時我們可以在程式中間加入 On Error Resume Next，代表有錯誤時不中斷繼續執行下去。

```
On Error Resume Next ' 錯了就換下一個
For Each oneFilePath In dataPathCollection
    colCounts = colCounts + 1
    Cells(1, colCounts).Value = "Data " & colCounts
    Set dataFile = Workbooks.Open(oneFilePath)
    lastRow = dataFile.Worksheets(1).Cells(1, 1).End(xlDown).Row
    dataFile.Worksheets(1).Range("A2", Cells(lastRow, "A")).Copy Destination:=collectionSheet.Cells(2, colCounts)
    dataFile.Close
Next oneFilePath
```

ON ERROR RESUME NEXT

- 忽略掉該錯誤即可完成：

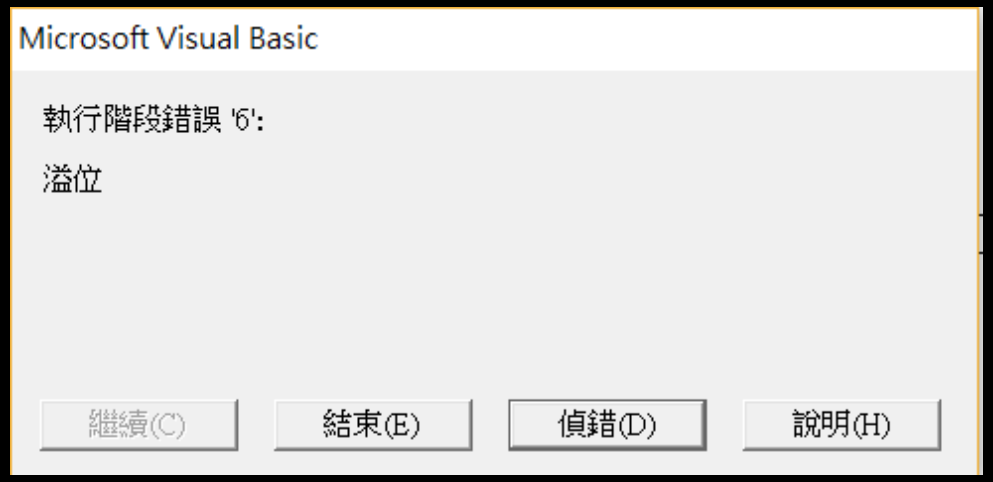
	A	B	C	D
1	Data 1	Data 2	Data 3	Data 4
2	0.75		"12"	0.75
3	0.73		"15"	0.73
4	0.06		"7"	0.06
5	0.79		"13"	0.79
6	0.43		"8"	0.43
7	0.15		"2"	0.15
8	0.1		"5"	0.1

與 ERR 合併

- 雖然可以完成，但是這是在忽略錯誤的情況下，我們其實想知道有那些地方有問題，之後可以在對該處進行確認與修正。
- 會希望能紀錄出錯的地方與問題，此時 Err 就派上用場了。

與 ERR 合併

- 在 data 2 出錯時，我們可以看到 :



- 代表裡面其實有紀錄 Number 為 6，可能資料太多或根本沒資料 。

```
On Error Resume Next ' 錯了就換下一個
Dim errorMessageSheet As Worksheet: Set errorMessageSheet = Worksheets("Error")
Dim errorCounts As Integer
Dim errorMsg As String
Dim errNum As Long

For Each oneFilePath In dataPathCollection
    colCounts = colCounts + 1
    collectionSheet.Cells(1, colCounts).Value = "Data " & colCounts
    Set dataFile = Workbooks.Open(oneFilePath)
    lastRow = dataFile.Worksheets(1).Cells(1, 1).End(xlDown).Row
     '有錯誤就紀錄
    If Err.Number <> 0 Then
        errNum = Err.Number
        errorCounts = errorCounts + 1
        errorMessageSheet.Cells(errorCounts, 1).Value = "Data " & colCounts

        Select Case errNum
            Case 6
                errorMsg = "資料數量過多或無資料"
            Case Else
                errorMsg = "未知錯誤"
        End Select
        errorMessageSheet.Cells(errorCounts, 2).Value = errorMsg
    End If
```

ON ERROR GOTO 與 ERR 合併

- 因此在出現代號為 6 時其實代表資料過多或根本沒資料。
- 注意 : On Error Resume Next 有一特點讓紀錄錯誤較為困難：
- 當錯誤發生後，只要在之後任何一次的變數輸出或輸入成功，則 Err 自動歸 0。
- 因此必須在可能錯誤的下一行即進行紀錄，但錯誤可能不只一個地方，如何全盤紀錄？
- 把有先共用的變數便成員變數，再把迴圈內容拆成另一個子程序。
- 子程序內再利用 On Error GoTo 紀錄。

ON ERROR GOTO 與 ERR 合併

```
For Each oneFilePath In dataPathCollection
    colCounts = colCounts + 1
    Call inputNextData(oneFilePath, colCounts)
Next oneFilePath

End Sub
```

```
Private Sub inputNextData(filePath As Variant, inputCol As Integer)
' 在此處加入 On Error GoTo 吧!!
Dim lastRow As Integer

collectionSheet.Cells(1, inputCol).Value = "Data " & inputCol
Set dataFile = Workbooks.Open(filePath)
lastRow = dataFile.Worksheets(1).Cells(1, 1).End(xlDown).Row
dataFile.Worksheets(1).Range("A2", Cells(lastRow, "A")).Copy Destination:=collectionSheet.Cells(2, inputCol)
dataFile.Close
End Sub
```

ON ERROR GOTO 與 ERR 合併

- 練習想看看要如何修改？新增一個可能的問題：我們希望所有資料是數字，但 data 3 就明顯內容是文字，出現該問題時怎麼處理？
- 這個問題可以試試看用 Excel 的 Value 函數，嘗試將字串轉為數值，但對於無法轉換的呢？
- 另外可能有一個問題：輸入的檔案如果是隨便一個圖檔，照樣可以讀取，但結果出來會是亂碼，怎麼辦？
- 可以試試看用 Split 檢查副檔名。