EXCEL VBA 進階班

SECTION 2. 進階變數型態介紹 (1)

Variant 陣列 與 Collection Variant 詳述

VARIANT 陣列

- 上次提到了 ParamArray，他為一個 Variant 陣列，裡面每個元素可以放入不同的變數型態（甚至是陣列）。
- 假設我們有一個投資組合如下：

股票名稱	投資金額
日月光	20,000
皇翔	50,000
聯電	30,000

VARIANT 陣列

- 我們想要把投資組合各個股票分開拆成陣列，以利後續回測。
- 每個陣列裡包含以下資訊：
  - 過去某日投資現值
  - 現值日期
  - 歷史價格
  - 歷史價格對應之時間

VARIANT 陣列

VARIANT 陣列

- 我們擁有的資料如下：

G11

	A	B	C	D
1	Date	日月光	皇翔	聯電
2	2010/1/4	17.19398	43.3268	13.2363
3	2010/1/5	17.72397	44.77542	13.3921
4	2010/1/6	17.87114	44.38034	14.3264
5	2010/1/7	18.60723	43.19511	14.3264
6	2010/1/8	18.60723	43.52434	14.1707
7	2010/1/11	18.43059	42.4708	14.4821
8	2010/1/12	18.37166	42.27326	14.2875
9	2010/1/13	17.84175	41.21972	13.8982
10	2010/1/14	18.07724	41.21972	13.976
11	2010/1/15	17.87114	41.48311	14.0149
12	2010/1/18	17.57672	42.27326	13.8982
13	2010/1/19	17.51787	41.08803	13.6646
14	2010/1/20	17.01735	40.49541	14.0928

	A	B	C	D
1	日期	2010/1/4		
2				
3				
4	名稱	日月光	皇翔	聯電
5	金額	20000	50000	30000

VARIANT 陣列

- 把個別股票分作 Variant 陣列，以單一股票來看，可以用以下的方法：

```
Option Explicit
Option Base 1

Sub dataToArray()
'陣列長度為四，放入四組資料
Dim tw2311(4) As Variant
'取得歷史資料長度
Dim lastRow As Integer: lastRow = Worksheets("投資組合現值").Cells(1, 1).End(xlDown).Row

'先進入現值工作表，輸入資料
Worksheets("投資組合現值").Activate

tw2311(1) = Cells(2, "A").Value '第一個元素放入現值日期
tw2311(2) = Cells(5, 2).Value '第二個元素放入現值

'進入歷史資料工作表，取得時間序列
Worksheets("投資組合現值").Activate

tw2311(3) = Range(Cells(2, 1), Cells(lastRow, 1)).Value '第三個元素放入歷史日期
tw2311(4) = Range(Cells(2, 2), Cells(lastRow, 2)).Value '第四個元素放入歷史股價

End Sub
```

VARIANT 陣列

- 為了方便觀察該股票資料，我們可以一個在即時運算視窗呈現陣列內容的子程序 。
- printPresentValue 子程序用來顯示現值 :

```
Sub printPresentValue(data() As Variant)
Debug.Print "淨值日期: " & data(1) & Chr(10) & "淨值: " & data(2) & Chr(10)
End Sub
```

VARIANT 陣列

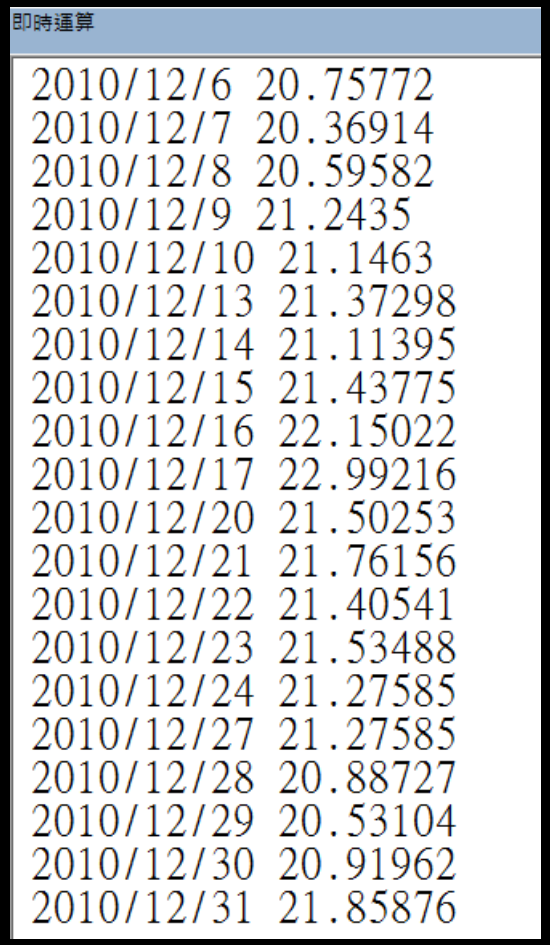
- 另外使用 printHistoricalData 子程序來列印歷史資料。
- 值得一提的是，日期與價格是在陣列中的陣列，那我們該如何取得單獨一個日期或價格呢？
- data(3) 可以取得日期陣列，由於是直接將儲存格的值轉為陣列，因此為一個二維陣列，第一個日期為 (1,1)。
- 因此可以以此類推 data(3)(1,1) 可以得到第一個日期。

VARIANT 陣列

```
Sub printHistoricalData(data() As Variant)

Dim nData As Integer: nData = UBound(data(3)) '取得歷史資料數量
Dim priceTable As String, _
    i As Integer

priceTable = "日期        " & "價格" & Chr(10)            '歷史資料表格表頭

For i = 1 To nData
    priceTable = priceTable & data(3)(i, 1) & " " & data(4)(i, 1) & Chr(10)
Next i

Debug.Print priceTable

End Sub
```

VARIANT 陣列

- 列印結果如下：



VARIANT 陣列

- 注意：
  - 及時運算視窗列印有長度限制，因此前半段的會被削掉。
  - 由於日期字數不同，會造成沒有對齊。
- 要如何改善？

VARIANT 陣列

- 練習 : 建立一個名為 portfolio 的 Variant 陣列，裡面包含各個單獨標的的 Variant 陣列。

COLLECTION

- 利用 Variant 陣列，我們可以建構一個每個元素都不同的陣列。
- 但 Variant 陣列有個缺點，我們利用順序來記各個元素的內容，如果某個地方輸入錯誤標籤並不容易察覺。
- 如果能將每個元素給予相應的名稱，對於維護上會較為方便。

COLLECTION

- Variant 陣列 :

COLLECTION

- 希望改為：

COLLECTION

- 針對這方面的需求，VBA 提供另一種資料型態，名為 Collection。
- Collection 與 Variant 陣列一樣，允許每個元素為不同資料型態。
- 除此之外，Collection 允許給每個元素一個字串作為名稱，在呼叫元素上會比陣列更加方便。

COLLECTION：建立

- Collection 在使用上並不用一開始宣告長度，但是它是一個物件，記得要加上 Set。

```
Option Explicit
Option Base 1

Sub dataToCollection()

Dim tw2311 As Collection: Set tw2311 = New Collection
'或是 Dim tw2311 As New Collection

End Sub
```

COLLECTION：新增

- 新增內容物的方式比較接近，Worksheets 與 Charts，使用 Add 加入新的內容物：

```
'先進入現值工作表，輸入資料
Worksheets("投資組合現值").Activate

'Item ： 內容物
'Key ： 標籤名稱
tw2311.Add Item:=Cells(1, "B").Value, Key:="presentValueDate"
```

COLLECTION：新增

- Item 為想要輸入之資料，Key 為該資料所屬之標籤。
- Item 預設為增加在上一個標籤之後，取值方式有兩種：
  - 使用數字順序之標籤
  - 使用命名的 Key

COLLECTION：新增

- 取值時預設起始標籤與 Option Base 相同。

```
'Item : 內容物
'Key : 標籤名稱
tw2311.Add Item:=Cells(1, "B").Value, Key:="presentValueDate"

'取第一個標籤的值查看
MsgBox tw2311(1)
```

COLLECTION：新增

- 使用 Key 只要將標籤用名稱代替就好：

```
'Item ： 內容物
'Key ： 標籤名稱
tw2311.Add Item:=Cells(1, "B").Value, Key:="presentValueDate"

'取第一個標籤的值查看
MsgBox tw2311("presentValueDate")
```

- 使用它整體程式的可讀性大幅增加了

COLLECTION：新增

- Collection 在新增元素時可以不給 Key。

```
'先進入現值工作表，輸入資料
Worksheets("投資組合現值").Activate

'Item ： 內容物
'Key ： 標籤名稱
tw2311.Add Item:=Cells(1, "B").Value
```

- 注意：Collection 各個元素的 Key 一旦決定（包含不給）就無法更改，只能移除並重新新增，稍後會提到更多關於新增與移除的功能。

COLLECTION：新增

- Collection 新增不一定要在最後面，與 Charts 和 Worksheets 一樣，利用 Before 與 After 可以決定新增在某一個元素的前後。

```
'先進入現值工作表，輸入資料
Worksheets("投資組合現值").Activate

'Item : 內容物
'Key : 標籤名稱
tw2311.Add Item:=Cells(1, "B").Value, Key:="presentValueDate"

'新增在 presentValueDate 前面
tw2311.Add Item:=Cells(5, "B").Value, Key:="presentValue", _
           Before:="presentValueDate"

MsgBox tw2311(1)
```

COLLECTION : 移除

- 利用 Remove 可以移除特定元素。

```
'Item : 內容物
'Key : 標籤名稱
tw2311.Add Item:=Cells(1, "B").Value, Key:="presentValueDate"

'新增在 presentValueDate 前面
tw2311.Add Item:=Cells(5, "B").Value, Key:="presentValue", _
           Before:=1

tw2311.Remove (1)
MsgBox tw2311(1)
```

COLLECTION

- 同 Variant 陣列，Collection 的元素可以是任何東西，包含物件，亦即可以是一個 Collection :

```
'先進入現值工作表，輸入資料
Worksheets("投資組合現值").Activate

'Item : 內容物
'Key : 標籤名稱
tw2311.Add Item:=Cells(1, "B").Value, Key:="presentValueDate"

'新增在 presentValueDate 前面
tw2311.Add Item:=Cells(5, "B").Value, Key:="presentValue", _
           Before:=1

Dim portdolio As New Collection
portdolio.Add Item:=tw2311, Key:="TW2311"
```

COLLECTION

- 練習：嘗試先前面 Variant 陣列的例子換成 Collection。

COLLECTION：取值

- Collection 的取值與陣列相似，不同之處在於可以將標籤換成 Key。

```
Sub collectionExample()

Dim personalInfo As New Collection

personalInfo.Add Item:="劉自強", Key:="name"
personalInfo.Add Item:=50, Key:="age"

MsgBox "姓名: " & personalInfo("name") ' 可以使用 Key
MsgBox "年齡: " & personalInfo(2)          ' 可以使用標籤順序


End Sub
```

COLLECTION：輸入

- 但假如我們要把年齡改成 51 會出現甚麼問題呢？

```
Sub collectionExample()

Dim personalInfo As New Collection

personalInfo.Add Item:="劉自強", Key:="name"
personalInfo.Add Item:=50, Key:="age"

personalInfo("age") = 51  '可以嗎？
End Sub
```

COLLECTION：輸入

- Collection 的內容物輸入之後即無法更改！!
- 只能透過 Remove，並重新在相同位置相同 Key 新增物件來更動。
- 與 Variant 陣列相較，是一件不具效率的事。
- 試試看自製一函數處理掰問題。

COLLECTION：確認是否含有某標籤

- 如同陣列的維度，VBA 本身並沒有一個函數可以來確認 Collection 是否存在某一名稱標籤。
- 但同樣利用 On Error GoTo，可以解決此問題，大家可以嘗試寫出該函數。

COLLECTION : FOR EACH

- Collection 也可以同陣列與 Range 一樣使用 For Each 的遞迴方式。
- 每個元素迴圈時一樣使用 Variant 變數。

SECTION 5. FUNCTIONAL PROGRAMMING (1)

Run

FUNCTIONAL PROGRAMMING

- VBA 為傳統的指令式程式語言（imperative language），是一般人較習慣的寫程式方式。
- 但除了指令式程式語言之外，另有一種程式語言被稱作函數式程式語言（functional language）。
- 近年來越來越多傳統的指令式程式語言中間加入許多函數式語言的特性，使程式在撰寫上更靈活方便。
- VBA 兩個自 2013 版新增的功能 : Run 與 Evaluate，使 VBA 也能做一些類似於函數式語言的功能。

FUNCTIONAL PROGRAMMING

- 函數式語言特性 1：
  - 函數可以是另一個函數的引數。
  - 函數也可以回傳一個函數，而且是依據引數創造出的新函數，不是事先定義好之函數。
- 對於 VBA 來說：
  - 將函數可以是引數 ➔ 2013 有了 Application.Run 之後該問題就可以用近似的方法解決。
  - 函數回傳函數 ➔ 參考 C++ 的作法，可以用物件導向做出近似的結果。（未來會提到）

RUN

- Run 的引數如下：
- Run(*Macro*, *Arg1*, *Arg2*, *Arg3*, *Arg4*, *Arg5*, *Arg6*, *Arg7*, *Arg8*, *Arg9*, *Arg10*, *Arg11*, *Arg12*, *Arg13*, *Arg14*, *Arg15*, *Arg16*,*Arg17*, *Arg18*, *Arg19*, *Arg20*, *Arg21*, *Arg22*, *Arg23*, *Arg24*, *Arg25*, *Arg26*, *Arg27*, *Arg28*, *Arg29*, *Arg30*)
- *Macro*：一個 VBA 子程序或函數之名稱字串。
- *Arg*：該子程序或函數的引數，最多可以允許有 30 個。

RUN

- 例如我們想要用一函數來計算我們所想要的移動平均，可以自行選擇需要簡單移動平均或加權移動平均

```
Option Explicit
Option Base 1

Function simpleMovingAverage(stockPrices As Variant) As Variant

simpleMovingAverage = appliction.Average(stockPrices)

End Function

Function weightedMovingAverage(stockPrices As Variant) As Variant

Dim dominator As Integer
Dim counts As Integer
Dim elem As Variant

For Each elem In stockPrices
    counts = counts + 1
    dominator = dominator + counts
    weightedMovingAverage = weightedMovingAverage + elem * counts
Next elem

weightedMovingAverage = weightedMovingAverage / dominator

End Function
```

RUN

- 原來想到的作法為：

```
Function movingAverage(stockPrices As Variant, method As String) As Variant

If method = "simpleMovingAverage" Then
    movingAverage = simpleMovingAverage(stockPrices)
ElseIf method = "weightedMovingAverage" Then
    movingAverage = weightedMovingAverage(stockPrices)
End If

End Function
```

RUN

- 但這樣修正有一個缺點，當種類越來越多的時候，我的 if else 要一直延伸下去，如果不小心忘記修改則無法用該方法計算。
- 因此可以改使用 Application.Run。

```
Function movingAverage(stockPrices As Variant, method As String) As Variant
movingAverage = Application.Run(method, stockPrices)
End Function
```

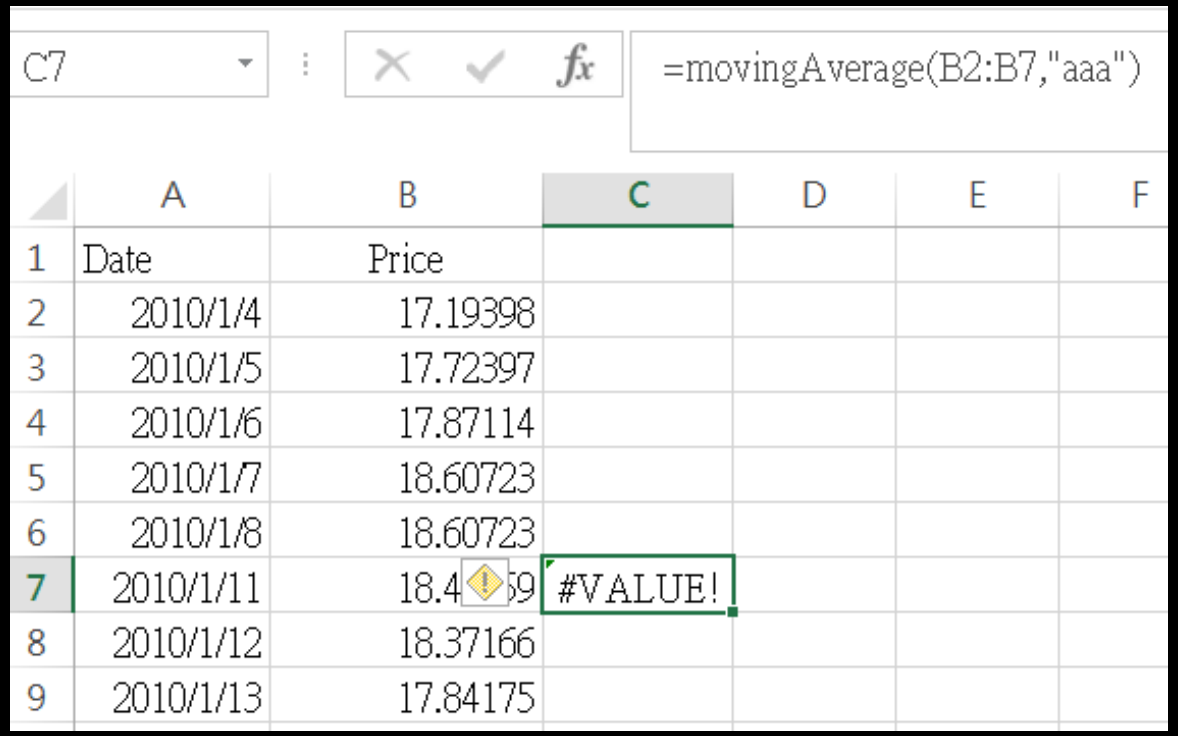
RUN

- 用 Run 的情況下，movingAverage 函數變得像是一個介面，當把資料（stockPrices）與想執行的函數（method）給其後，他便會執行：

$$movingAverage = method(stockPrice)$$

- 但假如我們輸入一個錯誤的函數名稱會如何呢？

RUN


C7
=movingAverage(B2:B7,"aaa")
A
B
C
D
E
F
1 Date Price
2 2010/1/4 17.19398
3 2010/1/5 17.72397
4 2010/1/6 17.87114
5 2010/1/7 18.60723
6 2010/1/8 18.60723
7 2010/1/11 18.4 9 #VALUE!
8 2010/1/12 18.37166
9 2010/1/13 17.84175

RUN

- 一直使用均線當例子，可能過於簡單。
- 接下來來介紹 Run 真正實用的地方：
  - 資料插補
  - 最適化問題

RUN：資料插補

- 資料插補不論是在機器學習或財務工程上都會常用碰到的問題，譬如某日市場上的 USD LIBOR 如下：

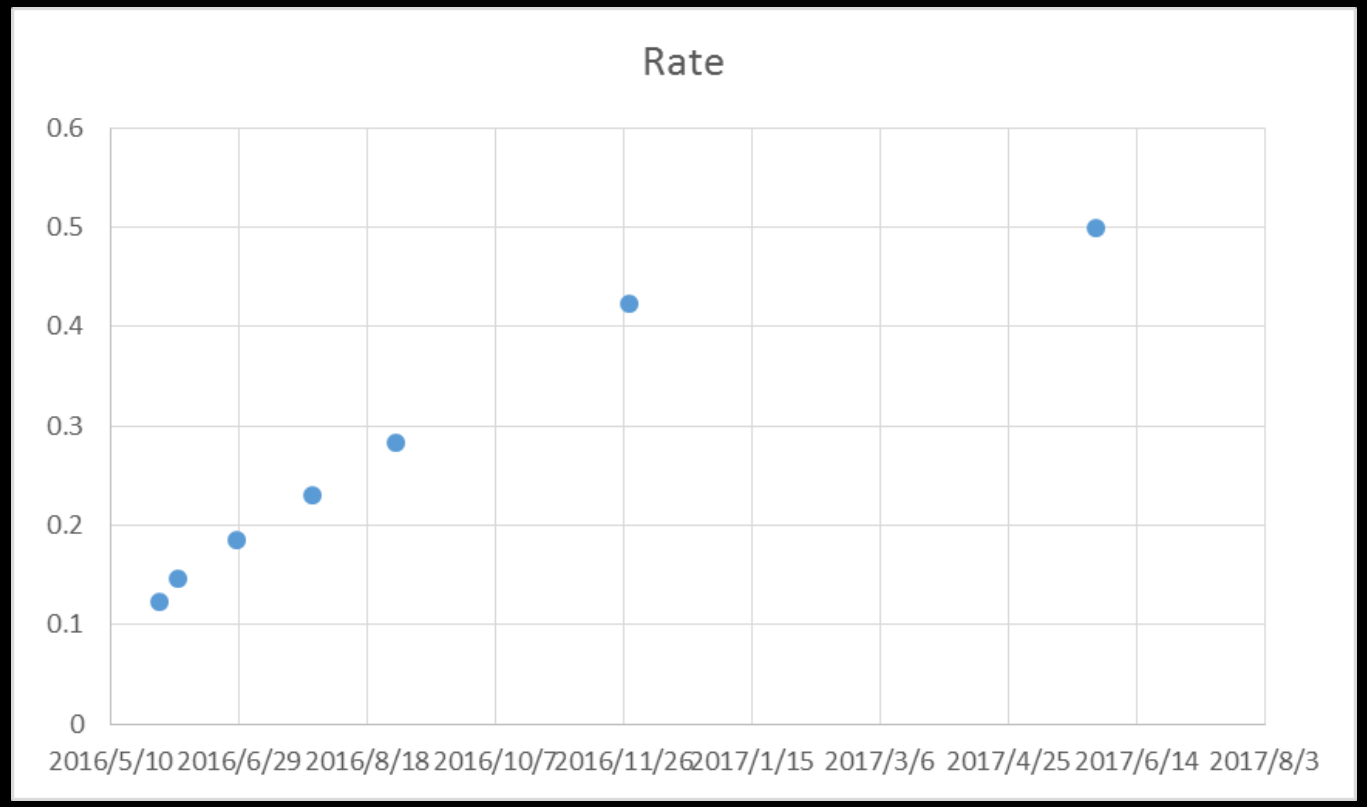
	A	B	C	D	E
1	Date	Rate		Value Date	2016/5/28
2	2016/5/29	0.123			
3	2016/6/5	0.14575			
4	2016/6/28	0.18535			
5	2016/7/28	0.23065			
6	2016/8/29	0.2835			
7	2016/11/28	0.42305			
8	2017/5/29	0.499			

RUN：資料插補

- 假如我們想要知道到期日為 2016/9/30 的 LIBOR rate 應該為多少，那應該怎麼看呢？
- 下面介紹最簡單的兩種插補方法：
  - Nearest-Neighbor
  - Linear Interpolation

RUN：資料插補

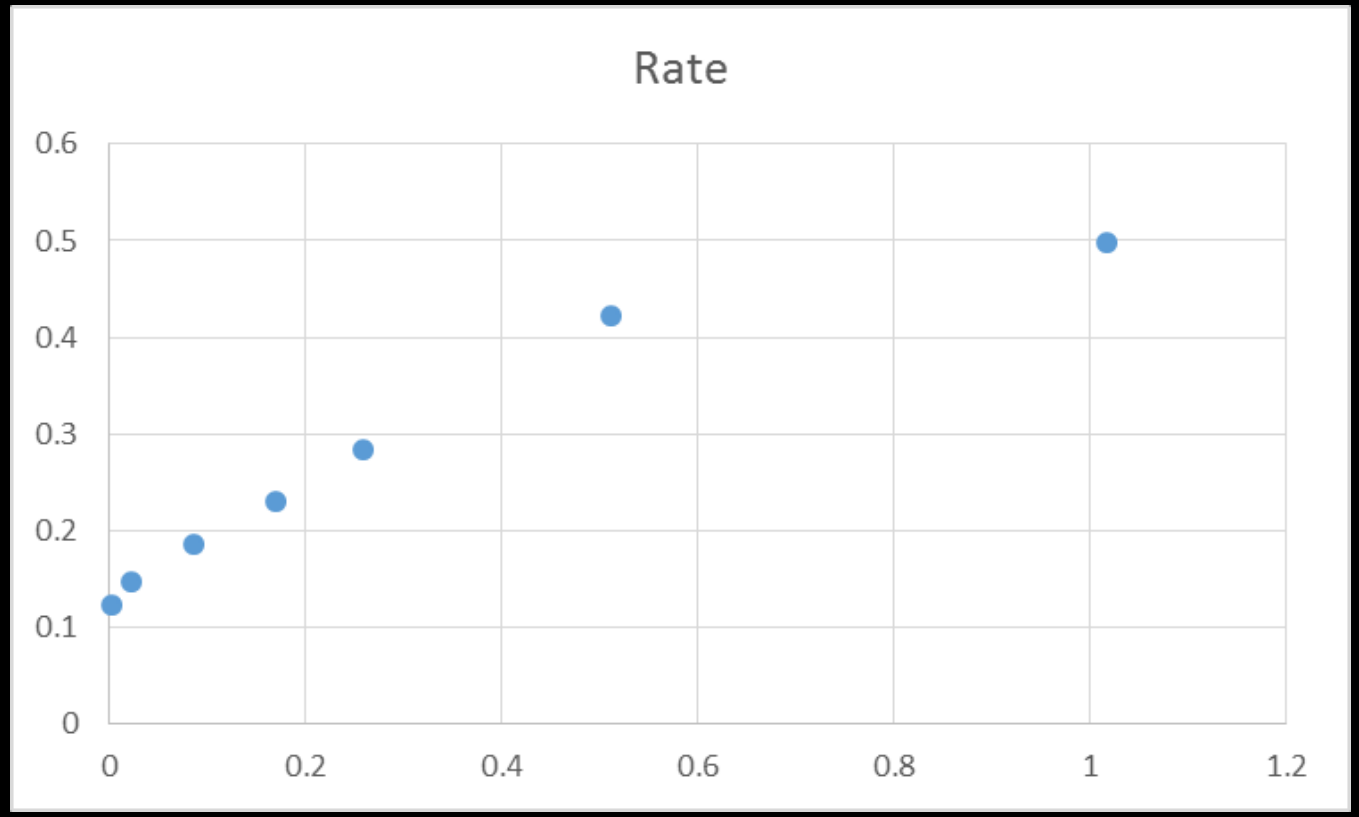
- 此為二維的內插，我們將天期當作 X 軸，利率做為 Y 軸，如下圖所示：



RUN：資料插補

- 通常並非使用日期當作 X 軸，而是使用年化時間，以 USD LIBOR 來說，天期的計算是使用 ACT/360，例如交易日為 2011/03/05，交割日為 2011/03/12，則
- 年化時間為 = (兩個日期相差的天數) / 360 = 7/360

RUN：資料插補



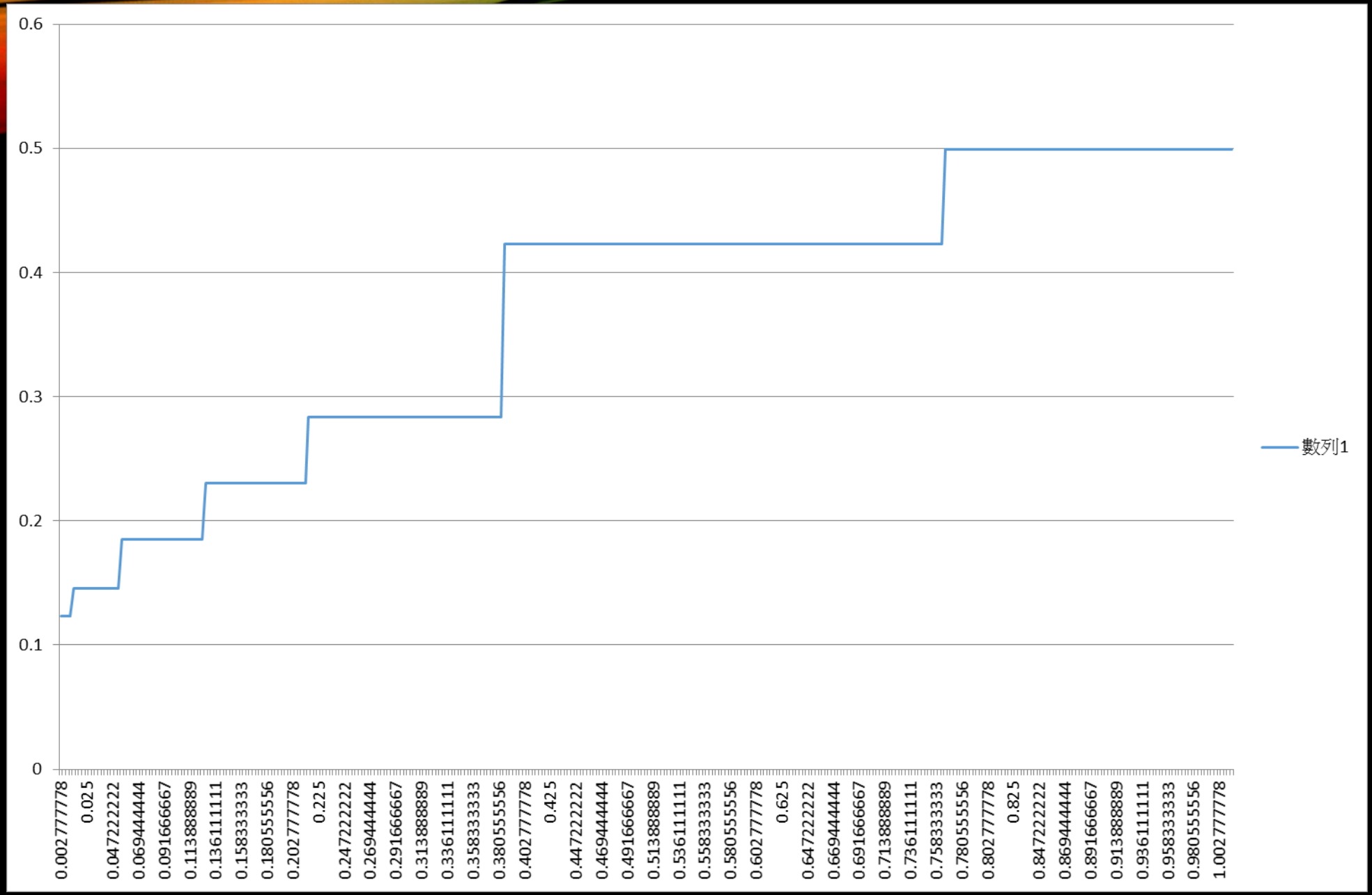
RUN：資料插補 NEAREST NEIGHBOR

- Nearest Neighbor 為最簡單的插補方法，給定某一個利率為未知之日期，其利率為日期最接近之已知利率。
- 以上例來說，2015/6/30 的內插結果為最接近之以之資料，亦即 6/28 之利率 0.18535。

	A	B	C	D	E
1	Date	Rate		Value Date	2016/5/28
2	2016/5/29	0.123			
3	2016/6/5	0.14575			
4	2016/6/28	0.18535			
5	2016/7/28	0.23065			
6	2016/8/29	0.2835			
7	2016/11/28	0.42305			
8	2017/5/29	0.499			

RUN：資料插補 NEAREST NEIGHBOR

- Nearest Neighbor 之演算法如下：
  - 1. 檢查欲搜尋之 Y 值的 X 座標，$\hat{X}$，是否坐落於所有天期之向量中
    - 1-2. 如果比最小的天期年化時間還小，則回傳最第一個利率。
    - 1-3. 反之比最大的天期還大，則回傳最後一個利率。
  - 2. 當作落在其中時，找出第一個小於等於 $\hat{X}$ 之 X 座標 X_1 與下一個 X 座標 (亦即第一個大於等於的 X 座標)
    - 2-1. 如 $|\hat{X} - X_1| \leq |\hat{X} - X_2|$，代表 $\hat{X}$ 比較靠近 X_1，回傳 X_1 之 Y 座標。
    - 2-2. 反之回傳 X_2 之 Y 座標。


0.6
0.5
0.4
0.3
0.2
0.1
0
0.002777778
0.025
0.047222222
0.069444444
0.091666667
0.113888889
0.136111111
0.158333333
0.180555556
0.202777778
0.225
0.247222222
0.269444444
0.291666667
0.313888889
0.336111111
0.358333333
0.380555556
0.402777778
0.425
0.447222222
0.469444444
0.491666667
0.513888889
0.536111111
0.558333333
0.580555556
0.602777778
0.625
0.647222222
0.669444444
0.691666667
0.713888889
0.736111111
0.758333333
0.780555556
0.802777778
0.825
0.847222222
0.869444444
0.891666667
0.913888889
0.936111111
0.958333333
0.980555556
1.002777778
數列1

RUN：資料插補 LINEAR INTERPOLATION

- 我們想知道 $\hat{t}$ 時間點之利率 $\hat{r}$，$\hat{t} \in (t_{i-1}, t_i), i = 1,2, \dots, n$。
- 假設 r_{i-1}、$\hat{r}$、r_i 三點共線，則 r_{i-1} 與 r_i 連成的線之斜率應與 r_{i-1} 及 $\hat{r}$ 連成的線之斜率相同，亦即：

$$\frac{\hat{r} - r_{i-1}}{\hat{t} - t_{i-1}} = \frac{r_i - r_{i-1}}{t_i - t_{i-1}}$$

RUN：資料插補 LINEAR INTERPOLATION

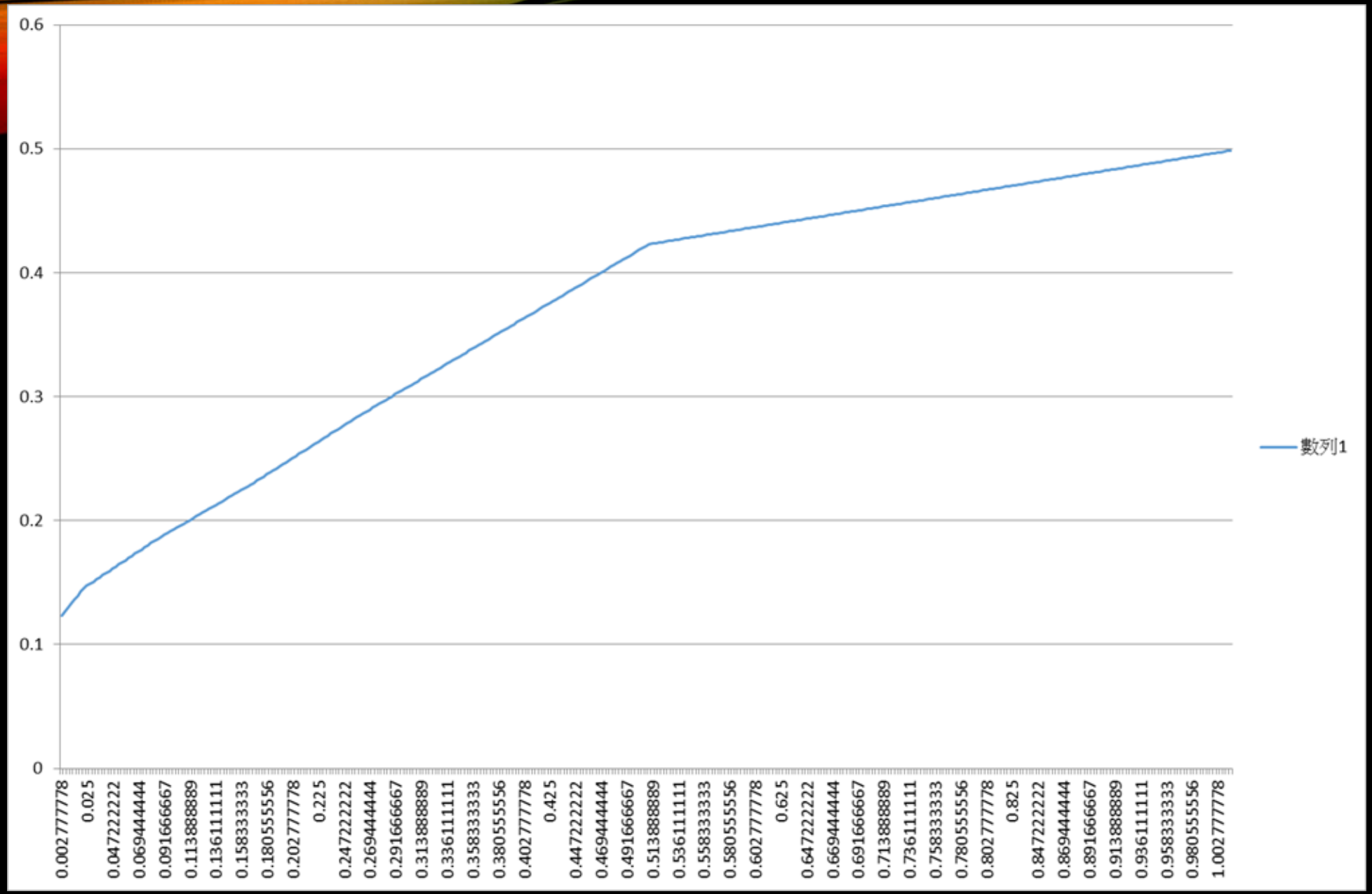
- 移項後可得：

$$\hat{r} = r_{i-1} + (r_i - r_{i-1})\left(\frac{\hat{t} - t_{i-1}}{t_i - t_{i-1}}\right)$$

- 此即為線性內插。

RUN：資料插補
LINEAR INTERPOLATION

- 上述兩種內插應用上極為簡單，但建構出之利率曲線不為平滑函數。
- 如 Schlogl（2013）書中提到，線性內插會使得遠期利率的樣式不利模型應用。


0.6
0.5
0.4
0.3
0.2
0.1
0
0.002777778
0.025
0.047222222
0.069444444
0.091666667
0.113888889
0.136111111
0.158333333
0.180555556
0.202777778
0.225
0.247222222
0.269444444
0.291666667
0.313888889
0.336111111
0.358333333
0.380555556
0.402777778
0.425
0.447222222
0.469444444
0.491666667
0.513888889
0.536111111
0.558333333
0.580555556
0.602777778
0.625
0.647222222
0.669444444
0.691666667
0.713888889
0.736111111
0.758333333
0.780555556
0.802777778
0.825
0.847222222
0.869444444
0.891666667
0.913888889
0.936111111
0.958333333
0.980555556
1.002777778
数列1

RUN：資料插補，練習

- 上述介紹的兩個方法只能在 VBA 下用 double 陣列使用，想辦法將其改成 Excel 可使用之函數，並可利用 Run 選擇方法與陣列運算。

RUN：插補應用在無母數機器學習

- 內插可以用來做簡單的無母數機器學習。
- 假設五日開盤價移動平均 (可以選擇使用加權移動平均或簡單移動均) 為 X 軸座標 (今日與前四日)。
- Y 軸座標為今日之漲跌 (收盤價 – 開盤價)。

RUN：插補應用在無母數機器學習

- 可以依據歷史資料不斷累積線段上的點並進行內插，並建構以下策略：
  - 內插結果 > 0 ➔ 本日會漲 ➔ 開盤融資買進，收盤平倉。
  - 內插結果 < 0 ➔ 本日會跌 ➔ 融券賣出，收盤平倉。
- 試試看各種均線與內插結果。

RUN

- 從前面幾個例子讓我們看到 Run 可以依據給予的函數名稱不同，讓我們切換某個函中某部計算所使用的函數。
- 該方法使得我們在函數的使用上能更加簡便更有彈性。
- 另外一個重要的例子為最適化問題。

RUN：最適化

- 最適化為財務模型校準上常見的問題。
  - 給定一個函數 $f(x)$。
  - 以及某些對其引數 x 的限制範圍或要求。
  - 嘗試去尋找滿足該限制之函數極小值。
- 例如利用最大概似估計法估計某一分配的參數，即是一個最適化問題。

RUN：最適化

- Excel 本身提供了三種最適化方法，但除此之外如果我們想運用其他方法呢？
- 如果我們想在 VBA 中使用而不透過儲存格呢？
- 可以利用 Run 達到想要的效果。

RUN：最適化

- GRG 所找到的解為區域極值，不保證為全域極值。
- 演化法速度表現不佳。
- 如想要另一個找全域極值之方法呢？
- 以下簡介一個著名的最適化方法：Nedler-Mead method。
- 如有人用過 MATLAB 之 findminsearch 即是使用該方法。

RUN : NEDLER-MEAD 演算法

- 假設 $f: \mathbb{R}^n \to \mathbb{R}$，我們的目標為找出一個向量 $\hat{x}$，使得 $f(\hat{x})$ 為該函數之最小值。
- step 0. 產生 $n+1$ 組 $\mathbb{R}^n$ 向量，$x_0, x_1, \ldots, x_n$ 作為初值，$f(x_0) \le f(x_1) \le \cdots \le f(x_n)$。
- step 1. 計算前 n 組 $\mathbb{R}^n$ 向量之平均，$\bar{x} = (x_0, x_1, \ldots, x_{n-1})/n$。
- step 2. 計算 reflection point，$x_r = \bar{x} + \rho(\bar{x} - x_n)$
  - 2-1. 如 $f(x_1) \le f(x_r) \le f(x_n)$，則 x_r 取代 x_n，並將 $f(x)$ 排序，回到 step 1. ，開始下次迭代。
  - 反之不滿足 2-1. ，則進入 step 3.

RUN : NEDLER-MEAD 演算法

- step 3. 如果 $f(x_r) \leq f(x_0)$，計算 extension point，$x_e = \bar{x} + \chi(x_r - \bar{x})$
  - 3-1. 如 $f(x_e) \leq f(x_r)$，則 x_e 取代 x_n，並將 $f(x)$ 排序，回到 step 1.，開始下次迭代。
  - 3-2. 如 $f(x_e) > f(x_r)$ ，則 x_r 取代 x_n，並將 $f(x)$ 排序，回到 step 1. ，開始下次迭代。
  - 3-3. 如 $f(x_r) > f(x_0)$，則進入 step 4.。
- step 4. 計算 outside contraction point，$x_{oc} = \bar{x} + \gamma(x_r - \bar{x})$
  - 4-1. 如 $f(x_{n-1}) \leq f(x_r) < f(x_n)$，且 $f(x_{oc}) < f(x_n)$ 則 x_{oc} 取代 x_n，並將 $f(x)$ 排序，回到 step 1. ，開始下次迭代。
  - 4-2. 反之不滿足 4-1.，進入 step 5.。

RUN : NEDLER-MEAD 演算法

- step 5. 計算 inside contraction point，$x_{ic} = \bar{x} + \gamma(x_n - \bar{x})$
  - 5-1. 如 $f(x_n) \leq f(x_r)$，且 $f(x_{ic}) < f(x_n)$ 則 x_{ic} 取代 x_n，並將 $f(x)$ 排序，回到 step 1.，開始下次迭代。
  - 5-2. 反之不滿足 4-1.，進入 step 6.。
- step6. 將 x_i 替換為 $x_i + \sigma(x_0 - x_i)$，對 $i = 1,2,\ldots,n$，並回到 step 1.。

RUN : NEDLER-MEAD 演算法

- 演算法中出現許多須自行設定的常數，一般來說 :
  - $\rho = 1$
  - $\chi = 2$
  - $\gamma = 1/2$
  - $\sigma = 1/2$

RUN : NEDLER-MEAD 演算法

- 演算法不斷迭代直到滿足下列兩者之一：
  - 達到迭代上限次數。
  - $|f(x_0) - f(x_n)| < \varepsilon$，$\varepsilon$ 為一個誤差之容忍範圍。

RUN : NEDLER-MEAD 演算法

- 測試 Nedler-Mead 函數之結果。
- 如要加入限制式，可參考下面文章 :
- http://www.emse.fr/~leriche/GBNM_SMO_1026_final.pdf