R 軟體入門

沈彥廷　副統計分析師

本期 eNews 將為大家介紹近年來在各領域中皆已成為最流行的資料分析工具之一：R 軟體。我們將帶領讀者循序漸進了解 R 軟體的基本操作、資料型態等核心功能，期望能幫助程式初學者輕鬆跨越 R 語言相較略微陡峭的學習曲線！

底下簡單用三個 W 來為 R 進行初步簡介：

What

什麼是 R？R 軟體是基於 S 語言所建構而成的一套自由軟體，被廣泛應用於資料處理、統計分析、圖形繪製等功能。最初由 Ross Ihaka 及 Robert Gentleman 這兩位學者所開發，現在則是由「R 開發核心團隊」在負責開發維護，其網站主頁為 http://www.r-project.org/，其中包含各式說明文件及各主流作業系統的 R 軟體安裝檔，讀者可選擇就近的鏡像位址下載安裝 R 軟體，也別忘了多在網站上挖寶喔！

Why

為什麼要使用 R？相較於其他資料分析工具，R 有什麼優勢？首先，R 語言程式功能完整、擴充性強。由於 R 本身就是一種程式語言，因此即便是面對一個較新或是較獨特的問題而沒有現成套件(package)可以處理時，我們依然可以自行撰寫程式來解決問題。

其次，R 語言在網路上可以找到相當豐富的討論及教學文件，這絕對可以在學習一個新的軟體或語言的時候，替我們省去許多除錯時間的浪費。

第三，快速銜接最新技術。由於 R 是一個自由軟體，許多最新發表的統計方法、技術可能都很快就會被熱心的開發者編寫成 R 的套件發佈，隨著 R 的使用人口快速膨脹，其官方套件的發佈數量也呈現指數成長。

最後，可能也是對多數人而言最重要的一點，免費！市面上有眾多資料分析軟體，其中許多商業軟體皆是價格不斐，相較之下，在各方面皆有出色表現的 R 軟體實在可說是便宜又大碗。

Who

哪方面的領域或什麼樣的人適合使用 R？在 R 的網站中有一個「Task Views」頁面，裡面詳列了 R 軟體在各領域的適用套件及相關說明，我們不難從中看出 R 軟體並不僅僅侷限於統計相關領域。事實上，R 只是一個分析工具，只有願不願意用的問題、而沒有適不適用領域的問題。同理，只要您不排斥撰寫程式語言，那麼 R 語言絕對是一個值得學習的工具。

➢ 安裝相關工具

所謂「工欲善其事，必先利其器」，找到一個適合自己的程式開發工具，必定可提升開發效率，達到事半功倍之效。以下推薦兩套開發工具，讀者可依個人習慣及偏好，在安裝完 R 軟體後自由選擇安裝。

RStudio(http://www.rstudio.com/)：

RStudio 是一個整合開發環境(IDE)，將 R 軟體的主控臺、命令稿、工作空間、說明文件及圖表呈現區作一個集成，也更便於進行檔案和套件的管理，同時包含語法高亮顯示、除錯提示等特色。另有部分簡易的圖型化

點選介面，相當適合初學者入門使用。

Notepad++(http://notepad-plus-plus.org/)：

Notepad++實際上就是一套類似記事本般的文字編輯器，不過它同時支援多種程式語言的語法高亮、字詞自動完成等便於開發的功能。對於偏好乾淨介面、或是同時使用多種程式語言做為開發工具的人來說，Notepad++是另一個不錯的選擇。

➢ 基本操作規則

準備工作一切就緒，讓我們先來看一段 R 的程式碼，了解一下 R 語言的基本撰寫規則：

```
1  # 基本操作規則說明
2  varname <- 1
3  VARNAME = 2
4  x = 3; y = 4; z = 5
5  install.packages("rgl")
6  library(rgl)
7  help(surface3d)  # 等同於 ?surface3d
```

程式碼第 1, 7 行中的「#」符號在 R 語言中代表註解，接續在註解符號後面的單行文字都不會被執行，我們可以善用它來添加說明，提升程式的可維護性。

初次接觸 R 語言的讀者可以試著複製上面的程式碼到 R 軟體中執行，

從第 2, 3 行程式碼的執行結果，我們了解到「<-」和「=」都是指派(assign)符號，也就是分別將 1 和 2 這兩個數值分派儲存至 varname 及 VARNAME 變數名稱中的意思，基本上這兩種用法是相同的，選擇自己習慣用法即可；同時也可以看到在變數命名上，大小寫是有差異的，varname 和 VARNAME 代表的是不同的變數，這點初學者需要特別留意。

另外，一個完整的運算式可以用分號「;」表示結束，也允許不加分號直接換行即可，如程式碼第 4 行所示。

最後，程式碼 5~7 行為常用的基本內建函數(function)展示。有時候我們想進行分析的相關函數並不包含在內件的套件中，這時可使用 install.packages()函數來下載安裝指定的套件；使用 library()函數載入已正確安裝的套件，才能呼叫附屬於該套件內的函數。

help()大概是所有函數裡面最重要的一個了，畢竟 R 的函數指令成千上萬，我們要去記得每一個函數的使用方法那是不可能的，因此正確的學習方式應該是去了解如何運用說明文檔，利用這種方式來查詢一個新函數的使用方法。大家也趕快動手試試看吧！

➢ 基本運算規則

基本上，只要了解四則運算規則，我們可以把 R 軟體當作計算機來使用，一起來看看下面的範例：

```
1  1 + 2
2  3 - 4
3  5 * 6
```

```
4  7 / 8
5  9 ^ 0  # 等同於 9**0
6  1 + 2 * (3 / 4) ^ 5  # 先乘除後加減，有括號先運算
```

程式碼 1~4 行是簡單的四則運算，第 5 行則為 9 的 0 次方的意思，次方可用「^」或兩個星號「**」表達。在 R 語言中，同樣有先乘除後加減、有括號先運算的通則，如程式碼第 6 行所示，運算結果為 1.4746。

那麼如果是長度超過一個元素的向量運算呢？這也是在 R 語言中經常會運用到的技巧：

```
1  x = c(1, 2, 3, 4, 5)  # 等同於 1:5
2  y = x + 1
```

上方程式碼第 1 行建立了一個長度為 5 的數值向量，並指派儲存至變數 x，其中 c()是一個連結函數，是建立向量的方法之一，冒號「:」則是另一種建立有序數列的方法。還記得 help()函數嗎？如果不清楚 c()函數的效用，就趕快使用 help(c)或是?c 來查詢說明文件吧！

對向量進行數學運算時，R 語言有所謂循環特性，當兩向量長度不同時，會將短向量重複循環直至與長向量長度相等再進行運算。舉例來說，程式碼第 2 行中的 x+1，實際上是 x 向量 1,2,3,4,5 去加上 1,1,1,1,1(向量 1 循環 5 次)，因此運算結果向量 y 即為 2,3,4,5,6。

➢ 資料屬性

資料屬性及變數型態是學習程式語言中的一切基礎，建議初學者務必

在這個環節透徹理解，千萬別當差不多先生，否則往後可能只會花費更多時間在程式的除錯上，卻又找不出原因！以下介紹 R 語言中幾種主要的資料屬性：

```
1  ## R 軟體資料屬性: 邏輯真假值(logical,T,F), 整數(integer),
2  ##  雙倍精確度數字(double, real, numeric),
3  ##  複數(complex), 文字字串(character, string),
4  ##  二進位資料(raw)
5
6  1; 20.0; 3e2  # 數值
7  class(1)
8  "stat"  # 文字
9  class("stat")
10 TRUE; T; FALSE; F  # 邏輯真假值
11 class(T)
```

如同註解文字的說明，第 6 行中皆為數值型態的表示方式，其中 3e2 為科學符號計數法，表示為 3 乘以 10 的 2 次方。

在 R 語言中，文字字串的建立需使用成對的雙引號「"」或單引號「'」將文字含括起來，如第 8 行程式碼所示。

第三種資料型態為第 10 行的邏輯值，邏輯值在程式語言中是用來進行條件執行、流程控制不可或缺的一種資料型態，在這邊使用全大寫的 TRUE 表示為真；FALSE 表示為假，或者也可簡寫為大寫的 T 跟 F。

參考程式碼的第 7,9,11 行，我們可以使用 class()這個內建函數來判別資料的屬性。開程式開發的過程當中，若遇到不確定資料屬性的情況，建

議養成查詢確認的好習慣，畢竟如果將文字"1"看成數值 1、把邏輯值 F 和文字"F"混為一談，在後續的運算或邏輯判斷上可是有可能會出現預期之外的結果的。

➢ 變數型態

接著來看看 R 語言中常見的幾種變數型態，例如向量(vector)、陣列(array)、串列(list)等等。我們可以把變數型態想像成各種不同的資料儲存格式，而每一種儲存格式各有其特色及應用時機。因此例如同樣是向量變數，根據儲存的資料屬性不同，就可能會有數值向量、文字向量或邏輯值向量的差別。以下我們就來看看幾種變數型態的介紹和範例：

```
## R 軟體變數種類: 向量(vector), 矩陣(matrix), 陣列(array),
##  因子(factor), 資料框架(data-frame), 串列(list)

## 向量(vector) ##
x = c(1, 3, 5, 7, 9)  # 建立向量，或聯結不同的向量
y = c(2, "stat", T)  # 元素屬性需相同

## 陣列(array), 矩陣(matrix) ##
X = array(1:6, c(3, 2))  # matrix(1:6, 3, 2) # 建立陣列變數
Y = array(1:12, c(2, 3, 2))
rbind(x, x); cbind(x, x)  # 使用 rbind 及 cbind 來建立 array

## 因子(Factor) ##
```

```
14  x = c(1, 1, 1, 2, 2, 2)
15  y = factor(x)  # 等同於 as.factor(x)
16  y - 1
17
18  ## 串列(List) ##
19  L = list(L1 = x, L2 = y, L3 = X)
20
21  ## 資料框架(Data-Frame) ##
22  D = as.data.frame(X)  # 將變數類型轉為 data-frame
```

程式碼 4~6 行是 R 語言中最基本的向量變數型態，例如第 5 行建立的就是一個長度為 5 的數值向量。要特別注意的是，一個向量裡面的每個元素都必須是相同的資料屬性，讀者不妨嘗試執行第 6 行程式碼，就可以發現如果我們將數值、文字、邏輯值資料強迫連結成為一個向量時，R 就會自動將所有元素都轉換為文字屬性。

程式碼 8~11 行為陣列、矩陣變數型態，我們可以使用內建的 array()或 matrix()函數來建立二維陣列，如第 9 行所示，表示建立了一個 3 列 2 行的數值矩陣，其值為 1~6。其中，矩陣變數型態其實就是陣列變數的二維特例，若是要建立二維以上的多維陣列就只能使用 array()函數，如第 10 行的 2*3*2 三維陣列範例。此外也可以透過合併向量或陣列的方式來建立一個新的陣列，第 11 行中的 rbind 就是列(row)合併的意思；cbind 為行(column)合併的意思。與向量變數型態相同，陣列及矩陣變數內的所有元素也必須是相同的資料屬性，較不熟悉 R 語言的初學者需特別留意這種變數特性，可能一個不小心就會將數值陣列誤轉為文字陣列，導致後續的數學運算出現預期外的錯誤！

程式碼 13~16 行為因子變數。因子變數很類似文字的資料型態，但多了一個分類名稱(levels)的資訊屬性，主要常用於統計分析中的分類變數資料。我們可簡單利用內建的 factor()或 as.factor()函數將一個向量轉換為因子變數型態，如第 14, 15 行所示。又因為因子變數值在 R 軟體的畫面呈現上並沒有使用單引號或雙引號來含括，因此例如像第 15 行中 y 這樣的因子變數有時也常被誤認為是一個數值向量，但實際上如果對一個因子變數進行像是第 16 行般的數值運算其實是沒有任何意義的。

前面所介紹的向量、陣列或矩陣都有所有元素需為相同資料屬性的限制，那麼是否有一種變數型態是可以將多種不同屬性的資料儲存在一起呢？有的！就是串列變數。串列變數可說是 R 語言中最為彈性的一種變數型態，可以同時存放多種不同資料屬性、長度維度不一的元素。以程式碼第 19 行為例，L 是一個長度為 3 的串列變數，其中第一個元素是長度為 5 的數值向量、第 2 個元素是長度為 5 的因子變數、第 3 個元素則為一個 3 列 2 行的數值矩陣。

最後介紹資料框架變數型態。資料框架變數在 R 軟體中是個特別的存在，其類矩陣般的行列型態以及允許各行分別擁有不同資料屬性的特性特別適合用於進行資料分析。在程式碼第 22 行中，我們使用 as.data.frame()函數將數值矩陣 X 轉換為一個資料框架變數。雖然資料框架和矩陣看起來非常相似，但其實兩者之間的變數特性相差甚遠。事實上，我們應該把資料框架和串列放在一起比較，資料框架是串列的一個特例，兩者的變數特性唯一差別在於資料框架內每一個元素(每一行)資料長度必須相同，也因此資料框架才能以類矩陣的行列形式作呈現。

➢ 指標系統

我們可以利用指標(indexing)系統來擷取部分的資料，基本用法是使用成對的中括號「[]」搭配位置或名稱指標。和許多其他程式語言不同，在 R 語言中，指標的起始位置是 1 而非 0，這種性質讓使用者可較為直觀的進行資料處理，因此也非常適合用於資料分析。

指標系統的用法大致上根據不同的變數型態可能略有不同，以下先介紹向量、陣列和矩陣的基本指標用法：

```
1  v = c(1, 3, 5, 7, 9)  ## 向量
2  M = rbind(v, v+1)  ## 矩陣
3
4  v[1]
5  v[c(2, 4)]
6  v[-3]
7  M[1, 5]
8  M[, 2]
9  M[, c(1, 3)]
10 M[-1, -2]
```

首先在程式碼 1, 2 行分別建立了一個向量變數 v 和一個二維陣列 M，如果對向量或陣列變數型態還不熟悉，趕快再往前複習一下喔～

程式碼第 4 行代表從 v 向量中取出第 1 個元素；同理，我們可以在指標系統中使用向量同時擷取多個元素，例如第 5 行即表示取出 v 向量的第 2 和第 4 個元素，因此執行結果為 3, 7 向量。另外，也可以使用「-」號代表要排除的元素，如程式碼第 6 行所示，表示排除 v 向量的第 3 個元素，執行

結果為 1, 3, 7, 9。

指標系統在陣列矩陣上的用法也是大同小異，唯一差別在於擷取多維度資料時，不同維度之間以英文逗號區隔。以二維矩陣為例，程式碼第 7 行表示取出 M 矩陣第 1 列第 5 行的值，因此執行結果為 9。若是想取出該維度所有資料，則保持空白即可，如第 8 行範例表示單獨取出第 2 行，因此取出值會自動轉變為一向量，執行結果為 3, 4。讀者們不妨舉一反三，試想看看程式碼第 9, 10 行的執行結果會是如何呢？

最後，我們再來看看串列和資料框架在使用指標系統上有何不同？前面有提到過資料框架其實是串列的一個特例，因此兩者之間的指標使用方式也相當雷同，看看以下例子：

```
1  data(iris)  ## iris dataset
2  iris[, 5]
3  iris[, "Species"]
4  iris$Species
5  iris[["Species"]]
6  iris["Species"]  # 等同於 iris[5]
```

第 1 行中的 iris 是一個 R 軟體內建的範例資料，共 150 筆資料、5 個欄位，其資料型態為資料框架，因此當然也具有串列性質。

由於資料框架和矩陣一樣皆為二維行列呈現，因此資料框架也支援類似矩陣的指標表達方式，逗號前面為列、後面為行，如同第 2, 3 行所示，除了使用數字指標代表位置之外，也可使用欄位名稱當作名稱指標。兩者代表的意義都是取出 iris 中的第 5 行 Species 這個欄位資料。

串列變數型態的資料有另外幾種獨特的指標用法，就是可以使用「$」

符號以及兩個中括號「[[]]」來取出部分指定的資料，惟獨要注意的是使用「$」號取資料時，僅能使用名稱指標而不能使用位置指標。程式碼 2~5 行的執行結果完全相同。

比較特別的是最後一行的指標用法，雖然同樣可以取得 iris 第 5 行的資料，但若我們進一步去檢查其變數型態，就會發現取出的資料仍是資料框架型態而非一組向量。事實上若我們已掌握指標系統的概念，就會知道第 6 行程式碼在做的事情，其實是使用名稱指標取出一個串列變數內元素名稱為 Species 的元素資料，這種方法的回傳值仍然會是一個串列變數型態。在某些情況下，不同的資料回傳型態可能會導致後續運算分析出現非預期的錯誤，使用上不可不慎。

本期 eNews 對 R 軟體的介紹就先到這裡為止，後續將會再陸續介紹 R 的資料處理、分析、資料視覺化等實務應用教學。有興趣的讀者也請自行多加練習、實際操作撰寫程式，程式語言的學習沒有捷徑，唯勤而已！