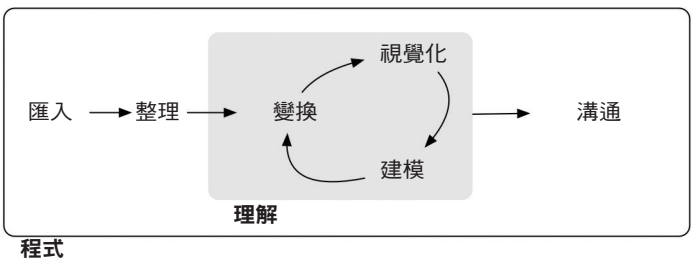
前言

資料科學（data science）是一門令人興奮的學科，能讓你將原始資料轉化為理解、洞察力，以及知識。*R 資料科學*的目標就是要協助你學習 R 中最重要的工具，讓你得以實作資料科學。閱讀本書之後，你會擁有應付各式各樣資料科學挑戰的工具，使用 R 最棒的部分。

你會學到什麼

資料科學是一個廣大的領域，你沒辦法只讀單一本書就精通它。本書的目標是要賦予你最重要工具的穩固基石。我們對於典型資料科學專案所需工具的模型看起來像這樣：



首先你必須將資料匯入（*import*）到 R 中。通常這代表你會將儲存在檔案、資料庫或 web API 中的資料載入到 R 中的 data frame。如果你無法將資料放到 R 中，就沒辦法對它進行資料科學研究！

匯入你的資料之後，通常你會接著進行一些**整理**（*tidy*）工作。整理資料意味著以一致的形式儲存它，讓其儲存方式符合資料集（dataset）的意義。簡單來說，如果你的資料是 tidy（經過整理）的，那麼每一欄（column）都應該是一個變數（variable），而每一列（row）都是一種觀察（observation）。Tidy data（經過整理的資料）之所以重要，是因為其一致的結構能讓你將心力集中在資料本身所要探討的議題，而非為了不同的功能忙著將資料轉換成正確形式。

一旦有了經過整理的資料，常見的下一步就是**變換**（*transform*）它。變換包括更仔細檢視感興趣的觀察（像是一個城市中所有的人，或是去年所有的資料）、建立以現有變數之函數所構成的新變數（像是從速度與時間計算出速率），以及計算一組摘要統計值（像是總數或平均）。整理與變換合稱為**整頓**（*wrangling*，或稱「角力」），因為將你的資料轉化為適合處理的形式，其過程就像是一場戰鬥！

只要得到帶有必要變數的 tidy data，你就可以運用兩種主要的知識產生引擎：視覺化（visualization）和模型建立（modeling，簡稱為「建模」）。它們的優缺點彼此互補，所以任何真正的分析工作，都需要在兩者之間來回進行許多次。

視覺化（*visualization*）是人類的一種基本活動。好的視覺化能夠展現出你沒預料到的資訊，或是對資料提出新的問題。好的視覺化也可能會提示你問錯問題了，或是需要收集不同的資料。視覺化能讓你感到驚訝，但其規模（scale）並不是很容易擴展，因為過程中需要人類進行解讀。

模型（*models*）是與視覺化互補的工具。只要你的問題足夠精準，就能使用一個模型來回答它們。模型基本上是一種數學或計算工具，所以它們的規模一般能夠輕易擴展。即使並非如此，購買更多電腦的花費通常會比雇用更多的人還要便宜！但每個模型都要做出一些假設（assumptions），而且就其本質而言，模型不能質疑它自己的假設，那代表一個模型基本上不會出乎你的意料。

資料科學的最後一步是**溝通**（*communication*），這是任何資料分析專案絕對關鍵的一部分。不管你的模型和視覺化讓你多麼了解你的資料，如果你無法將結果傳達給其他人，那就什麼用都沒有。

環繞在這些工具周圍的，就是**程式設計**（*programming*）。程式設計是你可以用在專案中每個部分、橫跨整個過程的工具。要當資料科學家，你不必是程式設計專家，但進一步學習程式設計是值得的，因為成為一名更好的程式設計師，能夠讓你自動化常見的任務，更輕易解決新的問題。

你會在每個資料科學專案中使用這些工具，但對大多數專案來說，它們並不足夠。這適用一種粗略的 80-20 規則：你能夠使用即將在本書中學到的工具處理每個專案大約 80% 的部分，但你還需要其他的工具來應付剩餘的 20%。在本書的各個地方，我們都會指引你何處可以找到進一步學習的資源。

本書的組織方式

前面對於資料科學工具的描述，大略是依據你分析時使用它們的順序來組織的（當然你會來回套用它們多次）。然而，在我們的經驗中，這並非學習它們的最佳方式：

- 從資料的攝入和整理開始，之所以是次優的，是因為 80% 的時間它們都是例行且無聊的工作，而另外 20% 的時間則是古怪且令人沮喪的。那並非開始學習一個新主題的好地方！取而代之，我們會從已匯入且整理過的資料之視覺化和變換工作開始。如此一來，你在攝入與整理自己的資料時，方能維持高動機，因為你知道那些痛苦是值得的。
- 某些主題最好與其他的工具一起說明。譬如說，我們相信，如果你已經知道視覺化、資料整理與程式設計在做些什麼，就會比較容易理解模型的運作方式。
- 程式設計工具本身並不一定有趣，但它們確實能讓你對付更具挑戰的問題。在本書的中間，我們會選出一些程式設計工具來介紹，然後你會看到它們如何能與資料科學的工具結合，處理有趣的模型建立問題。

在每一章中，我們都會試著依循一種相似的模式：先從一些有趣的例子開始，讓你可以看到整體畫面，然後再深入其中的細節。本書的每節都附有習題，幫助你複習學到的東西。雖然你會想要跳過習題，但沒有比實際練習真實問題還要好的學習方式了。

你不會學到什麼

有一些重要的主題本書不會涵蓋。我們相信，更重要的是堅持不懈地聚焦在最基本的部分，讓你可以盡快開始工作，進行分析。那代表本書沒辦法涵蓋每個重要主題。

Big Data

本書將焦點放在位於記憶體中的小型資料集（in-memory datasets）。這是起步的正確位置，因為沒有處理小型資料的經驗，你也無法應付大型資料。你在本書中學到的工具能夠輕易處理數百 MB 的資料，再花點小工夫，通常可以使用它們來處理 1 到 2 GB 的資料。如果你得經常處理大型資料（例如 10 到 100 GB），則應該更深入學習 data.table（*http://bit.ly/Rdatatable*）。本書沒有介紹 data.table，它的介面非常簡要，這使得它較難學習，因為提供的語法線索也較少，但如果你得處理大型資料，那麼能夠得到的效能增益，值得你多花費心力去學習它。

如果你的資料比這還要多，請仔細考量你的大型資料問題實際上是否可能是小型資料問題所偽裝。雖然完整的資料也許很多，但用來回答一個特定問題的資料量通常很少。你或許能夠找到某個子集、子樣本，或摘要，小到可以放入記憶體，但仍然能讓你用以回答你感興趣的問題。這裡的挑戰是如何找出那個正確的小型資料，這通常需要重複整個過程很多次。

另一種可能是，你的大型資料問題實際上是由為數眾多的小資料問題所組成。每個個別的問題也許都能放入記憶體，只是你有數百萬個這種小問題。舉例來說，你可能會想要將一個模型擬合（fit）你資料集中的每個人。如果你只有 10 或 100 個人，那將是輕而易舉，但你可能有一百萬個人。幸好每個問題都獨立於其他問題（這種情況有時被稱作 embarrassingly parallel，「平行到令人不好意思」），所以你只需要一個能讓你將不同資料集送到不同電腦上處理的系統（例如 Hadoop 或 Spark）。一旦你找出如何使用本書中所描述的工具回答單一個資料集的問題，你就能學會像是 sparklyr、rhipe 與 ddr 這些新工具來解決完整的資料集。

Python、Julia 和其他關聯工具

在本書中，你不會學到 Python、Julia 或其他對資料科學有用的程式語言。這並不是因為我們認為那些工具不好，它們很好！而在實務上，大多數資料科學團隊都會使用混合的語言，通常至少有 R 跟 Python。

然而，我們堅信，一次最好只學著掌握一個工具。如果你深入學習，你會更快上手，而不是把注意力分散到許多主題，每個都只分到薄薄一點。這並不代表你只需要熟悉一個工具就好，而是因為如果你一次只專注一件事，通常會學得比較快。在整個職業生涯中，你都應該奮力學習新東西，但是要移到下個有趣事物之前，請先確保你有紮實的理解。

我們認為 R 是開始你的資料科學旅程一個很棒的起點，因為它是一個從頭到尾都是為了支援資料科學而設計的環境。R 不只是一個程式語言，更是實作資料科學的一個互動式環境。為了支援這種互動，R 比其他類似的語言都還要有彈性得多。這種彈性伴隨著缺點，但它有很大的好處是，你可以輕易地為資料科學程序的特定部分發展出量身訂製的文法。這些迷你語言能幫助你以資料科學家的角度思考問題，同時讓你的大腦與電腦流暢地互動。

非矩形資料（Nonrectangular Data）

本書專門介紹矩形資料（rectangular data）：一組值（values）的集合，其中每個值都與一個變數（variable）和一項觀察（observation）有關聯。但有很多資料集並不符合這種模式，包括影像、音訊、樹狀結構，或是文字。不過矩形的資料框（data frames）在科學與工業界非常普遍，而我們相信它們會是開始你的資料科學之旅的好地方。

假設的確證（Hypothesis Confirmation）

資料分析可以分為兩個陣營：假設的產生（hypothesis generation）和假設的確證（有時稱為 confirmatory analysis，確證分析）。

本書的焦點毫不掩飾地放在假設的產生或資料探索之上。這裡你將會深入檢視資料，再配合你的主題知識，產生許多有趣的假設，來解釋為何會有那樣的資料。你會非正式地評估假設，發揮懷疑精神多方面挑戰你的資料。

與假設產生互補的是假設的確證。假設確證之所以困難，有兩個原因：

- 你需要精確的數學模型方能產生可證偽的預測（falsifiable predictions）。這通常需要博大精深的統計知識。
- 要確證一個假設，一項觀察只能使用一次。只要你多次使用，就回到了探索式分析（exploratory analysis）的工作。這表示，為了進行假設的確證，你得「預先註明」（preregister，事先寫好）你的分析計畫，而且即使你已經見過資料，也不能偏離計畫。我們會在第四部簡單介紹一些可讓這種工作更容易的策略。

我們常會認為建模（modeling）是用於假設確證的工具，而視覺化（visualization）則是產生假設的工具。但那是一種錯誤的二分法：模型（models）時常用於探索（exploration），而多留心一些，你就能使用視覺化來進行確證的工作。關鍵的差異在於，查看每項觀察的頻率：如果你只看一次，那就是確證；如果你會看多次，即為探索。

先備知識

關於你得先知道哪些知識才能從本書學到最多東西，我們有幾項假設。你應該對數字有一定的敏感度，而如果你已經有一些程式設計的經驗，那會有所幫助。若是你之前從未寫過程式，可能會發現 Garrett 所著的 *Hands-On Programming with R* 是實用的輔助讀本。

要執行本書中的程式碼，你需要四樣東西：R、RStudio、一組稱為 *tidyverse* 的 R 套件（packages），以及其他一些套件。套件是可再利用的 R 程式碼（reproducible R code）的基本單位。它們包括可重用的函式（functions）、描述使用方式的說明文件（documentation），以及樣本資料（sample data）。

R

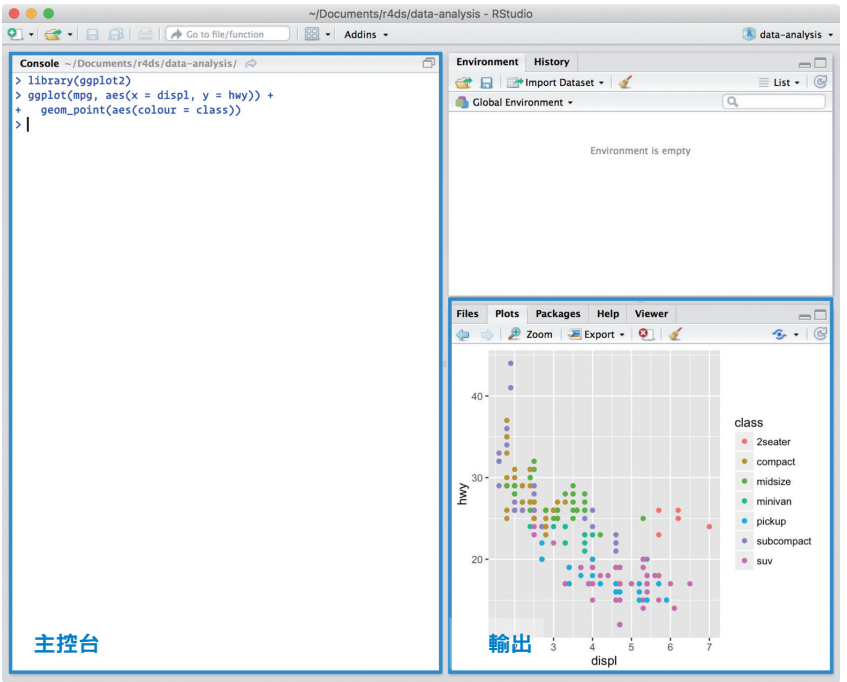
要下載 R，請前往 CRAN，即 *comprehensive R archive network*。CRAN 由一組鏡像伺服器所構成，分布在世界各地，用來散布 R 及 R 套件。不必試著去挑選離你較近的鏡像伺服器，只要使用雲端鏡像 *https://cloud.r-project.org* 即可，它就會自動為你找出最近的伺服器。

R 新的主要版本（major version）一年推出一次，而每年會發行二到三次的次要版本（minor releases）。定期更新會是個好主意。升級可能會有點麻煩，特別是主要版本，其中你得重新安裝你所有的套件，不過放著不管只會更糟。

RStudio

RStudio 是用於 R 程式設計的一個整合式開發環境（integrated development environment），或是 IDE。可以從 *http://www.rstudio.com/download* 下載並安裝它。RStudio 一年會更新數次，若有新版本可用，RStudio 會告知你。定期更新會是個好主意，如此你方能運用最新最好的功能。就本書而言，請確定你有 RStudio 1.0.0。

啟動 RStudio 後，你會看到介面上有兩個關鍵的區域：



現在你只要知道你是在主控台（console）面板輸入 R 程式碼，然後按下 Enter 執行它就行了。這一路上你會學到更多！

Tidyverse

你也需要安裝一些 R 套件。一個 R **套件**（*package*）是由一組函式、資料與說明文件所構成的集合，用來擴充基礎 R（base R）的功能。使用套件是成功運用 R 的關鍵。你會在本書中學到的大多數套件都是 tidyverse 的一部分。tidyverse 中的套件有關於資料與 R 程式設計的一套共通哲學，專門設計來互相搭配使用。

你可用單一行程式碼來安裝完整的 tidyverse：

```
install.packages("tidyverse")
```

在你的電腦上，於主控台輸入上面那行程式碼，然後按下 Enter 執行它。R 會從 CRAN 下載那些套件，並將它們安裝到你的電腦中。若安裝時發生問題，請先確定有連接到網際網路（Internet），以及 *https://cloud.r-project.org/* 並沒有被你的防火牆或代理伺服器阻擋。

使用 `library()` 載入（load）之前，你沒辦法使用一個套件中的函式、物件和說明檔。安裝好一個套件後，你可用 `library()` 函式來載入它：

```
library(tidyverse)
#> Loading tidyverse: ggplot2
#> Loading tidyverse: tibble
#> Loading tidyverse: tidyr
#> Loading tidyverse: readr
#> Loading tidyverse: purrr
#> Loading tidyverse: dplyr
#> Conflicts with tidy packages ----------------------------------
#> filter(): dplyr, stats
#> lag():    dplyr, stats
```

這裡告訴你 tidyverse 正在載入 **ggplot2**、**tibble**、**tidyr**、**readr**、**purrr** 及 **dplyr** 套件。這些被視為 tidyverse 的**核心**（*core*），因為幾乎在所有的分析中你都會用到它們。

tidyverse 中的套件變動的相當頻繁。你可以執行 `verse_update()` 來看看是否有更新可用，並選擇性地安裝它們。

其他套件

有許多其他的優秀套件並非 tidyverse 的一部分，因為它們解決不同領域的問題，或是背後的設計原則不同。這並不會讓它們顯得更好或更壞，只

是不同而已。換句話說，與 tidyverse（整齊宇宙）互補的並非 messyverse（雜亂宇宙），而是互相關聯的其他許多宇宙。隨著你以 R 處理更多的資料科學專案，會學到新的套件，以及思考資料的新方式。

在本書中，我們會使用 tidyverse 以外的三個資料套件：

```
install.packages(c("nycflights13", "gapminder", "Lahman"))
```

這些套件提供航空公司航班、世界發展，以及棒球的資料，我們會用它們來說明關鍵的資料科學想法。

執行 R 程式碼

前一節展示了 R 程式碼的幾個執行範例。本書中的程式碼看起來像這樣：

```
1 + 2
#> [1] 3
```

如果在你本地端的主控台（console）執行相同的程式碼，看起來會像這樣：

```
> 1 + 2
[1] 3
```

這裡有兩個主要的差異。在你的主控台中，你是在叫做**提示符號**（*prompt*）的 `>` 後面輸入程式碼，而我們在本書中不顯示那個提示符號。在本書中，輸出（output）會以 `#>` 來標示，而在你的主控台中，它會直接出現在程式碼的後面。這兩個差異代表著，如果你用的是本書的電子版本，就能輕易地從書中複製程式碼，並將之貼到主控台中。

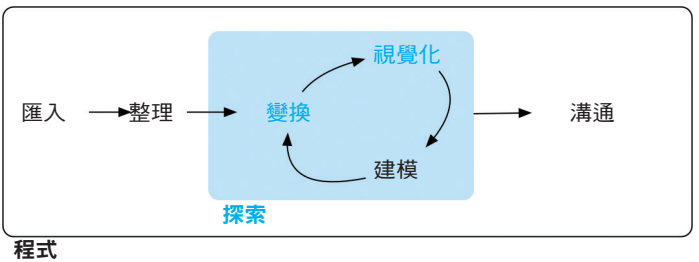
在本書中，我們使用前後一致的一組慣例來參考（refer to）程式碼：

- 函式是用程式碼字體，後面接著括弧（parentheses），例如 `sum()` 或 `mean()`。
- 其他的 R 物件（像是資料或函式引數）也是程式碼字體，但沒有括弧，例如 `flights` 或 `x`。
- 如果我們想要清楚表明某個物件源自哪個套件，我們會用套件名稱後面接著兩個冒號（colons），像是 `dplyr::mutate()` 或 `nycflights13::flights`。這在 R 程式碼中也是合法有效的。

第一部

探索

本書第一部的目標是要讓你盡快熟悉資料探索（*data exploration*）的基本工具。資料探索是檢視你的資料，迅速產生假設（hypotheses），快速測試它們，然後一再重複這個流程的技藝。資料探索的目標是要產生許多有潛力的路線，讓你之後可以更深入探索。



在本書的這個部分你會學到能有立即報償的一些實用工具：

- 視覺化（visualization）是學習 R 程式設計很好的起點，因為回報很明顯：你得以製作優雅又資訊豐富的圖表（plots），幫助你理解資料。在第 1 章中，你會深入視覺化的過程，學習 **ggplot2** 圖表的基本結構，以及將資料轉為圖表的強大技巧。
- 單是視覺化通常並不足夠，所以在第 3 章中，你會學到能讓你選擇重要變數、過濾出關鍵觀察、建立新變數，並計算出摘要資訊的重要動詞。
- 最後，在第 5 章中，你會將視覺化與變換（transformation）跟你的好奇心與懷疑精神結合起來，提出並解答有關資料的有趣問題。

建模（modeling）是探索過程中很重要的一個部分，但你尚未擁有有效學習或應用它的技能。我們會等到第四部，你具備更多更好的資料整頓（data wrangling）和程式設計工具時，再回頭討論它。

穿插在教你探索工具這三章之間的，是專注於你的 R 工作流程（R workflow）的另外三章。在第 2、第 4 與第 6 章中，你會學到編寫與組織你 R 程式碼的良好實務做法。長期來講，這將會帶給你成功，因為它們會賦予你組織工具，讓你在處理真實專案時，保持井然有序。

第一章

使用 ggplot2 的資料視覺化

簡介

> 簡單的圖表帶給資料分析者的資訊量，比任何其他工具都還要多。
>
> —John Tukey

本章會教你如何使用 **ggplot2** 視覺化（visualize）你的資料。R 用於製作圖表的系統有好幾個，不過 **ggplot2** 是其中功能最多樣也最優雅的一個。**ggplot2** 實作了**圖形的文法**（*grammar of graphics*），一個清楚易懂的系統，專門用來描述和建置圖形。藉由 **ggplot2**，你可以透過學習一個系統並將之應用在許多地方，而更快做到更多事情。

如果開始之前你想進一步了解 **ggplot2** 的理論基礎，推薦你閱讀「A Layered Grammar of Graphics」（*http://vita.had.co.nz/papers/layered-grammar.pdf*）。

先備條件

本章專門介紹 **ggplot2**，它是 tidyverse 的核心成員之一。要取得我們在本章中會使用的資料集（datasets）、說明頁面（help pages）及函式，請執行以下程式碼來載入 tidyverse：

```
library(tidyverse)
#> Loading tidyverse: ggplot2
#> Loading tidyverse: tibble
#> Loading tidyverse: tidyr
#> Loading tidyverse: readr
#> Loading tidyverse: purrr
```

```
#> Loading tidyverse: dplyr
#> Conflicts with tidy packages ----------------------------------
#> filter(): dplyr, stats
#> lag():    dplyr, stats
```

此程式碼會載入 tidyverse 的核心，也就是幾乎在你每天的資料分析工作中都會用到的套件（packages）。這也會告訴你來自 tidyverse 的哪個函式與基礎 R（base R，或你所載入的其他套件）中的函式有衝突（conflict）。

如果你執行了這段程式碼，卻得到錯誤訊息「there is no package called 'tidyverse'」（沒有叫做 tidyverse 的套件），就得先安裝它，然後再次執行 `library()`：

```
install.packages("tidyverse")
library(tidyverse)
```

每個套件只需要安裝一次，但當每次你啟動一個新的工作階段（session）時，就得重新載入它。

如果我們要清楚表明某個函式（或資料集）來自何處，會使用 `package::function()` 這種特殊形式。譬如，`ggplot2::ggplot()` 明確地告知我們所用的是來自 **ggplot2** 套件的 `ggplot()` 函式。

第一步

讓我們用第一個圖表來回答這個問題：大引擎的車子會比小引擎的車子使用更多的燃料嗎？你心裡或許已經有了答案，不過我們先試著讓這個問題更精確些。引擎的大小（engine size）與燃油效率（fuel efficiency）之間的關係看起來會像什麼樣子呢？是正相關？負相關？線性？非線性？

mpg 資料框

你可以使用 **ggplot2** 中的 `mpg`（即 `ggplot2::mpg`）資料框（data frame）來測試你的答案。一個 *data frame* 是由（欄中的）變數（variables）和（列中的）觀察（observations）所成的一個矩形集合。`mpg` 含有美國國家環境保護局（US Environment Protection Agency）基於 38 個車型所收集的觀察資料：

```
mpg
#> # A tibble: 234 × 11
#>   manufacturer model displ  year   cyl      trans   drv
#>          <chr> <chr> <dbl> <int> <int>      <chr> <chr>
#> 1         audi    a4   1.8  1999     4   auto(l5)     f
```

```
#> 2          audi    a4   1.8  1999     4 manual(m5)     f
#> 3          audi    a4   2.0  2008     4 manual(m6)     f
#> 4          audi    a4   2.0  2008     4   auto(av)     f
#> 5          audi    a4   2.8  1999     6   auto(l5)     f
#> 6          audi    a4   2.8  1999     6 manual(m5)     f
#> # ... with 228 more rows, and 4 more variables:
#> # cty <int>, hwy <int>, fl <chr>, class <chr>
```

mpg 中的變數有：

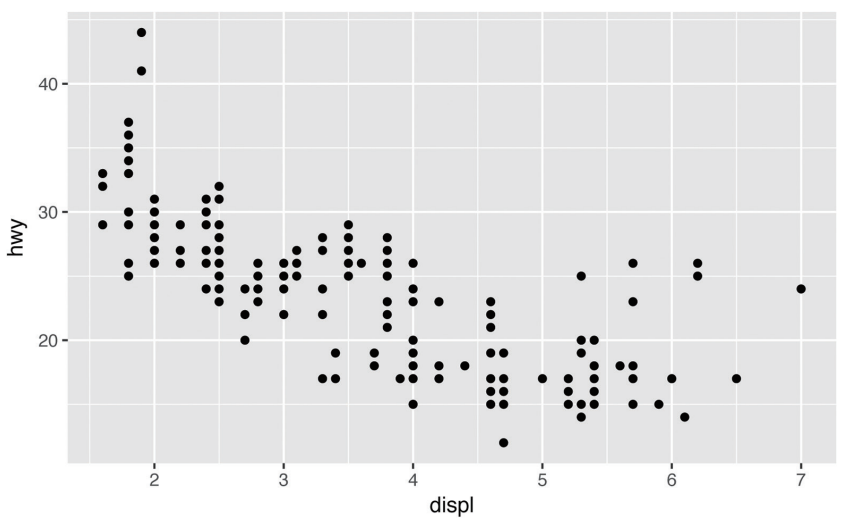
- displ，車子的引擎大小，單位為公升（liters）。
- hwy，車子在高速公路（highway）上的燃油效率，單位是每加侖的英里數（miles per gallon，mpg）。具有較低燃油效率的車子行進同樣的距離所消耗的燃料會比高燃油效率的車子還要多。

想了解 mpg 的更多資訊，就執行 ?mpg 來開啟它的說明頁面。

建立一個 ggplot

要繪製 mpg，就執行這段程式碼，把 displ 當 x 軸（x-axis），hwy 當 y 軸（y-axis）：

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



這個圖表顯示，引擎大小（displ）與燃油效率（hwy）之間有負相關（negative relationship）。換句話說，大引擎的車子使用的燃料更多。這確認還是駁斥了你對燃油效率和引擎大小之間關係的假設呢？

藉由 **ggplot2**，你會先從函式 ggplot() 開始繪製一個圖表，ggplot() 建立了一個座標系統（coordinate system），你能夠疊加不同的圖層上去。ggplot() 的第一個引數（argument）是要在圖表中使用的資料集。因此 ggplot(data = mpg) 會建立一個空的圖表，但那並不是很有趣，所以我不會在此展示它。

你可以新增一或多個圖層（layers）給 ggplot() 來完成你的圖表。函式 geom_point() 為你的圖表新增一個帶有黑點的圖層，這就建立了一個散布圖（scatterplot）。**ggplot2** 帶有許多 geom 函式，每個都能為一個圖表新增不同類型的圖層。你會在本章中學到很多個。

ggplot2 中的每個 geom 函式都接受一個 mapping 引數，它定義在資料集中的變數如何被映射（mapped）到視覺特性（visual properties）。mapping 一定會與 aes() 搭配使用，而 aes() 的 x 與 y 引數則指定哪些變數映射到 x 軸與 y 軸。**ggplot2** 會在 data 引數中尋找對應的變數，在此即為 mpg。

繪圖範本（Graphing Template）

讓我們將這段程式碼轉為一個可重用的範本（template），用來以 **ggplot2** 製作圖表。要製作一個圖表，就把下列程式碼中以角括號圍起的部分替換為一個資料集、一個 geom 函式，或一個 mapping 集合：

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

本章剩餘的部分會示範如何補完並擴充這個範本來製作不同類型的圖表。我們會先從 <MAPPINGS> 的部分開始。

習題

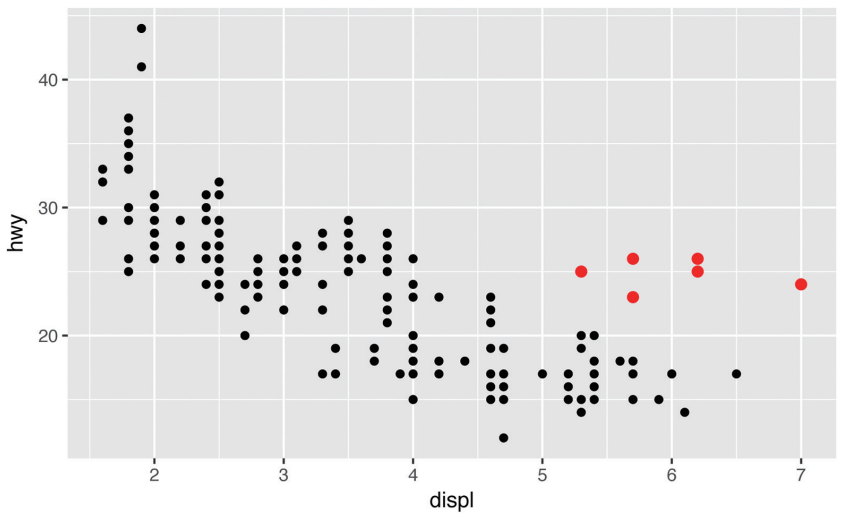
1. 執行 ggplot(data = mpg)。你會看到什麼？
2. mtcars 中有多少列（rows）？有多少欄（columns）？
3. drv 變數描述的是什麼？請閱讀 ?mpg 的說明來找出答案。
4. 製作 hwy 對 cyl 的散布圖。

5. 如果你製作出 class 對 drv 的一個散布圖會發生什麼事？為何這個圖表沒有用處？

Aesthetic Mappings（美學映射）

> 圖表最重大的價值在於，它迫使我們注意到從沒預料會看到的東西。
>
> —John Tukey

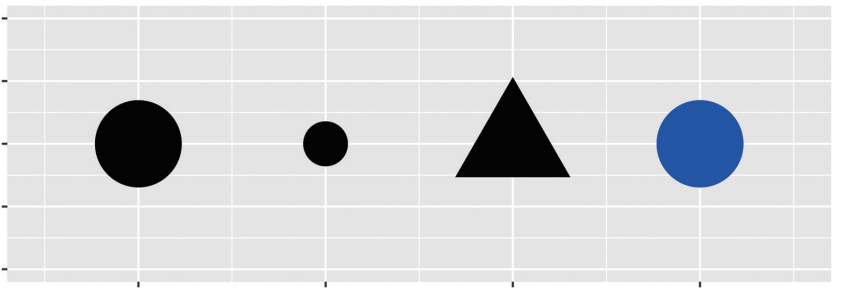
在下面的圖表中，有一組點（以紅色凸顯）似乎落到線性趨勢之外。這些車子的里程數比你預期的還要高。你如何解釋它們的表現呢？



讓我們假設那些車子是油電混合（hybrids）的。要測試這個假設，其中一個方式是查看每輛車的 class 值。mpg 資料集的 class 變數將車子分類為緊湊型車（compact）、中型車（midsize）和運動型多用途車（SUV）。如果那些異數點是油電混合車，它們應該被分類到緊湊型車輛，又或者是次緊湊型（subcompact）車（要記住的一點是，這些資料是在油電卡車與 SUV 車普及之前所收集的）。

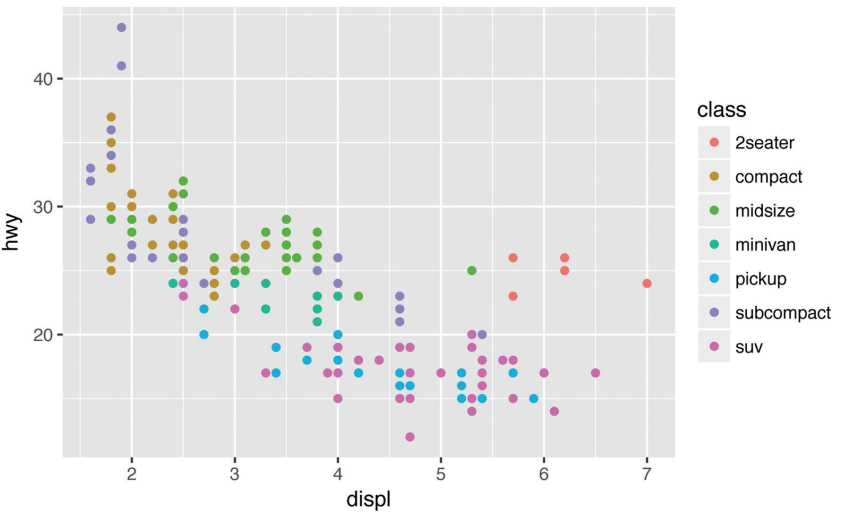
你可以新增第三個變數，例如 class，到二維（two-dimensional）的散布圖中，方法是將它映射到一個 *aesthetic*。一個 aesthetic 是你圖中物件的某個視覺特性。這裡的 aesthetics 包含像是點的大小、形狀或顏色等特性。你可用不同方式顯示一個點（就像接下來展示的圖），只要變更其美學特性

（aesthetic properties）的值就行了。既然我們已經用到「value（值）」這個詞來描述資料，那就讓我們用「level（程度、層次或級別）」來描述那些美學特性。這裡我們更改了一個點的大小、形狀及顏色的 level 來使點變小、變為三角型或變成藍色：



你可以把圖表中的 aesthetics 映射到資料集中的變數，以傳達有關你資料的資訊。舉例來說，你可以將點的顏色（color）映射到 `class` 變數來顯示每輛車的類型：

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



（如果你跟 Hadley 一樣偏好英式英文，可以用 `colour` 來取代 `color`。）

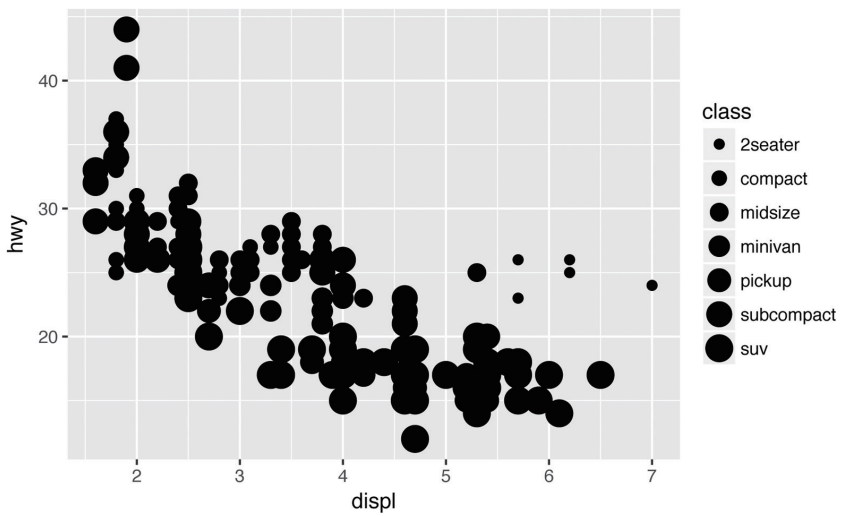
要把一個 aesthetic 映射到某個變數，就在 `aes()` 內將那個 aesthetic 關聯至該變數的名稱。**ggplot2** 會自動為該變數每一個獨特的值指定 aesthetic 的

一個唯一的 level（在此即唯一的顏色），這個過程稱為 *scaling*（**標定**）。**ggplot2** 也會新增一個圖例（legend）來說明哪個 level 對應到哪個值。

加上顏色之後，圖表顯示出那些奇特的點大多是兩人座的車（two-seater cars）。那些車子看起來不像是油電混合的，實際上應該是跑車才對！跑車跟 SUV 和貨卡（pickup trucks）一樣有大引擎，但車身較小，就像是中型車或緊湊型車，因此改善了它們的里程數。從後見之明來看，那些車不太可能會是油電混合的，因為它們有大型引擎。

在前面的例子中，我們將 `class` 映射到 color（顏色）aesthetic，但我們也能以同樣的方式將 `class` 映射到 size（大小）aesthetic。在這種情況下，每個點的大小反映出車子所屬的類型。在此我們會看到**警告**（*warning*）訊息，因為將無序變數（unordered variable，`class`）映射到有序的 aesthetic（`size`）並不是一個好主意：

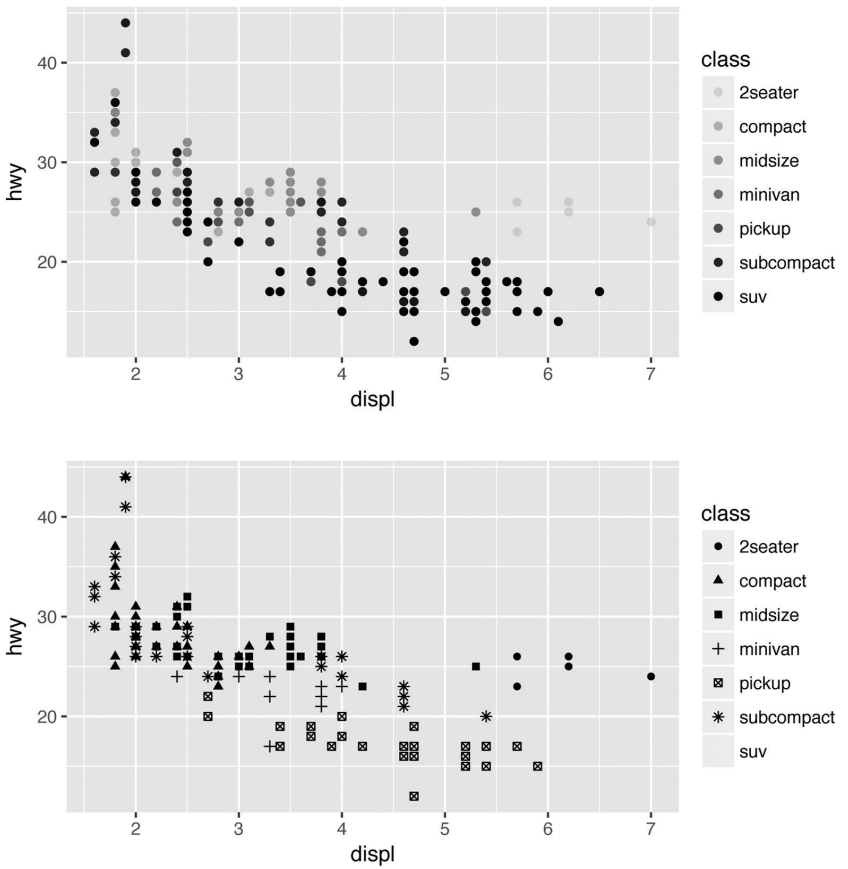
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, size = class))
#> Warning: Using size for a discrete variable is not advised.
```



又或者我們能把 `class` 映射到 *alpha* aesthetic，它會控制點的透明度（transparency），或點的形狀（shape）：

```
# 頂端
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class))
```

```
# 底部
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, shape = class))
```



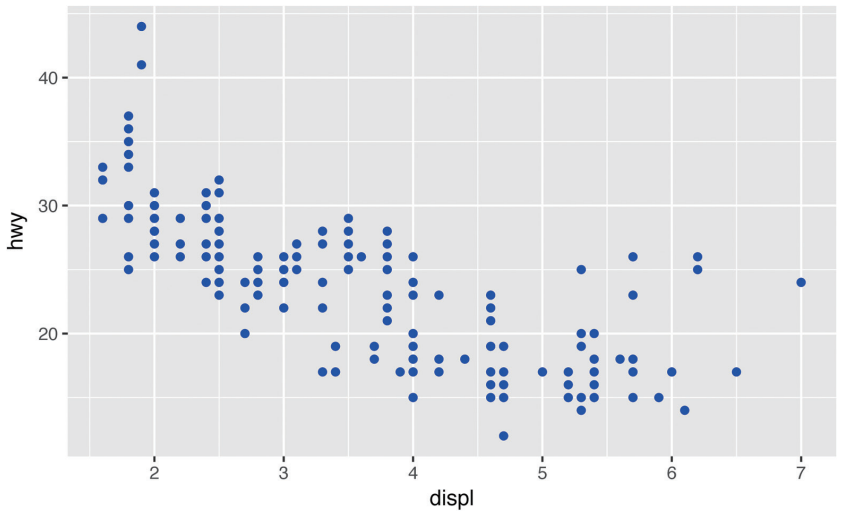
第二個圖中的 SUV 怎麼了呢？**ggplot2** 一次只會使用六個形狀。預設情況下，當你使用這個 aesthetic 時，額外的群組將不會被繪製出來。

對於你所用的每一個 aesthetic，aes() 會將那個 aesthetic 的名稱關聯至某個變數以顯示。aes() 會把一個圖層所用的每個 aesthetic mapping 聚集起來，並將它們傳給該圖層的 mapping 引數。這裡的語法凸顯出了有關 x 與 y 的實用洞見：一個點的 x 與 y 位置本身也是 aesthetics，即你可以將之映射到變數以顯示資料相關資訊的視覺特性。

只要你映射了一個 aesthetic，**ggplot2** 就會處理剩下的工作。它會選擇一個合理的標度（scale）來搭配所用的 aesthetic，並建構一個圖例（legend）來說明 levels 與變數值之間的映射關係。對 x 與 y 這兩個 aesthetics 來說，**ggplot2** 並不會建立圖例，而是建立帶有刻度及一個文字標籤（label）的軸線（axis line）。這種軸線的用途就跟圖例一樣，說明位置（locations）與值（values）之間的映射關係。

你也能手動設定（*set*）geom 的美學特性。譬如說，我們可以讓圖表中所有的點都變為藍色：

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), color = "blue")
```



在此，顏色並沒有傳達有關變數的資訊，只是改變圖表的外觀。要手動設定一個 aesthetic，就以此 aesthetic 的名稱作為你 geom 函式的一個引數，也就是要放在 `aes()` 外面。你得為此 aesthetic 挑選一個合理的值：

- 以顏色名稱作為一個字元字串。
- 點的大小以 mm（公釐）為單位。
- 一個點的形狀以數字表示，如圖 1-1 所示。其中有一些看似重複的形狀：例如 0、15 和 22 全都是方形。其差異來自於 `color` 與 `fill` aesthetic 的互動。那些中空的形狀（0–14）有由 `color` 所決定的邊框；實心的形狀（15–18）是由 `color` 所填滿；而填充的形狀（21–24）具有 `color` 的邊框並以 `fill` 填滿。

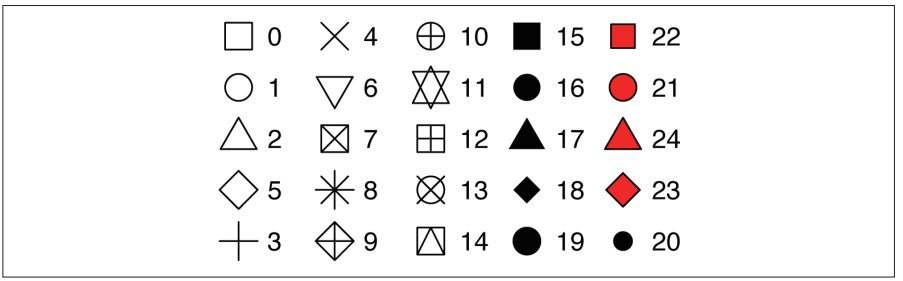
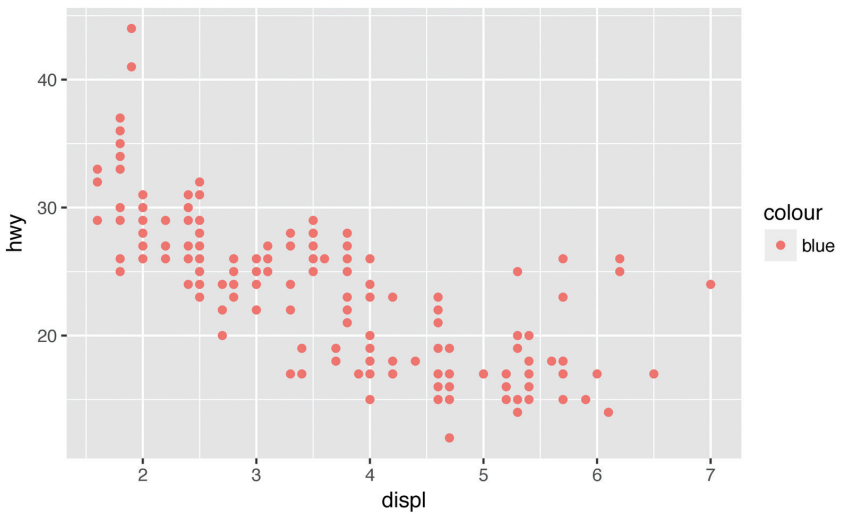


圖 1-1　R 具有 25 個以數字來識別的內建形狀

習題

1. 這段程式碼中有何錯誤呢？為何點不是藍色（blue）的？

```
ggplot(data = mpg) +
  geom_point(
    mapping = aes(x = displ, y = hwy, color = "blue")
  )
```



2. mpg 中的哪些變數是類別型（categorical）的呢？哪些變數是連續（continuous）的呢？（提示：輸入 ?mpg 來閱讀該資料集的說明文件）。當顯示 mpg 資料集時要如何才能看到這些資訊？

3. 將一個連續變數映射到 color、size 與 shape。就類別變數 vs. 連續變數而言，這些 aesthetics 在行為上有何不同呢？

4. 如果你將相同的變數映射到多個 aesthetics 會發生什麼事呢？

5. stroke aesthetic 是做什麼用的呢？它可以搭配什麼形狀使用？（提示：使用 ?geom_point）。

6. 如果你將一個 aesthetic 映射到不是變數名稱的東西，像是 aes(color = displ < 5)，會發生什麼事呢？

常見問題

隨著開始執行 R 程式碼，你很有可能會遇上問題。別擔心，每個人都會這樣。我已經編寫 R 程式碼好幾年，但每天仍然會寫出行不通的程式碼！

請先仔細比對你所執行的程式碼和書中的程式碼。R 非常挑剔，一個放錯位置的字元都可能造成完全不同的結果。請確定每個 (都有對應的)，而每個 " 都與另一個 " 成對出現。有時你執行了程式碼卻什麼都沒發生，那就檢查主控台（console）的左手邊，如果出現的是一個 +，則代表 R 認為你尚未輸入一個完整的運算式（expression），正在等候你完成它。在這種情況下，只要按下 Esc 來放棄目前進行中的指令，就能輕鬆地重新開始。

建立 **ggplot2** 時常碰到的一個問題是，你將 + 放錯了位置：它必須放在行結尾（end of the line），而非開頭。換句話說，得確定你沒有不經意地寫出像這樣的程式碼：

```
ggplot(data = mpg)
+ geom_point(mapping = aes(x = displ, y = hwy))
```

如果你仍然卡住，就試著查看說明頁面。你可以在主控台執行 ?function_name，或是選擇函式名稱並在 RStudio 中按下 F1，就能取得任何 R 函式的說明。如果說明文字看起來沒什麼幫助，也別擔心，請直接跳到範例（examples）的部分，尋找跟你試著要做的事情相符的程式碼。

如果那也幫不上忙，請仔細閱讀錯誤訊息（error message），有時候答案就埋藏在其中！不過如果你還是 R 的新手，答案也許在錯誤訊息中沒錯，只是你尚不知道如何理解它。另外一個很棒的工具是 Google：試著 google 錯誤訊息，因為很有可能其他人也遇過同樣的問題，並在線上發問而得到解答。