從貝氏定理衍生的貝氏推斷學派，不同於本書採用的「頻率學派」（frequentist approaches）和經典統計方法。

29 機率分布

到目前為止，我們學習到了擲骰子是　個隨機實驗，也知道了　次實驗中會出現樣本空間中的可能結果。每　個可能結果的機率加總起來必等於 1，但是，每　個可能結果的相對機率又是多少呢？我們將這個問題稱之為**機率分布**。機率分布描述的是，　個事件中會出現哪些可能結果，而這些結果各自發生的頻率又是多少。雖然機率分布可以用正式的數學公式呈現，我們在此想關注的是如何量化機率分布的輸出值。

在第 1 章中，你學習到了離散變數和連續變數。在機率論中，也存在著「離散機率分布」和「連續機率分布」的區別。我們先從離散機率分布開始介紹。

離散機率分布

繼續使用擲骰子的範例，這是　種**離散**機率分布，因為骰子點數是「可數的」整數，舉例來說，你可能會擲出 2 點或 3 點，但絕不可能扔出 2.25 點。

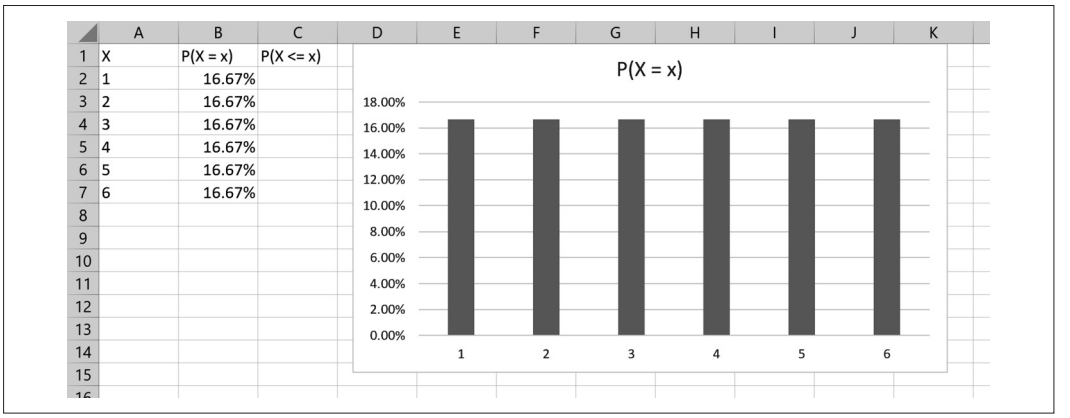
擲骰子是　種**離散均勻**分布，因為在每　次實驗的可能結果，其發生機率都是相同的。也就是說，在每　次扔骰子時，我們擲出 4 點的機率，和擲出 2 點的機率相同。更具體　點，每個點數的發生機率都是六分之　。

為了參照本章範例和其他 Excel demo，請下載本書範例檔（*https://oreil.ly/1hlYj*）的 *ch 2.xlsx*。在這些演練題中，我已經在工作表分頁中完成前置步驟，讓我們　起處理剩下的分析工作。先從 *uniform distribution* 分頁開始。`A2:A7` 儲存格範圍列出了 X 的每個可能結果。我們已知每個結果的出現機率是相同的，因此，在 `B2:B7` 範圍的公式應為 `=1/6`。$P(X=x)$ 表示某給定事件的可能結果之發生機率。

現在，請選取 `A1:B7` 範圍，並選擇功能區的 [插入]>[叢集欄位]。機率分布圖表應如圖 2 1 所示。

恭喜你得到了第　張機率分布圖。注意到圖表中個值之間的間隙了嗎？這表示這是**離散**的可能結果，而不是連續的結果。

有時候我們想知道，某個可能結果的**累積機率**。這時，我們會累計所有可能結果的發生機率，直到總和達到 100%（因為樣本空間的總和必等於 1）。某個事件的機率必然**小於或等於** C 欄內的給定結果。對 C2:C7 範圍套用公式 =SUM(B2:B2)。



	A	B	C
1	X	P(X = x)	P(X <= x)
2	1	16.67%	
3	2	16.67%	
4	3	16.67%	
5	4	16.67%	
6	5	16.67%	
7	6	16.67%	

圖 2-1　六面骰子的機率分布

現在，請選擇 A1:A7 範圍，按住 Ctrl 鍵或 Cmd 鍵，然後反亮選取 C1:C7 範圍，再建立第二個叢集圖表。從圖 2 2，你是否看出機率分布和**累積**機率分布的差別了呢？

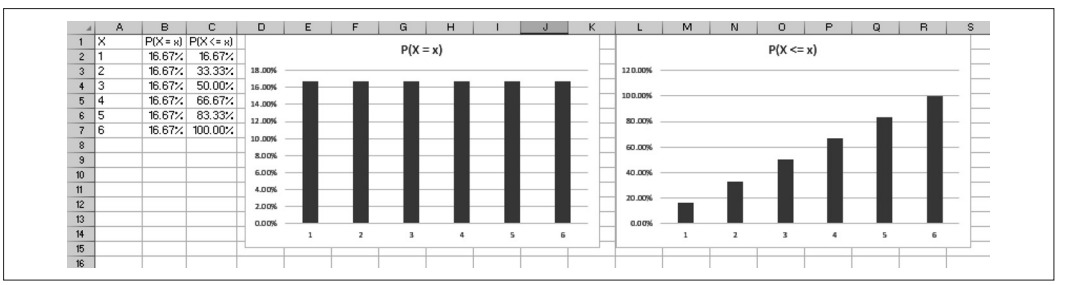
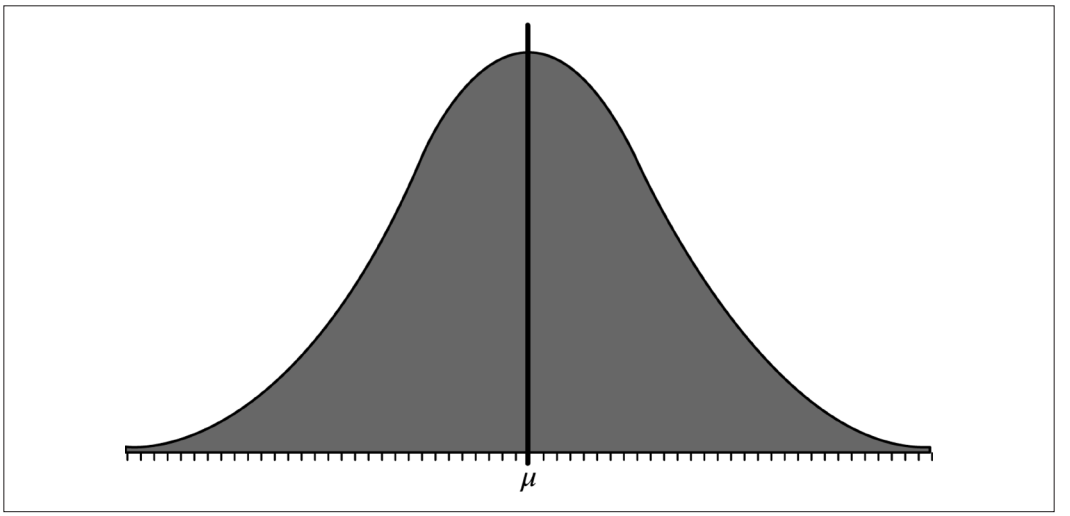


圖 2-2　六面骰子的機率分布 vs. 累積機率分布

根據邏輯思考和數學推導，我們假定了每　面骰子出現的機率都是 1/6，這稱為**理論機率**。我們也可以數次扔擲骰子並記錄結果，找出**實驗機率**。經過多次實驗後，我們將會發現，每　面骰子出現的機率**並不是**理論中的 1/6，骰子會傾向出現某　特定面。

推導出實驗機率的方法有好幾種：首先，我們可以真的進行實驗。不過，擲幾百次骰子並且記錄每　次出現的點數，這件事聽起來頗為枯燥。另　種方式是讓電腦代勞**模擬**實



33 圖 2-4　以長條圖繪製的常態分布

常態分布的例子在自然世界中相當常見。比方說，圖 2 5 分別展示了學生身高的分布情形以及紅酒的 pH 值。這些資料集分別位於本書範例檔（*https://oreil.ly/1hlYj*）中 *datasets* 資料集的 *heights* 和 *wine* 子資料夾中。

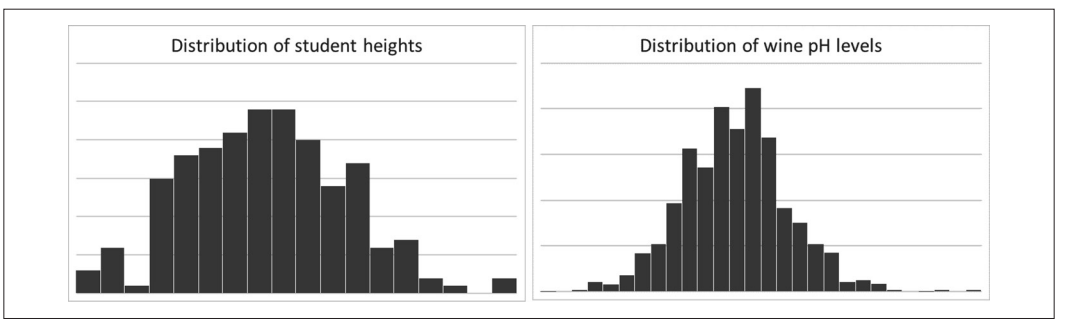


圖 2-5　兩個日常生活中的常態分布實例：學生身高與紅酒 pH 值

你可能會好奇，我們要如何得知某個變數是否為常態分布？這是　個很棒的問題。回想　下擲骰子的例子：我們列出了所有可能的結果，推導出　個理論機率分布，然後（透過模擬）推導出實驗機率分布，然後比較兩種分布情形。請將圖 2 5 的長條圖分別想成是學生身高和紅酒 pH 值的**實驗機率分布**：在這種情況下，資料點是人為收集的，而不是靠模擬得來的。

第四章

相關與迴歸

你有沒有聽說過「冰淇淋的消費量和鯊魚攻擊的頻率有相關」？顯然，大白鯊對薄荷巧克力口味冰淇淋很有興趣呢。圖 4 1 畫出了這個假設關係。

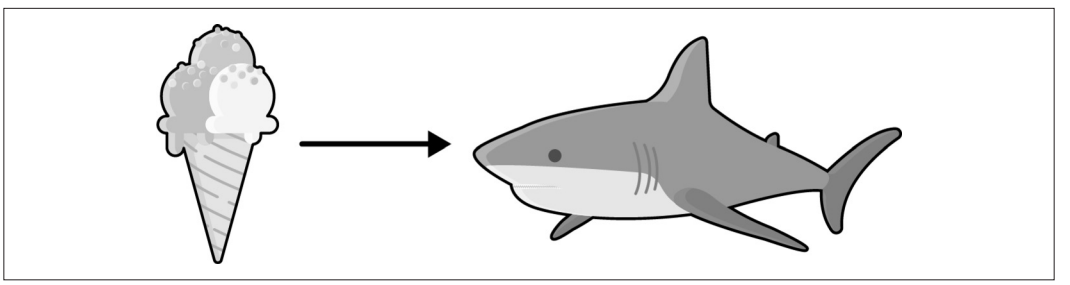


圖 4-1　冰淇淋消費量和鯊魚攻擊的假設關係

「開什麼玩笑」，你心中這麼想：「這不能代表冰淇淋消費量導致了鯊魚攻擊事件。」

你試著推理：「可能的情況是，當氣溫升高，為了消暑，人們越傾向購買冰淇淋。在天氣好的日子裡，人們在海邊逗留的時間越長。這種巧合導致了更多的鯊魚攻擊事件。」

「相關不代表因果」

「相關不代表因果」這句話你可能早已聽過千萬遍。

從第 3 章的內容中，你體會到**因果**（*causation*）在統計學中是一種令人憂心的表達方式。我們唯一能做的是拒絕「虛無假說」，畢竟我們不可能蒐集到所有資料，信誓旦旦地聲明事物之間的因果關係。撇開語意差異不談，相關性（correlation）和因果關係之間有任何關係嗎？標準定義有些過於簡化兩者的關係；在本章中，你將透過推論統計工具一探究竟。

這將是我們以 Excel 為中心討論的最後一個章節。你得以充分掌握分析框架，為使用 R 和 Python 進行統計分析做好準備。

相關性

到目前為止，我們主要是一次分析單個變數的統計資料。舉例來說，我們之前計算過閱讀測驗平均分數和房價變異數。這種分析方式被稱為**單變量分析**（*univariate analysis*）。

我們還做過一些**雙變量分析**（*bivariate analysis*）。比方說，我們使用了雙向次數分配表，比較兩個類別變數的頻率。我們也分析了按類別變數的多層次進行分組的連續變數，找出每一組別的敘述統計資料。

我們現在將使用相關性計算兩個連續變數的雙變量尺度。更具體而言，我們要使用**皮爾森相關係數**（*Pearson correlation coefficient*），測量兩個變數的線性關係之強度。如果兩者之間不具有線性關係，則不適用皮爾森相關係數。

那麼，該如何判斷手中資料是否呈現線性關係呢？當然存在更加嚴謹的檢驗方式，但老樣子，你可以從「視覺化」開始。我們將使用**散布圖**（*scatterplot*），繪出所有觀察值的 x 座標和 y 座標。

如果我們可以用一條線概括資料的整體散布情況，則資料呈線性，可使用皮爾森相關係數。如果得用曲線或其他圖形才能概括整體，則資料不具線性關係。圖 4-2 展示了一個線性關係圖，以及兩個非線性關係。

圖 4-2 給出了一個**正相關**（*positive*）的線性關係：當 x 軸的值增加，y 軸的值也會跟著（以固定的線性斜率）增加。

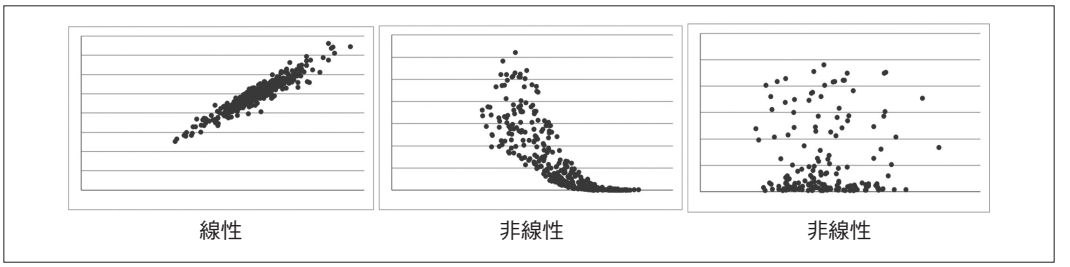


圖 4-2　線性 vs. 非線性關係

63 另外，還存在負相關（*negative correlation*），即我們可以用　條負向的直線總結變數的關係；如果可以用水平線概括變數之間的關係，則表示「零相關」（zero correlation）。圖 4 3 展示了這些各有不同的線性關係。請記住，變數必須具有線性關係，才能加以計算其相關係數。

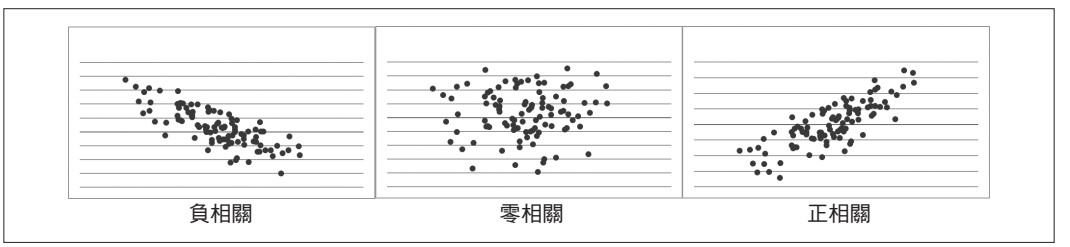


圖 4-3　負相關、零相關和正相關

旦確定了資料具有線性關係，我們就能計算相關係數。相關係數的值永遠介於 –1 ～ 1 之間，–1 表示完美的負線性關係，1 表示完美的正線性關係，0 表示完全沒有線性關係。表 4 1 列出了解釋相關係數強度的經驗法則。這並不是放諸四海皆準的檢驗標準，但適合做為詮釋相關性強弱的切入點。

表 4 1　對相關係數的解釋

相關係數	解釋
1.0	完美負相關的線性關係
0.7	強負相關
0.5	中負相關
0.3	弱負相關
0	零相關的線性關係

第十章

Excel 使用者開始使用 Python 的第一步

Python 是由吉多·范羅蘇姆（Guido van Rossum）在 1991 年開發的程式設計語言。和 R 一樣，Python 是免費、開源的程式語言。當時，之所以選用 Python 作為程式語言的名字，是因為他是 BBC 電視劇「*Monty Python's Flying Circus*」（蒙提・派森的飛行馬戲團）的劇迷。與專門用於資料分析的 R 不同，Python 的設計理念是讓該語言廣泛通用於各種場景，適合開發與系統互動、處理執行錯誤等工作。這一點對於 Python 語言如何「思考」和處理資料有著重要影響。例如，第 7 章中曾經提過，R 具備一個內建的「表格式」資料結構。在 Python 中並非如此；我們需要更加依賴外部套件來處理資料。

Python 和 R 一樣，擁有成千上萬個套件，由非常活躍的貢獻者社群持續維護。你會發現從 App 開發到嵌入式裝置，再到資料分析，以 Python 編寫的成果無所不在。Python 的使用者群體愈發多元，與日增長茁壯，它已經成為世界上最受歡迎的程式設計語言之一，不僅可在資料分析領域發揮特長，還適用於廣泛的運算目的。

Python 是一種通用的程式設計語言，而 R 是專門為統計分析而生的程式語言。

下載 Python

Python 軟體基金會（*https://python.org*）負責維護「官方」的 Python 原始碼。因為 Python 是開源的，所以任何人都可以取用、新增和重新發布 Python 程式碼。Anaconda 就是一種 Python 的發行版本，也是本書推薦的安裝版本。它由一家同名的營利組織進
162 行維護，並提供付費選項；我們將使用免費的個人版。Python 現在已經有了第三個版本，也就是 Python 3。你可以在 Anaconda 的網站（*https://oreil.ly/3RYeQ*）上下載 Python 3 的最新版本。

Python 2 和 Python 3

2008 年發布的 Python 3 版本，它對語言做了較大修訂而不能完全後向相容。這意味著為 Python 2 編寫的程式碼不一定能在 Python 3 上運行，反之亦然。在本書撰寫時，Python 2 已經正式退役，雖然如此，在你日後的 Python 之旅中，仍可能遇到一些 Python 2 時代的參照和程式碼殘留。

除了 Python 的簡化安裝版本之外，Anaconda 還附帶了額外的東西，包括一些我們將在本書後面使用的主流套件。它還附帶了一個 web 應用程式，我們將使用它來編寫 Python 程式：Jupyter Notebook。

開始使用 Jupyter

如第 6 章所述，R 語言傳承自 S 語言，是一種為執行 EDA（探索式資料分析）而設計的程式語言。由於 EDA 具有迭代性質，R 語言的預期工作流是執行選定的程式碼區段，並對輸出結果進行探索。這使得直接從 R 腳本（*.r* 格式）中進行資料分析變得很容易。RStudio IDE 還為 R 程式的編寫工作提供了額外支援，例如顯示說明文件和環境中物件內容的專用分割視窗。

相比之下，Python 在某些方面更像是一種「低層次」的程式設計語言，程式碼需要先編譯成機器可讀的檔案後才能被執行。這可能會使從 Python 腳本（*.py* 格式）中進行零碎的資料分析變得相對困難。在統計計算或更廣義的科學計算的應用上，這一痛點吸引了物理學家暨軟體開發人員 Fernando Pérez 的注意，他與同事們在 2001 年發起 IPython 專案，旨在為 Python 製作一個更具互動性的編譯器（IPython 是「interactive Python」

的簡稱）。這項專案最後發明了　種新的檔案類型：*IPython Notebook*，其檔案格式為 *.ipynb*。

2014 年，Fernando Pérez 宣布從 IPython 中衍生出　個名為 Jupyter 的專案。這是　個語言無關的專案，透過開發開源軟體，支援所有程式語言之間的互動式資料科學和科學計算。於是，IPython Notebook 成為了 Jupyter Notebook，保留了 *.ipynb* 檔案格式。Jupyter Notebook 是　個互動式 web 應用，使用者可以在這個計算環境中，將程式碼與文字、公式等內容結合起來，設計出媒體豐富的互動式檔案。事實上，Jupyter 專案的名稱是致敬伽利略筆下記錄木星之衛星的筆記本。所謂的**核心**（*kernel*）是 Jupyter 執行筆記本中程式碼的背景執行程式。下載 Anaconda 發行版後，你也同時設置好從 Jupyter Notebook 執行 Python 的所有必要條件：現在，你只需要啟動　個工作階段，就能開始編寫程式。

163

RStudio、Jupyter Notebook 和其他程式編寫方式

暫時告別 RStudio，學習另　個使用者介面，可能會讓你感到不適應。不過，請記住，在開源世界中，程式碼和應用程式通常是脫鉤的；使用者可以按照喜好「混搭」這些語言和平台。比方說，R 是 Jupyter 的幾十種核心語言之　，你當然可以使用 R 在 Jupyter Notebook 上編寫程式。除了致敬伽利略發現的土星衛星之外，它的名稱還綜合了 Jupyter 支援的三種核心程式語言：Julia、Python 和 R。

在 R 的 `reticulate` 套件的幫助下，使用者也可以從 RStudio 中執行 Python 腳本，也能更廣泛地從 R 執行 Python 程式碼。這意味著，我們可以在 Python 中匯入和處理資料，然後使用 R 將分析結果視覺化呈現。可處理 Python 程式碼的其他主流應用包括 PyCharm 和 Visual Studio Code。RStudio 也有自己的筆記本應用程式：R Notebooks，同樣採用和 Jupyter 相同的將程式碼和文字穿插呈現的概念，它支援包含 R 和 Python 在內的多種程式語言。

正如你所見，這個世界上有　系列可以用 R 和 Python 編寫程式的豐富工具，數量族繁不及備載。本書將焦點放在 RStudio 的 R 腳本和 Jupyter Notebooks 的 Python 腳本，因為這兩者比起其他工具，相對來說更加適合初學者，也更常見。　旦你適應了這些工作流後，不妨在網路上搜尋此處提到的其他開發環境。隨著你持續學習，你將會學到更多與這些語言互動的方法。

第十四章

總結與展望

在本書開篇序言中，我寫下了這個學習目標：

> 閱讀完這本書之後，你將能夠
> 「使用程式語言執行探索式資料分析與假說檢定」。

我真誠地希望你感到自己實現了這一目標，並且建立了足夠的自信心，勇敢探索資料分析的廣闊領域。作為本次分析學習之旅的尾聲，我想分享一些延伸主題，幫助你持續吸收更多知識。

分析堆疊的更多拼圖

第 5 章介紹了資料分析領域的四大軟體應用：電子試算表、程式設計語言、資料庫和商業智慧工具。因為本書主題是介紹基於統計的分析元素，我們把焦點放在這個分析堆疊的前兩塊拼圖中。讀者不妨重新翻閱這一章節，了解這些拼圖的作用，以及它們如何相輔相成。

研究設計與業務實驗

你在第 3 章中了解到，想要打造堅實的資料分析成果，首先要從堅實的「資料收集」步驟開始。有一句話是這麼說的：「垃圾左手進，垃圾右手出。」（Garbage in, garbage out.）在本書中，我們的假設前提是：我們收集了準確的資料，這些是符合分析目的的正確資料，並且包含一個代表性樣本。因為我們使用的是經過同行審閱（peer reviewed）的知名資料集，所以我們可以認為，這個假設前提是相對安全的。

然而，你通常無法對手上的資料如此肯定；你可能身兼多職，要負責收集資料，**並且**分析資料。那麼，你值得多加鑽研**研究設計**（*research design*）和**研究方法**（*research methods*）。這個領域的內容可能相當複雜且相對學術，但它確實被實際應用到了商業實驗領域。你可以翻閱 Stefan H. Thomke 的《*Experimentation Works: The Surprising Power of Business Experiments*》（Harvard Business Review），了解將堅實、符合邏輯的研究方法應用到商業實踐的方法及原因。

更多的統計方法

正如第 4 章所提到的，我們只觸及了眾多統計檢定的冰山　角，其中很大　部分的統計檢定，都採用了和第 3 章相同的假設測試框架。

如果讀者想要認識其他統計方法的概念，請參考 Sarah Boslaugh 所寫的《*Statistics in a Nutshell*》（O'Reilly）。不妨再搭配 Peter Bruce 等人所寫的《*Practical Statistics for Data Scientists*, 2nd Edition》（O'Reilly）　起閱讀，使用 R 和 Python 應用這些概念。正如後者書名所示，這本書跨越統計學和資料科學之間的界限，為兩者搭起了橋梁，繁體中文版《**資料科學家的實用統計學**》由碁峰資訊出版。

資料科學與機器學習

第 5 章回顧了統計學、資料分析和資料科學之間的差異，並作出以下總結：儘管方法上存在差異，但這些領域之間擁有極高的相似性。

如果你對資料科學和機器學習非常感興趣，請將你的學習精力集中在 R 和 Python 上，同時掌握　些 SQL 和資料庫知識。如果你想了解 R 如何應用在資料科學領域，請參考 Hadley Wickham 和 Garrett Grolemund 所寫的《*R for Data Science*》（O'Reilly），繁體中文版《*R* **資料科學**》由碁峰資訊出版。如果想使用 Python，那麼參考 Aurélien Géron 所寫的《*Hands On Machine Learning with Scikit Learn, Keras, and TensorFlow*, 2/e》（O'Reilly），繁體中文版《**精通機器學習｜使用** *Scikit Learn, Keras* **與** *TensorFlow* **第二版**》由碁峰資訊出版。

版本控制

第 5 章還提到了可重現性（reproducibility）的重要性。讓我們來認識在這概念中扮演關鍵角色的　個應用。你以前可能看過類似這樣的　組檔案：

- *proposal.txt*
- *proposal v2.txt*
- *proposal Feb23.txt*
- *proposal final.txt*
- *proposal FINAL final.txt*

也許　個使用者建立了 *proposal v2.txt*，另　個人則建立了 *proposal Feb23.txt*。然後又出現了 *proposal final.txt* 和 *proposal FINAL final.txt*，我們還得釐清這兩者之間的差異。這樣的　組檔案，讓人很難確認究竟哪　個檔案是「主要」副本，以及如何重建和遷移所有變更內容到這份副本上，並且記錄哪些人對此副本做出了貢獻。

215 這時，**版本控制系統**（*version control system*）可以拯救我們。這是　種按照時間先後追蹤專案變化的方式，例如它可以紀錄來自不同使用者所做的貢獻和修改內容。版本控制系統顛覆了過去的協作和追蹤修訂工作，讓　切變得截然不同，不過它的學習曲線相對陡峭，需要多花　些時間才能上手。

Git 是　個主流的版本控制系統，在資料科學家、軟體工程師和其他技術專業人士之間被廣泛使用。這些人經常使用 GitHub 這個雲端託管服務來管理 Git 專案。關於 Git 和 GitHub 的介紹，請參考 Jon Loeliger 與 Matthew McCullough 合著的《*Version Control with Git*, 2nd edition》（O'Reilly）《版本控制使用 Git》。如果想要了解將 Git 和 GitHub 與 R 和 RStudio 配對的方法，請參考 Jenny Bryan 等人整理的線上參考資源 *Happy Git and GitHub for the useR*（*https://happygitwithr.com*）。儘管目前 Git 和其他版本控制系統在資料分析的工作流中還不是太常見，由於資料分析工作對於可重現性的需求不斷增長，這些版本控制工具越來越受到人們歡迎。