前言

本書的主要目標是描述一個資料視覺化（dataviz）工具鏈，而該工具鏈在網際網路時代開始占據主導地位。此工具鏈的指導原則是，無論您設法從資料中挖掘出什麼有見地的金塊，都應該要在 Web 瀏覽器上找到一個家。放在網路上意味著，您可以透過使用身分驗證或限制，為本地網路輕鬆地選擇，資料視覺化發送給選定的少數人或整個世界。這是網際網路的偉大理念，也是 dataviz 正在快速擁抱的理念。這意味著 dataviz 的未來涉及到 JavaScript，而那是 Web 瀏覽器唯一的一流語言。但是 JavaScript 還沒有精煉原始資料所需的資料處理堆疊，這意味著資料視覺化不可避免地需要運用多個語言。我希望這本書能支援我的信念，也就是 Python 是 JavaScript 壟斷瀏覽器視覺化時自然而然的補充語言。

儘管這本書很大本（作者現在強烈地感受到了這一事實），但它必須非常有選擇性，省去很多酷到不行的 Python 和 JavaScript 的 dataviz 工具，並專注於提供最好的工具積木。那些我無法涵蓋的有用程式庫數量，反映了 Python 和 JavaScript 資料科學生態系統的巨大活力。即使在編寫本書的同時，出色的新 Python 和 JavaScript 程式庫也一直持續地出現，並且持續前進。

所有資料視覺化本質上都是變革性的，展示了從資料集（HTML 表格和列表）的一種反射到更現代、更具吸引力、互動性、並且在根本上是基於瀏覽器的資料集的反射間的轉變，提供了一種在工作環境中引入關鍵資料視覺化工具的好方法。這裡的挑戰在於將基本的諾貝爾獎得主維基百科列表，轉換為現代的、互動式的、基於瀏覽器的視覺化。因此，相同的資料集會以更易於存取、更具吸引力的形式來呈現。

從未處理的資料，到相當豐富的、使用者驅動的視覺化旅程，為最佳工具的選擇提供了一些資訊。首先，我們需要獲取資料集。這通常是由同事或客戶提供的，但為了增加挑戰並在此過程中學習一些非常重要的資料視覺化技能，我們會學習如何使用 Python

強大的 Scrapy 程式庫，來從網路，也就是維基百科的諾貝爾獎頁面中爬取（*scrape*）資料集。然後，這個未處理的資料集需要被改進和探索，要進行這些事沒有比 Python 的 pandas 更好的生態系統了。隨著 Matplotlib 的支援和由 Jupyter notebook 的驅動，pandas 正在成為這種取證資料工作的黃金標準。透過使用 SQLAlchemy 和 SQLLite 到 SQL 儲存和探索乾淨的資料，可以將精心挑選的資料故事視覺化。我推薦使用 Matplotlib 和 Plotly，來將靜態和動態圖表從 Python 嵌入到網頁中。但對於更雄心勃勃的任務來說，網路的最高資料視覺化程式庫是基於 JavaScript 的 D3，我們涵蓋了 D3 的基本要素，同時使用它們來製作諾貝爾獎資料視覺化展示品。

這本書是一系列工具的集合，形成了一個串鏈，而諾貝爾獎視覺化的建立提供了指導性的敘述。您應該能夠在需要時深入閱讀相關章節；本書的不同部分是獨立的，因此可以在需要時快速複習所學的內容。

本書分為五個部分。第一部分介紹基本的 Python 和 JavaScript dataviz 工具包，接下來的四部分展示如何檢索原始資料、清理資料、探索資料，最後將其轉換為現代的 Web 視覺化。現在就來總結一下每個部分的主要課程內容。

第一部分：基本工具包

基本工具包包括：

- Python 和 JavaScript 之間的語言學習橋梁。這是要讓兩種語言之間的過渡更為平順、突顯它們的許多相似之處、並為現代資料視覺化的雙語過程奠定基礎。隨著最新 JavaScript 的出現，[1] Python 和 JavaScript 的共同點越來越多，大幅降低它們之間切換的壓力 。
- 能夠輕鬆讀取和寫入關鍵資料格式例如 JSON 和 CSV，和資料庫 SQL 和 NoSQL 是 Python 的一大優勢，在 Python 中傳遞資料、轉換格式和更改資料庫是如此容易，資料的這種流暢移動是任何資料視覺化工具鏈的主要潤滑劑。
- 涵蓋開始製作現代的、互動式的、基於瀏覽器的資料視覺化所需的基本網路開發（webdev）技能。透過專注於單頁應用程式（*https://oreil.ly/yqv6C*）的概念，而非建構整個網站，能夠最小化傳統的 webdev，並將重點放在用 JavaScript 來編寫視覺化創作。可縮放向量圖形（Scalable Vector Graphics, SVG）的介紹是 D3 視覺化的主要積木，為第五部分建立的諾貝爾獎視覺化奠定了基礎。

1 基於 ECMAScript（*https://oreil.ly/0uwuN*）的 JavaScript 有很多版本，但提供大量新功能的最重要版本是 ECMAScript 6（*https://oreil.ly/owrsZ*）。

第二部分：獲取資料

本書的這一部分將研究如何使用 Python 從 Web 獲取資料，假設還沒有為資料視覺化工具提供一個漂亮、乾淨的資料檔案：

- 如果您幸運的話，放在易於使用的資料格式（即 JSON 或 CSV）的乾淨檔案會位於一個開放的 URL 中，只要一個簡單的 HTTP 請求就可以取得。或者，您的資料集可能會有一個專用的 Web API，運氣好的話會是 RESTful API，例如，看一下如何使用 Twitter API（透過 Python 的 Tweepy 程式庫）；我們還將瞭解如何使用 Google 試算表（Google spreadsheet），這是在 dataviz 中廣泛使用的資料共享資源。
- 當我們感興趣的資料被以人類可讀的形式出現在網路上時（通常以 HTML 表格、列表或分層內容塊的形式出現），事情就會變得更加複雜。在這種情況下，您必須求助於**爬取**（*scraping*）來獲取原始 HTML 內容，然後使用剖析器（parser）讓它嵌入的內容可用。我們將瞭解如何使用 Python 的輕量級 Beautiful Soup 爬取程式庫，以及功能更強大、更重量級的 Scrapy，它是 Python 爬取領域中的大明星。

第三部分：使用 pandas 來清理和探索資料

這一部分將使用 Python 強大的程式化試算表 pandas 這管大砲，來解決清理和探索資料集的問題。首先會瞭解 pandas 如何成為 Python NumPy 生態系統的一部分，NumPy 利用了非常快速且強大的低階陣列處理程式庫的力量，同時又使它們易於存取。這裡的焦點是使用 pandas 清理，然後探索諾貝爾獎資料集：

- 大多數資料，即使是來自官方網路 API 的資料，都是骯髒的資料。讓它變得乾淨和可用所占用您作為資料視覺化人員的時間，將比預期多很多。以諾貝爾獎資料集為例，我們在開始探索前會逐步地清理它、搜尋不可靠的日期、異常的資料型別、漏失的欄位，以及所有需要清理的常見污垢，然後將資料轉換為視覺化。
- 有了盡力所能做到的乾淨諾貝爾獎資料集之後，就來看看如何使用 pandas 和 Matplotlib，以互動方式探索資料、輕鬆建立內聯圖表、以任何方式切片資料以及大致上判讀是多麼容易，並在其中尋找您想要透過視覺化提供的那些有趣金塊。

第四部分：交付資料

在這一部分，將看出使用 Flask 建立最小資料 API，以靜態和動態方式向 Web 瀏覽器傳送資料有多麼容易：

首先，瞭解如何使用 Flask 來提供靜態檔案，然後介紹如何推出您自己的基本資料 API，以從本地資料庫提供資料。Flask 的極簡主義讓您可以在 Python 資料處理的成果，以及它們最終在瀏覽器上的視覺化之間，建立一個非常薄的資料服務層。

開源軟體的成就在於，您經常可以找到強固、易於使用的程式庫，它們比您更能解決您的問題。在本部分的第二章中，將看到使用同類最佳的 Python（Flask）程式庫來製作一個強固、靈活的 RESTful API 是多麼容易，來為您的線上資料做好準備；還會介紹使用 Python 愛好者的最愛 Heroku，輕鬆地線上部署此資料伺服器。

第五部分：使用 D3 和 Plotly 來視覺化您的資料

在本部分的第一章中，將瞭解如何以圖表或地圖的形式來獲取以 pandas 來驅動探索的成果，並將它們放在它們所歸屬的網路上。Matplotlib 可以產生出版標準的靜態圖表，而 Plotly 則將使用者控制項和動態圖表帶到表格中。我們將瞭解如何直接從 Jupyter notebook 中獲取 Plotly 圖表，並將其放入網頁中。

本書中涵蓋的 D3 部分是最具挑戰性的部分，但您很可能最終會利用它來建構所產生的那類多元素視覺化。D3 的樂趣之一是大量的範例（*https://oreil.ly/AIWkI*），這些範例很容易在網上找到，但大多數範例只示範一種技術，很少顯示如何編排多個視覺元素。在 D3 章節中，會看到如何在使用者過濾諾貝爾獎資料集，或更改獲獎度量（絕對值或人均值）時同步更新時間軸（包含所有諾貝爾獎）、地圖、直條圖和列表。

掌握這些章節中所展示的核心主題應該可以讓您放開想像力，邊做邊學；我建議選擇一些您打從心裡喜歡的資料，並以此設計一個 D3 作品 。

第二版

當 O'Reilly 請我寫這本書的第二版時，我有點不情願。第一版最終比預期的要厚，要更新和擴充它可能需要大量工作。然而，在審查所涵蓋的程式庫狀態以及 Python 和 JavaScript dataviz 生態系統的變化之後，很明顯的大多數使用的程式庫（例如 Scrapy、NumPy、pandas）仍然是可靠的選擇，只需要少部分更新。

D3 是變化最大的程式庫，但這些變化讓 D3 更易於使用和教學。JavaScript 模組也穩固地就定位，使程式碼更清晰，更適合 Python 愛好者（Pythonista）。

一些 Python 程式庫似乎不再是可靠的選擇，並且有幾個已被棄用。第一版相當廣泛地討論了 MongoDB，那是一種 NoSQL 資料庫。我現在認為，好的老式 SQL 更適合資料視覺化工作，如果需要資料庫，最小的基於檔案的無伺服器 SQLite 就是資料視覺化的最佳選擇。

與其用另一個 Python 程式庫來替換已棄用的 RESTful 資料伺服器，從頭開始建構一個簡單的伺服器會更具啟發性，在其中展示一些出色的 Python 程式庫使用，例如 marshmallow，它們在許多資料視覺化場景中很有用。

有了更新本書的時間，我決定使用第一本書的資料集來示範如何使用 Matplotlib 和 pandas，以探索和分析，重點是將所有程式庫更新到當前（截至 2022 年年中）的版本。這讓您可以把時間花在新內容上，其中會有一章專門介紹 Python 的 Plotly 程式庫，它讓您可以輕鬆地將探索性工作從 Jupyter notebook 轉移到具有使用者互動的 Web 簡報；這種方法的一個特別優勢是 Mapbox 地圖的可用性，這是一個豐富的地圖生態系統。

第二版的主旨是：

- 更新所有程式庫。
- 刪除和 / 或替換禁不起時間考驗的程式庫。
- 添加一些 Python 和 JavaScript dataviz 快速發展世界中的變化所建議的新素材。

我認為，dataviz 工具鏈的比喻仍然適用，從原始、未處理的 web 資料，到探索性的 dataviz 驅動分析，再到完善的 web 視覺化轉換生產線，仍然是學習這項工作關鍵工具的好方法。

本書編排慣例

本書使用以下排版慣例：

斜體字（*Italic*）

表示新的術語、URL、電子郵件地址、檔名和延伸檔名。（中文使用楷體字）

定寬字（`Constant width`）

用於程式列表，以及在段落中參照的程式元素，例如變數或函數名稱、資料庫、資料型別、環境變數、敘述和關鍵字。

定寬粗體字（**`Constant width bold`**）

概論

本書旨在讓您快速瞭解在我看來最強大的資料視覺化堆疊：Python 和 JavaScript。您將會對 pandas 和 D3 等大型程式庫有足夠瞭解，可以開始製作自己的 Web 資料視覺化並完善自己的工具鏈。實務會帶來專業知識，但就基本能力的學習來說，本書內容很淺。

如果您正在閱讀本文，我很想聽聽您的任何回饋。請將其寄到 *pyjsdataviz@kyrandale.com*。多謝。

您還可以在我的網站（*https://www.kyrandale.com/viz/nobel_viz_v2*）上找到本書所建構的諾貝爾獎視覺化的完整工作副本。

本書大部分內容講述其中一些數不清的資料視覺化故事，精心挑選來展示一些強大的 Python 和 JavaScript 程式庫和工具，共同構成一個工具鏈。該工具鏈在開始時會蒐集原始的、未經精煉的資料，並在結束時提供豐富的、引人入勝的 Web 視覺化。就像所有資料視覺化的故事一樣，這是一個有關轉換的故事——在此案例中，是將諾貝爾獎得主的基本維基百科列表轉變為互動式視覺化、使資料栩栩如生、讓探索諾貝爾獎的歷史簡單而有趣。

無論您擁有什麼資料，也無論您想用它來講述什麼故事，把它轉化為視覺化效果的最原始、自然方法就是網路。作為一個交付平台，它比以前的平台強大了幾個數量級，本書旨在使從基於桌上型電腦或伺服器的資料分析和處理，能夠順利過渡到 Web。

使用這兩種強大的語言，不僅可以提供強大的 Web 視覺化效果，而且還很有趣且引人入勝。

許多潛在的 dataviz 程式設計師，認為 *web* 開發和做他們想做的事情，也就是用 Python 和 JavaScript 來編寫程式之間存在很大鴻溝。Web 開發涉及大量關於標記語言（markup language）、樣式腳本（style script）和管理的神祕知識，並且如果沒有 *Webpack* 或 *Gulp* 等名稱奇怪的工具是無法完成的。如今，這個巨大的鴻溝可以縮小為一層易滲透的薄膜，讓您可以專注於自己擅長的事情：以最小的努力來編寫程式（見圖 I-1），並將 Web 伺服器委託給資料交付過程。

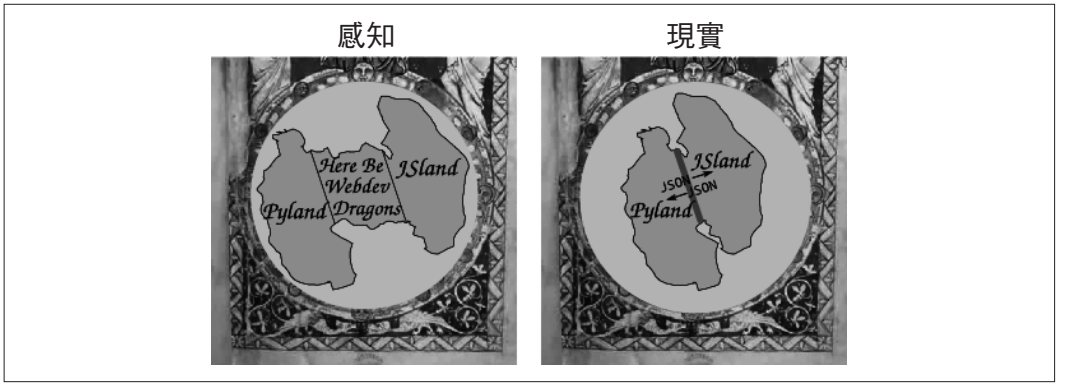

圖 I-1　以上是 webdev 惡龍

本書適合誰

首先，本書適合任何對 Python 或 JavaScript 有一定瞭解的人，他們想要探索資料處理生態系統中最激動人心的領域之一：網路資料視覺化的爆炸式增長領域。它還涉及解決一些具體的挑戰，而根據我的經驗，這些挑戰很常見。

當您接受委託編寫一本技術書籍時，您的編輯很可能會明智地提醒您要去考慮書中可以解決的痛點（*pain point*）。本書的兩個關鍵痛點可以透過幾個故事來完整詮釋，其中一個是我自己的故事，另一個是我認識的許多 JavaScript 開發者用各種形式告訴我的。

多年前，作為一名學術研究人員，我接觸到 Python 並愛上了它。我一直在用 C++ 編寫一些相當複雜的模擬，而 Python 的簡單性和強大功能讓所有樣版（boilerplate）Makefile、宣告、定義等都煥然一新，程式設計變得更加有趣。Python 是完美的黏合劑，可以有效地和 C++ 程式庫配合使用（Python 當時不是，現在也不是速度惡魔），並且非常輕鬆地完成所有在低階語言中讓人痛苦的事情（例如，檔案 I/O、資料庫存取和序列化）。我開始用 Python 編寫所有圖形使用者介面（graphical user interface, GUI）和

視覺化，並使用了 wxPython、PyQt 和一大堆其他令人耳目一新的簡單工具集。不幸的是，雖然其中一些工具非常酷，而且我很樂意和全世界分享它們，但封裝、發布它們以及確保它們仍然適用於現代程式庫所需的努力是我不太可能克服的障礙。

當時存在著理論上完美的通用發布系統，適用於我精心製作的軟體——即網頁瀏覽器。曾經，其實現在也是，地球上幾乎每台電腦都可用網頁瀏覽器，它們具有自己的內建直譯式程式語言：編寫一次，隨處執行。但 Python 並沒有在網頁瀏覽器的沙坑中發揮作用，瀏覽器無法實現雄心勃勃的圖形和視覺化，幾乎僅限於靜態影像和奇怪的 jQuery 轉換（*https://jquery.com*）。JavaScript 是一種「玩具」語言，綁定了一個非常慢的直譯器（interpreter），這對小型 DOM（*https://oreil.ly/QnE0a*）技巧很有用，但肯定無法接近我在桌面上使用 Python 可以做的事情。所以這條路線遭到漠視且失控，我的視覺化效果想放在網路上，但沒有通往它的途徑。

快轉十年左右，由於 Google 及其 V8 引擎發起的軍備競賽，JavaScript 現在快了幾個數量級；事實上，它現在比 Python 快得多[1]。HTML 也以 HTML5 的名義對其行為進行了一些整理。讓它使用起來更好，樣版程式碼更少。強大的視覺化程式庫，尤其是 D3 等，已經鞏固了可縮放向量圖形（Scalable Vector Graphics, SVG）等鬆散遵循且明顯不穩定的協議。現代瀏覽器必須與 SVG 以及越來越多的表達為 WebGL 及其子項（如 THREE.js）形式的 D3 完全搭配以工作。我在 Python 中進行的視覺化，現在可以在您的本地 Web 瀏覽器上實現，而且回報是，只需很少的努力，世界上的每台桌上型電腦、筆記型電腦、智慧型手機和平板電腦都可以存取它們。

那麼，為什麼 Python 愛好者不會蜂擁而至來，以他們指定的形式獲取資料呢？畢竟，除了自己製作之外，另一種選擇是將它留給其他人，而我認識的大多數資料科學家會發現這種作法很不理想。好吧，首先是 *web* 開發（*web development*）這個術語意味著複雜的標記、不透明的樣式表、一大堆需要學習的新工具和需要掌握的 IDE。然後是 JavaScript 本身，它是一種奇怪的語言，直到最近才認定它不只是一個有些不倫不類的玩具。我的目標是直接面對這些痛點，並展示您可以使用極少量的 HTML 和 CSS 樣版來製作現代 Web 視覺化（通常是單頁應用程式），讓您專注於程式設計，並讓 Python 愛好者輕鬆飛躍到 JavaScript。但您不必真的跳躍；第 2 章是一座語言橋梁，旨在透過突顯共同元素和提供簡單翻譯，來幫助 Python 愛好者和 JavaScript 人員去彌合語言之間的鴻溝。

1 請參閱 Benchmarks Game 網站（*https://oreil.ly/z6T6R*）所進行相當驚人的比較。

第二個故事在我認識的 JavaScript 資料視覺化人員中很普遍。用 JavaScript 來處理資料很不理想，它幾乎沒有重量級的程式庫，儘管最近對該語言的功能增強使資料處理變得更加愉快，但仍然沒有真正的資料處理生態系統可言。因此，在可用的極其強大視覺化程式庫，如 D3 這一如既往的最重要程式庫；與用來清理和處理傳遞給瀏覽器任何資料的能力之間，存在著明顯的不對稱。所有這些都要求您使用另一種語言，或像 Tableau 這樣的工具包清理、處理和探索資料，這通常會演變成零碎地嘗試已漸被遺忘的 Matlab、具有陡峭學習曲線的 R、或一兩個 Java 程式庫。

像 Tableau（*https://www.tableau.com*）這樣的工具包雖然非常令人印象深刻，但最終通常會讓程式設計師感到沮喪，在 GUI 中無法複製良好的通用程式設計語言的表達能力。另外，如果您想建立一個小型網路伺服器來傳送處理過的資料時該怎麼辦？這意味著至少要學習一種新的網路開發語言。

換句話說，開始擴展資料視覺化的 JavaScript 開發人員正在尋找一種互補的資料處理堆疊，僅需要投入一點點時間，而且學習內容並不深奧。

使用本書的最低要求

我一直不願意限制人們的探索力，如果學術界的殿堂落後於整個趨勢達光年之久，又該如何學習？特別在是這個充滿自學成才、學習速度如飢似渴、能抬頭挺胸，不受過去適用於學習模式所約束的程式設計和網路環境中。Python 和 JavaScript 在程式設計語言方面非常簡單，並且都是最佳第一語言的首選。解釋程式碼不會有很大的認知負擔。

本著這種精神，即使沒有任何 Python 和 JavaScript 經驗的專業程式設計師也可以閱讀本書，並在一週內編寫自定義程式庫。

對於剛接觸 Python 或 JavaScript 的初學者來說，這本書對您來說可能太進階了，我建議您利用當今大量的書籍、網路資源、螢幕錄影（screencast）等，它們讓學習變得如此容易。專注於個人的渴望、您想解決的問題、邊做邊學程式設計——這是唯一的方法。

對於使用 Python 或 JavaScript 進行過一些程式設計的人，我建議的入門門檻是您已經一起使用過幾個程式庫、瞭解您的語言的基本慣用語、看一段新穎的程式碼，就可以大概瞭解正在發生的事情——換句話說，就是可以使用標準程式庫的幾個模組的 Python 愛好者，以及使用過奇怪程式庫並能理解幾行原始碼的 JavaScript 人員。

為什麼選擇 Python 和 JavaScript ？

為什麼選 JavaScript 是一個容易回答的問題。現在和可預見的未來，只有一種一流的、基於瀏覽器的程式設計語言。即使已經出現各種進行擴展、擴增和篡奪的嘗試，但是老式的、普通的 JS 仍然是卓越的。如果您想製作現代的、動態的、互動式的視覺化效果，並希望按一下按鈕就可以它們交付給世界，那麼您必然會在某些時候遇到 JavaScript。您可能不需要精通，但基本能力是進入現代資料科學最令人興奮領域之一的基本代價。這本書將帶您進場。

為什麼不在瀏覽器中使用 Python ？

最近出現了在瀏覽器中執行有限版本 Python 的計畫。例如，Pyodide（*https://github.com/pyodide/pyodide*）是 CPython 到 WebAssembly 的轉接口，它們令人印象深刻且有趣，但目前在 Python 中製作 Web 圖表的主要方法是，讓它們由中介程式庫來自動轉換。

還有一些非常令人印象深刻的倡議 ，想讓 Python 產生的視覺化效果（通常建構在 Matplotlib（*https://matplotlib.org*）上）在瀏覽器中執行。它們透過使用基於 `canvas` 或 `svg` 的繪圖上下文，將 Python 程式碼轉換為 JavaScript 來達成這一點。其中最受歡迎和最成熟的是 Plotly（*https://plot.ly*）和 Bokeh（*https://bokeh.pydata.org/en/latest*）。在第 14 章中，您將看到如何使用 Plotly 在 Jupyter notebook 中產生圖表並將它們傳輸到網頁。對於許多使用案例來說，這是一個很棒的資料視覺化工具，可以放在您的工具箱中。

雖然這些 JavaScript 轉換器和許多可靠使用案例背後有一些出色的程式設計，但它們的確也有很大的局限性：

- 自動程式碼轉換可以完成這項工作沒有問題，但產生的程式碼通常對人類來說是非常難以理解的。
- 使用功能強大、基於瀏覽器的 JavaScript 開發環境來調整和客製化已產生的圖可能會很痛苦，第 14 章中將會看到如何使用 Plotly 的 JS API 來減輕這種痛苦。
- 您僅限於程式庫中當前可用的繪圖類型的子集合。
- 目前的互動性非常基本。客製化使用者控制項最好使用瀏覽器的開發人員工具來在 JavaScript 中完成。

請記住，建構這些程式庫的人一定是 JavaScript 專家，所以如果您想瞭解他們正在做的任何事情並最終以此表達自我，那麼您必須從頭開始學習一些 JavaScript。

為什麼使用 Python 進行資料處理？

為什麼您應該選擇 Python 來滿足資料處理需求？這有點複雜。首先，就資料處理而言還有幾個其他選擇。讓我們來談談這分工作的幾個候選人，從企業巨頭 Java 開始。

Java

在其他主要的通用程式設計語言中，只有 Java 提供與 Python 一樣豐富的程式庫生態系統，而且原生速度也快得多。但是，雖然 Java 比 C++ 之類的語言更容易進行程式設計，但在我看來，它並不是一種特別好的程式設計語言，因為它有太多乏味的樣版程式碼和過多的廢話。經過一段時間後，這種事情會開始變得沉重，並對程式碼造成了阻礙。至於速度，Python 的預設直譯器很慢，但 Python 是一種很好的膠合語言，和其他語言配合地很好。NumPy（以及它依賴的 pandas）、SciPy 等大型 Python 資料處理程式庫展示了這種能力，它們使用 C++ 和 Fortran 程式庫來完成繁重的工作，同時提供簡單腳本語言的易用性。

R

直到最近，受人尊敬的 R 一直是許多資料科學家的首選工具，並且可能是 Python 在該領域的主要競爭對手。與 Python 一樣，R 受益於非常活躍的社群、繪圖程式庫 ggplot2 等一些出色的工具，以及專為資料科學和統計所設計的語法。但這種專長是一把雙面刃。因為 R 是為特定目的而開發的，這意味著如果您希望編寫一個 Web 伺服器為您用 R 處理後的資料提供服務時，您必須跳到另一種語言以及所有隨之而來的學習額外負擔，或者嘗試把方釘擠進圓孔般的硬拗，把一些東西拼湊在一起。Python 的通用性質及其豐富的生態系統，意味著人們可以在不離開舒適區的情況下，完成資料處理生產線所需的幾乎所有事情（除了 JS 視覺效果）。我認為，為語法上的笨拙付出代價只是一個小小的犧牲。

其他

除了使用 Python 進行資料處理之外，還有其他替代方案，但它們都無法與具有豐富程式庫生態系統的通用、易於使用的程式設計語言所提供的靈活性和功能相提並論。例如，Matlab 和 Mathematica 等數學程式設計環境擁有活躍的社群和大量優秀的程式庫，但它們很難普遍化，因為它們一開始就設計來用在比較封閉的領域。它們也是專有的，這意味著大量的初始投資以及和 Python 響亮的開源環境不同的氛圍。

像 Tableau（*https://www.tableau.com*）這樣由 GUI 驅動的資料視覺化工具是偉大的產物，但很快就會讓習慣於自由程式設計的人感到沮喪。如果您只唱它們歌單中的歌，它們往往會工作地很好，偏離指定路徑時很快就會很痛苦。

Python 一直在改進

就目前情況而言，我認為可以說 Python 就是新興資料科學家的首選語言，但它也沒有因此而停滯不前。就目前情況而言，我認為可以說 Python 就是新興資料科學家的首選語言，但它也沒有因此而停滯不前。事實上，Python 在這方面的能力正在以驚人速度增長。換句話說，我已經使用 Python 程式設計 20 多年了，如果找不到 Python 模組來幫助解決手頭問題時我會很驚訝，但我發現自己對 Python 資料處理能力的發展感到驚訝，因為每週都會出現一個新的、強大的程式庫。舉個例子，Python 在統計分析程式庫上一直處於弱勢，R 遙遙領先。最近一些強大的模組，比如 statsmodels，已經開始快速縮小這個差距。

Python 是一個蓬勃發展的資料處理生態系統，具有無與倫比的通用目的，而且每週都在持續改善。社群中有這麼多人處於如此興奮的狀態是可以理解的——這非常令人振奮。

就瀏覽器中的視覺化而言，好消息是 JavaScript 除了在網路生態系統中的特權性、否決性、排他性的位置之外還有更多優勢。多虧了一場直譯器軍備競賽，效能有了驚人的突破，還出現一些強大的視覺化程式庫，比如 D3，它可以補充現有的任何語言，JavaScript 現在有了很大的進步。

簡而言之，Python 和 JavaScript 是 Web 資料視覺化的絕佳互補，兩者都需要對方提供重要的漏失元件。

您會學到什麼

我們的 dataviz 工具鏈中有一些大型 Python 和 JavaScript 程式庫，要全面介紹它們需要大量書籍才夠。儘管如此，我認為大多數程式庫的基礎知識（當然包括這裡的介紹內容）都可以很快地掌握。專業知識需要時間和練習，但可以說，提高生產力所需的基本知識是唾手可得的。

從這個意義上講，本書旨在為您提供實用知識的堅實支柱，足以承載未來開發的重擔。我的目標是讓學習曲線盡可能平緩，讓您渡過最初的攀登期，並開始掌握完善您的藝術所需的實用技能。

本書強調實用主義和最佳實務，涵蓋相當大的範圍，所以沒有足夠空間來繞著理論打轉。我介紹了工具鏈中程式庫中最常用的那些層面，並為您指出解決其他問題所需的資源。大多數程式庫都有函數、方法、類別等硬核心，它們是主要的功能子集。有了這些工具，您真的就可以做出一些事來。最後，如果有無法解決的問題，這時候這些好書、說明文件和線上論壇，都會是您的朋友。

程式庫的選擇

在選擇書中使用的程式庫時，我考慮了三件事：

- 像免費啤酒（*https://oreil.ly/WwriM*）一樣的開源（open source）——不需要投入任何額外金錢來學習這本書。
- 長壽——通常是成熟的、社群驅動的和流行的。
- 同類最佳（假設有良好的支援和活躍的社群），處於受歡迎程度和效用之間的最佳點。

您在這裡學到的技能，都應該禁得起時間的考驗。一般來說，最好的選擇很明顯，就是能夠自己當家作主的程式庫；但時機合適時，我也會強調替代選擇，並且告訴您我這樣做的理由。

先備知識

在透過工具鏈開始諾貝爾獎資料集變革之旅之前，需要先閱讀一些預備章節，涵蓋使其餘工具鏈章節更流暢地執行所必需的基本技能。前幾章包括以下內容：

第 2 章，〈Python 和 JavaScript 之間的語言學習橋梁〉

在 Python 和 JavaScript 之間搭建語言橋梁

第 3 章，〈使用 Python 讀寫資料〉

如何使用 Python 透過各種檔案格式和資料庫來傳遞資料

第 4 章，〈Webdev 101〉

涵蓋本書所需的基本 Web 開發

這些章節部分是教程、部分是參考，您可以直接跳到工具鏈的開頭，在需要的地方再回頭來看。

Dataviz 工具鏈

本書的主要部分展示了資料視覺化工具鏈，跟隨著諾貝爾獎得主的資料集，從原始的、新鮮爬取的資料，到引人入勝的互動式 JavaScript 視覺化的旅程。在蒐集過程中，示範一些大程式庫所進行的精煉和改造，總結在圖 I-2 中。這些程式庫是我們工具鏈的工業車床：豐富、成熟的工具，展示了 Python+JavaScript dataviz 堆疊的強大功能。以下部分地簡要介紹了工具鏈的五個階段及其主要程式庫。

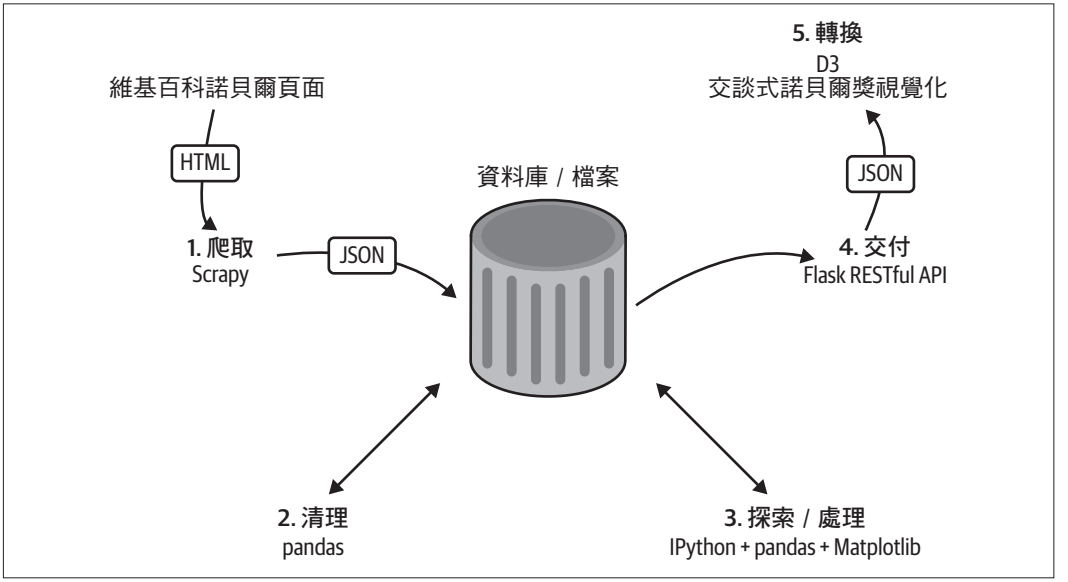

圖 I-2　資料視覺化工具鏈

1. 用 Scrapy 爬取資料

任何資料視覺化人員面臨的第一個挑戰是掌握他們需要的資料，無論是受到請求啟發，還是為了滿足個人需求。如果您很幸運，資料將以原始形式交付給您，但通常您必須去尋找它。我將介紹使用 Python 從 Web 獲取資料的各種方式，例如 Web API 或 Google 試算表。用於工具鏈示範的諾貝爾獎資料集是使用 Scrapy[2] 從維基百科網頁中爬取的。

2　網頁爬取（*https://oreil.ly/g3LPa*）是一種從網站淬取資訊的電腦軟體技術，通常涉及獲取和解析網頁。

Python 的 Scrapy 是一種具有工業強度的爬取工具，可以執行所有資料節流（data throttling）和媒體管線化（media pipelining）操作，當您計畫爬取大量資料時，這些都是必不可少的。爬取通常是獲取您感興趣資料的唯一途徑，一旦您掌握了 Scrapy 的工作流程，所有那些以前禁止存取的資料集都只在一個爬蟲程式網之外[3]。

2. 用 pandas 清洗資料

dataviz 的骯髒祕密是幾乎所有的資料都是骯髒的，把它變成您可以使用的東西可能會比預期花費更多的時間。這是一個平淡無奇的過程，很容易占用一半以上的時間，因此更有理由精通它並使用正確的工具。

pandas 是 Python 資料處理生態系統中的重要參與者。它是一個 Python 資料分析程式庫，其主要元件是 DataFrame，本質上是一個程式化的試算表。pandas 把 Python 強大的數值程式庫 NumPy 擴展到異質（heterogeneous）資料集領域，也就是資料視覺化工具必須處理的類別型（categorical）、時序型（temporal）、和順序型（ordinal）資訊。

除了非常適合交談式地探索資料（使用其內建的 Matplotlib 圖表）之外，pandas 還非常適合清理資料這種繁重工作，可以輕鬆定位重複記錄、修復不可靠的日期字串、查找丟失的資料欄位等等。

3. 使用 pandas 和 Matplotlib 探索資料

在開始把資料轉換為視覺化之前，您需要瞭解它。隱藏在資料中的樣式、趨勢和異常將為您試圖用它所講述的故事提供資訊，無論是要解釋最近小部件銷量為何逐年上升，還是要展示全球氣候的變化。

結合了極致的 Python 直譯器 *IPython*、pandas、以及 Matplotlib（添加了 seaborn），提供了一種理想方式來互動式地探索您的資料、從命令行產生豐富的內聯圖、對資料進行切片（slicing）和切塊（dicing）以揭示有趣的樣式。然後可以輕鬆地把這些探索的結果儲存到檔案或資料庫中，以傳遞給您的 JavaScript 視覺化。

3 Scrapy 的控制器稱為蜘蛛（spider）。

4. 用 Flask 交付您的資料

探索和精煉資料後，您需要把它提送給 Web 瀏覽器，以在其中讓像 D3 這樣的 JavaScript 程式庫可以對其進行轉換。使用像 Python 這樣通用語言的一大優勢是，它可以輕鬆地用幾行程式碼啟動 Web 伺服器，就像它使用 NumPy 和 SciPy[4] 等專用程式庫處理大型資料集一樣輕鬆。Flask 是 Python 最流行的輕量級伺服器，非常適合建立小型的 RESTful[5] API，JavaScript 可以使用這些 API 來從伺服器、檔案或資料庫中獲取資料到瀏覽器。正如我將示範的那樣，您可以在幾行程式碼中推出 RESTful API，並能夠從 SQL 或 NoSQL 資料庫傳輸資料。

5. 使用 Plotly 和 D3 將資料轉換為互動式視覺化

一旦資料清理和精煉之後，進入了視覺化階段，在其中顯示了資料集的選定反射，也許也會讓使用者以互動方式探索它們。根據資料的型態，這可能涉及傳統圖表、地圖或新穎的視覺化。

Plotly 是一個出色的圖表化程式庫，可讓您使用 Python 來發展圖表並把它傳輸到 Web。正如在第 14 章中將會提到的，它還有一個模仿 Python API 的 JavaScript API，免費為您提供原生 JS 圖表化程式庫。

D3 是 JavaScript 強大的視覺化程式庫，可以說是最強大的視覺化工具之一，並且它與語言無關。我們將使用 D3 來建立具有多個元素和使用者互動的新穎諾貝爾獎視覺化，使人們能夠探索資料集中感興趣的項目。D3 學習起來可能具有挑戰性，但我會帶您快速上手，讓您準備好開始磨練自己的技能。

較小的程式庫

除了涵蓋的大型程式庫外，還有大量作為配角的小型程式庫。這些是不可或缺的小工具，它們是工具鏈中的錘子和扳手。尤其是 Python，它擁有一個極其豐富的生態系統，幾乎所有可以想像到的工作都有小型的、專門的程式庫。在強大的配角陣容中，特別值得一提的是：

4 科學 Python 程式庫，NumPy 生態系統的一部分。

5 REST 是 Representational State Transfer 的縮寫，它是基於 HTTP 的 Web API 的主要風格，非常受推薦。

Requests

Python 的首選 HTTP 程式庫，完全符合其座右銘「給人類的 HTTP」。Requests 遠遠優於 urllib2，後者是 Python 的內建程式庫之一。

SQLAlchemy

最好的 Python SQL 工具包和物件關係映射器（object-relational mapper, ORM）。它功能豐富，讓使用各種基於 SQL 的資料庫變得輕而易舉。

seaborn

對 Python 強大的繪圖程式庫 Matplotlib 來說，它是一個很好的補充，添加了一些非常有用的繪圖類型，包括一些特別用於資料視覺化工具的統計類型。它還增加了卓越的美學，覆寫了 Matplotlib 預設值。

Crossfilter

儘管 JavaScript 的資料處理程式庫仍在開發中，但最近出現了一些非常有用的程式庫，其中 Crossfilter 就是其中的佼佼者。它可以非常快速地過濾列 - 行型式資料集，並且非常適合資料視覺化工作。這並不足為奇，因為它的建立者之一就是 D3 之父 Mike Bostock。

marshmallow

一個出色且非常方便的程式庫，可將物件等複雜資料型別與原生 Python 資料型別相互轉換。

如何使用本書

儘管本書的不同部分都遵循資料轉換的過程，但不需要從頭到尾全部讀完一遍本書。第一部分為基於 Python 和 JavaScript 的 web dataviz 提供了一個基本工具包，不可避免地會有許多讀者熟悉的內容。請挑選對您來說陌生的內容，有需要時再回頭翻看，或在後文找到相關性。對於那些精通這兩種語言的人來說，Python 和 JavaScript 之間的語言學習橋梁將是不必要的，儘管可能仍然有一些有用的至理名言在其中。

本書其餘部分會遵循我們的工具鏈，基本上是獨立的，它會把一個相當平淡的 Web 列表轉換為一個完全成熟的互動式 D3 視覺化。如果您想立即進入第 III 部分並使用 pandas 進行一些資料清理和探索，請繼續；但請注意，它假設存在一個骯髒的諾貝爾獎資料集。如果符合您的日程安排，您可以稍後再查看 Scrapy 是如何產生的。同樣的，如果您

想直接在第 IV 部分和第 V 部分中建立 Nobel-viz 應用程式，請注意，它們假定有一個乾淨的諾貝爾獎資料集。

無論採取何種方式，如果您打算把資料視覺化作為您的職業，我建議您最終要以掌握本書中涵蓋的所有基本技能為目標。

寫作背景概述

這是一本實用的書，假定讀者非常瞭解自己想要的視覺化內容，以及視覺化外觀和感覺，以及不受太多理論束縛的願望。儘管如此，借鑒資料視覺化的歷史既可以闡明本書的中心主題，又可以增加有價值的背景。它還可以幫助解釋為什麼現在是進入該領域的這麼激動人心的時刻，因為技術創新正在推動新的資料視覺化形式，人們正在努力解決網際網路背後所產生越來越多的多維資料問題。

資料視覺化背後有令人印象深刻的理論體系，我推薦您閱讀一些很棒的書籍（請參閱第 xxxvii 頁的「推薦書籍」來瞭解其中的一些選擇）。理解人類視覺獲取資訊方式的實際好處怎麼強調都不為過。可以很容易地證明，例如，圓餅圖幾乎總是一種呈現比較性資料的糟糕方式，而簡單的直條圖更為可取。透過進行心理測量實驗，我們現在非常瞭解要如何欺騙人類視覺系統，並讓資料中的關係更難掌握。相反的，我們可以證明某些視覺形式接近於放大對比度的最佳效果。文獻至少提供了一些有用的經驗法則，為任何特定的資料敘述提供了良好的候選者。

本質上，好的 dataviz 試圖呈現從世界測量而來（經驗），或是抽象數學探索的產物，例如，Mandelbrot 集合的美麗碎形樣式（*https://oreil.ly/w5BIV*）所蒐集的資料，並以繪製或強調可能存在的任何樣式或趨勢方式來呈現。這些樣式可以是簡單的，例如，按國家劃分的平均體重；也可以是複雜統計分析的產物，例如，更高維空間中的資料分群。

在未轉換的狀態下，我們可以將這些資料想像成一團模糊的數字或類別的雲。任何樣式或相關性都是完全模糊的。很容易忘記，但不起眼的試算表（圖 I-3 a）是一種資料視覺化——將資料排序為列 - 行形式是試圖馴服它，使其運算更容易，並突顯差異（例如，精算簿記）。當然，大多數人並不擅長在一排數字中發現樣式，因此開發了更易於理解的視覺形式，與我們的視覺皮層互動，視覺皮層是人類獲取世界資訊的主要管道。直條圖、圓餅圖[6]和折線圖進來了，更富有想像力的方法用來以更易於存取的形式精煉統計資料，最著名的方法之一是 Charles Joseph Minard 對拿破崙 1812 年災難性俄國戰役的視覺化（圖 I-3 b）。

6 William Playfair 的 1801 年的 *Statistical Breviary* 具有起源圓餅圖的可疑特徵。

圖 I-3 b 中較淺的棕褐色水流線表示拿破崙軍隊前進莫斯科；黑線表示撤退。寬度代表軍隊規模，隨著傷亡人數的增加而變窄。下面的溫度圖表用於指示沿途位置的溫度。請注意 Minard 結合多維資料，傷亡統計、地理位置、和溫度以優雅方式呈現出大屠殺印象，這很難以任何其他方式來理解（想像一下試圖從傷亡圖表跳轉到地點列表並進行必要的聯繫）。我認為現代互動式資料視覺化的主要問題與 Minard 所面臨的問題完全相同：如何超越傳統的一維直條圖（對很多事情都非常有用），並開發有效交流跨維度樣式的新方法。

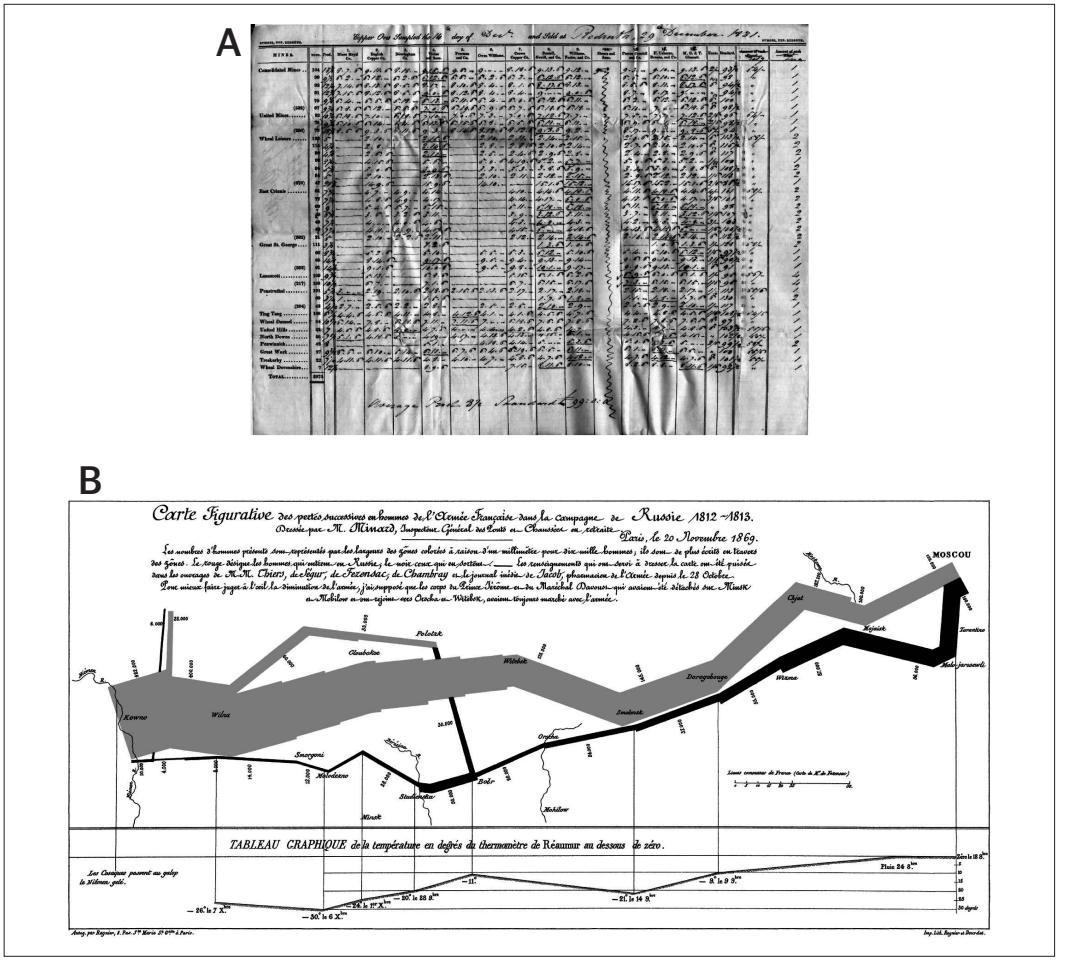

圖 I-3　(a) 早期試算表和 (b) Charles Joseph Minard 對拿破崙 1812 年俄國戰役的視覺化

直到最近，我們對圖表的大部分體驗和 Charles Joseph Minard 的聽眾體驗並沒有太大不同。它們是預渲染的和惰性的，顯示了資料的一種反射，並且希望那是重要且有見地的反射，但仍然在作者的完全控制之下。從這個意義上說，用電腦螢幕像素來代替真實的墨點只是分布尺度的變化罷了。

網際網路的飛躍只是用像素取代了新聞紙張，視覺化仍然是不可點擊的和靜態的。最近，一些強大視覺化程式庫（D3 是其中的佼佼者）的組合，和 JavaScript 效能的巨大改進開闢了一種新型視覺化的方式，它是一種易於存取和動態的、且實際上鼓勵探索和發現。資料探索和呈現之間的明顯區別被模糊化了。這種新型的資料視覺化是本書的重點，也是用於 Web 的 dataviz 能在此刻成為如此令人興奮領域的原因。人們正在嘗試建立新的方法來視覺化資料，並讓使用者更容易存取 / 使用它。這無異於一場革命。

總結

Web 上的 Dataviz 現在很令人興奮，互動式視覺化方面的創新不斷湧現，而且其中許多（如果不是大多數）都是用 D3 開發的。JavaScript 是唯一基於瀏覽器的語言，因此很酷的視覺效果必然要用它來編寫程式（或轉換成它）。但是 JavaScript 缺乏必要的工具或環境，來實現現代資料視覺化中不那麼引人注目但同樣重要的元素：資料的聚合、管理和處理。這就是 Python 的統治優勢，它提供了一種通用的、簡潔的、可讀性極強的程式設計語言，並且可以存取越來越穩定的一流資料處理工具。其中許多工具利用非常快速的低階程式庫強大功能，使 Python 資料處理既快速又簡單。

本書介紹了其中一些重量級工具，以及許多其他較小但同樣重要的工具。它還展示了 Python 和 JavaScript 如何共同代表最好的資料視覺化堆疊，適用於任何希望把他們的視覺化送到網際網路的人。

接下來是本書的第一部分，涵蓋了工具鏈所需的初步技能。您可以現在完成它，或跳到第二部分和工具鏈的開頭，需要時再回來參考。

推薦書籍

這裡有幾本關鍵的資料視覺化書籍可以滿足您的胃口，涵蓋了從互動式儀表板到精美而富有洞察力的資訊圖表所有領域。

- Bertin, Jacques. *Semiology of Graphics: Diagrams, Networks, Maps*. Esri Press, 2010.
- Cairo, Alberto. *The Functional Art*. New Riders, 2012.