前言

複雜性科學是涵蓋數學、電腦科學、自然科學的綜合性領域，它專注於有許多互動元件的**複雜系統**。

複雜性科學的核心工具之一是離散模型，包括網路與圖、細胞自動機、代理人基模型。這些工具在自然與社會科學中很有用，有時候還涵蓋人文藝術。

複雜性科學的概要見 *https://thinkcomplex.com/complex*。

為什麼要學習複雜性科學？以下是幾個原因：

- 複雜性科學很實用，特別是用於解釋自然與社會系統的行為。從牛頓起，物理數學就專注於少量元件與簡單互動的系統。這些模型在某些應用中非常有效，例如天體力學，而對經濟等則比較無效。複雜性科學提供另一種可接受的模型工具。
- 許多複雜性科學的結果出乎意料；本書反覆出現的主題是簡單模型可產生複雜行為，有時我們可以用簡單模型解釋真實世界中的複雜行為。
- 如第 1 章所述，複雜性科學是科學實踐緩慢轉變的核心，也是我們認為的科學的變化核心。
- 複雜性科學的研究提供學習多種物理與社會系統、發展與應用程式設計技巧、思考科學哲學基本問題的機會。

閱讀本書並進行練習讓你有機會探索你未曾遇過的主題與想法、練習 Python 程式設計、學習更多的資料結構與演算法。

本書內容包括：

技術性細節

大部分複雜性科學讀物是為了大眾寫作。它們省略了技術性細節，這對有能力處理它們的人會很失望。這本書包含認識模型如何運作所需的程式碼、數學、說明。

深度讀物

本書列出深度讀物，包括引用論文（大部分有電子檔）、相關維基主題、其他資源。

Jupyter notebook

每一章都有包含程式碼、額外範例、與動畫的 Jupyter notebook，以供你觀察模型的實際運作。

連線與解答

每一章最後附練習與解答。

書中大部分連結為 URL 重新導向。這種機制雖然會隱藏目的地，但能縮短 URL。更重要的是它能讓我更新。若發現連結失效請讓我知道以便更新。

本書讀者

範例以 Python 撰寫。你應該要熟悉 Python 的物件導向，特別是使用與定義類別。

若不熟悉 Python，可從《*Think Python* | 學習程式設計的思考概念》開始，它適合沒有寫過程式的人。若你已經熟悉其他程式設計語言，有很多 Python 讀物與線上資源可參考。

本書使用 NumPy、SciPy、NetworkX，出現時我會加以說明。

我假設讀者懂一點數學：有些地方會用到對數與向量，就這麼多。

第一版後的修改

第二版加上演化與合作演化二章。

在第一版中，每章都介紹了一個主題的背景，並提出了讀者可以進行的實驗。我在第二版做了那些實驗。每一章都以實際範例的形式介紹了實作和結果，也為讀者提供了額外的實驗。

第二版將部分程式碼改為 NumPy 與 NetworkX 標準函式庫。這麼做使程式更清晰也更有效率，並讓讀者能學習到這些函式庫。

Jupyter notebook 也是新加入的。每一章有兩個 notebook：一個是章節的程式、說明、連線；另一個是練習的解答。

最後，所有相關軟體都改為使用 Python 3（但大部分都能使用 Python 2）。

使用程式碼

本書使用的程式碼可從 GitHub 下載：*https://thinkcomplex.com/repo*。Git 是個版本控制系統，一組專案檔案稱為“程式庫”。GitHub 提供 Git 的網頁介面存取服務。

我的 GitHub 首頁提供多種存取程式碼的方式：

- 從右上角的 Fork 按鈕複製程式庫。你必須有個 GitHub 帳號。複製後，在 GitHub 上你會有自己的程式庫可供追蹤你在閱讀本書時所寫的程式碼，然後你可以複製程式庫到你自己的電腦上。
- 下載但不複製程式庫；也就是在你的電腦上下載一份拷貝。這就無需 GitHub 帳號，但你無法將你的修改寫回 GitHub。
- 若完全不想使用 Git，你可以按下“Clone or download”按鈕下載 ZIP 檔案。

我使用 Continuum Analytics 的 Anaconda，它是自由的 Python 版本，包含執行程式所需的所有套件（與其他東西）。我覺得 Anaconda 很容易安裝，預設上為使用者層級而非系統層級安裝，因此無需管理員權限。它支援 Python 2 與 Python 3。你可從 *https://continuum.io/downloads* 下載 Anaconda。

程式庫包含 Python 腳本與 Jupyter notebook。Jupyter 的使用見 *https://jupyter.org*。

Jupyter notebook 有三種使用方式：

在你的電腦上執行 *Jupyter*

若有安裝 Anaconda，你可以從終端機或命令列執行下列命令來安裝 Jupyter：

```
$ conda install jupyter
```

啟動 Jupyter 前，你可以 `cd` 到程式碼目錄：

```
$ cd ThinkComplexity2/code
```

然後啟動 Jupyter 伺服器：

```
$ jupyter notebook
```

啟動伺服器時，它會開啟預設瀏覽器或新的瀏覽器分頁。然後你可以打開並執行 notebook。

在 *Binder* 上執行 *Jupyter*

Binder 是在虛擬機器中執行 Jupyter 的服務。打開 *https://thinkcomplex.com/binder* 會進到帶有本書的 notebook 與相關檔案的 Jupyter 首頁。

你可以執行腳本並修改它們，但該虛擬機器是暫時的，若閒置則虛擬機器與你的修改會消失。

在 *GitHub* 上檢視 *notebook*

你可使用 GitHub 檢視 notebook 與我產生的結果，但不能修改或執行程式。

祝你好運，並玩得開心！

Allen B. Downey
Professor of Computer Science
Olin College of Engineering
Needham, MA

第一章

複雜性科學

複雜性科學相對較新；它從 1980 年代開始被視為一個領域並賦予名稱。但這不是因為它對新主題應用科學工具，而是因為它使用不同類型的工具、進行不同類型的工作、最終改變了所謂的“科學”的意義。

我會以一個經典科學的例子來展示其中的不同處：假設有人問你為何行星的軌道是橢圓形的。你可能會引用牛頓的萬有引力法則寫出描述行星運動的微分方程式，然後解此微分方程式並顯示答案為一個橢圓形。證明完畢！

大部分人滿意這種解釋。它包含數學推導（所以有嚴謹的證明），且它以一般引力法則解釋了橢圓軌道的觀察。

讓我們與不同類型的解釋比較。假設你搬到底特律這種種族隔離的城市，你想要知道為什麼會這樣。如果做一些研究，你會發現 Thomas Schelling 在一篇稱為“隔離動力模型”的論文中提出的一個種族隔離的簡單模型：

以下是摘自第 9 章我對此模型的描述：

> Schelling 模型是個 cell 陣列，每個 cell 代表一間房子。房子被兩種“agent”佔據，標示為紅與藍，數量大致相等。約有 10% 的房子是空的。
>
> 任何時間，一個 agent 可能高興或不高興，視其鄰居 agent 而定。有一個模型中的 agent 在至少兩個鄰居相同時會高興，若只有一個或零個則會不高興。
>
> 此模擬過程隨機選出 agent 並檢查它是否高興。若高興則沒事；若不高興則該 agent 會隨機選擇未佔據的 cell 並搬過去。

若從完全沒有隔離的城市開始執行此模型一段時間，就會出現相同 agent 的群聚。隨著時間，群聚會成長直到變成少量大群聚，且大部分的 agent 居住在同質社區中。

此模型中的隔離程度很驚人且能作為真實隔離的一種解釋。或許底特律的隔離是因為人們傾向不要居於少數，且鄰居讓他們不高興時願意搬家。

這種解釋能像行星運動的解釋一樣讓人滿意嗎？許多人不同意，但為何？

很明顯的，Schelling 模型非常抽象，並不真實。因此你可能會想說人比行星更複雜。但這不正確，畢竟有些行星上面有人，因此它們比人更複雜。

兩個系統都複雜，兩個模型都是基於簡化。舉例來說，我們在行星運動模型中引入行星與其太陽間的力並忽略行星間的互動。在 Schelling 的模型中，我們根據區域資訊引入個別決定並忽略人類行為。

但兩者的程度不同。對行星運動，我們能以被忽略的力較引入的力小來支持該模型，也可以擴充模型以引入其他的互動並顯示出該效應很小。Schelling 的模型則比較難評估這個簡化。

另一個差別是，Schelling 的模型並未引用任何物理法則，它只是簡單的計算而非數學推導。Schelling 這一類的模型看起來不像經典科學，許多人覺得比較沒說服力，至少乍看之下是如此。但我會展示出這些模型有用，包括預測、解釋、設計。本書的目標之一正是說明如何進行。

改變中的科學標準

複雜性科學不只是另一組模型；它還是逐漸改變中的標準，也就是可接受的模型種類。

舉例來說，經典模型傾向基於法則、以公式形式表示、以數學推導解決。複雜性的模型通常基於規則、以計算表示、使用模擬而非分析。

並不是每個人都接受這些模型。舉例來說，Steven Strogatz 在 *Sync* 發表某種螢火蟲自發性同步的模型。他展示此現象的模擬，但寫到：

> 我多次重複此模擬，使用隨機初始條件與其他振盪數值。每次都同步。[...] 接下來的挑戰是證明。只有一個堅實的證據可以證明，在沒有計算機的情況下，同步是不可避免的；最好的證據可以澄清**為什麼**它是不可避免的。

Strogatz 是個數學家，因此渴望證明是可以理解的，但他的證明對我來說並未解決此現象中最有趣的部分。為證明“同步是不可避免的”，Strogatz 做了幾個簡化的假設，特別是每個螢火蟲可以看到其他螢火蟲。

在我看來，有意思的是**看不到其他螢火蟲時如何同步**。第 9 章的主題就是區域互動行為如何產生這種全域行為，這種現象的解釋通常使用代理人基模型（以數學分析很難或做不到的方式）來探索容許或防止同步化的條件。

我是個電腦科學家，因此我熱衷計算式模型或許不意外。我並不是說 Strogatz 是錯的，而是對問什麼問題與使用什麼工具回答有不同看法。這些意見基於價值判斷，因此不需要同意。

無論如何，科學家之間對什麼是好的科學模型，什麼是邊緣科學、偽科學、甚至是非科學有大致的共識。

本書的一個中心論點是這一共識基於隨時間變化的標準，而複雜性科學的出現反映了這些標準的逐步轉變。

科學模型的軸

我將經典模型描述為基於法則、以公式形式表示、以數學推導解決。複雜性的模型通常基於規則、以計算實作。

我們可以將此趨勢視為隨著時間發生轉變的兩個軸：

基於公式 → 基於模擬

分析 → 計算

複雜性科學在很多方面均不同。我將它們列出以讓你知道會發生什麼，但在看過後面的例子前，你可能會覺得有些部分不合理。

連續 → 離散

經典模型傾向基於連續數學，例如微積分；複雜系統的模型通常基於離散數學，包括圖與細胞自動機。

線性 → 非線性

經典模型通常是線性的，或使用線性趨近非線性系統；複雜性科學對非線性模型更友善。

決定性 → 隨機

經典模型通常是決定性的，它反映底層的決定論，第 5 章會討論；複雜性模型通常包括隨機性。

抽象 → 細節

經典模型中的行星是質量點、平面沒有摩擦力、牛是個球體（見 *https://thinkcomplex.com/cow*）。這種簡化通常對分析是必要的，但計算模型更真實。

一兩個 → 多個

經典模型通常限制為少量元件。舉例來說，天體機制中的二體問題可用分析解決；三體問題不行。複雜性科學通常操作大量元件與大量互動。

同質 → 異質

經典模型的元件與互動傾向相同；複雜模型較常引入異質。

這是概觀，無需對它太認真。我也無意詆毀經典科學。複雜模型不一定比較好；事實上，通常比較糟。

我也不是要說這些轉變很突然或完整。相反的，可接受的界線正在漸漸改變。有些過去視為可疑的工具現在很常見，而過去有些廣泛接受的模型現在有審視的必要。

舉例來說，Appel 與 Haken 於 1976 年證明四色定理，他們用電腦列舉了 1936 個特殊情況，這些特殊情況在某種意義上是他們證據的引理。當時很多數學家並不認為他們的定理真正的證明了。現在電腦輔助證明很常見且普遍（都不是全部）接受。

相反的，有很多經濟分析基於一種稱為“經濟人”（又或稱為 *Homo economicus*）的人類行為模型。幾十年來，基於該模型的研究受到高度重視，特別是如果它涉及精湛數學技術。最近，該模型受到質疑，而包含不完全資訊與有限理性的模型是熱門話題。

不同模型有不同目的

複雜模型通常適用於不同目的與解釋：

預測 → 解釋

Shelling 的隔離模型可能說明了複雜社會現象，但對預測沒什麼幫助。另一方面，星體的簡單模型可精準預測日蝕。

現實主義 → 工具主義

經典模型有助於現實主義的解釋；舉例來說，大多數人都認為電子是存在的真實物質。工具主義是這樣一種觀點，即使它們假定的實體不存在，模型也會很有用。George Box 寫了可能是工具主義的座右銘：“所有模型都是錯誤的，但有些模型是有用的”。

還原論 → 整體論

還原論認為系統行為可透過認識其元件來解釋。舉例來說，元素週期表是還原論的勝利，因為它以原子電子模型解釋了元素的化學行為。整體論認為系統層級的某些現象不存在於元件層級，且不能以元件層級條件來解釋。

我們會在第 4 章討論模型、第 6 章討論工具主義、第 8 章討論整體論。

複雜性工程

前面討論過科學背景下的複雜系統，但複雜性也導致工程與社會系統的設計產生影響與變化：

集中 → 分散

集中系統在概念上比較簡單且容易分析，但分散系統更健全。舉例來說，網際網路的用戶端發送請求給集中化的伺服器；若伺服器停機則停止服務。在點對點網路中，每個節點同時是用戶端與伺服器。要停掉服務，你必須停掉每個節點。

一對多 → 多對多

許多通訊系統讓使用者互相溝通並建立、分享、修改內容，以增強甚至取代廣播服務。

上至下 → 下至上

在社會、政治、經濟系統中，許多通常由中央組織的活動現在以草根運動推行。甚至是標準階層結構的軍隊也朝向指揮與控制。

分析 → 計算

在經典工程中，可行設計受限於分析能力。舉例來說，埃菲爾鐵塔能夠建成是因為 Gustave Eiffel 開發出分析技術，特別是處理風壓。現在電腦輔助設計與分析工具能讓你做出幾乎任何你能想到的事情。Frank Gehry 的畢爾包古根漢美術館是我最喜歡的例子。

獨立 → 互動

在經典工程中，大型系統的複雜性由獨立與最小互動管理。這還是很重要的工程原則；無論如何，計算的能力使得設計具有組件之間複雜交互的系統變得越來越可行。

設計 → 搜尋

工程有時候被描述為在可能的設計中搜尋解決方案。搜尋程序自動化的程度越來越高。舉例來說，遺傳演算法探索巨大設計空間，並發現工程師想不到（或不喜歡）的解決方案。演化這個終極遺傳演算法產生出違反人類工程規則的設計。

複雜性思維

我們現在越講越遠，但我所說的科學模型標準的變化，與 20 世紀邏輯學和認識論的發展有關。

亞里士多德邏輯 → 多值邏輯

在傳統邏輯中，任何命題只能是真或偽。這種系統適用數學證明，但在許多真實世界應用中（嚴重）失效。替代方案包括多值邏輯、模糊邏輯、與其他用以處理非決定性、模糊、不確定的系統。Bart Kosko 在他的《*Fuzzy Thinking*》一書中討論這些系統。

頻率論 → 貝葉斯主義

貝氏機率已經存在了幾個世紀，但直到最近才因廉價的計算能力與機率厭惡主觀而被廣泛採用。Sharon Bertsch McGrayne 在《*The Theory That Would Not Die*》一書中展現這一段歷史。

客觀 → 主觀

啟蒙運動和哲學現代主義的基礎是對客觀真理的信仰，即與持有它們的人無關的真理。20 世紀的發展，包括量子力學、哥德爾的不完備理論、以及庫恩對科學史的研究，都引起了對“硬科學”和數學中看似不可避免的主觀性的關注。Rebecca Goldstein 在《*Incompleteness*》一書中介紹了其中的歷史背景。

物理法則 → 理論 → 模型

有些人會區分法則、理論、模型。“法則”意味客觀正確且不變；“理論”需要修正；“模型”是根據簡化與趨近的主觀選擇。

我認為它們是同一件事。有些稱為法則的概念其實是定義；有些實際上是特別適合預測或解釋特定系統的行為的斷言。我們會在第 51 頁“解釋性模型”、第 64 頁“這是什麼模型”、第 112 頁“還原論與整體論”等節回頭討論自然物理法則。

決定論 → 非決定論

決定論認為所有事件不可避免的都因之前的事件所導致。非決定論的形式包括隨機、機率因果、本質不確定。我們會在第 60 頁“決定論”與第 139 頁“湧現與自由意志”等節回頭討論這個主題。

這些趨勢並非普遍或全然，但重點是沿著這些軸變化。對 Thomas Kuhn 的《*The Structure of Scientific Revolutions*》的反應就是證據，這本書在出版時受到攻擊，但現在幾乎無爭議。

這些趨勢同時是複雜性科學的成因與效應。舉例來說，現在更能接受高度抽象化的模型，是因為期望每一個系統都有一個獨特、正確的模型。相反的，複雜系統的發展挑戰決定論與相關的物理法則概念。

這一章是本書的概要，但在看過範例前不一定都會覺得合理。讀完本書後，你可能會覺得回頭再讀這一章很有幫助。